SCAMP User Manual
Chapter 1: An Introduction to SCAMP

H. Van Dyke Parunak and Jonathan A. Morell

SCAMP (Social Causality with Agents using Multiple Perspectives) is a powerful multi-agent simulation package that

- subsumes previous causal modeling formalisms [5],
- is socially and psychologically realistic [9],
- uses models that can be constructed and modified by analysts rather than programmers.

The first two points are developed in the referenced publications. This document supports the third point. It describes the overall operation of SCAMP, defines the formats of the various input files, presents a modeling methodology, and explains the meaning of the output logs that the program produces.

This chapter summarizes *what* the component parts of a SCAMP model are, *how* one goes about constructing it, and *how* to install and run the SCAMP software.

Chapter 2 describes in detail the various input files and parameters needed to run a SCAMP model. This chapter is required reading for people constructing a model initially.

It is not necessary to define all of the detail specified in Chapter 2 in order to run the model, and in fact we recommend an incremental model development process that begins with default behaviors and refines them as one explores their impact on the model's behavior. The chapter includes a default mechanism that lets you develop and modify the model incrementally.

Chapter 3 documents files that are of special interest to people who are experimenting with an existing model rather than constructing a new one.

SCAMP logs its results for subsequent analysis by the user. **Chapter 4** describes the contents of these logs.

1 Core elements of a model

The MP in SCAMP stands for "Multiple Perspectives." It reflects our conviction that causality comes in multiple flavors that need to be represented in a realistic model. Our focus on the analyst as the user of SCAMP further indicates that distinct components of the model should reflect these different perspectives, so that users know where the information relevant to each perspective resides, and where to modify that facet of the model.

In this section, we outline the basic operation of SCAMP, then discuss the current four perspectives, and finally summarize different kinds of inter-agent interaction that these perspectives support.

1.1 SCAMP's Operation

The fundamental concept in SCAMP is that agents make *choices*, based on the *features* of the options available to them and their *preferences* over those features. An individual agent's preferences and the features of options are vectors in the same space. An agent forms the dot product of its preferences with the feature vector of each option, exponentializes the result so that it is non-negative, constructs a roulette wheel whose segments are proportional to these non-negative scores, and spins it to select the next option. The kinds of options available, the features they support, and the agents' preferences all depend on the active perspectives in the model.

Each agent belongs to a *group*. Each group has a baseline set of preferences, and when agents are initialized, their individual preferences are sampled around those baseline preferences. Thus agents belonging to a single group are close together in the vector space of preferences, but not identical with one another. Agents have one home group, but can affiliate with other groups if their preference vector is within a specified affiliation tolerance associated with the agent's home group. An agent's decisions are based on the weighted average of the preference vectors of its home and affiliated groups.

An important characteristic of real human decision-making is the ability to think ahead, mentally simulating alternative courses of action [2,3]. SCAMP implements this capability with the polyagent architecture [8]. This architecture models each domain entity (such as a person) as multiple agents: a single *avatar* that manages a swarm of *ghosts*. An agent's decision-making cycle is called a *generation.* In each generation, the avatar sends out several successive *shifts* of ghosts, with multiple ghosts in each shift. Ghosts use the full preference mechanism outlined in the previous paragraph to explore a specified number of steps into the future. The stochasticity of roulette selection means that each ghost may explore a different alternative future than its peers.

Ghosts communicate with each other and with the avatar by marking their presence in the options that they sample, a technique modeled on the use of pheromones by social insects. We say that ghosts deposit *digital pheromones*, recording that a ghost of a given type was at a location. The successive shifts ensure that ghosts can respond to one another's pheromones as well as to other features of the environment in exploring their options. After all the shifts in a given generation are complete, the avatar selects its next step, based only on the pheromones deposited by ghosts of its home and affiliated groups.

1.2 SCAMP's Current Four Perspectives

SCAMP currently supports four perspectives, but could be extended to accommodate more. The central perspective, and the only one that must be included in the model, is the *event* perspective, describing the causal relations among events in the world and how agents choose among them. There are three optional perspectives. The *goal* perspective recognizes that people's choices are driven not only by the immediate features of their environment, but also by higher-level objectives that they are seeking to achieve, or sometimes (in adversarial environments) to frustrate. The *geospatial* perspective describes in more detail events that require agents to move spatially, and accounts for spatial interactions among agents. The *social* perspective focuses on how people's acquaintances impact their own preferences, and supports a rich mechanism for changes in people's group loyalties.

The discussions of each perspective below include a summary of the input files and output logs relevant to that perspective. Input files are documented in detail in Chapter 2, while output logs are documented in detail in Chapter 4.

1.2.1 The Event Perspective

SCAMP's core model of the world is based on a narrative space, a directed graph of events such that any trajectory through the graph constitutes a possible history experienced by an agent. Such graphs are familiar in literary theory and game design, and have proven useful in intelligence analysis [11]. The occurrence of an event is indicated by the presence of agents on the event, and the strength of the occurrence depends on the number of agents on the event (its participation).

