An Object-Oriented Design for a New Generation Mobile Source Emissions Model

Alison Eyth, Kimberly Hanisak, Prashant Pai, Catherine Seppanen
MCNC Environmental Modeling Center
P.O. Box 12889, Research Triangle Park, NC 27709
http://www.emc.mcnc.org/projects/ngm
eyth@emc.mcnc.org

ABSTRACT

An object-oriented design for a new generation mobile source emissions model (NGM) is described, along with the process used to develop the design. EPA OTAQ plans for the NGM to replace its current generation of mobile source emissions models (i.e. MOBILE 6, PART, MOBTOX, NONROAD). The NGM will handle all pollutants emitted from diverse types of vehicles at geographic scales from individual intersections to the whole nation, and time scales from minutes to years. An object-oriented design is desirable because it results in maintainable, extensible software and it facilitates code reuse. The object-oriented design process used for the NGM began with interviews of users of current mobile source emissions models. The interviews were documented and distilled into use cases, which were documented in a standard format. A graphical user interface (GUI) for the NGM was designed, in addition to classes that could be used to implement the GUI. A system architecture and set of classes was developed to support all of the use cases. Many of the classes used are familiar concepts to mobile source emissions modelers (e.g. VehicleType, Fuel, Fleet, Pollutant). Key design concepts include a Data Manager, Project, Aggregator, Importers, and Exporters. A special class called RunSpecification defines a model run to generate a specific set of emissions [factors]. Default county and emission rate databases provide input to the emission factor calculation. The potential role of EPA ORD's Multimedia Integrated Modeling System (MIMS) in the development of the NGM is examined.

INTRODUCTION

The New Generation Model (NGM) for mobile source emissions is being developed by EPA's Office of Transportation and Air Quality (OTAQ). The NGM is intended to include, and improve upon, the functionality provided by current used emission factor estimation tools such as MOBILE, PART, MOBTOX, and NONROAD. The NGM will eventually estimate emissions for all mobile sources and all pollutants, at geographic scales ranging from individual intersections to the whole nation, and temporal scales from minutes to years. It must also facilitate model validation, uncertainty analysis, and sensitivity analysis. The NGM software that is developed must be updateable, useable, well documented, perform well, and work in conjunction with other models as needed. An object-oriented design process is being used to achieve these goals. For more information on the NGM, see http://www.epa.gov/otaq/ngm.htm.

The object-oriented design process followed for the NGM had three main parts:

- conduct user interviews to develop use cases,
- design a graphical user interface (GUI) that supports the use cases, and
- design a system architecture that supports the use cases.

Designing a system architecture includes the following: identifying major subsystems to be developed, considering existing tools that might assist with the implementation, and defining classes that can be used to develop the system.

OTAQ was interested in understanding whether the NGM could make use of the Multimedia Integrated Modeling System (MIMS) that is being developed by EPA ORD. The goal of MIMS is to provide a computing framework that will facilitate the coupling of diverse sets of models, such as those that would be involved in a multimedia simulation. MIMS will also serve as a replacement for EPA's Models-3 computer framework. The use of MIMS during NGM development is discussed.

BODY

Why an Object-Oriented Design for the NGM?

The NGM is being developed using an object-oriented design. Since most current mobile emission models today do not use an object-oriented design, why use one for the NGM? There are many reasons:

• Good object-oriented designs make use of real-world objects that users are familiar with (e.g. Fleet, Pollutant, Fuel). This **brings the system and the users closer together** and requires less of a mapping from the user's view of the world to the system view.

• By using **inheritance**, general concepts can be abstracted out and specialized as needed. Applying inheritance **reduces the volume of the code that must be developed** because it allows general pieces of code to be written once, and specialized code is written as needed for derived classes.

• The classes of an object-oriented design include both data and methods to operate on that data. When designed properly, a class is responsible for the state of its own data, and data in other classes is only modified via well-defined interfaces. This is known as **encapsulation**. Applying this principle reduces unwanted side effects in other parts of the system when a specific part of the system is modified, thereby speeding development by **reducing the number of bugs**.

• Applying encapsulation reduces the volume of the code that must be developed because it facilitates the development of useful subsystems and GUIs that can be **written once and reused many times**. A side effect of GUI component reuse is that the user interface appears more consistent.

• Good object-oriented designs are flexible systems, to which new features can be easily added and **development can proceed incrementally**. This is important to the NGM because it will be a complex system with a broad scope that will be developed incrementally over time.

Since inheritance is such an integral part of object-oriented design that may be new to many readers, here is an example of how it can be used:

Shape	**Circle** extends Shape	**Square** extends Shape	**Triangle** extends Shape

float area = -1;
abstract calcArea();
float getArea() { if area <0 calcArea(); return area; }
	float radius;
	calcArea() { area = PI*radius*radius; }
	float getRadius();
	setRadius(float r);
		float side;
		calcArea() { area = side*side; }
		float getSide();
		setSide(float s);
			float base, height;
			calcArea() { area = 0.5*base*height; }
			float getHeight();
			setBaseHt(float b, float h)

So, we have a class Shape, with the methods getArea and calcArea, and a floating point data object called "area". The classes Circle, Square, and Triangle each inherit from Shape. Inheritance embodies an "is-a" relationship. So, you can say a Circle "is-a" Shape, a Triangle "is-a" Shape, etc. The derived classes provide implementations for the calcArea method, which computes the area of the shape using the appropriate formula and stores it in "area". The system will automatically decide which version of calcArea needs to be called at run-time based on the specific class of the Shape that is being operated on. The classes derived from Shape have access to the getArea() method from their parent class, so that code only needs to be written once. The derived classes have additional methods and data not stored in the parent class. For example, the Square class has a data object called "side" that represents the length of its side, and corresponding methods getSide and setSide.

