Supporting Agile Modeling through Experimentation in an Integrated Urban Simulation Framework

Travis Kriplean[†], Alan Borning[†], Paul Waddell[‡], Christoffer Klang[+], and James Fogarty[†]

[†] Department of Computer Science & Engineering, University of Washington
[‡] Department of City and Regional Planning, University of California Berkeley
[+] Department of Computer Science & Communications, KTH, Stockholm

ABSTRACT

Decisions regarding major urban transportation projects and land use policies are frequently political and controversial, as well as having significant economic, social, and environmental consequences. UrbanSim is a disaggregate, behaviorally-realistic modeling environment that planning agencies can use to simulate the long-term effects of such decisions. We describe UrbanSim's evolution over the past decade from the perspective of supporting its appropriation by urban modelers, and identify support for experimentation as a key property that enables the adoption of an agile modeling methodology. Finally, we draw out three lessons for supporting agile modeling through experimentation: iterative development of models, providing appropriate domain-specific building blocks, and balancing the development of integrated tools versus interoperating with existing tools and the work practices that surround them.

Keywords

Simulation, urban modeling, agile modeling, appropriation, experimentation, UrbanSim

Categories and Subject Descriptors

I.6.7 [**Simulation and Modeling**]: Simulation Support Systems

1. INTRODUCTION

In cities worldwide, there are concerns about such issues as traffic congestion, resource consumption, and sustainability. Each region also has its own concerns about, for example, housing affordability, freight mobility, sprawl, or open space. Elected officials, professional planners, and interested citizens grapple with these difficult and controversial issues as they develop and evaluate alternatives, such as building a new freeway, establishing an urban growth boundary, or adopting policies such as congestion pricing.

dg.o 2010, May 17-20, 2010, Puebla, Mexico.
ACM [ISBN] 978-1-4503-0070-4/10/05

Urban modelers are tasked with modeling and simulating the complex interacting effects of land use and transportation alternatives for the purpose of informing public decision making. However, standard modeling systems are generally inadequate for modeling the full range of policies and effects of concern. Moreover, they are nearly exclusively black boxes that constrain modelers in tailoring models and scenarios to a region's circumstances.

The UrbanSim system provides an environment for modelers to project the impacts of land use and transportation projects over periods of 20 to 30 years [8, 28, 32]. It is an integrated framework for creating and configuring new land use, transportation, and environmental models, as well as facilities for running simulations and exploring results. Modelers can reach deep into the modeling system to radically change existing models, incorporate entirely new models, and to produce novel urban simulations. The framework has been put into operational use in a range of cities worldwide.

While each version of UrbanSim has become increasingly open and flexible, its flexibility has been inaccessible to all but the most technically proficient modelers. Even then, significant interaction with code was necessary. The overhead of finding and appropriating features into modeling practice was a barrier to modeling efforts. The complexity of the application made UrbanSim's flexibility nearly as problematic as the long tool chains of data, blackbox models, and other software that characterize urban modeling efforts outside UrbanSim.

In this paper, we use the development of a user interface for UrbanSim as an opportunity to examine, from the perspective of appropriation, a decade of work attempting to improve the state of the art in urban modeling technology and practice. The GUI provides an integrated modeling environment designed to render UrbanSim's flexibility actionable and appropriable into a modeling process characterized by quick and fluid iteration between data preparation, model creation and specification, scenario construction and execution, result analysis, and interaction with policy stakeholders. We call this methodology *agile modeling*. It is the targeted transformation in modeling practice that we aim for when we discuss the appropriation of UrbanSim into a planning agency. Agile modeling is a methodology informed by years of interactions with other researchers and practitioners, as well as our own team's modelers. The transformation is particularly challenging in that each phase of modeling usually involves different people, professional expertise, and sets of standard tools.

We identify the centrality of supporting experimentation for the appropriation of an agile modeling methodology. By "experimentation" we mean the iterative exploration of the modeling space and refinement of the modeling application, based on principled trial and error guided by expertise, intuition, experience, and communication with others. We unpack three lessons for supporting agile modeling through experimentation: (1) emphasize getting a full modeling pipeline set up before worrying about perfecting any part of it, (2) provide appropriate building blocks in a domain-specific representation that can be manipulated by modelers, and (3) balance the integration of functionality with interoperating with various professional tools with advanced phase-specific functionality.

The data presented here draws on a range of experience in developing and applying UrbanSim. One source of data is a lab study of research and practicing urban modelers who were brought in to evaluate the GUI. The task-based structure of the study alongside of ample interspersing of semi-structured interview questions allowed us to gather data about their current practice, their use of previous versions of UrbanSim and how the GUI might transform that practice. We also draw draw on formal and informal interactions with users during other lab studies, user group meetings, field deployments, long-term software development work done in concert with a number of planning agencies, and our multi-disciplinary research group's expertise in urban modeling.

The paper is organized as follows. We first give a description of the political and organizational context of urban modeling, what the work of modeling typically looks like, and a description of agile modeling. We then describe the three most recent versions of UrbanSim, put the phases in a trajectory, explain how each system was designed for increasing the appropriability of UrbanSim, and unpack the limitations that were encountered as a result of the development decisions. We then discuss key lessons about supporting agile modeling through experimentation.

2. URBAN MODELING

Urban modeling was established in the 1960's as a computer-based profession for predicting the possible effects of urban planning decisions. Now, virtually all large metropolitan regions in the developed world have a government agency that does some sort of modeling to help predict future trends and help evaluate proposed alternatives. Unfortunately, current practice is still dependent on inadequate modeling systems. These systems inhibit the accuracy of simulations, particularly in the ability of modelers to create models that account for regional differences.

We seek to provide an integrated modeling environment that modelers can use to construct, integrate, and share new models that account for local contingencies. We adopt the term *tailoring* [26] to denote this process.[1] Our goal is to foster a "tailoring culture" [17] that encourages and supports the rapid application of a modeling system to local conditions and the process of exploring alternatives within that space.

[1]This is in contrast to *customization*, which is used to connote how the system can be tweaked in order to get it to work in a particular setting [1, 10], a process usually accomplished early in the deployment of an infrastructure or application.

