carbon_model.rb

sabshere 8/15/10 should these just be required in the emitter gem's lib/emitter.rb?

```
module BrighterPlanet
  module Residence
    module CarbonModel
      def self.included(base)
        base.decide :emission, :with => :characteristics do
          committee :emission do
            quorum 'from fuel and electricity use and occupation and residents', :needs => [:fuel_oil_consumed, :natural_gas_consumed,
:dirty_electricity_generated, :propane_consumed, :biomass_consumed, :kerosene_consumed, :coal_consumed, :residents,
:electricity_emission_factor, :floorspace_estimate, :air_conditioner_use, :active_subtimeframe, :occupation] do |characteristics,
timeframe|
              ( characteristics[:fuel_oil_consumed]           * ResidenceFuelType.find_by_name('fuel oil').emission_factor    +
                characteristics[:natural_gas_consumed]        * ResidenceFuelType.find_by_name('natural gas').emission_factor +
                characteristics[:propane_consumed]            * ResidenceFuelType.find_by_name('propane').emission_factor     +
                characteristics[:biomass_consumed]            * ResidenceFuelType.find_by_name('biomass').emission_factor     +
                characteristics[:kerosene_consumed]           * ResidenceFuelType.find_by_name('kerosene').emission_factor    +
                characteristics[:coal_consumed]               * ResidenceFuelType.find_by_name('coal').emission_factor        +
                characteristics[:dirty_electricity_generated] * characteristics[:electricity_emission_factor]                 +
                characteristics[:floorspace_estimate] * characteristics[:air_conditioner_use].fugitive_emission * (timeframe /
timeframe.year) * characteristics[:occupation]
              ) *
              (characteristics[:active_subtimeframe] / timeframe) / characteristics[:residents]
            end

            quorum 'default' do
              raise "Residence's default emission quorum should never be called"
            end
          end

          committee :fuel_oil_consumed do # returns litres
            quorum 'from reports', :needs => :reported_annual_fuel_oil_consumption do |characteristics, timeframe|
              characteristics[:reported_annual_fuel_oil_consumption] * (timeframe / timeframe.year)
            end
            quorum 'from research', :needs => [:predicted_annual_fuel_oil_consumption, :predicted_fuel_shares, :missing_annual_energy,
:occupation] do |characteristics, timeframe|
              (characteristics[:predicted_annual_fuel_oil_consumption] + (characteristics[:missing_annual_energy] *
characteristics[:predicted_fuel_shares][:fuel_oil]).joules.to(:litres_of_fuel_oil) ) * (timeframe / timeframe.year) *
characteristics[:occupation] / base.fallback.occupation
            end
          end

          committee :natural_gas_consumed do # returns joules
            quorum 'from reports', :needs => :reported_annual_natural_gas_consumption do |characteristics, timeframe|
              characteristics[:reported_annual_natural_gas_consumption] * (timeframe / timeframe.year)
            end
            quorum 'from research', :needs => [:predicted_annual_natural_gas_consumption, :predicted_fuel_shares,
:missing_annual_energy, :occupation] do |characteristics, timeframe|
              (characteristics[:predicted_annual_natural_gas_consumption] + (characteristics[:missing_annual_energy] *
characteristics[:predicted_fuel_shares][:natural_gas])) * (timeframe / timeframe.year) * characteristics[:occupation] /
base.fallback.occupation
            end
          end
```

```
          committee :propane_consumed do # returns litres
            quorum 'from reports', :needs => :reported_annual_propane_consumption do |characteristics, timeframe|
              characteristics[:reported_annual_propane_consumption] * (timeframe / timeframe.year)
            end
            quorum 'from research', :needs => [:predicted_annual_propane_consumption, :predicted_fuel_shares, :missing_annual_energy,
:occupation] do |characteristics, timeframe|
              (characteristics[:predicted_annual_propane_consumption] + (characteristics[:missing_annual_energy] *
characteristics[:predicted_fuel_shares][:propane]).joules.to(:litres_of_propane)) * (timeframe / timeframe.year) *
characteristics[:occupation] / base.fallback.occupation
            end
          end

          committee :biomass_consumed do # returns joules
            quorum 'from reports', :needs => :reported_annual_biomass_consumption do |characteristics, timeframe|
              characteristics[:reported_annual_biomass_consumption] * (timeframe / timeframe.year)
            end
