SPATE-HPC

User Manual

December 20, 2009

Kristoffer Paro

Department of Information Technologies
Åbo Akademi University

Contents

Table of Contents **i**

List of Figures **iii**

1 Introduction **1**

1.1 Programs, input and output files 1

2 Compiling and running the programs **3**

2.1 Compiling SPATE-HPC from source code 3
2.2 Running SPATE-HPC 4
2.2.1 Running on multiple processors 5
2.3 Compiling SPATE-GUI from source code 6
2.4 Running SPATE-GUI 6

3 The simulator **8**

3.1 Tree 8
3.2 Spatial inter-tree dependence 9
3.3 Compartments 9
3.4 Simulation model 10
3.4.1 Variables 11
3.4.2 Environmental effects 13
3.5 Simulation phases 13
3.5.1 Initial stand 13
3.5.2 Growth 14
3.5.3 Mortality 14
3.5.4 Reproduction 14
3.6 Mark transformation 15
3.7 Species 16
3.8 Forest types 17
3.9 Management methods 17
3.9.1 Dimension and energy wood cutting 18
3.9.2 Weibull thinning 18

3.9.3 When is thinning carried out . . . 20
3.9.4 Clear cut . . . 20
3.10 Assortments . . . 21
3.11 Output files . . . 21

4 The input file editor **23**
4.1 General page . . . 23
4.2 Regeneration page . . . 25
4.3 Reproduction, Growth and Mortality pages . . . 25
4.4 Editing a model . . . 26
4.5 Initial forest page . . . 29
4.6 Forest type page . . . 29
4.7 Species page . . . 32
4.8 Management . . . 32
4.9 Assortments page . . . 36
4.10 Compartments page . . . 37
4.11 Notes page . . . 38

5 File formats **39**
5.1 Parameter file . . . 39
5.1.1 Simulation . . . 40
5.1.2 General parameters . . . 40
5.1.3 Model terms . . . 46
5.1.4 Point processes . . . 47
5.1.5 Species parameters . . . 47
5.1.6 Forest type parameters . . . 48
5.1.7 Management parameters . . . 51
5.1.8 Assortment parameters . . . 53
5.1.9 Compartment parameters . . . 54
5.1.10 Inspection parameters . . . 54
5.2 Polygon file . . . 56
5.2.1 Cut-out regions . . . 56
5.3 Empirical data . . . 57
5.4 Empirical cumulative distribution . . . 57
5.5 Tree data file . . . 58
5.6 Compartment statistics file . . . 58
5.7 Statistics table file . . . 59

Index **61**

List of Figures

1.1 The relationships between SPATE-GUI, SPATE-HPC, the input and the output files. 2

3.1 Diagram showing the workflow of a simulation. 9
3.2 A sample forest area with 38 compartments consisting of 57 polygons. 10
3.3 The simulated tree distribution is represented by the histogram and the estimated distribution by the solid curve. The dashed curve is the shifted curve. 18

4.1 The general simulation settings 24
4.2 The 'Regeneration' page showing the model for calculating the expected number of new trees per hectare in the reproduction phase. 26
4.3 Page showing the reproduction models. 27
4.4 The model editor showing the model $Y = \beta_0 + \beta_1 \cdot Diameter + \beta_2 \cdot Diameter^2 \cdot Local\ density + \beta_3 \cdot Epsilon$. The coefficients 'Beta 0' through 'Beta 3' are shown in bold font, which indicates that they have been assigned secondary-level models. 28
4.5 The smoothing function editor window showing the secondary-level model $\beta_1 = 0.4 + 0.7 \cdot Species$ 28
4.6 The 'Initial forest' page showing example settings for generating an initial stand. 30
4.7 The 'Forest types' page showing the forest type "spruce_dominated", which contains 80% spruce and has diameter transformations defined for both the initial stand and the reproduction phase. 31
4.8 The 'Species' page. Here, the species "Pine" is selected and its settings are shown in the panel to the right. 33
4.9 Management templates are defined on the 'Management' page. The selected template "LowThinning" is set to carry out low thinning removing 50m^3/ha every 13th year if the compartment volume exceeds 200m^3/ha. 34
4.10 Assortments are defined on the 'Assortments' page. Here, the assortment "sawlog" is selected and its size criteria for different species are shown in the panel to the right. 36
4.11 The compartments page showing the forest area. A compartment consisting of three polygons is selected. 37

5.1 XML elements are illustrated with these types of boxes. 39
5.2 The root element, “simulation”, and its direct children. 41
5.3 The *generalParameters* element contains assorted simulation parameters. 42
5.4 Parameters governing growth, mortality, reproduction and regeneration. 44
5.5 *initialCompartment* contains parameters for generating or loading trees for the initial stand. 45
5.6 A *modelTerm* element corresponds to a term in a model function. 46
5.7 The elements *homogeneousPoisson*, *inhomogeneousPoisson*, *homogeneousClustered* and *inhomogeneousClustered*. 47
5.8 The elements *homogeneousRegular* and *inhomogeneousRegular*. 48
5.9 A *species* element contains parameters specific to a certain species. 49
5.10 A *forestType* element defines a particular forest type. 50
5.11 A *managementTemplate* element defines a set of management rules. 52
5.12 An *assortment* element defines a specific assortment. 53
5.13 A *compartment* element contains compartment-specific parameters. 54
5.14 The *inspection* element enables more detailed inspection of the simulation process. 55

Chapter 1

Introduction

SPATE-HPC (Spatio-Temporal Stand Simulator with High Performance Computing) is a tree population dynamics simulator that simulates each tree individually. Forest growth and silvicultural treatments can be simulated in order to obtain statistics on wood raw material volumes. SPATE-HPC is a highly scalable, parallelized computer program that can be run both on small home computers as well as on large supercomputers.

SPATE-GUI is an input file editor that can be used to produce simulation parameter files for SPATE-HPC. SPATE-GUI is a stand-alone program that can be run on home computers.

This user manual describes how the simulator works, how to use it in order to run simulations, how to create input files and how to interpret output files. In chapter 2 it is explained how to compile and run SPATE-HPC and SPATE-GUI. The growth models, simulation mechanisms and implemented features of SPATE-HPC are described in chapter 3. How to use SPATE-GUI to create and edit input parameter files is shown in chapter 4. And finally, the different file formats of input and output files are detailed in chapter 5.

1.1 Programs, input and output files

The simulator program SPATE-HPC can be run either on the user's own workstation, another computer or a supercomputer. The simulator program is a command-line application, i.e., it has no graphical user interface. It is run by providing it a set of input files and it produces a set of output files after the simulation has completed. The main input parameter file is in the XML (extensible markup language) file format and serves as the main input file. Other input files, such as the polygon file that define the forest area and empirical data files for diameter and height distributions, are referenced by the XML input parameter file. When a simulation has finished statistics files on the stand's development and the final stand are compiled. During the simulation the program also writes log files that contain information about the different simulation steps performed. Additionally, tree data files containing the characteristics of all individual trees can be produced.

To facilitate creating and editing the XML input parameter file, the input file editor SPATE-GUI can be used. SPATE-GUI is a separate, graphical user interface application that can be run on the user's workstation. If the user also runs SPATE-HPC on the same computer as SPATE-GUI, the simulator can use the input files directly from the directory where they were created, otherwise, the input files must be copied to the computer that runs the simulation. Figure 1.1 shows the relationship between the programs and how they read and write files.

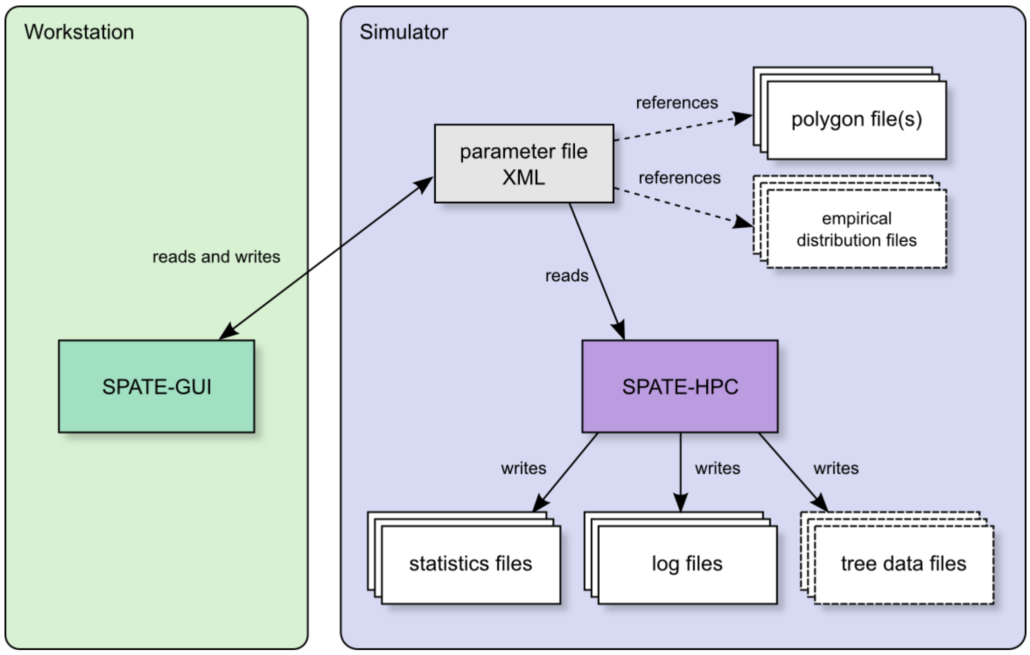


Figure 1.1: The relationships between SPATE-GUI, SPATE-HPC, the input and the output files.

Chapter 2

Compiling and running the programs

2.1 Compiling SPATE-HPC from source code

SPATE-HPC can be compiled from source code with the GNU Compiler Collection (GCC) C++ compiler or the Portland Group (PGI) C++ compiler using the provided makefiles. It can be compiled on Unix-based systems (e.g. Linux) or on Microsoft Windows using Cygwin. The following is a list of third-party libraries that are dependencies of SPATE-HPC. Some of them are required whereas others are optional and can be enabled or disabled via conditional-compilation flags.

- The AMD Core Math Library (ACML) is mandatory and should be at least version 3.6.0. ACML is available from http://developer.amd.com.
- Xerces-C++ is mandatory should be at least version 2.8. Xerces-C++ is available from http://xerces.apache.org/xerces-c/.
- The Scalable Parallel Pseudo Random Number Generators Library (SPRNG) is mandatory and should be at least version 4.0 for C++. SPRNG is available from http://sprng.cs.fsu.edu/.
- The GNU Scientific Library (GSL) is mandatory and should be at least version 1.11. GSL is available from http://www.gnu.org/software/gsl/.
- Message Passing Interface (MPI) is used to run SPATE-HPC in parallel. SPATE-HPC can be compiled without MPI support, but will in that case only be able to run on one processor and not in parallel. MPICH is available from http://www.mcs.anl.gov/research/projects/mpi/.
- CppUnit is used for unit tests and is not normally needed. If unit tests are explicitly enabled at compile time, however, the CppUnit library must be present. CppUnit is available from http://sourceforge.net/projects/cppunit/.

