

INTRODUCING GROWLAB: A TOOLKIT FOR LAYERED AGENT-BASED MODELING

N.B. WEIDMANN,* ETH Zurich, Switzerland

L. GIRARDIN, ETH Zurich, Switzerland

ABSTRACT

In this paper we introduce GROWLab (Geographic Research on War Laboratory) – a software toolbox to facilitate the modeling, simulation, analysis, and validation of complex social processes, with a special focus on geographic aspects. The paper aims to give a general, non-technical introduction to agent-based modeling with GROWLab. It focuses especially on the toolkit's support for model structure, i.e. the creation of complex agent configurations and hierarchies. An important feature of GROWLab is that it makes possible the integration of real-world empirical data collected with a GIS. More specifically, GROWLab can automatically create model structures from GIS datasets. This way, it is possible to run a model either on real or artificial geographies without changing the underlying data structures. We also introduce GROWLab's GeoModel, a geopolitical template model which makes different geographic and non-geographic datasets readily available to the modeler.

Keywords: Simulation toolkits, GIS, conflict process models, agent hierarchies

INTRODUCTION

As agent-based simulations are getting increasingly complex and sophisticated, simulation toolkits have become indispensable for the social science research community. Toolkits help the model designer to quickly setup, run and evaluate a model, without having to write the complete code from scratch. There are various toolkits out there (North et al. 2006; Parker 2001; Luke et al. 2004; Minar et al. 1996). However, these toolkits aim to be all-purpose products and try to target the entire broad field of social simulation. There is of course nothing wrong with creating an all-purpose toolkit. However, the fact that an all-purpose toolkit must necessarily remain general in order to make it applicable to a wide range of models limits the functionality offered to a particular subfield.

This paper reports on our efforts to create a simulation toolkit especially tailored to our requirements in the modeling of geopolitical processes. While developing a series of agent-based models in this field, we increasingly became aware of the shortcomings of general-purpose toolkits. We were often in need for agent structures more complex than grid spaces or simple networks. Also, as many of our models are increasingly relying on geographic data, we required support for different GIS data formats. Since there were quite a number of requirements common to all our models, it was natural to create a library including this functionality and in doing so avoid code redundancy.

* Corresponding author address: Nils B. Weidmann, International Conflict Research, ETH Zurich, 8092 Zurich, Switzerland; e-mail: weidmann@icr.gess.ethz.ch.

The result of these efforts is the “Geographic Research on War Laboratory” (GROWLab) library which supports modeling in the field of geopolitics and conflict research. The features of this toolkit might be useful to other disciplines as well, especially since it attempts to address challenges not specific to our area as for example the representation of agent hierarchies, and the integration of GIS data. In general however, the increased specificity of GROWLab as compared to other toolkits is likely to be useful to a narrower range of models.

Put very generally, ABM toolkits assist the researcher in three broad tasks: (1) setting up the *model structure*, (2) specifying the *simulation dynamics* and (3) collecting *output*. Model structure support is the storage and retrieval of (often different types of) agents and the representation of their relationships. Whereas model structure is about the static parts of the model and their relationships, toolkits also support simulation dynamics: What are the actions in the model, and when are they carried out? Here, the toolkit supports both scheduling within a single run of the model, but also more advanced executions such as batch runs across different parameter settings. Finally, toolkits usually provide considerable support when it comes to the collection of information from the model. Information about the current state of affairs can either be provided by graphical displays of the model space and dynamics charts, or can be collected as numerical output in files. This paper introduces GROWLab concepts and features along the three categories of *model structure*, *simulation dynamics* and *output collection*.

MODEL STRUCTURE

In many agent-based models agents live in a two-dimensional grid world. These Object2DGrids (in RePast) have two major functions: They store the agents themselves, and they define relations (such as neighborhood) between agents. Correspondingly, in GROWLab we introduce two interfaces capturing the two tasks: A *layer* is any collection of alike agents, and a *topology* is a set of relationships between them. In addition, in order to represent hierarchies of agents, we introduce the configuration interface. The following paragraphs explain layers, topologies and configurations in detail.

