Collaborative Urban Information Systems:
A Web Services approach

by

Raj R. Singh

AB Economics, Brown University, 1991
Master in City Planning, Massachusetts Institute of Technology, 1996

Submitted to the Department of Urban Studies and Planning
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Urban Information Systems and Planning

at the

Massachusetts Institute of Technology

September 2004

Signature of Author _____

Raj R. Singh
September 14, 2004

Certified by _____

Joseph Ferreira, Jr.
Professor of Urban Planning and Operations Research
Head, Urban Information Systems Group

Accepted by _____

Frank Levy
Daniel Rose Professor of Urban Economics
Chair, Ph.D. Committee

Collaborative Urban Information Systems:
A Web Services approach

by

Raj R. Singh

Submitted to the Department of Urban Studies and Planning
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Information Systems in Planning

Abstract

This thesis examines systemic problems with the way information is managed and processed in planning support systems. We find evidence of these problems when we attempt to: develop an analysis without spending most of the time gathering and organizing data sets; or build an analysis that can be re-run at low cost; or implement systems that interact collaboratively with those of other experts. This research starts with the hypothesis that these problems are related and systemic, and that a new paradigm of information management is needed if we can hope to address them effectively.

The research is divided into two main sections. First, we develop a theory about how information flows within and across planning organizations, and use the MassGIS buildout analysis to understand how physical planning is done in a cross-jurisdictional, real-world setting. We find that modern organizations do are good at creating and disseminating information, but find it difficult to keep users' copies of published information up-to-date. Furthermore, the technology for building interactive front-ends to analytic models is poorly matched to user needs, and the technology for enabling cross-organization collaborative analysis is non-existent.

In the second part of the thesis, we re-architect the information framework, guided by our new theoretical foundation and findings from practice. This new framework is based on Web services, an emerging technology for connecting information systems across organizations. It is called the Planning Analysis and Modeling Markup Language framework, or PAMML, consisting of an information processing vocabulary expressed in XML Schema, Web services based on the schema, and guidance on how to best use the framework to encourage the interconnection of planning and mainstream information technology.

We find that the PAMML framework can lower costs by leveraging mainstream technology, simplify the most basic data sharing activities, yet still allow organizations with different levels of technical sophistication to collaborate. PAMML captures the semantics of spatial planning problems, allowing them to be decomposed into fundamental information processing operations. Regarding user interfaces, we show that PAMML's structure allows multiple end user applications aimed towards different audiences to be easily built from the same core PAMML document.

Thesis Supervisor: Joseph Ferreira, Jr.

Title: Professor of Urban Planning and Operations Research

Thesis Readers: Lewis Hopkins, Lorlene Hoyt, and Ceasar McDowell

Acknowledgements

A piece of work like this is more the culmination of a stage in one's career than a research project, and as such there are quite a few people to thank. First, thanks to Joe Ferreira for his mentorship and fostering my academic growth for more years than I care to mention. To all the folks at MassGIS, with whom I've worked with almost as long, thanks for being a terrific partner in technology innovation. In trying to figure out where we're going with this technology stuff and why, my old friends at Syncline, and the software community at the OpenGIS Consortium, played a key role.

Thanks also go out to those who helped me turn a bunch of interesting ideas into a coherent thesis, professors Lorlene Hoyt, Lew Hopkins and Ceasar McDowell; and those who helped spur my initial interest in technology in planning, Mike Shiffer and Qing Shen. I'd like to also remember Aaron Fleisher, who was always such a wonderful sounding board for ideas.

And finally, to my wife Jennifer. Your support makes everything possible.

Table of contents

Chapter 1. Introduction

The way planners gather and analyze information is archaic and extremely expensive. It has been this way for decades, and unlike other industries, it does not seem to be getting cheaper. Until the planning information systems community addresses this problem in a holistic manner, data-based analysis will never fulfill its potential to inform urban planning.

Few ever dispute the common folklore that 80-percent of any analysis effort is spent gathering data, leaving 20-percent of one's resources for the actual work that needs to get done. This is not a new realization. At least ten years ago we believed that the rapid increase in GIS adoption, and the ubiquity of data in electronic form, would lead to lower data sharing costs (Obermeyer and Perloff 1994). However, there is no evidence that this occurred. In fairness, the quality and quantity of the data brought to bear on a problem has improved dramatically, but the most important factor—the *relevance* of analysis in the decision making process, leaves much to be desired. The reason is simple. Analysis is not *timely*. For example, if it takes years to assemble the data for a large environmental impact study, is it not likely that the analysis will be irrelevant before it is presented? And once this analysis is put in front of decision makers, how extensive is

their ability to provide feedback? Can someone propose, for example, an alternative economic strategy based on tourism instead of riverboat gambling and generate a new, 1,000-page report? Or is stakeholder feedback relegated to meetings and minutes, never being explicitly linked to the numbers it discusses?

This thesis argues that our data gathering practices are broken, and are not likely to improve until significant structural changes are made to urban information management systems. The traditional areas in which we focus our research—data modeling, analysis, and visualization—are developed far beyond the capacity of practitioners to use them. The software that does such a good job with those tasks does little to facilitate basic data acquisition and processing. We have for too long overlooked the medieval data gathering practices common at all levels of government. Corporations have moved into the 21st century with integrated information systems that connect businesses with upstream and downstream trading partners, so that data is no longer re-processed when it moves from one organization to another. The planning community, on the other hand, still operates like traders at a bazaar, making deals, bartering, mixing and matching the data sources that form the foundation of our analytic systems.

Improving the flow of data between and within organizations is the next great challenge for planning support systems (PSS). With the sheer quantity of information sources available to planners increasing every year, and the dramatic technology investments made in the late 1990s, this is an especially important time to re-examine the ability of information technology to inform decision making in planning. What we really must do is re-evaluate what it means to be in the practice of creating planning support systems. It does not mean to combine theory, data and a methodology into *a plan* of

action for a specific place and time. It means to create systems that provide stakeholders with the ability to *continuously make plans* (Hopkins 1999), have them pre-empted by others' actions, and re-plan based on the new conditions. A system that did this would truly aid the decision making process and completely change the debate around how information and specialists are used in the planning process.

Motivation and Background

Over the last ten years, few technologies have captured the interest and energy of information technology professionals like XML[1] and Web services. Recently the fruits of this investment have been seen in public-facing applications like new interfaces to the databases of Google and Amazon. But perhaps more important are the XML and Web service-driven applications buried in the corporate back-office IT infrastructure, seamlessly connecting them with their business partners, and allowing them to achieve operational efficiencies that were barely imaginable in the 1980s. This is how Amazon can sell you a used book from a small, independent bookstore in Allentown, PA for two dollars and still make a profit. This is how Wal-Mart can continuously adjust their prices and inventories to meet changing supply and demand and respond to the vagaries of consumer preference. What can planners learn from Amazon? That question is central to this thesis.

---

[1] All acronyms are defined in Appendix B.

Leveraging important technology trends

The research agenda of this paper is inextricably linked to a number of fundamental changes happening in how government collects, stores and distributes data, and how Internet-aware software is built. While planning cannot adopt corporate technology wholesale, we do not have the financial resources to develop our own basic technologies from scratch, like the military industry. This puts us in the precarious position of strategically choosing which technologies to adopt from other fields, and which ones we should develop ourselves. I list here some of the trends I believe PSS must follow and adopt to be successful in the next few decades.

An urban information explosion. There is more to solving planning issues than simply obtaining the right data, but it is certainly fair to say that information plays a key role in an effective planning support system. What exactly is this role? How do we conceptualize our information processing requirements? These issues are more important than ever as we enter an era where almost every device will have the capacity to contribute to the city's information undercurrent. The new standard for Internet addressing, IPv6, was created to greatly increase the number of IP addresses available in response to industry's desire to give unique Internet IDs to devices other than full-fledged computers. This standard is already in place and in use. Wireless Internet access is becoming increasingly common and is beginning to play a role in public sector computing (*Muniwireless* 2004). Hardware for wireless Internet access is less than $10 as of June, 2004. These three trends taken together make it probable that even low-cost devices such as phones, cameras, buses, watches, traffic sensors, air quality monitors, etc.

will be Internet-aware and addressable in the near future, leading to an exponential increase the quantity of information available about the urban landscape.

Geographic data sharing and systems interoperability. Efforts to standardize the way in which we describe geographic features are critical to our ability to share government data between different departments, levels of government, and commercial and educational institutions. For example, if all municipalities called parcels by the same name and used the same terminology—and meaning—for a parcel's attributes, the cost of regional planning and administrative operations would be greatly reduced. In Europe, the problem has been less acute as most data collection occurs at the federal level. Therefore, work in this area is mainly happening in North America, where there is a strong tradition of local independence from federal control. The U.S Federal Geographic Data Committee and ESRI have strong programs in place to promote a common description of the most basic data sets used in government.

Of equal importance is the ability to locate and ingest another party's data with little or no human intervention in the conversion process. This is *systems interoperability*. The OpenGIS Consortium's standards for geographic data encoding (GML/Geography Markup Language), geographic data publishing (WFS/Web Feature Service), and map publishing (WMS/Web Mapping Service) are being well received in the industry and provide one of the foundations upon which this work depends.

XML. Arguably the most disruptive technology since the advent of the World Wide Web is Extensible Markup Language, or XML. XML is really nothing by itself. It is simply a framework in which to write highly structured languages for describing things and passing messages between computers. 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.

Web services. "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).

Whether or not XML is better than other technologies, the software industry has quickly supported it, building powerful, reliable tools to read XML and develop Web Services on every operating system and application in common use. Perhaps the

strongest sign of XML's importance is that Microsoft, which does not have a strong reputation for encouraging interoperability with others' applications, has decided to base their enterprise development software on Web Services, and has changed the native file formats of Office documents to XML.

Reflecting on the science of GIS

How do we explain geographical phenomena through the application of appropriate methods of analysis, and models of physical and human processes? Under what circumstances is the scientist willing to trust data that he or she did not collect, and will the increased technological ability to share scientific data over the Internet...change them? Such questions about tools often have their roots in theoretical questions about appropriate representations, operations, and concepts.

—Goodchild, et al., IJGIS 1999

These fundamental questions are posed in a 1999 article co-authored by many of the elder states-people of the field, including Mike Goodchild, Max Egenhofer and Karen Kemp. One might suppose that thirty years into the evolution of GIS these issues would have been discussed in great depth. Yet the article introduces an initiative funded by the National Science Foundation, Project Varenius, which seeks to build the theoretical foundation of geographic information sciences that was neglected during decades of practice-oriented work.

This project, while concretely grounded in a prototype implementation, fits well into the research agenda expressed by Project Varenius. It provides a set of circumstances under which scientists (and engineers and planners) can share data and collaborate on analysis. We do not hope to provide the definitive solution—that will take years of work

by our community of researchers. The main goal is to encourage the field to step back and address fundamental, broad-based data management problems that must not be left to the fields of management information and computer sciences.

An Organizational Theory of Planning Support Systems

The field of planning support systems is defined as, "a conception of integrated systems of information and software which bring the three components of traditional decision support systems—information, models, and visualization—into the public realm" (Klosterman 1999). While Klosterman's three components have been well researched over the last two decades, work on *integration* has not received proper attention, especially in regard to the organizational setting through which information flows. This section discusses the dominant information management paradigms planners currently use, and the primary ways researchers have attempted to address shortcomings in the effectiveness of collaborative information systems. We see a mismatch between the problems we would like to solve, and the strategies employed to solve them, and we posit that this is why truly effective solutions have proven elusive. To address these systemic problems, it may be necessary to develop a technology strategy based upon a different theory of information sharing across organizations. This section develops such a theory, which informs the technology framework that is the topic of this work.

Dominant information management paradigms

The most basic information management paradigm is the ***single user*** system, where everyone manages their own copy of information for their own purposes. This strategy quickly falls apart in organizational settings, where productivity gains can be had by

centralizing data collection and management activities. This leads to a situation where data are in one place, and users are in many other places. This problem has been addressed using ***client-server*** information architectures. The central principle here is that data resides on a server, and multiple, heterogeneous clients all access a particular data set from that server. Over the past few decades this strategy has worked well. It fits (and perhaps has even influenced) the structure of many organizations, who try to centralize specialized activities like information technology in one department. Data producers are able to write, or publish, data into the centralized database server (data entry or publishing clients), and data users are able to read data out of the centralized system. There is no direct connection between data producers and users in this type of setup.

The client-server strategy is usually only employed within a single organization, because allowing users direct access to one's database is a potential security problem, and the system often requires some training and knowledge on the part of the user. In the 1990s, ***Web-based clients*** came into vogue. Data was more secure—database connection information was hidden from the user and buried in the Web server, and the database accessed through the Web was usually a duplicate, expendable version. Data usage and interpretation was also made simpler by using the increasingly familiar metaphor of the Web page for information presentation and manipulation. The security advantages of Web-based systems are clear, but the benefits of Web-based client software is less so. In the 1990s, when these technologies were being developed, users often had little experience with computing, so the Web strategy made sense. But in the near future, if not today, information users will have a sophisticated understanding of software user interfaces, and feel limited by the simplicity of Web-based clients. So while

Web-based clients have taught us a lot about addressing security concerns, Web pages may be reaching the limits of their usefulness as client software. Also, the Web has little to tell us about collaboration. The client-server paradigm has not changed, so there is no reason to expect the traditional Web server-Web page architecture to lead to revolutionary advances in information management and collaboration.

Geographic information sharing research

There already exists a strong body of literature in the area of geographic information sharing. The traditional line of inquiry researchers often take is to examine existing organizations and their efforts at collaboration (Evans, 1997) in an attempt to understand why goals are not better met. The most general problem is that organizational settings are highly complex. When embarking on an information sharing project, many issues may arise, such as reluctance to share GIS files due to a fear of losing autonomy, control over information sources, independence, organizational power, cost, complex inter-organizational interdependencies, and politics (Nedovic-Budic and Pinto, 1999-2, 54). Solutions to these problems usually address the social, political and organizational problems using an existing technology, or at best a new technology within an existing paradigm. On the other hand, research in planning support systems is usually geared towards technology that advances the state of the art in one of Klosterman's three pillars, with no formal attention devoted to how the technology addresses organizational issues. By considering technology fixed, information sharing researchers are led to false conclusions. For example, it has been found that the information sharing success is found when the parties have aligned interests and work well together. What about those organizations who do not have well-aligned goals; do we expect them to

never collaborate successfully? Is this an acceptable situation in planning? If we can only expect to build successful information sharing systems in that type of environment, we can never expect to change the balance of the 80-20 data management-analysis split.

Positioning PSS in the Theory of the Firm

We believe that technology research can do more to aid information sharing than the current dominant information management paradigms allow. Above all other problems, the geographic information sharing research community identifies cost as the main barrier to successful projects. While some people express a desire to collaborate motivated by altruism and efficient government, the cost in time, resources, and money to one's own organization, in conjunction with the value derived, most often determines participation and long-term success (Nedovic-Budic and Pinto 1999-1). So then if economic concerns drive behavior, then traditional economic theory should have much to offer the urban planning field. From this perspective, we can restate the information sharing problem as one in which the costs of the system must be less than the benefits. We know that the costs of data management and sharing are high enough so that the literature advises us that the benefits must be very high to achieve successful outcomes. The goal of technology work in this area is therefore distilled to a simple principle. The lower the cost of participation in a system, the less an organization must benefit from participation. And as benefits increase, so can cost. A large state organization whose mandate is information delivery can spend a great deal of money to accomplish this goal. However, a small non-profit whose primary mission is economic development and housing has limited time, resources and interest to devote to the issue. Yet both these groups, and many in between, must be accommodated within the same framework if the

technology of information sharing is to address the entire community that planners must serve.

Many in our field are uncomfortable with comparing government to corporate operations because of their different goals and motivations. However, they are more similar than different, as private sector firms have information management demands (and standards) at least as high as state and local government. We can learn a great deal from the business literature if we simply agree that both private and public agencies are groups of people organized to accomplish certain tasks in a cost-effective manner. Such is the case whether the tasks performed are part of a beer advertising campaign or a journey-to-work study. This viewpoint is not novel. Our term of art, planning support systems (PSS), is a direct descendant of the corporate term, decision support systems (DSS), that came into vogue in the 1980s (Klosterman 1999), and most researchers have believed for a long time that GIS should ultimately be part of MIS (Obermeyer and Pinto 1994). So we have always looked to our larger corporate brethren for guidance on how to use information to our advantage. In upcoming chapters we update that strategy and seek to assimilate PSS into the mainstream of distributed information technology, but here we provide the theoretical foundation needed to choose the right technology.

The theory presented here is built up by first specifying a strict definition of the types of roles information plays in planning support systems. Then, we propose a way to approach the problems conceptually. The only assumption made is that organizational behavior is the lens through which these problems should be viewed. Other issues, such as technology, are secondary to this. By developing an understanding of the nature of planning-related information and the organizational behaviors we must encourage to

improve our systems, a framework for PSS information management can then be developed.

PSS from an information processing perspective

Most planning support systems are reticent to admit that their purpose is to quantify planning problems. Instead their proponents hedge, stating that they are no more than a platform for public debate. If the creators of these systems really felt that way, would they put so much thought into their methodology, and effort into data processing? Or is it rather the case that most analyses have such a short shelf life that their cost must be justified in some other way than their ability to provide answers? I would argue that it is the latter, and whether it is called an *answer*, or a *model*, or a simplification of a complex system, anyone working in the field of PSS must operate under the assumption that they are creating systems that process data into more easily comprehensible information to help people interpret a complex reality that is beyond the ability of any single individual, corporation or special interest group to understand.

Planning support systems do provide answers through a process that quantifies most inputs, but they are always going to be at an intermediate level. They are no substitute for *decisions*. Therefore the PSS primarily exists to process information in ways that make it easier for people to make decisions—to understand issues and engage in highly informed debate, ideally in a collaborative environment. This is not to say that the analysis and presentation of information is not important, just that those functions are well studied, and advanced far beyond our ability to populate them with useful data (in fact, if this work is successful, someone might be writing ten years from now that PSS should be seen as an information presentation tool, because they will take for granted the

richness of information available for presentation). But no analysis or presentation or public participation can happen without a rich warehouse of information upon which to work.

The complexity of modern cities contributed to the need for the profession of planning, so it should be apparent that the information systems planners use should help cope with this complexity. Although we are in an, "information rich era in which high volumes of data flow through ubiquitous communication networks (Evans and Ferreira 1995), current practices are not able to make use of it, at least not in a cost-effective manner. In fact, organizations usually *resist* distributed processing efforts (Meredith 1995), leading to high project costs with little return. This problem will likely become even more noticeable as we try to take advantage of all the environmental sensing equipment embedded in the urban landscape, from security cameras to camera phones and location-tracked transit vehicles, the data sources we can and should incorporate into PSS will increase exponentially in the near future.

It is difficult to argue against the current systems, because the lack of any universal practice or system is more notable than anything else. How do planners manage data? Basically they acquire it, process it in some idiosyncratic way to get it into their database. While there are some standard software packages in use, and plenty of "best practices" available to cite, there are precious few *ubiquitous practices*. When practices become ubiquitous and generic enough that unrelated organizations can develop connections between their information systems, we have achieved *interoperability*. And that is the point of this work, to define the general, interoperable, practices that software packages must implement if we are to have any hope of making better use of the information available

today, and the immense increase in quantity and disparity of information that will be available tomorrow.

The primary raw material needed to create an analysis product is information. This information could be obtained by developing in-house data gathering capabilities, but the cost of that effort is beyond most organizations. Imagine sending teams of city planners (even if they *were* graduate students) out into the field to count traffic, go door to door asking people how much money they make, or how much they paid for their house. Why do this when organizations like the assessing department, the US Census Bureau, the Bureau of Labor Statistics, the realtor's Multiple Listing Service, and the actuarial databases of all kinds of insurance companies already have the information? It makes much more sense to form partnerships with these groups, and only develop custom data sets when absolutely necessary. For this reason, *the production of planning analysis depends upon inputs from multiple, disparate suppliers.*

So ultimately, to make use of a large body of data, it will be necessary to work with multiple, disparate suppliers. But it may be easier to start by looking only at the case of a municipal planning effort using solely municipal data sources. What is the private sector analog to a town? Is a town a firm or a conglomerate? This is where things get slightly complicated. Most of a town planner's information providers are other municipal agencies, such as property assessing, building permitting or zoning, so it is tempting to look at municipal government as one firm with different departments that support the development of different products, like tax bills, parking tickets, police officers, drinking water, etc. However, in practice a government bureaucracy operates more like a *multidivisional firm* than a company in the normal sense.

Multidivisional, or M-form, firms have many unique characteristics, but for our purposes the critical one is that the reward and decision making systems are constrained within a division, so that there is little incentive for one division to act in the best interests of another (Carlton and Perloff 1990). Some opportunities are lost this way, but at least the organization does not collapse under its own weight (Ba and Stallaert 2002). At the state and federal level, this probably makes a lot of sense because the information and coordination required to operate such large organizations is overwhelming, but municipalities may be emulating their larger relatives, without much thought paid to the reason. This theory suggests that one solution to this problem (and perhaps to many other problems) could be to institute more hierarchical forms of local and county government so that all divisions operate under a unified risk and reward structure. However, the task at hand is to redesign information systems, not government. So we will work within the given institutional parameters, which suggest that it is best to consider a local government as a multidivisional firm, and that it will stay that way in the future.

So town planners cannot count on other departments to act as partners in the creation of their product. In other words, the assessing department has little to gain from reducing costs in the planning department. We are left with a situation where, from an ownership perspective (either as a stockholder or taxpayer), we would like to see our government maximize production across all divisions (e.g. assess property values *and* undertake planning analysis). However, the organizational structure cannot change, and by the definition of a multidivisional firm, the highest levels of the firm are not provided with enough information to tell the divisions *exactly* what to do. This is a vexing problem,

and I believe it provides a good model of reality. In fact, this may be why we have so much trouble developing effective planning support systems—because we believe the government to operate like a single, unified firm.

When faced with the question of how to incorporate property value information into a planning support system, the PSS community has generally addressed the technical issues and assumed away the organizational ones. This would be fine if the organizational issues could be treated in isolation, but they are intertwined with technology. For example, since the 1990s, the trend in GIS has been to put data into an "enterprise" warehouse. "Enterprise" means that the data maintained by an organization (enterprise) resides in a centrally maintained database, with clients connecting over a network, and accessing those data sets a database administrator has granted them permission to use. This is fine as an intra-divisional solution, but the M-form theory suggests that enterprise databases find it difficult to cross divisions, and therefore enterprise solutions do little to address information management issues that cross divisional boundaries (Carlton and Perloff 1990). The theory is borne out by recent empirical data such as the following example. In a recent survey of 110 companies with revenue of at least $500 million, only 23% had their entire firm using one instance of ERP (enterprise resource planning) software.[2] And in one extreme example, as many as 400 different versions of a single vendor's ERP software were in use at a single, large company (Kock 2004).

---

[2] ERP is a term used to describe the process of managing an organization. The software usually keeps track of company-wide information regarding employees, facilities, etc. Unlike software used to achieve business objectives (like customer relationship management software), which might naturally be specialized for certain divisions or functions, one would expect enterprise resource planning operations to be easily centralized.

Examining local government from the perspective of an M-form, or multidivisional firm, provides some insight into past information management failings, but remember that a discussion of a municipality's relationship to other organizations was postponed. It is now quite easy to return to that issue, because our theory already considers different divisions as basically acting like different firms. Conceptualizing government as a multidivisional firm makes it easy to incorporate other levels of government, non-governmental entities, and even private firms. And later it will be shown that the theory suits not only the case when the analyst is a government entity, but the more realistic case when the analyst is a private entity working in loose collaboration with government, their own firm, and the public. There is no change required at the broadest theoretical level, although in practice minor differences will emerge—most likely around tighter data privacy requirements and perhaps higher costs and information licensing restrictions. While the differences between separate firms and different divisions within the same firm might be important in some ways, for the purposes of looking at how they share and process information, it is most useful to consider their relationship to be that of *trading partners.*

The way trading partners exchange information is by executing a contract. This is a tremendously important point. A contract is a specification of all the rules governing a business transaction between parties. A contract is needed when the parties doing business cannot count on each other to maximize performance without one (this is basically any time when the two parties have different bosses). The contract must anticipate and specify what happens in all possible scenarios, because if you could count

on the parties to behave properly in a situation not covered by the contract, the contract would not have been needed in the first place. While subcontracting and outsourcing continue to be cost-effective ways of doing business, this description of contracts begins to suggest how they can become quite expensive.

The cost of doing business with outside parties is addressed in a number of organizational behavior theories, most notably "Agency Theory" and "Transaction Cost Theory" (Vibert 2004). These theories help us decide when to outsource and when to keep a function in-house. Transaction costs have been identified as a key factor in geographic information sharing (Nedovic-Budic 1999), and being able to accurately predict these costs, and develop contractual agreements that govern the process, help ensure project success. Overall, cross-organization information sharing can achieve economies of scale, so planners should continue to outsource their data development needs, but these theories tell us that we still must put contracts in place. Even when cross-agency cooperation seems strong, tools like Memorandums of Understanding (MOUs) should be employed to ensure good results. What should these MOUs contain? This is where PSS research can inform policy. *No treatment of planning support systems is complete without attention paid to the rules by which information is transacted across agencies.* Either this policy work must be done for every PSS proposed, or the PSS must leverage a broader technology framework that has already accounted for these issues. This subtle interplay between technology and policy is a large part of the motivation for this work.

Framing the Issues from a Firm's Perspective

With a basic theory in place about PSS and its place in local government, we can begin to address the problems raised in the introduction. Planning analysis still has

relatively little influence on development when compared to highly deterministic tools like zoning and the transportation manual, which seem to single-handedly (along with developers' interpretation/manipulation of them) shape urban form. But we cannot seek to emulate zoning or engineering manuals. They are different kinds of tools. They are one generation downstream, offering patterns or heuristics to follow. We must operate upstream, providing the guidance by which these heuristics are created, or by which they are accepted or rejected at the time of decision making.

Data, data, data

In real estate, the three most important characteristics of a property are location, location and location. Planning analysts have a similar love affair, but with data. Urban environments have become such incredibly complex organisms that no single person or agency has enough knowledge to make responsible decisions. Instead we rely on a web of specialized disciplines to build and maintain the databases and analytic tools we bring to bear on planning problems. The cost of gathering and processing this data is arguably the most significant cost for planning analysis. In rare cases, one might undertake one's own data collection effort, such as a survey. But this only happens in research environments. The general case is one where practicing planners build their analysis around data that is readily available, and unless society develops the willingness to fund planning research (like we do in defense), this will continue to be the case. The point here is that planners are not data producers. We operate like traders at a bazaar, making deals, bartering, mixing and matching the data sources that build our analytic systems. I hope I have evoked a mental image of planners shuttling between medieval tents on muddy roads, because that is the state of civic information systems.

Information technology has certainly brought significant improvements to the speed and cost of creating and maintaining data, but we are still far from being good at bringing information to bear on a problem at the precise time when decisions are being made. In most urban information systems data comes from a variety of public and private sources, and can quickly became outdated. In the case of government agencies, whether at the federal, state, or local level, data is usually easy to acquire at any particular point in time, but difficult to keep current at all times. We often try to supplement government data with more current data from the private sector. For example, in urban growth studies, the most up to date source for new construction and land use is the developers building them. But there are no generally accepted best practices for integrating public and private data sources in a PSS, and without policy in place, we cannot expect anything more than *ad hoc* participation from the private sector.

Agency theory suggests that the data provision issues can be improved by having the concerned parties execute a contract specifying exactly the rules of engagement. This may sound simple, but this type of contract is rare. Most data sharing agreements do a good job of detailing what will be shared, but not how. This is probably because it is seen as going beyond the boundaries of politeness to tell another agency how to do their job. Yet who will tell them how to do it? In a multidivisional firm, we have learned that the "bosses" are prevented from having enough information to do this, so the appropriate rules must come out of a negotiation process between the interested parties; in other words, a contract.

So what should the contract say? One could image, for example, a program that could compare an old data set to a new one and make suitable updates. In this case, the

contract might say that a new data set will be provided whenever a change is made. If the data are large, however, this could be a very wasteful way of doing things. It would make more sense to write a piece of software that could receive messages telling it to update a particular data set in a particular way. In this case, the contract could specify that the data provider's software send messages to this new, smart piece of software. This puts a burden on the data provider, but the user could compensate them with the money saved from not having to do data updates any more. And the provider would be much more likely to agree if they could re-use the updating system (and the contract) with other users. Now the interplay between technology and policy should be becoming clear. In order for data users to solve their information management problems, they need to develop detailed, clear relationships with data providers. This clarity must be evident in contractual terms, public policy, and technological execution.

The timing of decision making

Planning support systems, like decision support systems, are tools. Their intent is not to produce maps and figures for annual reports, but to be ready partners in the process of decision making. Fulfilling this role requires that *PSS must be operated by stakeholders at the time when decisions are being made*. Current practice is for technicians to operate the system, and the usefulness of its results is usually tied closely to the time at which the data were acquired, or the analyses were run. In the Buildout analysis, we will see that MassGIS valiantly attempts to address this issue by providing online access to the analysis. This shows that they recognize the problem, but without up-to-date data, the fact that the analysis is available to a larger population does little to inform public debate.

At this point in the discussion, many studies of planning information systems tend to dive into the contentious arena of public participation and get diverted by issues of politics, power and class. "Rational," information-based decision making processes might be mentioned as a marginalized form of discourse, or even as a tool of the wealthy to erect a façade of objectivity around questionable decisions. This paper takes a slightly different position, and suggests that people's main motivation to use rational scientific analysis is honest; they genuinely believe in its power to inform good decisions. It is more productive to take the position that our community of information scientists provides the public realm with poor decision making tools. Our *analytic methodologies* are usually sound, but we have done little to adapt them to realistic decision making scenarios. Maybe the academic, prototyping environment in which our technologies are developed are to blame, or maybe there is some other cause, but systems that depend upon pre-prepared, static data sets have extremely limited value. And there seems to be a tacit understanding of this, leading to a general dissatisfaction with most urban information systems. Doing nothing but improving the timing of analysis would revolutionize the field, but achieving that goal requires the other changes discussed here as well.

The timing of expenditures

A finance expert will tell you that the predictability and non-volatility of an expense is just as important as its actual amount. "Lumpy" expenses are bad, because it is difficult to budget for them. The preference for stable receipts and payments can be seen in many facets of the economy. This is why companies are willing to pay more to lease equipment, and people can be driven bankrupt by an ill-timed job loss. Planners also can

ill afford *lumpy expenses*. We may go to our city council or governor and ask for a fourfold budget increase, just for the next couple of years to develop a twenty year master plan, but if conditions change, and in five years that plan is no longer valid, the money will not be there to re-do the work. A world that changes in complex ways at unpredictable times requires continuous planning and analysis. Yet the nature of public expenditures demands constancy and predictability. So the cost of performing planning analysis must reconcile these conflicting forces. The Buildout analysis suffers greatly from this problem. A great deal of good work is obsolete soon after it is complete. While policy may take the lead on this issue, any technology work in the field should also be aware of the importance of timing.