SPATE-HPC is compiled with the *Make* program. In order to run Make, copy the file named “makefile.sample” and name it “makefile”. The compilation process is configured by editing the makefile. It is most likely necessary to edit the makefile to reflect the system set up in order for SPATE-HPC to compile successfully. The following is a list of makefile variables that can be changed:

CC The name of the compiler executable. It should be set to “mpicxx” if the simulator is to be compiled with MPI support on MPICH or set to “CC” if it is to be compiled with PGI. If the simulator is to be compiled without MPI support with GCC, it should be set to “g++”.

USE_MPI A flag controlling whether to enable MPI support. It should be set to either “true” or “false”.

xx_DIR Specifies the directory where library “xx” is installed. Replace “xx” by “ACML”, “XERCES”, “SPRNG”, “GSL” or “CPPUNIT”.

xx_LIB Specifies the names of the library files of library “xx” to be linked into the resulting program file. Replace “xx” by “ACML”, “XERCES”, “SPRNG”, “GSL” or “CPPUNIT”.

INCLUDE_TESTS A flag that controls whether to build unit tests. It should be set to either “true” or “false”. The default value is “false”, since unit tests are not needed in order to use the simulator.

EXTRA_LIBS This variable can be used to include additional libraries.

CFLAGS This variable can be used to provide additional flags to the C++ compiler.

In addition, there are three C preprocessor flags that can be enabled or disabled in the makefile with the corresponding “CFLAGS” line: “CALCULATION_INSPECTION”, “INCLUDE_-ENVIRONMENTAL_EFFECTS” and “COMPARISONSEEDING”. The flags “CALCULATION_-INSPECTION” and “INCLUDE_ENVIRONMENTAL_EFFECTS” are both on by default and enable the inspection of model calculations and the generation of correlated random numbers for environmental effects, respectively. The calculation inspection feature might cause a slight performance downgrade, even when not actually used by the user. One can hence leave it out of the program altogether by disabling it in the makefile if the performance is crucial. The flag “COMPARISONSEEDING” is a debugging feature, is off by default and when enabled it makes the simulator artificially seed the random number generator in order to produce comparable simulation results.

To compile SPATE-HPC, run the command `make` from the base directory of SPATE-HPC in a command line prompt. If the compilation process succeeds, an executable file named “spate-hpc” on Unix or “spate-hpc.exe” on Windows will be created in the base directory. This is the program file used to run the simulator and it may be freely moved to another directory if the user so desires.

If the makefile has been altered after a partial or full compilation, it is necessary to clear out old object files before recompiling the program. This is done by first running `make clean`. Then `make` can be run again.

2.2 Running SPATE-HPC

SPATE-HPC is a command-line program, i.e., it must be run from a command prompt. SPATE-HPC requires one command-line argument: the name of the XML parameter file. On a single-processor system, SPATE-HPC can be run with the following command:

```
./spate-hpc input-file.xml
```

Here, the current working directory is the one where the executable “spate-hpc” resides and the XML input file is named “input-file.xml”.

When running SPATE-HPC, the paths of externally referenced input files are relative to that of the XML parameter file. In other words, if the XML parameter file resides in a given directory all other input files—such as the polygon files—are expected to reside in the same directory if the filenames of the external files are given without a full path. For the sake of coherency, it is recommended to put externally referenced files in the same directory as the XML parameter input file.

Output files, however, are not created in a directory relative to that of the input file. Ouput files are created in the current directory (the directory from which SPATE-HPC is executed) or in a directory relative to that. If the user provides an output directory name, such as “output-%t”, a directory with that name with “%t” replaced by the current time stamp is created and all output files put into that directory.

2.2.1 Running on multiple processors

If SPATE-HPC has been compiled with MPI support it can be run on a multi-processor system in order to reduce total execution time and increase the maximum domain size. This guide will describe how to run SPATE-HPC in parallel using MPICH 2 on a Linux operating system. However, the exact procedure may vary between different systems and set-ups. Large supercomputers might use their own batch system to run parallel programs.

Firstly, the MPI background daemon (`mpd`) must be running. Start it with:

```
mpd &
```

SPATE-HPC can then be run in parallel by prefixing it with `mpirun`:

```
mpirun -n 4 spate-hpc input-file.xml
```

The number following the letter “n” is the number of processors to run on. This number can be any integer between one and the maximum number of processors available on the system.

However, depending on the forest size, SPATE-HPC limits the number of processors that can be used in a particular simulation. The forest area is divided amongst the processors into rectangles of different width and height, depending on the shape of the compartments[1]. Any such rectangle must not have a width or a height smaller than 100 meters. Since the size of the rectangles cannot directly be controlled by the user, the number of processors to run on must simply be chosen small enough not create rectangles smaller than 100 meters.

Load balancing

SPATE-HPC can dynamically rebalance the load amongst the processors as the simulation is running. Because the initial domain decomposition is solely based on the area, the relative processor load might become uneven due to uneven tree density, which can be a result of uneven growth or felling. An uneven load balance leads to longer execution times. *Iteration load balancing* counteracts this by adjusting the inter-process boundaries and transferring trees from heavily loaded processors to more lightly loaded processors.

[1] Compartments are described in section 3.3.

Since the rebalancing procedure takes some time to carry out, it is often beneficial to allow a slight imbalance among the processors and not perform a rebalance after every iteration step. This allowed imbalanced is controlled with the *allowed imbalance percentage* parameter. The imbalance percentage is the ratio between the difference in number of trees and the minimum number of trees of any processor. When the imbalance percentage exceeds the allowed imbalance percentage a rebalance is carried out.

2.3 Compiling SPATE-GUI from source code

The input file editor SPATE-GUI can be compiled from source code with the GNU Compiler Collection (GCC) C++ compiler. The program uses the widget toolkit wxWidgets for the graphical components which makes it possible to compile it natively on Microsoft Windows (using Cygwin), Linux and Mac OS X. The only third-party library dependency of SPATE-GUI is wxWidgets, which should be at least of version 2.8. wxWidgets is available from http://www.wxwidgets.org.

Akin to SPATE-HPC, SPATE-GUI is compiled with the Make program. There is a sample makefile called “makefile.sample” which should be copied an named “makefile”. The makefile should then be edited to reflect the system on which SPATE-GUI is to be compiled. The following is a list of makefile variables that can be changed:

CXX The name of the compiler executable. The default is “g++”, which is the GCC C++ compiler.

WX_CONFIG The name of the wxWidgets compilation utility. The default value, “wx-config”, should be appropriate on most systems.

TARGET_OS This variable should be set to one of “windows”, “linux” or “mac” depending on the operating system for which SPATE-GUI is to be compiled.

EXTRA_LIBS This variable can be used to include additional libraries.

LDFLAGS This variable is used to provide additional flags to the linker.

CXXFLAGS This variable is used to provide additional flags to the C++ compiler.

To compile SPATE-GUI, run the command `make` from the base directory of SPATE-GUI in a command line prompt. If the compilation process succeeds, an executable file named “spate-gui” on Linux or “spate-gui.exe” on Windows will be created in the base directory. This is the program file used to run the program and it may be freely moved to another directory if the user so desires. On Mac OS X, a proper application bundle named “SpateGUI” is created.

If the makefile has been altered after a partial or full compilation, it is necessary to clear out old object files before recompiling the program. This is done by first running `make clean`. Then `make` can be run again.

2.4 Running SPATE-GUI

SPATE-GUI is graphical user interface program and can be started on most operating systems by double clicking on the executable file. On Windows this file is named “spate-gui.exe”, and on

Linux it is named “spate-gui”. On Mac OS X the application bundle “SpateGUI” is used to run the program.

Chapter 3

The simulator

SPATE-HPC performs a simulation as an iterative process consisting of a sequence of time steps. The length of a time step is chosen by the user, but is at least one year long. A time step is further decomposed into a series of individual phases, which each governs a certain aspect of the stand development. The phases are *growth*, *mortality*, *reproduction* and *management*. In the growth phase pre-existent trees grow, in the mortality phase trees die, in the reproduction phase new trees are generated, and in the management phase silvicultural treatments are carried out. The changes introduced in the growth, mortality and reproduction phases do not take affect until after all three phases have been performed. When the growth, mortality and reproduction phases have been performed and the changes applied the management phase is carried out.

SPATE-HPC can run several replications, which are simulations with identical initial conditions both with differently drawn random numbers. The results of the different replications are combined for the final statistics in order to produce figures within a Monte Carlo confidence interval. Figure 3.1 shows a diagram of the simulation workflow.

3.1 Tree

In SPATE-HPC, each tree is simulated individually. A single tree is also the smallest entity within a simulation. A tree can be viewed as collection of attributes or characteristics, which are the following:

X and Y coordinates	Two real numbers that represent the tree's coordinates in a two-dimensional Euclidean space.
Diameter	DBH, or diameter at breast-height (1.3m).
Height	Total height.
Initial heterogeneity	A normally distributed random number assigned to the tree upon its creation. Also referred to as tau (τ).
Age	Age in years.
Species	Tree species (pine, spruce, birch, etc.).

The characteristics diameter and height are collectively referred to as *marks*.

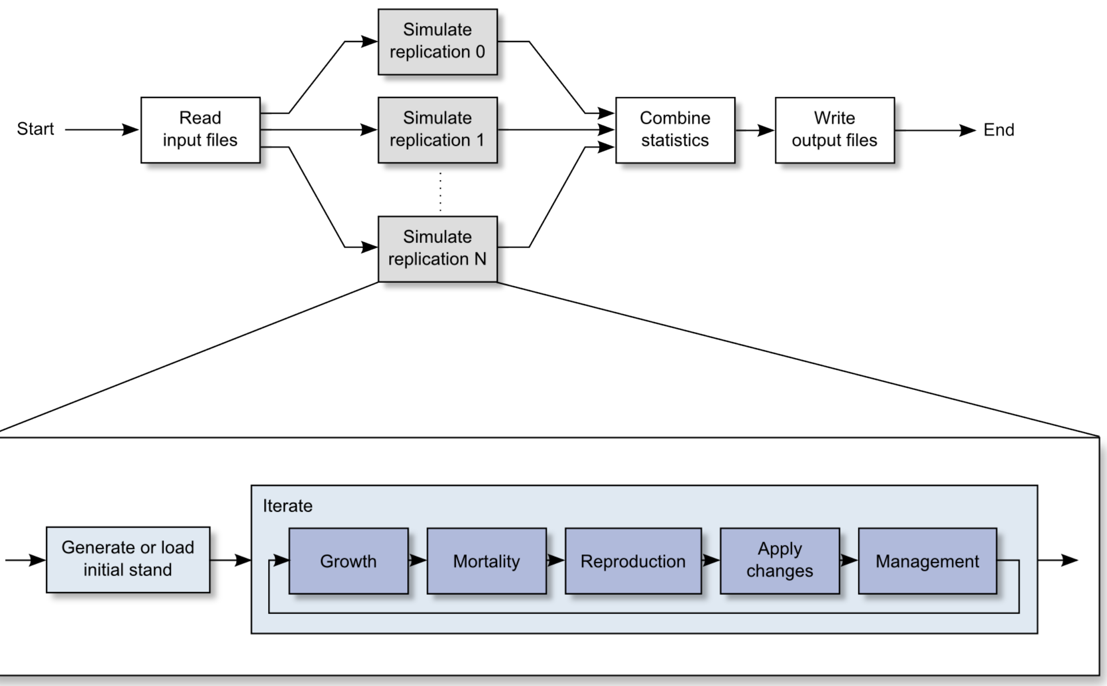


Figure 3.1: Diagram showing the workflow of a simulation.