For new modeling technology to be put into practice, it needs to be appropriated into multiple contexts of use that cut across organizational, political, and geographic boundaries. These contexts of use pose significant challenges for appropriation. For example, even though it is well known that land use decisions are impacted by transportation decisions and vice versa, historical developments have led to the professional separation of land use and transportation modeling.[2] Land use planning and research takes place in departments of architecture and urban planning, while transportation planning takes place in departments of civil engineering. Student education reflects this split. It is also typically perpetuated in planning organizations, with separate departments for land use and transportation. Independent modeling systems and other tools further reify this artificial separation. Most modeling systems simulate either land use or transportation, but not their interaction.

Moreover, regional government planning agencies operate under considerable political pressure. For example, U.S. federal funding for highways is at risk in regions that violate air quality conformity requirements; planning agencies are sometimes charged with predicting the probability of violating these emission standards. While standard modeling tools are inflexible and known to be inaccurate, their use as *de facto* standards provides a certain professional safety net in their use. Their limitations are well understood and many agencies employ them. Adopting new modeling systems can therefore pose professional and political risk. At the same time, existing models and the decision-making process have come under increasing scrutiny and criticism since the 1980's, leading to substantial pressure to revise both [4]. Planning agencies that do not try to make their models more realistic run the risk of a major lawsuit [14].

Planning agencies are thus left with two unappealing prospects: use crude models with major known inadequacies, or use relatively new, experimental systems that may open their agency to professional and political risk. One of our major research goals with the UrbanSim system is to make its appropriation technically, professionally, organizationally, and politically appealing. In the rest of this section, we describe necessary background about modeling practice. This background is meant to contextualize the system development we describe later. We end by describing the agile modeling approach that we seek to support.

2.1 Modeling a Region

For those agencies that undertake a full modeling effort, the process can be roughly divided into four phases. The exact nature of these phases will vary, but the basic outline is likely to be similar across agencies. Each phase is a significant collaborative undertaking. Different staff members in the agency will have different areas of responsibility as well as expertise, for example, in using geodatabases and GISs, econometric software, and in running travel models.

Preparing input data. The first phase is preparing the input data. Different systems have quite different requirements here and it can resource intensive [30]. For

[2]Land use modeling entails building predictive models of where people will live, where jobs will be located, and where real estate development or redevelopment will occur; transportation modeling entails predicting such things as traffic volumes, mode share, and delays due to congestion.

UrbanSim, a preliminary application might involve a few person-months, but data for more realistic use might involve a person-year or more of effort. The data comes from multiple sources, including census, employment, land ownership, transportation network, and environmental data. This data will usually need to be converted to the format required by the modeling system. A larger problem is that there will often be missing or erroneous data, which needs to be filled in or corrected. Much of the data is geographic, and the preparation will involve GIS tools and expertise, usually also involving facility with SQL database management.

Specifying and estimating component models. The next phase involves fitting the system to the region being simulated. The system will typically use a set of predictive variables (or "model variables") to determine its behavior. For example, one component of any land use model will be simulating where people decide to live. This simulated decision will be based on factors such as household income and number of children. Each factor is represented as a model variable. Their coefficients are estimated to the data for the given region. A major difference among modeling systems is whether the set of model variables is fixed or extensible, and whether the estimation is done using external econometric software or is integrated with the modeling environment itself. These are key properties, since a fixed set of model variables will generally not capture particular features that are important in one region but not others. For example, in the southern U.S., it is important to incorporate soil type variables in land price models because red clay is harder to build on than other soil types.

Creating and running scenarios. Once the data is available for the region and the models have been estimated, the simulation can be run for the region. Regions will normally have an agreed-on "baseline" long-range transportation and land use plan, and this can be simulated for 20–30 simulated years. As packages of transportation improvements, zoning changes, and other policy alternatives are proposed, these can be simulated as well. For example, one such package (or "scenario") might involve extending a freeway to an exurban area and rezoning it for suburban development.

Examining results and preparing reports. Finally, results from the simulations are reported as *indicators* and examined. While the model application is being applied and debugged, many of these indicators will be diagnostic. Later, as the modeling moves into operational use, indicators that are useful for assessing how well alternate scenarios support or undermine different regional goals will become important. Still later in the process, the simulation results will be used in formal reports, presentations, and press releases.

2.2 Agile Modeling

The different phases of modeling an urban region do not proceed in a linear order. Instead, the process involves iteration, backtracking, and refinement. For example, early examination of the results is essential in helping refine the model specification. A common sequence is to run the estimation with an initial specification that contains key variables, then examine the significance level of each variable, and from this decide which variables to keep and which to eliminate. This usually requires several iterations and sometimes the creation of new and more suitable variables. Creating a comprehensive list of good predictor variables is often a challenging task requiring considerable experimenting with the model and the data. This process must be done for the entire ensemble of component models. This process may reveal problems in the data, requiring going back and fixing up the data prepared in the first phase. Touching still more stakeholders, as the results are used in policy assessment, if a proposed scenario isn't achieving regional goals, it may be necessary to try modified plans.

Current tools make this process sufficiently cumbersome that it is impossible to explore very many policy alternatives, let alone to quickly explore them in response to citizen or policymaker interest. In many ways, tools do not enable modelers to do the jobs they actually want to do. Instead, they become trapped by standardized models, spending much of their time trying to make the data they have conform to its requirements. As a participant in our lab study (described later) states, *"Not everyone has the same data... You typically need to whip it to make it like data used in other cities so that it can be used by the model."*. One participant in our lab study called the current standard land use model (DRAM/EMPAL) "lockstep" and stated that its lack of flexibility to incorporate new variables rendered it practically useless for informing public policy.

There is widespread desire for the process to be much faster and responsive to policy questions [33]. In analogy with agile software development processes [3, 9], we call this desired process "agile modeling." There are significant parallels between agile modeling and software development, including the importance of gracefully accommodating change, always having a running system, and automated testing. However, a major difference is the centrality of experimentation in agile modeling (as developed in the remainder of the paper). The importance of experimentation arises in the need for ongoing refinement of models, increasing the scope of analysis the models are capable of, and adaptation to evolving political conditions.