Layer

A *layer* is a container for a set of alike and atomic agents. Layers offer general functionality to manage the agents contained in them, but can also be used to collect aggregate data about the entire population. A layer itself does not know about the neighborhood relations of its agents – instead, this is achieved by imposing one or more *topologies* on a layer.

Topology

A topology is always defined on a layer of agents and defines a set of neighborhood relationships between them. In this sense, a topology is equivalent to a network. Based on the connectivity between agents, it can compute the neighborhood set of a given agents as well as their distance from each other.

Configuration

Whereas topologies can only exist between agents of the same kind, GROWLab offers the possibility to connect agents of different types to yield agent hierarchies. This is done using *configurations*, which typically connect agents from two layers – the parent layer and the child layer, as we call it in GROWLab. Configurations exist in different forms. The most general one is the many-to-many configuration, which allows the connection of a parent to many children, but also of a child to many parents. A more restrictive configuration is the one-to-many type, relating one parent to many children, but permits at most one parent per child. The one-to-one configuration adds the final constraint of only allowing exactly one child per parent.

A GROWLab model structure created with these building blocks is automatically kept in sync: For example, an agent removed from a layer is also removed from the topologies defined on that layer. Figure 1 illustrates the three core interfaces with a simple example of states and their provinces.

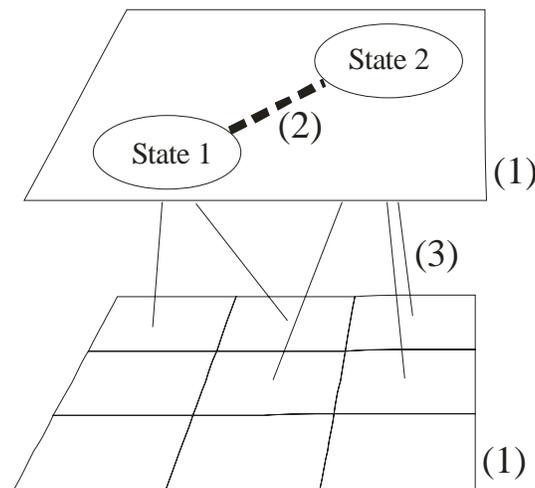


Figure 1: Illustration of the GROWLab model structure building blocks: (1) *Layers* are containers for agents. The top layer holds two states, and the bottom layer serves as a container for province agents. (2) A *topology* defined on the state layer keeps track of the relations between states. (3) A *configuration* stores the membership of provinces in states.

Spaces and Mappings

In order to represent agents in a spatial environment, we distinguish between a space which is an empty set of locations, and a mapping which takes care of the assignments of agents

to locations in this space. This flexible design allows us to put agents at more than one position (e.g. states can occupy more than one province in a grid), or even to use one space for many different mappings. For example, this is useful when representing the extent of states and ethnic groups in the same geographic space: Only one space object is required, whose locations are then linked in two mappings.

GROWLab provides different types of spaces. On the one hand, it supports abstract spaces such as grids and hexagonal spaces. On the other hand, there is support for spaces with and explicit geographic reference, for example a GIS rastered space. Here, a location not only knows its x- and y-coordinates, but also its precise coordinates in latitude/longitude. Moreover, geographic spaces can compute the geodesic distance between locations.

MODEL EXECUTION

buildModel() and step() methods

Model execution in GROWLab follows closely the procedure introduced by RePast. Each model essentially needs to implement two methods: `buildModel()` and `step()`. The former is executed when a model run is initialized. Its purpose is typically to create data structures and agents required for this run. The latter is called at every time tick and contains the simulation steps to be run repeatedly. An important difference to RePast is the implementation of the simulation engine. In GROWLab, a simulator object takes care of initializing and running the model. The advantage of this approach is that one can select a simulator according to one's needs: GROWLab offers simulators with different graphical and batch run features.

Parameters

All parameters required for a model must be implemented using the parameter classes offered by GROWLab. More precisely, a parameter is encapsulated in a special class that not only allows the storage of the parameter's value but also its name and description. All parameters are registered when the model is constructed initially such that they can be used both in graphical and non-graphical runs. Parameter classes exist for all kinds of numeric parameters, booleans, strings and enumerations.

