Raj R. Singh

rajsingh@mit.edu
For presentation at ACSP/AESOP 2003

# Speaking the Same Language: Using XML for Distributed and Collaborative Planning Analytics

June 2, 2003

## INTRODUCTION

In his famous article, "Requiem for Large-Scale Models," Lee (1973) concludes, "Probably the most important attribute any model should have is transparency. It should be readily understandable to any potential user with a reasonable investment of effort. 'Black-box' models will never have an impact on policy other than possibly through mystique, and this will be short lived and self-defeating." Almost thirty years later, it seems that this advice has largely been ignored. My doctoral dissertation seeks to make analysis more transparent to domain experts in planning and related fields by defining a common language, or notation, for the process of modeling. This paper provides an overview of the language and presents some examples of how the language may be used.

For decades, computer-based models have played a prominent role in policy formulation, yet little research has gone into making the *process* of analysis more open, and accordingly more useful beyond one researcher or project. We, as a profession, are good at *presenting our results*, but the models that led to those results are shielded from our colleagues.

While the opacity is the primary issue, the planning domain also suffers greatly from a systemic inability for stakeholders to efficiently share information resources. The current state of the art is to pass files—digital or paper—between organizations. This requires the receiving party to have the infrastructure to run their own analyses, and burdens both parties with a bureaucratic process of data exchange every time an update is needed. Seeing as how the general rule of thumb is that more than half of any analytic exercise consists of data gathering, this problem can not be emphasized enough.

Solutions to these problems become even more critical as we enter a time when billions of dollars will be spent on digital, or e-government. The World Wide Web has attracted the attention of all levels of government in the U.S., from the federal E-Government Act of 2002 to the National Civic League's 8th revision of the Model City Charter. The question is how should government leverage electronic networks to improve operations. Corporations were asking themselves the same thing in the 1990s, and responding to the challenges of a digitally connected business environment led to major paradigm shifts at all levels of the enterprise. I believe the same thing will take place in the public sector, offering researchers a golden opportunity to affect dramatic change in the way government administers, archives, and shares information.

This work suggests that models can be described in an exact, domain-independent fashion, and proposes an XML-based language for doing this. I build upon classic analytic techniques, but employ an entirely new way of describing them and how they interact with data to construct models. The result is a "web" of spatial data and geoprocessing services that support distributed and collaborative urban planning analytics. As the full universe of modeling techniques is too broad to be addressed here, I focus on core geospatial modeling algorithms such as map algebra and "McHargian" map overlay analysis.

## RELEVANT LITERATURE

### Place-based modeling

The primary literature consulted for this work is in the field of place-based modeling. Modern work in this area started with Ian McHarg (1969) in his seminal book, *Design with Nature*, where he lays out the technique of map overlay analysis. Despite, or maybe due to, its simplicity, it is still the most common analytic technique in land-based analysis and planning. In fact, the focus of PAMML in this first incarnation is to support McHarg-ian style map overlay analysis.

Geographic modeling is a very large field, and attempting to undertake an exhaustive review of the modeling literature would only lead to confusion. However, a general background in the types of models is appropriate, and Alberti (1999) provides an excellent review, describing a wide range of modeling techniques used in the last forty years. The major categories are:

- The Lowry gravity model and its descendants

- Economic market-based models such as California Urban Futures 2
- Micro-simulations, such as UrbanSim and Clarke's (1997) urban growth model

For this research, what is important in these models is that they are rarely self-contained, but rely on endogenous variables and models from other disciplines, like transportation and economics. Also, they usually make use of advanced statistical techniques like logistic regression. The prevailing trend in modeling in the last decade or so has been to more accurately simulate change by using Monte Carlo approaches, allowing change to be based on previous actions. Finally, the latest trend has been towards using cellular automata, or agent-based modeling to focus the process on understanding urban processes from the point of view of actors in a changing economic and geographic landscape. I envision this work being extended to support these more complex urban models in the future, but not in this paper.

## XML

The most important technology since the advent of the Web is Extensible Markup Language, or XML. XML is really nothing by itself. It is simply a framework in which to write languages for data encoding and system-to-system messaging. What XML provides is a consistent language structure and a way of describing the language's content in the form of XML Schema. Because structure and content are described in XML syntax, the software industry has built powerful, reliable tools to read XML on every operating system and application in common use. It is also very important that XML languages are plain text, so that their content is transparent to humans, even in the absence of computer programs that can read and manipulate the XML. This has a profound effect on people's trust in the content and in the ability of the content to be used in almost all current and future computing environments. The primary literature on XML is found on the World Wide Web Consortium's Web site, http://www.w3.org, which includes specifications for XML, XML Schema, and XML query languages.