The edges in such a graph convey one type of causal information, that which reflects the successive choices made by an agent during its existence. These edges are of two types: "then" (indicating that one event follow another) and "thenGroup" (indicating that one event leads to several others that execute concurrently).[1], We call these edges in the basic narrative graph, "agency edges," since they reflect the choices made by an agent. (One might argue that an agent sometimes has no choice, but this is reflected in the existence of only one outgoing edge from an event.) If we restrict the CEG to agency edges, each group has its own subgraph, though some nodes may be accessible to multiple groups and thus belong to separate subgraphs.

In addition to causal influence among events modulated by agent choice, sometimes one event impacts another, though agents do not move from one to the other. For example, a drought (an act of God) may depress the economy, influencing the choices that people make about where to live. To capture these relations among events, SCAMP extends the narrative graph with an additional category of causal edge, the "influence edge." We call the extended graph, a Causal Event Graph, or CEG.

When an agent moves in the CEG, it chooses stochastically among subsequent events accessible via agency edges from its current event. Influence edges allow other events to modulate how accessible each subsequent event is. They are of four types: "Enable" means that the destination of the edge is not accessible unless the source is occurring, "Enhance" means that increased participation on the source event increases the probability that agents will choose the destination, "Prevent" means that occurrence of the source makes the destination inaccessible to incoming agency edges, and "Inhibit" means that participation in the source decreases the probability of the destination being chosen.

One important use for Influence Edges is to model the impact of acts of nature. We define an Environment group with a single agent. The Environment agency subgraph of the CEG describes reasonable stories about sequences of environmental events. Influence edges from the Environment subgraph to other events capture the impact of the environment on agent decisions, while influence edges from other events to the Environment subgraph can modulate the probability of a particular environment event taking place.

1.2.1.1 Input Files

The following files, described in more detail in Chapter 2, implement the Event Perspective.

The **Causal Event Graph (CEG)** itself is an xml file, but analysts produce it using the CMapTools concept mapping environment, a freeware system available from IHMC [1]. It represents the events (with names and unique id numbers) and both agency and influence edges among them.

Agents.csv defines the number of agents to be initialized in the system and their characteristics. If the model includes multiple groups, it defines these characteristics for each group.

The **<model>.xls** file has seven tabs, three of which are required for the basic event model.

---

[1] The original CEG language defines "then-many-in" (TMI," from multiple source events to a single destination event, indicating that agents must arrive from multiple input events for a subsequent event to occur), and "then-many-out" (TMO, from a single source event to multiple subsequent events, indicating that occurrence of one event may lead concurrently to multiple successors). These are now deprecated; in fact, TMO turns out not to be well defined. To achieve the results of TMI, use enable or enhance influence edges. The new relation "thenGroup" does what TMO was intended to do.

- The **events** tab lists all the nodes in the model, and several characteristics of each one. The columns relevant for the Event Perspective are:
    - Exogenous features of the event to which agents respond in making their choices. These represent the impact the event would have on participating agents, and thus may vary depending on the group to which the agent belongs.
    - For each group, whether the group is eligible to participate in the event or not (whether the group "has agency" for the event)
    - Timing features for the event: a TransitTime describing how long an agent participates in an event before moving on to choose its next event, and an EffectTime describing how long the agent's participation on the event persists to influence other agents and events, both by modulating the Presence feature of the event, and via influence edges. Each of these values serves as the parameter of an exponential distribution sampled by each agent when it enters an event, and the actual transit and effect times for that agent are the result of this sampling. We use the exponential because it captures the inter-event time of Poisson-distributed events.
    - Probability, an *a priori* weighting of any agent's probability of choosing this event. The actual weighting depends on the number of eligible events that an agent is considering. After computing its transition probability for all the accessible events, the agent multiplies each by the event's probability value, divided by the sum of the probability values across the accessible agents.
- The **fixedFeatures**[2] tab gives the number, names, and abbreviations for the exogenous features included in the events tab. Currently, we model three: the impact of the event on an agent's economic, physical, and psychological well-being. The abbreviations are used in composing the column labels in the events tab.
- The **groups** tab lists the different groups of agents active in the model. Here are the columns relevant for the Event perspective.
    - Each group has a unique ID number, a name, and a GroupAbbr(eviation) used in composing column headings in this and other sheets. The group also specifies the base preferences for agents belonging to each group over the features of events. These preferences cover not only the fixedFeatures, but also two sets of time-varying features. The PrefGoal<group> columns are used with the Goal perspective, and described below. The PrefPres<group> columns record how strongly agents of this group are attracted to or repelled by agents of each of the other groups.
    - Agents can affiliate with groups other than their own. Three columns describe how agents in one group relate to other groups. IndependentAgency indicates whether or not an agent in this group can participate in events without affiliating; if it is 0, the agent is Neutral, with no home group. ReceivesAffiliation indicates that this group is a legitimate target for affiliation by agents from other groups, while MakesAffiliation indicates whether agents in this group can affiliate with other groups. The Environment group neither receives nor makes affiliations. A neutral agent can make but not receive affiliations, and so forth.

---

[2] In Ground Truth, these features are indeed fixed. We now call them "Exogenous Features" to anticipate future extensions in which these features may be modified by external information, but not by the operation of the model itself.