Batch Runs

As stated above, the simulation toolkit must also provide a facility for automatic parameter sweeps, i.e. "batch runs" in RePast terminology. By automatically initializing and running the model for different values of the input parameters, the researcher can collect statistics about the behavior of the model under varying conditions. Batch runs rely on the set of parameters as described above. Batch runs can also be performed in parallel to get the results faster, both on multiprocessors machines and on distributed grid computers.

MODEL OUTPUT

Separating Visual and Batch Models

We encourage, and enforce to a certain extent, the developer to provide separate implementations of the model for graphical and non-graphical output. Essentially, apart from the basic Model interface specifying the `buildModel()` and `step()` methods, two extensions define how visual models and batch models should look like. A visual model will have to implement the `buildUI()` method where all graphical elements are set up. A batch model should provide information about which parameter sweeps are to be executed. Of course, GROWLab simulators tailored to either visual or batch runs will only be able to run the corresponding model. This structure makes sure that the behavior of the model – regardless of the desired way of output – remains the same.

Visualizing Model Structure

Model structure is displayed both as a graphical representation of model structures, like agents in a space, and a textual output with a detailed list of information about agents. The latter resembles the “probes” introduced by Swarm and still present in many other toolkits. For each of the three core concepts layer, topology and configuration introduced above, GROWLab has a set of predefined dynamic graphical visualizations for the inspection of the model. Layers can be portrayed by a list of agents and their attributes. Neighborhood relationships of a topology can be displayed graphically as a network structure, and textually as a paired list of connected partners. The structure of a configuration can be examined as a tree table. A set of two-dimensional graphical displays takes care of visualizing spatial layers and the agents contained in them.

Collecting Statistics from the Model

The way to extract statistics from a GROWLab model is done with the help of so-called “collector” classes. This mechanism is very flexible and can be used both for visual and batch models. Collectors are standardized data collection facilities storing the data in the format required for the analysis. For example, in a batch run one will typically use file-based collectors which simply output the assembled data to a file. For visual simulations, GROWLab offers collectors which prepare the data for display in a chart. Similarly, we provide a collector outputting a sequence of image files which can then be assembled to an animation of the simulation. Collectors are registered in the simulator executing the simulation. It takes care of activating the collector after each tick or at the end of a run.

Collectors get their data from variables within the model. However, as for the parameters introduced above, variables are implemented using the wrapper classes offered by GROWLab. Beyond the storage of a value these classes add meta-information about the variables such as their name and description. Additionally, variables can also be computed on the fly.

The GROWLab User Interface

During the development of GROWLab, special emphasis was put on the design of the graphical user interface. Our general approach is to have a GUI where multiple views on different aspects of the simulation are closely linked together. At the present stage, the GROWLab GUI features a set of interconnected views on the simulation, such as *spatial views* which display the simulation space, *configuration views* which allow for agent hierarchies to be displayed, and *process views* tracing the actions performed and the results produced in the model over time. Figure 2 illustrates the GROWLab user interface with the different views. The views are interconnected in such a way that selecting an agent in one view causes this agent to be displayed in another view.