            quorum 'from research', :needs => [:predicted_annual_biomass_consumption, :predicted_fuel_shares, :missing_annual_energy,
:occupation] do |characteristics, timeframe|
              (characteristics[:predicted_annual_biomass_consumption] + (characteristics[:missing_annual_energy] *
characteristics[:predicted_fuel_shares][:biomass])) * (timeframe / timeframe.year) * characteristics[:occupation] /
base.fallback.occupation
            end
          end

          committee :kerosene_consumed do # returns litres
            quorum 'from reports', :needs => :reported_annual_kerosene_consumption do |characteristics, timeframe|
              characteristics[:reported_annual_kerosene_consumption] * (timeframe / timeframe.year)
            end
            quorum 'from research', :needs => [:predicted_annual_kerosene_consumption, :predicted_fuel_shares, :missing_annual_energy,
:occupation] do |characteristics, timeframe|
              (characteristics[:predicted_annual_kerosene_consumption] + (characteristics[:missing_annual_energy] *
characteristics[:predicted_fuel_shares][:kerosene]).joules.to(:litres_of_kerosene)) * (timeframe / timeframe.year) *
characteristics[:occupation] / base.fallback.occupation
            end
          end

          committee :coal_consumed do # returns kg
            quorum 'from reports', :needs => :reported_annual_coal_consumption do |characteristics, timeframe|
              characteristics[:reported_annual_coal_consumption] * (timeframe / timeframe.year)
            end
            quorum 'from research', :needs => [:predicted_annual_coal_consumption, :predicted_fuel_shares, :missing_annual_energy,
:occupation] do |characteristics, timeframe|
              (characteristics[:predicted_annual_coal_consumption] + (characteristics[:missing_annual_energy] *
characteristics[:predicted_fuel_shares][:coal]).joules.to(:kilograms_of_coal)) * (timeframe / timeframe.year) *
characteristics[:occupation] / base.fallback.occupation
            end
          end

          committee :dirty_electricity_generated do
            quorum 'from electricity generated and green electricity', :needs => [:electricity_generated, :green_electricity] do
|characteristics|
              characteristics[:electricity_generated] * (1.0 - characteristics[:green_electricity])
            end
          end

          committee :green_electricity do
            quorum 'default' do
              base.fallback.green_electricity
            end
          end
```

```
          committee :electricity_generated do # returns kWh
            quorum 'from electricity used and loss rate', :needs => [:electricity_used, :electricity_loss_rate] do |characteristics|
              characteristics[:electricity_used] / (1.0 - characteristics[:electricity_loss_rate])
            end
          end

          committee :electricity_used do # returns kWh
            quorum 'from reports', :needs => :reported_annual_electricity_use do |characteristics, timeframe|
              characteristics[:reported_annual_electricity_use] * (timeframe / timeframe.year)
            end

            quorum 'from research', :needs => [:predicted_annual_electricity_use, :predicted_fuel_shares, :missing_annual_energy,
:occupation] do |characteristics, timeframe|
              (characteristics[:predicted_annual_electricity_use] + ((characteristics[:missing_annual_energy] *
characteristics[:predicted_fuel_shares][:electricity]).joules.to(:kilowatt_hours))) * (timeframe / timeframe.year) *
characteristics[:occupation] / base.fallback.occupation
            end
          end

          committee :missing_annual_energy do # returns joules
            quorum 'from fuel reports', :needs => [:predicted_annual_fuel_oil_consumption, :predicted_annual_natural_gas_consumption,
:predicted_annual_propane_consumption, :predicted_annual_kerosene_consumption, :predicted_annual_biomass_consumption,
:predicted_annual_coal_consumption, :predicted_annual_electricity_use], :appreciates => [:reported_annual_fuel_oil_consumption,
:reported_annual_natural_gas_consumption, :reported_annual_propane_consumption, :reported_annual_kerosene_consumption,
:reported_annual_biomass_consumption, :reported_annual_coal_consumption, :reported_annual_electricity_use] do |characteristics|
              energy = 0
              if characteristics[:reported_annual_fuel_oil_consumption] and
characteristics[:reported_annual_fuel_oil_consumption].zero?
                energy += characteristics[:predicted_annual_fuel_oil_consumption].litres_of_fuel_oil.to :joules
              end
              if characteristics[:reported_annual_natural_gas_consumption] and
characteristics[:reported_annual_natural_gas_consumption].zero?