3.2 Spatial inter-tree dependence

Growth, mortality and reproduction are calculated independently for each tree, but the outcome can be made dependent on neighboring trees. The dependency is achieved through competition indices and spatially correlated random numbers. Competition indices are variables whose values are calculated based on the target tree and its local neighboring trees. The spatially correlated random numbers are drawn so that trees within closer proximity are more strongly correlated than trees farther apart, see section 3.4.2.

When competition indices are calculated, only the neighboring trees located within a certain competition distance, or competition radius, are taken into consideration. These neighboring trees are referred to as competitors. The competition distance is global for the entire forest, but may vary between iterations.

3.3 Compartments

The entire forest may be subdivided into one or several compartments, and a compartment comprises one or several polygons, see figure 3.2. Different compartments may have different forest types and may be assigned different management methods. The forest is spatially continuous across compartment boundaries, i.e., trees in different compartments near compartment borders interact with each other as if no compartment boundaries existed. The management methods, however, are applied separate to each compartment. When a simulation has finished, statistics are compiled individually for each compartment. Each compartment is associated with a unique, user-defined numerical ID that is used to identify the compartment.

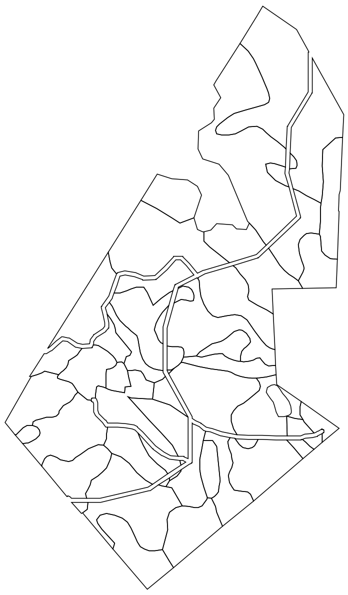


Figure 3.2: A sample forest area with 38 compartments consisting of 57 polygons.

3.4 Simulation model

Conventionally, the models used in the growth, mortality and reproduction phases, as well as in the initial stand generation, follow the pattern

$$y = \boldsymbol{x\beta} + \alpha\tau + \gamma\xi + \varepsilon \tag{3.1}$$

where $\boldsymbol{x}$ is a row vector of competition indices, $\boldsymbol{\beta}$ is a column vector of coefficients, τ is the initial heterogeneity, ξ is a correlated random number, α and γ are coefficients and ε is a normally distributed random number, corresponding to the error term.

The models defined by the user, however, need not strictly take this form, but may follow a more generic pattern:

$$y = \beta_0(x_{00}^{p_{00}} \cdot x_{01}^{p_{01}} \cdot \ldots) + \beta_1(x_{10}^{p_{10}} \cdot x_{11}^{p_{11}} \cdot \ldots) + \ldots \tag{3.2}$$

Here, β_i are coefficients, x_{ij} are variables and p_{ij} are exponents. The variables in equation (3.2) can be competition indices as well as τ, ξ and ε. A model in SPATE-HPC is a set of terms where each term has a coefficient and a set of variables. There can be one or several variables in a term and they can be exponentiated.

Furthermore, the coefficients can be expanded into a *secondary-level* model:

$$\beta_0 = \beta_{00}^*(w_{00}^{q_{00}} \cdot w_{01}^{q_{01}} \cdot \ldots) + \beta_{01}^*(w_{10}^{q_{10}} \cdot w_{11}^{q_{11}} \cdot \ldots) + \ldots \tag{3.3}$$

Here, β_{ij}^* are coefficients, w_{ij} are secondary-level variables and q_{ij} are exponents. Analogously to the top-level model, a secondary-level model also comprises a set of terms, where each term has a coefficient and a set of variables. Secondary-level variables can also be multiplied by each other and exponentiated. Secondary-level models are also referred to as *smoothing functions*.

3.4.1 Variables

The following is a list of variables that can be used as elements of $\boldsymbol{x}$.

Intercept
: The constant 1.

Diameter
: The diameter at breast-height (DBH): d_t

Diameter log
: The natural logarithm of diameter: $\ln(d_t)$

Local density
: The tree density within the competition radius. (Trees per hectare.): $10000N/A$

Average distance
: The average distance to competitors: $\sum_1^N R_i/N$

Inverse distance
: The sum of the inverse distance to all *larger* competitors. A competitor is larger if it has a larger diameter. $\sum_1^N 1/R_i$

Diameter sum
: The sum of all competitors' diameters: $\sum_1^N d_i$

Diameter difference
: The sum of the differences between the target tree's diameter and the competitors' diameters: $\sum_1^N (d_t - d_i)$

Height difference
: The sum of the differences between the target tree's height and the competitors' heights: $\sum_1^N (h_t - h_i)$

Height diameter ratio
: The quotient of the target tree's height and diameter: h_t/d_t

Basal area of competitors
: The competitors' total basal area relative to a hectare: $10000B/A$

Relative basal area
: The target tree's basal area relative to the competitors' basal area: $\pi \cdot (d_t/200)^2/B$

Diameter quotient
: The sum of all *larger* competitors' diameters divided by the target tree's diameter: $\sum_1^N d_i/d_t$

Basal area to mean diameter ratio
The competitors' basal area divided by the mean diameter of the competitors: $B/(\sum_1^N d_c/N)$

Tau The target tree's initial heterogeneity: τ

Xi A correlated random number: ξ

Epsilon
A normally distributed random number. It is drawn anew for each calculation. $\varepsilon \sim \mathcal{N}(0, 1)$

The variables referred to are defined as:

N	Number of competitors
r	Competition distance [m]
R_i	Distance between target tree and neighbor i [m]
d_t	Diameter of target tree [cm]
h_t	Height of target tree [m]
d_i	Diameter of competitor i [cm]
h_i	Height of competitor i [m]
A	Competition area [m^2]: πr^2
B	Basal area of competitors [m^2]: $B = \pi \sum_1^N (d_i/200)^2$

The variables that can be used in the secondary-level model (***w***) are the following:

Intercept	The constant 1.
Management	A value given the management method of the tree's compartment. If the compartment is assigned several management methods, the value of the first one is used.
Species	A value given the tree's species.
Age	The tree's age in years.
Age class	The tree's age class as an index (0, 1, . . .).
Size class	The tree's diameter size class as an index (0, 1, . . .).
Height class	The tree's height class as an index (0, 1, . . .).
Indicator variable: species	Replace "species" with the name of a species. The variable will assume the value 1 if the species matches that of the target tree and 0 otherwise.
Epsilon	A normally distributed random number: $\varepsilon \sim \mathcal{N}(0, 1)$.

The size class indices all start from zero. In other words, if a tree belongs to the first age class the variable "Age class" will assume the value 0, and if it belongs to the second class the variable will assume the value 1, etc.

3.4.2 Environmental effects

Environmental effects are correlated random numbers that are generated in such a way that random numbers for trees in close proximity are more strongly correlated than those for trees farther apart. The covariance between two trees i and j is calculated with the following function:

$$\sigma_{i,j} = \begin{cases} \exp(\theta_0 + \theta_1 r_{ij}) & \text{if } R_{ij} \leq r \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where $\theta_0,\ \theta_1 \leq 0$ are parameters chosen by the user, R_{ij} is the distance between the trees and r is the competition distance.

The fundamental algorithm for calculating the correlated random numbers is to construct a variance–covariance matrix $\mathbf{\Sigma}_\xi$ whose cells are given by formula (3.4). The matrix is then decomposed into a lower-triangular matrix with the Cholesky decomposition: $\boldsymbol{L}_\xi \boldsymbol{L}_\xi^\top = \mathbf{\Sigma}_\xi$. Finally, correlated random numbers are obtained by multiplying the lower-triangular matrix by a vector of uncorrelated normally distributed random numbers: $\boldsymbol{\xi} = \boldsymbol{L}_\xi \boldsymbol{z}$, $\boldsymbol{z} \sim \mathcal{N}(0,1)$. Because $\mathbf{\Sigma}_\xi$ must be a positive definite matrix in order to perform a Cholesky decomposition, the values of θ_0 and θ_1 must be chosen so that $\mathbf{\Sigma}_\xi$ is positive definite in all situations.

Due to technical limitations the full variance–covariance matrix and the lower-triangular matrix for all trees in the forest cannot be calculated. Instead, the correlated random numbers are approximated by performing multiple smaller local Cholesky decompositions. The local Cholesky decomposition algorithm performs one decomposition per tree, where the variance–covariance matrix is constructed of only the immediate neighboring trees of the target tree. The trees included are the ones that are located within the competition distance multiplied by the *local correlation factor*. The default value of the local correlation factor is 1, but it can be increased by the user. Increasing it to a higher value will enlarge the local variance–covariance matrix and thus likely decrease the approximation error, but also prolong the simulation time.

3.5 Simulation phases

This section describes how the simulation phases initial stand, growth, mortality and reproduction are performed.

3.5.1 Initial stand

The initial stand can either be generated or loaded from a file. The file may also be incomplete, i.e., it may lack certain tree characteristics, whereupon the missing characteristics are generated with a model. If no tree data file is given, the first step is to generate tree positions for the new trees. The number of new trees to create is drawn as a Poisson random number with a mean value provided by the user. The trees are then placed by a point process.

When the trees have been placed they are assigned tree marks (diameter and height). The tree marks are calculated according to models given by the user, which follow the form of formula (3.2). After a mark has been calculated for all trees in the stand it can optionally be transformed to adhere to a given distribution. The marks are calculated and transformed in order: first the diameter is calculated and transformed, then the height is calculated and transformed. The height calculation can hence be made dependent upon the (transformed) diameter. Section 3.6 contains more information about mark transformations.

3.5.2 Growth

A tree grows by multiplying a mark's value of the previous time step by a growth factor. The growth factor is the value of the exponential function with y from formula (3.2) as the parameter, i.e.,

$$m_t = m_{t-1} \cdot \exp(y) \tag{3.5}$$

Here, m_t is a mark value (diameter or height) of a tree at time step t. The growth is calculated separately for each mark.