- Three columns describe the dynamics of the model, using the polyagent model [7]. Agents plan their future actions by sending out multiple simpler agents ("ghost agents") to explore alternative futures. An agent sends out several shifts of ghosts, with multiple ghosts per shift, so that later shifts can respond to digital pheromones generated by earlier ones. A set of shifts is called a Generation. GhostsPerShift defines how many ghosts go out in each shift, ShiftsPerGen defines how many shifts must run before the main agent (the Avatar) moves, and MaxGhostSteps defines the number of steps into the future each ghost takes. The larger these numbers are, the more thorough the Avatar's exploration of its decision options are, but the slower SCAMP runs. (the Environment agent, for example, cannot). We get a good balance between exploration and execution time on modern laptops using 4 GhostsPerShift, 6 ShiftsPerGen, and 2 MaxGhostSteps for all groups.

An agent moves by selecting accessible options with probability proportional to the dot product of its preference vector (initially sampled around the base preferences of its group, but in some cases evolving over time) and the feature vector (including both fixed and variable features) of the events it is comparing. In the base Event perspective, these features are the FixedFeatures and the Presence features.

1.2.1.2 Output Logs

The following logs are generated by the Event perspective. In addition, the Phase 3 code copies over agents.csv and groups.csv from the input files, but these will be replaced by the overall configuration graphml file into which the input files are compiled.

summary.json: Dump of Repast parameters active in the run (see Chapter 3)

affs.csv: Agent group affiliations and affiliation strengths through time

agentHistory.csv: The main log for the Event perspective; shows which agents are participating in which events, and when. The message type column contains only Proceed in the Event perspective, but other perspectives add additional message types.

agentMeetings.csv: Logs the agents whom a given agent meets at each domain time

basePrefs.csv: the preferences for each agent, as sampled from the baseline values for its home group.

entropyLog.csv: Records the entropy over events of various parameters (including pheromone by group, or count by group), as a function of time

eventLog.csv: The total number of ghosts and avatars that have visited each event during the entire run

features.csv: The feature string for each event, over time, as seen by each avatar.

fullPrefs.csv: The time-varying preferences for each agent

influences.csv: A time series of the contribution of influence edges in the CEG, giving the type of influence, the source and destination events, and the participation level on the influencing event.

meetings_ceg.csv: The total number of meetings (in the CEG), and the total length of meetings, for each pair of agents.

consoleLog.txt: A log of console output generated by the run

graphLog.txt: An analysis of the graph structure of the CEG, documenting cycles, unreachable events, and events that are dead ends.

graph.xml: An XML version of the CEG as run.

1.2.2 The Goals Perspective

In the simplest model with only an Event perspective, agents choose subsequent events without any strategic objectives, guided simply by their preferences, the accessibility of other events (modulated by influence edges), and their attraction or repulsion to other individual agents based on the PrefPres features. The Goals perspective allows us to model more strategic reasoning, based on the model of a Hierarchical Goal Network (HGN) [12]. The idea is that each group may a high-level goal, which can be decomposed into subgoals. Subgoals can relate to a higher-level goal either via AND (meaning that all subgoals must be achieved to satisfy the higher-level goal) or OR (indicating that any one subgoal is sufficient). Subgoals can have further subgoals, yielding a directed acyclic graph (since a subgoal can support multiple higher-level goals), but at some point we reach subgoals that are not further subdivided.

These subgoals are connected (we say “zipped”) to events in the CEG that can either support or block them. Importantly, the group described in the HGN does not need to have agency for all the events that are zipped to its HGN, since actions performed by other groups may very well advance or hinder its own goals.

Each (sub)goal in the HGN maintains two variables: its *satisfaction* and its *urgency*. As participation changes on events zipped to HGN subgoals, satisfaction propagates up through the HGN to the root, following rules that depend on the nature of the hierarchical links (AND and OR). The root then computes its urgency as 1 – satisfaction, and propagates this back downward through its subgoals. The urgency at each leaf subgoal is then recorded as the Goal feature for that HGN’s group in the events zipped to the subgoal.

A model can include some groups with HGNs and others without HGNs.

1.2.2.1 Input Files

The following files, in addition to those required in the Event perspective, implement the Goals perspective.

Each group’s HGN(s) is a separate **<group>.xml** file, generated from a CMap file, analogous to the preparation of the CEG.xml file.

In the **model.xls** file,

- The **zips** tab records which events are zipped to which HGN subgoals;
- The optional **hgnLinks** tab allows cross-linkages across different HGNs

1.2.2.2 Output Logs

The following files are generated, in part or whole, by the Goals perspective.

features.csv: This file includes the number of supporting and blocking zips for each event.

satLog.csv: The satisfaction level at the root of each HGN, through time

1.2.3 The Geospatial Perspective

Certain events in the CEG are geospatial events, meaning that execution of these events requires movement through geospace from the agent's current location to some destination (which may vary based on the agent's group). The transit time for such events is not computed from the transit time in the events tab, but is the total time required to reach the destination, which depends not only on the length of the trajectory, but also on the characteristics of the geospatial cells through which the agent moves and the agent's home group. If an agent reaches its destination, it logs its transit time in its current event and then chooses another event. If it times out, it aborts and restarts the current event.