In summary, applying object-oriented design principles to the NGM should allow for faster system development, and result in a more usable, extensible system than earlier generations of mobile source emissions models.

User Interviews

Prior to developing the NGM design, MCNC and EPA OTAQ conducted user interviews with users of current mobile emissions models from governments (federal, state, and local), universities, and industry. The users interviewed were: Matt Barth (UC Riverside), Kip Billings (WFRC), Rick Dowling, Alan Huber (EPA ORD), Rob Ireson, Mark Janssen (LADCO), Mike Keenen (NY DEC), Mohamed Khan (MD Dept. of Environment), Harold Nudelman (NYC DEP), Alison Pollack (ENVIRON), Catherine Seppanen (MCNC), Bruce Spear (US DOT), Greg Stella (EPA OAQPS), and many members of the NGM Team in Ann Arbor, Michigan.

The interviews examined the use of the models with respect to 20 use cases defined by OTAQ. These use cases included topics such as macroscale inventory development for report generation, policy evaluation studies, and interaction with transportation models and emission processing systems. Use cases look at model design and system requirements from a user's standpoint. Each use case is a narrative that describes the sequence of events that a user goes through to use the system for a specific task. This provides a natural mapping between the user's tasks and the system requirements and allows for a better system design. The information from the user interviews was incorporated into a standard format for each use case. Notes from the NGM user interviews are available on the web at http://www.emc.mcnc.org/projects/ngm/users.html.

Each user interview included some basic questions, in addition to topics that were specific to each use case. Some of the basic questions were:

- Which mobile source emission models (MSEMs) do you normally use?
  (e.g. MOBILE, NONROAD, PART, also travel demand models)
- How do you typically use these models? (for each use case)
- How long does the modeling process currently take for each study?
- How often do you perform the modeling process?
- What changes that could be made to the MSEMs that would make your work easier?
- Do you run any models or programs that feed data to or get data from MSEMs?
- What data do you use as inputs to MSEMs, and for which inputs do you use defaults?
- What data do you get from the MSEMs and what do you do with it?
- Are there any analysis tools you use with the MSEMs or standard reports that you generate?

MCNC Derived Use Cases

After the user interviews were completed, the use cases were organized into an object-oriented structure. This structure allowed use cases to extend (or derive/inherit from) and use (or include) other use cases. The use cases were grouped into the following categories: Create Emission Factors, Create Inventory (macro, meso, and microscale), Compare (against another inventory or against a target), Interact with External Models, Evaluate Policies, Analyze System, and Extend Model. For each use case, run specifications (that define the system goals and behavior), inputs, outputs, and user and system courses of action were identified. The MCNC derived use cases are shown in Table 1.

Table 1. MCNC Derived Use Cases

I. Create Emission Factors	IV. B. Interact with Models that Provide Input
II. Create Inventory	1. Interact With Growth Models
A. Create Macroscale Inventory	2. Interact With Transportation Models
- Generate Trends Report	**V. Evaluate Policies**
- Generate Greenhouse Gas Emis. Report	A. Perform Regulatory Impact Analysis
- Generate International Reports	B. Evaluate New Vehicle/Equipment Tech.
B. Create Mesoscale Inventory	C. Evaluate New Fuels
C. Create Microscale Inventory	D. Evaluate In-Use Emissions Reductions
III. Compare	E. Evaluate Travel Time Reductions
A. Compare Against Another Inventory	F. Evaluate Travel Modifications
- Develop SIPS	**VI. Analyze System**
B. Compare Against Targets	A. Validate Model
- Analyze Conformity	B. Perform Sensitivity Analysis
- NEPA	C. Assess Model Uncertainty
- Analyze Rate of Progress	**VII. Extend Model**
IV. Interact with External Models	A. Extend System
A. Interact with Models that Drive the NGM	B. Adapt for International Use
1. Interact With Dispersion Models	C. Update Data
2. Interact With Emission Processors	D. Create New Objects

The Use Case Diagram in Figure 1 shows how the use cases relate to one another. Each bubble represents a different use case. Solid lines represent inheritance, where the lower use case inherits from the higher use case. For example, **Macroscale Generation** extends **Create Inventory** so it is a specialization of the Create Inventory use case. Dotted lines indicate that one use case “uses” another use case. Thus, the actions of the “using” use case include those of the “used” use case. For example, **Create Inventory** uses **Create Emission Factors**, so emission factors are created during the process of generating an emissions inventory. In the detailed use case documentation, the fields that appear for each use case include: Name, Description, Users, Frequency, Extends, Uses, Derived Use Cases, Inputs, Run Specifications, Outputs, User and System Actions, Issues, Decisions, Interviewees, and Notes. A full description of the use case process and documentation of all the derived use cases can be found in NGM Use Cases by MCNC, 2/11/2002.

NGM Graphical User Interface Design

Following the user interviews and development of the use cases, work began on the NGM Graphical User Interface. Use cases are an important predecessor to the system design because one measure of the design's sufficiency is that it supports all of the use cases. A number of GUIs were designed to satisfy the use cases. The Run Specification GUI shown in Figure 2 grew out of the need to support setting up the run specifications for the Create Inventory use cases. It contains all the information needed to create a set of emission factors using the NGM. This includes time periods, vehicles / equipment and road types to include, geographic regions, pollutants, and emission processes that factors are needed for, control strategies to apply, input data sets and miscellaneous model controls to use, and the output directory and file formats. The goal is to have the user define the scope of the run based on what he / she wants to get out of it, and then the NGM will **only calculate what the user asked for, as efficiently as possible**. The data volume of the output can be controlled by letting the user specify the level of detail or "Breakdown Level".