## Web services and GIS

"Web services" is an umbrella term to describe systems that allow applications to communicate between computers using XML as a messaging language. The different communication implementation strategies go by many names (the most well known being SOAP, or Simple Object Access Protocol). However, the implementation strategies are not important in this context. What is most important is that all Web services strategies use a well-known and widely implemented Internet protocol for communication—HTTP—the foundation upon which all Web sites operate. While some technologists decry the drawbacks of the Web protocol, the advantages are numerous. The most obvious is that most organizations already have a Web infrastructure in place, so implementing Web services can be handled in a familiar way, and the wealth of Web software can be used to develop and run new Web service-based applications. The other important aspect of Web services is that they use XML for passing messages between computers, preserving the

transparency that has made XML so popular and useful (although some implementations, most notably those promoted by Microsoft in their .NET framework, often still hide the actual message content (data) in a non-human readable format).

The OpenGIS Consortium has chosen a long-term Web services strategy for all of its distributed GIS work. Their specifications and discussion papers, which can be found at http://www.opengis.org, have greatly influenced this work.

## ISSUES IN THE SELECTION OF A NOTATION (META) LANGUAGE

The decision to use XML as a notation language for this task was not a given, Many other possibilities were evaluated, such as the Unified Modeling Language (UML), programming languages such as Smalltalk, Java and C#, and scripting notations. Currently, interfaces to computing systems are mediated by people and their language, and usually by a hierarchy of people. In the terminology of economists, this results in a very high transaction cost for any change to the analytic process. This research seeks to reduce that transaction cost by creating a medium for the computer systems we have come to rely upon to interact directly with each other. This requires a language through which computers can "talk about" plans and models. What is the language of modeling? How can models be part of planning discourse when they are not written in a "language" that everyone understands? What theories drive the design of this language? This section provides the background in which these questions will be explored.

### What the language needs to express

At the computing level, the language must be able to describe basic spatial data and algorithms that are in common use. These are well defined after over thirty years of spatial information system use in planning. The nature of GIS is changing in response to the importance of the Internet, and this makes a host of new GIS paradigms important to capture. I take as a guide in this area the work of the OpenGIS Consortium, an international GIS standards organization that has encoded GIS software's fundamental properties into a series of documents called the "Abstract Specification". The language must also contain placeholders for concepts that are not currently be expressed well in software. These placeholders may look like traditional "black box" models, but at least they may be accommodated in the framework of a more open modeling language.

The next step in understanding language requirements is to understand the planning models that should be expressed. This is, of course, an infinite task, so I will focus my work on the most influential and widely used analytic model, site suitability analysis as first presented by Ian McHarg. Site suitability analysis includes the major elements of many planning analyses, from the basic manipulations of spatial data, to the assignments of weighted importance values based on subjective reasoning, to the importance of being able to substitute alternative data sets and sub-models when desired.

## Choice of language environment

The first step in embarking upon this research area is to decide how to write the language. It is important to understand that software development or design is a secondary concern of this work. It only is addressed in regards to prototyping and proof-of-concept work. At its heart it is concerned with the design of a language to describe a process, specifically the planning analysis process, and formally tying it to computing methods.

My requirements for the language are as follows:

1. Human *and* computer readable
2. Operating system and application independent
3. Interoperable with World Wide Web and GIS standards
4. Clearly described and coupled with model execution

Based on these requirements, three possible implementations were identified: UML, XML and Java. Using a programming language such as Java, or even a scripting language like Perl or Python, fulfills many of the requirements, but code is only readable by programmers. This drawback could be partially alleviated by heavily documenting the code and using tools to present the documentation in narrative and graphic form, but I feel that this represents too close a tie between programming and modeling.

The UML (Unified Modeling Language) is an excellent candidate for this exercise because its graphic notation is accessible to a wider audience, while still retaining the features of a formal method. As stated by Muller (2000), "A method defines a reproducible path for obtaining reliable results. All knowledge-based activities use methods that vary in sophistication and formality. Cooks talk about recipes…architects use blueprints, and musicians follow rules of composition. Similarly, a software development method describes how to model and build software systems (Muller, 2000)." The UML method represents the software industry's consensus on how to graphically describe a software system. The primary problem with the UML is that a graphic notation is not computer readable. Also, the UML is independent of any execution environment, making it difficult to ensure that different applications can interpret the model in the same way and therefore interoperate. Software engineers use the UML to explain high-level ideas about system design, not to directly specify system execution.