We define agile modeling as a process:

- that can respond quickly to changing requirements
- that enables fluid experimentation
- that supports the iterative development of a model from a crude but functioning prototype through a more polished, operational version
- that supports continuous assessment of simulation results

3. URBANSIM: SYSTEM AND EVOLUTION

UrbanSim is an open source modeling framework implemented as a set of interacting component models that simulate different actors or processes within the urban environment at a disaggregate, behaviorally-realistic level. It has been developed by an interdisciplinary group over the past decade, involving researchers from urban design and planning, computer science, public affairs, and statistics.

In this section, we give a high level description of UrbanSim's evolution through three distinct phases. The section serves three purposes. First, it unpacks how design decisions have been impacted by the goal of supporting the appropriation of an agile modeling practice, and what the

results of the choices were. Second, it sets the stage for our later analysis of the centrality of experimentation for adopting an agile modeling methodology. Third, it highlights the data sources from which we make our claims.

3.1 UrbanSim 1-3: Java + Eclipse IDE (c. 1995-2004)

The first full implementation of UrbanSim was a prototype application to Eugene/Springfield, Oregon in 1996 [27]. In this version, model variables were hard-coded, so that any change to the model specification required programming changes. UrbanSim 3 was a full re-implementation of the system in Java, intended to improve robustness and performance [21, 29]. Features included the coupling with a range of other packages and applications, notably the MySQL database for storing configuration information, input data, results, and other information. Software development followed agile programming methodologies, including extensive unit tests, source code versioning, and a homebrew continuous build system [11].

UrbanSim 3 represented a considerable advance in the state of the art in integrated urban modeling and a small but enthusiastic user community grew around it. Prototype applications were developed for a number of regions, including Salt Lake City, the Puget Sound region, and Houston. Nevertheless, there was significant interest in being able to better tailor the model specifications to a region. For example, users preferred moving from using grid cells to describe locations and real estate, to using parcels and buildings, which are more directly observable and behaviorally realistic units of analysis. Further, the process of tailoring to a new region involved substantial experimentation and effort. The process was hampered by the complexity of the task of, for example, adding or changing a model variable, which required changes to the Java codebase. Despite efforts to make the code modular, readable, and well-commented, the modelers understandably were reluctant to touch the code. Any change required involving software developers.

To mitigate this, we implemented a GUI for UrbanSim as an Eclipse IDE plug-in. The GUI provided basic functionality for running a simulation and accessing the data. However, the complexity of the surrounding Eclipse functionality was foreign and disconcerting to many modelers—they could never incorporate it strongly into their practice. While the extensive use of unit tests was valuable, and worked well at the level of checking the functionality of individual components, the division between the modelers and the developers hindered testing overall system functionality and an agile process for tailoring to a region. The relationship between modelers and developers was more akin to a contractual relationship than a tight collaboration.

3.2 UrbanSim 4: Python + Opus (c. 2004-2008)

The current version, UrbanSim 4, is implemented using a new framework called Opus (Open Platform for Urban Simulation) [31]. Opus is written in Python, has a much more open and flexible architecture, and makes extensive use of configuration objects. A data architecture manages caches of simulation data, optimized for the retrieval and manipulation patterns used in the choice and regression models that are the backbone of UrbanSim. It also manages lazy loading of data (to keep memory use bounded) and on-demand computation of new or changed model variables. These model variables can be defined in a domain-specific programming language that we designed and implemented, supporting more effective tailoring to a new region. In addition, UrbanSim 4 includes integrated tools for estimating models, producing indicators, and interoperating with SQL databases and GIS (Geographic Information System) software. The resulting system is highly flexible and configurable.

In early UrbanSim work and in other existing tools, estimating models required using a proprietary, external econometric package. Model estimation is in the inner loop of modeling practice, and modelers have universally reported that the process of transferring data and estimation results when using external econometric software is laborious and error-prone [31]. Estimating the parameters for a model might require weeks of effort. Implementing estimation routines within the Opus framework accelerated this process substantially. But accessing these estimation routines required interacting with them via commands to the Python interpreter, and writing Python scripts to manage configurations. While the system had become highly modular and configurable, the complexity of the configuration objects as Python dictionaries, scattered among hundreds of modules, still restricted the number of modelers for whom this functionality was accessible. A few researchers who were proficient in Python, principally modelers on our own team, could quickly configure new variants of models, but this process was impenetrable for most users. To others, the distributed, configurable, and modularized scripts constituted a "script hell", as one of our lab study participants put it.

3.3 UrbanSim 4: Python + GUI (c. 2008-2010)

As noted above, modelers outside the UrbanSim research group were generally unable to take full advantage of the flexibility of Opus and UrbanSim 4. The goal of the Opus Graphical Interface is to make this rich functionality accessible to a broader range of modelers to engender a qualitative shift to an agile modeling practice.

The first strategy we followed in increasing accessibility is to move the configuration objects from Python dictionaries into external project files stored as XML. These project files contain the configuration information needed to apply UrbanSim to a particular region, as well as model variable definitions, tool scripts, and other information. Project files support inheritance, so one project can add or override whatever additional information is needed beyond the default. Inheritance supports coordination between our research group and the various planning agencies for making high-level changes to the modeling system, and we hypothesize that the easily sharable XML format will facilitate easier collaboration between different participants in the modeling effort. We believe that XML with inheritance will thus act as a boundary object that can tie together aspects of the modeling process that have previously been caught up in complex tool chains that are burdensome for supporting collaboration.