In Figure 2, the name of the inventory is shown on the top of the window. The GUI has a menu bar at the top, and the lower portion is divided into two panes. The left pane shows groups of data that can be specified for the run. Clicking on one of the groups fills the right pane with widgets that allow the user to set up data relating to that section. The example in Figure 2 has the **Time Periods** data group selected, so information about setting time periods appears on the right-hand pane. The user can specify multiple time periods for which to create emission factors or inventories. A new time period is specified by clicking on **Add**, which causes a small GUI to appear that allows the user to specify a **Start** and **Stop** date (and time, if desired), a **Time step** and a **Time zone**. The resulting time period is displayed in the **Selected** list. Multiple time periods can be added. Emission factors or inventories will be generated for each **Timestep** in each of the time periods in the **Selected** list. Once the time periods have been selected, the user can click **Next** on the bottom of the window or choose another section from the pane on the left.

The NGM Main GUI was designed as a set of menus that provide access to all major components of the NGM. For example, the Run Specification GUI can be accessed by selecting **Create Inventory** from the **Actions** menu on the main NGM Main GUI.. Some additional NGM GUIs include:

- Create Multiple Inventories GUI - allows the user to select multiple Run Specifications that define different inventories and to run the model to create all the inventories in a batch mode;
- Compare Inventories GUI - the user selects inventories to compare against a base inventory and the form in which the comparison should be presented (e.g. difference table, graphics, statistics);
- Compare Against Target GUI - the user selects inventories to compare against a "target", which is a set of numeric limits for specific pollutants in specific locations;
- Project GUI (from MIMS) - a generic interface for creating and editing many types of NGM objects (see the MIMS section for more details);
- Data Browser GUI - assists the user in choosing appropriate data sets / input files for a model run.

Screen mock-ups for each of these GUIs are shown in the document NGM Graphical User Interface Design by MCNC, 2/11/2002.