While the UML may be used in this dissertation to help explain and document the language and software prototypes, the language itself will be written in XML. I feel that XML offers the best balance between the UML, which offers no implementation, and a programming language such as Java, which is inherently geared towards software engineers, and is too highly structured to offer the types of language elements necessary to construct a narrative. XML is a natural choice for three reasons. First, it offers the flexibility of full control over language definition, while remaining in a structured framework (XML Schema) that all systems can interpret. Second, it is becoming the dominant meta-language for network-aware, computer-to-computer communications, ensuring that many people will be familiar with its basic syntax, and will therefore shorten the learning curve

for familiarizing themselves with the language. Finally, the major Web and GIS standards organizations use XML extensively, and there are a number of XML languages available to build upon that do useful things such as define database queries, encode geographic data and describe the Internet locations of resources.

## A little more about XML

This research will use XML to develop a language called Planning Analysis and Modeling Markup Language (PAMML). XML stands for eXtensible Markup Language. It is a meta-language—a language designed for developing other languages. XML was developed as a way to tag information with metadata and enforce structural rules without requiring that the information be stored in or adhere to the strict rules of a database. It has proved to be a highly successful strategy, as the language is barely five years old and is already considered the standard for describing information outside of databases.

The benefit to writing a language in XML is that you can take advantage of a vast collection of software already developed to process XML, and only write the software that deals with the specifics of your particular language. Furthermore, one XML language can use others to describe generic entities. For example, XML language developers do not have to describe how a person's address should be written. They can simply use an XML address language developed by another information community (such as software companies that develop address book software). More importantly, a great deal of infrastructure needed to make an application work is common to all applications, such as security, authentication, data validation, etc. Using XML makes it possible for a language writer to be confident that their language can take advantage of advances in these areas without requiring major changes to their own work.

The way one develops an XML-based language is to write a rulebook. This is done in an XML language called XML Schema. This document functions as a dictionary—defining the set of terms that can be used—and also as a grammatical reference—enforcing rules about how words are put together to make sense. Additionally, XML Schema has the ability to reference other XML Schemas. This makes it possible to leverage existing work in related areas. PAMML will make extensive use of this mechanism to avoid re-inventing the wheel in the areas of networking, identity management, databases, and GIS. For example, whenever a link to an online resource is required, PAMML will use the World Wide Web Consortium's (W3C) XLink specification to identify the resource. Database access may take advantage of W3C's evolving XQuery specification. In the geographic field, a number of OpenGIS Consortium (OGC) specifications will be used. GML (Geography Markup Language) will be a supported data set format, and GML will also be used as the "native" geographic object language. WFS (Web Feature Service) will be a supported data format, in concert with the Filter encoding specification, which defines queries on geographic data.

## KEY ASPECTS OF PAMML

The language under development goes by the name of Planning Analysis and Modeling Markup Language, or PAMML (when spoken it rhymes with 'camel'). The full notation rules are too long to reproduce here, but can be found at http://web.mit.edu/rajsingh/www/xml/schemas/pamml/pamml.xsd. At this point, explaining the key features of the language is more important than the details.

The first thing one must understand is that PAMML is not software—it is a language around which people may build software. One might build an application that can run a model described with PAMML, or one might build an application that lets users build PAMML models in a graphical interface, possibly never seeing XML.

**Figure 1: Standalone setup**

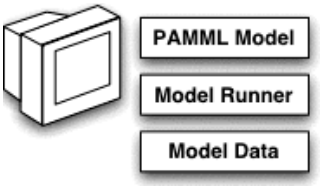



**Figure 2: Model description separate from data and application**

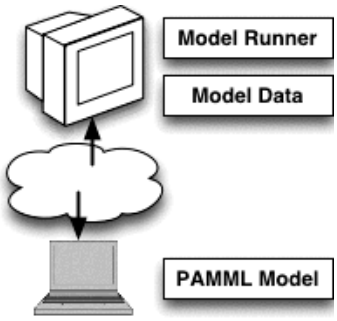
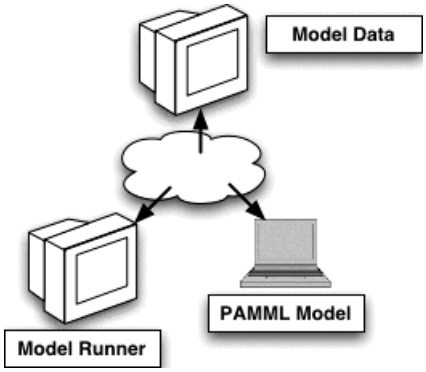



**Figure 3: Complete distribution of resources**




### Internet-centricity

In PAMML, a model is an XML document that can be accessed via one or many URIs (Uniform Resource Identifiers, see http://www.w3.org/Addressing/ for a definition). URIs allow the model to be run, read, changed, or substituted with an alternative model. URI identification is critically important so that:

- Local resources can be described in the same way as remote ones
- Model output can be a static file, or the result of a dynamic request, using standard Web technologies, e.g. CGI, Java Servlets or Active Server Pages
- Model publishing and execution can leverage Web server technologies for sharing and collaboration