Research Question & Methods

How can planning take advantage of these cutting-edge technologies that are changing the corporate IT landscape? This question motivates the research presented here. Surely there are benefits to be had from the Web services paradigm of information flow, but do the benefits outweigh the costs of adoption? And as mentioned earlier, we must be smart about how much we adopt, and how we adapt technology to fit the needs of government and urban planning.

Proving that the future of urban information systems lies within an XML/Web Services information paradigm is a difficult task at best—there are few tools or precedents for proving the value of paradigm shift. Falling short of this, the best strategy is to position the field within a theoretical framework that helps explain why some issues

are successfully handled, and others remain intractable problems. Starting from the theoretical position put forth above, we make a strong case for Web services through hypothetical syllogism (Weston 1992, 51). We do this by showing that government organizational behavior is similar to corporate structures, and therefore that the solutions corporations employed to successfully address information management and collaboration issues can be employed in planning, which includes both governmental and private sector organizations.

Thesis Organization

This chapter introduced a vexing problem. We seem to be constantly progressing in our ability to capture, store and disseminate data, but our ability to manage and make efficient use of this information leaves much to be desired. The PSS literature focuses too heavily on the traditional specialties of data management, modeling and visualization, paying too little attention to their integration, or issues regarding implementation within an organization. On the other hand, the information sharing literature often takes technology as a given, and seeks to address information sharing and collaborative planning issues from an organizational behavior perspective. We propose a blended approach. The major points made here about how organizations collaborate are that *transaction costs* and the *chain of command* are important factors in the ability of organizations to function effectively. Executive managers must have very good information about the costs and benefits of different actions and outcomes if they hope to run their agency effectively. If an organization is too large (or inefficient) for executives to get the information needed to make these decisions, they must cede

decision making authority to lower levels. This makes small *divisions* effective, but magnifies cross-divisional problems—exactly the situation we observe in government today.

In Chapter 2, modern, Internet-centric, distributed information technologies are reviewed with a focus on how they address information management problems. Chapter 3 presents the Massachusetts buildout analysis, an urban growth model developed as part of the Community Preservation Initiative (CPI), an effort to better engage towns in planning for growth management and open space preservation. The CPI is interesting in itself, and is discussed in more detail elsewhere (Hodges 2004), but here we look only at the buildout analysis in its role as a practical tool with great potential, but limited usefulness, because it suffers from the problems predicted by the theories put forth in Chapters 1 and 2.

With this background, we are able to design a new framework for urban information management. Chapters 4 and 5 present solutions to common PSS requirements such as data sharing, participatory decision making, and expert collaboration. These solutions are expressed within a Web services framework, which uses a shared, formal, XML-based vocabulary called PAMML (Planning Analysis and Modeling Markup Language). The PAMML framework consists of a language in which abstract *data sharing*, *transformations* (arithmetic operations, format translations) and *public feedback loops* can be expressed, and a suite of Web *services* that allow organizations to advertise their ability to perform specific tasks, such as the transfer of a particular data set, or the execution of a particular spatial operation. The entire PAMML vocabulary is expressed in the XML Schema language, and is listed in Appendix A.

The theoretical argument is strengthened by reference to an example of a real planning support system being used in Massachusetts. Armed with a theory and an actual system that exhibits shortcomings common to its kind, the thesis presents a solution based on XML and Web Services. As stated, there is no definitive way to unequivocally *prove* the system's value, but it is hoped that the preponderance of evidence presented here should *convince* the reader that paradigm shift is worthwhile.

Chapter 2: Technology Frameworks for Information Sharing

Information-rich analysis efforts are characterized by their struggles with data preparation. This process can take months or years to complete (Waddell 2004), creating a situation where the "dirty little secret" of information analysis is that the majority of the time and effort is spent in data acquisition and formatting. The planning profession has generally ignored this problem, considering it a software issue which will improve with time and progress in the general field of information systems. This point of view seems reasonable, but much evidence suggests otherwise. If that is the case, it would seem that we would have observed significant improvements over the last few decades, but the results are mixed. We are digitizing less data, and using more data in our analyses, yet we continue to duplicate data development efforts, and we rarely implement systems whose data stays relevant from year to year. The problem is exacerbated by the fact that the organizations information moves between have different professional cultures, goals, and skills. Administrative divisions like property assessing have little in common culturally with the planning department, or a zoning board, or a local watershed protection group. These communities require their own methodologies for information processing, visualization and dissemination, and any proposal for improving information integration must not put restrictions on any organization's natural operational processes. A well-known concept in decision support is the idea that our systems should help people engage in the transformation of data into information into knowledge. Our current technologies have been good at providing decision support to individuals or

small groups using self-contained systems, but when the system is like most planning analyses, having multiple, heterogeneous participants in every area—from the creation of data, to the modeling, to the presentation of results—they break down under the operational costs of the information transactions.

This situation suggests that the root causes of our data dilemma are not in what information systems or data converters we happen to use, but in defining an overall *framework* for processing information. A framework is an extensible structure for describing a set of concepts, methods, technologies, and cultural changes necessary for a complete product design and manufacturing process (CERN 2004). It is more than a set of software recommendations, or even a new technology proposal, but all those things in conjunction with the cultural and institutional changes necessary to effect real progress. This chapter presents a technology framework in which we can reduce costs, while developing urban information systems that hold up to increasing demands from participants in data input (data), information development (modeling), and knowledge creation (visualization and public participation). First, the concept of a planning support system is positioned generically as a distributed computing environment. This allows planners to leverage the systems that computer scientists have created for distributed information processing instead of inventing our own technology baseline. While there are a few alternative technologies for doing distributed computing, a Web Services framework is chosen. This decision helps solve the next issue, which is to develop planning-specific decision support systems within the distributed computing environment. In a Web Services framework, domain-specific information models are

developed in a semantic meta-language like RDF or XML. While these tools have various pros and cons, Web Services software available today is designed to use XML, and the practicality of using RDF has yet to be shown. In the following chapters, we adopt the Web services framework, and use it to prototype a new urban information system based on data and analysis services. This is presented through a series of use cases relating to data publishing, urban modeling, and participatory GIS where case-specific solutions are developed. Finally, a full system is presented in Chapter 7, and the MassGIS buildout analysis is presented in this new framework. The XML vocabulary is called *Planning Analysis and Modeling Markup Language*, or PAMML, and the Web Services built on it are referred to as *PAMML services.*

An introduction to distributed computing

A distributed computing environment is one in which information and the applications that make use of it are physically located on different computers. In order for these computers to know that others of their kind exist, and how to talk to them, computers need a whole host of hardware and software. For the purposes of this work, we will assume that communication occurs via what is commonly called the Internet, which includes Ethernet and TCP/IP.

Figure 2-2:
Abstract 3-Tier Architecture

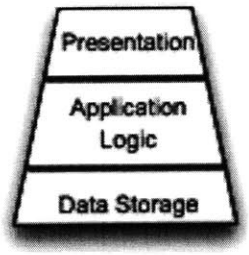
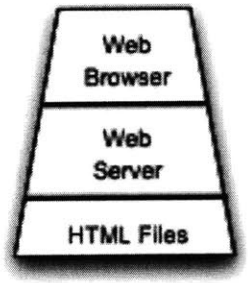


Figure 2-2:
Web 3-Tier Architecture



In this environment, an information warehouse is called a *resource*, and the system that provides information is generally called a *service*. So in this parlance, information, or data, is retrieved from a *resource* through interaction with a *service*. The agent that requests information—for example a person, a computer or a computer program—is called a *client*. What has just been described is usually called a "three-tier architecture" in computing. This architecture underlies most of the important systems in use today, including e-mail, instant messaging, and the World Wide Web.

In this architecture, any information store, such as a parcel database or an address book, becomes an abstract concept. The actual data can only be accessed by making a request to a service, which serves as the gatekeeper to the data. PAMML is a language that describes how to build services, so that different services can be expected to reliably interact with one another.

This architecture is quite complex and difficult to implement in practice, so why bother? The best answer is that distributed computing is flexible enough to mirror the organizational situations we encounter in the real world. For example, if everyone was required to have an email server on his or her computer and they could only read their email on that computer, it is doubtful that email would be in widespread use today. In government, our interest centers on the distributed nature of information and domain knowledge. For example, the assessing department uses parcel data more than any other

agency. Therefore, they are best able to make sure that parcel information is up to date and captures the knowledge about parcels required for municipal administration. The same applies to other domain experts, such as traffic engineers, natural resource managers, and infrastructure providers. Unlike most of these other organizations, planning practice is defined by the ability to integrate and analyze information from other domains. If successful planning outcomes were not so dependent upon having access to the right information, such close attention would not have to be paid to the information infrastructure of all the professions involved in collection information about places.

The IT world offers various solutions for implementing distributed computing applications. EDI, or electronic data interchange, is decades old and has been favored by organizations with high security and reliability needs like banks and airlines. While the technology is proven, participation in an EDI system requires a great deal of programming and system administration skills, which would eliminate the potential participation of most local governments and non-profits.

In the early 1990s a system called CORBA became popular. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network. CORBA has been widely used to connect corporate information systems, and is getting some attention in the GIS field (Preston, Clayton and Wells 2003). A full analysis of this is beyond the scope of this

paper, but in general, CORBA seems to be too "tightly coupled", requiring too high a level of coordination and cooperation between agencies, despite its language and operating system independence (Gottschalk 2000).

Web Services

As personal computing and the World Wide Web gained popularity in the 1990s, the IT landscape changed. Information sharing and processing was no longer the sole purview of big corporations. There was suddenly a vision of all organizations and individuals participating in a global information community. The old systems were not offering answers to these new challenges, so computer scientists looked at the Web and tried to understand why it had been so successful. It was found that the Web architecture requires only a minimal set of standards—HTTP as the basic application level protocol, and HTML for formatting information—but it delivers the ability to communicate without centralized planning or control, and to integrate a heterogeneous mix of platforms and programming models (Curbera 2001). The result is a very shallow interaction model between a very heterogeneous set of clients and servers that allows simple things, like sending a text file to someone's computer, to be easy; and complicated things, like buying a book with a credit card, to be possible.

The Web still has many limitations. HTML was designed as a way to mark up text for display, and HTTP is best at handling communications between only two computers at a time. In order to improve upon the quality of information available on the Web, and the systems that enable multi-computer, multi-organization transactions, something

more was needed. XML, the successor to HTML, and Web Services, a descendant of EDI and CORBA built on Web standards, address these needs.

XML and XML Schema defined

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 extensively used to formally describing information that does not fit nicely into the relational database paradigm. What XML provides is a consistent structure and a way of formally describing a language's vocabulary. The World Wide Web Consortium defines XML's design goals as follows (World Wide Web Consortium 2004):

1. XML shall be straightforwardly usable over the Internet.

2. XML shall support a wide variety of applications.

3. XML shall be compatible with SGML.

4. It shall be easy to write programs which process XML documents.

5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

6. XML documents should be human-legible and reasonably clear.

7. The XML design should be prepared quickly.

8. The design of XML shall be formal and concise.

9. XML documents shall be easy to create.

10. Terseness in XML markup is of minimal importance.

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, field 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 can use this mechanism to avoid re-

inventing the wheel in the areas of networking, identity management, databases, and GIS. For example, whenever a PAMML document needs to reference to a resource located somewhere on the Internet, the World Wide Web Consortium's (W3C) XLink vocabulary can be used to identify the resource. Database access may take advantage of W3C's evolving XQuery vocabulary. In the geographic information systems 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.

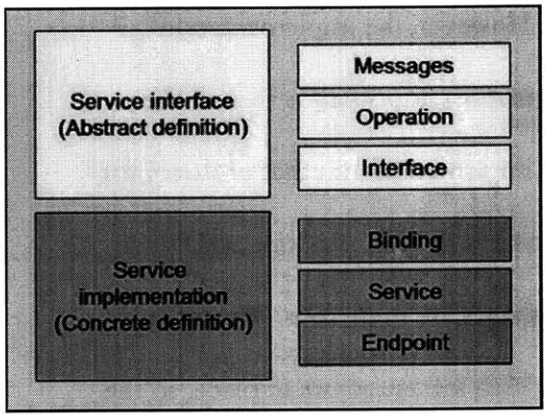
Web services defined

"Web services" is an umbrella term used to describe systems that allow computer software to communicate 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 the well-known and widely implemented Internet protocol for communication—HTTP—the foundation upon which all Web sites operate. While HTTP's simplicity has many drawbacks, the advantages are numerous. The most obvious is that most organizations already have a Web infrastructure in place, so the most basic Web Services implementations can be handled in a familiar way, and the extensive range 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.

The description of a Web service can be modeled in two parts. In the abstract part, WSDL describes a Web service in terms of messages it sends and receives through a type system, typically W3C XML Schema. Message exchange patterns define the sequence and cardinality of messages. An *operation* associates message exchange patterns with one or more messages. An *interface* groups these operations in a transport and wire independent manner. In the concrete part of the description, *bindings* specify the transport and wire format for interfaces. A service *endpoint* associates network address with a binding. Finally, a service groups the endpoints that implement a common interface. Figure 2-3 shows the conceptual WSDL component model.

Figure 2-3: WSDL conceptual model



Some alternative frameworks

As mentioned earlier, precursors to Web services were EDI and CORBA. Also in this group are other frameworks having their roots in computer programming languages, like RMI (remote method invocation) and DCOM (distributed component object model), and programming languages in general. The problem with these systems is that they are too "tightly coupled," meaning that the two organizations wanting to exchange information with each other need to know a great deal about the other's systems and use similar technologies to build the communication software. When one organization changes their database or a piece of code, it is likely that the other organization will have to do the same. This type of system will only work out if there are a limited number of groups involved and they have a strong motivation to collaborate.

Systems that seek to integrate organizations on a larger scale need "loosely coupled" frameworks. In a loosely coupled system, most aspects of an organization's information system are hidden, or abstracted, from the world. There is no need for particulars such as operating system, database software, and even the information model, to be shared with others. Organizations exchange information via computer-to-computer messages, which are understood by all the partners in the exchange. The earlier description of XML and Web services obviously fits this description, but two other frameworks seek to do similar things, UML and the Semantic Web.

UML

"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components" (Object Management Group 2003, page xxv). This notion of a standard way to write a system's blueprints makes UML a candidate for developing a generic planning information system, because this helps to fulfill the requirements of a loosely coupled system. Its strengths are that its primary output is a visual diagram; it can be used to describe a system in a very loose, unspecific manner; but can also be highly specific if necessary, retaining the features of a formal method. As stated by Muller, "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 UML's strengths are also its weaknesses. While a graphic notation is great for humans, it is not computer readable. Also, generalized UML models are *too loose.* It is 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. There have been efforts to overcome these limitations by specifying an XML vocabulary for UML diagrams, and develop standards for highly specific models, but these efforts quickly begin to look like Web services, and will probably end up as such.

Web Ontology Language

The Web Ontology Language (OWL) is a relatively new initiative from the World Wide Web Consortium. It represents a major step in the maturation process of efforts to define formal semantics about Internet-accessible information content. These efforts began with a DARPA-funded effort called DAMML+OIL and more recently has moved forward under the Resource Description Framework (RDF) specification (http://www.w3.org/TR/rdf-primer/). OWL and RDF are part of a broad effort geared towards improving the description of information on the Web, called the Semantic Web. The World Wide Web Consortium (W3C) defines the Semantic Web as, "the representation of data on the World Wide Web...It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming" (*http://www.w3.org/2001/sw/*). Here is the W3C's definition of OWL:

"OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these

languages in its ability to represent machine interpretable content on the Web. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL (*http://www.w3.org/TR/owl-features/*)."

OWL and RDF have many similarities to XML Schema. In fact, they both use XML Schema as their recommended expression language. The major difference between XML Schema and the semantic languages seems to be in the amount of flexibility allowed in defining relationships. XML Schema is limited in its ability to say that one thing is like another without defining them as being of the same data type. It is also difficult to construct relationships between resources without prior cooperation between the developers of those resources. On the other hand, OWL and RDF have very specific language constructs to explicitly define the relationships between objects. This makes the semantic languages very good at creating taxonomies and reconciling the different taxonomies that various organizations may create. Where the semantic languages run into trouble, however, is when one tries to build a data-centric application. The very flexibility that is such a positive feature in some situations becomes a negative when an application must count on a certain data field being present in every object it encounters (Forsberg and Dannstedt 2000).

OWL may eventually become an appropriate framework in which to build a collaborative planning support system vocabulary, but the technology is too young to consider for practical experimentation at this time, and this project did not identify any information modeling issues that were beyond the capabilities of XML Schema to handle.

Chapter 3: A Study of Regional Growth Planning: the MassGIS CPI buildout analysis

Using empirical evidence is always helpful in elucidating a theory. In this case, we are interested in looking at a common class of planning analysis that is undertaken by practitioners (as opposed to academics), is widely used, and is relatively modern. The analysis presented here was chosen for the following reasons:

1. It addresses growth management, one of the most pervasive concerns of urban planning.
2. It covers many different types of places, being intended for use by all 351 cities and towns in Massachusetts.
3. The range of jurisdictions involved is diverse, including state agencies, local planners, zoning boards, elected officials, and private sector consultants.
4. Growth planning has intrinsic spatial qualities, ensuring that work on this problem will take into account the special concerns of spatial information.

This chapter begins with an overview of Massachusetts' Buildout analysis, a planning support system that calculates maximum residential and commercial land use based on a town's current zoning. As this is not a formal case study, many details are left out, such as its evolution, its role in the state's larger growth management efforts, and even its successes and failures. Instead, we infer its importance by the fact that it was enacted and funded, and all 351 municipalities have been analyzed. After a brief introduction to the enabling legislation that funded it, a detailed description of the analysis is presented,

focusing on data requirements and analytic methodology. Finally we abstract out the major themes of the analysis, the types of organizations involved and the level of coordination required of them, and the sustainability of the work, or its ability to be repeated as its assumptions change. The intent of this chapter is to take a concrete, practical analysis and use its strengths and weaknesses to highlight the issues that must be addressed by any framework for urban information management, and ultimately to drive the design of new software.

Policy Background

Where do you want to be at buildout?[1] That is the fundamental question posed by Massachusetts' Community Preservation Act (CPA). Initiated by the Executive Office of Environmental Affairs (EOEA) and enacted in December 2000, this effort seeks to, "promote smarter land use to preserve and enhance the quality of life in communities across the Commonwealth." (*Buildout Book*, 2001). Put in a broader context, this is a statewide planning initiative geared towards curtailing unchecked land development falling squarely in the policy arena of "smart growth." The Act contains a number of policy instruments designed to help municipalities make their own, better informed planning decisions. Small grants are given to develop Community Development Plans, and "Fiscal Impact" and "Alternative Futures" tools have been built and are available for local use. The focus here is on a tool developed by MassGIS and regional planning

[1] Buildout is defined as the maximum development allowed by right according to a municipality's regulations—most notably zoning, but also including environmental protection, site suitability, etc.

agencies, the Buildout analysis, which maps out the consequences of full development under current zoning regulations.

The general objective of the buildout analysis is to predict the maximum number of new homes, residents and businesses allowable under current zoning regulations. The hope is that having this information will encourage towns to revise their zoning to better reflect the amount and type of development they desire. The analysis begins by excluding protected open space and other lands having permanent development restrictions from development. All previously built up residential, commercial and industrial areas are also excluded at this point (a side effect is that this model does not allow for redevelopment). The remaining land is then assigned values for new homes and businesses based on the lands' zoning classification. In cases where there is likely to be some limitation to development, as in wetlands and on steep slopes or poor soils, a heuristic is applied to reduce the development potential of the area by some amount.

The intent was not to build an operational model that would help towns develop better growth policies, but to simply spur communities to become concerned about the issue. No one really believes that full buildout will occur throughout the Commonwealth or even throughout a community. But it is well within the realm of possibility that full buildout could occur in a block or neighborhood, and this can have a devastating impact on the character of a community.

Buildout is not a particularly exciting analysis from a modeling standpoint. There are only two time periods available for examination—the current state of the town, and its state at full development. Also, the development rules are very simple. In this model, development is mainly limited by environmental factors. If the land's building capacity is

not constrained by steep slopes, bad soils, wetlands or floodplains, it gets developed to the highest density allowed by zoning. There is no accounting for economics or transportation constraints, for example. In addition, since time is not a part of the model, buildout could occur in ten years or ten thousand. However, these factors that make the model less realistic from a growth planning point of view are there for a reason. Each one of the 351 cities and towns in the Commonwealth has been run through the analysis. The data requirements and analytic methodology were designed to be within the abilities and budgets of even the smallest towns, so that the effects of development could be seen not only for every town, but also regionally across jurisdictions. This comprehensiveness makes the buildout analysis extremely interesting from the point of view of one interested in examining information-dependent analysis systems that have wide application.

Process

The buildout model has the following general structure:

1. Identify zoning districts that permit development.
2. Remove areas that are already developed (even if they might be under-developed).
3. Remove areas that are absolutely unsuitable for development (due primarily to environmental constraints).
4. Identify areas that may only support partial development due to environmental constraints such as the presence of wetlands or floodplains. Compute a statistic for these areas that indicates how much "less" developable these lands are than those with no constraints.

5. Compute the number of new residences and businesses that can be developed based on zoning attributes such as floor-area ration (FAR) and lot setbacks.

The maximum buildout envelope—Zoning (step 1)

The buildout analysis uses a town's zoning laws to determine maximum development. This may seem logical, but it is actually quite different from the approach taken in common growth models such as CUF or UrbanSim, who base their development estimates on more realistic assumptions than *full* zoning buildout. The point being made in Massachusetts, however, is why have a zoning plan that you have no desire to see realized? The intention being to have communities thoughtfully revisit their land use regulations from the standpoint of what do they desire twenty years from today. This is MassGIS' guidance on how to integrate zoning data:

The contractor will develop or update zoning (ZONE) and zoning overlays (OVER) from the most current town zoning map or maps, digitized with reference to the most current town zoning by-law and registered to the town boundary layer from MassGIS. The polygon attribute table of these GIS layers must conform to the MassGIS/RPA standard for attributes as implemented in the MassGIS library which is attached to this contract. Zoning overlays should be digitized only if they will have a real impact on development – in many cases they impose minor restrictions which won't affect the basic buildout analysis.†

The incorporation of zoning data into the model would seem straightforward, but this is complicated by the need to unify all the towns' zoning classifications and

definitions to a single standard. Otherwise there would have to be a (slightly) different model for every different zoning manual.

Current buildout—Land use and Subdivisions (step 2)

The MacConnell land use will be part of the analysis and needs to be reclassified to show residential, commercial/industrial and undeveloped land.†

Establishing baseline development involves three data inputs. MassGIS starts with a statewide land use coverage to identify areas already developed as residential or industrial/commercial. This data set was developed from aerial photography interpretation. Since these photographs were taken throughout the 1980s and 1990s, they are a bit out of date, and they are not very precise, so small, isolated land uses are missed due to their 1:25,000 (1 inch equals about 0.4 miles) scale value where the minimum mapping unit was one acre.

In order to map subdivisions and/or to update the land use mapping, which will be critical inputs to the process, the contractor should look at the history of subdivision filings since the date of MacConnell land use mapping. If there are a sufficient number of non-ANR subdivisions to warrant, a separate subdivision layer should be created. Essential attribute information to be collected and assigned to the subdivision polygons includes subdivision-id, name, date, number of lots, number of houses built to date and total acreage. Ideally this information would come in soft-copy form and could be linked to the subdivision mapping. Additionally the contractor should obtain any available map showing the new subdivisions at a scale suitable for transfer to a town-wide map.†

Augmenting this statewide land use coverage with local knowledge can solve both the precision and currency issues. For this reason MassGIS requires towns to update

land use using a higher resolution aerial photography set flown in 2001. All towns, however, will not have ready access to someone skilled in the art of aerial photography interpretation, so all that can be asked of the town is that they identify residential and commercial/industrial uses, whereas the official statewide land use data set classifies land use into 21 to 33 types, depending on who did the interpretation and when it was done.

2001 is still a bit old for some towns, even in one of the slower growing regions of the country. The final input to current development is a residential subdivision data set that the town may optionally provide. This can only be created in a cost-effective manner if developers have submitted electronic plans and the local government uses them.

Just like zoning, land use data must be provided to the model in a generic data schema, so towns must follow MassGIS' guidance on land use updates and subdivision data development.

Absolute constraints to development (step 3)

Some lands are considered not developable in this model. In addition to those already built up areas described above, there are a number of land use types that are excluded from development by either environmental or legal constraints. In this model, this refers mainly to permanently protected open space and farmland. This type of property is defined as "land which is held in fee ownership by a government agency or a private non-profit organization for the purpose of conservation or water supply protection or which has deeded restrictions on development" (MassGIS). MassGIS

already is the official maintainer of a statewide data set cataloging open space in a high degree of detail, so the acquisition and use of this data is trivial.

Partial constraints to development (step 4)

The buildout analysis has a concept of "partially developable" lands. These include wetlands, steep slopes and flood plains. These types of land are considered un-developable in most models, and this model is no different in that it does not allow structures to be built in these areas, this model is more realistic if a portion of these areas are projected to be part of a built-up lot, because they could fall into that lot's setback or open space allocation.

The actual amount of development permitted in these areas is based upon a combination of site-specific factors, including the size of the zoning district, the size of the partially developable area in relation to the district, and the type of development allowed in the district. For this reason, these factors are computed on a case-by-case basis in a spreadsheet.

Finally, after analysis of the town zoning by-law and the other source documents collected above, the contractor will determine if any other legal, physical or environmental factors will so significantly influence or constrain future development in the town that no reasonable buildout analysis can be done without considering them.

Finally, MassGIS allows each town to have a "wildcard" layer. This allows towns to use their own judgment to exclude from development anything that the generic analysis overlooked. This is a very interesting feature of the methodology, as it seems to contradict the basic principles of doing a standardized analysis. But in order to have truly

committed participation in the system, this is a useful way to make sure every town's unique needs are addressed.

Buildout computation (step 5)

Three types of summary table may be produced from the polygon attribute table for potentially developable land from step 7. One table gives, for each zoning district classification, the total area within the town for each combination of constraints present within that zoning district. Thus, if floodplains are mapped as a partial constraint, the town might have 2000 hectares of R1 *district without any constraint, and an additional 100 hectares of land in the* R1 *district that are in the 100 year floodplain. This table can be the basis of the analysis of a generalized analysis that provides a rough estimate of buildout potential. If all constraints are treated as absolute constraints, then there is simply one record for each zoning category giving the total potentially developable area within that district.*

Optionally, a second, more detailed analysis will require summarizing by individual zoning polygon – this would be appropriate where the distribution of partial constraints is very irregular and certain polygons end up with little or no allowable building because of an atypical concentration of constraints. In this case, the zoning polygon –id should be referenced to a map with those –ids printed for the individual zoning polygons. Finally, if parcel mapping is available, the analysis can be done to summarize for each parcel (or each parcel above a certain minimum) the characteristics of that parcel.†

The buildout is ultimately a computation of the number of new residences and offices that may be developed. The analysis just described, which was mainly spatial in nature, provides a list of zoning districts and the proportion that may be developed. In the case of areas with no constraints, this proportion is 100%. In areas with partial constraints, the number is less, and where the constraints are absolute, the number is

zero. At this point, it is a matter of simple mathematics to compute the number of structures that can be developed based on the zoning code's attributes, such as minimum lot size, setback requirements, road frontage, etc. This step is also performed in a spreadsheet environment.

The results

The results of a buildout analysis are a series of maps and statistics describing maximum buildout potential in the municipality. The series of maps have already been presented here, and they serve the same purpose as they do here, which is to graphically illustrate the analytic process. The statistics are the buildout computation described in step 5. It is worth reiterating that the intended result is not to tweak this model so that the maximum buildout based on zoning regulations matches the town's development objectives. EOEA simply hoped to catalyze local interest in urban growth policy. This is no different, however, than the goal of most planning efforts—even those based heavily on expert analysis.