Figure 1. MCNC Derived Use Case Hierarchy for the NGM

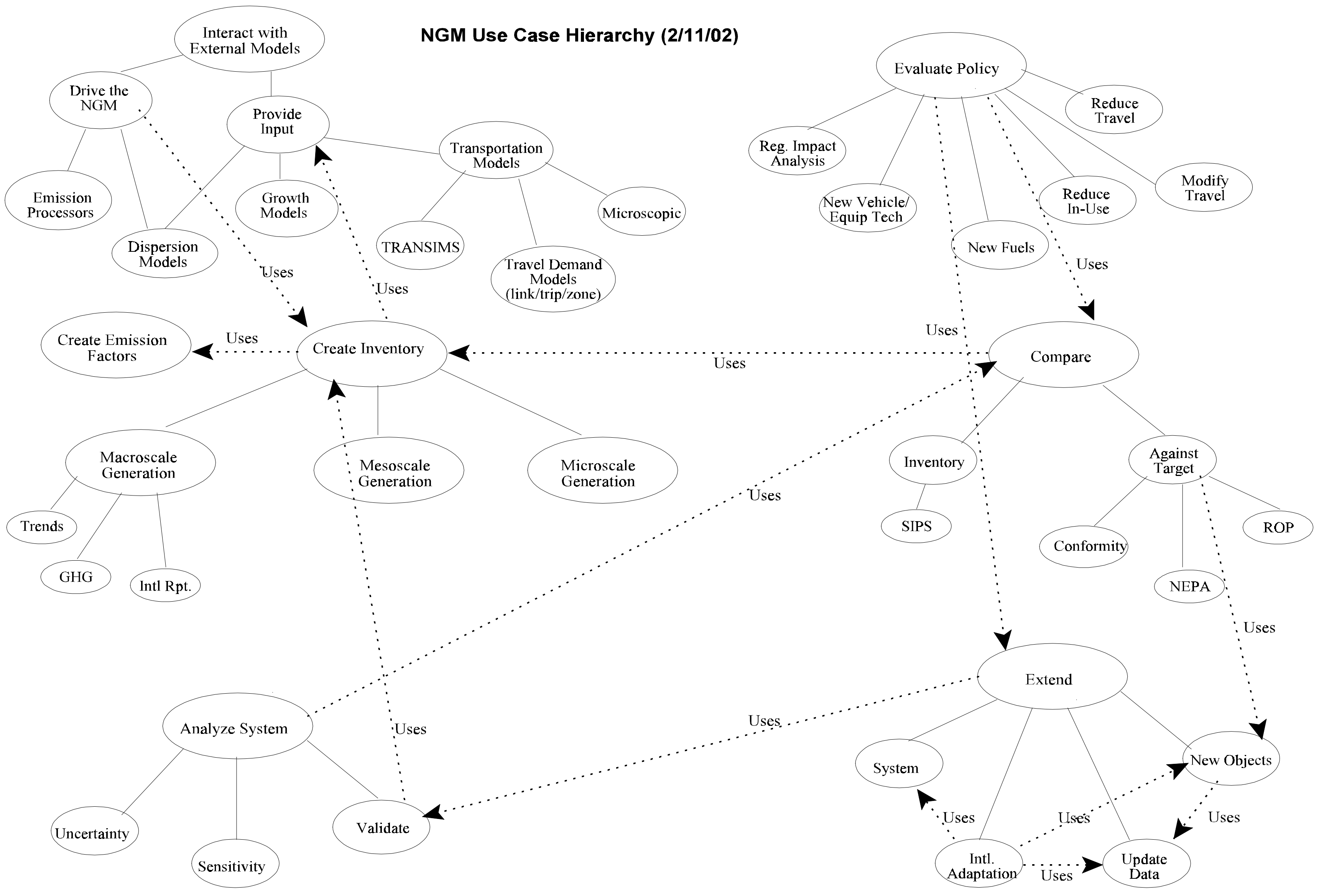


Figure 2. Run Specification GUI

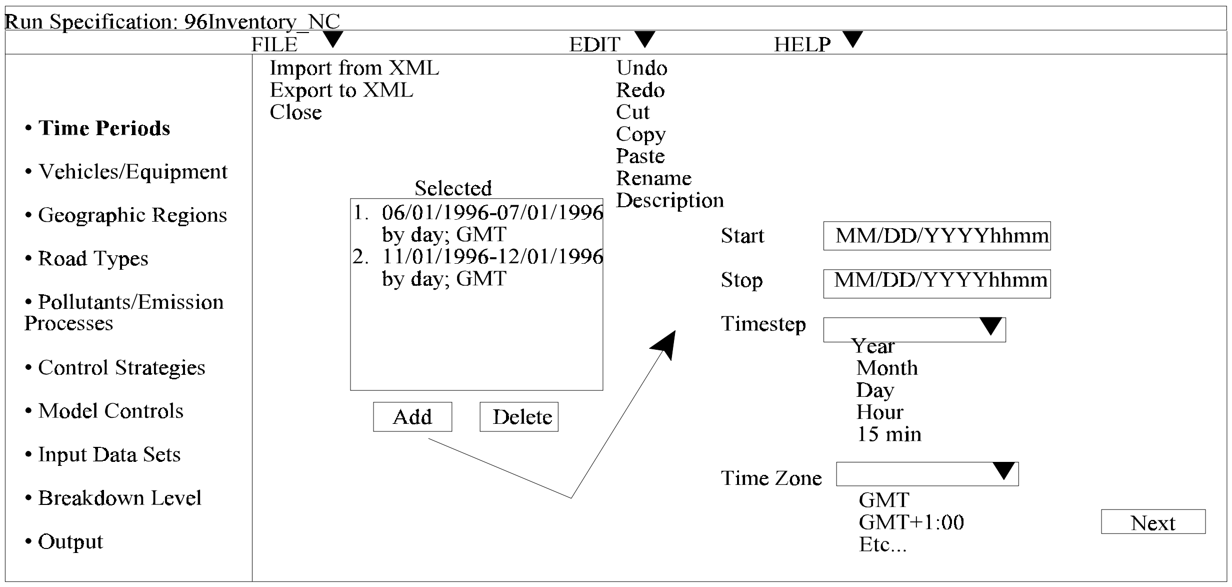


Figure 3. Geographic Regions Pane

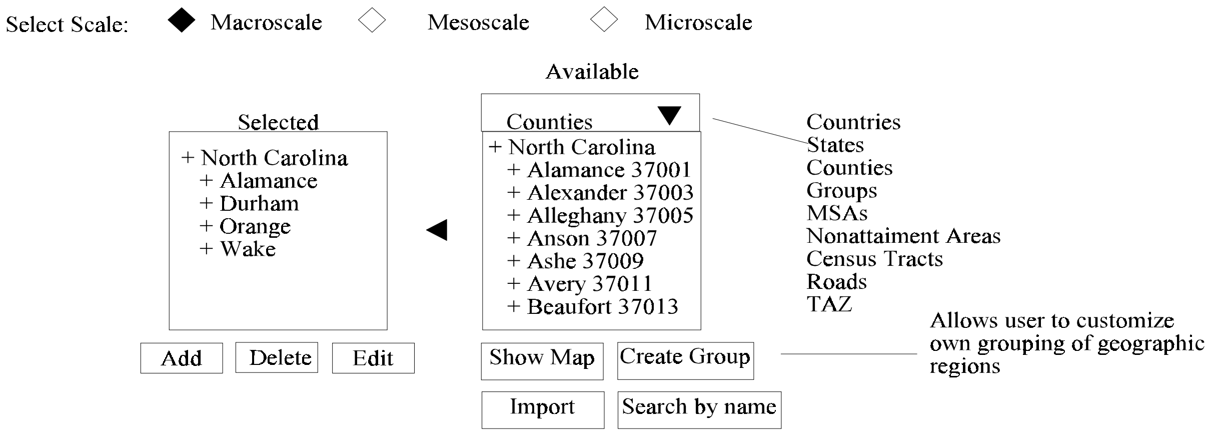


Figure 4 MIMS / NGM Project GUI

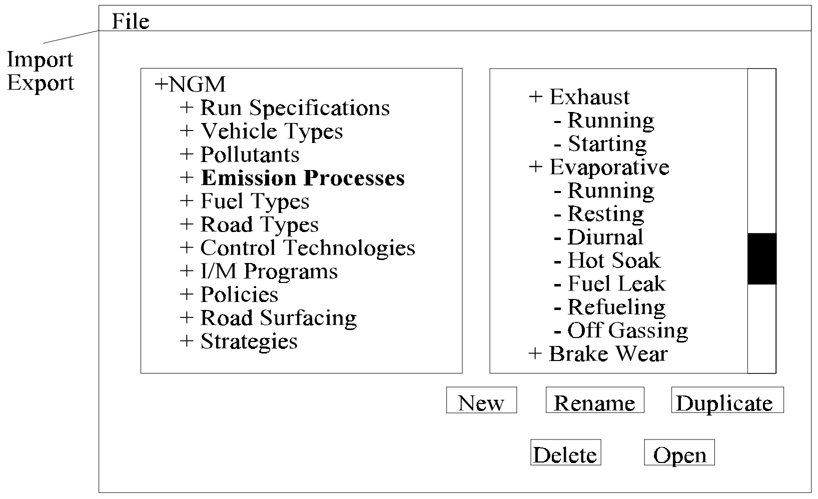


Figure 3 shows the **Geographic Regions** pane of the Run Specification. Before selecting the geographic regions / objects to model, the user must select a scale (i.e. **Macroscale**, **Mesoscale**, or **Microscale**). Based on this selection, the NGM will limit the geographic (and input data set) options to those that correspond to the chosen scale. The user must select at least one geographic object to model. Geographic objects are grouped hierarchically for easier selection. The objects in the **Available** tree will be populated according to the selected value in the drop down menu above the tree (e.g. Counties). The user can select any or all of these available geographic objects and click the arrow to copy them to the **Selected** window. The user can create their own customized group by selecting **Create Group** and these groups will be saved for later use. The user can **Import** geographic objects to select from as needed (for example, importing a specific road network would allow the user to pick actual roads for modeling). The user can also **Search by name** to select objects or select them geographically from a **Selectable map**. This GUI and an accompanying Selectable Map can be reused to select geographic objects throughout the NGM.

MIMS and the NGM

An important topic to OTAQ was to understand the potential role that EPA ORD's MIMS could play in the NGM development. MCNC's assessment of this was that using MIMS would speed NGM development by providing:

- basic GUI components for building the NGM GUI using on tables of parameters (which can be simple - like float, integer, or String, or complex - like an air quality model grid),
- a straightforward method by which custom NGM GUI components and parameters can be added,
- sensitivity analysis and other tools to run the NGM (and other models) iteratively,
- a basic implementation of a Project,
- a growing suite of plotting and visualization tools (e.g. line plots, bar chars, and GIS),
- a mechanism for defining and executing sets of related models,
- and an existing code base that supports the above functionality.

For example, NGM development could be sped up by using a slightly modified version of the MIMS Project GUI and Project classes for the NGM. The new version of the Project GUI is shown in Figure 4. The objects in the Project are indexed by type. Selecting the object type on the left populates the window on the right with the available objects of that type. From this GUI, many types of NGM objects can be **created, opened, renamed, duplicated,** and **deleted**. The objects can be imported or exported for use in other NGM projects using **Import** and **Export** on the **File** menu. Note that all NGM Run Specification objects reside within some Project, and the Project is used as a source of data when Run Specifications are populated.

NGM GUIs and Databases

The "Subsystems and GUIs" diagram in Figure 5 shows the relationship of the GUIs to major NGM data sources and components. At the center of this diagram is the **Mobile Emissions Inventory Generator** (MEIG). This is the core of the NGM. It generates the emission factors or inventories that the user wants according to the parameters defined in Run Specifications. The NGM GUI is accessible from the MEIG and will provide access to all of the other GUIs in the NGM system via a set of menus. The dashed arrows represent method calls that cause other GUIs to appear.