Agents move through geospace using the same roulette mechanism that they do in the CEG, with two differences.

1. The Goal features now contain the value of the gradient toward each group's destination, in the current spatial node.
2. All exogenous features are set to the terrain difficulty level (which is proportional to the gradient of the terrain).

1.2.3.1 Input Files

The following files, in addition to those required in the Event perspective, implement the Geospatial perspective.

We use the GIMP open-source drawing program [13] to present the geospatial laydown. Each layer corresponds to a distinct region that can be referenced in the other files, and uses distinctive colors that are used to parse out the characteristics of individual spatial tiles. This file should be exported to **model.ora**, a format that contains a series of png files for the various layers, consumed by SCAMP.

In the **model.xls** file,

- The **regions** tab identifies the characteristics of the different regions defined in the geospatial map
- The **events** tab includes each group's destinations for those events that involve geospatial movement
- The **groups** tab includes starting regions for the agents of each group, the initial locations that are assigned to them when the model starts up.

1.2.3.2 Output Logs

agentHistory.csv: This log also records the current location of each agent at the initiation of each event in which it participates, and the destination (if any) of its current event. In addition, it supports ToGeo, FromGeo, and AbortGeo message types showing the movements of the agent into and out of geospace.

agentTrack.csv: The complete movement of each agent through geospace over time. The TrackPlotter plug-in to GIMP allows visualization and animation of this log.

meetings_geo.csv: The total number of meetings (in geospace), and the total length of meetings, for each pair of networks

1.2.4 The Social Perspective

The Social perspective allows agents to modulate their preferences based on the other agents they meet, and also to change their home group dynamically. By recognizing an additional group, Gufe (in Jewish mysticism, the realm of departed spirits), we can use this mechanism to bring new agents into existence or remove existing ones.

1.2.4.1 Input Files

All additional input for the social perspective is contained in the **<models>.xls** file.

- The **groupChanges** tab defines the transition rules that cause agent birth, death, group change, and location change.
- The **namedAgents** tab defines specific agents that initially occupy designated roles in the events and transitions (e.g., government official, head of state, leaders of each group)

1.2.4.2 Output Logs

agentGroup.csv: Agent group over time, useful now that agents can change groups.

agentHistory.csv: Also includes Born, Dead, ChangeGroup, and Move message types to support group transitions.

groupChanges.csv: Logs details of the agent transitions that occur

groupInfluencers.csv: Identifies the leader and influential members of each group

groupNetwork.csv: Logs the aggregate group wellbeing and strength of links to other groups over time. Because of affiliations, a group's link to its own group, while high, is usually not 1.

implicitNetwork.csv: Logs the strength of each agent's implicit links (based on preference similarity) with other agents over time

realizedNetwork.csv: Logs the strength of each agent's realized links (based on meetings) with other agents over time

1.3 Interactions among Agents

An important feature of agents in real life is that they influence one another. Each of SCAMP's perspectives supports distinctive means of interactions.

- In the Event perspective, agents can be attracted or repelled by which groups have participated in the various events they are considering choosing. In addition, agents participating in one event can influence the accessibility of another event via Influence edges.
- In the Goals perspectives, a single HGN can be zipped to many different events, including those for which different groups have agency. Agents participating on one zipped event modulate the satisfaction and ugency levels of that HGN, and thus the Goal features of other events zipped to the same HGN.
- In Geospace, agents who would never meet on a common event may encounter one another, either attracting or repelling one another.
- In the Social perspective, the agent transition mechanism supports complex patterns of changes to one agent based on other agents. In addition, agents modulate their own preferences based on their acquaintances.

2 SCAMP Modeling Process

SCAMP is a uniquely powerful modeling platform. Before defining its distinctives, we set SCAMP within an overall context of modeling, defined by two contrasts: Prediction vs. Explanation, and Modeling vs. Experimentation.

2.1 Prediction vs. Explanation

Models have two broad purposes: prediction and explanation. Prediction has a technological purpose. Its value is to anticipate events for the purposes of taking action. The reasoning that generates prediction does not have to be true. In fact, there are good reasons to assume that good prediction may be based on a faulty understanding of causal dynamics. Further, any predictive model will inevitably have a "prediction horizon" past which the amount of error will swamp one's ability to take action. An explanation however, no matter how correct, may yield poor prediction.

Because the best predictions may be based on faulty causal assumptions, and the best insight may derive from models that do not predict very well, modelers must decide on their objectives – prediction, explanation, or both. The choices matter because they will affect the architectures of the models that are built, the ways those models are interrogated.

Both explanation and prediction are valuable in their own right, but the relationship between them should be recognized. Prediction often breaks down because no matter how a model is tweaked, it cannot escape the incorrect (or incomplete) causal dynamics on which it is based. When that happens, "explanation," or put differently the science of the phenomenon of interest, touches the technology of prediction. Technology can succeed beyond the boundaries of explanation, but when technology does fail, its (often implicit and unknown) scientific justification needs to be questioned.