The second strategy is the development of a GUI that is driven by the XML project files. This allows different sets of functionality to be provided for students in a class, GIS analysts working on data preparation, modeling experts who will be specifying and configuring models, or policy analysts

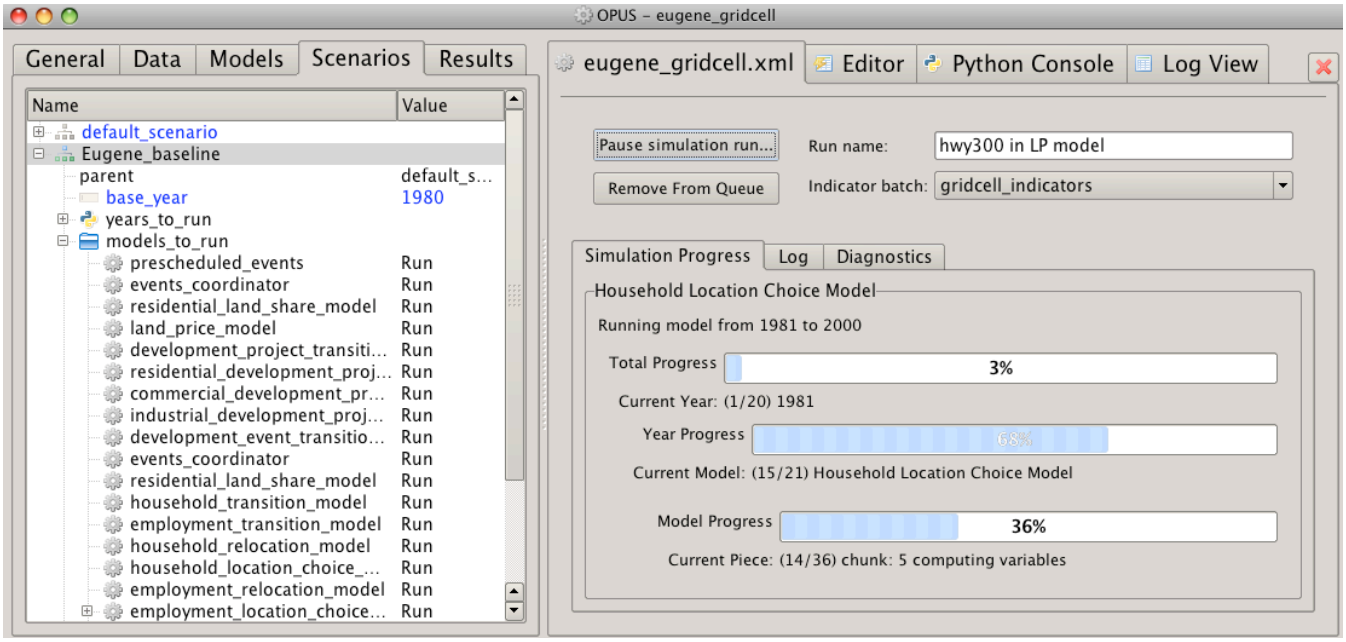

Figure 1: The Opus/UrbanSim Graphical User Interface

who will be primarily running scenarios and examining results. After starting the GUI, the first action is opening a project file. The user is then presented with a window with two major panes (Figure 1). There are five tabs on the left pane: General, Data, Models, Scenarios, and Results. The General tab has information such as the name of its parent configuration (from which it inherits). The remaining tabs mirror the four phases of modeling. The Data tab has two subtabs: one providing a set of tools for interoperating with SQL and geodatabases, and another for examining UrbanSim data stored in local caches. The Models tab provides access to the functionality for creating, specifying, and estimating new or existing models, including selecting variables and defining new ones. The Scenarios tab supports running different scenarios, including diagnostic tools to allow the simulation to be monitored as it runs. Finally, the Results tab provides tools for interactive exploration of the results of simulations and for producing batches of indicators to be computed and visualized on demand. It also includes tools to export indicator results to SQL databases and mapping software. The GUI also provides a Variable Library where new model variables and indicators can be defined using the expression language. This design was informed by extensive discussions with modelers and through testing with early prototypes. The division into the four tabs came primarily from work with the Phoenix area metropolitan planning organization, although it also mirrors other standard practice.

Field deployments. Probably the most convincing evidence of the worth and utility of the system is information regarding field adoption. UrbanSim has been applied operationally in Detroit, Houston, the Puget Sound region in Washington State, and Salt Lake City. The research team has also worked with other agencies in applying UrbanSim in the urban areas around Eugene, Honolulu, Phoenix, and San Francisco. There have also been research and pilot applications in such diverse regions as Amsterdam, Burlington, Durham, El Paso, Melbourne, Tel Aviv, and Zurich. Four Users Workshops have been held in the U.S. and Europe. Modelers outside the UrbanSim research group have also begun writing papers describing how they applied UrbanSim to Paris, Brussels, and Lausanne[20, 25].

Lab study. To augment the information obtained from field adoption, we undertook a qualitative lab study to help assess UrbanSim 4, including the GUI and its integrated domain-specific language. The tasks UrbanSim supports cannot be fully investigated in a lab study. Instead, our goal was to provide experts with a context for discussing their practices, limitations of their existing systems, and how the GUI could enhance their work, as well as identifying usability issues. Our study included ten participants. Of these, six were practitioners working for regional planning organizations and four were university-level researchers (faculty or Ph.D. students). Geographically, eight worked or studied in the U.S. and two in Europe. All had prior experience using the command-line version of UrbanSim, ranging from use in a class to being involved in multi-year applications to a particular metropolitan region.

We gave each participant a set of agile modeling tasks that exercise each of the four main components of the GUI, and asked the participants to talk aloud as they worked through the tasks. A semi-structured interview format was used to intersperse specific questions that probed their understanding of what the system was doing with general questions about modeling practice and their current use of UrbanSim or other systems. Each sessions lasted approximately two hours. Most of the sessions were conducted remotely by speakerphone and screensharing software. Local participants came to our lab. No one had any problems finished any of the tasks.

The task order mirrored the four principal tabs of the GUI. The first set of tasks concerned the tools for inspect-

ing and querying the system's database of inputs and results, and were designed to probe the issue of providing tools within the GUI itself versus interoperating with external tools. The next set of tasks concerned constructing, specifying, and estimating models. We first had the participants construct a new regression model for land price from scratch, estimate it, and check the coefficients and T-values for the results. The next portion asked the participants to investigate the effects of proximity to a highway on land price, which eventually entailed them using the variable library to define a new model variable. Then they substituted it in to the model specification and re-estimated the model. This is a snapshot of a very typical modeling activity, which provided a concrete example around which to ask a set of questions regarding model construction and estimation.