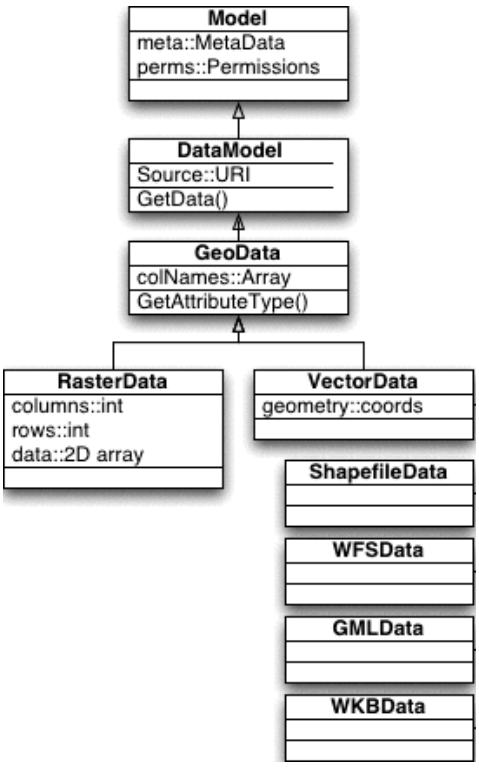
As shown in Figure 1, Figure 2 and Figure 3, models; the software that executes them; and the data on which they run may be located anywhere on the Internet, as long as Web service-based access is possible.

### Object-oriented design

Object-oriented (OO) is a term more often used with programming languages than data schema or language encoding, but many of the same ideas apply. OO has dominated programming language design for the last twenty years, beginning with Smalltalk and gaining widespread popularity with C++, Java, and

most recently, C#. There are many benefits to OO design, but the most important to this effort is inheritance, which is the idea that an object can inherit functionality from another. This makes OO languages easier to understand, extend and program. For example, Figure 4 defines a GMLData (Geography Markup Language) data object. The reader of my model may have no idea what GML is, but because the GML object inherits from (is a child of) the abstract GeoData object, and the reader understands the properties of geographic data in general, most of the meaning is still conveyed. More importantly, a model can be constructed that specifies a requirement for geographic data input via a GeoData object without being concerned about the actual data source, which may come from a Shapefile on disk (ShapefileData), a GML file (GMLData), or some other source.

**Figure 4: Geographic data object hierarchy**




## Design Patterns

Design patterns are ways to describe, at an abstract level, problems or situations that occur frequently. Christopher Alexander, speaking in an architectural context, describes their purpose saying, "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of a solution to that problem, in such a way that you can use this solution a million times over, without ever doing the same thing twice." The design pattern concept applies to decisions involved in designing the modeling language, but it also applies to the use of the language in practice. Patterns for solving planning problems can be captured, shared and applied in varied places and contexts.

The most pervasive pattern driving the encoding of the language is that components of a model are models themselves. This concept follows the **Composite design pattern**, which creates objects in such a way that different types of objects, as well as combinations of those objects, can be acted upon in the same way. The importance of this design decision has implications throughout the creation and use of models. For example, every model has three basic features: data is input to the model; operations occur on that data; and data is an output from the model. This means that very complex models may be grouped together as one and described to a particular audience as a very simple, monolithic analysis. Yet details are readily available to more technical audiences by examining the model's constituent parts. Another important benefit is extensibility. Even when the operations that occur are yet to be described by this proposed language,

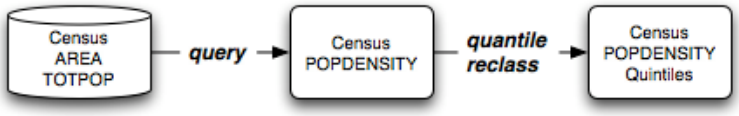
the analysis may still be included in, and useful to, a larger model that relies on it. For example, PAMML does not yet encode population growth forecasting models. However, the geographic data set output from the growth forecast can be used in a PAMML model.

## A DEMOGRAPHIC MODELING SCENARIO

Perhaps the best way to explain the full importance of this exercise is through an example scenario. Using Census data to understand demographics is one of the most basic planning analyses. Yet this seemingly simple task is executed with varying levels of quality, sophistication and depth of understanding. This scenario encodes a sophisticated analysis of population density into the PAMML and discusses how the model facilitates use and re-use of the technique.

Two analyses are presented in this example. In the first a very simple study is performed, calculating population density as the total population of a Census block group divided by its total area. Figure 5 diagrams the analysis. It shows raw Census data being queried to calculate a new attribute called POPDENSITY by dividing total population by area. Then a reclassification is performed to group the block groups into five classes, each having the same number of members—a quintile classification scheme.