3.5.3 Mortality

There are two types of mortality in SPATE-HPC: regular and irregular, random mortality. The regular mortality occurs due to deficient growth, whereas the irregular mortality occurs randomly, but with a probability affected by local competition.

The regular mortality occurs when a tree's diameter growth in the current iteration is less than a specific *mortality threshold*. A mortality threshold of 1.0 would hence mean that trees are not allowed to shrink. There may be a separate mortality threshold for the diameter and the height. If no mortality threshold is given, the regular mortality is disabled.

The irregular mortality is modelled with a logistic regression model. The probability π of an individual tree dying is calculated as

$$\pi = \frac{1}{1 + \exp(-y)} \tag{3.6}$$

where y is obtained from formula (3.2). A uniformly distributed random number $u \sim U(0, 1)$ is drawn. If $u \leq \pi$ the tree dies. One may provide a separate irregular mortality model for each mark.

3.5.4 Reproduction

In the reproduction phase new trees are generated. The number of trees to generate is a Poisson random number whose mean is obtained from a linear model called the *regeneration* model. The trees are then placed by a homogeneous Poisson point process. The point process is hard-core, i.e., new trees cannot be placed inside the trunk of pre-existent ones. Finally, the new trees are assigned marks according to a model of formula (3.2) and the mark values are transformed to target distributions analogously to the corresponding process in the initial stand. Section 3.6 contains more information about mark transformations.

Regeneration model

The regeneration model, which is used to determine the expected number of new trees in the reproduction phase, is a linear model of the form

$$n = \max\{0, ax_0 + bx_1 + cx_2 + \ldots\} \tag{3.7}$$

where a, b, etc. are constants and x_0, x_1, etc. are variables. The model is hence restricted to produce only non-negative results. The available variables, which cannot be combined or exponentiated, are the following:

Intercept	The constant 1.
Volume	The volume of all trees in the compartment per hectare [m^3/ha].
Basal area	The basal area of all trees in the compartment per hectare [m^2/ha].
Trees	The total number of trees in the compartment per hectare [1/ha].
Inverse volume	The inverse of *Volume*.
Inverse basal area	The inverse of *Basal area*.
Inverse trees	The inverse of *Trees*.
Epsilon	A normally distributed random number $\varepsilon \sim \mathcal{N}(0, 1)$. The value is limited to the *standard normal limiting factor*.

3.6 Mark transformation

When new tree marks (diameter and height) are generated for the initial stand and in the reproduction phase, they will be normally distributed with an infinite support if ‘epsilon’ or ‘xi’ are used in the models. To allow generated marks to adhere to an arbitrary distribution, and to prevent negative mark values, marks can be transformed to adhere to a user-defined distribution.

The transformation process begins by building a cumulative distribution of the calculated untransformed mark values. This is done when mark values for all trees in the entire forest have been calculated. The second step is to build a cumulative distribution according to the target distribution specified by the user. Different compartments may have different target distributions. Finally, the mark values are transformed to adhere to the target distribution through cubic spline interpolation.

When mark values are calculated and transformed, these steps are performed separately for the diameter and the height. First, the diameter values are calculated and transformed, and then the height values are calculated and transformed. This means that if the height model uses the diameter as a model variable, the variable’s value will be the transformed one.

The following are the options the user may choose for the target distribution.

Normal distribution
: A normal distribution with a mean μ and standard deviation σ given by the user. The support is truncated to $[\mu - \delta, \mu + \delta]$ where $\delta = \min\{\mu, 3\sigma\}$, i.e., the support is no more than six standard deviations wide, but not allowed to encompass negative values.

Weibull distribution
: A Weibull distribution with the shape, scale and origin (location) parameters given by the user..

Empirical data

The user provides a set of data points from which a cumulative distribution is built. The endpoints of the support are the smallest and the largest data points given. The data format is described further in section 5.3.

Empirical cumulative distribution

The user provides an arbitrary cumulative distribution as a set of $\langle x, F(x)\rangle$ pairs. The data format is described further in section 5.4.

3.7 Species

There can be one or several tree species and every tree is of a specific species. Species are defined in the parameter input file, which means that users are free to define their own species. It is possible for different species to have different growth, mortality and reproduction behavior, and trees of different species may have different stem volumes.

Each species has a *model value*. The model value is used for the variable "Species" in the second-level models. This way, different species can be modeled to experience different growth, mortality and reproduction.

Two different methods for estimating a tree's trunk volume is used: taper curve functions and a cylinder model. The cylinder model is only used when the tree is extremely small whereas taper curves are used in all other cases. The taper curve functions used are the ones suggested by Laasasenaho[1], which provide an estimate for the tree's diameter at any arbitrary height position. The relationship between the diameter at an arbitrary height and the diameter at 20% of the height is calculated using the model

$$f_b(x) = \frac{d_l}{d_{.2h}} = b_1x + b_2x^2 + b_3x^3 + b_5x^5 + b_8x^8 + b_{13}x^{13} + b_{21}x^{21} + b_{34}x^{34} \tag{3.8}$$

where d_l is the diameter at height l, $d_{.2h}$ is the diameter at 20% height, h is the tree's height and $x = 1 - l/h$. Since $d_{.2h}$ is not known, it is estimated as

$$\hat{d}_{.2h} = \frac{d_{1.3}}{f_b(1 - 1.3/h)} \tag{3.9}$$

where $d_{1.3}$ is the DBH. Finally, the diameter at an arbitrary height l is estimated as

$$\hat{d}_l = f_b\hat{d}_{.2h}(1 - l/h) \tag{3.10}$$

The coefficients $b_1, b_2, \ldots, b_{34}$ are species-specific and are provided by the user.

When the volume of a tree is calculated the calculation does not start from the ground, but from the heigh position of an imagined stump. The stump height h_s (in meters) is calculated using the model

$$h_s = \max\{(a \cdot d_{1.3} + b \cdot h)/100, h_{ms}\} \tag{3.11}$$

[1] Jouko Laasasenaho. *Taper Curve and Volume Functions for Pine, Spruce and Birch.* Finnish Forest Research Institute, Helsinki, 1982

where a and b are species-specific coefficients given by the user and $h_{ms} \geq 0$ is the lowest possible stump. The diameter is in centimeters and the height in meters.

Taper curve functions are sidestepped and a simple cylinder model is used instead when the tree is shorter than four meters or has a diameter less than three centimeters. The cylinder model approximates the tree to a cylinder with the same height as the tree and with the tree's DBH as the cylinder's diameter. In addition, the cylinder volume is multiplied by the *small volume coefficient*, which is species-dependent and has the default value 0.6.

A species also has a *competition distance model*. Different species may have different competition distance models and the computed value from competition distance model might be different for different trees. However, all trees have the same effective competition distance, which is the maximum value obtained by calculating the competition distance of all trees in the forest. The competition distance model takes the same form as model (3.2) and use variables and secondary-level variables described in section 3.4.1. However, the model calculation cannot take neighboring trees into account, which makes only the following variables useful in this model: 'Intercept', 'Diameter', 'Diameter log', 'Height diameter ratio', 'Tau' and 'Epsilon'. All the secondary-level variables can be used though.

3.8 Forest types

A forest type defines a particular composition of tree species, and a compartment is of a specific forest type. A forest type has the following attributes:

Name
A unique name that is used to associate a forest type with a compartment.

List of species
Each species is assigned a number that corresponds to its relative amount in the initial stand. If desired, the species can be assigned a separate relative amount for the reproduction phase. Furthermore, the species' mean diameter can be offset from the global one in the initial stand and reproduction phases by assigning diameter factors.

Initial stand transformations
One may provide target distributions for the diameter and height marks that are applied in the initial stand generation phase.

Reproduction transformations
One may provide target distributions for the diameter and height marks that are applied in the reproduction phase.

3.9 Management methods

Management methods (silvicultural treatments) are applied to compartments in order to thin or harvest the stand by removing trees according to a specific procedure. The management methods implemented in SPATE-HPC are *dimension cutting, energy wood cutting, low thinning, selective thinning* and *clearcutting*.

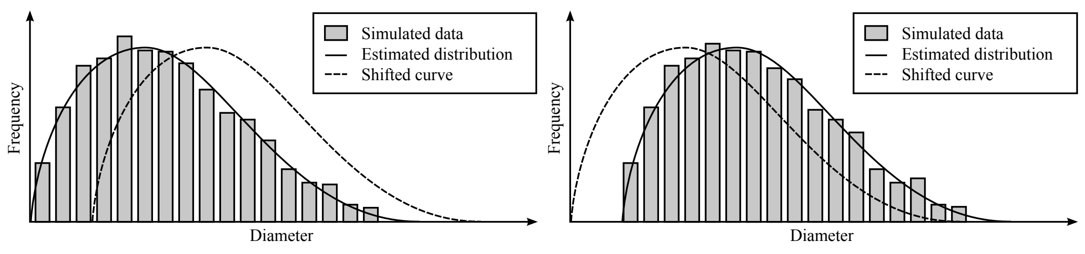
Management methods are assigned to compartments via *management templates*. A management template may contain one of dimension cutting, energy wood cutting, low thinning, selective thinning, or none of them. In addition, a management template may contain a clear cut. A compartment can associated with none, one or several management templates. Furthermore, the same management template can be assigned to several different compartments.

3.9.1 Dimension and energy wood cutting

Dimension and energy wood cutting operate in the same way but their results are diametrically opposite. Dimension cutting is only given one parameter: *diameter limit*, and when dimension cutting is carried out, all trees with a diameter wider than the diameter limit are cut down. Conversely, when energy wood cutting is carried out trees with a thinner diameter than the diameter limit are cut down.

3.9.2 Weibull thinning

Low and selective thinning are collectively referred to as Weibull thinning. In Weibull thinning trees are categorized according to diameter into size classes. The default width of a size class is 4cm, but this can be changed by the user. Furthermore, the diameter distribution is approximated by a Weibull distribution. The curve that represent the approximate Weibull distribution is then shifted in either direction: to the right in low thinning and to the left in selective thinning.



(a) Low thinning (b) Selective thinning

Figure 3.3: The simulated tree distribution is represented by the histogram and the estimated distribution by the solid curve. The dashed curve is the shifted curve.

This is readily illustrated by drawing a histogram of the number of trees in the size classes and the Weibull curves, scaled to conform to the height of the histogram bars. There is now an overlapping area between the two Weibull curves where the histogram bars are likely to protrude above the shifted Weibull curve. This area is to the left in low thinning and to the right in selective thinning. The objective is to stochastically cut trees so that the protruding histogram bars are trimmed down the height of the shifted Weibull curve. Figure 3.3 illustrates how the curves are shifted.

The amount of trees to cut in a given size class is controlled by the thinning rate. The thinning rate is a size-class-dependent value between 0 and 1 and represents the number of trees in the corresponding size class that are to be cut. The thinning rate is calculated as the part of the histogram bar protruding above the shifted curve in relation to the whole histogram bar.