2.2 Modeling vs. Experimenting

In broad strokes, the knowledge needed to work with any model can be divided into two categories, modeling and experimenting. Modeling can be thought of as developing an overall theory of some phenomenon of interest. Who (or what) are the participants? What connections do they have among each other? What are their goals? What drives their decision making? Experimenting involves running the model, observing its behavior, and observing change. A rough way to look at this is to say that modeling involves building a theory and experimenting involves testing hypotheses.

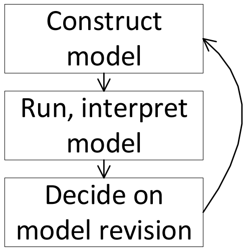


Figure 1: Process to Extract Meaning from Modeling Exercise

The most profound value of SCAMP is the opportunity it offers to probe the model relative to the results of running the model. Doing so is sure to raise questions because the model will undoubtedly produce unexpected behavior. Surprise in inevitable because the model will not have been constructed with perfect knowledge, because of compromises and conflicting assumptions among the model builders, the realities of sensitive dependence through events, the possibilities of emergent behavior, network effects among nodes and edges, and interactions between event paths and goal hierarchies. But whatever the limitations of SCAMP's predictive capacity, the results of model runs will provide much fodder for conversation and debate. It is that debate, and the results of ensuing model revisions, that will produce whatever the model builders are after – better prediction or causal explanation. Figure 1 shows this process.

2.3 SCAMP's Unique Functionality

SCAMP is an agent-based model. As such it can discover and show how small changes among the elements of a system can generate long-term change in the system's behavior. This knowledge cannot be derived from equation-based modeling methodologies. SCAMP, however, adds three unique elements to agent-based approaches.

2.3.1 SCAMP's Approach to Causality

The first is how it treats causality [6]. In general, a causal formalism C = <G, E>, where

- G is a collection of nodes bearing values and directed edges among those nodes,
- $\mathcal{G}$ is the set of all graphs with the same structure as G but differing in (some) node values, and
- E: $\mathcal{G} \rightarrow \mathcal{G}$ is an evaluation mechanism that changes the values of (some of) the nodes each time it is applied.

In most causal formalisms (e.g., path diagrams [14], Bayesian causal graphs [10], fuzzy cognitive maps [4]), E is analytic. That is, evaluation consists of solving an equation. In SCAMP, E is algorithmic, and consists of the agent behavior. This shift in focus allows us to incorporate psychological and social elements in the causal dynamics in a consistent and natural way.

2.3.2 SCAMP's Approach to Model Development

A limitation with most modeling platforms is that establishing and revising the model requires technical programming skills that most people who want to use models for their own research purposes do not have. SCAMP allows researchers without programming skills to develop and revise models. This is not to say that researchers using SCAMP need *no* special skill. SCAMP is after all, a modeling platform that requires people to specify the architectural and behavioral aspects of a model. That is what this volume is for – to provide modelers with the instructions to use SCAMP – instructions that do not require writing code.

2.3.3 SCAMP's Approach to Agents' "Motivation"

Not to be too anthropomorphic, but SCAMP allows modelers to specify how agents "think" and "feel" with respect to how they "arrive at" decisions. Caution is needed because of course; agents are nothing more than lines of computer code. Also, even in the real world, the term "agent" can refer to entities that do not think and feel, e.g. fish, classrooms, or city governments. But no matter what those lines of computer code represent, understanding can be derived from going beyond traditional agent-based methods and adopting a modeling methodology that allows researchers to specify internal decision-making dynamics. Those internal dynamics in turn, affect the overall behavior of the model.

2.3.4 Constructing a SCAMP Model

Inputs to a run come from two sources (which may sometimes be the same person): the modelers, and someone experimenting with the model. Distinguishing these two is important because it defines what we can easily vary in an experiment, without major changes to the model.

The **modelers** build

- The **model.xlsx** Excel workbook
- **CMaps for CEG and (optionally) HGNs**, with a unique node number for each event and each subgoal.
- An optional **GIMP.xcf** file for geospace

Chapter 2 describes these files in more detail.

The **experimenters** do not need to build anything, since they can directly modify **model.xlsx**. However, for experimental flexibility, several optional files are available, which Chapter 3 describes in more detail.

Figure 2 shows the proposed workflow.