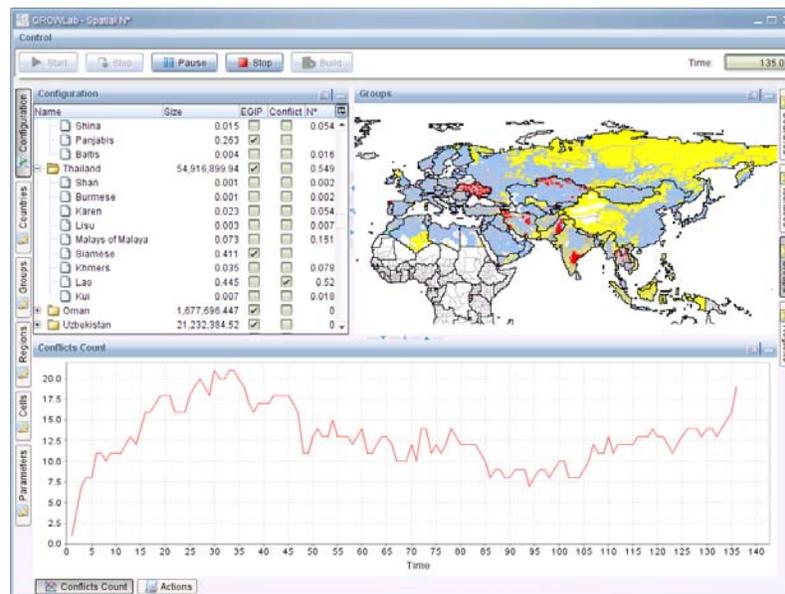


Figure 2: The different elements of the GROWLab user interface: The spatial view (top right), the configuration view (top left), and the process view (bottom). There can be more than one view per type.

USING GEOGRAPHIC DATA IN GROWLAB MODELS

Using GIS Data for Agent-based Modeling

We can distinguish two ways of how GIS data can be used in modeling applications. The first category of models takes the geographical input as a realistic landscape where the model dynamics is then run on. The crucial feature of this approach is that typically the geographic data remain constant throughout the model run. Examples include the creation of a realistic road network to run traffic simulations. The second category of simulations is more complex. Here, the geographic data is not kept constant but rather endogenous to the model. For example, in all Geosim-like models state borders vary over time. In order to be able to represent these changes, we need a data format which is able to accommodate time-variance in geographic features.

The models GROWLab is designed for typically belong to the second category. Whereas in GIS vector data the degrees of freedom for changes are unlimited, in a raster-based representation this complexity is significantly reduced. For example, a country represented as a polygon can be modified by moving the polygon's corners, or by adding or removing existing corners. Obviously, the possible alterations are infinite which makes a vector format less well applicable for simulations with an endogenous geography. On the other hand, we could represent a country as a (mostly contiguous) set of raster cells. The tradeoff we incur is the lower resolution and precision, but since the atomic spatial unit – the grid cell – is fixed, changes to the shape of the country can be represented as a re-assignment of the spatial units to other states.

GIS and GROWLab data structures

GROWLab is able to read GIS data and to create its own data structures from it. The development of a model with geographic reference typically starts with the definition of a geographic space – a raster space with geographic reference. All spatial data added to the model uses this space as a reference.

Raster data to be included in the model has to be provided at the same resolution as defined in the underlying space. It is then up to the user to tell GROWLab which kind of data structure it should create from a raster input file. For example, a raster of countries (where cell values indicate the country a cell belongs to) is best represented as a one-to-many mapping of a country object to locations in the space, in other words, an assignment of country objects to locations where each country occupies more than one location. To give another example: When representing ethnic groups and their location, we use a many-to-many mapping of groups to locations. Obviously, a group can occupy many locations, but one location can also be shared among different groups.

The data structures briefly described here enable the researcher to craft an agent-based model with geographic reference according to one's needs. However, if only some standard GIS datasets are required, one can also rely on a readily implemented template model.

GeoModel: A Geopolitical Template Model

Based on the template model GeoModel, geo-coded real-world data can be integrated in the modeling process. This template can be extended by inheriting the built-in functionality and by adding some custom behaviors and mechanisms or complement it with additional layers of data.

GeoModel's default space is a rasterized representation of the entire globe, using the WGS84 projection. The raster can be used in two different resolutions: 15 arc-minutes (~30km), and 30 arc-minutes (~60km). All the geographic data is based on this space.

