Style_Net_01:
A Spatial Model of Hunter-Gatherer Lithic Transport

Andrew A. White
andy.white.zpm@gmail.com

August 3, 2021

Style_Net_01 (hereafter SN01) is a spatial agent-based model designed to serve as a platform for exploring geographic patterns of tool transport and discard among seasonally mobile hunter-gatherer populations.

The SN01 model has four main levels: artifact, person, group, and system. Persons make, use, and discard artifacts. Persons travel in groups within the geographic space of the model. The movements of groups represent a seasonal pattern of aggregation and dispersal, with all groups coalescing at an aggregation site during one point of the yearly cycle. The scale of group mobility is controlled by a parameter. The creation, use, and discard of artifacts is controlled by several parameters that specify how many tools each person carries in a personal inventory, how many times each tool can be used before it is discarded, and the frequency of tool usage. A lithic source (representing a geographically-specific, recognizable source of stone for tools) can be placed anywhere in the geographic space of the model.

This model is not intended to represent all details of any particular hunter-gatherer system. The exclusion of extraneous detail is a purposeful strategy to aid in constructing a model whose structure and behavior are understandable. The representations in the model are generic and broadly applicable to a variety of hunter-gatherer systems. In the terminology of Gilbert (2008) SN01 is a “middle range” model. Middle range models “aim to describe the characteristics of a particular social phenomenon, but in a sufficiently general way that their conclusions can be applied” to many examples of the same phenomenon (Gilbert 2008:42).

SN01 was built using Repast Simphony. Repast (Recursive Porous Agent Simulation Toolkit) is a free, open-source agent-based modeling and simulation toolkit that was created at the University of Chicago in collaboration with Argonne National Laboratory. Documentation of Repast can be found at www.repast.sourceforge.net.

The code for the model is supplied in the Java files that are read by Repast Simphony: Style_Net01Builder.java; Model.java; Artifact.java; Person.java; Group.java; AggSite.java; LithicSource.java.

The first section of this guide provides a description of the representations of time and the entities (artifacts, persons, groups, aggregation sites, lithic sources) in the model and a very brief overview of the major groups of methods in the model. The second section describes model-level variables, parameters, and lists. The third section describes the “rules” and the operations of the model in detail with specific reference to sections of code.

REPRESENTATIONS IN THE MODEL

Time

Time passes in the model in the form of discrete steps, each representing one week (5200 steps representing 100 years). A week was chosen as the fundamental unit of time to allow for granular movement across the landscape.

The model maintains a “seasonal clock” that resets to 1 at the beginning of every 52-step cycle. This clock is used to control the orientation of travel toward and away from aggregation sites.

Persons

Each agent in the model represents an individual person. An initial population of persons is created at the start of a model run. In this version of the model, persons are uniform entities with no representation of age or sex. Persons make, use, and discard artifacts.

The variables and lists that are associated with each person are summarized in **Table 1.**

Table 1. Instance variables of persons.

Variable	Type	Description
id	integer	Unique identifier for each person
x, y	integers	XY coordinates of person
aggSite	AggSite	Aggregation site to which a person’s group reports
group	Group	Group that the person is in
toolList	ArrayList (Artifact)	List of current tools in the person’s possession
proximateLithicSource	LithicSource	Lithic source that is within the *personalMobilityRadius* of the person

Artifacts

Artifacts are the tools that are created, used, and discarded by persons in the model. Artifacts are created as needed based upon the size of each person’s tool inventory, which is controlled by the value of a model-level parameter (*toolInventorySize*). Each tool in each person’s inventory is potentially subject to use each step. A model-level parameter (*toolUsesPerStep*) controls the number of times each step that a person uses an artifact. The beginning use-life of all artifacts is controlled by the value of a model-level parameter (*toolUseLife*). Each use of an artifact subtracts one from its remaining use-life (tracked by the variable *remainingUseLife*). When an artifact has a *remainingUseLife* of 0, it is discarded (removed from the person’s tool inventory).

The variables associated with each artifact are summarized in **Table 2**.

Table 2. Instance variables of artifacts.