                energy += characteristics[:predicted_annual_natural_gas_consumption]
              end
              if characteristics[:reported_annual_propane_consumption] and characteristics[:reported_annual_propane_consumption].zero?
                energy += characteristics[:predicted_annual_propane_consumption].litres_of_propane.to :joules
              end
              if characteristics[:reported_annual_kerosene_consumption] and
characteristics[:reported_annual_kerosene_consumption].zero?
                energy += characteristics[:predicted_annual_kerosene_consumption].litres_of_kerosene.to :joules
              end
              if characteristics[:reported_annual_biomass_consumption] and characteristics[:reported_annual_biomass_consumption].zero?
                energy += characteristics[:predicted_annual_biomass_consumption]
              end
              if characteristics[:reported_annual_coal_consumption] and characteristics[:reported_annual_coal_consumption].zero?
                energy += characteristics[:predicted_annual_coal_consumption].kilograms_of_coal.to :joules
              end
              if characteristics[:reported_annual_electricity_use] and characteristics[:reported_annual_electricity_use].zero?
                energy += characteristics[:predicted_annual_electricity_use].kilowatt_hours.to :joules
              end
              energy
            end
          end

          committee :electricity_loss_rate do # returns percentage
            quorum 'from egrid region', :needs => :egrid_region do |characteristics|
              characteristics[:egrid_region].loss_factor
            end

            quorum 'default' do
```

```
            EgridRegion.fallback.loss_factor
          end
        end

        committee :electricity_emission_factor do # returns kg CO2 / kWh
          quorum 'from egrid subregion', :needs => :egrid_subregion do |characteristics|
            characteristics[:egrid_subregion].electricity_emission_factor
          end

          quorum 'default' do
            EgridSubregion.fallback.electricity_emission_factor
          end
        end

        committee :egrid_region do
          quorum 'from egrid subregion', :needs => :egrid_subregion do |characteristics|
            characteristics[:egrid_subregion].egrid_region
          end
        end

        committee :egrid_subregion do
          quorum 'from_zip_code', :needs => :zip_code do |characteristics|
            characteristics[:zip_code].egrid_subregion
          end
        end

        committee :occupation do
          quorum 'default' do
            base.fallback.occupation
          end
        end

        committee :residents do
          quorum 'from cohort', :needs => :cohort do |characteristics|
            characteristics[:cohort].weighted_average :residents
          end

          quorum 'default' do
            base.fallback.residents_before_type_cast
          end
        end

        committee :air_conditioner_use do
          quorum 'default' do
            AirConditionerUse.fallback
          end
        end

        committee :predicted_fuel_shares do # returns an array of percentages
          quorum 'from research', :needs => [:predicted_annual_energy_consumption, :predicted_annual_fuel_oil_consumption,
:predicted_annual_natural_gas_consumption, :predicted_annual_propane_consumption, :predicted_annual_kerosene_consumption,
:predicted_annual_biomass_consumption, :predicted_annual_coal_consumption, :predicted_annual_electricity_use] do |characteristics|
            {
              :fuel_oil => characteristics[:predicted_annual_fuel_oil_consumption].litres_of_fuel_oil.to(:joules) /
characteristics[:predicted_annual_energy_consumption],
              :natural_gas => characteristics[:predicted_annual_natural_gas_consumption] /
characteristics[:predicted_annual_energy_consumption],
              :propane => characteristics[:predicted_annual_propane_consumption].litres_of_propane.to(:joules) /
characteristics[:predicted_annual_energy_consumption],
              :kerosene => characteristics[:predicted_annual_kerosene_consumption].litres_of_kerosene.to(:joules) /
characteristics[:predicted_annual_energy_consumption],
```

```
              :biomass => characteristics[:predicted_annual_biomass_consumption] /
characteristics[:predicted_annual_energy_consumption],
              :coal => characteristics[:predicted_annual_coal_consumption].kilograms_of_coal.to(:joules) /
characteristics[:predicted_annual_energy_consumption],
              :electricity => characteristics[:predicted_annual_electricity_use].kilowatt_hours.to(:joules) /
characteristics[:predicted_annual_energy_consumption]
            }
          end
        end

        committee :predicted_annual_energy_consumption do # returns BTUs
          quorum 'from research', :needs => [:predicted_annual_fuel_oil_consumption, :predicted_annual_natural_gas_consumption,
:predicted_annual_propane_consumption, :predicted_annual_kerosene_consumption, :predicted_annual_biomass_consumption,
:predicted_annual_coal_consumption, :predicted_annual_electricity_use] do |characteristics|