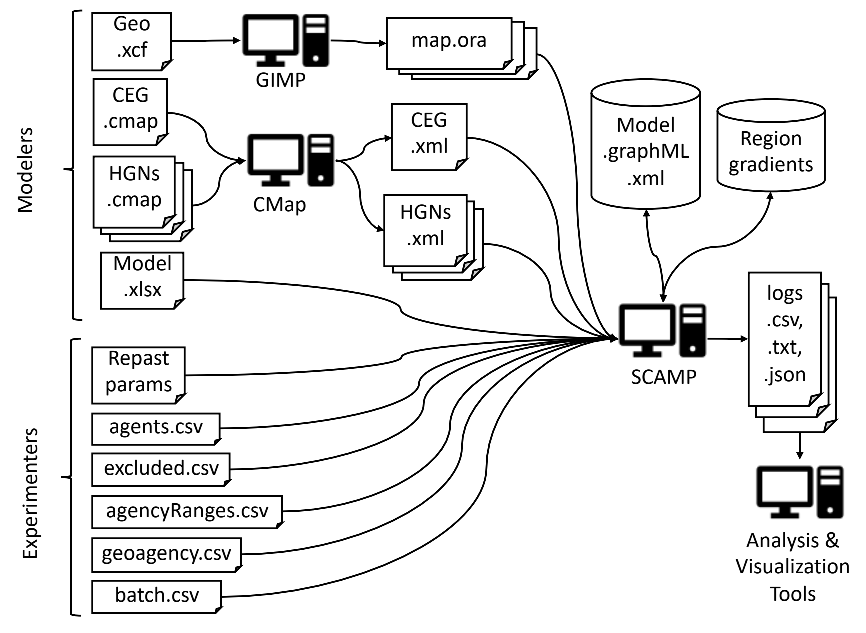


Figure 2: Proposed Workflow. Asterisks indicate optional files.

SCAMP compiles the files provided by modelers into two databases, a directory of gradients over geospacer (from **map.ora**, generated by the modelers from the GIMP .xcf model of the terrain), and **model.graphML.xml** (from the .xml versions of the various CMap graphs and from **model.xlsx**). SCAMP checks the dates of these databases, and if they are older than the source files, it recompiles them. SCAMP then reads **model.graphML.xml**, a new **agents.csv** if the experimenter has defined one, and any other experimenter files, and generates log files, which can be examined with analysis and visualization tools. There are currently three tools in this last category: Excel, the animation plug-in for GIMP, and a utility that reports event participation and visits over time.

3 Getting Started with SCAMP

This section describes how to install SCAMP on your machine, how to run the model, and how to install a new model or modify an existing one.

3.1 Installing SCAMP

First, get a copy of the SCAMP installer, SCAMPsetup.jar, and be sure you have the runtime environment for Java 1.8 or later on your machine. (Java 1.8 is sometimes called Java 8; if your machine reports either of these as being installed, you're OK.) Double-click this file, and your system should use the Java runtime to start it. If this does not work, enter a command prompt (Windows) or terminal window (Mac, Linux) and type,

```
java -jar SCAMPsetup.jar
```

On Windows, it may be necessary to say

```
java -jar SCAMPsetup.jar.zip
```

The installer will put four directories and two files in the location you specify. The directories are:

- Uninstaller: contains a Java file (.jar) that can be used to remove SCAMP from your machine

- repast.simphony: the files that support the platform, Repast, on which SCAMP runs
- groovylib: a library of files used by repast.simphony
- SCAMP: the model itself and its associated data. SCAMP comes with the complete model developed for the DARPA Ground Truth program

The two individual files are start_model.bat (for Windows machines) and start_model.command (for Mac and Linux systems).

Within SCAMP, you will spend most of your time in two subdirectories.

- SCAMP/data contains a subdirectory for each model that you want to run. When you unpack the system, you will see the SCAMP.CEG0073 model, the most mature version of the model used in Ground Truth. Chapter 2 tells you how to modify these files, or construct them from scratch. You can add other directories to SCAMP/data containing your own models.
- SCAMP/logs contains a subdirectory of logs for each run that you run. Chapter 4 describes each of these files.

You should also update the file date of two files, or the simulator will recompile the model information, which (if your model includes geospatial data) can take several hours. These files are in the SCAMP/data/<model_name> and (if it exists) SCAMP/data/<model_name>/geo directories. On Mac, open a terminal window, move to these directories, and invoke the touch command on model.graphML.xml and geo/heightmap.png. Windows doesn't have a touch command, but you can accomplish the same thing by copying a file in place. In the SCAMP\data\<model_name> directory,

```
copy /b model.graphML.xml +,,
```

in SCAMP\data\<model_name>\geo,

```
copy /b heightmap.png +,,
```

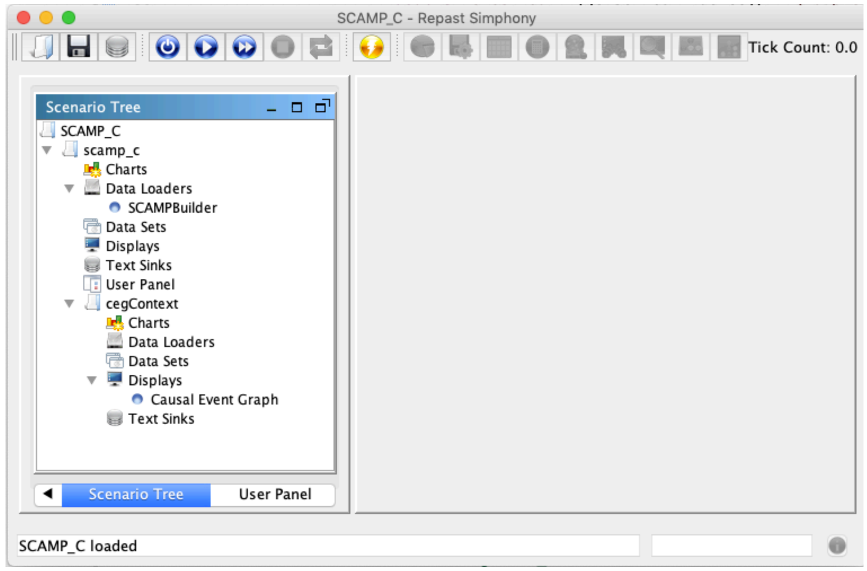
The track visualizer for GIMP is in SCAMP/misc/track_plotter.py. Put it in your plugins folder for GIMP. Chapter 4 gives further details.