Variable	Type	Description
artifactId	integer	Unique identifier for each artifact
lithicRawMaterial	LithicSource	Artifacts that are created within a specified geographic proximity to a LithicSource are made using that source. Artifacts not within range of a LithicSource will have a null value for this variable.
remainingUseLife	integer	The number of uses remaining before the artifact is discarded
discardLocation	GridPoint	The location where an artifact is discarded
sourceDiscardDistance	double	The distance (in km) between the location of the LithicSource and the *discardLocation*
Owner	Person	The person who owns the artifact (without a mechanism of exchange, this is the same as the maker of the artifact)

Groups

Groups are co-residential collections of persons that travel together across the landscape. The number of persons in a group is controlled by the Model-level parameters *groupSize, maxGroupSize*, and *minGroupSize*. The variables and lists associated with each group are summarized in **Table 3**.

Table 3. Variables of groups.

Variable	Type	Description
groupId	integer	Unique identifier for each group
x,y	integer	XY coordinates of group
aggSite	AggSite	Aggregation site to which group reports
groupMemberList	List (Person)	List of all current members of group

Aggregation Sites

Aggregation sites are locations on the landscape to which groups travel on an annual basis. The variables and lists associated with each aggregation site are summarized in Table 4.

Table 4. Variables of aggregation sites.

Variable	Type	Description
aggId	integer	Unique identifier for each aggregation site
x,y	integer	XY coordinates of aggregation site
populationList	Person	List of all persons associated with the aggregation site
aggGroupList	Group	List of all groups that report to the aggregation site

Lithic Sources

Lithic sources are locations on the landscape that contain a distinct variety of stone for making artifacts. The variables and lists associated with each lithic source are summarized in Table 5.

Table 5. Variables of lithic sources.

Variable	Type	Description
lithicSourceId	integer	Unique identifier for each lithic source
x,y	integer	XY coordinates of aggregation site

MODEL-LEVEL PARAMETERS, VARIABLES, AND LISTS

Model-level parameters establish values for key aspects of the system. The values of these parameters do not change as a result of the dynamics of the model, but can be changed or randomized as part of an experiment. Model-level parameters are described in **Table 6**.

Table 6. Model-level parameters and variables.

Parameter	Type	Description
modelRunId	integer	A random number useful for discriminating runs produced in batch mode
sizeX, sizeY	integer	Size of the physical world; each cell represents 10 km
stepsPerYear	integer	Number of steps within a single yearly cycle (always set to 52 for Style_Net_01)
numPersonsPerAggSite	integer	Number of persons created to populate each aggregation site
numAggSites	integer	Number of aggregation sites to be created
numLithicSources	integer	Number of lithic sources to be created
groupSize	integer	Number of persons in each group

maxGroupSize	integer	Maximum number of persons permitted in a group (used to constrain group size when persons are permitted to switch groups)
minGroupSize	integer	Minimum number of persons permitted in a group(used to constrain group size when persons are permitted to switch groups)
season1Ticks	integer	The number of ticks (steps) in season 1
t1Start	integer	Step at which first data collection period (T1) begins
t1Stop	integer	Step at which first data collection period (T1) ends
t2Start	integer	Step at which second data collection period (T2) begins
t2Stop	integer	Step at which second data collection period (T2) ends
toolInventorySize	integer	The "standard" number of artifacts in each person's inventory
toolUseLife	integer	The number of uses remaining in a newly-created artifact
toolUsesPerStep	integer	The number of times per step that each person uses artifacts
exhaustionIndex	double	A calculated index reflecting the overall rate at which tools are discarded: (*toolInventorySize* * *toolUseLife*) / *toolUsesPerStep*
productionFactor	integer	Parameter used to control production level of artifacts when person is within close proximity to a lithic source; a value of 1 results in no change from normal operation; a value of 2 results in person producing enough artifacts to increase the size of the person's inventory to 2x *toolInventorySize*
replacement	boolean	Parameter used to control pre-emptive replacement of used artifacts when person is within close proximity to a lithic source; a value of "true" results in a person discarding all artifacts that have a *remainingUseLife* < *toolUseLife*; a value of "false" results in no change from normal operation
targetSampleSize	integer	Minimum number of artifacts to be included in the sample from each run
minDistanceSample	integer	The minimum number of artifacts required to calculate the percentage of artifacts from a lithic source within a specified distance range from that lithic source
moveDistance	integer	The distance (in cells) that a group moves each step
pGroupChange	double	The probability (each step) that a person will try to switch groups
personalMobilityRadius	integer	The radius (in cells) within which it is possible for a person to switch groups and/or access a lithic raw material source
pInterAggMovement	double	The probability that a person can switch to group that reports to a different aggregation site

Model-level variables are measures that change as a result of operations, behaviors, or the passage of time during a model run. Model-level variables are listed in **Table 7.**

Table 7. Model-level variables.

