carbon_model.rb

```
require 'leap'
require 'timeframe'
require 'date'
require 'matrix'

module BrighterPlanet
  module Purchase
    module CarbonModel
      def self.included(base)
        base.extend ::Leap::Subject

        base.decide :emission, :with => :characteristics do
          committee :emission do
            quorum 'from impacts', :needs => :impacts do
|characteristics|
              characteristics[:impacts].to_a.sum
            end
          end

          committee :impacts do
            quorum 'from economic flows and impact vectors',
:needs => [:economic_flows, :impact_vectors] do |characteristics|
              x = characteristics[:impact_vectors]
              y = characteristics[:economic_flows]
              x = x.respond_to?(:value) ? x.value : x
              y = y.respond_to?(:value) ? y.value : y
              x * y
            end
          end

          committee :impact_vectors do
            quorum 'from database' do
              adapter =
BrighterPlanet::Purchase.impact_vectors_adapter
              adapter.matrix
          end
        end

        committee :economic_flows do
          quorum 'from sector shares, a', :needs =>
[:sector_shares, :sector_direct_requirements] do
|characteristics|
            y = characteristics[:sector_shares]
            leonteif_inverse =
characteristics[:sector_direct_requirements]
            y = y.respond_to?(:value) ? y.value : y
            leonteif_inverse =
leonteif_inverse.respond_to?(:value) ? leonteif_inverse.value :
leonteif_inverse
            leonteif_inverse * y
          end
        end

        committee :sector_direct_requirements do
          quorum 'from database' do
            adapter =
BrighterPlanet::Purchase.sector_direct_requirements_adapter
            adapter.matrix
          end
        end

        committee :sector_shares do
          quorum 'from industry sector shares', :needs =>
:industry_sector_shares do |characteristics|
            shares = BrighterPlanet::Purchase.key_map.map do
|key|
              characteristics[:industry_sector_shares][key] ||
0
            end
            Vector[*shares]
          end
        end

        committee :industry_sector_shares do
          quorum 'from industry sector ratios', :needs =>
[:industry_sector_ratios, :adjusted_cost] do |characteristics|
              characteristics[:industry_sector_ratios].inject({})
do |new_ratios, (io_code, ratio)|
                new_ratios[io_code] ||= 0
                new_ratios[io_code] += ratio *
characteristics[:adjusted_cost]
                new_ratios
              end
            end
          end

          committee :industry_sector_ratios do
            quorum 'from industry ratios', :needs =>
:industry_ratios do |characteristics|
              naics_codes =
characteristics[:industry_ratios].keys
              industry_sectors = IndustrySector.where(:naics_code
=> naics_codes)
              characteristics[:industry_ratios].inject({}) do
|new_ratios, (naics_code, ratio)|
                industry_sectors.
                  find_all { |i| i.naics_code == naics_code }.
                  each do |industry_sector|
                  new_ratios[industry_sector.io_code] ||= 0
                  new_ratio = ratio * industry_sector.ratio
                  new_ratios[industry_sector.io_code] +=
new_ratio
                end
                new_ratios
              end
            end
          end
```