Three databases will be used by MEIG to generate emission factors or inventories (in addition to user supplied input files, which are not shown on this diagram). The first, in the lower left portion of the diagram, is the **Default County Database**. It will contain information for each county in the United States required for macroscale analyses, such as average meteorology, VMT from HPMS, fleet /

registration information, fuel usage information, and inspection and maintenance programs. The **Emission Rate Database** will contain emission rates for the emission processes available in the NGM (e.g. exhaust, evaporative, brake wear, refrigerant leakage, …) . The emission rates are stored by the operating modes that are appropriate for each process. The single-headed solid arrows from the Default County and Emission Rate databases show the direction of the data flow. They mean that MEIG reads the data from these databases, but does not write it. A **Mobile Emissions Database Editor** has a GUI that will assist NGM data developers with editing and entering new items in the Default County and Emission Rate databases. The double-headed arrows between the databases and the Mobile Emissions Database Editor mean that the editor both reads and writes data in the databases.

The execution of MEIG involves a third "database", shown in the upper left portion of the diagram - a **MIMS Project**. MIMS Projects contain objects such as Run Specifications, Fuels, Vehicle Types, Road Types, and Control Strategies. These objects are used during the set up and execution of MEIG runs. Objects in MIMS Projects are stored to disk using a **Simple Object Database** (currently based on Java serialization). MIMS Projects can also read and write XML. Three types of GUIs are available from the MIMS Project. First, the **Project GUI** shows the types of objects in the Project, and the instances (i.e. choices) for each object type (see Figure 4). It is shown when a Project is opened. The **Generic Object GUIs** provided by MIMS are a default way to view / edit the objects in the project as sets of parameters and processes. Any object in the Project can be viewed with a Generic Object GUI, but custom GUIs can also be defined. The third type of GUI accessible from the project, the **Run Specification GUI**, is such a custom GUI. It allows the user to define all of the information needed to run the MEIG, such as time periods, geographic regions, emission processes, pollutants, and more.

When the user selects input data sets from the Run Specification GUI, the conceptual **MIMS Data Browser GUI** (shown in the upper right portion of the diagram) will be used. This GUI will allow the user to query data sets from various data sources known to MIMS by attributes such as time period, geographic region, data type, and file format. The values for the queriable attributes are stored in a **Metadata Database**. This database may or may not be part of the Simple Object Database.

The **Aggregated Emission Factor Database** is the primary output from MEIG. It contains all the emission factors generated by a MEIG execution. This data may be analyzed by three other NGM subsystems shown on the right side of the diagram. First, the **Compare Against Target GUI** will allow the user to compare emissions against a numeric target such as a SIP emissions cap. The **Compare Inventories GUI** will allow the user to compare multiple emission inventories to a base inventory. **Visualization Tools** will allow inventories to be visualized and graphics generated.