**Figure 5: Flow of a simple density analysis**




Here are the same analysis steps presented in PAMML notation. For the first step, accessing Census 2000 population data for Cambridge, MA block groups, the PAMML might look like this:

**Figure 6: PAMML for Census data**

```
<ShapefileModel name="cambridgecensusbg"
    shpfile="file://camcen2kbg.shp"
    dbffile="file://camcen2kbg.dbf"
    shxfile="file://camcen2kbg.shx">
    <Meta>
        <Description>Cambridge, MA Census 2000 blockgroups</Description>
    </Meta>
    <AttributeInfo>
        <Attribute name="totpop" datatype="xs:int" maxval="44444" minval="0"/>
        <Attribute name="area" datatype="xs:decimal" maxval="44444" minval="0"/>
    </AttributeInfo>
</ShapefileModel>
```

There are two major components to the ShapefileModel. The most important is the location of the data, which is specified here with `shpfile`, `dbffile`, and `shxfile` attributes. The second is the listing of non-spatial attributes. These do not necessarily

represent all the attributes present in the data, only those that the data model builder chooses to expose to others. The PAMML is a notation language, so it can not on its own prevent users from accessing all the attributes in the Shapefile, but one can imagine a data server that mediates between users and data, using the model's attribute list to enforce these rules. Finally, note that some Attributes have `minval` and `maxval` items. This is optional information that makes building models out of other models easier.

The second step is to calculate population density:

**Figure 7: PAMML for query**

```
<SpatialQueryVModel name="cambridgecensuspopdensity">
    <AttributeInfo>
        <Attribute name="uniqueid" datatype="xs:int"/>
        <Attribute name="popdensity" datatype="xs:decimal" query="totpop div area"/>
    </AttributeInfo>
    <Meta>
        <Description>Computes a population density attribute from Cambridge, MA Census 2000
blockgroups</Description>
    </Meta>
    <ShapefileModel name="cambridgecensusdata">
        <RemoteInfo uri="file://cambridgecensusbg.xml"/>
    </ShapefileModel>
</SpatialQueryVModel>
```

This model simply allows new attributes to be added to a spatial data set. The input is our ShapefileModel. It is specified via a reference to the first model, which contains the actual references to the Shapefile data. The output is a new spatial data set with two Attributes, `uniqueid`, which is copied from the ShapefileModel, and `popdensity`, which is the result of dividing the ShapefileModel's `totpop` Attribute by its area (notice that this model only uses the Attributes specified in the ShapefileModel). These two models could have been combined into one, but it is important that the most basic data model does not assume an execution environment more complex than the simple ability to read the data. The density calculation requires an execution environment that can do math, which is already outside the capabilities of standard Web servers.

The final step is to classify population density into meaningful groups. A quintile classification scheme is used, and the model looks like this:

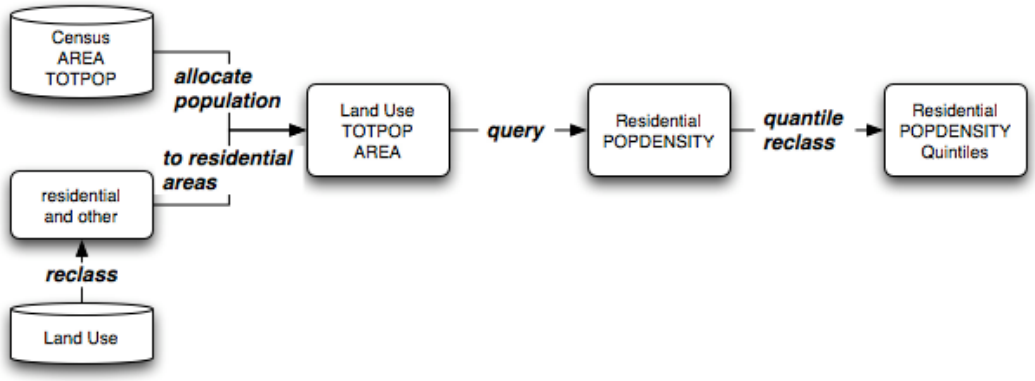
**Figure 8: PAMML for quintile classification**

```
<SpatialQuantileVModel name="Basic Population Density by quintile"
        usefeaturetype="popdensity"
        numranges="5">
    <AttributeInfo>
        <Attribute name="popdensityquintile" datatype="xs:decimal"/>
        <Attribute name="uniqueid" datatype="xs:int"/>
    </AttributeInfo>
    <Meta>
        <Description>Classified population density</Description>
    </Meta>
    <VectorDataModel name="cambridgecensuspopdensity">
        <RemoteInfo uri="file:///./cambma_popdensity.xml"/>
    </VectorDataModel>
</SpatialQuantileVModel>
```

The important parameters in this model are `usefeaturetype`, which specifies the Attribute to classify, and `numranges`, which specifies how many groups to create. Also notice that in this case a *VectorDataModel* is the input. A *SpatialQueryVModel* is a type of *VectorDataModel* (so is the *ShapefileModel*). There is no need here to know that the input data originated as a Shapefile, only that it represents spatial vector data, so the more general model type may be used.