            energy = 0
            energy += characteristics[:predicted_annual_fuel_oil_consumption].litres_of_fuel_oil.to :joules
            energy += characteristics[:predicted_annual_natural_gas_consumption]
            energy += characteristics[:predicted_annual_propane_consumption].litres_of_propane.to :joules
            energy += characteristics[:predicted_annual_kerosene_consumption].litres_of_kerosene.to :joules
            energy += characteristics[:predicted_annual_biomass_consumption]
            energy += characteristics[:predicted_annual_coal_consumption].kilograms_of_coal.to :joules
            energy += characteristics[:predicted_annual_electricity_use].kilowatt_hours.to :joules
          end
        end

        committee :reported_annual_fuel_oil_consumption do # returns litres
          quorum 'from volume estimate', :needs => :annual_fuel_oil_volume_estimate do |characteristics|
            characteristics[:annual_fuel_oil_volume_estimate]
          end

          quorum 'from cost', :needs => :annual_fuel_oil_cost, :appreciates => :state do |characteristics, timeframe|
            relaxations = []
            relaxations << { :timeframe => timeframe,           :location => characteristics[:state] } if characteristics[:state]
            relaxations << { :timeframe => timeframe.last_year, :location => characteristics[:state] } if characteristics[:state]
            relaxations << { :timeframe => timeframe,           :location => Country.united_states   }
            relaxations << { :timeframe => timeframe.last_year, :location => Country.united_states   }
            if price_per_unit = ResidenceFuelType[:fuel_oil].price_per_unit(relaxations)
              characteristics[:annual_fuel_oil_cost] / price_per_unit
            else
              nil
            end
          end
        end

        committee :predicted_annual_fuel_oil_consumption do # returns litres
          quorum 'from cohort', :needs => :cohort do |characteristics|
            characteristics[:cohort].weighted_average(%w(heating_space heating_water appliances).map { |purpose|
"annual_energy_from_fuel_oil_for_#{purpose}".to_sym }).to_f.joules.to(:litres_of_fuel_oil)
          end

          quorum 'default' do
            base.fallback.annual_fuel_oil_volume_estimate
          end
        end

        committee :reported_annual_natural_gas_consumption do # returns joules
          quorum 'from volume estimate', :needs => :monthly_natural_gas_volume_estimate do |characteristics|
            characteristics[:monthly_natural_gas_volume_estimate] * 12
          end

          quorum 'from cost', :needs => :monthly_natural_gas_cost, :appreciates => :state do |characteristics, timeframe|
```

```
            relaxations = []
            relaxations << { :timeframe => timeframe,           :location => characteristics[:state] } if characteristics[:state]
            relaxations << { :timeframe => timeframe.last_year, :location => characteristics[:state] } if characteristics[:state]
            relaxations << { :timeframe => timeframe,           :location => Country.united_states   }
            relaxations << { :timeframe => timeframe.last_year, :location => Country.united_states   }
            if price_per_unit = ResidenceFuelType[:natural_gas].price_per_unit(relaxations) #FIXME joules vs. cubic metres issue
              characteristics[:monthly_natural_gas_cost] * 12 / price_per_unit
            else
              nil
            end
          end
        end

        committee :predicted_annual_natural_gas_consumption do # returns joules
          quorum 'from cohort', :needs => :cohort do |characteristics|
            characteristics[:cohort].weighted_average(%w(heating_space heating_water appliances).map { |purpose|
"annual_energy_from_natural_gas_for_#{purpose}".to_sym }).to_f
          end

          quorum 'default' do
            base.fallback.monthly_natural_gas_volume_estimate * 12
          end
        end

        committee :reported_annual_propane_consumption do # returns litres
          quorum 'from volume estimate', :needs => :annual_propane_volume_estimate do |characteristics|
            characteristics[:annual_propane_volume_estimate]
          end

          quorum 'from cost', :needs => :annual_propane_cost, :appreciates => :state do |characteristics, timeframe|
            relaxations = []
            relaxations << { :timeframe => timeframe,           :location =>
characteristics[:state].petroleum_administration_for_defense_district } if characteristics[:state]
            relaxations << { :timeframe => timeframe.last_year, :location =>
characteristics[:state].petroleum_administration_for_defense_district } if characteristics[:state]
            relaxations << { :timeframe => timeframe,           :location => Country.united_states   }
            relaxations << { :timeframe => timeframe.last_year, :location => Country.united_states   }
            if price_per_unit = ResidenceFuelType[:propane].price_per_unit(relaxations)