Variable	**Type**	**Description**
season	integer	Variable used to track whether group movement is away from aggregation site (*season* = 1) or toward aggregation site (*season* = 2)
seasonTick	integer	Variable used to track time in order to switch seasons
currentTick	integer	The current tick count of a model run (1 tick = 1 step)
timePeriod	integer	Variable used to track when the model is in one of the data collection periods (T1 or T2)

Model-level lists are used to hold the identities of the individual entities that exist in the world during a run. These lists are updated throughout a run. Model-level lists are described in **Table 8.**

Table 8. Model-level lists.

List	Class	Description
aggSiteList	AggSite	All aggregation sites in the world of the model
lithicSourceList	LithicSource	All lithic sources in the world of the model
personList	Person	All persons in the world of the model
groupList	Group	All groups in the world of the model
discardList	Artifact	All artifacts that are discarded during a data collection period

There are numerous variables in the model that are used for storing the information needed to produce summary data outputs at the end of the run. **Table 9** summarizes the variables that are calculated as data outputs. The values of many of the parameters that set the conditions of the model (see **Table 6**) are also reported as data outputs.

Table 9. Data output variables.

Variable	Description
numInterAggMoves	Number of times persons moved between groups reporting to different aggregation sites during the course of a data recording period
numInterGroupMoves	Number of times persons moved between groups during the course of a data recording period
meanTransportDistance	The mean linear distance (in km) from the location of the lithic source to the locations of discard for all artifacts made from a lithic source and discarded during a data recording period
maxTransportDistance	The maximum linear distance (in km) from the location of the lithic source to the location of discard for all artifacts made from a lithic source and discarded during a data recording period
maxRangeKm	Maximum distance (in km) that any group travelled from the aggregation site during a data recording period
percentSourced_0_20	The percentage of artifacts between 0 and 20 km from a lithic source that were made from that lithic source
percentSourced_[x1]_[x2]	The percentage of artifacts between [x1] and [x2] km from a lithic source that were made from that lithic source

MODEL STRUCTURE AND METHODS

This section describes the methods and inter-relationships between methods that affect the behaviors and interactions of the entities in the model. Methods in the model are initiated and performed by the model, individual persons, or groups. The major methods of the model are representations of rules governing personal mobility, group mobility, and the creation, use, and discard of artifacts.

The parenthetical numeric designations inserted in the narrative descriptions below (and in the comments in the model code) are an effort to help the reader locate relevant sections of code. The designation “Model 12”, for example, means that the method or code being discussed is labeled “12” in the code within the Model.java file. Numbered sections of code are placed in numerical order within each model file. The descriptions provided here are also given in the order they are numbered, although operations in the model are not called in strictly numerical order. Model-level methods are described first, followed by Group-level methods and Person-level methods.

Model-Level Step Method

The *step* method in Model.java (**Model 6**) is called at the beginning of every step to initiate a sequence of operations. The order of the operations called by the *step* method is shown in **Table 10.**

Table 10. Operations in the *step* method in Model.java.

Operation	No.	Short Description
checkSeason	7.	Check the current value of *seasonTick* to see if it's time to switch seasons
moveGroups	8.	For each group, check the value of *season* and call method to either move away from aggregation site (*season* == 1) or toward the aggregation site (*season* == 2)
artifactMethods	9.	For each person, check proximity to a lithic source and discard used tools if appropriate (if *replacement* == true); call sequential methods to create new tools and use tools
movePersons	10.	For each person, call methods to move to a different group
checkTickCount	11.	Check the tick count to see if run is in a data collection period (i.e., T1) or at the end of a data collection period; if at the end of a data collection period, call the method to report summary data

Model-Level Methods

- **Model 1: Create Aggregation Site(s)**

  This method is called from the Builder (**Builder 1**) at the start of a run to create aggregation sites. The code in SN01 is written to create a single aggregation site. The XY coordinates of the aggregation site are set in this method and the aggregation site is added to the *aggSiteList*.

- **Model 2: Create Lithic Source(s)**

  This method is called from the Builder (**Builder 2**) at the start of a run to create lithic sources. The code in SN01 is written to create a single lithic source. The XY coordinates of the lithic source are set in this method and the lithic source is added to the *lithicSourceList*.