Most planners with some amount of GIS training can perform the above analysis. It is not very accurate, however. Census analyses of this kind present a skewed portrait of where people live because it does not account for the fact that people do not live in office parks, forests, lakes, etc. A much better analysis would try to account for these obvious biases. The next analysis uses land use data to constrain Census population counts to areas defined as residential. Notice in Figure 9 that the last three steps are the same as in the above analysis. The only difference is the input data set is residential land use with a `TOTPOP` attribute instead of the raw Census block groups.

**Figure 9: Land use sensitive population density analysis**




The model starts with a land use data set that is described similarly to the Census model shown in Figure 6. This is combined with a table describing land use codes that for brevity will not be shown here, but it specifies that 'R0', 'R1', and 'R2' are residential land

use codes. The model in Figure 7 is used to group the land use data set into two categories, residential ('R') and other ('X'). The data is then "dissolved" using this attribute so that all adjacent polygons having the same land use code are merged. Note that this example nests a model "in-line," instead of referring to it via `RemoteInfo` notation.

After performing this task and then proportionally allocating Census population counts to residential areas (not shown in PAMML), the rest of the notation follows the first example, calculating a density attribute and classifying the values into quintiles.

**Figure 10: PAMML for spatial reclassification and dissolve**

```
<SpatialDissolveVModel name="cambridgeresidentiallanduse"
      usefeaturetype="res_lu">
   <AttributeInfo>
      <Attribute name="id" datatype="xs:int"/>
      <Attribute name="res_lu" datatype="xs:string"/>
      <Attribute name="area" datatype="xs:decimal"/>
   </AttributeInfo>
   <Meta>
      <Description>Residential land uses in Cambridge, MA</Description>
   </Meta>
   <SpatialReclassVModel name="cambridgeresluclasses">
      <AttributeInfo>
         <Attribute name="cambridgelanduse_id" datatype="xs:int"/>
         <Attribute name="res_lu" datatype="xs:string"/>
         <Attribute name="area" datatype="xs:decimal"/>
      </AttributeInfo>
      <Meta>
         <Description>Cambridge, MA 1:25000 land use</Description>
      </Meta>
      <ShapefileModel name="cambridgelanduse">
         <RemoteInfo uri="file:///cambma_lu.xml"/>
      </ShapefileModel>
      <ReclassTable name="landuse" joinfeature="lu21_code">
         <AttributeInfo>
            <Attribute name="lu21_code" datatype="xs:int"/>
            <Attribute name="residential" datatype="xs:string"/>
         </AttributeInfo>
         <table>
            <tr><att>R0</att><att>R</att></tr>
            <tr><att>R1</att><att>R</att></tr>
            <tr><att>R2</att><att>R</att></tr>
            <tr><att>*</att><att>X</att></tr>
         </table>
         <Meta>
            <Description>
               Changes land use to R (residential) and other (X)
            </Description>
         </Meta>
      </ReclassTable>
   </SpatialReclassVModel>
</SpatialDissolveVModel>
```

This may seem like an overwhelmingly complex description of a demographic analysis. In a sense, it is, and rightly so because it is a complex analysis, and a narrative description would be no shorter. This is, however, the wrong way to evaluate the language. In the early evolution of a machine language, the raw notation is usually written by hand, but as usage increases, tools are built to shield casual users from the complexity of the notation. In the second example, only five pieces of information are required from the user (and only the first two are needed for the simpler analysis presented previously):

1. The location of the Census data set

2. The Census total population and area fields
3. The location of the land use data set
4. The land use data's land use code field
5. The residential types of land use code

One can easily envision a graphical interface that makes obtaining this information from the user no more difficult than filling out forms on a Web page. Before these case-specific values are filled in, the model may be thought of as a template, or a design pattern. This term was used earlier in a software engineering context, where it helped to make decisions about how to design the language. It is also useful at a higher level, where we can think of spatial "design patterns" and planning "design patterns"—best practices models of how to extract certain kinds of knowledge from information.

## Developing internal consistency with design patterns

Design patterns are generic problem-solving models. They are the building blocks of analysis, like mathematical theorems. In fact, mathematical equations are the best-known formal design patterns. For example, calculus teaches us that rate of change can be determined by taking the first order derivative of the equation describing the phenomena. In statistics, we study the distribution of a sample by examining its variance and standard deviation. In the planning profession, we rarely use such formal patterns, but when we do, their power is overwhelming. Zoning is a good example. The building rules articulated in a town's zoning code form a specific, and therefore very powerful pattern of how to develop land. Many planners feel that traditional Euclidean zoning is a flawed development strategy, but its rules continue to dominate the American landscape. I would argue that the closer a planning theory gets to a design pattern, the more it is used in practice, mainly because planning theorists do not build cities, engineers, developers and elected officials do. And laws, which are simply a democratic definition of pattern-based rules, govern the activities of these people. PAMML will help connect planning theory to analytic models and computational execution.