3.2 Running SCAMP

To run the model, double-click on the appropriate start_model file for your machine. After a few moments, you should see two windows open: a command prompt or terminal window, and the Repast interface (Figure 3).

The command prompt or terminal window will display status information issued by the simulator as it runs, and if the model crashes, its contents will help you (or a member of the SCAMP team) figure out what went wrong.

You can run this model with the buttons across the top of the Repast window. The on-off icon (⏻) will initialize the program, displaying the various models components (the CEG and the HGNs). Figure 4 is a screenshot showing the CEG. The interface supports zooming in to see details. This particular screenshot shows the parameters panel, and is taken at the end (see "Stopped" on status line at bottom) of a 200 tick run (upper right corner).

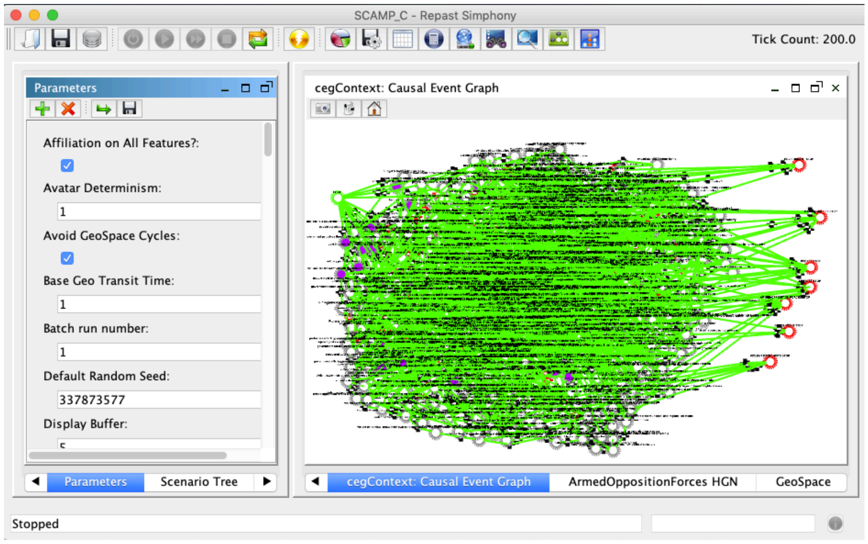


SCAMP_C loaded

Figure 3: Initial Repast Display

When the model is not running, you can navigate among the panes in this display. (If you try to navigate while the model is running, you may crash it.) The arrows under the right-hand window give you access to the other displays, and if you double-click a node, a windows will open giving you information about the current state of that object.

The panes on the left are mostly of interest to developers, but one of them, labeled “Parameters,” gives you access to the internal Repast parameters, discussed in Chapter 3. These are where you change things such as the length of the simulation run (which you can specify either in ticks with runLen, or with duration in domain days in runDur). You should modify these before pressing the on-off icon. The changes will persist until you press on-off again, or you can save them with the disk icon (💾). Once the model is initialized, you can run it (▶), single-step it (⏩) or pause (⏸) or stop (■) it. The reset button (something like this: 🔄) is supposed to reload the parameters and take you back to a point at which you can reinitialize the model (although when SCAMP in Windows, attempts to restart in this way lead to a crash, and it’s better just to exit Repast entirely and start with start_model.bat again). As the model runs, the Tick Count at the top right corner will advance, and when it reaches runLen (or when the model reaches runDur domain days, which is not monitored in the interface), it will stop. You will know that it has stopped because the status bar at the bottom of the window will say “Stopped,” and the only run-control button that is available is the reset button.


SCAMP_C - Repast Simphony
Tick Count: 200.0
Parameters
cegContext: Causal Event Graph
Parameters
Scenario Tree
cegContext: Causal Event Graph
ArmedOppositionForces HGN
GeoSpace

Affiliation on All Features?: ☑
Avatar Determinism: 1
Avoid GeoSpace Cycles: ☑
Base Geo Transit Time: 1
Batch run number: 1
Default Random Seed: 337873577
Display Buffer:

Stopped

Figure 4: Repast layout after initializing

3.3 Installing or Modifying the Model

You can modify some features of the model directly in the parameters panel of the interface, but for others you need to modify one or more of the model files. These are contained in the SCAMP directory that was formed when you unzipped the model (the one that's in the same place that your start_model file is). Inside this directory, you'll find a "data" directory, and inside that, another directory that right now is called "SCAMP.CEG0073." All the model files live inside this directory. You can specify the name of this directory with the fileRoot Repast parameter. It's set to "data/SCAMP.CEG0073" now, but you can change the directory name, or even point it to a totally different directory (e.g., "/Users/greanya/Documents/SCAMP.CEG0073"). Similarly, a "logDir" parameter configures the logging directory, the directory where your output will go. It is a path, like "fileRoot." It currently defaults to "logs", which means that the data will be placed in a "logs" directory under SCAMP, but again, it can point anywhere on your machine. Within the logs directory, each run goes in its own subdirectory with a name like "2020-05-14-13-03-43-336", which specifies the year, month, day, hour, minute, second, and millisecond at which the run began.