Low thinning

Low thinning starts with the smallest size class and proceeds with larger size classes. The last size class to be included in the thinning procedure is the one where the intersection point between the two Weibull curves is located. Trees are cut stochastically and the thinning probability of a tree being cut is the thinning rate, i.e., all trees in a given size class have the same thinning probability.

Selective thinning

Selective thinning starts with the largest size class and proceeds with smaller size classes. As in low thinning, the last size class to be included is the one where the intersection point lies. In contrary to low thinning, selective thinning also considers the local tree density. The likelihood of trees being cut in denser neighborhoods is higher than that of trees in sparser neighborhoods.

Each tree in a size class is assigned its own individual thinning probability ω, which is calculated using the logistic regression model:

$$\omega = \frac{1}{1 + \exp\big(-\beta_0 - \beta_1 \ln(\lambda_U/\lambda_A)\big)} \tag{3.12}$$

where λ_U is the local density around the target tree and λ_A is the overall stand density of the compartment. The coefficients β_0 and β_1 are the *thinning level* and the *density coefficient*, respectively, which are parameters the user may specify. The number of trees to cut in a size class is determined by a Poisson random number whose mean value is the class's thinning rate multiplied by the total number of trees in the class. The program then tries to stochastically cut this amount of trees according to the trees' individual thinning probabilities. It can however fail to cut the desired number of trees if the trees' thinning probabilities are overall low.

Achieving the target volume

The user controls the amount of trees to cut with the *removal per hectare* parameter. The objective is to cut trees so that the accumulated volume of trees removed matches the desired removal volume. The result of the thinning is controlled by the distance the Weibull curve is shifted. However, no algorithm is implemented to directly determine the shift distance based on the target removal volume. Instead, the target volume is achieve through iteration, i.e., by testing different shift values until a sufficiently good one is found. The amount the result may deviate by from the target volume is controlled with the *removal tolerance* parameter, whose default value is 5%. It is possible that the program concludes that the target volume cannot be achieved. In this case the program will reject a resulting volume larger than the target volume, but will accept a smaller resulting volume.

Weibull curve with linear tail

The tail of the Weibull curve can be linearized so that it does not approach zero asymptotically like it normally would, but instead transition into a linear function that reaches zero prematurely. The purpose of this functionality is to reduce the Weibull curve's tail in order to increase the thinning rate of the largest size classes. If the *linear tail ratio* parameter is specified, the linear tail will start when the ratio between the gradients of two consecutive size classes, both with negative slopes, is greater than the linear tail ratio parameter. Formally, if $a_k/a_{k+1} > \alpha$, where a_k is the gradient at the centre of size class k and α is the linear tail ratio, the linear tail will start at the centre of class k and have the gradient a_k while the rest of the Weibull curve is discarded.

3.9.3 When is thinning carried out

There are three parameters to control when the thinning methods are to be applied: *lower basal area*, *lower compartment volume* and *thinning interval in years*. These parameters apply to all management methods, save for the thinning interval in years, which does not apply to clearcutting.

The lower basal area is the smallest basal area the compartment may have before thinning can be applied. Thinning is carried out if the compartment's basal area is larger than the lower basal area, and the unit of the lower basal area parameter is square meters per hectare (m^2/ha). Analogously to the lower basal area, the lower compartment volume is the smallest volume the compartment may have before thinning can be applied. The unit of the lower compartment volume is cubic meters per hectare (m^3/ha). If both the lower basal area and the lower compartment volume criteria are specified, thinning will be carried out whenever *either* of them is fulfilled.

The thinning interval is the minimum number of inactive years between two subsequent thinnings. In practice, the actual interval between two thinnings might be longer than the number of years given, in case the other criteria (lower basal area or minimum volume) have not been satisfied earlier. The thinning interval is only enforced between two subsequent thinnings and is thus not affected by the stand's initial age. The first thinning might very well take place the first year of the simulation.

3.9.4 Clear cut

Clearcutting cuts almost every tree in the compartment. Clearcutting is not affected by the criteria described in section 3.9.3. Instead, the frequency of clear cuts is specified with the *years between clear cuts* parameter. The first clear cut takes place when the stand has reached an age that is a multiple of the years-between-clear-cuts parameter. For example, if the age of the initial stand is 40 years and the years-between-clear-cuts parameter is 50, the first clear cut takes place the tenth simulated year and second one the sixtieth simulated year, etc. If the lower basal area or lower compartment volume criterion is given, however, the clear cut might be postponed which consequently offsets subsequent clear cuts.

Clearcutting does not fell all trees but preserves some of them. This is controlled with the *trees left per hectare* and *forced tree distance* parameters. The program will try to find and spare the desired number of trees that stand no closer to one another than the forced tree distance. This works by first categorizing trees into size classes according to diameter. The program then tries to find the trees to spare by analyzing the trees in the largest size class. If the desired number of trees is not attained, the program tries again by also including the second largest size class in the search, and then the third, etc. Larger trees are hence prioritized in the search for trees to spare.

Moreover, the size of trees to spare can be limited to a certain diameter range. Trees larger than the maximum diameter and trees smaller than the minimum diameter will thus not be considered when searching for trees to spare. In case the program cannot find as many trees to spare as desired, it will accept the reduced number, i.e., the trees found will be spared, the rest cut down nonetheless and a pertinent message added to the log file.

3.10 Assortments

Assortments are categories of wood raw material that are obtained by cutting the stem of a felled tree into logs, for example saw log and pulpwood. If assortments are defined SPATE-HPC will compile assortment-specific volume statistics after a simulation has completed. An assortment may have a set of size criteria—*minimum diameter* and *log length*—that dictate how the assortment shall be calculated. Additionally, there may be different size criteria for different species.

SPATE-HPC calculates assortment volumes of a felled tree according to the following algorithm. It starts at the height of the stump and progresses toward the top of the tree. The program tries to cut a log according to the first assortment's length criterion. If the diameter at both ends of the log is at least as wide as the minimum diameter criterion the log is cut off the stem and the program continues by trying to cut another log of the same assortment. If the size criteria are not fulfilled, the log will not be cut and the program switches to the next assortment and tries to cut logs according to that one's criteria. The process continues until there is no stem left or there are no assortments whose size criteria can be fulfilled.

If an assortment does not have size criteria it can be used for the top of the tree. When the program switches to an assortment without size criteria the rest of the stem all the way to the top of the tree will be used for that assortment.

3.11 Output files

SPATE-HPC produces four types of output files during and after a simulation: compartment statistics files, statistics table files, tree data files and log files. There is one compartment statistics file for each compartment, which is named "simulationname_statistics_compartment_xxxx.dat" where "simulationname" is replaced by the name of the simulation and "xxxx" by the corresponding compartment's numerical ID. A compartment statistics file contains statistics pertaining only to that compartment. In addition to the compartment statistics files the simulator also produces a statistics table file. The statistics table file is named "simulationname_statistics_table.dat" and contains a table of the final characteristics of all compartments. Compartment statistics files are described further in section 5.6 and statistics table files in section 5.7.

Tree data files are files containing the characteristics of all simulated trees in a table format. Tree data files are optional and can be produced for each replication at the end of the simulation, and additionally, for each replication at the end of each iteration step. The files produced at the end of the simulation are named "simulationname_replication_xxxx.dat", where "xxxx" is the replication number, and the files produced after each iteration step are named "simulationname_replication_xxxx_iteration_yyyy.dat", where "yyyy" is the iteration number. The format of the tree data files are described in section 5.5. Enabling iteration-wise tree data files will likely consume large amounts of hard disk space and slow down the simulation process noticeably.

Log files are written continuously as the simulation is running and contains information about each simulation step as it is carried out. Log files may hence prove a valuable resource in understanding the simulation process or tracking down error conditions. There is one log file written by each processor, and the corresponding log file contains only information related to the that processor. If SPATE-HPC is compiled without MPI support, the single log file is named "output.txt", whereas if SPATE-HPC is compiled with MPI support, the log files are named "xxxx-output.txt", where “xxxx” is the processor number.

Chapter 4

The input file editor

This chapter describes the input file editor SPATE-GUI. SPATE-GUI can be used to create and modify the input parameter file, which is in the XML file format. Other input files, such as polygon files, tree data files and files containing empirical distributions cannot be edited with SPATE-GUI and must be created using other programs.

Although the input parameter file can reference external files located in any location, it is recommended for the sake of coherency that all input files be put in the same directory. One can thus begin by creating a new directory for the simulation project, move or copy polygon and other external files to that directory and then use SPATE-GUI to create the input parameter file in that directory.

4.1 General page

On the 'General' page are settings controlling general aspects of the simulation. Here are also the 'Save settings' and 'Load settings' buttons located, which are used to save and load the XML input file, respectively. A screenshot of the 'General' page is shown in figure 4.1.

The settings that are configurable on the 'General' page are the following:

Output directory The directory where output files should be created. If a relative path is given SPATE-HPC will create the output directory relatively to the directory from which SPATE-HPC is run.