- **Model 3: Create People**

  This method is called from the Builder (**Builder 3**) at the start of a run to create persons. The number of persons is determined by multiplying the value of the model-level parameter *numPersonsPerAggSite* by the number of aggregation sites that were created (**Model 1**). Each person is added to the *personList*.

- **Model 4: Assign People to Aggregation Sites**

  This method is called from the Builder (**Builder 3**) at the start of a run to assign each person created (**Model 3**) to an aggregation site (**AggSite 1**). Because the code in SN01 creates a single aggregation site, all persons are assigned to it.

- **Model 5: Assign People to Groups**

This method is called from the Builder (**Builder 3**) at the start of a run to assign each person created (**Model 3**) to a group. The method divides the population of each aggregation site into a number of groups. The number of groups to be created is determined by dividing the population of the aggregation site by the value of the model-level parameter *groupSize*. The groups are created and added to the aggregation site's *aggGroupList* (**AggSite 3**). After the groups are created, the aggregation site is told to add the persons in its population to those groups (**AggSite 4**).

- **Model 6: Step Method**

  See above.

- **Model 7: Check Season**

  This method checks the current value of *seasonTick* and compares it to the values of the parameters *season1Ticks* and *stepsPerYear*. If *seasonTick* equals *season1Ticks*, season 1 is over and it is time to switch the value of *season* from 1 to 2. If *seasonTick* equals *stepsPerYear*, season 2 is over and it is time to reset season back to 1. The value of *seasonTick* is incremented at the end of the step.

- **Model 8: Move Groups**

  This method goes through the *groupList* and initiates group movement for each group. If the run is in season 1, each group will move away from the aggregation site (**Group 3**). If the run is in season 2, each group will move toward the aggregation site (**Group 4**)

- **Model 9: Artifact Methods**

  This method goes through the *personList* to call methods related to artifacts. Each person on the list first calls the method to check for proximity to a "named" lithic source (**Person 3**). If *replacement* == true and the person is within range of a "named" lithic source (i.e., within the *personalMobilityRadius*), the method is called to discard all tools that have been used from the person's inventory (**Person 4**). Next, the method to create new tools is called (**Person 5**), followed by the method to use tools (**Person 6**).

- **Model 10: Move Persons**

  The model goes through the *personList* and determines if each person will attempt to move to a different group. This is a probability-based method that compares a random number between 1 and 0 to the value of *pGroupChange* (thus when *pGroupChange* == 0 there is no inter-group mobility). If the person attempts to move and the size of the person's current group is greater than *minGroupSize* + 1, the changeGroups method is called (**Person 8**).

- **Model 11: Check Tick Count**

  This method checks the tick count to see if run is in a data collection period (i.e., T1) or at the end of a data collection period. If the run is at the end of a data collection period, the method is called to record and report the summary data (**Model 12**) prior to ending the run.

- **Model 12: Record Summary Data**

This method records the set values of model-level parameters used in a run and calculates and records lithic transport data. First, mean and maximum transport distance of the artifacts on the *discardList* are calculated by calling the *calculateMeanTransportDistance* (**Model 14**) and *calculateMaxTransportDistance* (**Model 15**) methods. Then, artifacts on the *discardList* are sorted into 20 km bins according to their distance from the "named" lithic source (**Model 16**). Finally, the percentages of "sourced" artifacts (artifacts made from the "named" lithic source) are calculated from the assemblage within each 20 km bin (**Model 17**).

- **Model 13: Append Summary Data**

  This method appends the summary data from each run onto a .txt file, producing a single file of the data from a batch of runs.

- **Model 14: Calculate Mean Transport Distance**

  This method calculates the mean distance (in km) that artifacts made from the named lithic source were discarded away from that source. The model goes through the *discardList* and totals the source-to-discard distance for every artifact that is made from the named lithic source. It calculates the average source-to-discard distance by dividing the total distance by the number of artifacts included in the total.

- **Model 15: Calculate Maximum Transport Distance**

  This method calculates the maximum distance (in km) that any artifact made from the named source was discarded away from that source. The model goes through the *discardList* and retrieves the source-to-discard distance for each artifact that is made from the named lithic source. If that distance is greater than the previously recorded maximum distance, it is recorded as the new maximum transport distance.

- **Model 16: Sort Discarded Artifacts**

  This method begins by checking the size of the *discardList* to ensure that it contains a sufficient sample of artifacts. If the number artifacts on the *discardList* is less than *targetSampleSize*, the run is ended.