Figure 3-1: Absolute Constraints for Sutton, MA Buildout

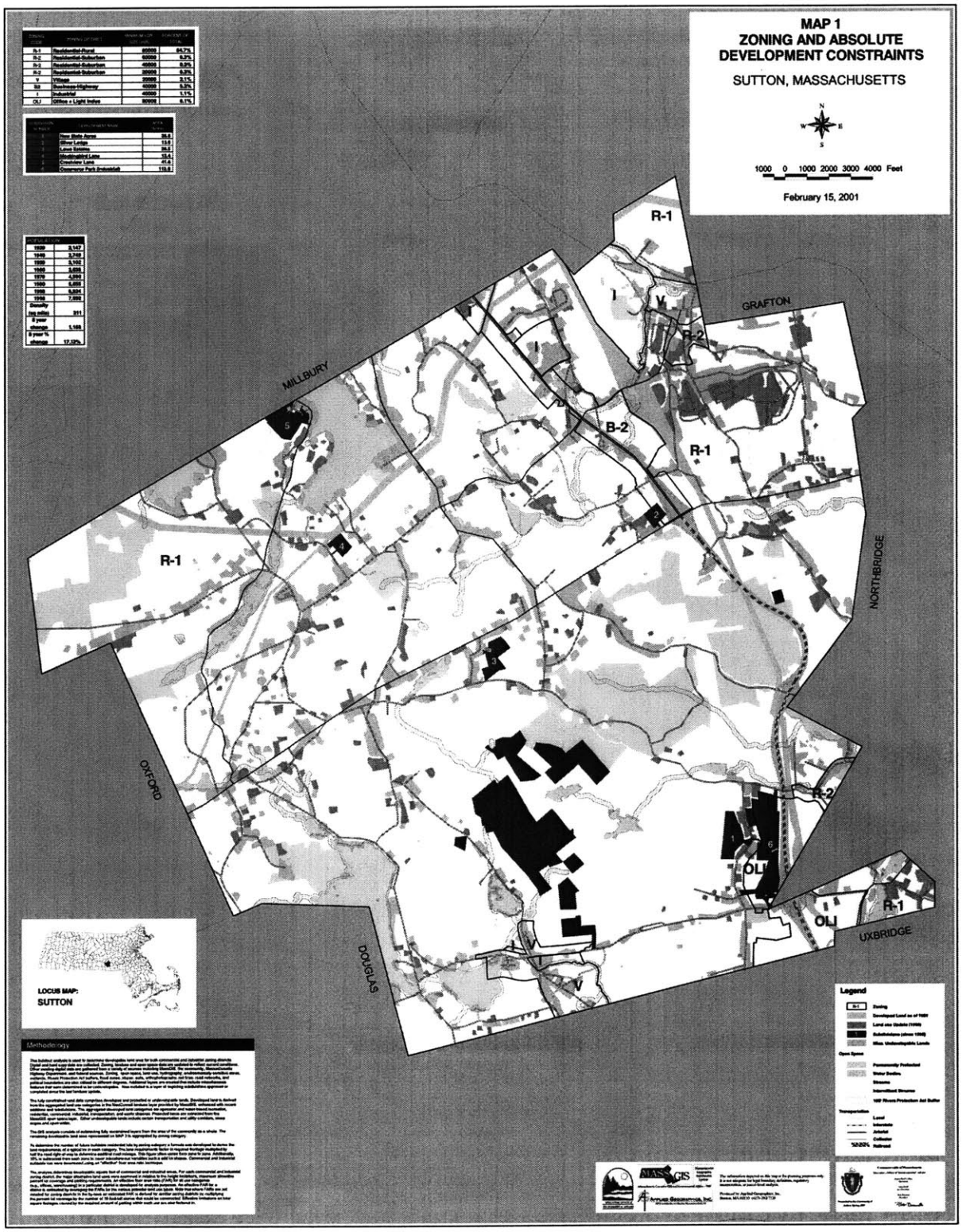


Figure 3-2: Developable Lands and Partial Constraints for Sutton, MA Buildout

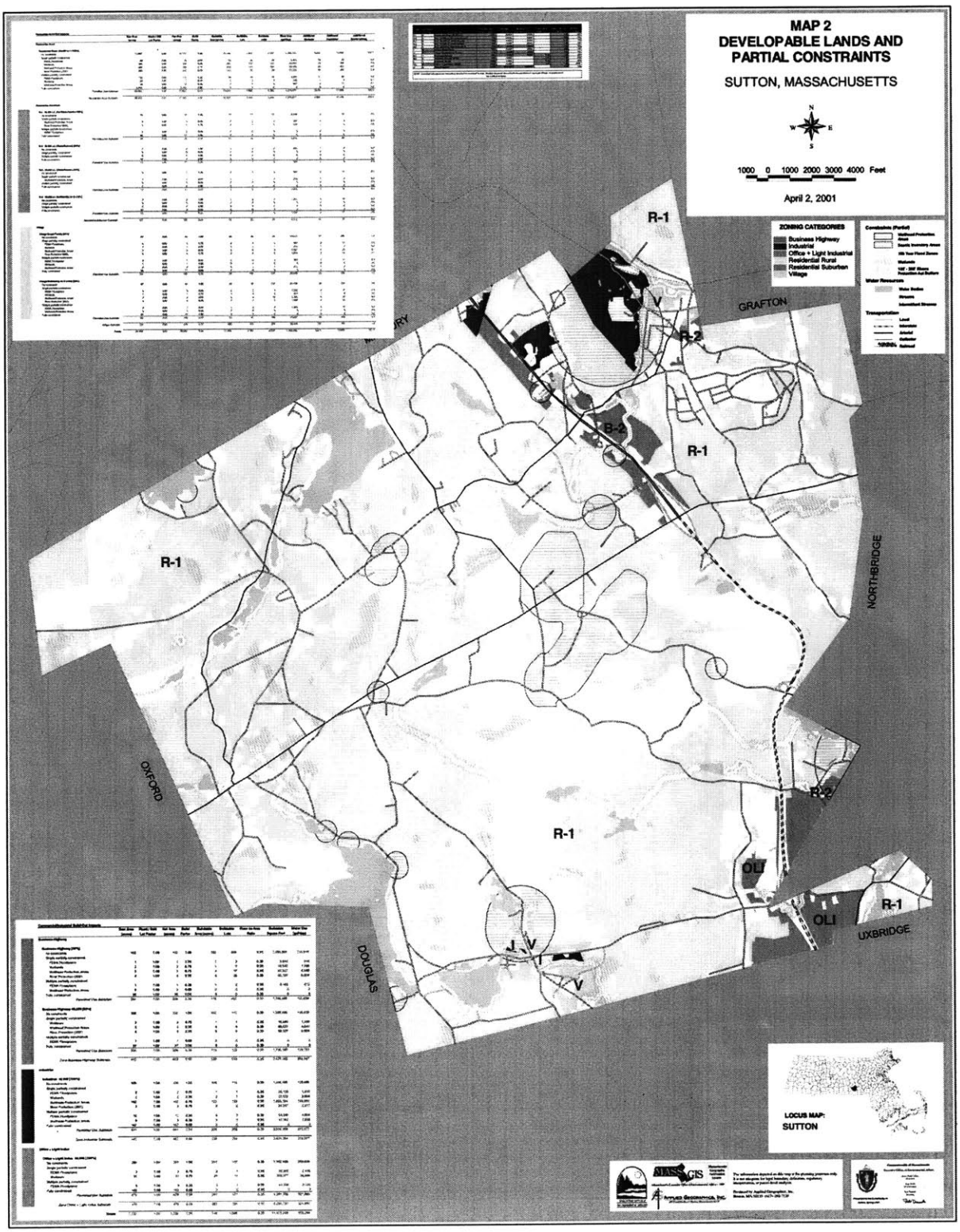


Figure 3-3: Composite Development for Sutton, MA Buildout

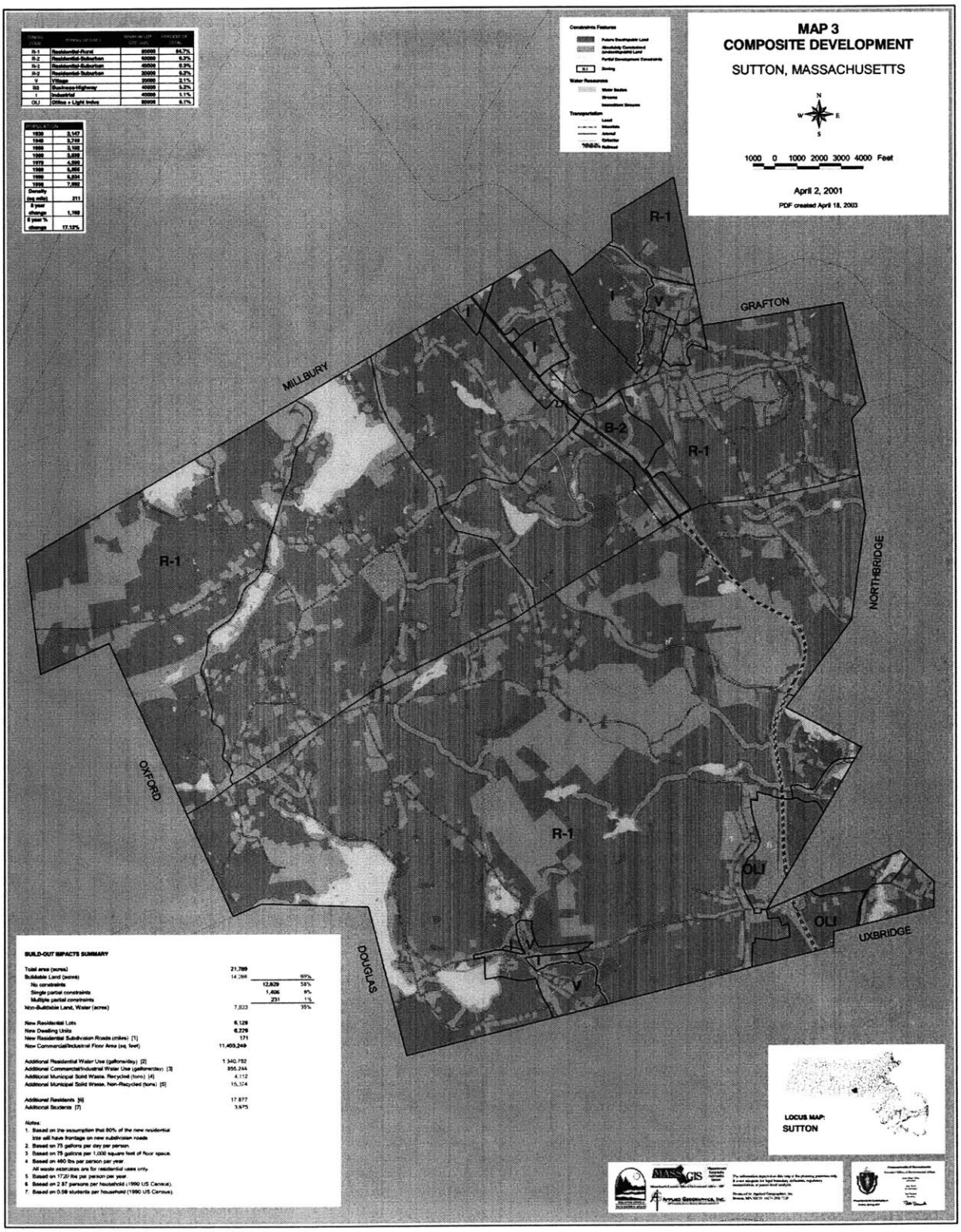
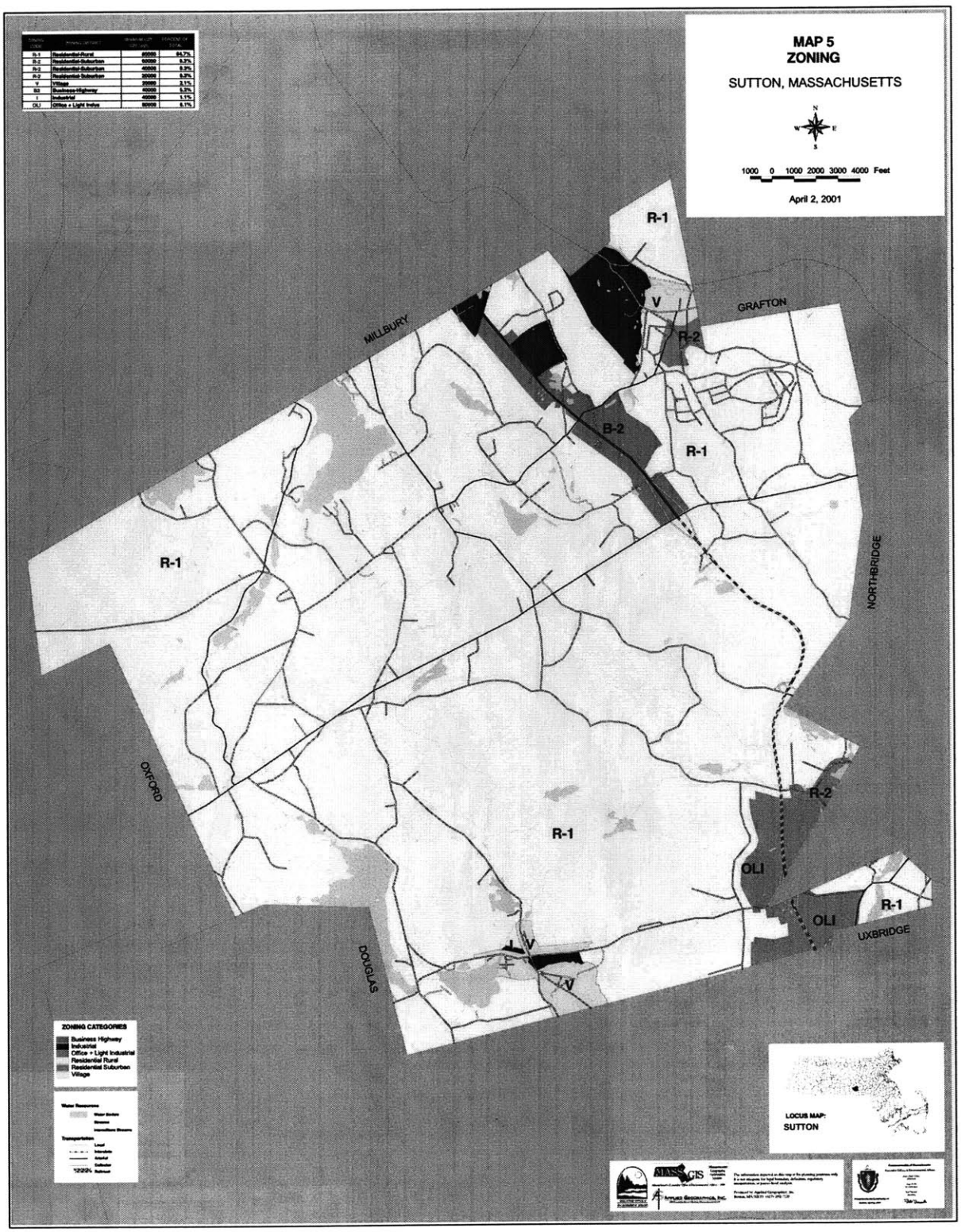


Figure 3-4: Zoning for Sutton, MA Buildout



Key Concepts & Systemic Problems

This section highlights key aspects of the buildout analysis' methodological structure and information requirements in order to develop an argument regarding why the procedures used have inherent, systemic drawbacks that can not be addressed without a major shift in the way organizations integrate information technology into their work.

By most accounts, the buildout analysis has been a qualified success in that it has brought growth management tools to every town in the state in a consistent way (Hodges, 2004). Most Massachusetts' towns are small and have almost no full-time government, let alone planning staff—yet home rule dictates that land use decisions be made at the local level. Combine this with the lack of any significant government structure at the county level, and the Commonwealth is left with a significant challenge to its ability to manage development. The buildout analysis tries to bridge this gap by presenting growth from the perspective of real land use policies, instead of abstract projections of trends in migration, job creation, housing policy and such. This strategy is powerful because it is based on the data, policies and regulations that towns control. But on the other hand, basing a model on real data and real laws creates the expectation that the model is integrated with those data and always up to date. This, of course, is where we want to be as a profession; but not where we are now.

Aside from the actual veracity of the model, some would say that the real purpose of the project has been to spur interest in land use planning and growth management. From a policy perspective this could lead to positive change without a buildout analysis leading directly to a change in zoning. However, it seems like a waste of money to simply use

"scientific" analysis to generate interest in a topic. More attention must be paid to what it actually means to *use the buildout analysis to continuously inform an ongoing planning process*. In other words, if the project is able to spark a policy debate, it should be a useful tool in that debate and should continue to provide stakeholders with a means to develop knowledge out of the vast quantity of information we maintain about place during normal government operations.

What follows is a critique of the buildout analysis, despite the fact that it represents "good" planning analysis and mechanisms for stakeholder participation. It still suffers from a host of systemic problems in the way the study is designed and executed. These problems are so important because they are present in most planning support systems, so a close study of MassGIS Buildout should be useful as a general theory. The large, systemic issues highlighted here so that they may be addressed throughout the rest of this paper.

Simple math

The simplest aspect of buildout is the analytic methodology. The basic concept is to perform the type of site selection analysis that planners have used for decades (Lynch and Hack 1984). Instead of a single site, however, the analysis is performed for an entire town, being limited mainly by environmental constraints, which are determined in a manner which differs little from Ian McHarg's seminal overlay techniques (McHarg 1969).

So the analytic theory behind buildout is thirty years old, but so is the math. Areas to be developed are determined by cutting out unsuitable lands from the zoning map. This basic type of spatial overlay is what geographic information systems were created for in the 1960s. Buildout uses none of the latest techniques like spatial statistics or agent-based modeling. After developable areas are identified, the actual amount of development is determined by overlaying areas that impose partial constraints on construction. This concept is expressed as a potential construction percentage, between 0 and 100, and the maximum amount of development allowed by zoning is multiplied by this percentage. This part of the analysis could have easily been performed twenty years ago using Tomlin's map algebra language and software (Tomlin, 1983). But MassGIS chose to simplify it even further, by performing this step in a spreadsheet, so that the technical requirements are acceptable to virtually every person in the state.

Extensive data requirements, from multiple agencies

The buildout project's data requirements stand in stark contrast to the simplicity of the analysis. MassGIS has developed a large storehouse of GIS data for Massachusetts, especially pertaining to the natural environment. Buildout uses many of their statewide data sets, including open space, land use, aerial imagery, wetlands, flood plains, topography, areas of critical environmental concern, and roads. While most of these are developed, or at least edited by MassGIS, some come directly from federal government agencies such as USGS and Census. This information has all been put online in a single archival data format and documented formally. MassGIS has performed regular updates of their data warehouse and consistently maintained online access for years.

The creation of the zoning data set is unique in that zoning is created and controlled by each individual town. Especially in a home rule state like Massachusetts, it is difficult to translate every town's zoning regulation to a common standard, so the development of a statewide zoning layer is even more impressive.

The final data requirement is for the most recent subdivisions, which serve to update the land use plan with the latest development. Up to this point, we have had the involvement of a state GIS agency, one or two federal agencies, and the municipal zoning board. Subdivision data brings in the local assessors office, and may even require data from private developers, giving the project an information landscape that includes every type of data provider except for individual residents.

Zombie data

This might sound like a strange term to use in a scientific paper, but our profession currently has no term to describe this condition (and it is hard to solve a problem you can not name). Zombie data is not quite alive, because it has been detached from its native environment and is no longer being checked and updated. However, it is not quite dead because it is still being used in the way only living data should be.

Administrative agencies usually are the only ones with living data, and planners almost always have zombie data. For example, town assessors and registries of deeds have ownership and cadastral information; building departments have construction permits and new subdivision applications; and banks have the latest sales and loan-to-value ratios. But planners usually have old, out of date data sets that have a life of their own. Not only do these data sets get used in analyses, they move around in planning

support systems, sometimes supplemented with additional calculations or personalized updates when they should have been replaced by fresh, live data a long time ago. And since they are not simply out of date, but may actually contain useful new information, they are even more difficult to put to rest and replace with an updated copy.

Often the issue is less of a methodological one, because an analysis based upon slightly out of date information is probably still sound. The larger issue is likely to be public confidence. Most people (in fact, anyone who did not construct the analysis) will not have the time or the inclination to understand the analysis well enough to know whether its results require the most up to date information. They will simply assume that outdated data equals an outdated analysis. So the problem of zombie data is threefold: It can invalidate the results of an analysis; it can make future updates difficult; and it can shake public confidence in the study.

In the case of MassGIS Buildout, the two data sets that are most susceptible to this problem are parcels and zoning. Property development is always one of the most dynamic data urban data sets, and when the study is about growth, new development is under an even brighter spotlight. The buildout analysis highlights the importance of accurate, current parcel information by discussing a number of ways to acquire it. There is no mention, however, of how to make the information gathering process replicable across towns, or over time.

Stakeholder participation

Oddly enough, the buildout system architecture does little to facilitate or encourage its stated goals. Just as strange is that this is not uncommon. Remember that the goal is

to educate communities about the impact of unchecked development and motivate them to rationally plan for growth. The Community Preservation Act as a whole is able to work towards these goals, but the analysis piece is disconnected from the policy work.

The inclusion of stakeholders' concerns into the planning process is always mentioned as an important phase of the project, but what are we doing *methodologically* to facilitate this interaction? Is there any mention of how feedback is incorporated into the model, or at least the public record of the project? What is the project's public record anyway? The lack of attention to these questions is by no means unique to the buildout project. The two disciplines of analysis and collaborative decision making seem to always be holding each other at arms length. At this point the intention is only to draw attention to the concern, so that we may come back and address it later in the paper.

Interactive end-product

The standard deliverable from a project of this type is a bound paper report containing maps and tables embellished with plenty of explanatory text. The buildout analysis provides these for all 351 Massachusetts' municipalities, but two other more interactive end-products are also offered, putting the project on the leading edge of providing the public with participatory tools and transparency in government operations.

The first interactive end-product is accessed through the EOEA's Community Preservation Web site, `http://commpres.env.state.ma.us/content/buildout.asp`. Here a visitor can create a regional buildout analysis by choosing any number of towns within a region. The site basically adds up the data for each town chosen on-the-fly. While this is

computationally simple, it provides some limited ability to see what the aggregate impacts of development might be.

The second product may be downloaded from this Web site, but it must be run on one's own Windows™-based desktop computer. This product consists of the GIS data files used to create the "official" buildout analyses, plus proprietary scripts to reproduce the analysis. If an individual or group can meet the software requirements—ESRI ArcView GIS and Microsoft Excel—and has the technical capacity to use the software and understand the analytic methodology, all aspects of the analysis can be altered and re-generated (Jacqz, 2004).

For the sake of discussion, let's assume that all municipalities have easy access to ArcView, are skilled in its use, have in-house planning expertise, and have a complete understanding of all aspects of the modeling process. In this scenario, a town is able to take the analysis and update the base data to account for changes in zoning, new developments, open space acquisitions and such. In this way, the analysis for the town can always be up to date and accurate. They may also challenge some of the model assumptions and want to adjust variables like the average number of children per household, water and sewer usage, or automobile trip generation.

As you can see, the buildout analysis can be a powerful planning tool in the right hands. The first inherent problem with this utopian scenario is that most Massachusetts communities have no planning staff—professional or amateur—so it is highly unlikely that more than twenty to thirty of the state's 351 municipalities have the resources to contemplate making buildout analysis a regular part of their quarterly or yearly planning work.

The Community Preservation Act tries to address this by providing grant money to hire consultants, but these funds may only be available once or twice in a twenty year period, so the buildout analysis is likely to remain a static document. And while larger towns have the staff to use the buildout analysis, they are the most likely to ignore it because the methodology only allows new construction on undeveloped land. This works best in rural and suburban areas, which have little or no regular planning staff, not our dense cities like Boston, Framingham, Lawrence, New Bedford or Worcester, where new development will usually involve infill, or the replacement of pre-existing structures.

If the buildout analyses are to be used effectively by smaller towns, it will have to be through a partnership between towns and regional planning agencies (RPAs). But this brings data issues back to the forefront. Municipalities can not even share data across departments, let alone with another level of government, so we are left with a systemic mismatch between information flows, land use regulation and planning analysis. In my opinion, addressing this mismatch is one of the decade's great challenges for planning support systems.

Next steps

Major shortcomings in the Buildout analysis have now been identified. Information technology offers numerous solutions to those problems, which will all require tradeoffs in regards to cost, complexity and business process re-engineering. Therefore it is critical that the chosen solution be based upon sound theories describing the nature of the problem. This chapter developed those theories and showed their relevance to the Buildout analysis. Some strong suggestions were made regarding the problems

technology should solve to move the profession forward. The rest of the paper presents one solution—a suite of technologies that conform to the theoretical foundation laid down here, and have the ability to fundamentally and structurally improve the efficacy of planning support systems.

Chapter 4. Sharing Data through Web Services

Data sharing would seem to be a simple task. Agencies have been making their data publicly available through the Internet since the 1980s. The World Wide Web in its early form can be thought of as a big, read-only file sharing network. High-speed networks allow gigabytes of data to be moved from one place to another in very little time, and the cost of these networks keeps decreasing. So why is sharing data still a problem?

In the buildout analysis, a host of data sources are used. In the case of zoning, the primary challenge was translating each town's zoning categories into matching categories. With land use, the big problem was finding and acquiring the most up-to-date data sources, systematizing their inclusion into the analysis. The latest data is usually the most disaggregated, and in the hands of the smallest organizations with the least incentive to participate in a larger system. In this case these are the developers who are building the newest residential subdivisions.

Sharing data with government, and supporting planning support systems are not the primary mission of developers, yet highly detailed data sets are critical in an urban information infrastructure. They are usually created and maintained by small, local organizations, so there must be a mechanism for data publishing that conforms to their level of technological sophistication. However, at the other end of the spectrum the system must be sophisticated enough to support complex analyses. This chapter lays out a Web services strategy for meeting these seemingly conflicting goals.

WSDL

We start with a very simple example, because an important design element is the ability to offer simple solutions for simple requirements. In this case, the requirement is to enable an organization to publish spatial data as easily as they publish Web pages. The most common spatial data format is the ESRI Shapefile. Shapefiles are like Adobe PDF files in that the data format is public and free to use, and the files are small and easily emailed, making the Shapefile the *de facto* standard in the GIS world. Instead of simply placing these files on a Web site, publishing them through a Web service interface allows the data to be more tightly integrated into information processing systems, hopefully in a more fully automated manner.

First of all, it is important to emphasize the similarities between a Web site and a Web service. In the strictest sense, any part of a Web site can be a Web service if it is described formally. For example, a Web page is a text file containing data in Hypertext Markup Language (HTML). It is accessed using the Hypertext Transfer Protocol (HTTP) by sending a GET request to a particular Universal Resource Locator (URL). If the previous two sentences are written formally in a particular dialect of XML called Web Services Description Language (WSDL), the Web page becomes a Web service.

Code Listing 4-1 presents a simple WSDL file that serves to publish a Shapefile as a Web service. A WSDL file has four sections, `service`, `binding`, `interface`, and `types`. The service section tells a user what Web address to access in order to invoke the Web service. The interface sections tells the user what commands the service understands, and the types section describes the format of these commands and the responses that may be returned. The binding section has technical details relating to how the commands

described in the interface section must be expressed in a particular language. A service could have one interface and many bindings, meaning that the same command can be expressed in many different languages. Another important concept is that the WSDL expression of a service is an abstraction. There could be other Shapefiles on this Web site, and they may or may not be "published." There could also be other services that "publish" the same data, but use a different WSDL file—meaning that the data is published in a different way to a different audience.

In this way the Web service can be crafted to meet the exact requirements of an organization. This can be a useful concept if we think of the WSDL file as bridging the gap between organizational and technical concerns. In formally describing the data sources, and the means of accessing them in a highly structured manner, WSDL becomes not only a technical solution to data sharing, but a *contract* between the data provider and the data user. This is the contract that trading partners require to ensure a stable relationship in regards to information exchange.

Basic Data Sharing: one Shapefile

In order to publish a Shapefile as a Web service, three things must be put on a Web server:

1. The data files being published.
2. A WSDL file describing certain generic aspects of a Shapefile.
3. An XML file describing the specific Shapefile being published.

The generic aspects of a Shapefile are described in the `types` section of Code Listing

4-1. We see there an XML Schema element called *ShapefileWriter*, named so to distinguish between a service message that outputs, or writes, a Shapefile, and one that ingests, or reads one. Note the XML attribute *srsName*. All spatial data has a particular spatial reference system (SRS)—a way of referencing locations on the earth. Cartographers have hundreds of different ways of doing this, based on tradeoffs between accuracy, scale and other considerations. These different systems have all been given a name, and that is what would be stored in the *srsName* attribute. of the *ShapefileWriter* element. Shapefiles store their data in three files having .shp, .dbf, and .shx suffixes. The locations of these files are specified in the *ShpFile*, *DbfFile*, and *ShxFile* elements as URLs.

The interface, binding, and service sections combine to say that the Web request, *http://www.city.us/wetlandsShapefile.xml*, will be answered with an XML file conforming to the XML Schema defined by the *ShapefileWriter* element. In this case, a possible response is shown in Code Listing 4-2. A small, unsophisticated agency could put the two XML files on their Web site along with the three Shapefile components, and consider the data published by giving interested parties the URL to the WSDL file. This is the bare minimum required to participate in the collaborative framework envisioned in this paper.

Code Listing 4-1: WSDL file for Shapefile publishing

```
<definitions name="DataPublishing">

<types>
   <xs:schema targetNamespace="http://web.mit.edu/pamml.wsdl">
      <xs:element name="ShapefileWriter" type="ShapefileWriterType"/>
      <xs:complexType name="ShapefileWriterType">
         <xs:sequence>
            <xs:element name="ShpFile" type="xs:anyURI"/>
            <xs:element name="DbfFile" type="xs:anyURI"/>
            <xs:element name="ShxFile" type="xs:anyURI"/>
         </xs:sequence>
         <xs:attribute name="srsName" type="xs:string"/>
      </xs:complexType>
      <xs:element name="NullMessage" nillable="true"/>
   </xs:schema>
</types>

<interface name="PublishDataInterface">
   <operation name="GetData" pattern="http://www.w3.org/2003/11/wsdl/in-out">
      <input message="tns:NullMessage"/>
      <output message="tns:ShapefileWriter"/>
   </operation>
</interface>

<binding name="PublishDataBinding" type="tns:PublishDataInterface">
   <http:binding verb="GET"/>
   <operation name="HTTPBindingGetDataOperation>
      <http:operation location="/wetlandsShapefile.xml"/>
      <input>
         <http:urlReplacement/>
      </input>
      <output>
         <mime:content type="text/xml"/>
      </output>
   </operation>
</binding>

<service name="PublishDataService">
   <documentation>Geospatial data accessible from this server</documentation>
   <endpoint name="DataServiceURL" binding="tns:PublishDataBinding">
      <http:address location="http://www.city.us"/>
   </endpoint>
</service>

</definitions>
```

Code Listing 4-2: XML instance document for Shapefile publishing

```
<ShapefileWriter srsName="EPSG:26986">
   <ShpFile dataFile="http://www.city.us/wetlands.shp"/>
   <DbfFile dataFile="http://www.city.us/wetlands.dbf"/>
   <ShxFile dataFile="http://www.city.us/wetlands.shx"/>
</ShapefileWriter>
```

A full explanation of the WSDL specification is beyond the scope of this work. However it is important to note a few characteristics of this approach. A WSDL file is quite complex, and a small organization would probably need to contract out for its development. But still it is only a text file, so no additional software or hardware, beyond what is required to publish Web pages, is needed to participate in what will be shown to be a sophisticated system. This point is so important because it matches so well the way organizations function. Most organizations—even small non-profits—are able to initiate large, complex projects because it is at the beginning when the project's champions are still in place and there is usually some commitment of resources. Problems usually arise over time, or after the project is "officially" over (meaning no longer explicitly funded), when time, maintenance and upkeep must be incorporated into a general operational cost structure. With finite resources and turnover in leadership, old projects tend to lose funding and time commitments and cease to operate if their upkeep requires any extraordinary effort. In publishing this Web service we have a complicated project initiation stage, where the data and XML files must be created and posted on the Web site, but a simple maintenance stage that only requires the upkeep of a Web server, which is probably critical to other organizational initiatives as well.

Professional Data Sharing

The previous section focused on the requirements of small agencies whose technology infrastructure was limited to a Web server. This is a sensible baseline

technology, considering that even millions of individuals in the U.S. have their own Web site. The implementation strategies outlined above do not meet the needs of professionals, however. GIS agencies, planners, assessors, and the like have broader requirements, and a more sophisticated technology infrastructure, than a basic Web server. In this section we address the needs of these more traditional spatial data providers. Generally, these are municipal, regional and state agencies that publish numerous data sets, often in multiple formats. Sometimes these data sets do not reside on disk, but in a database, or are generated on request. Another important characteristic of these kinds of organizations is that they often update their data, so their customers must be made aware of this fact and consider the update event in managing their own business processes. Finally, these agencies are concerned about their data's provenance. Making sure their users know when a data set was created, last updated, or its level of accuracy are concerns that have significant organizational, if not legal, ramifications.