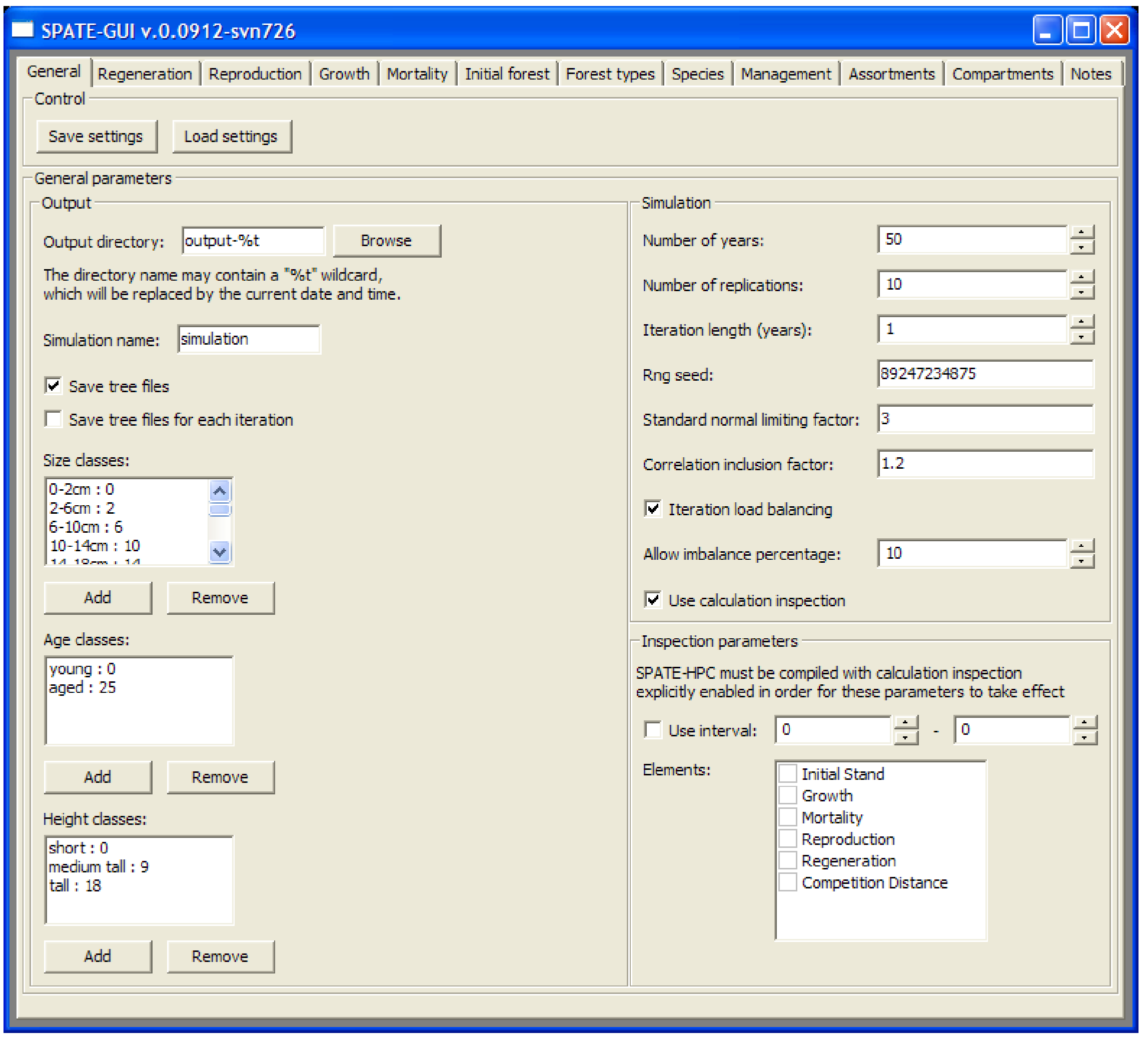
Simulation name A name describing the simulation. Output files will be prefixed with this name.

Save tree files Whether to save tree data files. Enabling this option saves tree files for each replication at the end of a simulation.

Save tree files for each iteration Whether to save tree data files after each iteration step as well. This feature is likely to consume large amount of hard disk space and slow down the simulation process noticeably.

Size classes Size classes are used to classify trees according to DBH.

Age classes Age classes are used to classify trees according to age.


SPATE-GUI v.0.0912-svn726

General | Regeneration | Reproduction | Growth | Mortality | Initial forest | Forest types | Species | Management | Assortments | Compartments | Notes

Control

Save settings | Load settings

General parameters

Output

Output directory: output-%t | Browse

The directory name may contain a "%t" wildcard, which will be replaced by the current date and time.

Simulation name: simulation

- [✓] Save tree files
- [ ] Save tree files for each iteration

Size classes:

```
0-2cm : 0
2-6cm : 2
6-10cm : 6
10-14cm : 10
```

Add | Remove

Age classes:

```
young : 0
aged : 25
```

Add | Remove

Height classes:

```
short : 0
medium tall : 9
tall : 18
```

Add | Remove

Simulation

Number of years:	50
Number of replications:	10
Iteration length (years):	1
Rng seed:	89247234875
Standard normal limiting factor:	3
Correlation inclusion factor:	1.2

- [✓] Iteration load balancing

Allow imbalance percentage: 10

- [✓] Use calculation inspection

Inspection parameters

SPATE-HPC must be compiled with calculation inspection explicitly enabled in order for these parameters to take effect

- [ ] Use interval: 0 - 0

Elements:

- [ ] Initial Stand
- [ ] Growth
- [ ] Mortality
- [ ] Reproduction
- [ ] Regeneration
- [ ] Competition Distance

Figure 4.1: The general simulation settings

Height classes Height classes are used to classify trees according to height.

Number of years The number of years to simulate ahead of time.

Number of replications The number of replications to simulate.

Iteration length The length of an iteration step in years.

Rng seed A preferably large number that is used to seed the random number generator.

Standard normal limiting factor Some Gaussian random numbers are limited to the standard deviation multiplied by this factor.

Correlation inclusion factor The local correlation factor, see section 3.4.2.

Iteration load balancing Whether to readjust process boundaries during simulation in order to obtain a better load balance amongst processors. This option is only meaningful if SPATE-HPC is run in a parallel environment.

Allow imbalance percentage Since rebalancing takes some time to perform, the load balancing procedure will allow a slight imbalance amongst processors which is controlled with this parameter.

Use calculation inspection Enabling calculation inspection will output very detailed information about calculations performed during a simulation. It is not recommended for other than very small simulations when one needs to verify e.g. growth models, since it produces large amounts of log data and slows the simulation process down considerably.

Use interval One may specify a certain iteration interval when calculation inspection is to be used. The interval is inclusive.

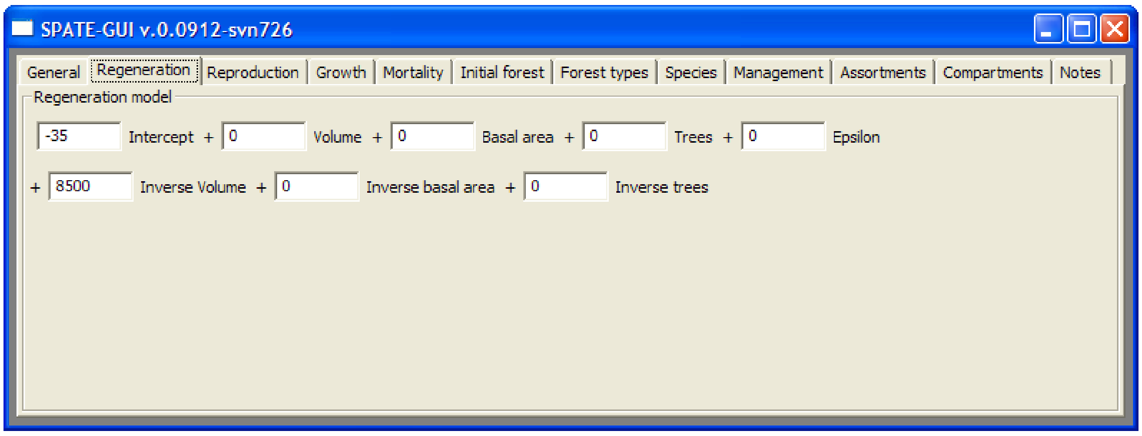
Elements There is one checkbox for each type of model or other calculation one can inspect.

4.2 Regeneration page

The 'Regeneration' page contains the coefficients of the regeneration model, see figure 4.2, which is used to calculate the number of new trees in the reproduction phase. Although not explicitly stated, the variables units are per hectare. The regeneration model is described in section 3.5.4.

4.3 Reproduction, Growth and Mortality pages

The 'Reproduction', 'Growth' and 'Mortality' pages contain the models for calculating reproduction, growth and mortality, respectively. The three pages are analogous and are therefore described collectively in this section. The 'Number of marks' input field is used to select how many marks (zero, one or two) should be calculated in the corresponding phase. If the value is set to zero the phase is disabled altogether. The 'Reproduction' page is shown in figure 4.3.


SPATE-GUI v.0.0912-svn726

Regeneration model

-35 Intercept + 0 Volume + 0 Basal area + 0 Trees + 0 Epsilon

+ 8500 Inverse Volume + 0 Inverse basal area + 0 Inverse trees

Figure 4.2: The ‘Regeneration’ page showing the model for calculating the expected number of new trees per hectare in the reproduction phase.

Each mark has its own separate model, which is represented by a by panel labeled ‘Mark’ within the editor window. The ‘Mark’ panel contains a ‘Name’ drop-down menu and the sub-panels ‘Environmental effects’ and ‘Model’. The ‘Name’ menu should be set to the appropriate mark. There cannot be two models of the same mark. The fields ‘Theta0’ and ‘Theta1’ within the ‘Environmental effects’ panel represent the coefficients θ_0 and θ_1 in equation (3.4), respectively. The actual model is shown in the ‘Model’ panel. In the same panel are also the buttons ‘Edit’ and ‘Clear’, which are used to modify and empty the model, respectively. On the ‘Mortality’ page there is an additional field called ‘Mortality threshold’. The mortality threshold is used to control the regular mortality, whereas the mortality models are used for the irregular mortality.

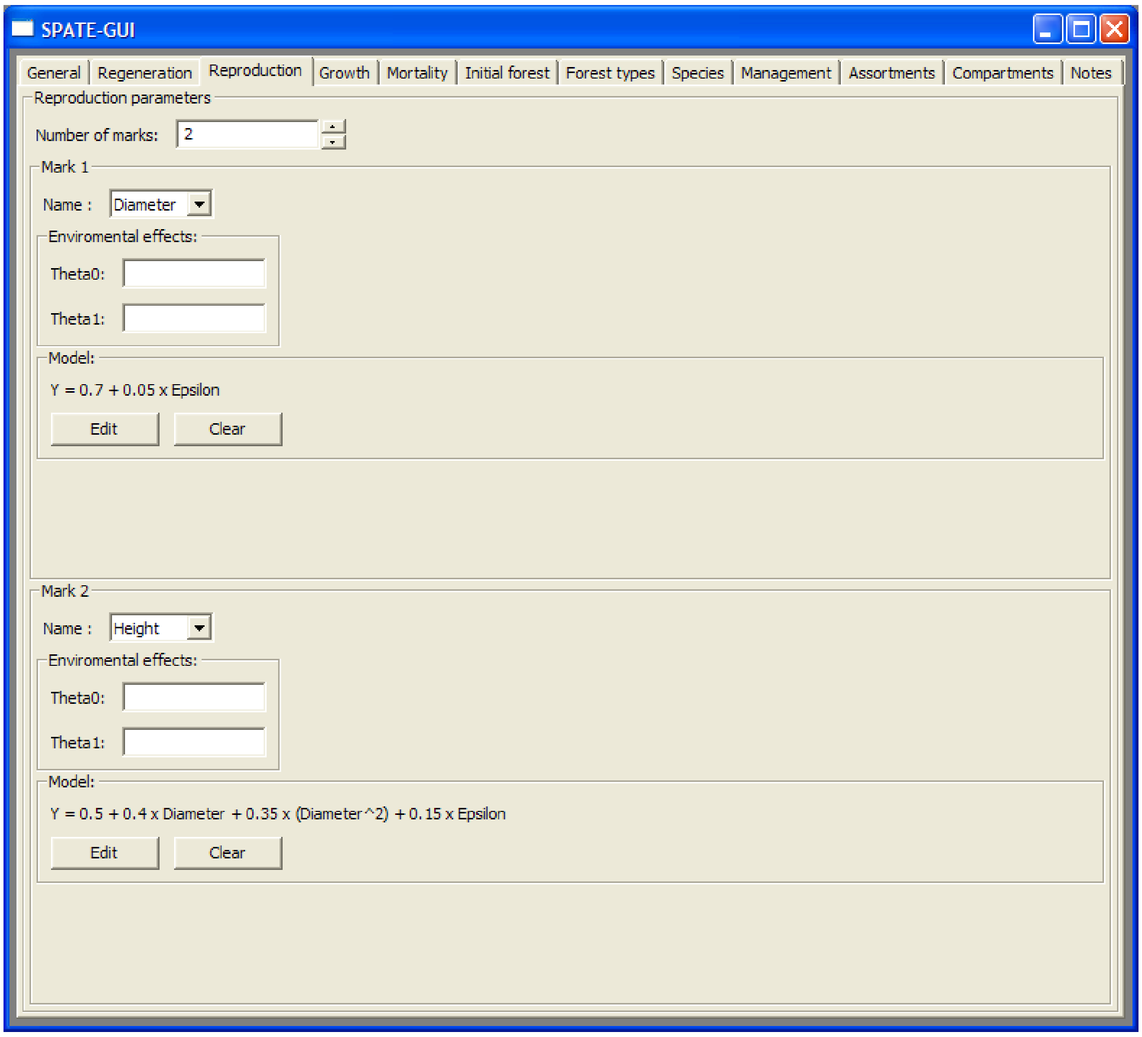
The reproduction model is described in section 3.5.4, the growth model is described in section 3.5.2 and the mortality model is described in section 3.5.3.