  If there are a sufficient number of artifacts on the *discardList*, the model sorts them into 20-km bins based on their distance from the named lithic source. If the artifact is 79 km from the named lithic source, for example, it will be placed into the 60-80 km bin. Each bin is an ArrayList to store Artifacts.

- **Model 17: Calculate Percentages of Sourced Artifacts**

  This method calculates the percentages of sourced artifacts in each distance bin (as determined in Model 10). For each distance bin, the model determines how many of the artifacts in the bin were made from the named lithic source. That number is divided by the size of the ArrayList for the bin in order to calculate the percentage. If the size of the list is too small (i.e., less than *minDistanceSample*), the percentage is not calculated.

- **Model 18: Clear Distance Lists**

  These method clears all the ArrayLists used for sorting artifacts into bins by distance (in preparation for the next run).

- **Model 19: Null Discarded Artifacts**

  This method goes through the *discardList* and nulls each artifact on it.

- **Model 20: End Run**

  This method ends a run and nulls the context.

AggSite-Level Methods

- **AggSite 1: Add Person to List**

  This method (called from **Model 4**) adds a person to the *populationList* stored by the aggregation site.

- **AggSite 2: Remove Person from List**

  This method removes a person to the *populationList* stored by the aggregation site. It is called when a person moves to a group that reports to a different aggregation site (**Person 8**).

- **AggSite 3: Add Group to List**

  This method (called from **Model 5**) adds a group to the *aggGroupList* stored by the aggregation site.

- **AggSite 4: Add People to Groups**

  This methods (called from Model 5) adds a group to the *aggGroupList* stored by the aggregation site.

Group-Level Methods

- **Group 1: Add Person to Group**

  This method is called to add a person to a group. The method takes the person sent and adds the person to the group's *groupMemberList* and calls person-level methods to set the person's XY coordinates to match those of the group. This method is called from AggSite (**AggSite 4**) during the initial set-up of a run as well as when a person changes groups (**Person 8**).

- **Group 2: Remove Person from Group**

  This methods is called to remove a person from a group. The method removes the sent person from the group's *groupMemberList*. This method is called when a person changes groups (**Person 8**).

- **Group 3: Move Away from Aggregation Site**

  This method is called from the model (**Model 8**) during the step method to move a group away from the aggregation site during season 1. The method first calculates the current

distance between the group and its aggregation site. It then collects a list of all the cells that are of the distance from the group that the group is required to move (determined by the value of the model-level parameter *moveDistance*). From that list, the method creates a sub-list of only those cells that are (1) farther from the aggregation site than the group's current location and (2) not occupied by another group. That sub-list is randomly shuffled, and the group is moved to the first cell on the list. A person-level method (**Person 1**) is called to move each person on the group's *groupMemberList* to the group's new location.

If the model is in a data collection period, it compares the distance of the group from the aggregation site (in km) to the value of *maxRangeKm* stored by the model. If the group is farther from the aggregation site than any previous group during the run, the distance is stored as the new value of *maxRangeKm*.

- **Group 4: Move Toward Aggregation Site**

This method is called from the model (**Model 8**) during the step method to move a group toward the aggregation site during season 2. The method first checks to see if the group is currently at the aggregation site. If not, it calculates the current distance between the group and its aggregation site. If the distance is less than the distance a group is required to move (determined by the value of the model-level parameter *moveDistance*), the group is moved directly to the aggregation site (**Group 5**).

If the group is not at or within *moveDistance* to the aggregation site, the method collects a list of all the cells that are moveDistance away from the group's current location. From that list, the method creates a sub-list of only those cells that are (1) closer to the aggregation site than the group's current location and (2) not occupied by another group. That sub-list is randomly shuffled, and the group is moved to the first cell on the list. A person-level method (**Person 1**) is called to move each person on the group's *groupMemberList* to the group's new location.

Group 5: Move To Aggregation Site

This method is called (from **Group 4**) to move a group directly to its aggregation site. It is called during season 2 when a group is within *moveDistance* from its aggregation site. A person-level method (**Person 1**) is called to move each person on the group's *groupMemberList* to the aggregation site.

Person-Level Methods

- **Person 1: Move With Group**

This method is called for each person in a group when that group moves (**Group 3, 4, and 5**). The person is moved to the new location of its group and stores the new X and Y coordinates of its group.