Metadata

Information about a data set is generally referred to as *metadata.* The subject of what should be recorded in metadata is an active field of inquiry. In the U.S., the Federal Geographic Data Committee (FGDC) has for over a decade championed the FGDC Metadata Standard. Internationally, the International Standards Organization (ISO) has issued a standard called Geographic Information — Metadata, which is commonly referred to by its document identification number, ISO19115. What these organizations are trying to do is to capture, in broad terms, the general characteristics of geographic information so that potential users can search for information relevant to their task, and quickly decide whether that information meets their needs. This involves capturing

spatial metadata, such as the geographic extent of a data set, attribute metadata, such as the names and data types of attributes, and administrative metadata, such as the responsible agency, date of creation, and update frequency.

Metadata is not the focus of this research, but it certainly plays a complementary role. The latest metadata standardization efforts of organizations like the FGDC, ISO, and OpenGIS rely on XML technologies, so the XML-focused work presented here can easily incorporate metadata by simply using XML's built-in extensibility mechanisms. Code Listing 4-3 supplements the XML definition of *ShapefileWriter* from Code Listing 4-1 to support metadata. A new element, *Metadata*, is added to the object, and it is defined in a very general way in the *MetadataType* object. This is simply an object that can have any XML content in it, allowing an organization to incorporate their metadata efforts with their distributed planning support systems work.

Code Listing 4-3: Adding metadata to data

```
<xs:element name="ShapefileWriter" type="ShapefileWriterType"/>

<xs:complexType name="ShapefileWriterType">
   <xs:sequence>
      <xs:element name="Metadata" type="MetadataType"/>
      <xs:element name="ShpFile" type="xs:anyURI"/>
      <xs:element name="DbfFile" type="xs:anyURI"/>
      <xs:element name="ShxFile" type="xs:anyURI"/>
   </xs:sequence>
   <xs:attribute name="srsName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="MetadataType">
   <xs:sequence>
      <xs:element name="Publisher" type="xs:string"/>
      <xs:element name="Date" type="xs:date"/>
      <xs:any minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

Object inheritance, and sharing multiple files through a single service

The number of common spatial data formats seems endless. Organizations that publish spatial data often make it available in multiple formats, to support the various software environments of their users. There are a number of file-based formats that are similar to Shapefiles in that they are defined by the locations of their component files. Another big class of spatial data format is the spatial relational database. This includes Oracle Spatial, IBM DB2, PostGIS, and MySQL. Accessing data in these formats generally involves making a database connection, which requires some authentication and network location information. An example of how a PostGIS data source could be modeled is shown in Code Listing 4-4.

Code Listing 4-4: Accessing spatial data in PostGIS

```
<xs:element name="PostGISWriter" type="pamml:PostGISWriterType"/>

<xs:complexType name="PostGISWriterType">
  <xs:sequence>
    <xs:element name="User" type="xs:string"/>
    <xs:element name="Passphrase" type="pamml:PassphraseType"/>
    <xs:element name="Host" type="xs:anyURI"/>
    <xs:element name="Port" type="xs:int"/>
    <xs:element name="Driver" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="srsName" type="xs:string"/>
</xs:complexType>
```

Notice that, like `ShapefileWriter`, `PostGISWriter` has the `srsName` attribute. It would also have the `Metadata` element, if fully defined, but instead of repeatedly defining objects that are common to many other objects, XML allows objects to *inherit* the characteristics of another. What we would like to say is that every data model in our system may have metadata, and must have a spatial reference system definition. Code Listing 4-5 expresses this.

Notice that `Metadata` is defined in the `ModelType` object. Here we introduce the concept that some data models might not represent spatial data. Every model may have metadata, but those that represent spatial data also have a spatial reference system (the `srsName` attribute modeled in the `GeoData` object). The concept of inheritance will be used extensively in this work. It not only provides clarity to an information model, but offers practical benefits in system implementations.

Code Listing 4-5: An object-oriented model of spatial data

```
<xs:element name="Model" type="ModelType"/>
<xs:complexType name="ModelType">
  <xs:sequence>
    <xs:element ref="Metadata" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="GeoDataType">
  <xs:complexContent>
    <xs:extension base="ModelType">
      <xs:attribute name="srsName" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="ShapefileWriter" type="ShapefileWriterType"/>
<xs:complexType name="ShapefileWriterType">
   <xs:complexContent>
    <xs:extension base="GeoDataType">
       <xs:sequence>
          <xs:element name="ShpFile" type="xs:anyURI"/>
          <xs:element name="DbfFile" type="xs:anyURI"/>
          <xs:element name="ShxFile" type="xs:anyURI"/>
       </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="PostGISWriter" type="PostGISWriterType"/>
<xs:complexType name="PostGISWriterType">
   <xs:complexContent>
    <xs:extension base="GeoDataType">
          <xs:sequence>
          <xs:element name="User" type="xs:string"/>
          <xs:element name="Passphrase" type="PassphraseType"/>
          <xs:element name="Host" type="xs:anyURI"/>
          <xs:element name="Port" type="xs:int"/>
          <xs:element name="Driver" type="xs:string"/>
       </xs:sequence>
    </xs:extension>
   </xs:complexContent>
</xs:complexType>

<xs:element name="GeoDataModels" type="GeoDataModelsType"/>
<xs:complexType name="GeoDataModelsType">
  <xs:sequence>
    <xs:element name="GeoDataModel" type="GeoDataType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Developing a better object-oriented data model also provides flexibility when we look at publishing more complex data services. In theory, multiple data sets could be

published using the strategy recommended above (for a small agency publishing one Shapefile). In practice, however, this system could be difficult to maintain for the publisher, because it requires each data set to have its own WSDL file, and since they will all be very similar, making a small change, such as updating the agency's phone number, requires changes to many files. The way organizations have traditionally published data has been to advertise one Web site with data download functionality. Perhaps this "data warehouse" paradigm is less compelling in a Web services framework, and it is better to use the one data set per service concept, but that is a debate for another time. Here we simply show that the data warehouse idea can be supported.

Code Listing 4-6 describes a Web service that publishes multiple data sets in multiple formats. The main difference between this service and the basic one is that there must be a "conversation" between the client and the service to determine which data set to give the client and in what format. In the most general sense, this is a search task. The client is searching for data of a particular type, and will be able to identify it by some characteristic, like its name, subject matter or geographic region. Searching and cataloging will probably only be done well by specialized services. This is the case with the Web in general. Individual Web sites used to all have their own internal search engine, but nowadays most sites let Google handle search.

While a handful of the largest spatial data libraries may implement their own search and cataloging functionality, most will only need to publish a short list of data sets in their holdings. This is best accomplished by creating an object that lists spatial data models. The `GeoDataModelsType` object shown in Code Listing 4-5 fills this role. Code Listing 4-6 shows how that list of available data sources is accessed by making a

`GetDataListing` request to the service (in this example, the service is invoked using a SOAP binding). From this list, the user can choose the data set they desire. The final problem to solve is how services uniquely identify data sets. The most common way of doing this is to give every object a unique ID. While this requires some mechanism to ensure that the ID is unique, in the Internet space this is usually made easier by the ability of an organization to prefix the identification token with their Internet domain name, avoiding cross-organization naming problems. In order to employ this strategy a new attribute must be added to all of our model objects, so we add an `id` attribute to the `ModelType` object. This allows the requesting client to get at the `id` attribute of the model, which is needed to make a full model request using the `GetDataSourceByID` message of the `GetDataSource` operation.

Code Listing 4-6: Service description (WSDL) for data publishing

```
<definitions name="DataPublishing">
<xs:import namespace="http://web.mit.edu/pamml"
          location="http://web.mit.edu/pamml.xsd"/>
<types>
   <xs:schema targetNamespace="http://web.mit.edu/pamml.wsdl">
      <!-- insert elements from Code Listing 4-5 -->
      <xs:element name="GetDataListing" nillable="true"/>
      <xs:element name="GetDataSourceByID" type="xs:string"/>
   </xs:schema>
</types>

<interface name="PublishDataInterface">
   <operation name="QueryData" pattern="http://www.w3.org/2003/11/wsdl/in-out">
      <input message="tns:GetDataListing "/>
      <output message="tns:GeoDataModels"/>
   </operation>
   <operation name="GetDataSource" pattern="http://www.w3.org/2003/11/wsdl/in-out">
      <input message="tns:GetDataSourceByID"/>
      <output message="tns:GeoDataModel"/>
   </operation>
</interface>

<binding name="PublishDataSOAPBinding" type="tns:PublishDataInterface">
   <soap:binding style="document"
                 transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="QueryService">
      <soap:operation soapAction="http://www.scituate.ma.us/QueryService"/>
      <input>
         <soap:body use="literal"/>
      </input>
      <output>
         <soap:body use="literal"/>
      </output>
   </operation>
   <operation name="GetData">
      <soap:operation soapAction="http://www.city.us/DataService"/>
      <input>
         <soap:body use="literal"/>
      </input>
      <output>
         <soap:body use="literal"/>
      </output>
   </operation>
</binding>

<service name="PublishDataService">
   <documentation>Geospatial data accessible from this server</documentation>
   <endpoint name="DataServiceURL" binding="tns:PublishDataSOAPBinding">
      <soap:address location="http://www.city.us/DataService"/>
   </endpoint>
</service>

</definitions>
```

Sharing data in multiple formats

In Code Listing 4-6 we did not explicitly define a mechanism for publishing the same data set in multiple formats. We only devised a way to publish multiple data sets. Those data sets *could* represent the same data, but it would be nice to have a way to make this relationship explicit. The concept that an output data source is really one concrete representation of some abstract data object is an important one, though. The unique ID just discussed pertains to one particular concrete instance of the data—a Shapefile, PostGIS source, etc.—not the underlying data model, which should be described aside from its output format. For our data modeling efforts, this means that any object that outputs data should have some internal representation of spatial data, as shown in Code Listing 4-7, where `ShapefileWriter` and `PostGISWriter` now have an internal `GeoDataType` object. If an organization published a data set in Shapefile and PostGIS formats, this internal object could be the same (have the same ID), although the `ShapefileWriter` and `PostGISWriter` objects would have different IDs (and would rightly be semantically different objects). The information modeling tools required to design this structure are readily available in the XML language, making it easy to add this level of inheritance, indirection and nesting.

Code Listing 4-7: Modeling spatial data output

```
<xs:complexType name="ShapefileWriterType">
    <xs:complexContent>
     <xs:extension base="GeoDataType">
        <xs:sequence>
           <xs:element name="ShpFile" type="xs:anyURI"/>
           <xs:element name="DbfFile" type="xs:anyURI"/>
           <xs:element name="ShxFile" type="xs:anyURI"/>
           <xs:element name="DataSource" type="GeoDataType"/>
        </xs:sequence>
     </xs:extension>
   </xs:complexContent>
</xs:complexType>

<xs:complexType name="PostGISWriterType">
    <xs:complexContent>
     <xs:extension base="GeoDataType">
           <xs:sequence>
           <xs:element name="User" type="xs:string"/>
           <xs:element name="Passphrase" type="PassphraseType"/>
           <xs:element name="Host" type="xs:anyURI"/>
           <xs:element name="Port" type="xs:int"/>
           <xs:element name="Driver" type="xs:string"/>
           <xs:element name="DataSource" type="GeoDataType"/>
        </xs:sequence>
     </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Some practical considerations

In addition to creating a Web services framework, the research agenda included prototyping applications that implement the services (presented in Chapter 7). From this experience, a number of issues emerged that did not arise in the pure data modeling exercise. These do not have a direct significance to any planning problem, but were crucial in designing a language from which applications could be developed. These features must be presented now for the upcoming code examples and graphics to make sense.

Spatial data typing issues

Most important is that the abstract concept of spatial data has little use in application development. GIS software is designed to work primarily with one of two types of spatial data, vector and raster. Furthermore, the overwhelming majority of vector data formats model spatial objects as a set of geometry objects (one of the seven "simple features" defined in the OpenGIS Consortium Abstract Specification) linked to an attribute table. Raster data sets are even simpler, with each cell having only one attribute. The common models for vector and raster data sets are shown in Code Listing 4-8, Code Listing 4-9, and Figure 4-1, along with the rest of the spatial data model hierarchy used in this work.

Efficient design of a data processing application *requires* that the type of data be known beforehand. It also *helps* to know what attributes the data set has, as well as their types. Therefore we include attribute information in the `VectorDataType`'s `AtributeInfo` object. For example, a client may want to access wetlands data in conjunction with a habitat model. One simple application would be to summarize the different types of wetlands present. This would require knowing what data attribute contained the information describing the wetland type, so it is extremely helpful to advertise these features of the data set. This concept is discussed in more detail later. In fact, only the most important modeling concepts are discussed in this text. Many decisions made to facilitate practical implementations are only detailed in the full, working XML Schema in Appendix A.

Code Listing 4-8: The complete spatial data model hierarchy

```
<xs:complexType name="ModelType">
   <xs:sequence>
      <xs:element ref="Metadata" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="GeoDataType">
   <xs:complexContent>
      <xs:extension base="ModelType">
         <xs:attribute name="srsName" type="xs:string" use="required"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>

<xs:complexType name="VectorDataType">
   <xs:complexContent>
      <xs:extension base="GeoDataType">
         <xs:sequence>
            <xs:element ref="AttributeInfo" minOccurs="0">
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>

<xs:complexType name="RasterDataType">
   <xs:complexContent>
      <xs:extension base="GeoDataType">
         <xs:attributeGroup ref="rasterAttributes"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>

<xs:complexType name="ShapefileWriterType">
    <xs:complexContent>
      <xs:extension base="VectorDataType">
         <xs:sequence>
            <xs:element name="ShpFile" type="xs:anyURI"/>
            <xs:element name="DbfFile" type="xs:anyURI"/>
            <xs:element name="ShxFile" type="xs:anyURI"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>

<xs:complexType name="ASCIIIntegerGridReaderType">
   <xs:complexContent>
      <xs:extension base="RasterDataType">
         <xs:sequence>
            <xs:element name="DataFile" type="DataFileCompressable"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

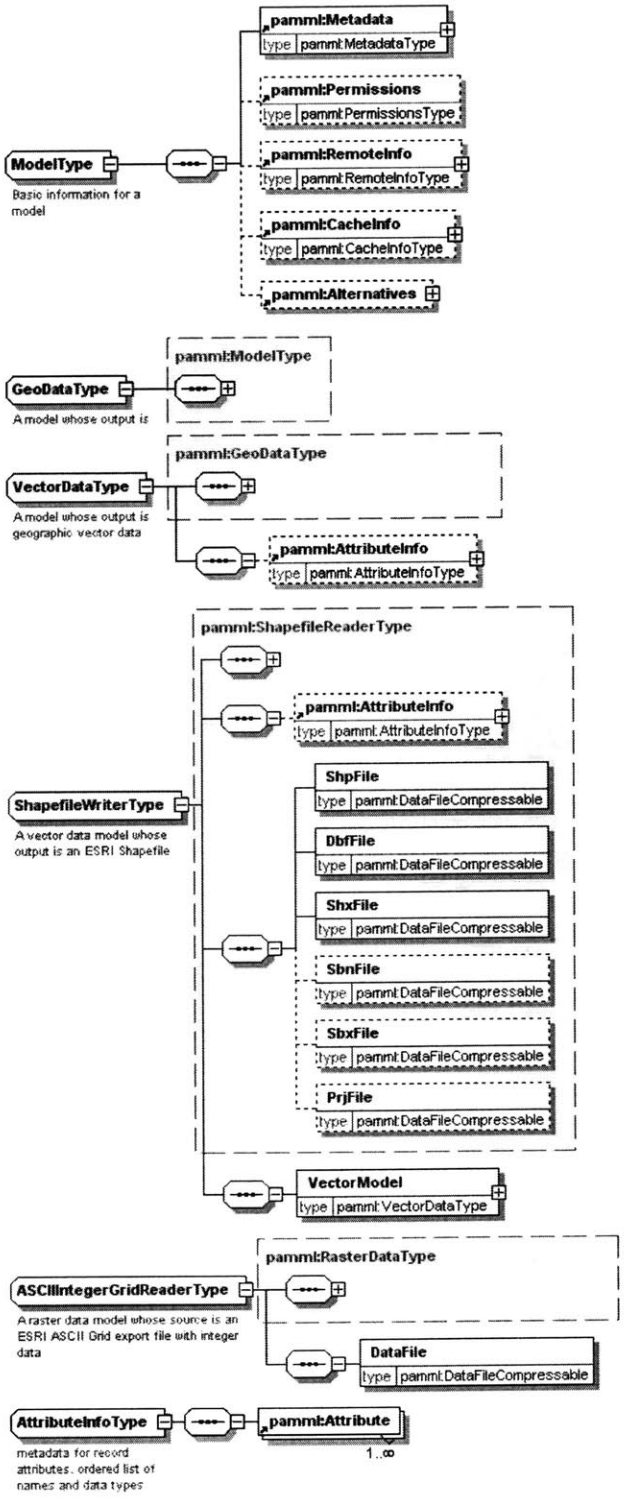
Code Listing 4-9: Spatial and tabular data feature definition

```
<xs:element name="AttributeInfo" type="pamml:AttributeInfoType"/>

<xs:complexType name="AttributeInfoType">
   <xs:sequence>
      <xs:element ref="pamml:Attribute" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>

<xs:element name="Attribute">
   <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="dataType" type="xs:anySimpleType" use="required"/>
      <xs:attribute name="minVal" type="xs:string" use="optional"/>
      <xs:attribute name="maxVal" type="xs:string" use="optional"/>
      <xs:attribute name="query" type="xs:string" use="optional"/>
   </xs:complexType>
</xs:element>
```

Figure 4-1: Common spatial data objects



Performance issues

One of the biggest shortcomings to distributed systems is the tremendous difference in performance between a desktop application using hard drive-bound data, and an Internet-based application. Whether or not this is actually the case, people seem to be uncomfortable with the idea that the data underlying their work is out of their control. They may not articulate their feelings in this way, but it was felt that to have widespread acceptance, a key design feature of this Web service-based framework would be to offer the benefits of both systems. At the simplest level, the language describes information processing in a fully distributed manner. However, there are objects built into the language that provide "hooks" that software developers can use to implement the system in such a way that all data and models are stored locally on the user's computer. We can still take advantage of the distributed framework, by making sure the software stays synchronized with the original data sources, but users get the performance benefits of using data on their hard drive, and the peace of mind of knowing that no one can arbitrarily cut off their access to the data. This last feature does in fact have a direct planning application, in that one of our target audiences is small, community-based non-profit organizations, who often have a (real or perceived) adversarial relationship with government agencies, and are therefore not likely to adopt a system that relies completely upon a constant level of cooperation with city hall and the state house.

In order to provide users with these benefits, a few additional objects must be added to the language that will only be used by software implementers, not end users. `RemoteInfo`, in Code Listing 4-10 is the construct that provides the language hooks that

software can use to implement local caching schemes. Consider that the model being read is potentially a copy whose origin is unknown. The model may have been acquired by a Web search, or someone may have emailed it to you. In that case, you have a file sitting on your computing device (which could be a computer, mobile phone, etc). You know that your computer can not execute this model, so it must have a means of telling you how it can be executed, and this requires semantics describing the original location of the model description (the `ModelLoc` object), and the location of a computer that is able to execute the model (the `ModelRunnerLoc` object). Those two objects make distributed computing more flexible. The next object, `LocalCache`, is the one that enables the local storage of data. Notice that `LocalCache` is itself a Model, which does not need to be of the same type as the original model. This allows the implementing software to, for example, cache a complex spatial operation as a simple Shapefile, while still having the option to re-compute the analysis from the remote source when desired. This example underscores the importance of PAMML's highly decomposable design. The abstraction of a spatial processing operation into a function that outputs a vector data set, combined with the fact that any PAMML operation will output only one data set, creates a very simple basic structure, which greatly facilitates the loose coupling of distributed resources.

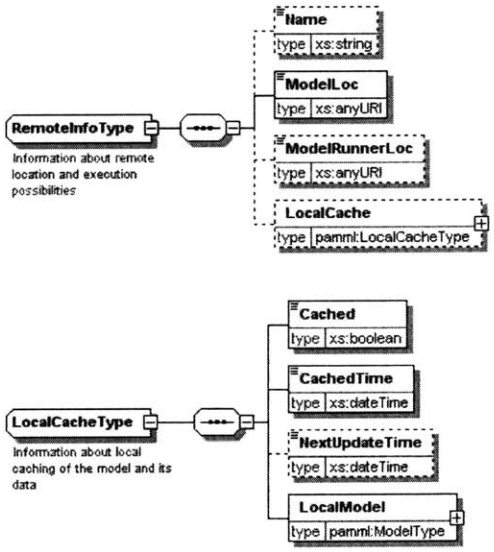
Code Listing 4-10: Objects that make distributed computing perform like desktop computing

```
<xs:element name="RemoteInfo" type="RemoteInfoType"/>

<xs:complexType name="RemoteInfoType">
   <xs:sequence>
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
      <xs:element name="ModelLoc" type="xs:anyURI"/>
      <xs:element name="ModelRunnerLoc" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="LocalCache" type="LocalCacheType" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="LocalCacheType">
   <xs:sequence>
      <xs:element name="Cached" type="xs:boolean"/>
      <xs:element name="CachedTime" type="xs:dateTime"/>
      <xs:element name="NextUpdateTime" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="LocalModel" type="ModelType"/>
   </xs:sequence>
</xs:complexType>
```

Figure 4-2: RemoteInfoType and LocalCacheType object diagrams



This chapter has laid out a strategy for addressing one of the primary causes of high information management costs, the process of moving data sets from producers to users and into analysis systems with a minimum of human intervention. In the past decade or so, we have made great strides in our ability to distribute data efficiently. Most data are

stored in electronic format, and content encoding formats are standardized enough so that translation is more of an annoyance than a real barrier to use. What we have not addressed until now is the orchestration of the process to the level of detail where human intervention can be replaced by computer-to-computer negotiation. This not only achieves significant cost reductions through automation—replacing expensive human resources with cheap computing cycles—but also creates the opportunity for new levels of efficiency, and better systems. For example, this architecture permits software to be developed that runs a quick analysis based on locally cached information resources, or a slower, more thorough one that reaches out to remote warehouses to make sure it is using the most up-to-date data. In the next chapter we build upon this methodology, adding analysis to the data sharing framework.

Chapter 5. Web Services for Collaborative Modeling and Decision Making

The data publishing framework presented in the last chapter is one example of a more general strategy, which in computer programming is called the *adapter design pattern.* When computing systems need to interoperate with each other, they usually only need to know a few things about each other; they do not need to understand each other's entire realm of functionality. Therefore, it often makes sense to create a connection object that *encapsulates* a program's functionality into the few key parameters that other systems might be interested in. This connection object is called an adapter. This chapter relies heavily on the concepts of adapters and encapsulation to extend the data sharing framework into a system that integrates that data across systems for analysis, decision support and participation.

Computing design patterns for distributed Web services

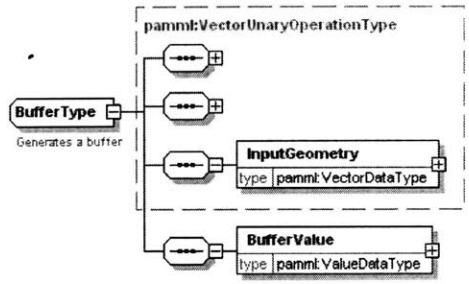
In the previous chapter, data were presented as abstracted, stylized *models* of real-world phenomena and processes (Keller 1999). Data and analytic models are often thought of as being different concepts, but this distinction is false. More generally, a model is a simplified description of a complex entity or process. It could represent a number, a spatial data set, or an analysis that predicts population growth. The practical difference between data models and analytic models is that the latter can usually be described algorithmically and therefore be reproduced by computers. One might even

say that what we call data are models for which we either have not yet discovered the algorithm, or algorithmic detail is not relevant to the problem at hand.

One example is rainfall. Say that the federal government wants to modernize the National Weather Service (*http://www.nws.noaa.gov*). A gardener might be interested in having access to a service that told them how much rain is likely to fall, but they would have little use for the meteorological model that underlies the rainfall forecasts. In this case, the gardener only needs the rainfall model to be published as data. A corporate farmer, however, might be very interested in the details of the model, and have the resources to integrate it into an internal production forecast model. But they in turn probably would not need access to the level of detail that a NOAA scientist would want whose job was to re-calibrate the model.

This can be better explained with a simple municipal planning exercise. Our task is to design a linear park that runs along an urban river. We need to give the landscape designers a plan for allocating space to various activities, including walking and bicycling. As a starting point, our plan is to preserve a 50-foot buffer of natural vegetation alongside the water, then have a 15-foot wide path for pedestrians, and finally a 25-foot wide path for cyclists. A generic model of a buffer is shown in Figure 5-1 and Code Listing 5-1. This plan can be described by three spatial buffer operations. In this model the computational process of creating buffer areas around geometries is hidden, or encapsulated. All that is made explicit are the required inputs—a spatial data set and a buffer distance—and the single output—a new spatial data set representing the buffer areas. The actual plan is shown as a map in Figure 5-2, in XML form in Code Listing 5-2 (some attributes, like `srsName` and `id`, are omitted in this example for illustrative

Figure 5-1: Model of a spatial buffer operation



Code Listing 5-1: Model of a spatial buffer operation

```
<xs:element name="Buffer" type="pamml:BufferType"/>

<xs:complexType name="BufferType">
   <xs:complexContent>
         <xs:extension base="pamml:VectorDataType">
            <xs:sequence>
               <xs:element name="InputGeometry" type="VectorDataType"/>
               <xs:element name="BufferValue" type="ValueDataType"/>
            </xs:sequence>
         </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

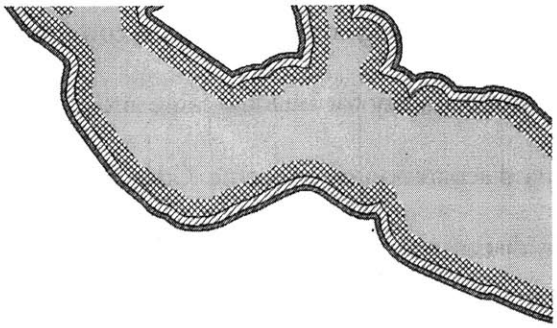
purposes), and as a diagram in Figure 5-3. The adapter design pattern allows the linear park plan to be *composed* of three buffer models, with each being an input to the next.

Composability is the ability to put together a piece of software from several components. Think of Lego™ building blocks and you have a good idea of how powerful and intuitive are systems exhibiting strong composability. In computer science, this is generally thought to be an essential property for building large and complex systems as it enables modularization and separation of concerns. Composability is made possible by the use of design patterns discussed earlier, such as *inheritance* (all vector spatial data types are descendants of a single generic type), *adapters*, and *encapsulation*. In systems distributed across computers and organizations, modularization and separation are more than critical; they are basic requirement. It is extremely elegant, therefore, that

composability not only enables the development of a distributed system, but also can be made to mirror organizational specialization.

Take the linear park model as an example. Code Listing 5-2 and Figure 5-3 present a site planning model composed out of three buffer models, with spatial data passing from one buffer model's output to the next one's input. But what format is the data in? This is not specified, so this model must be executed on a single computer system. In that case, there is no need to specify concrete data formats, allowing the software to choose its own preferred internal format (this is an important feature for commercialization, as it allows software companies to differentiate themselves, and charge a premium, based on their ability to execute algorithms well, even if all software packages are using a common language to describe the algorithm). In many cases, however, the system will not reside on a single computer, but will be distributed across multiple agencies. For argument's sake, let us assume that the river boundary data comes from the USGS; the extent of the natural buffer around the river is determined by the state department of environmental protection, and the recreational paths are set by the municipal parks department.

Figure 5-2: Linear park model, cartographic visualization



Code Listing 5-2: Linear park planning model