NGM Classes

The use cases and GUI design helped to jump-start the process of defining classes to be included the NGM system design. Some of the basic classes derived from the use cases and GUI design are the following:

- VehicleType (e.g. truck, car, tractor, lawnmower)
- Fuel (e.g. gasoline, diesel, natural gas, LPG, RFG)
- Fleet - the distribution of vehicles at a specific time and location
- Pollutant (e.g. HC, NOx, CO_2, CFC, benzene)
- EmissionProcess (e.g. running exhaust, brake wear)
- ControlStrategy - a description of how emissions should be modified, when, and where
- Region (e.g. state, county, non attainment area, country)
- RoadType (e.g. freeway, arterial, local, ramp)

- RoadSurface (e.g. asphalt, concrete, stone, dirt)
- Road - a specific road, including coordinates
- RoadNetwork - a collection of Roads

In addition to the classes above that should represent familiar concepts to the user, new classes were derived based on the GUI requirements, such as RunSpecification (all the information needed for a particular model run), RunSpecificationGUI to display the information in the RunSpecification, GeographicObjectSelectionPanel for selecting sets of geographic objects, etc.

Considering the actual process to be used for calculating emission factors or emissions also resulted in many classes being added to the design. The process is illustrated in Figure 6, "Process to Calculate Emission Factors or Emissions". Pseudo-code that describes this process in more detail, and complete descriptions of the current state of the NGM class hierarchy and class list are given in the document NGM Object-Oriented Design by MCNC, 2/11/2002. The calculation is initiated either from the RunSpecificationGUI, or from the NGM Main program as a batch process. Either way, the execute method of the RunSpecification class is called. This method will iterate over each time step and location that the user specified for the run. It begins its work for a particular time and location by gathering the data it needs from the DataManager, such as the Fleet and RoadNetwork. The DataManager collects the data needed for a particular time and location needed for the run from the Default County Database, input files specified by the user, and the Project.

Next, the execute method loops over each emission process in the RunSpecification, The emission rates for the current emission process are retrieved from the Emission Rate Database. The rates, Fleet, and other data are passed to the calcEmissionFactors method for the EmissionProcess. This method calculates the emission factors using a calculation specific to the emission process (e.g. running exhaust emissions are calculated using a different formula than brake wear emissions). The calculated factors are then aggregated to the user's level of interest by the Aggregator, and exported in the user's desired formats by appropriate Exporters.

In Figure 6, many of the classes shown are not derived as directly from the use cases, but are inferred from them and are used to make the system work and to make it modular. The general principle of placing a class in between the core of the system and any data sources or sinks is shown in the DB Mgr classes, Importers, Exporters, and at a higher level, in the DataManager. The DB Mgr classes handle the details of actually getting data into and out of databases. Subclasses of the Importer class are used to insulate the internals of the NGM from the various file formats and data contents of files that are inputs to the NGM (e.g. different types of transportation models). Subclasses of the Exporter class will be able to take data stored in the Emission Factor DB and output it in various formats.

The DataManager is a bit smarter than the Importers and DB Mgrs, it knows how to make use of all of its data sources in order to return key pieces of data to the rest of the system for specific times and places. Thus, the DataManager insulates the rest of the system from having to worry about the details of how to gather the data needed for the calculations. Another inferred class is the Aggregator, which implements the system requirement that users want output data at different levels of resolution by providing the ability to aggregate over vehicle type, age, model year, fuel type, emission process, road type, SCC, and other categories.

CONCLUSIONS

An object-oriented design can benefit the NGM. It will provide a flexible framework capable of satisfying the many design goals of the NGM. Some of the key design concepts applied are:

- use objects familiar to the users wherever possible,
- insulate the core of the model from the peculiarities of file formats and database accesses using the concepts of the Data Manager, Importers, Database Managers, and Exporters;
- use a Project to contain objects used in model runs
- capture the areas of user interest for a particular model run in the RunSpecification class, and only perform calculations necessary to meet those user goals
- use inheritance (e.g. for EmissionProcess classes) to exploit the commonalities but account for the differences in the factors calculated for each process
- create an Aggregator to aggregate the emission factors to the level the user is interested in.

It was shown that the use of MIMS could speed the NGM development process. The proposed design was found to satisfy the use cases. The mechanisms for supporting each use case are shown in Table 2.

BIBLIOGRAPHY

Eyth, A.M., K. Hanisak, P.Pai, C. Seppanen, “NGM Use Cases by MCNC – 2/11/2002", Prepared for the U.S. Environmental Protection Agency (EPA) Office of Transportation and Air Quality by the MCNC Environmental Modeling Center, 2002.

Eyth, A.M., K. Hanisak, P. Pai, C. Seppanen, “NGM Graphical User Interface by MCNC – 2/11/2002", Prepared for the U.S. EPA Office of Transportation and Air Quality by the MCNC Environmental Modeling Center, 2002.

Eyth, A.M., P. Pai, K. Hanisak, C. Seppanen, “NGM Object-Oriented Design by MCNC – 2/11/2002", Prepared for the U.S. EPA Office of Transportation and Air Quality by the MCNC Environmental Modeling Center, 2002.

KEY WORDS

Mobile Source
Emission Model
New Generation Model
Object-oriented Software

ACKNOWLEDGMENTS

This work was funded by the US Environmental Protection Agency’s Office of Transportation and Air Quality under GSA Task Order ID 4TAG68012752.

DISCLAIMER

This design is still under review by EPA and should not be considered a final design for the New Generation Mobile Source Emissions Model.