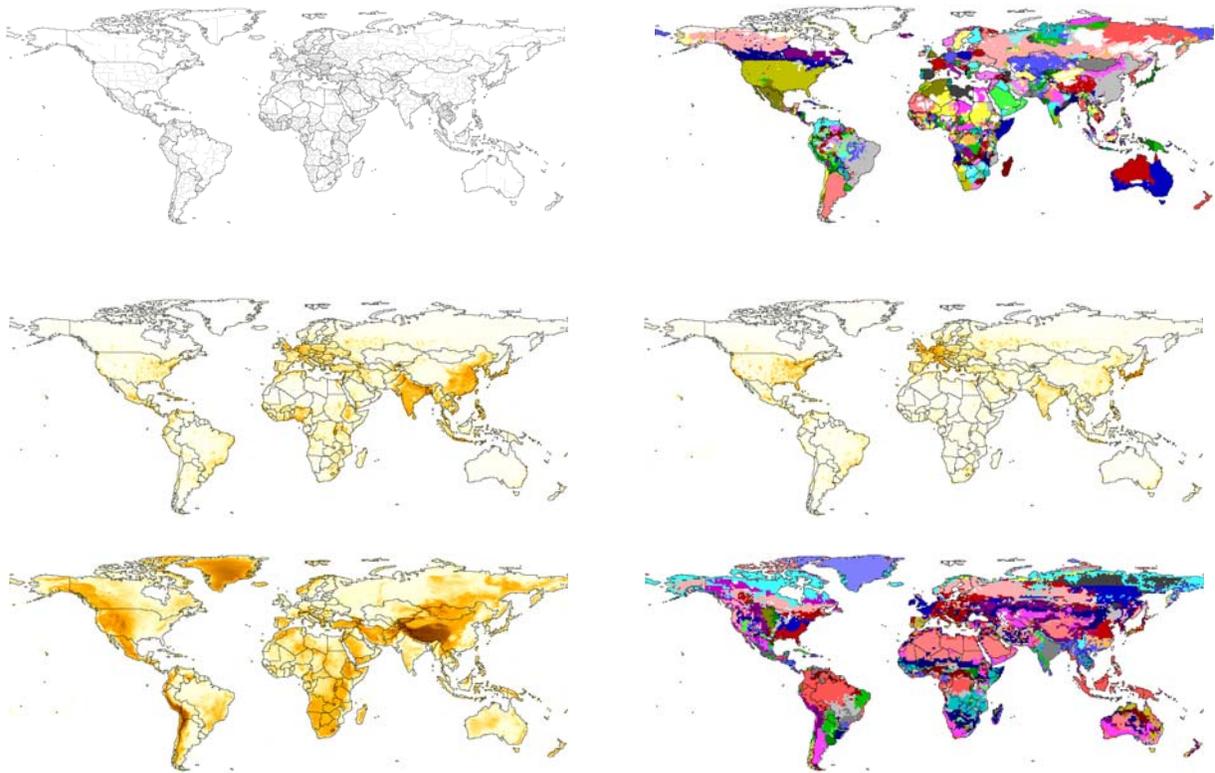


Figure 3: Geographic data contained in the GeoModel template: borders, ethnic groups, population, spatial GDP, elevation and vegetation (left to right panel).

Figure 3 shows some of the information contained in the GeoModel template including (1) country border and administrative divisions, (2) ethnic groups across countries, (3) population density, (4) spatial GDP figures, (5) elevation data, and (6) vegetation type. For each country, we provide their borders as of 1964 and 1994, and also try to reconcile their ISO, FIPS and COW codes through customized mapping. To check adjacency of countries, the Minimum Distance data from Gleditsch and Ward (2001) is also included to query for neighboring countries that are separated by water. At this point, all ethnic groups are directly based on the GREG definitions. For each ethnic group in a country, there is also information about the “ethnic group in power” (EGIP) coding by Cederman and Girardin (2007).

In addition, we provide disaggregated data for every cell in the system for population (downsampled from the Gridded Population of the World v. 3 provided by CIESIN (2005)) and elevation (downsampled from GTOPO30 (2007)), as well as local GDP estimates, compiled by the G-Econ project (at a 1-degree resolution) from Nordhaus (2006). They relieve the modeler from the tedious task of having to collect and merge complicated datasets and thus provide a prototyping environment for geographic agent-based models.

EXAMPLES IMPLEMENTED IN GROWLAB

GROWLab comes with some twenty models that serve as example that can be used as template for developing new models, as test cases to verify the inner working of the simulator, as well as stereotype models to evaluate the effectiveness of GROWLab architecture.