```
<Buffer name="Pedestrian path">
  <InputGeometry name="Bike path" type="BufferType">
    <InputGeometry name="Natural area" type="BufferType">
      <InputGeometry name="River" type="ShapefileReaderType">
        <ShpFile
          dataFile="http://www.usgs.gov/data?type=hydro&amp;fips=4&part=shp"/>
        <DbfFile
          dataFile="http://www.usgs.gov/getdata?type=hydro&fips=4&part=dbf"/>
        <ShxFile
          dataFile="http://www.usgs.gov/getdata?type=hydro&fips=4&part=shx"/>
        <SbnFile
          dataFile="http://www.usgs.gov/getdata?type=hydro&fips=4&part=sbn"/>
      </InputGeometry>
      <BufferValue name="Natural area extent" type="SimpleIntValue"
        units="feet" value="50"/>
    </InputGeometry>
    <BufferValue name="Bike path right-of-way" type="SimpleIntValue"
      units="feet" value="25"/>
  </InputGeometry>
  <BufferValue name="Pedestrian path right-of-way" type="SimpleIntValue"
    units="feet" value="15"/>
</Buffer>
```

Figure 5-3: Linear park model, diagrammatic visualization

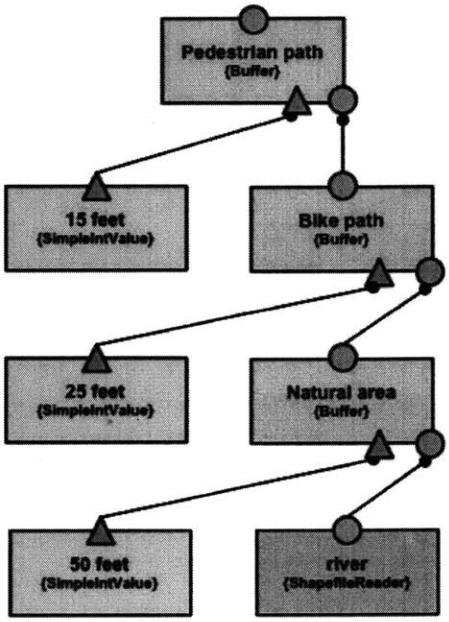


Figure 5-4: Linear park model, distributed across agencies

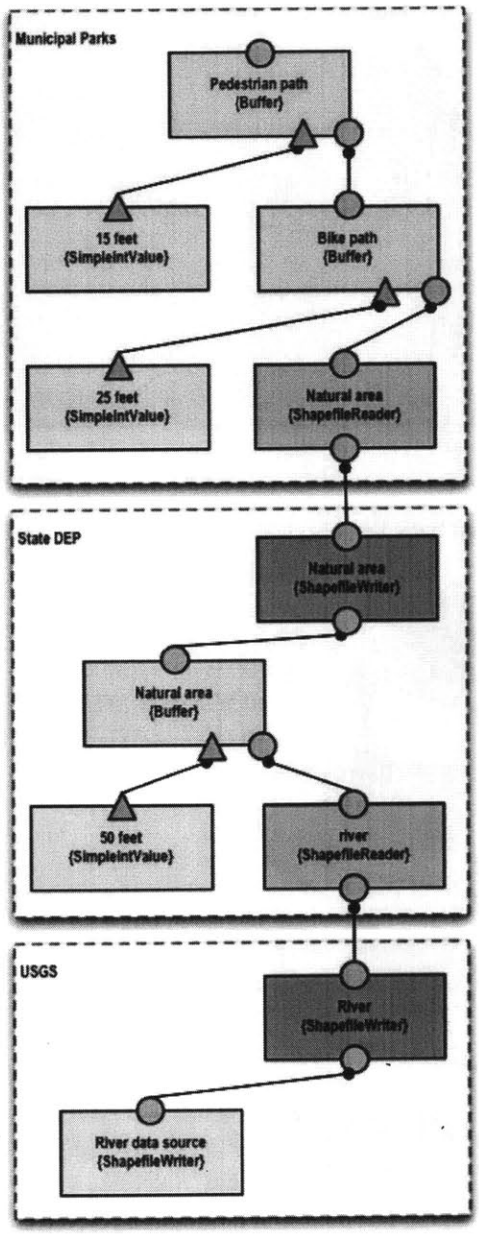
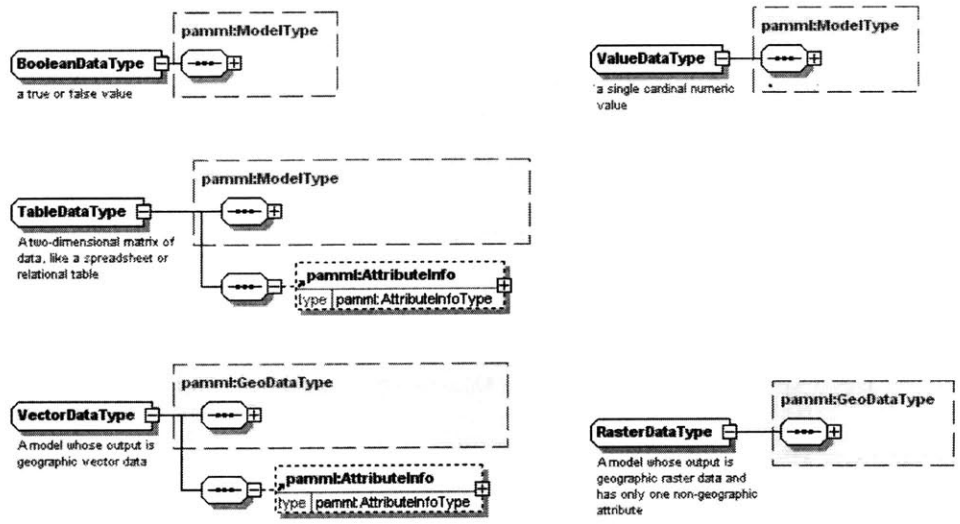


Figure 5-5: Some basic data models



In this scenario, we must use the adapter design pattern to guarantee interoperability across potentially heterogeneous systems in the various agencies. Figure 5-4 illustrates how our language implements the adapter pattern through data "readers" and "writers." While data is processed within a system, operations can be described abstractly, as in Figure 5-3 above. But whenever data moves from one system to another, a Writer adapter must be used on exit, and a Reader adapter must be used on entrance. In this way, most systems can be integrated into a distributed computing environment, as long as we can agree on a few basic data types. Some are shown in Figure 5-5. Note that even a simple data type like an integer is descended from the `ModelType` object. This has practical benefits in that this allows the value to acquire all the nice features of a model object, like metadata. More importantly is the semantic meaning. Even simple numbers are "abstracted, stylized models of real-world phenomena and processes." When the U.S. Environmental Protection Agency says that X parts per billion of heavy metals in a fish is not hazardous to an adult's health, X is not simply a number. It's a complex model.

Supporting legacy, or "black box" systems

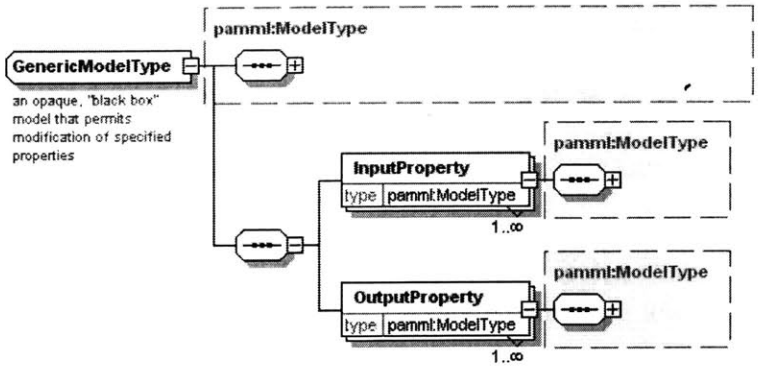
Not all models are as simple to describe as a buffer. It probably does not make sense to create a universal language that describes complex, scientific models in minute detail. There is little to gain and much to lose as there are probably many good reasons why expert domains have their own, unique discourse. What is more useful is to recognize that while core parts of an analytic model might always be a "black box"—indecipherable to all but a small group of experts—significant parameters may still be exposed to the computer systems of other modelers (like the corporate farmer mentioned earlier), and to the intellects of human decision makers. Therefore, the most important model in our system is the *GenericModel* (Figure 5-6). The *GenericModel* is important because it provides a quick way for organizations with "legacy" systems to participate in the new distributed framework without making major changes to their business processes. The drawback is that a system defined using generic models has less semantic meaning inherent in its description than others, and is therefore more difficult to integrate into collaborative analysis or decision support systems.

Code Listing 5-3: GenericModel XML Schema

```
<xs:element name="GenericModel" type="GenericModelType"/>

<xs:complexType name="GenericModelType">
  <xs:complexContent>
    <xs:extension base="ModelType">
      <xs:sequence>
        <xs:element name="InputProperty" type="ModelType" maxOccurs="unbounded"/>
        <xs:element name="OutputProperty" type="ModelType" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 5-6: GenericModel, integrating legacy models



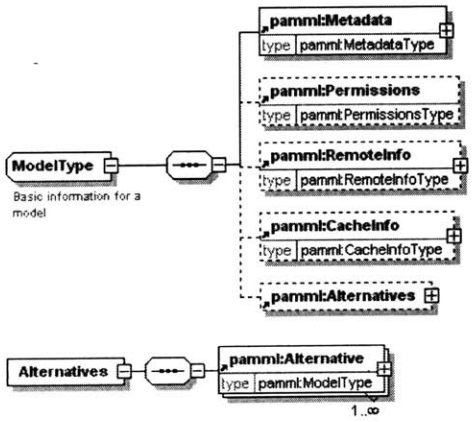
Collaborative planning: linking models with decision makers

There is a large body of work within the PSS field on participatory decision making, but the systems proposed are rarely integrated with the system used by the experts. This calls into question those systems' ability to truly capture feedback. In fact, the fault lies at both ends of the process because information systems rarely have the ability to capture and store any kind of debate around an analysis' results or techniques. So while researchers have experimented with effective systems that help explain complex phenomena to non-expert audiences, and enabled these audiences to play out scenarios that use alternative weights and values, the results of these experiences are generally captured *outside* of the original information system, in some form of human communication to the "experts." Although it would be hard to find a researcher in this field who did not think "feedback loops" were of critical importance, it would be just as hard to find software that actually implemented what could be called a feedback information system—a system for the storage, retrieval and analysis of discussion and debate around a planning model.

While the concept of a feedback information system is compelling, and is probably a necessary step in the evolution of participatory PSS, a thorough treatment of the topic is beyond the scope of this paper. For example, the types of feedback are numerous, from anecdotal commentary to survey instruments and voting. Each type of system might suggest a different information model, and might also require a different way of describing the participants, because the dynamics of public meetings are such that the type of participant matters as much as the issues being discussed. Here we seek to take a small step towards such a system by defining some structure in which to express the idea that what often happens in participatory PSS is that stakeholders want to explore "what-if" scenarios by substituting alternative values for a model's initial parameters. This one example will illustrate that the language has the ability, in general, to work in concert with potential participatory planning information systems.

Modeling systems usually do a good job of allowing the *analyst* to explore different scenarios by changing parameters, but they pay little attention to preserving this information. And if they do, they usually take an engineering approach, saving model runs in a scripting language or some other shorthand geared to be an efficient means of

Figure 5-7: Capturing feedback in the information system



restarting the modeling process. The approach espoused here is more of an information modeling approach, geared towards storing different opinions in a manner suited to visualization and analysis of the debate. Figure 5-7 shows the `ModelType` object with a new object called `Alternatives`. This object may contain an unlimited number of models that may be considered alternatives to the enclosing object. Software that implemented the `Alternatives` object would need to make sure that the output data type of the alternative models matched that of the enclosing object (XML Schema has no elegant way of articulating this constraint), but aside from that there is wide range of possibilities a software package could exploit using the `Alternatives` object. There are many other important issues to consider when designing a participatory information system, but they are not unique to the language being developed here, and are better handled in a more general collaborative computing research agenda.

This chapter has presented some features of PAMML most relevant to the challenges found in the buildout analysis. PAMML includes a number of additional objects and operations, which can be examined in the full XML Schema in Appendix A. Some of these, like `Union`, `Intersection`, `Difference`, and `Dissolve`, fill out the language's library of spatial operations. We have mainly discussed operations involving vector data, but the schema includes enough basic raster data types and operations to implement map algebra. Finally, some others add "inline" data types, which are XML-based descriptions of a data set, such as a table or a number. This simply allows the data to be included in the model, instead of requiring a remote reference.

Now that the PAMML framework has been developed, in the next chapter we return to the information management challenges that motivated this work, and we use these tools to reconstruct the buildout analysis, and show how the new framework reduces the cost and complexity of information processing.

Chapter 6. Prototyping the Buildout Analysis

An important test of the PAMML services framework advocated by this paper is its ability to model the trial case presented in Chapter 3, the MassGIS buildout analysis. This empirical experiment is presented here. By referencing the problems identified in Chapter 3 and detailing how the PAMML framework addresses them, we are able to argue that PAMML not only is able to reproduce the types of analyses commonly performed by physical planners, but is also able to address the high costs of collaborative information management and processing. In this way we go beyond the basic argument, common in many disciplines, that says that the use of Web services has proven to reduce costs; therefore if we can rebuild our traditional planning support tools on top of a Web services architecture, we will naturally reduce costs in the planning discipline. This argument is persuasive, but one could argue that the planning discipline exhibits unique characteristics that prevent it from benefiting from the adoption of technologies from other fields. By explicitly addressing the information management problems exposed in the buildout analysis, we greatly strengthen the case for PAMML.

Zombie data

Recall the concept of zombie data developed earlier. This is data that are acquired from its maintainer, then used for months or years, and perhaps modified with local knowledge, with little consideration for the changes the maintainer may have made over that time. These data are dead in that they have been disconnected from their living, up-

to-date sources. Yet they are also alive because their owner is still finding them useful. The zombie data problem is at the core of the information management cost dilemma. People have come to expect applications that are lightweight and Internet-aware to have limited functionality, like the online EOEA buildout tool mentioned earlier, that aggregates statistics for multiple towns. They have been conditioned to believe sophisticated, feature-rich analysis tools like GeoVista or ArcView will depend mainly upon local data sources, and that the data management problem is external to the analysis software.

This is the key problem with MassGIS' buildout strategy. They provide excellent analysis tools, in the form of ArcView and Excel. They also provide a system for automating analytic processing in the form of ArcView and Excel macros, called the Buildout Analysis Toolkit. What they do not attend to is the data management question. This would be fine if information management was not central to the ongoing usefulness of the analysis. If the data rarely changed the cost of doing things differently would be out of proportion to the benefits. But this is not the case. Planners do want to continuously plan—they just have no feasible options to make this cost-effective. Therefore our task is to deliver the PAMML framework at reasonable costs.

In earlier chapters we discussed the issue of technology sizing. The costs of implementing a system should be heavily weighted towards the beginning of a project, when one-time funds are allocated and project advocates are energized. Ongoing costs must be as low as possible. Otherwise the technology infrastructure will disintegrate from lack of maintenance. To achieve these low costs, planning IT infrastructure must utilize general IT infrastructure as much as possible. Here we go into deeper detail,

showing an operational model of how the PAMML framework addresses the zombie data syndrome with close attention paid to the technology sizing issue. Table 1 lists a cost/effort matrix for three different data management strategies. The first, "Send data," is the most traditional, involving a data maintainer sending mailing or emailing a data set to each user. When the data changes, the entire process must be repeated. In the second strategy, "Publish data: Web site," which is the current state-of-the-art, the data maintainer uses the Web to avoid sending updates to each and every user. She instead updates one copy of the data on a Web site, then informs users so that they can download it. This strategy has proven to be a great time-saver in that the maintenance agency no longer has to handle requests for data—the Web is a self-service system—but the users' costs have not been addressed.

Table 1: Data publishing system designs

	Send data	**Publish data: Web site**	**Publish data: PAMML service**
(publisher) step 1	**Extract from operational system**	**Extract from operational system**	**Extract from operational system**
(publisher) step 2	**Copy to media**	**Copy to Web site**	**Copy to Web site**
(publisher) step 3	**Publicize updated data availability**	**Publicize updated data availability**	
(publisher) step 4	**Process data requests**		
(publisher) step 5	**Send media**	Design-Publish Web page	Code-Publish WSDL, XML
(user) step 6	**Copy from media**	**Download from Web site**	Subscribe to data service in PAMML-enabled software
(user) step 7	**Import into operational system**	**Import into operational system**	**Update local cache of the service**
Maintenance steps (bold)	Repeat 1,2,3,4,5,6,7	Repeat 1,2,3,6,7	Repeat 1,2,7

The PAMML strategy requires many of the same initial publishing efforts, but then the software takes care of ongoing updates between data publishers and users. As described in Chapter 4, the simplest data publishing technique PAMML offers is much like posting data files on a Web site. The main difference is that instead of designing an HTML Web page to complement the data file, the publisher designs a PAMML WSDL (Web Services Description Language) file and a PAMML data instance file. To make use of the data, a user "subscribes" to the data service, and from that point on, the user's software is able to create a local copy of the data set (to maximize performance), and periodically check back with the original data publisher for updates. This reduces the burden on users *and* publishers, minimizing the ongoing, operational cost of information management. The costs of keeping the data up to date are shifted to the software design and development stage, where they can be spread over thousands of users, instead of having thousands of users each develop their own individual solutions.

PAMML also addresses another type of zombie data problem. Data sharing often occurs without the knowledge of the official data maintainer. In addition to the data being more likely to be out of date, this leads to situations where data may be used in ways for which it was not originally intended. Addressing concerns like these motivate the data cataloging work of agencies such as the FGDC (`http://www.fgdc.gov`). In the PAMML framework, the data description file is shared (Code Listing 6-1), not the data. The new user takes this XML file and uses it to subscribe to the data service directly from the publisher. This serves two purposes. First, the new user is getting the latest version of the data. This is a nice feature, but the real significance of this strategy is that users are not passing data sets around. In effect they are passing along *a contract* to engage

with the data publisher. When the new user attempts to access the data, the publisher has the opportunity to decide whether or not to "do business" with that user. If the data are public and open, nothing important happens at this stage; it is simply sent to the user. However, if the data are sensitive in some way, the publisher would at this point check the user's credentials, and act accordingly. If, of course, users do not want to intentionally subvert the system, they will choose to use the PAMML framework over the old ways because, as just discussed, PAMML is cheaper and easier. And in doing so, we strengthen the contractual relationship—social, technical, or business—between data publishers and users.

Code Listing 6-1: XML instance document for Shapefile publishing

```
<ShapefileWriter srsName="EPSG:26986">
   <ShpFile dataFile="http://www.city.us/wetlands.shp"/>
   <DbfFile dataFile="http://www.city.us/wetlands.dbf"/>
   <ShxFile dataFile="http://www.city.us/wetlands.shx"/>
</ShapefileWriter>
```

In the case of an isolated data set, the idea of a contract between publisher and user seems trivial. It becomes much more significant when discussing a real model like the buildout analysis, where a number of contractual issues could arise. Is the client using server processing resources? If so, should we allow this? Are they a public agency, an individual, or a land developer? Should we charge for-profit enterprises for access? All these issues have technical solutions, and PAMML applications, by virtue of their adoption of Web services, are likely to be able to respond to them cheaply, because they can use generic authentication and security techniques designed for any Web service, instead of inventing new systems for government or planning.

The zombie data discussion started with the goal of reducing the costs and complexity (effort) of keeping data up to date. We have just shown how PAMML can solve that problem, but there is a more general issue to address. As stated earlier, there is no real difference between a data set and a model. A data set could be thought of as a concise summary of some analytic process. Therefore, any solution to the zombie data problem should also apply to analytic models. In fact, this is the case. Recall that in a PAMML framework, the user gains access to the data by subscribing to a PAMML service. The details of this subscription were contained in a PAMML XML data instance file. It would have been more accurate to call that a model instance file, because we know that PAMML does not distinguish between the two. So in the PAMML architecture—from the user's perspective—there is no difference between accessing a data set and accessing a complex model. However, from the publisher's perspective, the difference may be great. If the publisher's intent is to provide interactive access to the model, then the publisher probably needs more than a simple Web server to achieve this goal. They must first describe the model in PAMML. The simplest way to do this is to use the `GenericModel` object, which allows one to give the model a name, then describe its inputs (Figure 6-1) and outputs (Figure 6-2). Then they must implement some sort of PAMML-enabled data processing software so that users can change model parameters and run their own analyses. This more complex system is well within the capabilities of agencies, like MassGIS, who may desire them, so it is believed that PAMML offers a good match between the sophistication of agency needs and the required sophistication of their IT infrastructure.

Figure 6-1: Buildout model inputs (a representative sampling)

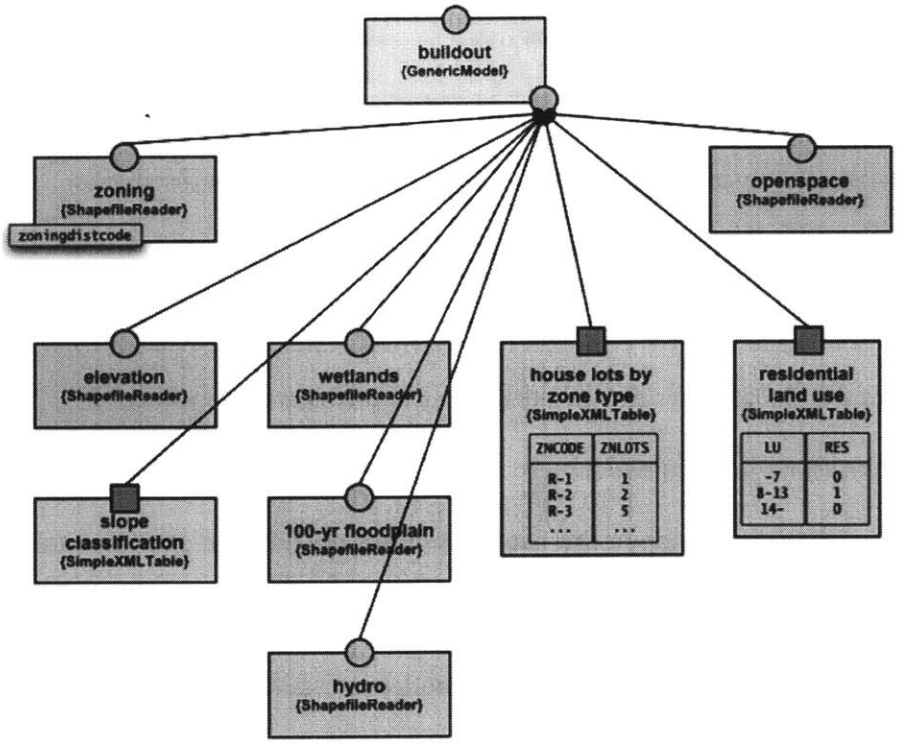
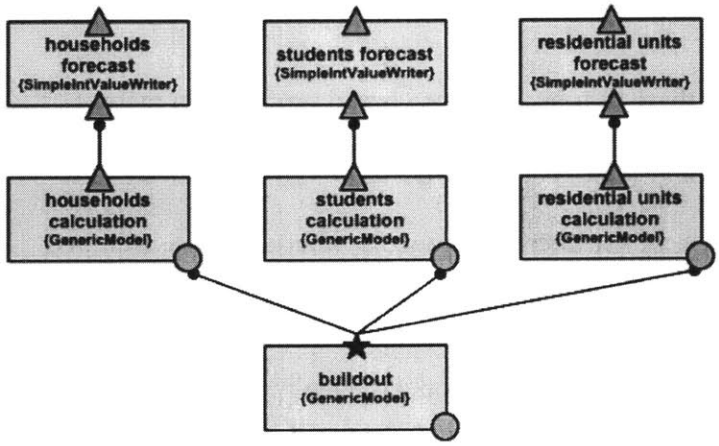


Figure 6-2: Buildout model outputs (a representative sampling)



A NOTE ON GRAPHIC CONVENTIONS: This chapter includes a number of box diagrams like those shown here. These diagrams show the process of performing analytic operations to create new data sets, which are in turn used in the next stage of operations. The diagram should be read from bottom to top, with the upper-most box being the end result of all data processing. The boxes all have a name, and an operation type—the text in curly brackets—that corresponds to an XML element in the PAMML XML schema (see Appendix A). Note the small shapes at the top and center of each box. This shape represents the type of data output by this box. A circle represents Vector data output; a square represents a table (or 2D matrix); a triangle is a single numeric value (Boolean, integer, or decimal). A star has multiple outputs. The color-coding of the boxes provides a quick visual hint relating to the type of PAMML model as well as its output data type (color differences may be difficult to distinguish in a black and white copy of this document).

Stakeholder participation

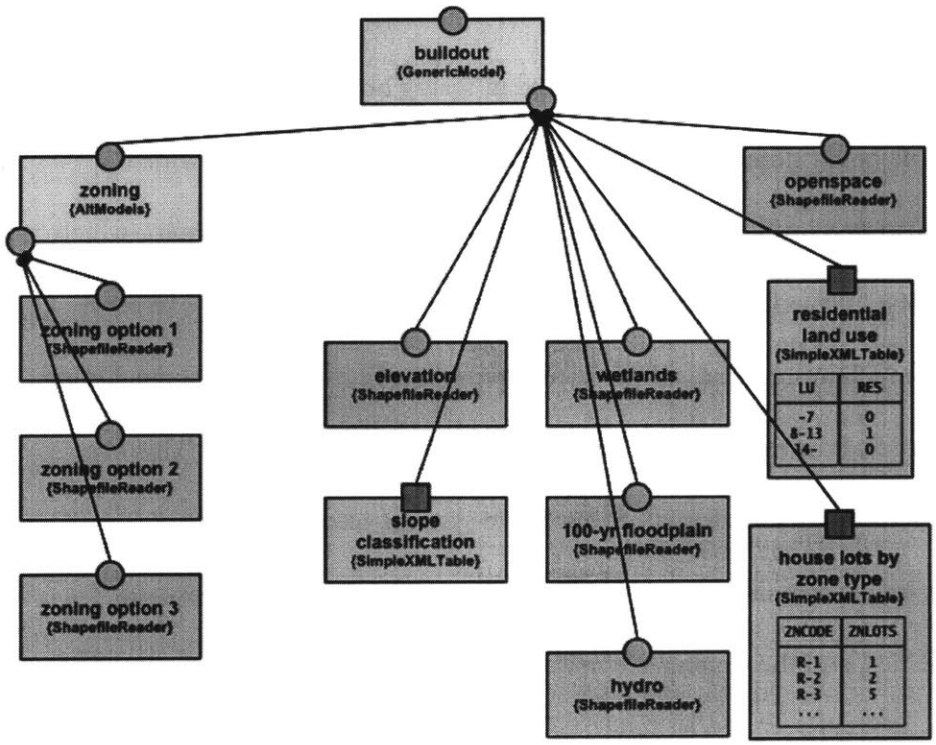
If this research did nothing but address the zombie data-model problem, it would be a success. But the PAMML framework also is able to make progress on a problem with stakeholder participation observed in the buildout analysis. There we saw a disconnect between the analysis effort and the debate, discussion, and alteration that occurs when the analysis is brought to a municipality, as discussed by Hodges (2004). While social scientists may capture this debate after the fact, this rarely happens at a time when something can be done about it. Even when there is an effort to drive new analyses, or "model runs," based on stakeholder input, this usually requires the creation of a separate system designed specifically for use in meetings, or other venues far from the analyst's workbench. Sometimes this is necessary because each model runs takes hours or days to complete, but more often it is because the modeling software is not designed to: a) be accessed outside of the office; and b) have its "data analyst" user interface be replaced by a "decision-support" user interface.

PAMML facilitates solutions to these non-performance based problems in many ways. PAMML is inherently designed to be accessed outside of one office because of its Web services roots. All operations occur via Internet protocols, whether they are limited to one computer in one office, or multiple computers scattered across the globe. The ability to apply different user interfaces to a PAMML model is even more significant. This feature is largely a result of using XML, which was designed for this purpose. The technology of how this works is discussed in more detail below.

So we see that the early design decisions to build upon standards like XML and Web services help address concerns that have often been seen as idiosyncratic of the planning

profession. But one thing we still must do ourselves is to capture public debate and discussion, and integrate it with modeling efforts. Recall that we have designed a PAMML object called `AltModel` to facilitate this. In the buildout analysis, its most obvious use would be to capture different zoning scenarios (Figure 6-3), so that zoning changes could be evaluated based on their impacts on future growth, such as changes in the number of schools required, or the increased stress on water and sewer systems.

Figure 6-3: Buildout model showing alternative zoning options



This should by no means be seen as a complete response to the stakeholder participation issue, but rather a starting point for further research. We see below how well the PAMML framework handles rich user interfaces, but the more interesting issue here is the types of information one might want to capture. For example, planners often use voting and "weighting and rating" games in participatory settings. These techniques

require an expanded information model from that presented here, but if the PAMML framework, or at least XML, is used as a starting point, the likelihood of being able to integrate the technologies is high.

Collaborative Planning

There are, of course, many more potential points of exploration and debate other than zoning regulations. For example, environmental issues are always a concern. People might debate how far away from wetlands and environmentally sensitive habitat development must be. Or they might be interested in seeing how important are the presence of multiple environmental factors in restricting the right to build. None of these issues are illustrated in Figure 6-3, because it is based on the `GenericModel`, which provides a higher level view of the analysis. We can, however, articulate these issues by modeling them in much more detail, which in turn permits a finer level of debate. Doing so brings analysis out of the modeler's workshop and into the public forum. This has always been the focus of participatory PSS, but those systems have rarely been implemented in a way that maintains a direct connection between participatory or collaborative activities, and the original analysis.

As a very basic example, recall our experience with buffer models from the previous chapter. Say that, in the buildout model, streams are protected by a 100-foot buffer zone, and this is described by a buffer model (as appears in the bottom-left corner of Figure 6-6, which will be described shortly), which is comprised of a value model and a vector data set. As full-fledged models in their own right, the buffer, the value, or the vector data could each be observed and replaced with alternatives. The result is that people with

different expertise can focus on exploring and refining different sections of the model, and this is what expert collaboration is about.

The buildout analysis was not difficult to express in PAMML, although the XML code is not easy to follow without careful study. The model is more likely to be seen using some sort of visual analysis tool, and this is how it is presented here. Remember from the earlier discussion of the buildout analysis that the general flow of the analysis follows these steps:

1. Take already developed land as-is. No redevelopment of these areas.
2. Take other areas and remove places under permanent protection from development.
3. Identify areas with partial restrictions on development.
4. Calculate maximum residential and commercial development for areas identified in step 2, and use step 3 to apply a penalty factor, arriving at a final buildout value for the area.

Figure 6-4 shows a model of step 1. Areas already developed are specified using MassGIS *landuse* GIS data and selecting out those areas whose land use identifies them as already being developed. This is the *Reclass* model named *developed, in land use database*. Note that a *Reclass* model is comprised of a vector data set, *landuse*, and a table (in purple), *MacConnell land use*. This table is used to reclassify the land use data set, mapping values of the *LU* property to values of a new property, *DEV*. In this case, the reclassification table says that for the *LU* property of *landuse*, all values equal to or less

than 7 map to a *DEV* value of 0. All values between 8 and 13, inclusive, map to a *DEV* value of 1, and so forth.

The *landuse* data is up to ten years old in some places, so it is useful to supplement it using local surveys—the *newlu* model—as well as the latest projects from developers—the *subdivisions* model. These are combined via a *Union* operation to form a model of *newly developed subdivisions*. These are in turn *Union*-ed with the older data to create a full model of developed land, which we simply call *developed*. This is the fine-grained model of developed land that the planner creating the buildout model would use. However, someone else might have no need to know all the considerations that led up to the overall conception of developed land, only the final result. In this case they would never need to look deeper than the *developed* model. At that level of detail the model looks like a vector data set with a name, which offers human users with semantic clues regarding the data set's content.[1] They would only see how that model of *developed* land was constructed if they drilled down deeper.

Figure 6-5 creates a spatial data layer containing all the various environmental conditions that could restrict development in a particular area. The final decision on how greatly they impact development is decided by human judgment, rather than an algorithm, so the purpose of this model is to do some geographic accounting. The end result of this is that the analyst has every area of the town tagged directly with its

---

[1] An information community could develop a better strategy for passing along semantic information other than the name of the model by adding a custom object within the *Metadata* object, which is a component of every *Model*.

environmental issues, so that a decision regarding development potential can be more easily made.

Figure 6-6 models lands that can not be developed for environmental reasons, including steep slopes, flood plains, and wetlands. These all go into a model of `partially developable` land. The notable characteristic of this model is the great diversity of primary data sources used. In Massachusetts, environmental data is usually acquired from MassGIS, who gets it from a federal agency, but performs some further processing to make it easier to use for regional work. Wetlands data may still be acquired from other sources, and three possible choices are illustrated in this model. In fact, MassGIS specifically discusses this wealth of choice for wetlands data, so its inclusion here is a necessity, not just a nice way to use the `Alternatives` model discussed in the last chapter.