Table 2. How the Design Satisfies the Use Cases

Use Case	Design Concept(s)
Create Emission Factors	RunSpecification defines factors to create. See also DataManager, RunSpecification.execute(), EmissionProcess.calcEmissionFactors()
Create Inventory	RunSpecification defines what inventory to create. Execution triggered from GUI or batch. Aggregator converts from emission factor to emission inventory.
Create Macroscale Inventory, Trends, Greenhouse gas, International Reprtg	See Create Inventory case; Exporters output data in formats needed for reports, and use Regions for geographic objects.
Create Mesoscale Inventory	See Create Inventory; Importers to get data from transportation models, and Roads for geographic objects
Create Microscale Inventory	See Create Inventory; Importers to get data from trans. models; use Roads and Intersections for geographic objects
Compare Inventories, SIP	Compare Inventories GUI
Compare Against Target, Conformity, NEPA, Rate of Progress	Compare Against Target GUI
Interact with Models that Drive the NGM to get its output	Java API allows other Java models to drive NGM, others use file-based (e.g. XML) version of RunSpec. and a cmd line interface. Exporters write data in other formats
Interact with Transportation Models	Importers read data output by various transportation models into the NGM.
Interact with Growth Models	Data Manager calls growth modules appropriately for years in the future. Importers read data from other models.
Extend System	Object-oriented design makes this straightforward
Update Data	Mobile Emissions Database Editor
Create New Objects	MIMS Project to store NGM Objects; implement Parameter Specifiers and use Parameters for NGM Objects
Adapt for International Use	See Extend System, Update Data, Create New Objects
Evaluate Policies	Control Strategies and Strategy Rules. See also Create Inventory, Compare Inventories, Compare Against Target, Create New Objects, and Update Data
Sensitivity Analysis	MIMS iteration and sensitivity analysis tools.
Uncertainty Analysis	Include standard deviation in emission rate database; MIMS uncertainty analysis tools.
Model Validation	Piecewise due to object-oriented design. Also use Compare Inventories and Compare Against Target GUIs.