To give few examples, we use the iterated prisoner's dilemma model (Cohen et al. 1998) as an example for teaching purpose and to test GROWLab basic spaces (torus, grid, soup) and neighborhood functions. Schelling's segregation model (Schelling 1978) is used to showcase and test moving agents. For more elaborated agent structures, we mainly rely on the Geosim model (Cederman 1997), which features hierarchical agents and moving state borders. To showcase the GeoModel template model, we provide some statistical and exploratory models at the forefront of research.

CONCLUSION

In this paper we presented our GROWLab simulation toolkit. With GROWLab, we tried to improve the computational infrastructure for geopolitical models, especially with regard to the use of geographic data. We feel that our approach of using native GROWLab data structures to represent GIS data is promising, as it does not require the use of special GIS classes in the model. This way, one can for example switch back and forth between an artificial space and a space with geographic reference in the same model.

An issue which we have not yet taken into account is the incorporation of real time in the model. Many GIS datasets are available with explicit temporal coordinates, as for example ACLED (Raleigh and Hegre 2005). Further development of GROWLab will also focus on a better support for different agent activation regimes: In our models, synchronous updating is mostly the desired scheme, but in many cases it brings with it a lot of computational problems. A toolkit could provide implementations of agent prototypes with built-in data structures for synchronous activation.

ACKNOWLEDGEMENTS

The authors are grateful to Lucas Serpa Silva for research assistance. Nils Weidmann's work is supported by ETH (Research grant TH -4/05-3).

REFERENCES

- Cederman, Lars-Erik. 1997. *Emergent Actors in World Politics: How States and Nations Develop and Dissolve*. Princeton, NJ: Princeton University Press.
- Cederman, Lars-Erik, and Luc Girardin. 2007. "Beyond Fractionalization: Mapping Ethnicity onto Nationalist Insurgencies." *American Political Science Review* 101 (01):173-85.
- Center for International Earth Science Information Network CIESIN, Columbia University; and Centro Internacional de Agricultura Tropical CIAT. 2005. "Gridded Population of the World v3 (GPWv3)." Available at <http://sedac.ciesin.columbia.edu/gpw>.

- Cohen, Michael D, Rick L Riolo, and Robert Axelrod. 1998. "The Emergence of Social Organization in the Prisoner's Dilemma: How Context-Preservation and Other Factors Promote Cooperation." *SFI Working Papers*.
- Gleditsch, Kristian Skrede, and Michael D Ward. 2001. "Measuring Space: A Minimum-Distance Database and Applications to International Studies." *Journal of Peace Research* 38 (6).
- Luke, Sean, Claudio Cioffi-Revilla, Liviu Panait, and Keith Sullivan. 2004. MASON: A New Multi-Agent Simulation Toolkit. Paper prepared for the SwarmFest Workshop.
- Minar, N., C. Burkhart, C. Langton, and M. Askenazi. 1996. "The Swarm simulation system: a toolkit for building multi-agent simulations." Santa Fe. SFI Working paper 96-06-042.
- Nordhaus, William D. 2006. "Geography and macroeconomics: New data and new findings." *Proceedings of the National Academy of Sciences USA* 103 (10):3510-7.
- North, M. J., N. T. Collier, and J. R. Vos. 2006. "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit." *ACM Transactions on Modeling and Computer Simulation* 16 (1):1-25.
- Parker, Miles T. 2001. "What is Ascape and Why Should You Care?" *Journal of Artificial Societies and Social Simulation* 4 (1).
- Raleigh, Clionadh, and Havard Hegre. 2005. Introducing ACLED: An Armed Conflict Location and Event Dataset. Paper prepared for the Disaggregating the Study of Civil War and Transnational Violence Conference, San Diego, CA.
- Schelling, Thomas C. 1978. *Micromotives and Macrobehavior*. New York: Norton.
- US Geological Survey. 2007. "GTOPO30 Digital Elevation Model." Available at <http://edc.usgs.gov/products/elevation/gtopo30/gtopo30.html>.