The third set of tasks concerned modifying a scenario configuration and running the simulation. The last set of tasks focused on producing tables and maps of indicators from the results of the simulation, including a task that entailed creating a new indicator. This task required a more complex expression, involving averaging the population densities in each gridcell contained in the zones. Doing this using external tools would require using multiple systems and involve a fair number of steps. They were also asked to use the GUI to create, configure, and execute a script that generates indicators for each of many runs. These two modes of interaction, interactive browsing and report generation, mirror two typical modes of interaction with UrbanSim results.

4. LESSONS FOR SUPPORTING AGILE MODELING

In this section, we give three lessons for designing in support of agile modeling. We pay particular attention to enabling fluid and ready-to-hand experimentation across all phases of the modeling effort in order to engender a tight loop between configuration, execution, and evaluation of the result.

4.1 Emphasize iterative modeling

One clear recommendation is to construct an imperfect but full working pipeline as soon as possible, a foundation on which progressive refinement of data, models, and scenarios can then take place. This is a lesson learned somewhat painfully by our own group when applying UrbanSim to the Puget Sound region in Washington State. UrbanSim models are data-hungry, and preparing the data for producing realistic simulations is a very large task. Modelers on our team and at the agency spent about two years on very careful data preparation and cleaning, but found that the subsequent iterative process of diagnosing models and refining them led to decisions to significantly restructure the data and the models. Others have discovered this lesson as well, for example as described in an UrbanSim experience paper by Patterson and Bierlaire [25], and also confirmed in our empirical study. Instead, we recommend that the modeling effort be approached iteratively and experimentally. Start with *some* albeit imperfect form of the data, so that one can begin doing simulations and experiments, and then iteratively refine the input data interleaved with work on subsequent steps. Recent changes to the UrbanSim modeling environment provide additional support for such iterative modeling, particularly in the XML-driven GUI.

Facilitate exploration and learning with project templates. The possible datasets, models, model variables, and scenarios that might be created and configured in the UrbanSim environment are immense. UrbanSim 4 provides a particularly flexible framework, but as we learned, this flexibility can be daunting to explore.

To facilitate experimentation on a full working pipeline at an early stage, we distribute a sample project configuration that contains full datasets, configured models and variables, and sample scenarios for a fully-fledged model application for Eugene (and more recently for Seattle as well). Several participants of the lab study, and other users we have talked with, indicated that this led to a particular strategy for applying UrbanSim to a new region: start with the sample Eugene project, and one-by-one replace the Eugene-based datasets with ones for the region at hand. In other words, gradually morph Eugene into Paris (or whatever the city might be). This reflects Trigg and Bødker's [26] observation that "tailors start with concrete realizations, using generalization from experience to work backwards toward (re)design and analysis." This strategy has some merits, in terms of starting with something that works. But it also has some clear downsides, as noted by participants in our study. First, parts of the data that are not completely understood, or that cannot be updated because of lack of data, will remain as Eugene. Also, models that are relevant for Eugene may not be relevant for Paris and vice versa. The tendency to use this strategy is parallel to what Balka and Wagner [1] found in the appropriation (or lack of appropriation) of a wireless call infrastructure, which remained configured to mirror the previous system when it was meant to only be a starting place.

Lab study participant P8 indicated that he would like to see two types of sample projects: a bare-bones project and a full example. A bare-bones project is meant to "facilitate learning" about how a model system "hangs together", while a full example shows what is possible in a complete project (e.g., Eugene). P8 insisted on both because of a tension: the full example can show a real application that pushes the limits of the modeling system; but it can be difficult to determine what in the configuration is essential versus what provides additional accuracy in the case of Eugene (but not necessarily some other city). On the other hand, if we provide just a bare-bones project, without examples that demonstrate the range of capabilities, P8 fears that modelers will not flesh out the model.

Even with a GUI and example projects at hand, the model configuration space is daunting to explore. A number of our participants expressed their desire to share their configurations with other regions, and to be able to explore how other regions have configured their model systems. They envisioned the development of a shared, centralized variable library where modelers from multiple regions could download, upload, comment on, and document variables. Participants also frequently expressed an interest in sharing their model configurations, for example, by releasing data flow diagrams. Experimentation within a region might be greatly facilitated by leveraging the work done in other regions, and the establishment of a professional network around UrbanSim would have increasing returns on appropriation in other regions.[3]

[3]A professional network might help mitigate the risk of adopting tailorable modeling systems like UrbanSim. The central site could be designed to allow modelers to post their

This desire to share configurations to facilitate collaborative exploration of the possible uses of a system reflect early work by Mackay [16] who drew attention to its importance.

Enable continuous reflection about the modeling effort. Convenient tools for continuously assessing and testing the modeling results are needed, ideally integrated with the GUI. We originally provided these as external tools, then in some cases with scripts that could be run from the command line. Some of these, such as automatic diagnostic map creation during simulation runs, are now available in the GUI.

This is an area that our lab study showed could use more attention, which we have subsequently acted upon. For example, P3 stated that he wants to have the ability to automatically generate a data flow diagram that maps the input data to the model variables to the models, as well as how models are configured to interact. He pointed out that standard modeling systems usually provide such a diagram in their manual and they are very helpful in understanding and diagnosing problems. However, because of UrbanSim's flexibility, regions' data flow diagrams will vary, and, if modelers are tailoring, the model flow will likely change throughout the application process. Thus, the manual approach would not be enough. The data flow diagram would then help support modelers as they experiment with different model configurations. It could also serve as a boundary object for communicating with everyone involved in the modeling effort, including on the policy side for whom modelers need to justify their modeling decisions. Since the study, we have implemented this functionality.

4.2 Provide appropriate domain-specific building blocks

A fundamental requirement is that the controls for the simulation be conveniently available to the modelers. However, we have found that a further level of control is crucial for agile modeling: the underlying building blocks need to be made accessible so that modelers can configure existing component models and construct new ones in more flexible ways. Beyond being accessible, they need to be made actionable and reconfigurable in a fluid manner, using representations with which the modelers are familiar. By providing these building blocks, domain experts are empowered to experiment and iterate more effectively. For modelers, these include datasets, model variables, models, and indicators.