Planning design patterns currently are less formal. If you ask a planner how to site a new subdivision, a whole range of analyses will come to mind, involving land suitability, services provisioning, congestion, etc. While the solution to the subdivision siting problem is constrained somewhat by a person's professional education and practice (i.e. most planners can agree upon the validity of a number of subdivision siting methodologies), we do not build our analyses upon formal models, which are in turn constructed on a foundation of proofs and theorems.

The flow diagrams in Figure 5 and Figure 9 hint at an approach to presenting analysis from a pattern-based perspective, where the actions (lines) along with their inputs and outputs represent a unit of operation. But, by its nature as an XML Schema-based language, PAMML requires much more precision in its vocabulary. The two figures below illustrate the beginnings of a pattern language for generic spatial analysis (Figure 11), and a translation of that into a pattern language using the vocabulary of planning (Figure 12).

**Figure 11:**
**Generic design pattern for analyzing intensity distributions**

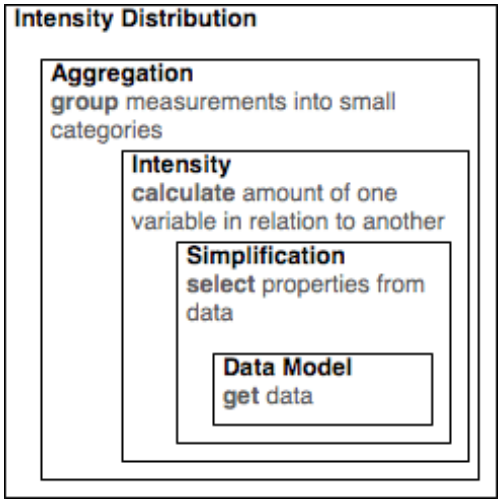
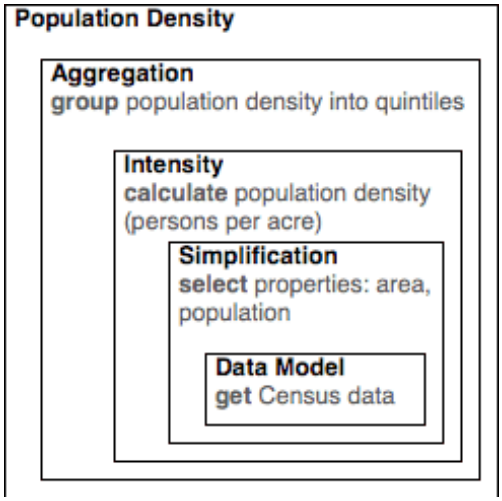



**Figure 12:**
**Population density modeling as an intensity distribution design pattern**




These analyses, obviously, are not novel, and neither is the idea of design patterns in planning, GIS, or computing. What is truly new is the ability to connect these powerful problem-solving modes from different disciplines in a way that enhances them all, facilitating more and higher quality analysis and driving intuitive user-interface design for the creation and presentation of PAMML-described models.

## FINAL THOUGHTS ON SIGNIFICANCE TO THE PLANNING PROFESSION

This paper has presented an overview of an XML vocabulary for describing analyses commonly performed by planners. It has also given one example of PAMML's applicability to population density modeling. This, however, has been an extremely limited introduction to the potential uses of the language. I would like to end with a description of two application areas that I am particularly interested in studying.

### Community Indicators and performance measurement efforts

More than anything else, this work aims to change the paradigm of analysis from the current one-time major effort to produce a document to many small efforts that produce a continuous information flow. As Lew Hopkins states, "Making plans for urban development is something you do constantly, not once" (Hopkins, 1999). The underlying assumptions that go into a plan, such as economic conditions and development activity constantly change, yet most plans are static. This is a necessary compromise based on the cost of marshalling the resources required to prepare useful plans. The framework suggested here allows plans to become dynamic tools—more like monitoring and early warning instruments than traditional plans. This may sound threatening to those who consider plans to be embodiments of a community's vision about their place, but Hopkins notes that plans are really the strategic implementation of visions, not the visions themselves. In this new type of plan, the community's vision is still present. It simply

manifests itself in a different form, such as the point at which development triggers a moratorium or an infrastructure investment.