4.4 Editing a model

Models are edited with the model editor dialog, which is brought up by clicking the ‘Edit’ button within a mark model panel. The user is first presented the main part of the model which corresponds to equation (3.2). A screenshot of the window for editing the main part of a model is shown in figure 4.4.

The terms of the model are listed vertically where each term comprises a smoothing function coefficient represented by a ‘Beta’ label and a set of variables. Here, the ‘Beta’ coefficients correspond to β and the variables to x in equation (3.2). New model terms are added by clicking the ‘Add model term’ button and existing terms are removed by clicking the ‘X’ button at the right end of the term’s row. Furthermore, new variables are added to a term by clicking the ‘Add variable’ button and existing variables are removed by selecting the ‘Delete variable’ option from the variable’s drop-down menu. Naturally, the same variable can be used several times in different configurations and with different exponents.

Double-clicking on a ‘Beta’ label brings up a window for editing the corresponding secondary-level model, or smoothing function. A screenshot of this window with a sample secondary-level



Reproduction parameters

Number of marks: 2

Mark 1

Name : Diameter

Enviromental effects:

Theta0:

Theta1:

Model:

Y = 0.7 + 0.05 x Epsilon

Mark 2

Name : Height

Enviromental effects:

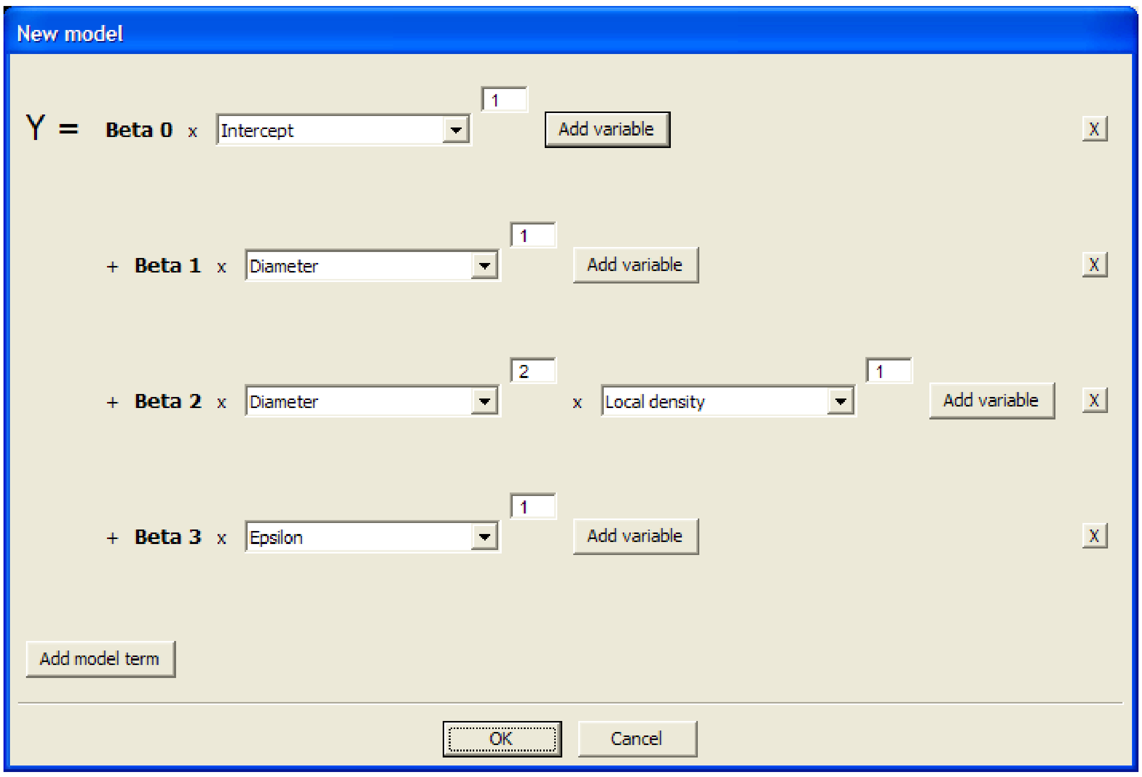
Theta0:

Theta1:

Model:

Y = 0.5 + 0.4 x Diameter + 0.35 x (Diameter^2) + 0.15 x Epsilon

Figure 4.3: Page showing the reproduction models.


New model

Y = **Beta 0** x Intercept 1 Add variable X

+ **Beta 1** x Diameter 1 Add variable X

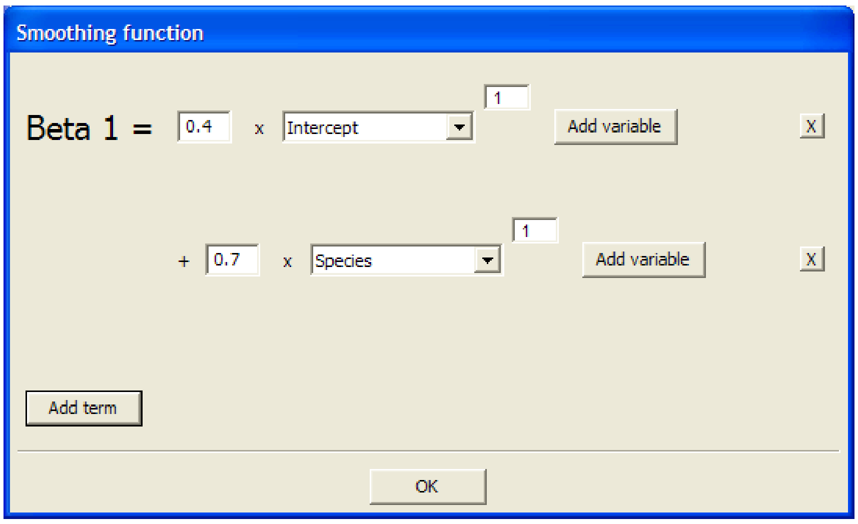
+ **Beta 2** x Diameter 2 x Local density 1 Add variable X

+ **Beta 3** x Epsilon 1 Add variable X

Add model term

OK Cancel

Figure 4.4: The model editor showing the model $Y = \beta_0 + \beta_1 \cdot Diameter + \beta_2 \cdot Diameter^2 \cdot Local\ density + \beta_3 \cdot Epsilon$. The coefficients ‘Beta 0’ through ‘Beta 3’ are shown in bold font, which indicates that they have been assigned secondary-level models.


Smoothing function

Beta 1 = 0.4 x Intercept 1 Add variable X

+ 0.7 x Species 1 Add variable X

Add term

OK

Figure 4.5: The smoothing function editor window showing the secondary-level model $\beta_1 = 0.4 + 0.7 \cdot Species$

model is shown in figure 4.5. Analogously to the main model window, the terms of the secondary-level model are listed vertically, where each term comprises a coefficient and a set of variables. The coefficients correspond to β^* and the variables to w in equation (3.3). The procedures for adding and removing variables and adding and removing terms are identical to the ones used in the main model window.

When a new model is created from scratch or when new model terms are added to an existing model, the model terms' smoothing functions will be uninitialized. This is indicated by the font of the 'Beta' label; a normal font means the function is uninitialized whereas a bold font means the function has been initialized. In order to be able to save the model all smoothing functions must be properly initialized.

4.5 Initial forest page

The settings for creating an initial stand are located on the 'Initial forest' page, see figure 4.6. The user may choose to load the initial stand from a tree data file or generate it. Selecting a tree data file is done by browsing for it using the 'Load initial forest data from file' input field. The tree data file should be in the file format described in section 5.5.

If the trees are to be generated one of the options 'Homogeneous Poisson', 'Inhomogeneous Poisson', 'Homogeneous clustered', 'Inhomogeneous clustered', 'Homogeneous regular' or 'Inhomogeneous regular' should be selected. The selected option corresponds to the point process used to place the trees. Moreover, one should choose an average age of the trees in the 'Average initial age' input field and an average number of trees per hectare in the 'Expected number of initial points per hectare' input field. The average initial age and trees per hectare parameters are common to the whole simulation, but they can be readjusted for different compartments by specifying compartment-specific factors.

The diameter and height marks of initial trees are generated according to the models specified on this page. This is done by setting the 'Number of marks' input field and adding models according the description in section 4.4.