A core section of the GUI was devoted to the creation of an interface for modelers to interact with data, drawing on the tools they are familiar with, such as ArcGIS, and SQL databases, and providing support to move data between those environments and OPUS. In addition, a plug-in architecture was generated to allow both core developers and other modelers to add tools and groups of tools to prepare input data and address data problems. For example, we are currently adding interfaces to tools that employ machine learning algorithms to support data imputation for missing data and outliers.

The UrbanSim 3 GUI provided the basic simulation controls, and its SQL database represented variable coefficients for the models in an easily editable way. However, the majority of changes to the basic building blocks of a modeling application, such as adding new model variables and incorporating them into a model, required programming in Java. In the command-line version of UrbanSim 4, the model components (such as superclasses for discrete choice models or regression models) are provided in a more modular fashion, and the use of Python allowed technically savvy modelers to construct new models themselves by programming in Python. Moreover, the design and implementation of the Tekoa domain-specific language for defining model variables made this aspect of tailoring much easier and faster, encouraging experimentation [7].

But while the command-line version of UrbanSim 4 lowered the barriers to tailoring, the need to interact extensively with the source code limited the actual tailoring modelers were able to accomplish. Even for those with programming experience, the overhead of figuring out what to change was often prohibitive. Three participants had previously created a variable (one had a more technically-inclined student implement it for him) and four had incorporated a variable into a model. Even one of our most technically sophisticated users remembered spending several hours figuring out what to change, computing a variable, and verifying whether it was correct. For one team, instead of having the capacity to experiment with and set model coefficients, they borrowed their coefficients from a large nearby city to be able to begin experimenting with the system. This lack of agility in experimentation can lead to poor models and simulation results with consequences for their use in public policy deliberation.

The Opus/UrbanSim 4 GUI provides access to these building blocks for non-programmers, including tools for adding new variables to the expression library and for creating new models using graphical model templates. Feedback from user studies and user community, has been very positive about these developments. P2 remarked that *"[The GUI] influences me to go more deeply into UrbanSim."* Nearly everyone said they felt variable creation through the expression language and the Variable Library significantly lowered the barrier on creating a new variable. P1 thought that their team could probably create their own model internally with the GUI relatively easily without hiring new staff with programming experience. P8 said it would allow a shift in the type of work that he would be able to do. *"Flexibility has typically been hindered by the need for programming expertise and by simple time constraints. Now I don't have to spend as much time creating a variable rather than analyzing."* As he worked through the process of building a new model from scratch in the GUI and specifying it, P3 noted that the extent and ease with which the GUI allows configuring a model system to a region's needs is unprecedented. *"Even though people know that [UrbanSim] is supposed to be open and flexible, people don't really know whether that flexibility can be explored...the GUI is allowing this flexibility to be more accessible and more apparent"*. A flexible system does not immediately mean that it can become part of an agile process. The building blocks need to be expressed in the proper domain-specific representations.

The technique of providing appropriate domain-specific building blocks has been used in other domains. As two early examples, ThingLab [5] was a constraint-based simulation laboratory. It was a "kit-building kit," in which the underlying constraint/object system could be used to construct graphical building blocks (e.g., resistors and batteries, or constrained geometric shapes) that could then in turn be

current model configuration and get it vetted by modelers from other regions. As a whole, it could serve as a sort of distributed safety net. This, however, is a hypothesis to be investigated in future work.

used as building blocks for the given domain (e.g., electrical circuits or demonstrations of geometric theorems). Oval [18] supported "radical tailorability" in the CSCW domain by providing objects, views, agents, and links as building blocks; from these, a set of well-known CSCW applications were re-created.

4.3 Balance integration and interoperability of tools

Central to modelers' ability to experiment effectively with the modeling environment is how ready-to-hand the relevant functionalities are, and the relative ease of collaborating with other members of the modeling team. Unfortunately, current practice typically involves analyzing, reformatting, and transferring data through a chain of disparate tools and people. These chains can be problematic because they (1) often take a long time (actively formatting data and passively waiting for data transfer), (2) interrupt task flow with frequent context switches to different interfaces for data manipulation, transfer and analysis, and (3) are prone to error. Consider P3's experience moving back and forth between the proprietary modeling software he was using when iteratively updating the specification of a logit model. *"I need to take coefficients from LIMDEP and then transfer them by hand to MEPLAN, very frustrating given the different syntax and input specifications. There is an enormous chance for error — it's kind of a minefield. And if I need to change the specification, I need to go back to LIMDEP and do it again!* Other common tool chains include transfer between (1) a SQL database for examination in a database browser to find errors in input data and to analyze results, (2) statistical analysis tools such as SAS, R, or SPSS to specify and estimate models, and (3) GIS mapping tools such as ArcMap or QGIS for spatial data analysis.

These tool chains can place a high overhead on experimentation. It may therefore be desirable to internally replicate the functionality either by reimplementing features the tool provides or, if possible, using a well-supported library that provides the functionality directly (or seamlessly interfaces with the external tool). The primary reason to consider this option is if the tool is critical to the "inner loops" of the modeling process, especially if the existing chain is onerous. A good example is estimation software. In UrbanSim 3, we used external econometric tools, such as LIMDEP. This was painful: first the modeler had to re-encode the variables in LIMDEP's rigid format, then coefficients were estimated, then transferred back to UrbanSim, then the models were run. As P7 pointed out, statistical analysis of the results is also necessary to check the residuals and independence of model variables for regression models, which meant importing and exporting to the R statistical environment as well. The results may indicate the need to modify existing variables or add new ones, and repeat the process. The consequence was that estimating a model might take months, strongly discouraging experimentation. With UrbanSim 4 and the GUI, we created integrated estimation tools that allow adding a variable and re-estimating in a matter of minutes, all within the UrbanSim framework and the GUI.