Figure 6-4: PAMML model of developed land

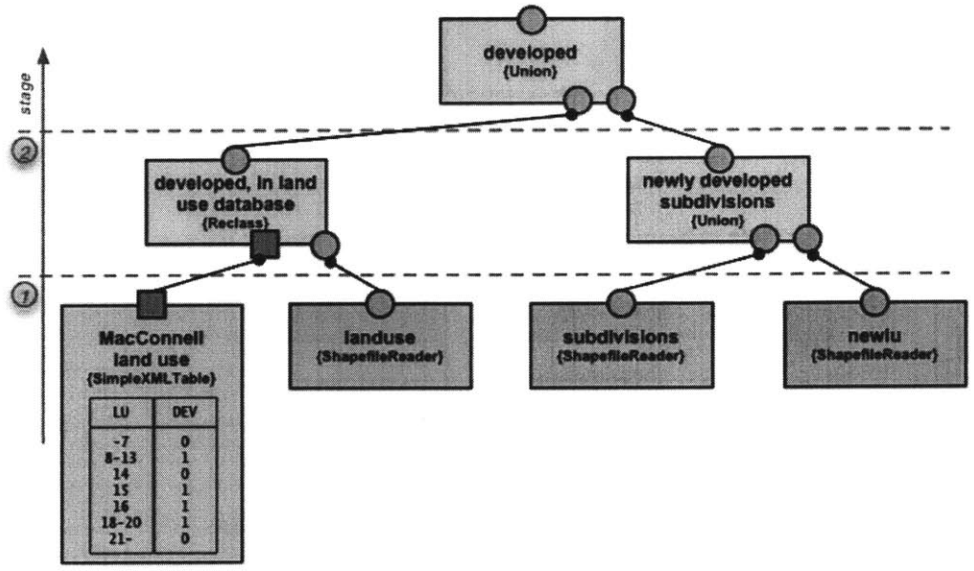


Figure 6-5: PAMML model of lands with development constraints

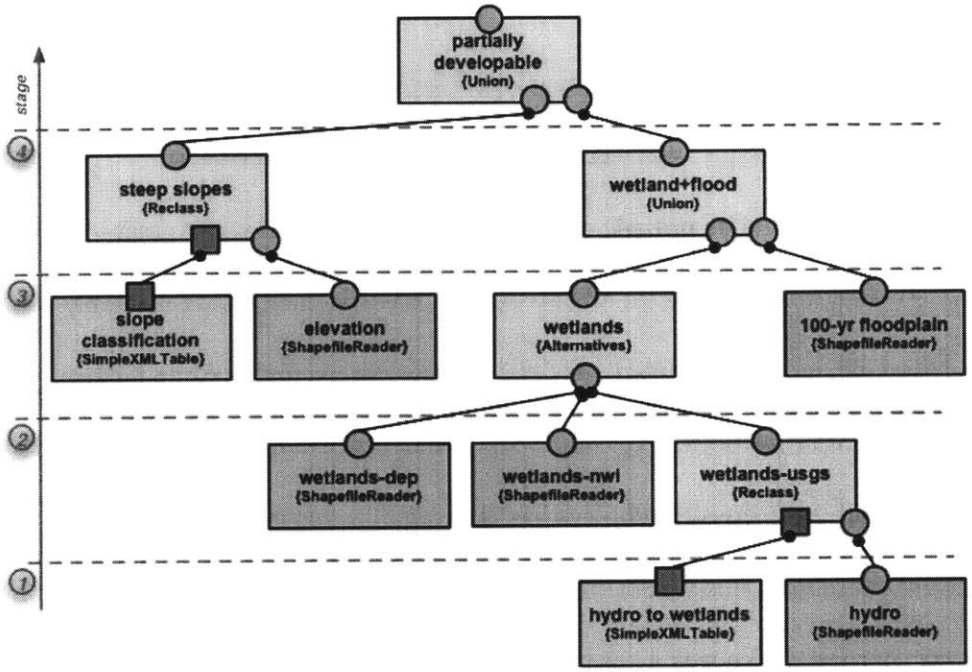


Figure 6-6: PAMML model of land that is exempt from development for environmental reasons

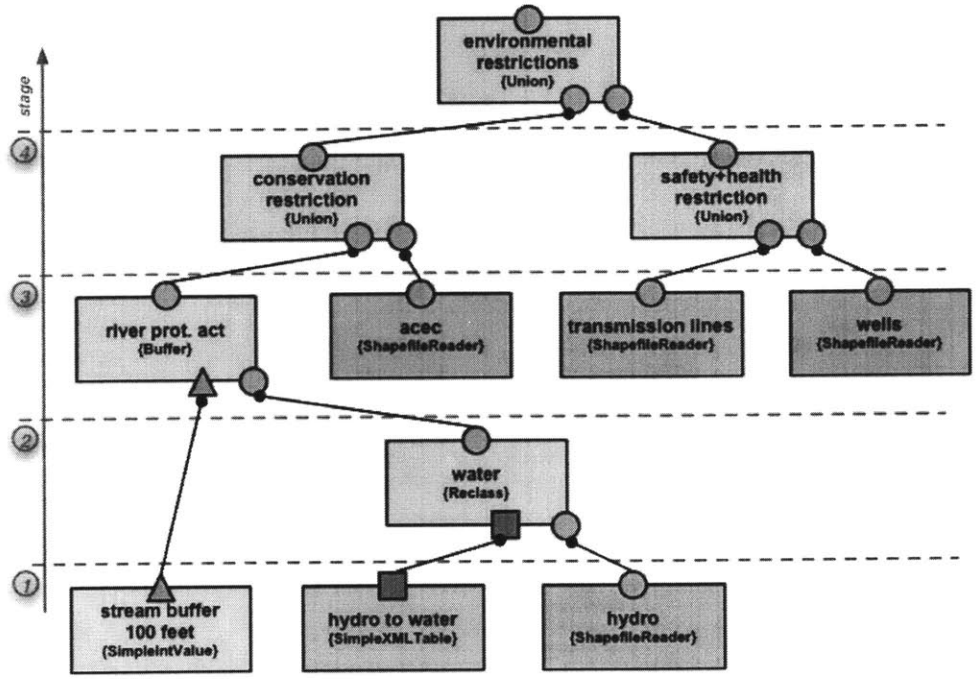


Figure 6-7: PAMML model of buildout
("partially developable", "environmental restrictions", and "developed" models are summaries of models shown above)

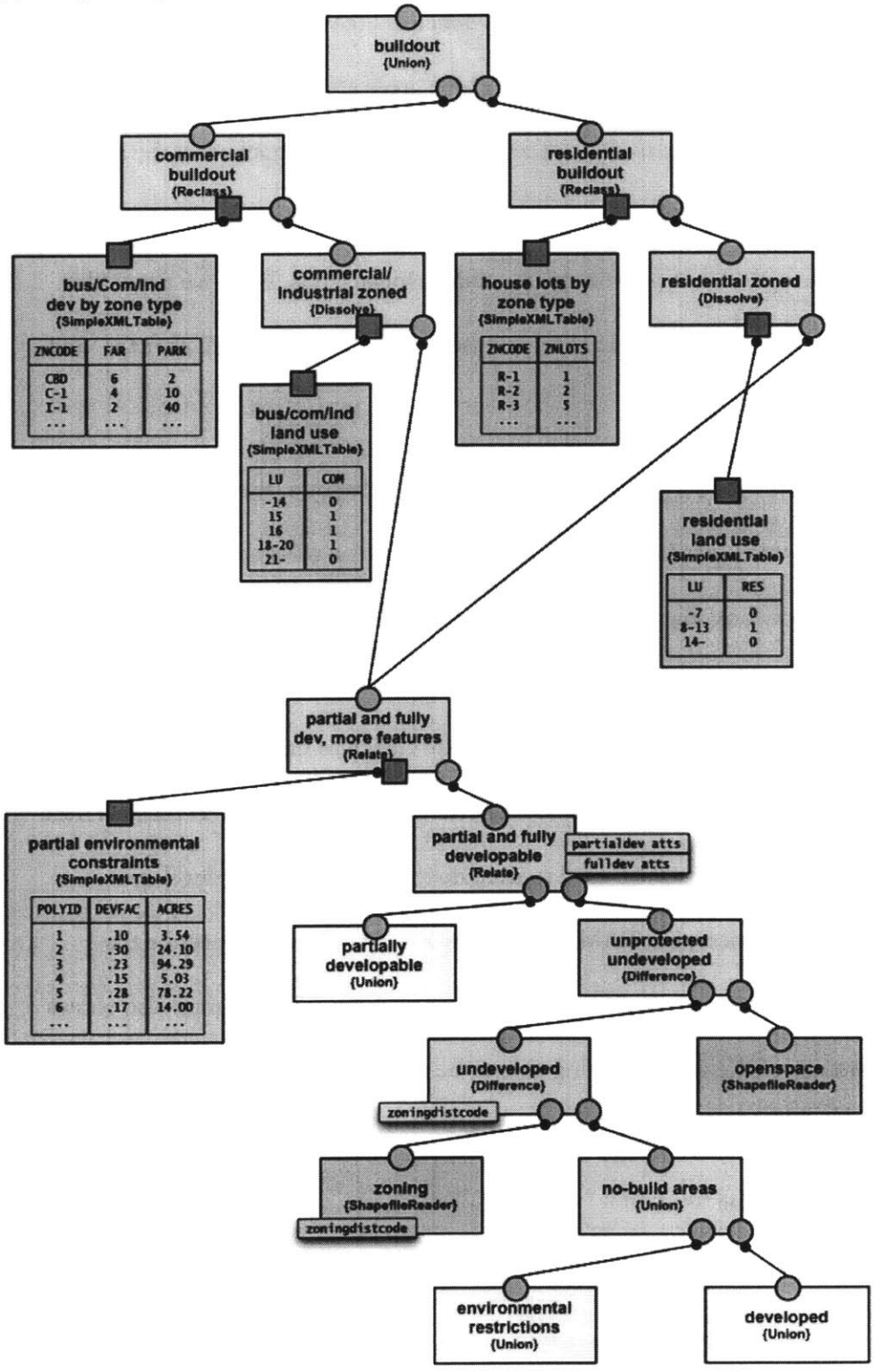


Figure 6-7 illustrates the final steps of the buildout analysis. Reading from the bottom, *developed* land and areas with full *environmental restrictions* are combined using a union model to create *no-build areas*. These areas are then subtracted from the town using the *zoning* data layer so that the zoning codes can be attached to the areas that are left in are model of *undeveloped* land. We then remove permanently protected *openspace*, to create a model of *unprotected, undeveloped* lands. Next we attach the attributes of the *partially developable* areas model to create the model, *partial and fully developable* areas.

Now we encounter the most important feature of this diagram, the reclassification, or lookup, tables. *Partial environmental constraints* is the table that an analyst would create to identify how big an impact partial constraints (from Figure 6-5) would have on development of that particular piece of land. In the MassGIS work, this step was performed in Microsoft Excel, whereas ESRI ArcView was used for the spatial operations, and this made it a bit difficult to track the analysis from start to finish. These two software programs could still be used to execute the analysis if they developed support for PAMML services, but PAMML gives us a way to formally describe the process without depending on any particular software package. This diagram also shows that this crucial stage of the analysis, being able to be represented as a simple lookup table, could easily be made available on the Web for interactive scenario generation, even if the rest of the model was more of a "black box."

Bus/com/ind land use *and residential land use* are simply used to dissolve the model of *partial and fully developable land* into residential and commercial,

because they are analyzed differently. The analyst then uses the final two lookup tables, *house lots by zone type* and *bus/com/ind dev by zone type*, to develop maximum development figures based on the density of development permitted under the zoning code. *Residential buildout* and *commercial buildout* are then merged back together to create a final, unified view of *buildout*.

Constituents want to plan continuously, not once. EOEA recognized this, and provided two avenues for further analysis. The most basic is the online application described in Chapter 3, which mainly allows a user to get aggregate statistics on multiple, neighboring communities. The more flexible option is to download all the data sets used in the analysis, and use them with one's own software (ArcView and Excel and the analysis toolkit) to create a custom buildout analysis by changing key inputs such as building setbacks, road widths, or natural resource protection buffers.

In Chapter 3 we identified the main problem with these options—they are poorly matched to their user communities. One might imagine regional planning agencies having the most need for aggregate statistics, but they are also the most likely to be planning professionals, and desire more sophisticated tools than the Web site provides. On the other hand, smaller rural and suburban towns have the most use for a system that allows them to play out scenarios based on changes to local land use regulations, yet they are unlikely to have the resources necessary to make full use of the ArcView/Excel system. And even if they did, that system is still flawed in that it exacerbates the zombie data problem. A local planner is not likely to have the time to: a) acquire and develop the skills necessary to use the ArcView/Excel system; b) work with other departments to get the latest zoning and development data; and c) find a way to share these updated data

sets with regional and state agencies. These activities must all depend upon one another if we hope to see them always performed.

The PAMML services framework does this. In order to do custom analysis, a local planner acquires the PAMML model from the state. To run the model, they either buy commercial software that understands PAMML, or they might remotely access an online suite of analysis tools that understood PAMML.[2] Either way, whether the model processed the analysis locally in commercial software, or remotely using a Web service, the core data sets would still be accessed via Web services. The local user *could* physically change the model to point to a local copy of the data, but *it would be easier* to leave the model alone, and update the original data set. This strategy updates the data for everyone (who is using the PAMML framework). Note that this framework is flexible. People can still do things the old way, but it's *easier* to do things right. This concept is a key design feature of PAMML in that the time and effort required to accomplish a task is aligned with the desired outcomes.

Machine-to-machine interaction

With the detailed model we have now developed, one gets a better picture of how extensive the opportunities for exploration are, but also how complex even a simple analysis like buildout can be. We have already digested the model into a diagram instead of presenting the raw PAMML XML code, and it is still complicated—not because the

---

[2] A state agency like MassGIS might develop and provide local planners access to such a system.

XML technology is cumbersome, but because the process of analysis is inherently quite intricate when every step is formally articulated.

We struggle to share data and collaborate on analysis in part because it is difficult to manage all these steps, and this is where the value of a Web services architecture really becomes apparent. As we observed in the buildout analysis, planning problems usually require data inputs from many sources, and the expertise of many different kinds of people. This situation implies that many different types of computing systems are involved in the solution to any problem. The Web services architecture can be thought of as a programming language for distributed, loosely coupled computers. This is in stark contrast to most programming languages, which are designed for developing software programs that will be run on a single computer on a single operating system.

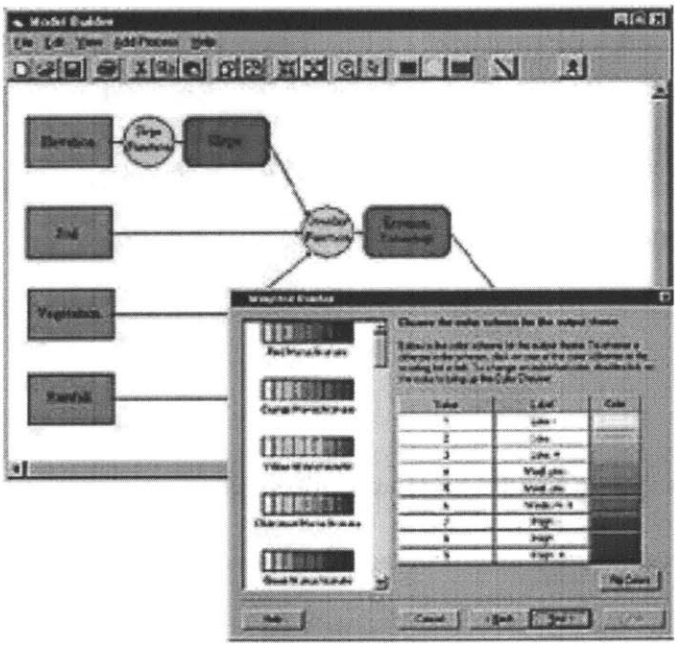
By using a technology framework designed to leverage industry-wide solutions to machine-to-machine interaction problems, we do two important things. First, we use a framework that is well-aligned with the distributed nature of organizational relationships in planning. Second, we are able to focus on planning problems instead of inventing new technology solutions from scratch. With a Web services architecture in place, we know that complex problems can be broken down into more manageable pieces, so that different people (or organizations) may develop the part of the system in which they have expertise. We have been able to do this before Web services, but it has been executed poorly, or at too high a cost. Now we have the tools needed to make machine-to-machine interaction feasible.

Interactive End Products

While the complexity of information processing may be managed through a distributed computing architecture, people must still strive to understand problems as a whole. Clearly interaction with the system must be mediated by applications and user interfaces that focus on a particular task or audience. The ability to build and interact with complex models through a visual interface is a hallmark of modern geographic information systems. Up to now, the case for PAMML has been based mainly on its importance in accurately capturing the *process* of information management and sharing, and through a better articulation of this process, creating the opportunity for automation and componentization. However, if we hope to ever "plan continuously," the PAMML framework must not only save time, money, and effort, but must also drive the rich visual interfaces that professional planners demand. Visual modeling and analysis interfaces are common features of commercial software. In the planning field, ESRI's ArcGIS is the most popular package. It's main interface is a map, into which data sources can be added and styled cartographically. A visual tool called ModelBuilder™ has recently been added to ArcView, allowing users to design an analysis using a wiring diagram metaphor (Figure 6-8). While ModelBuilder™ captures the modeling process in a powerful visual metaphor, it does not go beyond being a front-end to an internal scripting language. It does nothing to expand the analyst's role beyond that of the desktop, or enterprise GIS user by, for example, facilitating collaboration across users of ArcGIS, let alone other software packages. While ModelBuilder is a new product, visual model building software has been an active area of research in PSS as well as computer science in general for years. Proposals for generic enterprise modeling and analysis

toolkits can be found in abundance (Ledeczi, et. al. 1999, Delen and Benjamin 2000), but more relevant are the geospatial applications. GeoVista (Gahegan, et. al. 2002), shown in **Error! Reference source not found.**, is probably the most mature. It provides a graphic interface for spatial data analysis, exploration and visualization. A staff of researchers at the Pennsylvania State University are tasked with the software's continued development and maintenance. Visual Map Algebra (Figure 6-10) is a graphical user interface to Tomlin's ubiquitous map algebra raster analysis framework (Egenhofer 1995).

Figure 6-8: ESRI's ModelBuilder, a visual user interface to Spatial Analyst



While these systems have a multitude of useful analysis features, when viewed through the lens of this work their similarities are more striking than their differences. All of these systems have two primary characteristics, the artisan work model, and the lack of any attention to systems interoperability. The artisan model is one where tasks are accomplished by a few, skilled people in a workshop alone with their tools and materials.

The artisan strives to improve their skills and acquire new tools to produce better products. This has been translated into software as products with words in their name like "workbench" or "toolkit," and the analysis environment and the tools are always seen to be idiosyncratic of the artisan-user. These products often have sophisticated tools allowing a user to build their own model, invent new model types, or save the description of the model for later re-use; but we do not observe an effort to share models by promulgating a standard language, or support a model structure that supports multiple users collaborating. This mindset stands in contrast to the PAMML framework, in which data are not materials, but other tools. And tools must at some level be shared, which requires systems interoperability. So a key concern of this work is to retain the useful analytic features and user interfaces of visual modeling software, while using PAMML to address those aforementioned drawbacks, that hinder progress towards reducing the costs of planning analysis.

Visual modeling

An experiment was performed to see how well PAMML would be able to integrate with the type of visual model construction environments being advocated. This is a test of the framework's ability to appeal to traditional PSS designers and users. The task was viewed as an exercise to represent objects and their semantics in a visual environment. As a graphical user interface environment, a generic network diagramming library called JGraph (*http://www.jgraph.com*) was used. This provided a set of tools for drawing shapes, moving them around the screen, connecting them with lines, and automatically laying out network diagrams. As JGraph was developed in an object-oriented programming language, Java, it was a relatively straightforward exercise to extend

JGraph's standard graphic objects to include a fragment of PAMML XML code. It was then possible to create JGraph PAMML objects that acquired unique characteristics, such as color and shape, based on their PAMML type (Figure 6-11). Note that it was possible to fully replicate the rich graphic conventions used to illustrate the MassGIS buildout model.

A separate issue was the need to move models from their representation as XML text in a file, to visual objects on a computer screen. A number of tools were investigated that could programmatically accomplish this task. It seems that while a number of toolkits exist to automate the creation of visual interfaces to XML data, none of them were fully developed enough to use for this work. This could be because XML is not conducive to this kind of automation, but it is probably only that XML tools are still maturing; it took many years for good visual HTML editing tools to be widely available, and XML is only about five years old.

One popular mainstream application with nascent support for visual editing of XML data is Macromedia Flash MX (`http://www.macromedia.com/software/flash/`). Its roots in the Web design world (as compared to the information modeling community) are quite evident here as Flash has no way of building an interface directly from an XML Schema. It needs a concrete XML instance document for this. That technique can work for small, simple documents where every data field is always populated, such as a purchase order, but in our case, we have an extensive language in which no one model ever uses the entire vocabulary. In other words, Flash can be taught how to build a user interface for `Buffer` models, or `ShapefileWriter`s only, but it offers no tools to simplify

development of a user interface for the entire language. Hopefully the program will mature in this respect in coming years.

Figure 6-9: GeoVista Studio's Design Box, showing connected components

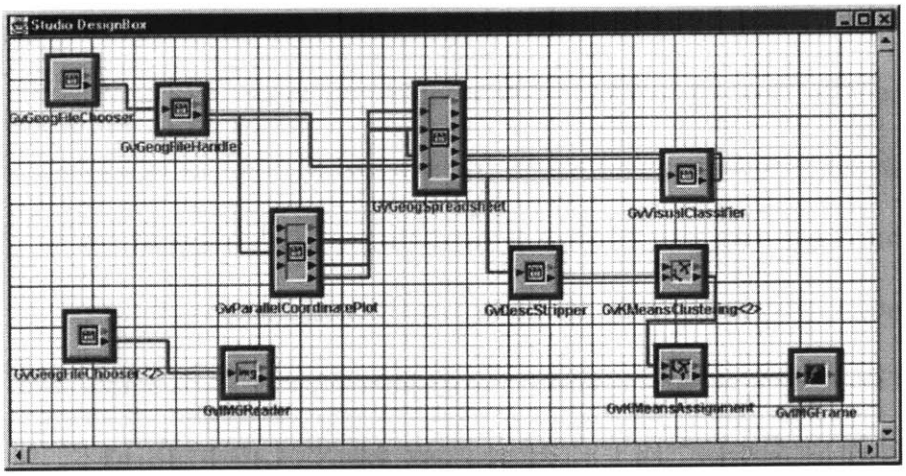
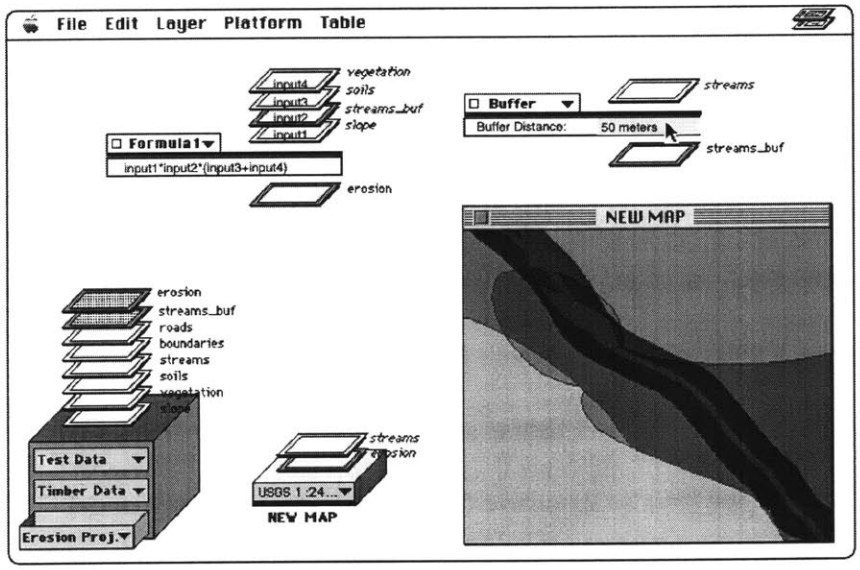


Figure 6-10: Visual Map Algebra as used in the Geographer's Toolkit



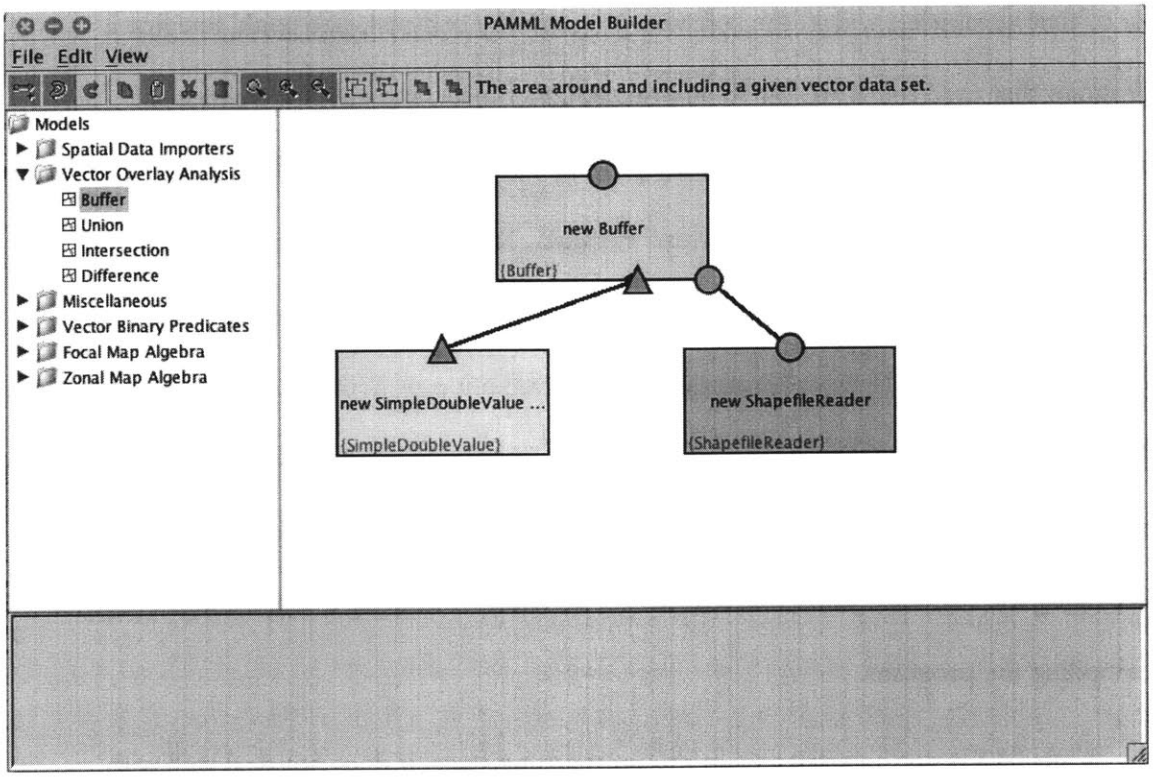
While Flash comes from the community of Web design, Sun Microsystems' Java for XML Binding (JAXB) library comes from programmers. JAXB is part of Sun's Web Services Developer Pack (`http://java.sun.com/webservices/jwsdp/index.jsp`). It is used to generate a Java version of a given XML Schema—in this case PAMML. The motivation for JAXB, and other similar applications, is that there should be a clear separation between the *translation* of an XML vocabulary from text to software, and the *use* of that vocabulary within the software program (by contrast, one could imagine a program that used the XML file as its primary data object, and any information processing that occurred in the software program would be reflected directly in the XML that it produced). JAXB was able to read PAMML XML and instantiate it as Java objects. It was then a simple matter to connect these PAMML Java objects to the JGraph objects. The library also automated the translation of PAMML back out of Java and into XML. This was an indispensable tool during the language development phase, because it allowed applications to be built while the language was still being fine-tuned. Changing the PAMML Java code was simply a matter of running a script and re-compiling the program.

While JAXB was a useful tool in the prototyping stage, it does not deal well with some of the advanced features of XML Schema.[3] Once these became important to PAMML's design, JAXB could no longer be used. While this was disappointing, the benefits of JAXB's automation features are less compelling when dealing with a stable XML Schema. It is just as easy, and more flexible, to write one's own XML processing

[3] Such as the XML Schema Instance <type> element, more commonly known as <xsi:type>, which is XML Schema's mechanism for implementing object-oriented inheritance.

and visualization routines. The failure of JAXB, and the subsequent ability to carry on without it, underlines one way in which the choice of an XML framework was a sound initial decision. Despite the relative youth of XML, no barriers were encountered that suggest that the PAMML Web services framework would impede the research and development of visual interfaces to analytic models.

Figure 6-11: PAMML modeling using JGraph and JAXB



Non-technical user interfaces

The previous discussion has covered rich, visual interface tools geared towards planners and modeling professionals. However, users with less modeling expertise must be engaged in the planning process also. This requires that our XML models take on a

much simpler interface than those just discussed. It is not surprising that PAMML can accommodate this requirement, but the way the requirement is met is extremely interesting.

The end result of the buildout analysis is the series of maps shown in chapter 3, along with a group of statistics like those listed for Sutton, MA in Figure 6-12. We know that in a PAMML framework, these statistics would be the output of models, which were described in XML. If we wanted to reproduce the Web page in Figure 6-12 from a PAMML model (which would allow the page to always display the latest projections), we could use one of a number of industry-standard XML processing tools that generate HTML code from XML. This would be cheap, not only because a host of tools already exist in commercial and open source marketplaces, but also because a host of skilled labor (Web designers) exists that can develop these applications with little additional training. Now we find our strategy of embracing mainstream technology taking us beyond direct reduction of technology costs, and into labor market efficiencies, underscoring once again the importance of integration.