              characteristics[:annual_propane_cost] / price_per_unit
            else
              nil
            end
          end
        end

        committee :predicted_annual_propane_consumption do # returns litres
          quorum 'from cohort', :needs => :cohort do |characteristics|
            characteristics[:cohort].weighted_average(%w(heating_space heating_water appliances).map { |purpose|
"annual_energy_from_propane_for_#{purpose}".to_sym }).to_f.joules.to(:litres_of_propane)
          end

          quorum 'default' do
            base.fallback.annual_propane_volume_estimate
          end
        end

        committee :reported_annual_kerosene_consumption do # returns litres
          quorum 'from volume estimate', :needs => :annual_kerosene_volume_estimate do |characteristics|
            characteristics[:annual_kerosene_volume_estimate]
          end
        end
```

```
          committee :predicted_annual_kerosene_consumption do # returns litres
            quorum 'from cohort', :needs => :cohort do |characteristics|
              characteristics[:cohort].weighted_average(:annual_energy_from_kerosene).to_f.joules.to(:litres_of_kerosene)
            end

            quorum 'default' do
              base.fallback.annual_kerosene_volume_estimate
            end
          end

          committee :reported_annual_biomass_consumption do # returns joules
            quorum 'from volume estimate', :needs => :annual_wood_volume_estimate do |characteristics|
              characteristics[:annual_wood_volume_estimate]
            end
          end

          committee :predicted_annual_biomass_consumption do # returns joules
            quorum 'from cohort', :needs => :cohort do |characteristics|
              characteristics[:cohort].weighted_average(:annual_energy_from_wood)
            end

            quorum 'default' do
              base.fallback.annual_wood_volume_estimate
            end
          end

          committee :reported_annual_coal_consumption do # returns kg
            quorum 'from volume estimate', :needs => :annual_coal_volume_estimate do |characteristics|
              characteristics[:annual_coal_volume_estimate]
            end
          end

          committee :predicted_annual_coal_consumption do # returns kg
            quorum 'default' do
              base.fallback.annual_coal_volume_estimate
            end
          end

          committee :reported_annual_electricity_use do # returns kWh
            quorum 'from use estimate', :needs => :monthly_electricity_use_estimate do |characteristics|
              characteristics[:monthly_electricity_use_estimate] * 12
            end

            quorum 'from cost', :needs => :monthly_electricity_cost, :appreciates => :state do |characteristics, timeframe|
              relaxations = []
              relaxations << { :timeframe => timeframe,           :location => characteristics[:state] } if characteristics[:state]
              relaxations << { :timeframe => timeframe.last_year, :location => characteristics[:state] } if characteristics[:state]
              relaxations << { :timeframe => timeframe,           :location => Country.united_states   }
              relaxations << { :timeframe => timeframe.last_year, :location => Country.united_states   }
              if price_per_unit = ResidenceFuelType[:electricity].price_per_unit(relaxations)
                characteristics[:monthly_electricity_cost] * 12 / price_per_unit
              else
                nil
              end
            end
          end

          committee :predicted_annual_electricity_use do # returns kWh
            quorum 'from cohort', :needs => :cohort, :appreciates => [:clothes_machine_use, :refrigerator_count, :freezer_count,
:dishwasher_use, :lighting_efficiency] do |characteristics|
```

```
            cohort = characteristics[:cohort]
            energy = cohort.weighted_average([:annual_energy_from_electricity_for_clothes_driers,
                                              :annual_energy_from_electricity_for_dishwashers,
                                              :annual_energy_from_electricity_for_freezers,
                                              :annual_energy_from_electricity_for_refrigerators,
                                              :annual_energy_from_electricity_for_air_conditioners,
                                              :annual_energy_from_electricity_for_heating_space,
                                              :annual_energy_from_electricity_for_heating_water,
                                              :annual_energy_from_electricity_for_other_appliances]).to_f

            if clothes_machine_use = characteristics[:clothes_machine_use]
              energy -= cohort.weighted_average(:annual_energy_from_electricity_for_clothes_driers)
              clothes_machine_use_cohort =
recs_cohort(characteristics.slice(*([:clothes_machine_use].push(*ResidentialEnergyConsumptionSurveyResponse::INPUT_CHARACTERISTICS))),
ResidentialEnergyConsumptionSurveyResponse::SUBCOHORT_THRESHOLD)
              if clothes_machine_use_cohort.any?