Runtime modification files specified in Chapter 3 (for example, for example, **agency.csv**) are all in the fileRoot directory.

Figure 5 shows the expected layout of fileRoot. Files labeled "generated" are generated by SCAMP from files that you supply. The program does not read the files in modelSources, but you will need these if you want to modify the CEG, the HGNs, or the geospatial map.

<fileRoot> is the runtime configuration directory. Modelers provide the contents of **<fileRoot>/model** and make updates there. **model.graphMLxml** is regenerated if any file in the model directory is newer than it is. The whole geo directory is wiped and regenerated if the **map.ora** file is newer than the heightmap.png file. Regenerating the **model.graphML.xml** only take a few seconds, but regenerating the geo directory can take most of a day, so map.ora shouldn't be updated on a whim.

All files directly in the fileRoot location (not in the geo or model subdirectories) are copied to the logs directory, including **model.graphML.xml**, so your logs will include a complete record of what you ran. Also, the adjacency matrix is being written there as well.

```
<fileRoot>/
    model/
        model.xlxs
        CEG.xml
        <group name>.hgn.xml (optional, needed for hgns)
        map.ora (exported from GIMP, optional, needed for geospace)
    source/
        (One CMap file for the CEG, and one for each HGN)
        (GIMP file for geospacer)
        (codebook for geospace)
    model.graphML.xml (generated)
    agency.csv (optional)
    agencyRange.csv (optional)
    batch.csv (optional)
    excluded.csv (optional)
    geoagency.csv (optional)
    geo/ (hierarchy generated for geospace)
        gradients/ (generated for destination regions)
            <Region id>_<falloff>.png (generated for each destination region)
        heightmap.png (extracted from ora; if this is missing, geospace will not function)
        <layer name>.png (layers extracted from the ora)
```

Figure 5: Layout for the model directory under SCAMP/data

Acknowledgements

The development of SCAMP was funded by the Defense Advanced Research Projects Agency (DARPA), under Cooperative Agreement HR00111820003. The content of this paper does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

[1]IHMC. IHMC CmapTools - Download. Pensacola, FL, 2013. https://cmap.ihmc.us/cmaptools/

[2]D. Kahneman, A. Tversky. The Simulation Heuristic. In D. Kahneman, P. Slovic, and A. Tversky, Editors, *Judgment under Uncertainty: Heuristics and Biases*, 201-208. Cambridge University Press, Cambridge, UK, 1982.

[3]G.A. Klein. *Sources of Power: How People Make Decisions*. Cambridge, MA, MIT Press, 1998.

[4]B. Kosko. Fuzzy cognitive maps. *International Journal of Man-Machine Studies*, 24(1):65-75, 1986.

[5]H.V.D. Parunak. The SCAMP Model of Social Causality. Wright State Research Institute, Beavercreek, OH, 2020.

[6]H.V.D. Parunak. Agent Evaluation for Social Causal Models. In *Proc. 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, (submitted), IFAAMAS, 2021.

[7]H.V.D. Parunak, S. Brueckner. Concurrent Modeling of Alternative Worlds with Polyagents. In *Proc. the Seventh International Workshop on Multi-Agent-Based Simulation (MABS06, at AAMAS06)*, 128-141, Springer, 2006.

[8]H.V.D. Parunak, S. Brueckner. Concurrent Modeling of Alternative Worlds with Polyagents. In, *Multi-Agent-Based Simulation VII*, *LNCS*, 128-141. Springer, Berlin, Germany, 2007.

[9]H.V.D. Parunak, J.A. Morell, L. Sappelsa. The SCAMP Architecture for Social Simulation. Wright State Research Institute, 2020.

[10]J. Pearl. *Causality*. 2 ed. Cambridge, UK, Cambridge University Press, 2009.

[11]L. Sappelsa, H.V.D. Parunak, S. Brueckner. The Generic Narrative Space Model as an Intelligence Analysis Tool. *American Intelligence Journal*, 31(2):69-78, 2014.

[12]V. Shivashankar. *Hierarchical Goal Networks: Formalisms and Algorithms for Planning and Acting*. Ph.D. Thesis at University of Maryland, Department of Computer Science, 2015.

[13]The GIMP Team. GIMP: GNU Image Manipulation Program. 2020. www.gimp.org.

[14]S. Wright. The Method of Path Coefficients. *Ann. Math. Statist.*, 5(3):161--215, 1934.