Figure 6-12: Buildout analysis summary for Sutton, MA

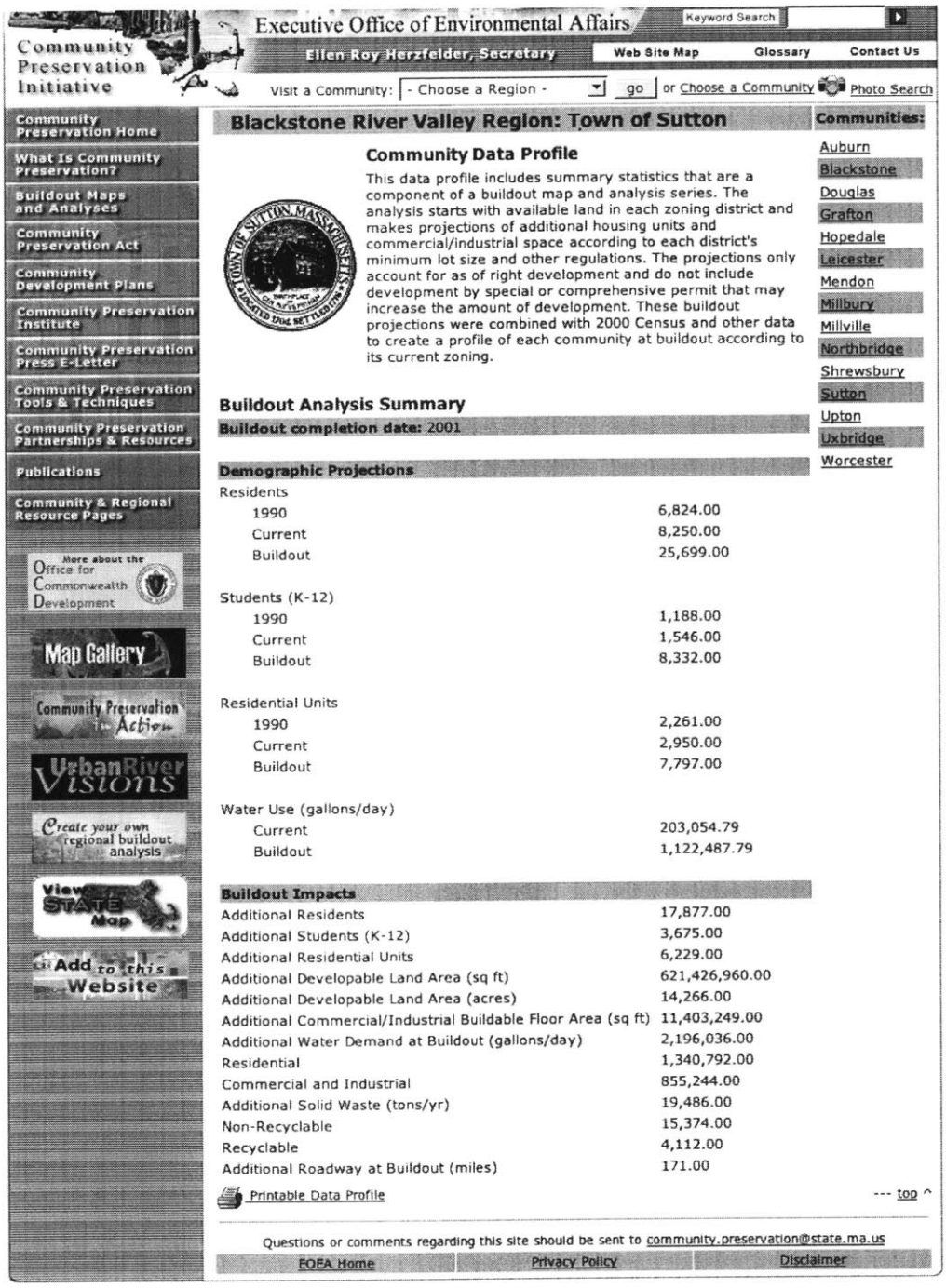
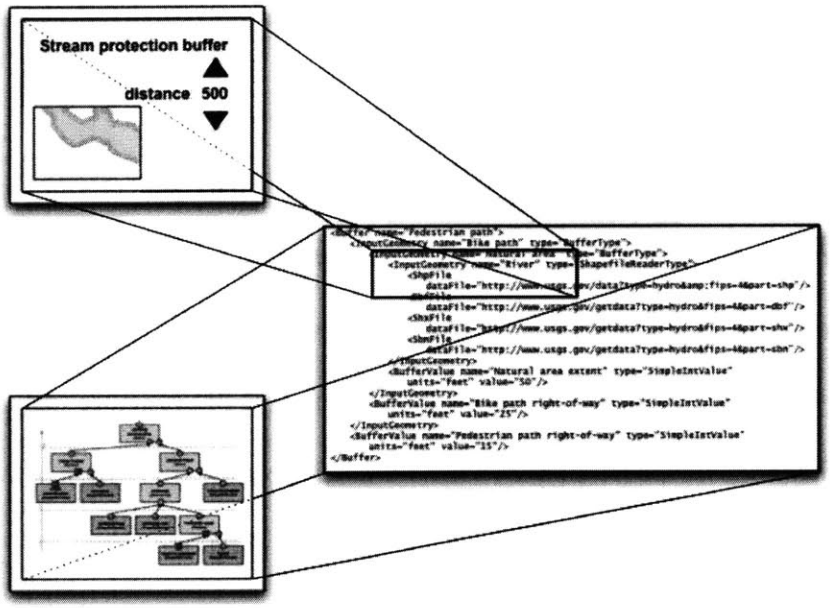


Figure 6-13: Multiple user interfaces using the same code



The rich, visual modeling environment geared towards analysis professionals was prototyped at the front end by transforming PAMML XML into a programming language (in this case Java™), which could then be used within a traditional software development environment to develop any desired application. That user interface is still a tool whose output is a PAMML model, which describes an information processing job. This job must then be executed by an information processing engine, like a GIS system, which might reside on one computer, or be spread out among many. This strategy could be adopted to provide non-technical end users with visual interfaces as well. They key would be to present only limited sections of the model to a user, and design the interface with the user's skill level in mind. Figure 6-13 describes this scenario. An XML model is shown on the right. In the bottom left, the entire model is brought into a visual model building application like that described above. In the top left, however, we see a small part of the model (one buffer operation), presented in a much different fashion. One

variable is shown, and the value is changed by clicking on up and down arrows. In this case the user's software is smart enough to generate a preview of the buffer based on the spatial data set and the distance value chosen by the user.

We could also take an approach similar to that used above for Web page generation. There are a number of mainstream technologies available to automatically generate user interfaces from XML documents, most notably Flash and XForms (*http://www.w3.org/MarkUp/Forms/*). These are currently too simplistic or immature for developing applications for modeling professionals, but hold much promise for lighter weight, simpler applications—especially those designed for Web sites. And once again the same cost efficiencies would be realized for using mainstream technologies and a mainstream skill base. By mixing and matching the right XML-aware technologies with the right audience, we can begin to imagine how even small municipalities, with help from their regional planning agencies, might be able to provide their constituents with continuously updated, dynamic, interactive information. The completion of a housing project could trigger an update of septic loading. Or a new store opening could add congestion to a traffic model. And most importantly, any data visualization carries with it the underlying PAMML model description, so that one can imagine "copying" statistics off a Web page and pasting them into a (PAMML-aware) spreadsheet, which would not actually copy the text on the page, but the underlying XML PAMML code, so that the data does not revert to "zombie" status, and the contract between data user and provider is retained.

This chapter has presented a range of prototyping efforts, from systems design to actual software development. In each case the information management problems that are so pervasive in our profession, and are observed in the buildout analysis, were addressed. In many different ways we have seen that systems built upon the PAMML framework are likely to be cheap, scale well with organizational needs, and integrate well with mainstream technology trends. This evidence goes far towards proving that planning support systems built in this manner have a chance to avoid the systemic information management problems we observe today.

Chapter 7. Reflections, Critiques, and Future Directions

This thesis began with the concern that planning analysis is too expensive due to systemic problems with the information technologies in use today. No matter how small or straightforward the analysis is, it seems that much of the project time is spent gathering data and preparing it for use, and once a project is complete, its results can rarely be updated without incurring costs approaching those of the original analysis effort. And yet the sheer amount of data about the urban environment increases yearly as we install traffic counters on our roads, and air quality sensors on our rooftops. Making use of these data will not only require better information management techniques, but also better ways for experts to engage in collaborative analysis. However, the current state of technology makes it difficult to imagine that any but the largest projects and/or agencies will have the resources to marshal the expertise of various planning disciplines and their detailed data sources to analyze urban problems.

A look at the literature on geographic information sharing offers little help. That field tends to hold technology as fixed, which leads to solutions where organizational behavior issues are the focus. We do, however, make two important findings. One is that the most universal determinant of geographic information sharing success is the cost-benefit ratio of implementation. In most situations, the cost of sharing and collaboration is high, which creates a need for large benefits to all participating organizations. The second is that organizations often function as independent "trading partners" rather than one entity with different departments, or agencies. In this institutional context we

attempt to implement one of two technologies; either we use some type of "enterprise" system, which centralizes information management too much for the comfort of most business partners, or a Web site based system, which is good at disseminating read-only data, but not at fostering collaboration.

What would happen if we instead adapt technology to fit the organizational behavior we observe in practice? This is the question we set out to answer in this thesis. It led to a research agenda based on Web services, a relatively new paradigm for developing multi-participant computing systems where the parties involved are "loosely coupled," meaning that their collaborative interactions do not require changes to the systems designed to achieve their core internal goals. The Web services paradigm was chosen because it is flexible enough to allow a discipline like planning support systems to build its own specialized tools, but defined enough so that the basic enabling software works for multiple industries, and is therefore a relatively cheap commodity.

The flexibility of the Web services architecture requires that our analytic tools must be re-developed within this new framework, and that was the focus of the second half of this work. Using the MassGIS buildout analysis as a lens through which we could look at the key information management challenges in PSS, we built up a Web services-based PSS framework called PAMML. PAMML consists of a vocabulary for describing the components, or building blocks, or information exchange and processing, along with a suite of Web services that can execute the requests articulated in the vocabulary. We showed how data sharing, stakeholder participation, collaboration, and iteration could be approached from this framework. We showed that the costs could be extremely low for

small organizations with simple needs, but they could still interact with larger organizations with much more sophisticated systems.

Finally, we used the tools we created to re-develop the buildout analysis using PAMML and discussed some potential user interfaces that might drive a PAMML-based system. While this began to show what planning support systems might "look like" from a user's perspective, this thesis has been primarily about the development of the framework. This is unusual in that it is more common for technology research in urban planning to be about an implementation (software design and development) of an existing technology framework. This means that the technology presented here has been at a more abstract level, which creates a tension as the reader may have been expecting to see a concrete software application as the final product of this work. Fully developing a software implementation in this way would put the emphasis, and therefore the evaluation, of PAMML in the wrong place. This thesis concludes with some discussion of the difficulty in evaluating a work of this kind, some of the accomplishments made despite this difficulty, and some as yet unmentioned areas of urban planning that PAMML has a chance to significantly impact.

Critiquing the PAMML vocabulary

The vocabulary developed in this thesis has concentrated on spatial analysis and distributed processing. It is far from being a complete planning, or analysis, language. There are two main reasons for this. First, it would be impossible to cover the entire field of planning, or analysis in one research effort. That is rightly the work of a research community. Second, XML languages are designed to integrate multiple vocabularies,

allowing people to specialize in certain areas. For example, many XML languages will require a way to describe someone's address. XML was designed so that everyone can reference and use the postal service's address definition, instead of inventing their own.

This is the strategy envisioned here for implementing a number of features that were not discussed. For example, the ability to construct database queries was not a feature of the PAMML framework describe here. This is not because database query is not a critical feature in PSS; in fact, the next level of sophistication in the treatment of data sharing would have been to discuss the need for data users to get incremental updates of large data sets, and synchronize their personal changes with the official repositories. These features were not discussed because they are so central to IT in general that the solution must come from the database query specialists, not the planning community. There is a great deal of work underway in this area,[1] but at this time, no standard query vocabulary has been widely adopted. This argument applies to many areas that would seem central to the development of a planning analysis vocabulary, such as metadata, where the Federal Geographic Data Committee (FGDC) leads efforts to standardize the way in which data developers describe a data set's provenance.

So, while this work has shown that PAMML addresses a wide range of physical planning issues, the language is not complete. But it can not be judged harshly on the basis of what features are missing, because this lack of completeness is by design. One could question how well PAMML is able to complete itself through integration with other vocabularies, but this is largely taken care of by the design of XML itself. XML

---

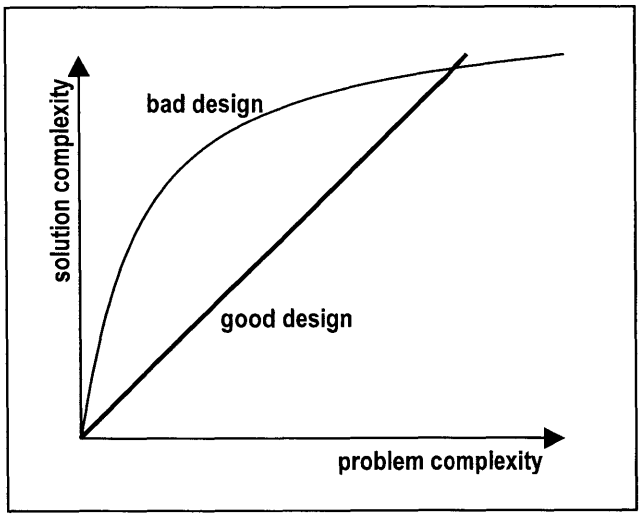
[1] Most notably XSQL (`http://xsql.sourceforge.net`) and XQuery (`http://www.w3.org/XML/Query`), a specification from the World Wide Web Consortium.

guarantees that vocabulary integration is possible, but it does not speak to the efficacy of this integration. But although that criterion is important, it is so subjective that it is difficult to discuss here.

Another way to think about evaluating the language is to look at the vocabulary from a semantic perspective. Is it too wordy? Can a term have multiple, confusing meanings? Are the meanings of terms easily recognizable to the community they serve? These issues have been addressed through careful adoption of well-vetted concepts within the spatial planning community. In terms of data modeling, we believe vector and raster and tabular data (along with some numeric types), are the key basic types to consider. For spatial operations, we believe that map algebra and set theory operations (union, intersect, etc.) provide the foundation upon which most spatial analysis is based. If one disagrees with this assessment, then PAMML will seem poorly conceived. However, these data types and operations seem to be well-accepted within our community, and therefore are not confusing, or verbose.

One risk, however, is that the language has been specified at too coarse a level of detail. For example, there are no requirements regarding the type of geometry within a vector data set. We said that this should be one of the seven types described in the OpenGIS Simple Features for SQL specification, but must it be one type, like polygon, or can a vector object in PAMML consist of a mix of different types, like polygons and lines. PAMML simply provides the syntax with which a user can specify either choice, which seems to be the strategy employed by most specification efforts, so we neither resolve, nor exacerbate any existing debates on this issue.

In our research and professional work on specifications and language definitions, the most important factor of success by far has been community adoption. If a language is able to garner widespread use, it serves its purpose. Because XML ensures that a language designer can not break the most important rules of information modeling, the success of an XML-based dialect has little to do with vocabulary syntax. The most



important factor seems to be the ability to solve the most basic problems simply and tersely. And another important factor is to make sure solutions scale smoothly in relation to the problem solved. In other words, if problem complexity and solution complexity were the axes of a graph, a good system design would map problems and solutions along a straight line.

If a large group of users can quickly solve their most basic problems, then they are likely to adopt the technology. This creates a large user base, and that is the key to any successful Web services framework. Unlike traditional desktop software, service-oriented frameworks have little value until a community of users exist that take advantage of the system. These users will fix any problems with syntax and meaning, and develop

innovative new grammars on top of the original language that address new challenges that will arise, but the community must be developed in the first place, which leads us to our final critique—that these services are best tested in a multi-participant environment.

Exploring the nature of Web services as contracts

This research has primarily been about defining a technology framework rooted in Web services, however few concrete Web services were described. There are many reasons for this omission. First of all, the syntax of Web Services Description Language (WSDL) conveys little information to a human being. Second, every operation or object in the PAMML language could be expressed directly as a service, so from that perspective many services are described here. More important than these two considerations is the fact that the Web services are less important than the paradigm shift of moving to a service-oriented architecture (SOA) from a data-oriented one. This conceptual shift transfers the emphasis of information-intensive planning from data management to process management. In the buildout case, for example, once the planners set up data sharing agreements with local municipalities, a service-oriented framework like PAMML makes it as easy for the analysis software to always use the most up-to-date local information as it is to use zombie data off a hard drive.

This works in theory, but the theory has not been tested in practice. Concrete implementations of the services we describe in the abstract are important because that is where the "contract" between business partners is defined. In the first chapter we postulated that collaborative initiatives that use technologies that in some way capture the social or legal contractual relationship between parties may succeed more often. We

expect to see this result due to the ability to better align expectations, in the form of "real" contracts, with performance, the information sharing system, thereby reducing the chance that either party will feel that they were not receiving the agreed-upon benefits. This postulate has not been tested here. It requires a level of empirical research beyond the scope of this work. PAMML reduces costs in many other ways, so this research stands on its own, but work on contracts has the potential to increase the strength of our argument.

When we speak of a service-oriented architecture, we are talking not only of the interaction between two parties, but of a whole network of content producers and consumers. A town assessing office may be a consumer when they acquire property information from private developers, but they are a producer when they then share their property database with a state agency. And if the state then sells this information (in an aggregated form, of course) to private firms, what is the town's relationship to the final end user? Do they have a contract with each other? And as a practical matter, what happens when someone cannot keep their server running? These are critical implementation questions whose answers will determine whether or not Web services, and service-oriented architectures in general, will work. Although as planners we must be concerned about this issue, there is little we can do. It is a concern for the entire information technology community. Many standards communities are working to define these multi-participant service relationships[2], yet no clear strategy has emerged. We can not even be sure that the effort in this area is warranted. WSDL already gives us the

[2] With names like Web services choreography language, Web services orchestration language, Web services flow language, Business process execution language, and Web services modeling language.

tools to define the relationships between a service consumer and producer. Why must multi-service systems require a different structure? This has yet to be proven, and may or may not become an important concern for planners.

Further implications for the planning profession

This thesis has concentrated on the value of the PAMML framework to improve the way in which planners engage in information management and analysis, two areas very central to the discipline of information technology. While some attention has been paid to the requirements of participatory decision making, for the most part the discussion has been restricted to those areas of planning practice traditionally associated with IT. However, if the PAMML framework were to become the backbone of planning support systems, this would create an opportunity for other practices to redefine the way they engage in spatial analysis. Two of these are explored here.

Democratizing urban design

One significant implication of this work is the potential to create, articulate and implement urban design patterns using PAMML. As Alexander says, "[a design pattern] describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" (Alexander, 1977). In the computer science domain, like architecture, design patterns are generic problem-solving models. They are the building blocks of analysis, like mathematical theorems.

Urban planning 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 "image-able city" (Lynch), Christopher Alexander's "pattern language" (Alexander 1977), or Alan Jacobs' "great streets" (Jacobs).

Urban design, as it is currently conceived, is inherently an expensive endeavor. Good urban design usually requires the attention of a team of professional planners over the course of weeks or months. People's time does not come cheaply, and therefore only the wealthiest communities, or the most important projects, receive the long-term attention of professional urban designers. The rest must make due with "commodity" design tools, resulting in communities whose aesthetic is driven mainly by the interactions of zoning regulations, road construction manuals, and the profit-maximizing tendencies of private developers.

We believe that an urban design language can be articulated using the basic spatial syntax found in PAMML. And if this is the case, then any design theory expressed in PAMML has the potential to be implemented computationally, instead of by professional designers, thereby greatly reducing the cost of design. One recent example of this principle at work is Bill Hillier's Space Syntax theory. This design theory has been described computationally and packaged into many different software programs (`http://www.spacesyntax.org/software/index.htm`), including ArcView

(*http://www.casa.ucl.ac.uk/venue/space_syntax.html*), the most common GIS package in the world. It does not seem a coincidence that in little over a decade the technique has been used in localities all over the world, guided by people other than the original designer. Of course we do not expect the design process or the end product to be as rich as if it were performed by human designers, but the reach of one's ideas is exponentially broader.

Enabling community statistical systems

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. "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 rule books. 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 on 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.

Information technology developments that change the nature of planning tools also affect the planning process itself (Schuur, 1994). It is difficult to imagine that current planning support systems can facilitate the development of community statistical systems in a scalable, cost-effective way. Hopefully PAMML Web services can change the nature of planning tools and help elevate the discourse of urban planners among the voices competing to shape our society.

In this work we have used a standard framework for building multi-organization, distributed computing systems to redefine the information architecture upon which planning support systems are built. As an initial proof of its efficacy, we designed and prototyped a system that solved many of the issues found in spatial analysis and physical planning. This has been an important exercise, as physical planning is one of the core areas of PSS, but the real importance of PAMML will be seen in its ability to integrate the work of previously separate fields. For example, the computational expression of urban design, as discussed above, might be built using intermediate spatial analysis tools built by geographers, transportation planners, and environmental experts, but our current technology paradigms make it hard to imagine this type, or depth, of collaboration at any cost. Hopefully, the PAMML framework allows us to begin to envision a new era in planning support systems with fewer limits to our ability to collaboratively and continuously plan for the future.