But there are often professional and organizational constraints on integrating functionalities of these external tools into the modeling environment. Each of the four phases of modeling practice (described earlier) reflect pockets of professional expertise as well as groups of mature applications in which these experts are trained and do their work. An example is GIS software. Particularly in the U.S., Environmental Sciences Research Institute (ESRI) has a virtual monopoly on such software in both university curricula and in planning agencies. Many modelers will be skilled in its use, and it will be central to their practices regarding spatial data preparation and producing publication-quality maps. This leads to a common chain for viewing a map that interview participants described: (1) exporting simulation results to a SQL database, (2) executing queries to aggregate the data to the proper geographic level, (3) running a join query on the spatial table, and (4) loading the data into the GIS program. Depending on the size of the dataset, this process could take substantial time, effort and expertise.

To address this tension, we suggest that a modeling environment should integrate functionality critical to the tight modeling loop, and provide tools to interoperate with external tools for advanced processing and analysis. This approach balances the need for experimentation for agile modeling, while allowing the necessary embedding in organizational and occupational practice for increased appropriation. The strategy is an application of the 80/20 rule – by reimplementing 20% of the functionality of an external tool, 80% of its value can be realized in a lightweight fashion, increasing agility. For example, we expect most UrbanSim users to continue using ESRI software for data preparation and producing final polished results for presentation to the agency's executive board. Therefore we have implemented tools for importing and exporting data directly to ESRI's ArcMap software. At the same time, quickly inspecting simulation results as maps is important for diagnosis and iterative model refinement. So we also provide integrated mapping software in the GUI using mapnik, which produces less polished maps, but that can be generated in a matter of seconds.

Participants in the lab study encouraged us to continue this interoperability strategy. Interestingly, while we anticipated that modelers would want to export to their favorite (or organizationally-provisioned) applications, we also learned that some external applications act as boundary objects for collaboration amongst interdisciplinary teams (e.g. P9 used SAS and P2's group used the GIS database). Without attending to these collaborative dependencies, appropriation of the modeling system could be seriously jeopardized.

5. DISCUSSION

To our knowledge, researchers have not previously explored the central role of experimentation in the appropriation of technology in domains such as urban modeling.[4] We suspect that this results from the classes of technologies that researchers have examined from the standpoint of appropriation: document management (e.g., virtual workspaces [2]), communication infrastructures (e.g., a wireless call centers [1]), and communications applications (e.g., Lotus Notes [22], calender systems [23]). These studies have generally spoken to the success of the deployment and integration of technologies that provision a closed set of

[4]While the importance of experimentation in urban modeling has not been adequately addressed in the research literature to date, the need for experimentation is recognized in other classes of simulation systems, for example, flow modeling systems such as STELLA [15], which have a much simpler underlying structure.

features and are intended to disappear into the background after they have been deployed and configured. Systemic experimentation is unlikely to be salient in these contexts.

In contrast, UrbanSim is a member of a class of systems that provide a framework for creating and applying predictive models, often incorporating models of human behavior (e.g., water demand models, climate models, financial forecasting). In the literature on appropriation, studies of spreadsheets [19] and CAD tools [13] are closest to these systems because tailoring the systems is an active part of work practice. Those studies tend focus on the "local developers" who are integral to appropriation, and do not discuss whether a form of experimentation is salient in those contexts.

As a result of the system evolution described previously, UrbanSim is moving toward better supporting agile modeling practices. Participants in the lab study were excited about speeding up the turnaround between execution and evaluation and eliminating switching between applications. For example, P1 stated that the GUI *"will make us more productive, for trying out different specifications and incorporating new variables."* P3 noted that *"This is actually plug-and-play. Different cities might have completely different UrbanSims ... This modeling approach is a pretty large leap forward. It is revolutionary."* We believe that increased agility will allow modelers to concentrate more on modeling rather than data formatting, contracting with developers, and being constrained by black-box models. In turn, we hope to engender a qualitative change in the degree of regional tailoring that modelers are able to accomplish under their rigid time and resource constraints. But to uncover further evidence about how UrbanSim may or may not be fostering the appropriation of agile modeling practices, in-depth ethnographic studies of planning agencies will be necessary.

Beyond urban modeling, there is evidence of the need to support agile modeling elsewhere. For example, Patel et al. [24] have been studying how to support software engineers in applying statistical machine learning (ML) algorithms which are typically accessible only to ML experts. Like urban modeling, applying statistical ML include a number of phases: data gathering, feature creation and extraction, algorithm selection and parametrization, and testing against training data. In their investigations, they have found similar lessons regarding the centrality of experimentation. For example, they found that it is critical to get a full pipeline early. They also found that current tool support for applying ML reflects that of urban modeling: long tool chains that create "information gaps" and inhibit iterative exploration of the pipeline through experimentation. To help address this, Patel et al. call for integrated development environments that track "experimentation history." Participants in our study echoed similar desires. For example, P4 pointed out that we need to provide better support for storing and comparing the results of multiple model estimations in order to help modelers choose a specification. Support for agile modeling and experimentation in integrated modeling environments appears to be a fruitful direction for future research.

6. CONCLUSION

As we move into an era with increasingly urgent societal challenges around sustainability, economic health, and the environment, coupled with massive data availability and computer processing power, using simulations to inform decision-making will become more and more important. In this arena, our contributions are threefold:

- Presented UrbanSim, its interface, and a sociotechnical description of its evolution over the past ten years.
- Advocated *agile modeling* as a methodology that can better support the appropriation of a modeling system, facilitating its adoption by new cities in the context of existing political and organizational processes.
- Highlighted the role of *experimentation* as a key activity for agile modeling. A flexible system does not necessarily entail agility. Agility in the modeling domain requires quick experimentation across a range of phases that span organizational, professional and political boundaries.

This work, focused on supporting the modeling phases of the larger decision making process, is part of a larger effort to bolster our collective ability to make difficult but important decisions on urban planning issues by supplying more scientifically sound data to ground public deliberation [6, 12]. In the future, we aim to provide tighter integration between the modeling and political aspects of decision making.