industries = the industries needed to produce the purchased item ratios = the portion of the purchase amount that goes to each industry

```
          committee :industry_ratios do
            quorum 'from non trade industry and industry product
ratios', :needs => [:non_trade_industry_ratios,
:industry_product_ratios] do |characteristics|
              combined_ratios =
characteristics[:non_trade_industry_ratios].
                merge(characteristics[:industry_product_ratios])
do |key, non_trade, ip_ratio|
                  non_trade + ip_ratio
          end
        end
      end

      committee :industry_product_ratios do
        quorum 'from product line industry product ratios',
:needs => :product_line_industry_product_ratios do
|characteristics|
          naics_product_codes =
characteristics[:product_line_industry_product_ratios].keys
          industry_products =
IndustryProduct.where(:naics_product_code => naics_product_codes)

characteristics[:product_line_industry_product_ratios].inject({})
do |new_ratios, (naics_product_code, ratio)|
            industry_products.
              find_all { |i| i.naics_product_code ==
naics_product_code }.
              each do |industry_product|
              new_ratios[industry_product.naics_code] ||= 0
              new_ratios[industry_product.naics_code] +=
ratio
            end
            new_ratios
          end
        end
      end

      committee :product_line_industry_product_ratios do
        quorum 'from product line ratios', :needs =>
:product_line_ratios do |characteristics|
          ps_codes =
characteristics[:product_line_ratios].keys
          plips = ProductLineIndustryProduct.where(:ps_code
=> ps_codes)
          characteristics[:product_line_ratios].inject({}) do
|new_ratios, (ps_code, ratio)|
            plips.find_all { |p| p.ps_code == ps_code }.each
do |plip|
              new_ratios[plip.naics_product_code] ||= 0
              new_ratio = ratio * plip.ratio
```

```
                  new_ratios[plip.naics_product_code] +=
new_ratio
                end
                new_ratios
              end
            end
          end

          committee :product_line_ratios do
            quorum 'from trade industry ratios', :needs =>
:trade_industry_ratios do |characteristics|
              naics_codes =
characteristics[:trade_industry_ratios].keys
              industry_product_lines =
IndustryProductLine.where(:naics_code => naics_codes)
              characteristics[:trade_industry_ratios].inject({})
do |new_ratios, (naics, ratio)|
                industry_product_lines.
                  find_all { |i| i.naics_code == naics}.
                  each do |industry_product_line|
                  new_ratios[industry_product_line.ps_code] ||= 0
                  new_ratio = ratio * industry_product_line.ratio
                  new_ratios[industry_product_line.ps_code] +=
new_ratio
                end
                new_ratios
              end
            end
          end

          committee :non_trade_industry_ratios do
            quorum 'from industry', :needs => :industry do
|characteristics|
              if characteristics[:industry].trade_industry?
                {}
              else
                { characteristics[:industry].naics_code.to_s => 1
}
              end
            end
```

NAICS 339991 chosen because it’s emissions intensity is close to the average of the entire U.S. economy (calculated by multiplying each sector’s emissions intensity by it’s share of total 2002 value)

```
              quorum 'from merchant category industries', :needs =>
:merchant_category_industries do |characteristics|
                characteristics[:merchant_category_industries].
                  reject { |mci| mci.industry.trade_industry?
}.inject({}) do |ntir, merchant_category_industry|
                    ntir[merchant_category_industry.naics_code] ||=
0
                    ntir[merchant_category_industry.naics_code] +=
merchant_category_industry.ratio
                    ntir
                end
              end



              quorum 'default' do
                { '339991' => 1 }
              end
            end

            committee :trade_industry_ratios do
              quorum 'from industry', :needs => :industry do
|characteristics|
                if characteristics[:industry].trade_industry?
                  { characteristics[:industry].naics_code.to_s => 1
}
                else
                  {}
                end
              end

              quorum 'from merchant category industries', :needs =>
:merchant_category_industries do |characteristics|
                characteristics[:merchant_category_industries].
                  select { |mci| mci.industry.trade_industry?
}.inject({}) do |tir, merchant_category_industry|
                    tir[merchant_category_industry.naics_code] ||=
0
                    tir[merchant_category_industry.naics_code] +=
merchant_category_industry.ratio
                    tir
                end
              end

              quorum 'default' do
                {}
              end
            end
```

a dictionary to go from merchant categories to industries

```
          committee :merchant_category_industries do
            quorum 'from merchant category', :needs =>
:merchant_category do |characteristics|

characteristics[:merchant_category].merchant_category_industries
            end
          end

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

          committee :adjusted_cost do
            quorum 'from cost and date', :needs => [:cost, :date]
do |characteristics|
```

FIXME TODO: Import CPI conversions

```
              @cpi_lookup ||= {
                2009 => 1.189, 2010 => 1.207, 2011 => 1.225, 2012
=> 1.245,
                2013 => 1.265 }

              date = characteristics[:date]
              date = date.is_a?(String) ? Date.parse(date) : date
              conversion_factor = @cpi_lookup[date.year] || 1.207

              characteristics[:cost].to_f / conversion_factor
            end
```

This is the average federal government purchase card transaction in 2003, converted to 2002 dollars, with tax taken out See http://www.sba.gov/advo/research/rs226tot.pdf

```
            quorum 'default' do
              517
            end
          end

          committee :cost do
            quorum 'from purchase amount and tax', :needs =>
[:purchase_amount, :tax] do |characteristics|
              characteristics[:purchase_amount].to_f -
characteristics[:tax].to_f
            end
```

Based on http://www.thestc.com/STrates.stm weighted by US Census 2010 projected state population (exclude samoa, guam, pr)

```
            quorum 'from purchase amount', :needs =>
:purchase_amount do |characteristics|
              characteristics[:purchase_amount].to_f / 1.0711
            end
          end

          committee :date do
            quorum 'default' do
              Date.today
            end
          end
        end
      end
    end
  end
end
```