This paradigm implies that the goal of analysis and modeling should change from report generation to situation monitoring and performance measurement. The need to support this effort is evident in many places. The National Neighborhood Indicators Partnership is an effort to build "advanced information systems with integrated and recurrently updated information on neighborhood conditions in their cities (http://www.urban.org/nnip/concept.html)." This is the most explicit example of this change in focus, but the trend presents itself in many other places. The Heinz Center's *Report on the State of the Nation's Ecosystems* (2002) recommends that environmental quality be monitored and reported in a consistent, constant way, in the manner of well-known federal economic indicators such as durable goods orders, housing production, consumer spending, etc. Indirectly related efforts include local government efforts to define a strategy for integrating the Internet into their mission. The National Civic League addresses this in the 8th revision of their Model City Charter. A joint project of the National Association of Counties and the National League of Cities seeks to support the ability of towns to automate government transactions over the Web through their "Totally Web Government" program. Literature in this area will help define requirements for the collaborative aspects of the language.

## Design Patterns for Urban Design

One significant implication of this work is the potential to create, articulate and implement urban design patterns using PAMML. The field provides us with a rich history of design patterns that are beautifully articulated, but are difficult to apply outside the designer's original context. Looking back at Olmstead's emerald necklaces of the 19th century, that designer was able to implement his vision of how to integrate green space into a metropolis, but despite the popularity of the idea and the implementation, there have been few emerald necklaces designed after the Olmstead era. The same can be said of Kevin Lynch's "good city form," Christopher Alexander's "pattern language," or Alan Jacobs' "great streets". The argument can be applied to the more popular urban models in recent times, such as Landis' California Urban Futures or Bill Hillier's Space Syntax. The expertise of these models must be captured in a way that allows their rules to be applied in different scenarios and physical contexts.

Kevin Lynch, in *The Image of the City* (Lynch, 1960), argues that planners must understand the city as a place whose function is to create an environment over which residents can take emotional ownership. This vision of the city contains familiar, private home neighborhoods, lively commercial and civic centers, and navigable and easily identified paths between all of these districts. These building blocks of the urban environment are called nodes, paths, districts, edges and landmarks. In his words:

> Nodes are points, the strategic spots in a city into which an observer can enter, and which are the intensive foci to and from which he is traveling. They may be primarily junctions, places of a break in transportation, a crossing or convergence of paths, moments of shift from

> one structure to another. Or the nodes may be simply concentrations, which gain their importance from being the condensation of some use or physical character, as a street-corner hangout or an enclosed square. Some of these concentration nodes are the focus and epitome of a district, over which their influence radiates and of which they stand as a symbol.
>
> *The Image of the City*, pp. 47-48

While the theories Lynch proposed in Image of the City are highly respected by urban designers, I have never seen them used in any U.S. town's comprehensive planning process. I have argued (Singh, 1999) that this is probably because the model as presented by Lynch requires an army of planners to collect the material for an Image of the City-style study. In 1999, I proposed a model for computing nodes based on detailed land use data and map-algebra style spatial analysis. Describing this model, and others like it, is a focus of future work with PAMML. Hopefully this can help bring urban design theory into planning practice.

## REFERENCES

Alberti, M. 1999. "Modeling the urban ecosystem: a conceptual framework" *Environment and Planning B: Planning & Design*.

Alexander, C., Ishikawa, S. and Silverstein, M. 1977. *A pattern language: towns, buildings and construction*.

Guhathakurta, Subhrajit. 2002. "Urban modeling as storytelling: using simulation models as a narrative" *Environment and Planning B: Planning & Design*.

Heinz, Center for Science, Economics and the Environment. 2002. *The state of the nation's ecosystems: measuring the lands, waters and living resources of the United States*.

Hopkins, Lew. 2001. *Urban Development: the logic of making plans*.

Klosterman, Richard 1998. "Computer applications in planning" *Environment and Planning B: Planning & Design*.

Lee, Douglass B. 1973. "Requiem for large scale models" *Journal of the American Institute of Planners* **39** 163-178.

Lynch, Kevin. 1960. *The Image of the City*.

McHarg, Ian L. 1969. *Design with Nature*.

Muller, P. 2000. *Instant UML*.

OpenGIS Consortium. 2002. *Abstract Specification, Topic 12: The OpenGIS Service Architecture*, http://www.opengis.org/techno/abstract/02-112.pdf.

OpenGIS Consortium. 2001. *OpenGIS® Geography Markup Language (GML) Implementation Specification*, http://www.opengis.net/gml/01-029/GML2.pdf.

OpenGIS Consortium. 2002. *OpenGIS® Web Feature Service Implementation Specification*, http://www.opengis.org/techno/specs/02-058.pdf.

Singh, Raj. 1999. "Sketching the city: a GIS-based approach," *Environment and Planning B*.

Steinitz, Carl. 1995. "Design is a verb; Design is a noun" *Landscape Journal*.