7. ACKNOWLEDGEMENTS

Many thanks to Jesse Ayers, Koos Kleven, Aaron Racicot, Hana Ševčíková, Liming Wang, and all the other members of the UrbanSim research group for their help with the design and implementation of the system and its interface; to the participants in our lab study; and to the UrbanSim user community for their feedback and encouragement. This work has been funded in part by NSF grants IIS-0534094 and IIS-0705898, and in part by the Maricopa Association of Governments in Arizona.

8. REFERENCES

[1] E. Balka and I. Wagner. Making things work: dimensions of configurability as appropriation work. In *Proc. of CSCW*, 2006.

[2] J. P. Bansler and E. Havn. Sensemaking in technology-use mediation: Adapting groupware technology in organizations. *Comput. Supported Coop. Work*, 15(1):55–91, 2006.

[3] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, Mass., 2000.

[4] E. Beimborn, R. Kennedy, and W. Schaefer. Inside the blackbox: Making transportation models work for livable communities. Citizens for a Better Environment and Environmental Defense Fund, 1996. Available from Environmental Defense Fund, Washington D.C.

[5] A. Borning. The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4):353–387, Oct. 1981.

[6] A. Borning, B. Friedman, J. Davis, and P. Lin. Informing public deliberation: Value sensitive design of indicators for a large-scale urban simulation. In *Proc. 9th European Conference on Computer-Supported Cooperative Work*, Paris, Sept. 2005.

[7] A. Borning, H. Ševčíková, and P. Waddell. A domain-specific language for urban simulation variables. In *Proceedings of the 9th Annual International Conference on Digital Government Research*, Montréal, Canada, May 2008.

[8] A. Borning, P. Waddell, and R. Förster. UrbanSim: Using simulation to inform public deliberation and decision-making. In H. Chen et al., editors, *Digital Government: E-Government Research, Case Studies, and Implementation*, pages 439–464. Springer Verlag, 2008.

[9] A. Cockburn. *Agile Software Development*. Agile Software Development Series. Addison-Wesley, 2002.

[10] P. Dourish. The appropriation of interactive technologies: Some lessons from placeless documents. *Computer Supported Cooperative Work*, 12(4):465–490, 2003.

[11] B. Freeman-Benson and A. Borning. YP and urban simulation: Applying an agile programming methodology in a politically tempestuous domain. In *Proceedings of the 2003 Agile Development Conference*, Salt Lake City, Utah, June 2003.

[12] B. Friedman, A. Borning, J. Davis, B. Gill, P. Kahn, T. Kriplean, and P. Lin. Laying the foundations for public participation and value advocacy: Interaction design for a large scale urban simulation. In *Proc. of Conference on Digital Government Research*, 2008.

[13] M. Gantt and B. A. Nardi. Gardeners and gurus: patterns of cooperation among CAD users. In *Proc. of CHI*, 1992.

[14] M. Garret and M. Wachs, editors. *Transportation Planning on Trial: The Clean Air Act and Travel Forecasting*. Sage Publications, Thousand Oaks, CA, 1996.

[15] iSee Systems. `http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx`.

[16] W. E. Mackay. Patterns of sharing customizable software. In *Proc. of CSCW*, 1990.

[17] A. MacLean, K. Carter, L. Lövstrand, and T. Moran. User-tailorable systems: pressing the issues with buttons. In *Proc. of CHI*, 1990.

[18] T. W. Malone, K.-Y. Lai, and C. Fry. Experiments with Oval: a radically tailorable tool for cooperative work. In *Proc. of CSCW*, 1992.

[19] B. A. Nardi and J. R. Miller. An ethnographic study of distributed problem solving in spreadsheet development. In *Proc. of CSCW*, 1990.

[20] D. Nguyen-Luong. An integrated land use-transport model for the Paris region (SIMAURIF): Ten lessons learned after four years of development. Technical report, Institut d'Aménagement et d'Urbanisme de la Région d'Ile-de-France, Paris, Apr. 2008.

[21] M. Noth, A. Borning, and P. Waddell. An extensible, modular architecture for simulating urban development, transportation, and environmental impacts. Technical Report 2000-12-01, Dept. of Computer Science, University of Washington, 2000.

[22] W. J. Orlikowski. Learning from notes: organizational issues in groupware implementation. In *Proc. of CSCW*, 1992.

[23] L. Palen. Social, individual and technological issues for groupware calendar systems. In *Proc. of CHI*, 1999.

[24] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison. Investigating statistical machine learning as a tool for software development. In *Proc. of CHI*, 2008.

[25] Z. Patterson and M. Bierlaire. Development of prototype UrbanSim models. Technical Report TRANSP-OR 080814, Transport and Mobility Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, Aug. 2008.

[26] R. H. Trigg and S. Bødker. From implementation to design: tailoring and the emergence of systematization in CSCW. In *Proc. of CSCW*, 1994.

[27] P. Waddell. The Oregon prototype metropolitan land use model. In *1998 ASCE Conference on Transportation, Land Use and Air Quality: Making the Connection*, Portland, Oregon, May 1998.

[28] P. Waddell. UrbanSim: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American Planning Association*, 68(3):297–314, 2002.

[29] P. Waddell, A. Borning, M. Noth, N. Freier, M. Becke, and G. Ulfarsson. Microsimulation of urban development and location choices: Design and implementation of UrbanSim. *Networks and Spatial Economics*, 3(1):43–67, 2003.

[30] P. Waddell, C. Peak, and P. Caballero. UrbanSim: Database development for the puget sound region. Technical report, Center for Urban Simulation and Policy Analysis, University of Washington, 2004.

[31] P. Waddell, H. Ševčíková, D. Socha, E. Miller, and K. Nagel. Opus: An open platform for urban simulation. Presented at the Computers in Urban Planning Conference, London, 2005.

[32] P. Waddell, G. Ulfarsson, J. Franklin, and J. Lobb. Incorporating land use in metropolitan transportation planning. *Transportation Research Part A: Policy and Practice*, 41:382–410, 2007.

[33] M. Wegener. Current and future land use models. In *Travel Model Improvement Program Land Use Model Conference*, Dallas, Texas, 1995.