Appendix A. Planning Analysis and Modeling Markup Language XML Schema

```
<xs:schema targetNamespace="http://web.mit.edu/rajsingh/www/xml/ns/pamml"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:pamml="http://web.mit.edu/rajsingh/www/xml/ns/pamml">
  <!--
  ********************
  Basic Model Types
  ********************
  -->
  <xs:element name="Model" type="pamml:ModelType"/>
  <xs:complexType name="ModelType">
    <xs:annotation>
      <xs:documentation>Basic information for a model</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="pamml:Metadata" minOccurs="0"/>
      <xs:element ref="pamml:Permissions" minOccurs="0"/>
      <xs:element ref="pamml:RemoteInfo" minOccurs="0"/>
      <xs:element ref="pamml:Alternatives" minOccurs="0"/>
    </xs:sequence>
    <xs:attributeGroup ref="pamml:globalAttributes"/>
  </xs:complexType>
  <xs:element name="Models">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="pamml:Model" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--
  ********************
  Generic
  ********************
  -->
  <xs:element name="GenericModel" type="pamml:GenericModelType"/>
  <xs:complexType name="GenericModelType">
    <xs:annotation>
      <xs:documentation>an opaque, "black box" model that permits
modification of specified properties</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ModelType">
        <xs:sequence>
          <xs:element name="InputProperty" type="pamml:ModelType"
maxOccurs="unbounded"/>
          <xs:element name="OutputProperty" type="pamml:ModelType"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  BooleanData
  ********************
  -->
  <xs:element name="BooleanData" type="pamml:BooleanDataType"
abstract="true"/>
  <xs:complexType name="BooleanDataType">
    <xs:annotation>
      <xs:documentation>a true or false value</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ModelType"/>
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  ValueData
  ********************
  -->
  <xs:element name="ValueData" type="pamml:ValueDataType"/>
  <xs:complexType name="ValueDataType">
    <xs:annotation>
      <xs:documentation>a single cardinal numeric value</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ModelType">
        <xs:attribute name="units" type="pamml:ValueUnits" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  TableData
  ********************
  -->
  <xs:element name="TableData" type="pamml:TableDataType"/>
  <xs:complexType name="TableDataType">
    <xs:annotation>
      <xs:documentation>A two-dimensional matrix of data, like a spreadsheet
or relational table</xs:documentation>
      <xs:documentation>number of Attribute elements must match the data
set</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ModelType">
        <xs:sequence>
          <xs:element ref="pamml:AttributeInfo" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="key" type="xs:string" use="optional"/>
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  Basic Geographic Models
  ********************
  -->
  <xs:complexType name="GeoDataType">
    <xs:annotation>
      <xs:documentation>A model whose output is geographic
data</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ModelType">
        <xs:attribute name="srsName" type="xs:anyURI" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  Vector Data Model
  ********************
  -->
  <xs:element name="VectorData" type="pamml:VectorDataType" abstract="true"/>
  <xs:complexType name="VectorDataType">
    <xs:annotation>
      <xs:documentation>A model whose output is geographic vector
data</xs:documentation>
      <xs:documentation>and whose attributes are in tabular
format</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:GeoDataType">
        <xs:sequence>
          <xs:element ref="pamml:AttributeInfo" minOccurs="0">
            <xs:annotation>
              <xs:documentation>attribute information the model author
chooses to expose</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="geometryType" type="pamml:GeometryType"
use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  Raster Data Model
  ********************
  -->
  <xs:element name="RasterData" type="pamml:RasterDataType"/>
  <xs:complexType name="RasterDataType">
```

```
    <xs:annotation>
      <xs:documentation>A model whose output is geographic raster data and
has only one non-geographic attribute</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:GeoDataType">
        <xs:attributeGroup ref="pamml:rasterAttributes"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  Inline (written directly in PAMML) data encodings
  ********************
  -->
  <xs:element name="SimpleBooleanValue">
    <xs:annotation>
      <xs:documentation>A single true or false value</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:BooleanDataType">
          <xs:sequence>
            <xs:element name="Value" type="xs:boolean"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="SimpleIntValue" type="pamml:SimpleIntValueType"/>
  <xs:complexType name="SimpleIntValueType">
    <xs:annotation>
      <xs:documentation>a single cardinal integer value</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ValueDataType">
        <xs:sequence>
          <xs:element name="Value" type="xs:int"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="SimpleDoubleValue" type="pamml:SimpleDoubleValueType"/>
  <xs:complexType name="SimpleDoubleValueType">
    <xs:annotation>
      <xs:documentation>a single cardinal decimal value</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ValueDataType">
        <xs:sequence>
          <xs:element name="Value" type="xs:double"/>
        </xs:sequence>
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="SimpleXMLTable">
    <xs:annotation>
      <xs:documentation>An inline table</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:TableDataType">
          <xs:sequence>
            <xs:element name="table">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="tr" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:documentation>a data record, e.g. a
row</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="att" type="xs:anySimpleType"
maxOccurs="unbounded">
                          <xs:annotation>
                                <xs:documentation>a record data
item</xs:documentation>
                          </xs:annotation>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="numRecs" type="xs:int"
use="optional"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!--
  ********************
  Spatial Operations
  ********************
  -->
  <!--
  ********************
  Base Types for operations involving one spatial dataset
  ********************
  -->
  <!-- Base Vector Type -->
  <xs:complexType name="VectorUnaryOperationType">
    <xs:annotation>
```

```
      <xs:documentation>Base Type for Spatial Vector Operations involving
one vector dataset</xs:documentation>
      <xs:documentation>All attributes should be maintained in the new data
set</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:VectorDataType">
        <xs:sequence>
          <xs:element name="InputGeometry" type="pamml:VectorDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- Base Raster Type -->
  <xs:complexType name="RasterUnaryOperationType">
    <xs:annotation>
      <xs:documentation>Base Type for Spatial RasterOperations involving one
raster dataset</xs:documentation>
      <xs:documentation>An application may maintain all attributes in the
new data set</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:RasterDataType">
        <xs:sequence>
          <xs:element name="InputRaster" type="pamml:RasterDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  Base Types for operations involving two spatial datasets
  ********************
  -->
  <!-- Base Vector Type -->
  <xs:complexType name="VectorBinaryOperationType">
    <xs:annotation>
      <xs:documentation>Base Type for Spatial Vector Operations involving
two vector datasets</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:VectorDataType">
        <xs:sequence>
          <xs:element name="BaseGeometry" type="pamml:VectorDataType"/>
          <xs:element name="OpGeometry" type="pamml:VectorDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- Base Raster Type -->
  <xs:complexType name="RasterBinaryOperationType">
    <xs:annotation>
```

```
      <xs:documentation>Base Type for Spatial RasterOperations involving two
raster datasets</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:RasterDataType">
        <xs:sequence>
          <xs:element name="InputRasterA" type="pamml:RasterDataType"/>
          <xs:element name="InputRasterB" type="pamml:RasterDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--
  ********************
  Map Algebra
  ********************
  -->
  <!-- Raster Algebra Focal Model -->
  <xs:element name="RasterFocal">
    <xs:annotation>
      <xs:documentation>Basic map algebra.</xs:documentation>
      <xs:documentation>Cell values are calculated based on a constant or
another raster grid</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:RasterUnaryOperationType">
          <xs:choice>
            <xs:element name="OperationRaster"
type="pamml:RasterDataType"/>
            <xs:element name="OperationValue" type="pamml:ValueDataType"/>
          </xs:choice>
          <xs:attribute name="operation" type="pamml:FocalOperation"
use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- Raster Algebra Zonal Model -->
  <xs:element name="RasterZonal">
    <xs:annotation>
      <xs:documentation>Cell values are calculated based on operations on
neighboring cell values. neighborhood size is a constant or a value from
another raster grid</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:RasterUnaryOperationType">
          <xs:sequence>
            <xs:choice>
              <xs:element name="NeighborhoodSizeRaster"
type="pamml:RasterDataType"/>
```

```
                <xs:element name="NeighborhoodSizeValue"
type="pamml:ValueDataType"/>
              </xs:choice>
            </xs:sequence>
            <xs:attribute name="operation" type="pamml:ZonalOperation"
use="required"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <!-- ************************************************************      -->
    <!--   Constructive Spatial Operations                              -->
    <!-- ************************************************************      -->
    <!--
    ********************
    Buffer
    ********************
    -->
    <!-- the vector case -->
    <xs:element name="Buffer" type="pamml:BufferType"/>
    <xs:complexType name="BufferType">
      <xs:annotation>
        <xs:documentation>Generates a buffer</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="pamml:VectorDataType">
          <xs:sequence>
            <xs:element name="InputGeometry" type="pamml:VectorDataType"/>
            <xs:element name="BufferDistance" type="pamml:ValueDataType"/>
            <!-- add choice to use a lookup table to vary the distance based
upon a feature value -->
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- the raster case -->
    <xs:complexType name="RasterBuffer">
      <xs:annotation>
        <xs:documentation>Generates a buffer</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="pamml:RasterUnaryOperationType">
          <xs:sequence>
            <xs:element name="BufferValue" type="pamml:ValueDataType"/>
            <!-- add choice to use a lookup table to vary the distance based
upon a feature value -->
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!--
    ********************
    Dissolve
```

```
  ********************
  -->
  <!-- the vector case -->
  <xs:element name="Dissolve">
    <xs:annotation>
      <xs:documentation>Generates new geometry by merging adjacent features
where the useFeatureType attribute is the same</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorDataType">
          <xs:sequence>
            <xs:element name="FeatureName" type="xs:string"/>
            <xs:element name="InputGeometry" type="pamml:VectorDataType"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- a raster case of dissolve does not make sense-->
  <!-- Relate -->
  <xs:element name="Relate">
    <xs:annotation>
      <xs:documentation>Adds features to GeoData</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorDataType">
          <xs:sequence>
            <xs:element name="FeatureName" type="xs:string"/>
            <xs:element name="InputGeometry" type="pamml:VectorDataType"/>
            <xs:element name="FeatureTable" type="pamml:TableDataType"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!--
  ********************
  Set-Theoretic Spatial Overlay Operations
  ********************
  -->
  <xs:element name="Union">
    <xs:annotation>
      <xs:documentation>Returns all areas from the two
geometries</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorBinaryOperationType"/>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
```

```
  <!-- Intersection -->
  <xs:element name="Intersection">
    <xs:annotation>
      <xs:documentation>Returns all areas from 1st Vector that fall within
2nd</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorBinaryOperationType"/>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- Difference -->
  <xs:element name="Difference">
    <xs:annotation>
      <xs:documentation>Returns all areas from 1st Vector that do not fall
within 2nd</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorBinaryOperationType"/>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- Symmetric Difference-->
  <xs:element name="SymDifference">
    <xs:annotation>
      <xs:documentation>Areas of 1st and 2nd Vectors that do not intersect
each other. Opposite of Intersection.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorBinaryOperationType"/>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!--
  ********************
  Miscellaneous Spatial Operations
  ********************
  -->
  <!--
  ********************
  Allocate
  ********************
  -->
  <!-- the vector case -->
  <xs:element name="Allocate">
    <xs:annotation>
      <xs:documentation>Add an attribute from 2nd Vector to 1st and
calculate its value based on the percentage of overlap.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
```

```
      <xs:complexContent>
        <xs:extension base="pamml:VectorBinaryOperationType">
          <xs:attribute name="useFeatureType" type="xs:string"
use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- the raster case -->
  <xs:element name="RasterAllocate">
    <xs:annotation>
      <xs:documentation>Add an attribute from 2nd Raster to 1st and
calculate its value based on the percentage of overlap.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:RasterBinaryOperationType">
          <xs:attribute name="useFeatureType" type="xs:string"
use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- Convex Hull?? -->
  <!-- Basic arithmetic ops -->
  <xs:element name="Query">
    <xs:annotation>
      <xs:documentation>Generates new attributes on a spatial
model</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorUnaryOperationType">
          <xs:sequence>
            <xs:element name="NewAttributes"
type="pamml:AttributeInfoType"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- Quantile -->
  <xs:element name="Quantile">
    <xs:annotation>
      <xs:documentation>Generates aggregate geometry by grouping the values
of an attribute into ranges with equal numbers of members</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorUnaryOperationType">
          <xs:attribute name="useFeatureType" type="xs:string"
use="required"/>
          <xs:attribute name="numRanges" type="xs:int" use="required"/>
```

```
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- Reclass -->
  <xs:element name="Reclass">
    <xs:annotation>
      <xs:documentation>Changes attribute value based on a lookup
table.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorUnaryOperationType">
          <xs:sequence>
            <xs:element name="TableData" type="pamml:TableDataType"/>
          </xs:sequence>
          <xs:attribute name="reclassFeatureType" type="xs:string"/>
          <xs:attribute name="minValFeatureType" type="xs:string"
use="optional"/>
          <xs:attribute name="maxValFeatureType" type="xs:string"
use="optional"/>
          <xs:attribute name="newValFeatureType" type="xs:string"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="RasterReclass">
    <xs:annotation>
      <xs:documentation>Changes attribute value based on a lookup
table.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:RasterUnaryOperationType">
          <xs:sequence>
            <xs:element name="TableData" type="pamml:TableDataType"/>
          </xs:sequence>
          <xs:attribute name="reclassFeatureType" type="xs:string"/>
          <xs:attribute name="minValFeatureType" type="xs:string"
use="optional"/>
          <xs:attribute name="maxValFeatureType" type="xs:string"
use="optional"/>
          <xs:attribute name="newValFeatureType" type="xs:string"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- ********************************************************     -->
  <!--   Spatial Binary Predicate Operations                 -->
  <!--   these all return true or false                           -->
  <!-- ********************************************************     -->
  <!-- Base Vector Type -->
  <xs:complexType name="VectorBooleanBinaryOperation">
```

```
    <xs:annotation>
      <xs:documentation>Base Type for Spatial Vector operations that compare
two vector datasets</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:BooleanDataType">
        <xs:sequence>
          <xs:element name="InputGeometry" type="pamml:VectorDataType"
minOccurs="2" maxOccurs="2"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- Equals -->
  <xs:element name="Equals" type="pamml:VectorBooleanBinaryOperation">
    <xs:annotation>
      <xs:documentation>Interiors intersect and no part of the interior or
boundary of one intersects the exterior of the other</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!-- Intersects -->
  <xs:element name="Intersects" type="pamml:VectorBooleanBinaryOperation">
    <xs:annotation>
      <xs:documentation>The data sets share at least one point in common--
opposite of disjoint</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!-- Disjoint -->
  <xs:element name="Disjoint" type="pamml:VectorBooleanBinaryOperation">
    <xs:annotation>
      <xs:documentation>The data sets share no points in
common</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!-- Touches -->
  <xs:element name="Touches" type="pamml:VectorBooleanBinaryOperation"/>
  <!-- Crosses -->
  <xs:element name="Crosses" type="pamml:VectorBooleanBinaryOperation"/>
  <!-- Within -->
  <xs:element name="Within" type="pamml:VectorBooleanBinaryOperation"/>
  <!-- Contains -->
  <xs:element name="Contains" type="pamml:VectorBooleanBinaryOperation"/>
  <!-- Overlaps -->
  <xs:element name="Overlaps" type="pamml:VectorBooleanBinaryOperation"/>
  <!--
  ********************
  Non-Geographic Data Access Models
  ********************
  -->
  <!-- Simple ASCII Table -->
  <xs:element name="SimpleASCIITable">
    <xs:annotation>
```

```
      <xs:documentation>An ASCII text file where: the first line is a tab-
separated list of attribute names, the second line is a tab-separated list of
data types, and the remaining lines are tab-separated lists of data
(records)</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:TableDataType">
          <xs:attribute name="dataFile" type="xs:anyURI"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- Value Table -->
  <xs:element name="ValueTable">
    <xs:annotation>
      <xs:documentation>A table that uses ValueModels for
data</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:TableDataType">
          <xs:sequence>
            <xs:element name="table">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="tr" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:documentation>a data record, e.g. a
row</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="Value"
type="pamml:ValueDataType" maxOccurs="unbounded">
                          <xs:annotation>
                            <xs:documentation>a record data
item</xs:documentation>
                          </xs:annotation>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="numRecs" type="xs:int"
use="optional"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
```

```
  <!-- Relational Database as a Table -->
  <xs:element name="GenericRDBMSTable">
    <xs:annotation>
      <xs:documentation/>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:TableDataType">
          <xs:sequence>
            <xs:element name="User" type="xs:string"/>
            <xs:element name="Passphrase" type="pamml:PassphraseType"/>
            <xs:element name="Host" type="xs:anyURI"/>
            <xs:element name="Port" type="xs:int"/>
            <xs:element name="Driver" type="xs:string"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!--
  ********************
  Geographic Data Access Models
  ********************
  -->
  <!--  -->
  <!-- ASCII Grid Models -->
  <!--  -->
  <xs:element name="ASCIIIntegerGridReader"
type="pamml:ASCIIIntegerGridReaderType"/>
  <xs:complexType name="ASCIIIntegerGridReaderType">
    <xs:annotation>
      <xs:documentation>A raster data model whose source is an ESRI ASCII
Grid export file with integer data</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:RasterDataType">
        <xs:sequence>
          <xs:element name="DataFile" type="pamml:DataFileCompressable"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="ASCIIDoubleGridReader"
type="pamml:ASCIIDoubleGridReaderType"/>
  <xs:complexType name="ASCIIDoubleGridReaderType">
    <xs:annotation>
      <xs:documentation>A raster data model whose source is an ESRI ASCII
Grid file with decimal data</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:RasterDataType">
        <xs:sequence>
          <xs:element name="DataFile" type="pamml:DataFileCompressable"/>
```

```
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--  -->
  <!-- Shapefile Model -->
  <!--  -->
  <xs:element name="ShapefileReader" type="pamml:ShapefileReaderType"/>
  <xs:complexType name="ShapefileReaderType">
    <xs:annotation>
      <xs:documentation>A vector data model whose source is an ESRI
Shapefile</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:VectorDataType">
        <xs:sequence>
          <xs:element name="ShpFile" type="pamml:DataFileCompressable"/>
          <xs:element name="DbfFile" type="pamml:DataFileCompressable"/>
          <xs:element name="ShxFile" type="pamml:DataFileCompressable"/>
          <xs:element name="SbnFile" type="pamml:DataFileCompressable"
minOccurs="0"/>
          <xs:element name="SbxFile" type="pamml:DataFileCompressable"
minOccurs="0"/>
          <xs:element name="PrjFile" type="pamml:DataFileCompressable"
minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="ShapefileWriter" type="pamml:ShapefileWriterType"/>
  <xs:complexType name="ShapefileWriterType">
    <xs:annotation>
      <xs:documentation>A vector data model whose output is an ESRI
Shapefile</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:ShapefileReaderType">
        <xs:sequence>
          <xs:element name="VectorModel" type="pamml:VectorDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!--  -->
  <!-- Inline Well-Known Text Model -->
  <!--  -->
  <xs:element name="InlineWKTReader" type="pamml:InlineWKTReaderType"/>
  <xs:complexType name="InlineWKTReaderType">
    <xs:annotation>
      <xs:documentation>A vector data model whose source is
WellKnownText</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
```

```
      <xs:extension base="pamml:VectorDataType">
        <xs:sequence>
          <xs:element name="WKTGeometry" type="xs:string"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="InlineWKTWriter" type="pamml:InlineWKTWriterType"/>
  <xs:complexType name="InlineWKTWriterType">
    <xs:annotation>
      <xs:documentation>A vector data model whose output is
WellKnownText</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:InlineWKTReaderType">
        <xs:sequence>
          <xs:element name="VectorModel" type="pamml:VectorDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- GML 2.1 file Model -->
  <xs:complexType name="SimpleGML2.1Reader">
    <xs:annotation>
      <xs:documentation>A vector data model whose source iconforms to OGC
GML v2.1 </xs:documentation>
      <xs:documentation>and having the same FeatureTypes for every
Feature.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:VectorDataType">
        <xs:sequence>
          <xs:element name="XMLFile" type="pamml:DataFileCompressable"/>
          <xs:element name="XMLSchemaFile"
type="pamml:DataFileCompressable" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- Relational database spatial data Model -->
  <xs:complexType name="RDBVectorDataType">
    <xs:annotation>
      <xs:documentation>A vector data model whose source is a relational
database</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="pamml:VectorDataType">
        <xs:sequence>
          <xs:element name="User" type="xs:string"/>
          <xs:element name="Passphrase" type="pamml:PassphraseType"/>
          <xs:element name="Host" type="xs:anyURI"/>
          <xs:element name="Port" type="xs:int"/>
```

```
            <xs:element name="Driver" type="xs:string"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!-- PostGIS spatial data Model -->
    <xs:element name="PostGISReader" type="pamml:PostGISReaderType"/>
    <xs:complexType name="PostGISReaderType">
      <xs:annotation>
        <xs:documentation>A vector data model whose source is a PostgreSQL
PostGIS database</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="pamml:RDBVectorDataType"/>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="PostGISWriter" type="pamml:PostGISWriterType"/>
    <xs:complexType name="PostGISWriterType">
      <xs:annotation>
        <xs:documentation>A vector data model that provides a connection to a
PostGIS database</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="pamml:PostGISReaderType"/>
      </xs:complexContent>
    </xs:complexType>
    <!-- Oracle Spatial spatial data Model -->
    <xs:element name="OracleSpatialReader">
      <xs:annotation>
        <xs:documentation>A vector data model whose source is an Oracle
Spatial database</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="pamml:RDBVectorDataType"/>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <!-- ESRI SDE spatial data Model -->
    <xs:element name="ESRISDEReader">
      <xs:annotation>
        <xs:documentation>A vector data model whose source is an ESRI SDE
database</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="pamml:RDBVectorDataType"/>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <!-- VectorToRaster Model -->
    <xs:element name="VectorToRaster">
      <xs:annotation>
```

```
      <xs:documentation>A vector to raster conversion
model</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:RasterDataType">
          <xs:sequence>
            <xs:element name="ConversionInfo">
              <xs:complexType>
                <xs:attribute name="cellValue" type="xs:string"
use="required"/>
                <xs:attribute name="dataType" type="xs:anySimpleType"
use="optional"/>
                <xs:attribute name="cellSize" type="xs:double"
use="required"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="InputVector" type="pamml:VectorDataType"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- RasterToVector Model -->
  <xs:element name="RasterToVector">
    <xs:annotation>
      <xs:documentation>A raster to vector conversion
model</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="pamml:VectorDataType">
          <xs:sequence>
            <xs:element ref="pamml:RasterData"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!--
  ********************
  Alternatives Models
  ********************
  -->
  <xs:element name="Alternatives">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="pamml:Alternative" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Alternative" type="pamml:ModelType"/>
  <!--
```

```
  ********************
  Helper components
  ********************
  -->
  <!-- ********************************** -->
  <!-- *****  RemoteInfo     ***** -->
  <xs:element name="RemoteInfo" type="pamml:RemoteInfoType"/>
  <xs:complexType name="RemoteInfoType">
    <xs:annotation>
      <xs:documentation>Information about remote location and execution
possibilities</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
      <xs:element name="ModelLoc" type="xs:anyURI"/>
      <xs:element name="ModelRunnerLoc" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="LocalCache" type="pamml:LocalCacheType"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ********************************** -->
  <!-- *****  LocalCacheType  ***** -->
  <xs:complexType name="LocalCacheType">
    <xs:annotation>
      <xs:documentation>Information about local caching of the model and its
data</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Cached" type="xs:boolean"/>
      <xs:element name="CachedTime" type="xs:dateTime"/>
      <xs:element name="NextUpdateTime" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="LocalModel" type="pamml:ModelType"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ********************************** -->
  <!-- *****  Attribute   ***** -->
  <xs:element name="Attribute">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="dataType" type="xs:anySimpleType" use="required"/>
      <xs:attribute name="minVal" type="xs:string" use="optional"/>
      <xs:attribute name="maxVal" type="xs:string" use="optional"/>
      <xs:attribute name="query" type="xs:string" use="optional"/>
      <xs:attribute name="note" type="xs:string" use="optional"/>
      <!-- string, double or int -->
      <!-- XPath expression -->
    </xs:complexType>
  </xs:element>
  <!-- ********************************** -->
  <!-- *****  AttributeInfo   ***** -->
  <xs:element name="AttributeInfo" type="pamml:AttributeInfoType"/>
  <xs:complexType name="AttributeInfoType">
    <xs:annotation>
```

```
      <xs:documentation>metadata for record attributes. ordered list of
names and data types</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="pamml:Attribute" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ********************************** -->
  <!-- *****  Metadata    ***** -->
  <xs:element name="Metadata" type="pamml:MetadataType"/>
  <xs:complexType name="MetadataType">
    <xs:annotation>
      <xs:documentation>Helpful info</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Description" type="xs:string"/>
      <xs:element name="Reference" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="VisualPreview" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="FGDCMetadata" type="xs:anyURI" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ********************************** -->
  <!-- *****  Permissions   ***** -->
  <xs:element name="Permissions" type="pamml:PermissionsType"/>
  <xs:complexType name="PermissionsType">
    <xs:annotation>
      <xs:documentation>collection of user, group and other
Permissions</xs:documentation>
    </xs:annotation>
    <xs:attribute name="user" type="pamml:PermissionType" use="optional"/>
    <xs:attribute name="group" type="pamml:PermissionType" use="optional"/>
    <xs:attribute name="other" type="pamml:PermissionType" use="optional"/>
  </xs:complexType>
  <!-- A sequence of characters similar to Unix permissions.
    Characters that are understood are 'r', 'w', 'x', 'a' and '-'.
    r=read, w=write, x=execute, a=create alternative, -=no permission
    Full permission would be specified as rwxa. A '-' instead of
    one of those letters means no permission. For example:
    r-xa gives read, execute and create alternative permissions. -->
  <xs:simpleType name="PermissionType">
    <xs:annotation>
      <xs:documentation>A sequence of characters similar to Unix
permissions, rwx, plus an 'a'</xs:documentation>
      <xs:documentation>for alternatives allowed. 'u' is for
undefined.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="[rwxau]{4}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="GeometryType">
    <xs:annotation>
```

```
      <xs:documentation>A string identifying the geometry type of all
vectors in the data set</xs:documentation>
      <xs:documentation>taken from the "Simple Features for SQL" OGC
specification</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="point"/>
      <xs:enumeration value="linestring"/>
      <xs:enumeration value="polygon"/>
      <xs:enumeration value="multipoint"/>
      <xs:enumeration value="multilinestring"/>
      <xs:enumeration value="multipolygon"/>
      <xs:enumeration value="geometrycollection"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CompressionType">
    <xs:annotation>
      <xs:documentation>A string identifying a type of
compression</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="zip"/>
      <xs:enumeration value="gzip"/>
      <xs:enumeration value="targzip"/>
      <xs:enumeration value="bzip"/>
      <xs:enumeration value="tarbzip"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:attributeGroup name="globalAttributes">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="id" type="xs:string" use="required"/>
    <xs:attribute name="altOK" type="xs:boolean" use="optional"
default="true"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="rasterAttributes">
    <xs:attribute name="numCols" type="xs:int" use="required"/>
    <xs:attribute name="numRows" type="xs:int" use="required"/>
    <xs:attribute name="minX" type="xs:double" use="required"/>
    <xs:attribute name="minY" type="xs:double" use="required"/>
    <xs:attribute name="cellSize" type="xs:double" use="required"/>
  </xs:attributeGroup>
  <xs:complexType name="PassphraseType">
    <xs:attribute name="word" type="xs:string"/>
    <xs:attribute name="cryptoType" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="DataFileCompressable">
    <xs:attributeGroup ref="pamml:DataFileCompressableAtts"/>
  </xs:complexType>
  <xs:attributeGroup name="DataFileCompressableAtts">
    <xs:attribute name="dataFile" type="xs:anyURI" use="required"/>
    <xs:attribute name="compression" type="pamml:CompressionType"
use="optional"/>
  </xs:attributeGroup>
```

```
  <xs:simpleType name="ValueUnits">
    <xs:restriction base="xs:string">
      <xs:enumeration value="abstract"/>
      <xs:enumeration value="meters"/>
      <xs:enumeration value="kilometers"/>
      <xs:enumeration value="miles"/>
      <xs:enumeration value="feet"/>
      <xs:enumeration value="grams"/>
      <xs:enumeration value="liters"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="FocalOperation">
    <xs:restriction base="xs:string">
      <xs:enumeration value="add"/>
      <xs:enumeration value="subtract"/>
      <xs:enumeration value="multiply"/>
      <xs:enumeration value="divide"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ZonalOperation">
    <xs:restriction base="xs:string">
      <xs:enumeration value="add"/>
      <xs:enumeration value="subtract"/>
      <xs:enumeration value="multiply"/>
      <xs:enumeration value="divide"/>
      <xs:enumeration value="mean"/>
      <xs:enumeration value="variance"/>
      <xs:enumeration value="stddev"/>
      <xs:enumeration value="variety"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Appendix B. Glossary

C#: The preferred programming language for Microsoft's .NET Web services architecture.

COM: Component Object Model. A software architecture used by Microsoft's Windows operating system that allows applications to be built from binary software components.

CORBA: Common Object Request Broker Architecture. A platform-independent protocol for building distributed, platform-independent enterprise applications.

DCOM: Distributed Component Object Model. An extension of Microsoft's Component Object Model (COM) to that permits the sharing of program components across a network.

DSS: Decision Support System. Information technology and software that taps database resources to present information in a form that helps people at all levels of the organization make decisions.

EDI: Electronic Data Interchange. The exchange of highly standardized electronic versions of common business documents between computer systems through communications lines with standard contracts. Generally the contracts are formulated within each industry.

HTML: Hypertext Markup Language. A formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell HTML clients (e.g. Web browsers) how to display text, position graphics and form items, and display links to other pages.

HTTP: Hypertext Transfer Protocol. HTTP is the set of rules for exchanging files on the World Wide Web. Relative to the TCP/IP suite of protocols—the basis for information exchange on the Internet—HTTP is an application protocol.

GIS: Geographic Information Systems. Lately used to stand for Geographic Information Sciences, suggesting a true scientific discipline separate from the technology.

GML: Geography Markup Language.

IT: Information Technology. Includes all matters concerned with the furtherance of computer science and technology and with the design, development, installation, and implementation of information systems and applications [San Diego State University]. An information technology architecture is an integrated framework for acquiring and evolving IT to achieve strategic goals. It has both logical and technical components. Logical components include mission, functional and information requirements, system configurations, and information flows. Technical components include IT standards and rules that will be used to implement the logical architecture (from http://www.ichnet.org/glossary.htm).

.NET: Both a business strategy from Microsoft and its collection of programming support for what are known as Web services, the ability to use the Web rather than your own computer for various services.

OWL: Web Ontology Language. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

RDF: Resource Description Framework. A formal data model from the World Wide Web Consortium (W3C) for machine understandable metadata used to provide standard descriptions of web resources.

SOAP: Simple Object Access Protocol. A message layout specification that defines a uniform way of passing XML-encoded data.

SQL: Structured Query Language. A standard interactive and programming language for getting information from and updating a database.

UML: Universal Modeling Language. A standard notation and modeling technique for analyzing real-world objects, developing systems, designing software modules in object-oriented approach.

URL: Universal Resource Locator. The address of a resource, or file, available on the Internet. Consists of the protocol of the resource (e.g. http:// or ftp://), the domain name for the resource (e.g. www.example.com), and an identifying string. Most strings hint at their

underlying content. They often look like a file path (e.g. /pages/2003/song.mp3) or a command (e.g. /servlet/StockTicker?symbol=EFF).

WSDL: Web Services Description Language. Defines services as collections of network endpoints whose abstract definition of interfaces and messages is separated from concrete network deployment or data format bindings.

XML: Extensible Markup Language. The universal format for structured documents, messages, and data on the Web. XML is a meta-language (a way to define tag sets) that allows you to design your own customized markup language for many classes of information.

Appendix C. Bibliography

Alexander, Christopher, Sara Ishikawa and Murray Silverstein. 1977. *A pattern language: towns, buildings and construction.* Oxford, UK: Oxford University Press.

Anselin, Luc. 1992. *Spatial data analysis with GIS: an introduction to the application in the social sciences.* Santa Barbara, CA: National Center for Geographic Information and Analysis.

Armstrong, Joe. 2003. *Making reliable distributed systems in the presence of software errors.* Ph.D. dissertation, Swedish Institute of Computer Science. Dept. of Microelectronics and Information Technology.

Azad, B, and Lyna Wiggins. 1995. *Dynamics of Inter-Organizational Geographic Data Sharing: A conceptual framework for research.* In Harlan J. Onsrud and Gerard Rushton (eds). *Sharing Geographic Information.* New Brunswick, NJ: Rutgers Center for Urban Policy Research, 22-43.

Ba, Sulin, and Jan Stallaert. 2002. *Designing IT-Supported Market Mechanisms for Organizational Coordination.* Edited by Clyde Holsapple and Varghese Jacob. *Business Modeling: Multidisciplinary Approaches.* Boston: Kluwer.

Carlton, Dennis, W., and Jeffrey M. Perloff. 1990. *Modern Industrial Organization.* Glenview, IL: Scott, Foresman and Company.

CERN. 2004. "Engineering Data Management Glossary." <http://cedar.web.cern.ch/CEDAR/glossary.html>. June, 2004.

Curbera, Fransisco, William Nagy, and Sanjiva Weerawarana. 2001. *Web Services: Why and How.* IBM T.J. Watson Research Center. August 9, 2001.

Delen, Dursun, and Perakath C. Benjamin. 2000. *An Enterprise Modeling and Analysis Toolkit.* Edited by J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick. *Proceedings of the 2000 Winter Simulation Conference.*

Egenhofer, Max, and T. Bruns. 1995. *Visual Map Algebra: A Direct-Manipulation User Interface for GIS.* Edited by S. Spaccapietra and R. Jain. *Visual Database Systems 3, Visual Information Management,*

Proceedings of the Third IFIP 2.6 Working Conference on Visual Database Systems, Lausanne, Switzerland: Chapman & Hall, 235-253.

Evans, John D. and Joseph Ferreira, Jr. 1995. *Sharing Spatial Information in an Imperfect World: Interactions between technical and organizational issues.* Edited by Harlan J. Onsrud and Gerard Rushton. *Sharing Geographic Information.* New Brunswick, NJ: Rutgers CUPR, Chapter 27.

Evans, John D. 1997. *Infrastructures For Sharing Geographic Information Among Environmental Agencies.* Ph.D. diss. Massachusetts Institute of Technology.

Executive Office of Environmental Affairs. 2001. *Buildout Book: Where do you want to be at buildout?* Boston, MA: Executive Office of Environmental Affairs.

Foot, David. 1981. *Operational urban models.* London: Methuen.

Forsberg, Kerstin, and Lars Dannstedt. 2000. "Extensible use of RDF in a business context." *Computer Networks* 33, 347-364.

Gahegan, Mark, Masahiro Takatsuka, Mike Wheeler and Frank Hardisty. 2002. "Introducing GeoVISTA Studio: an integrated suite of visualization and computational methods for exploration and knowledge construction in geography." *Computers, Environment and Urban Systems* 26, 267-292.

Goodchild, Michael, Max Egenhofer, Robin Fegeas. 1997. "Interoperating GISs: Report of a specialist meeting held under the auspices of the Varenius project." Panel on Computational Implementations of Geographic Concepts December 5-6 Santa Barbara, California.

Goodchild, Michael, Max Egenhofer, Karen Kemp, David Mark, and E. Sheppard. 1999. "Introduction to the Varenius project." *International Journal of GIS* 13:8 731-745.

Gottschalk, Karl. 2000. "Web Services architecture overview: The next stage of evolution for e-business." *IBM Developerworks.* September 2000. <http://www-106.ibm.com/developerworks/web/library/w-ovr/>. March 18, 2004.

Gouin, D. & P. Morin. 1997. "Solving the Geospatial Data Barrier." *Geomatica* 51:3, 278-287.

Guhathakurta, S. 2002. "Urban modeling as storytelling: using simulation models as a narrative." *Environment and Planning B: Planning & Design* 29, 895-911.

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.*

Hodges, Christopher J. 2004. Fostering Land Use Dialog: Community preservation as a growth management strategy in Massachusetts. Masters thesis, Massachusetts Institute of Technology Dept. of Urban Studies and Planning.

Hopkins, Lewis. 1999. "Structure of a planning support system for urban development." *Environment and Planning B: Planning & Design* 26, 333-343.

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

Hopkins, Lewis, Nikhil Kaza, Varkhi George. 2003. "Planning Markup Language: Representing the Meanings of Plans and Regulations." July 2003. Leuven, Belgium: AESOP/ACSP Joint Conference.

Jacobs, Alan B. 1993. *Great Streets.* Cambridge, MA: MIT Press.

Jacqz, Christian, MassGIS Director, and Jane Pfister, Community Preservation Initiative GIS Coordinator. 2004. Interview by author. May, 2004.

Keller, S. F. 1999. "Modeling and sharing geographic data with INTERLIS." *Computers & Geosciences* 25, 49-59

Kock, Christopher. 2004. "Nike rebounds: how (and why) Nike recovered from its supply chain disaster." *CIO Magazine.* June 15. <http://www.cio.com/archive/061504/nike.html>. July 2, 2004.

Klosterman R. E. 1997. "Planning support systems: a new perspective on computer-aided planning." *Journal of Planning Education and Research* 17, 45-54.

Klosterman, R. E. 1998. "Computer applications in planning." *Environment and Planning B: Planning & Design* 25, pp.32-36.

Ledeczi, Akos, Miklos Maroti, Gabor Karsai, and Greg Nordstrom. 1999. *Metaprogrammable Toolkit for Model-Integrated Computing.* Nashville, TN: Engineering of Computer Based Systems (March), 311-317.

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.* Cambridge, MA: MIT Press.

Lynch, Kevin and Gary Hack. 1984. *Site Planning.* Cambridge, MA: MIT Press.

Mark, D., N. Chrisman, A. Frank, P. McHaffie, J. Pickles. 1997. *The GIS History Project.* <http://www.geog.buffalo.edu/ncgia/gishist/bar_harbor.html>. November 27, 2003.

March, James G., and Johan P. Olsen. 1976. *Ambiguity and choice in organizations.* Bergen, Norway: Universitetsforlaget.

McHarg, I. 1969. *Design with Nature.* New York: Wiley.

Meredith, P. H. 1995. *Distributed GIS: If its time is now, why is it resisted?* In Harlan J. Onsrud and Gerard Rushton (eds). *Sharing Geographic Information.* New Brunswick, NJ: Rutgers Center for Urban Policy Research, 7-21.

Muller, P. 2000. *Instant UML.* Chicago: Wrox.

MuniWireless.com. 2004. "New York City to deploy massive wireless broadband network." June 19, 2004. <http://www.muniwireless.com/archives/000369.html>. May 28, 2004.

Nedovic-Budic, Zorica, and Jeffrey K. Pinto. 1999. "Interorganizational GIS: Issues and prospects." *Annals of Regional Science* 33, 183-195.

Nedovic-Budic, Zorica, and Jeffrey K. Pinto. 1999. "Understanding Interorganizational GIS Activities: A conceptual framework." *Journal of Urban and Regional Information Systems Association* 11(1): 53-64.

Obermeyer, Nancy J., and Jeffrey K. Pinto. 1994. *Managing Geographic Information Systems.* New York: The Guilford Press.

Object Management Group. 2003. OMG Unified Modeling Language Specification. <http://www.omg.org/cgi-bin/doc?formal/03-03-01>. March 1, 2003.

Object Technology International, Inc. 2003. "Eclipse Platform Technical Overview". <http://www.eclipse.org/whitepapers/eclipse-overview.pdf>. March 1, 2003.

OpenGIS Consortium. 1999. *OpenGIS® Simple Features Specification for SQL: Revision 1.1.* <http://www.opengis.org/docs/99-049.pdf>. December 15, 2003.

OpenGIS Consortium. 2001. *OpenGIS® Geography Markup Language (GML) Implementation Specification.* <http://www.opengis.net/gml/01-029/GML2.pdf>. December 15, 2003.

OpenGIS Consortium. 2002a. *OpenGIS® Abstract Specification, Topic 12: The OpenGIS Service Architecture.* <http://www.opengis.org/techno/abstract/02-112.pdf>. December 15, 2003.

OpenGIS Consortium. 2002b. *OpenGIS® Web Feature Service Implementation Specification.* <http://www.opengis.org/techno/specs/02-058.pdf>. December 15, 2003.

Preston, Michael, Peter Clayton, and George Wells. 2003. "Dynamic run-time application development using CORBA objects and XML in the field of distributed GIS." *International Journal of Geographical Information Science* 17:4, 321-341.

Schuur, J. 1994. "Analysis and Design in Computer-aided Physical Planning." *Environment and Planning B: Planning & Design* 21, 97-108.

Shiffer, Michael. 1992. "Towards a collaborative planning system." *Environment and Planning B: Planning and Design* 19, 709-722.

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

Tomlin, C. D. 1983. "A Map Algebra." *Proceedings of the 1983 Harvard Computer Graphics Conference*, Cambridge, MA.

Waddell, Paul, Alan Borning. 2004. "UrbanSim." May 24, 2004. Seattle, Washington, USA: National Conference on Digital Government Research.

Weston, A. 1992. *A Rulebook for Arguments.* Indianapolis: Hackett.

Wiggins, Lyna, and Michael Shiffer. 1990. "Planning with Hypermedia." *APA Journal* (Spring), 226-235.

World Wide Web Consortium. 2001. *XML Schema Part 0: Primer. W3C Recommendation.* 2 May 2001. <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>. June 23, 2003.

World Wide Web Consortium. 2001. *XML Linking Language (XLink), version 1.0. W3C Recommendation.* 27 June 2001. <http://www.w3.org/TR/2001/REC-xlink-20010627>. June 23, 2003.

World Wide Web Consortium. 2004. *XML Query 1.0 and XPath 2.0 Full-Text. W3C Working Draft.* 09 July 2004. <http://www.w3.org/TR/2004/WD-xquery-full-text-20040709>. June 23, 2003.

World Wide Web Consortium. 2004. *Web Services Definition Language (WSDL), version 2.0 Part 1: Core Language. W3C Working Draft.* 26 March 2004. <http://www.w3.org/TR/2004/WD-wsdl20-20040326>. June 23, 2003.

Yeh, Anthony. 2004. "Guest editorial: Planning, government, information, and the Internet" *Environment and Planning B: Planning & Design* 31 pp.163-165.

3130-91