- **Person 2: Add Tool to the Tool List**

This method is called after a new tool is created. It simply adds the new tool to a person's *toolList*.

- **Person 3: Check Lithic Source Proximity**

This method is called from the artifact methods (**Model 3**) in the step method for each person at each step to search the cells within the *personalMobilityRadius* of a person to

determine if a lithic source is present within that radius. If a source is present, it is stored as a person's *proximateLithicSource*.

- **Person 4: Discard Used Tools**

  This method is called to discard all used tools (artifacts with a *remainingUseLife* of less than the value of the model-level parameter *toolUseLife)*. It is only called when the value of *replacement* == true and the person is within proximity to a lithic source. It is called from the artifact methods (**Model 3**) in the model's step method.

- **Person 5: Create Tools**

  This method is called from the artifact methods (**Model 3**) in the model's step method to create new tools for a person's tool inventory. The method first compares the number of tools currently in the person's inventory (the size of the person's *toolList*) to the value of the model-level parameter *inventorySize* in order to determine how many (if any) new tools the person will create. If the person is in proximity to a lithic source (as determined by **Person 3**), the model re-calculates the number of new tools to create by multiplying the value of *inventorySize* by the value of the model-level parameter *productionFactor*. If *productionFactor* equals 1, then the number of new tools to be created will not change. If *productionFactor* is greater than 1, then the person will over-produce tools while in proximity to the lithic source.

  If the creation of new tools is called for, they are made sequentially by creating new artifacts using the constructor in Artifact. Each new tool is added the person's *toolList*.

- **Person 6: Use Tools**

  This method is called from the artifact methods (**Model 3**) in the model's step method to use tools in each person's inventory. The method first obtains the number of tool uses per step from the model (the value of the model-level parameter *toolUsesPerStep*). It then enters a loop to perform the required number of uses, randomly selecting an artifact from the person's tool inventory for each use. When an artifact is selected, the use method (**Artifact 1**) is called for that artifact.

  At each iteration of the loop, the model compares the number of uses still to be performed with the total number of uses remaining in the person's tool inventory. If the number of uses to be performed exceeds the capacity of the existing tool kit, the create tools method (**Person 5)** is called to immediately replenish the tool inventory.

- **Person 7: Remove Artifact from List**

  This method is called from discard method (**Artifact 2**) to remove an artifact from the person's tool inventory. If the run is in a data collection period, the artifact may be saved to the model's *discardList* for analysis. That determination is made subject to a random number to retain a percentage sample of discarded artifacts. If an artifact is not saved, it is simply removed from the person's *toolList* and replaced with a null value.

- **Person 8: Change Groups**

  This method is called from the move persons method in Model (**Model 4**) when a person decides to attempt to change groups. The model first creates a list of all the groups within the *personalMobilityRadius* of the person's current group that are below the

maximum group size (the value of *maxGroupSize*). From these eligible groups (if there are any) the model randomly selects a group to which the person will move. The person is removed from the person's current group (**Group 2**) and placed in the new group (**Group 1**). If the new group reports to a different aggregation site than the person's current group, movement to the new group is conditioned by the value of the model-level parameter *pInterAggMovement*, which specifies the probability that a person can join a group that reports to a different aggregation site.

Artifact-Level Methods

- **Artifact 1: Use Artifact**

  This method is called by a person (**Person 6**) to subtract one use from an artifact's *remainingUseLife*. If the subtraction results in the artifact having a *remainingUseLife* of 0, the artifact is discarded (**Artifact 2**).

- **Artifact 2: Discard Artifact**

  This method is called to discard an artifact. It is called when an artifact is used to the point of having a *remainingUseLife* of 0 (**Artifact 1**) and when a person discards all used tools (**Person 4**) when *replacement* == true.

  When an artifact is discarded, this method calculates the distance between the location of discard and the location of the lithic source. It stores the source-to-discard distance of the artifact (in km). Finally, it sends the artifact to the person-level method to remove the artifact from the person's *toolList* (**Person 7**).

MODEL STARTUP

The StyleNet01Builder.java file is called when a model run is initiated. This method creates a new instance of model, sets the space and grid, creates an aggregation site (**Builder 1**) and a lithic source (**Builder 2**), and creates the people and groups to populate the world (**Builder 3**).

REFERENCES

Gilbert, Nigel
2008 Agent-Based Models. *Quantitative Applications in the Social Sciences* 153. Sage Publications, Thousand Oaks, California.