Figure 5. NGM Subsystems and GUIs

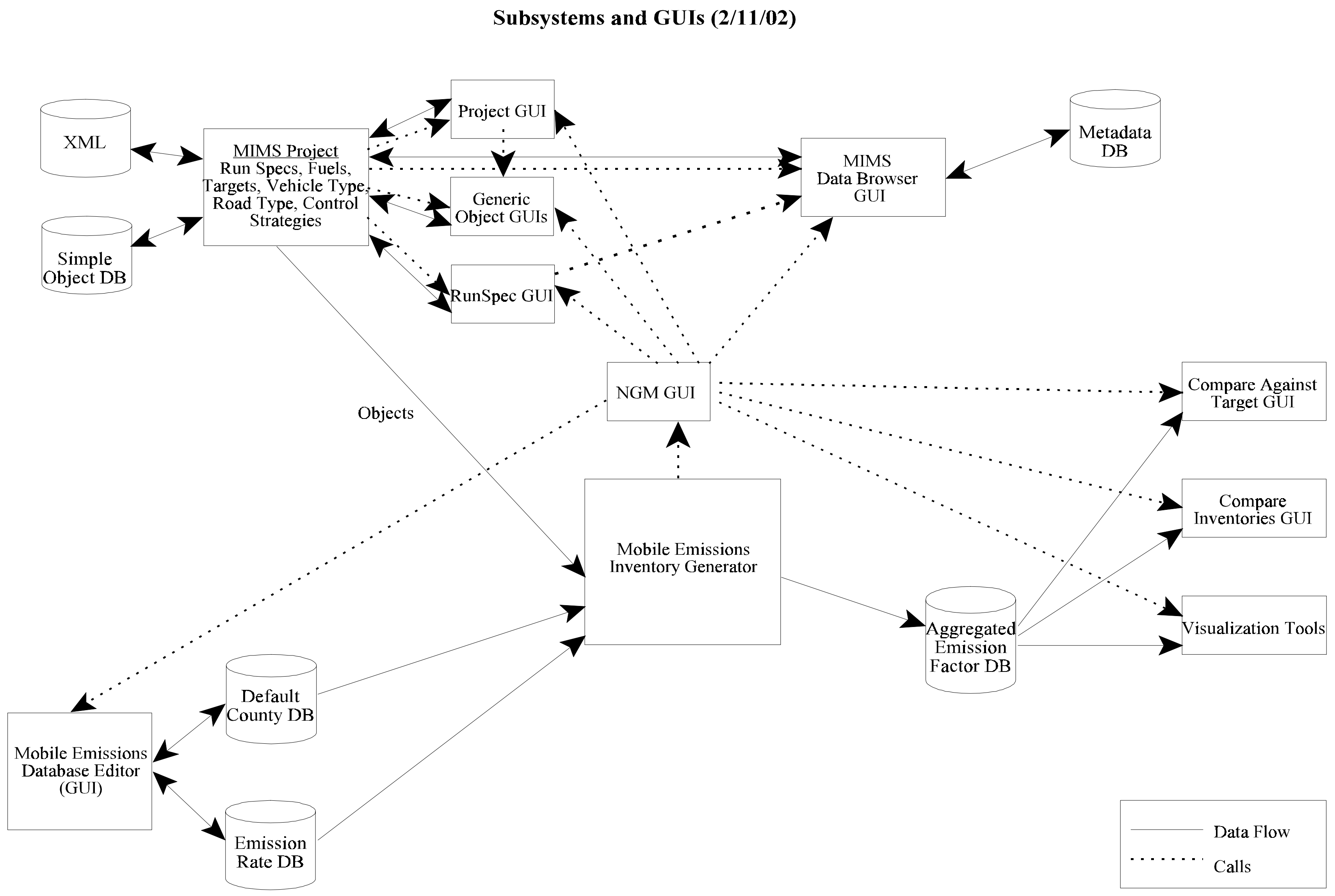
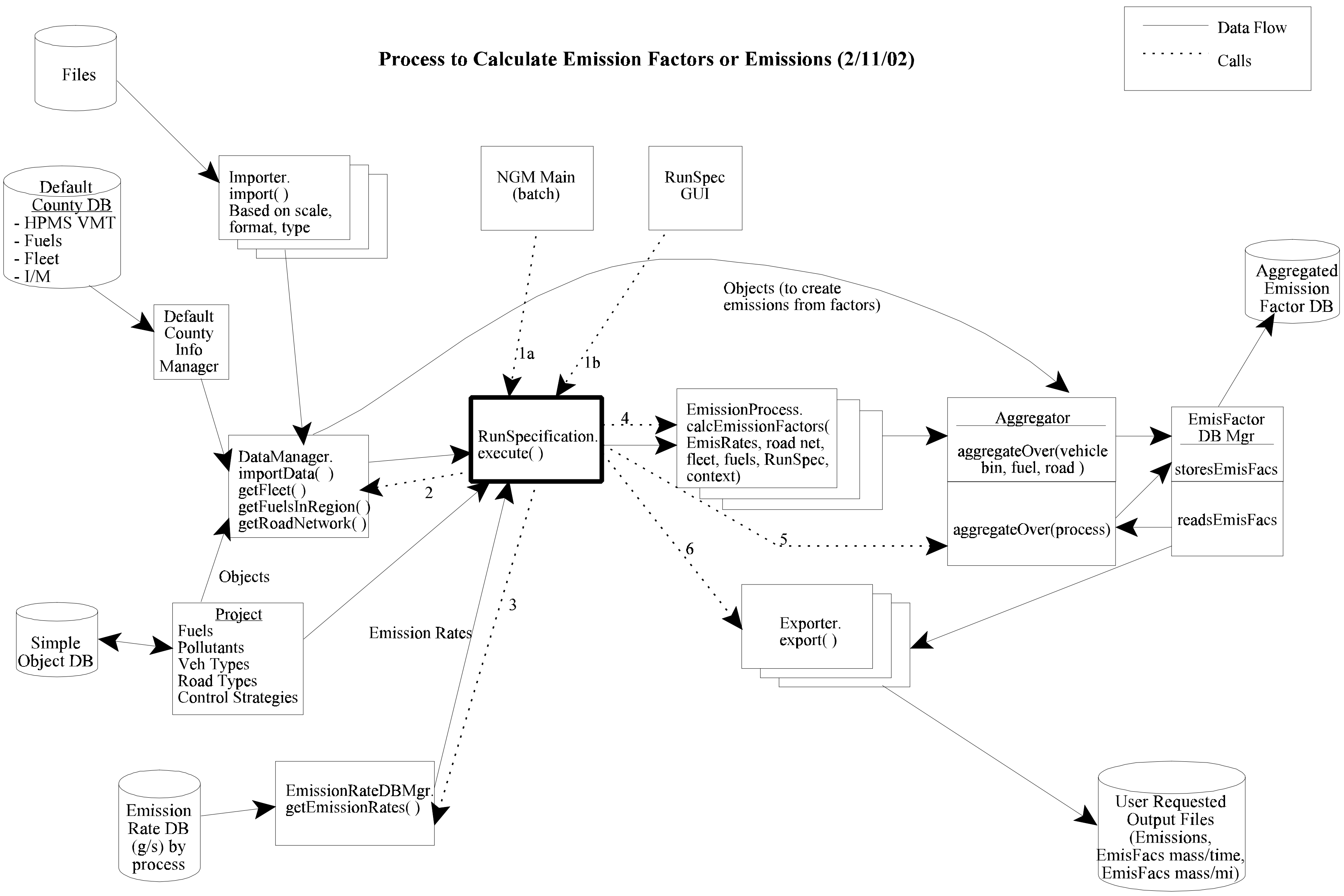

Subsystems and GUIs (2/11/02)
XML
Simple Object DB
MIMS Project Run Specs, Fuels, Targets, Vehicle Type, Road Type, Control Strategies
Project GUI
Generic Object GUIs
RunSpec GUI
MIMS Data Browser GUI
Metadata DB
NGM GUI
Objects
Compare Against Target GUI
Compare Inventories GUI
Visualization Tools
Mobile Emissions Inventory Generator
Aggregated Emission Factor DB
Mobile Emissions Database Editor (GUI)
Default County DB
Emission Rate DB
Data Flow
Calls

Figure 6.


Process to Calculate Emission Factors or Emissions (2/11/02)
Data Flow
Calls
Files
Importer.
import( )
Based on scale,
format, type
NGM Main
(batch)
RunSpec
GUI
Default
County DB
- HPMS VMT
- Fuels
- Fleet
- I/M
Default
County
Info
Manager
Objects (to create
emissions from factors)
Aggregated
Emission
Factor DB
1a
1b
DataManager.
importData( )
getFleet( )
getFuelsInRegion( )
getRoadNetwork( )
RunSpecification.
execute( )
2
4
EmissionProcess.
calcEmissionFactors(
EmisRates, road net,
fleet, fuels, RunSpec,
context)
Aggregator
aggregateOver(vehicle
bin, fuel, road)
aggregateOver(process)
EmisFactor
DB Mgr
storesEmisFacs
readsEmisFacs
5
6
Objects
Simple
Object DB
Project
Fuels
Pollutants
Veh Types
Road Types
Control Strategies
Emission Rates
3
Exporter.
export( )
Emission
Rate DB
(g/s) by
process
EmissionRateDBMgr.
getEmissionRates( )
User Requested
Output Files
(Emissions,
EmisFacs mass/time,
EmisFacs mass/mi)