                energy += clothes_machine_use_cohort.weighted_average(:annual_energy_from_electricity_for_clothes_driers).to_f
              else
                energy += characteristics[:clothes_machine_use].annual_energy_from_electricity_for_clothes_driers
              end
            end

            if refrigerator_count = characteristics[:refrigerator_count]
              energy -= cohort.weighted_average(:annual_energy_from_electricity_for_refrigerators)
              if refrigerator_count == 0
                energy += 0
              else
                refrigerator_count_subcohort =
recs_cohort(characteristics.slice(*([:refrigerator_count].push(*ResidentialEnergyConsumptionSurveyResponse::INPUT_CHARACTERISTICS))),
ResidentialEnergyConsumptionSurveyResponse::SUBCOHORT_THRESHOLD)
                if refrigerator_count_subcohort.any?
                  energy += refrigerator_count_subcohort.weighted_average(:annual_energy_from_electricity_for_refrigerators).to_f
                else
                  energy += refrigerator_count * ResidenceAppliance.annual_energy_from_electricity_for(:refrigerators)
                end
              end
            end

            if freezer_count = characteristics[:freezer_count]
              energy -= cohort.weighted_average(:annual_energy_from_electricity_for_freezers)
              if freezer_count == 0
                energy += 0
              else
                freezer_count_subcohort =
recs_cohort(characteristics.slice(*([:freezer_count].push(*ResidentialEnergyConsumptionSurveyResponse::INPUT_CHARACTERISTICS))),
ResidentialEnergyConsumptionSurveyResponse::SUBCOHORT_THRESHOLD)
                if freezer_count_subcohort.any?
                  energy += freezer_count_subcohort.weighted_average(:annual_energy_from_electricity_for_freezers).to_f
                else
                  energy += freezer_count * ResidenceAppliance.annual_energy_from_electricity_for(:freezers)
                end
              end
            end

            if dishwasher_use = characteristics[:dishwasher_use]
              energy -= cohort.weighted_average(:annual_energy_from_electricity_for_dishwashers)
              dishwasher_use_cohort =
recs_cohort(characteristics.slice(*([:dishwasher_use].push(*ResidentialEnergyConsumptionSurveyResponse::INPUT_CHARACTERISTICS))),
ResidentialEnergyConsumptionSurveyResponse::SUBCOHORT_THRESHOLD)
              if dishwasher_use_cohort.any?
                energy += dishwasher_use_cohort.weighted_average(:annual_energy_from_electricity_for_dishwashers).to_f
```

```
                else
                  energy += characteristics[:dishwasher_use].annual_energy_from_electricity_for_dishwashers
                end
              end

              if lighting_efficiency = characteristics[:lighting_efficiency]
                lighting_electricity_use_in_cohort =
                  cohort.weighted_average(:lighting_efficiency) * cohort.weighted_average(:lighting_use) *
research(:efficient_lightbulb_power) +
                  (1 - cohort.weighted_average(:lighting_efficiency)) * cohort.weighted_average(:lighting_use) *
research(:standard_lightbulb_power)
                energy -= lighting_electricity_use_in_cohort.watt_hours.to :joules
                lighting_electricity_use_in_residence =
                  lighting_efficiency * cohort.weighted_average(:lighting_use) * research(:efficient_lightbulb_power) +
                  (1 - lighting_efficiency) * cohort.weighted_average(:lighting_use) * research(:standard_lightbulb_power)
                energy += lighting_electricity_use_in_residence.watt_hours.to :joules
              end

              energy.joules.to(:kilowatt_hours)
            end

            quorum 'default' do
              base.fallback.monthly_electricity_use_estimate * 12
            end
          end

          committee :active_subtimeframe do
            quorum 'from acquisition and retirement', :needs => [:acquisition, :retirement] do |characteristics, timeframe|
              Timeframe.constrained_new characteristics[:acquisition].to_date, characteristics[:retirement].to_date, timeframe
            end
          end

          committee :acquisition do
            quorum 'from construction year', :needs => :construction_year do |characteristics|


              characteristics[:construction_year] - 1.year
            end
            quorum 'from retirement', :appreciates => :retirement do |characteristics, timeframe|
              [ timeframe.from, characteristics[:retirement] ].compact.min
            end
          end

          committee :retirement do
            quorum 'from acquisition', :appreciates => :acquisition do |characteristics, timeframe|
              [ timeframe.to, characteristics[:acquisition] ].compact.max
            end
          end



          committee :floorspace_estimate do
            quorum 'from cohort', :needs => :cohort do |characteristics|
              characteristics[:cohort].weighted_average :floorspace
            end

            quorum 'default' do
              base.fallback.floorspace_estimate
            end
          end

          committee :cohort do
```

FIXME this is an imperfect substitution for a line in https://app1.yerba.brighterplanet.com/changesets/9463

This is kindof “hacky” As implemented, this needs to be above floorspace committee or else cohort will always use the base.fallback

```
            quorum 'from residential energy consumption survey', :appreciates =>
ResidentialEnergyConsumptionSurveyResponse::INPUT_CHARACTERISTICS do |characteristics|
              cohort = recs_cohort characteristics
              if cohort.any?
                cohort
              else
                nil
              end
            end
          end

          committee :bathrooms do
            quorum 'from fractional bathroom counts', :needs => [:full_bathrooms, :half_bathrooms] do |characteristics|
              characteristics[:full_bathrooms] + 0.5 * characteristics[:half_bathrooms]
            end
          end

          committee :census_region do
            quorum 'from census division', :needs => :census_division do |characteristics|
              characteristics[:census_division].census_region
            end
          end

          committee :census_division do
            quorum 'from state', :needs => :state do |characteristics|
              characteristics[:state].census_division
            end
          end

          committee :state do
            quorum 'from climate division', :needs => :climate_division do |characteristics|
              characteristics[:climate_division].state
            end
          end

          committee :floorspace do # returns a Range of floorspaces
            quorum 'from floorspace estimate', :needs => :floorspace_estimate do |characteristics|
              floorspace_estimate = characteristics[:floorspace_estimate]
              (floorspace_estimate - 50)..(floorspace_estimate + 50)
            end
          end

          committee :heating_degree_days do # returns a Range of degree-days
            quorum 'from climate division', :needs => :climate_division do |characteristics|
              days = characteristics[:climate_division].heating_degree_days
              (days - ClimateDivision::RADIUS)..(days + ClimateDivision::RADIUS)
            end
          end

          committee :cooling_degree_days do
            quorum 'from climate division', :needs => :climate_division do |characteristics|
              days = characteristics[:climate_division].cooling_degree_days
              (days - ClimateDivision::RADIUS)..(days + ClimateDivision::RADIUS)
            end
          end

          committee :climate_division do
            quorum 'from zip code', :needs => :zip_code do |characteristics|
              characteristics[:zip_code].climate_division
            end
          end
```

```
        end
      end

      def self.research(key)
        case key
        when :efficient_lightbulb_power
          17.5 # watts https://brighterplanet.sifterapp.com/projects/30/issues/433
        when :standard_lightbulb_power
          67.5 # watts https://brighterplanet.sifterapp.com/projects/30/issues/433
        end
      end

      def self.recs_cohort(characteristics, threshold = ResidentialEnergyConsumptionSurveyResponse.minimum_cohort_size)
        conditions = characteristics.keys.inject({}) do |memo, k|
          case v = characteristics[k].value
          when ActiveRecord::Base
            assoc = ResidentialEnergyConsumptionSurveyResponse.reflect_on_association(k)
            memo[assoc.primary_key_name.to_sym] = v.send(v.class.primary_key)
          else
            memo[k] = v
          end
          memo
        end
        ResidentialEnergyConsumptionSurveyResponse.big_cohort conditions, :minimum_cohort_size => threshold
      end
    end
  end
end
```