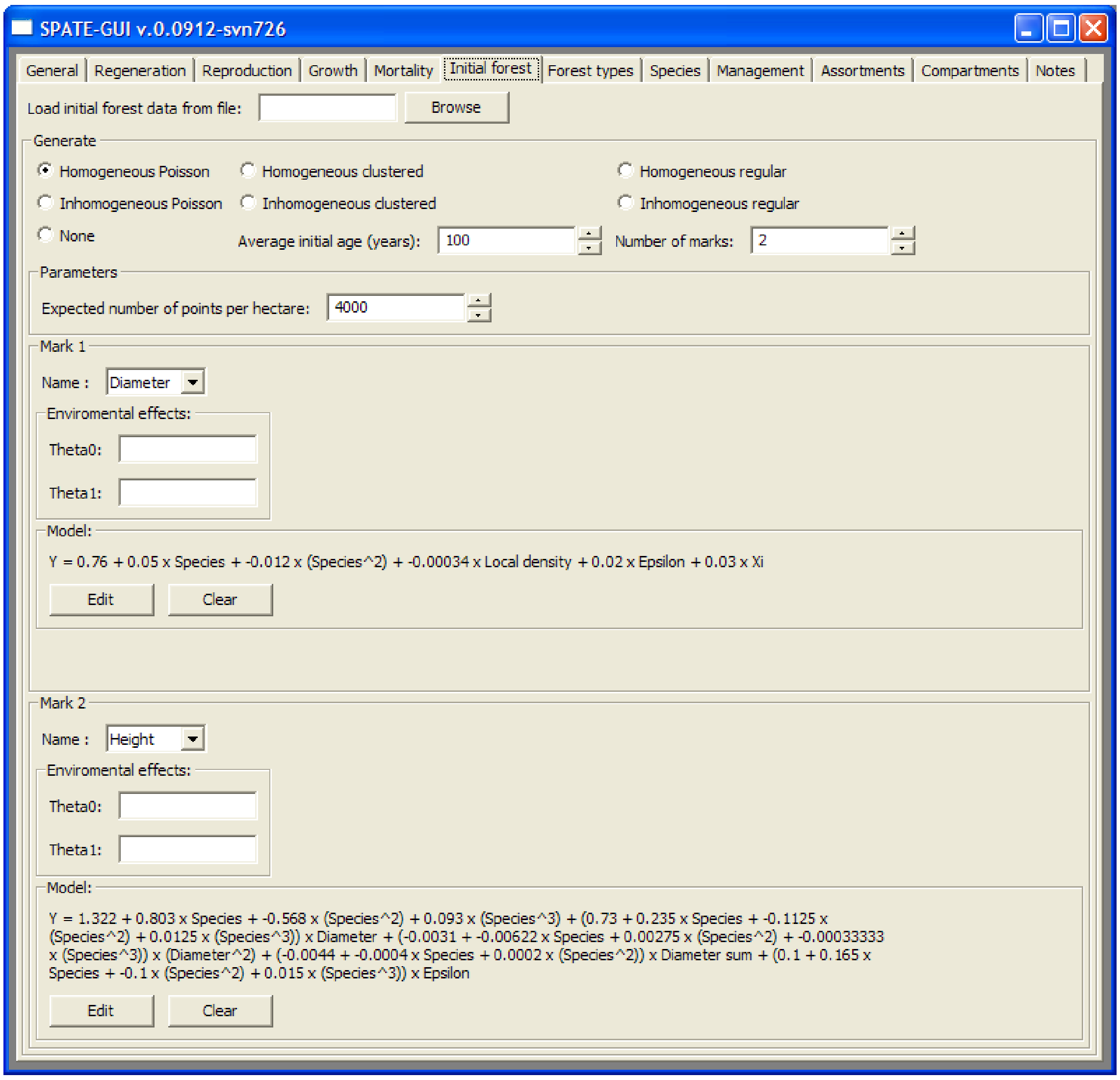
The mechanisms for generating the initial stand are described in section 3.5.1.

4.6 Forest type page

Forest types are described in section 3.8. A forest type is a composition of different tree species and there must be at least one forest type. Forest types are edited on the 'Forest types' page, see figure 4.7. There must be at least one species defined in order for the forest types page to be usable.

On the left side of the window there is a list of the forest types that are currently defined. Selecting a forest type in the list brings up its settings on the right side of the window. A new forest type can be created by clicking 'New' and an existing forest type removed by clicking 'Delete'.

A forest type must be given a unique name using the 'Name' input field. The name is used to refer to the forest type from a compartment. The species that the forest type should contain are chosen by adding them to the 'Species' list. The relative amount of trees of the species with respect to other species is specified by adjusting the 'Proportion' parameter. The proportion does not need to be a percentage; the species proportions are only compared to each other relatively. If one wants


SPATE-GUI v.0.0912-svn726
General | Regeneration | Reproduction | Growth | Mortality | Initial forest | Forest types | Species | Management | Assortments | Compartments | Notes
Browse
Edit
Clear

Load initial forest data from file:

Generate

[✓ Homogeneous Poisson] Homogeneous clustered Homogeneous regular

Inhomogeneous Poisson Inhomogeneous clustered Inhomogeneous regular

None

Average initial age (years): 100

Number of marks: 2

Parameters

Expected number of points per hectare: 4000

Mark 1

Name : Diameter

Enviromental effects:

Theta0:

Theta1:

Model:

Y = 0.76 + 0.05 x Species + -0.012 x (Species^2) + -0.00034 x Local density + 0.02 x Epsilon + 0.03 x Xi

Mark 2

Name : Height

Enviromental effects:

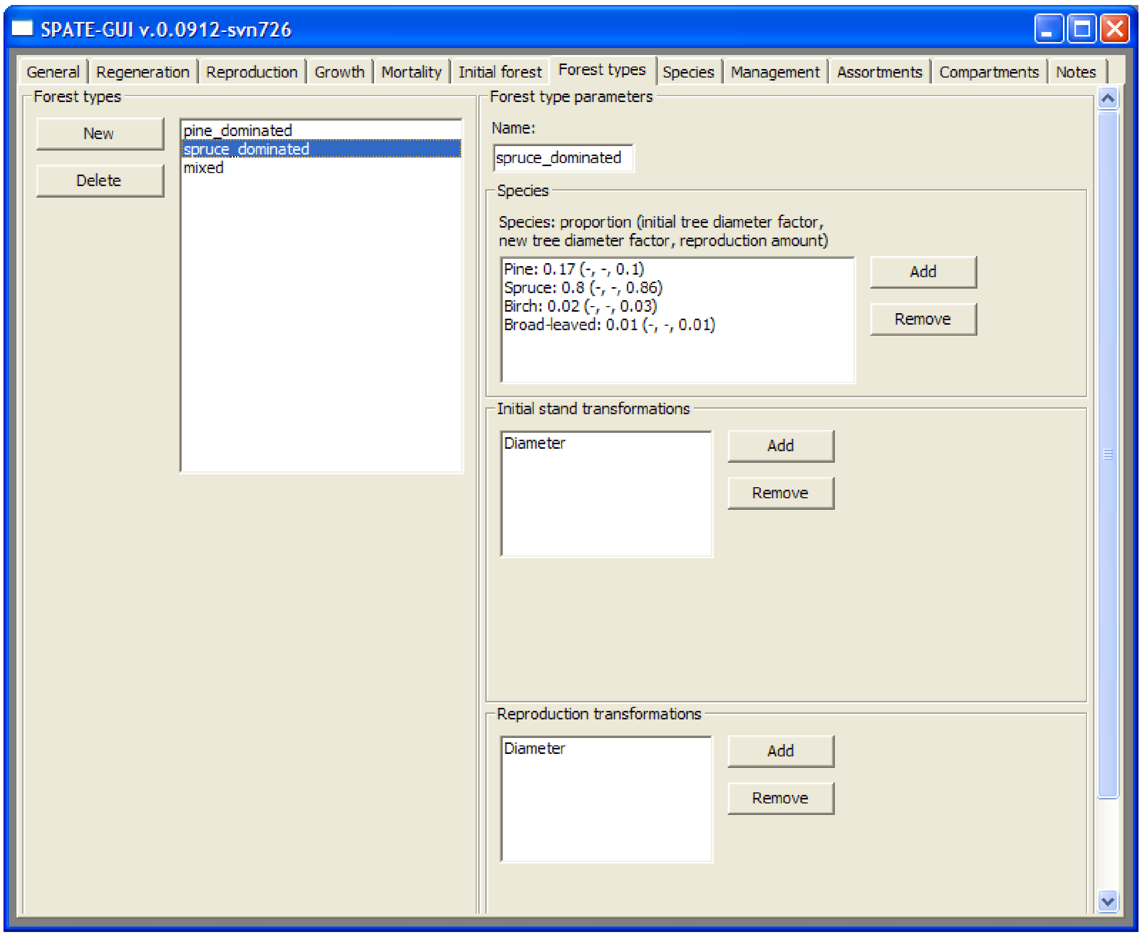
Theta0:

Theta1:

Model:

Y = 1.322 + 0.803 x Species + -0.568 x (Species^2) + 0.093 x (Species^3) + (0.73 + 0.235 x Species + -0.1125 x (Species^2) + 0.0125 x (Species^3)) x Diameter + (-0.0031 + -0.00622 x Species + 0.00275 x (Species^2) + -0.00033333 x (Species^3)) x (Diameter^2) + (-0.0044 + -0.0004 x Species + 0.0002 x (Species^2)) x Diameter sum + (0.1 + 0.165 x Species + -0.1 x (Species^2) + 0.015 x (Species^3)) x Epsilon

Figure 4.6: The 'Initial forest' page showing example settings for generating an initial stand.



Forest types

pine_dominated
spruce_dominated
mixed

Forest type parameters

Name: spruce_dominated

Species

Species: proportion (initial tree diameter factor, new tree diameter factor, reproduction amount)

Pine: 0.17 (-, -, 0.1)
Spruce: 0.8 (-, -, 0.86)
Birch: 0.02 (-, -, 0.03)
Broad-leaved: 0.01 (-, -, 0.01)

Initial stand transformations

Diameter

Reproduction transformations

Diameter

Figure 4.7: The ‘Forest types’ page showing the forest type “spruce_dominated”, which contains 80% spruce and has diameter transformations defined for both the initial stand and the reproduction phase.

trees of the species to have a smaller or larger diameter than the overall stand, one may enter a factor into the input fields ‘Initial tree diameter factor’ and ‘New tree diameter factor’ for trees in the initial stand and new trees in the reproduction phase, respectively. Leaving the fields empty is the same as entering the number “1”. Furthermore, the tree proportion in the reproduction phase can be separate from the one in the initial stand. The reproduction proportion is specified using the ‘Reproduction amount’ input field. Leaving this field empty means that the same proportion is used both in the initial stand and the reproduction phase.

If the generated tree marks in the initial stand and the reproduction phase are to be transformed to adhere to an arbitrary distribution, this is specified in the ‘Initial stand transformations’ and ‘Reproduction transformations’ panels. There can be one transformation for each mark (diameter and height), and they are specified by clicking the ‘Add’ button. The user may select one of the options ‘Weibull distribution’, ‘Normal distribution’, ‘Empirical data’ or ‘Empirical cumulative distribution’. If ‘Empirical data’ is selected, one should provide a file of the format described in section 5.3, and if ‘Empirical cumulative distribution’ is selected, one should provide a file of the format described in section 5.4. The process of transformation marks is explained in section 3.6.

4.7 Species page

Tree species are edited on the ‘Species’ page, see figure 4.8. There must be at least one species defined, and species must be defined before defining forest types. Species are described in section 3.7.

On the left side of the window there is a list of the species that are currently defined. Selecting a species in the list brings up its settings on the right side of the window. A new species can be created by clicking ‘New’ and an existing species removed by clicking ‘Delete’.

The following is a list of the settings that are configurable for a species:

Name The species must have a unique name, which is used to refer to the species from a forest type.

Secondary model value The value used for the variable “Species” in the secondary-level model, see section 3.4.1.

Small volume coefficient If a tree is too small for taper curves and a cylinder model is used instead, this coefficient is used as a weight.

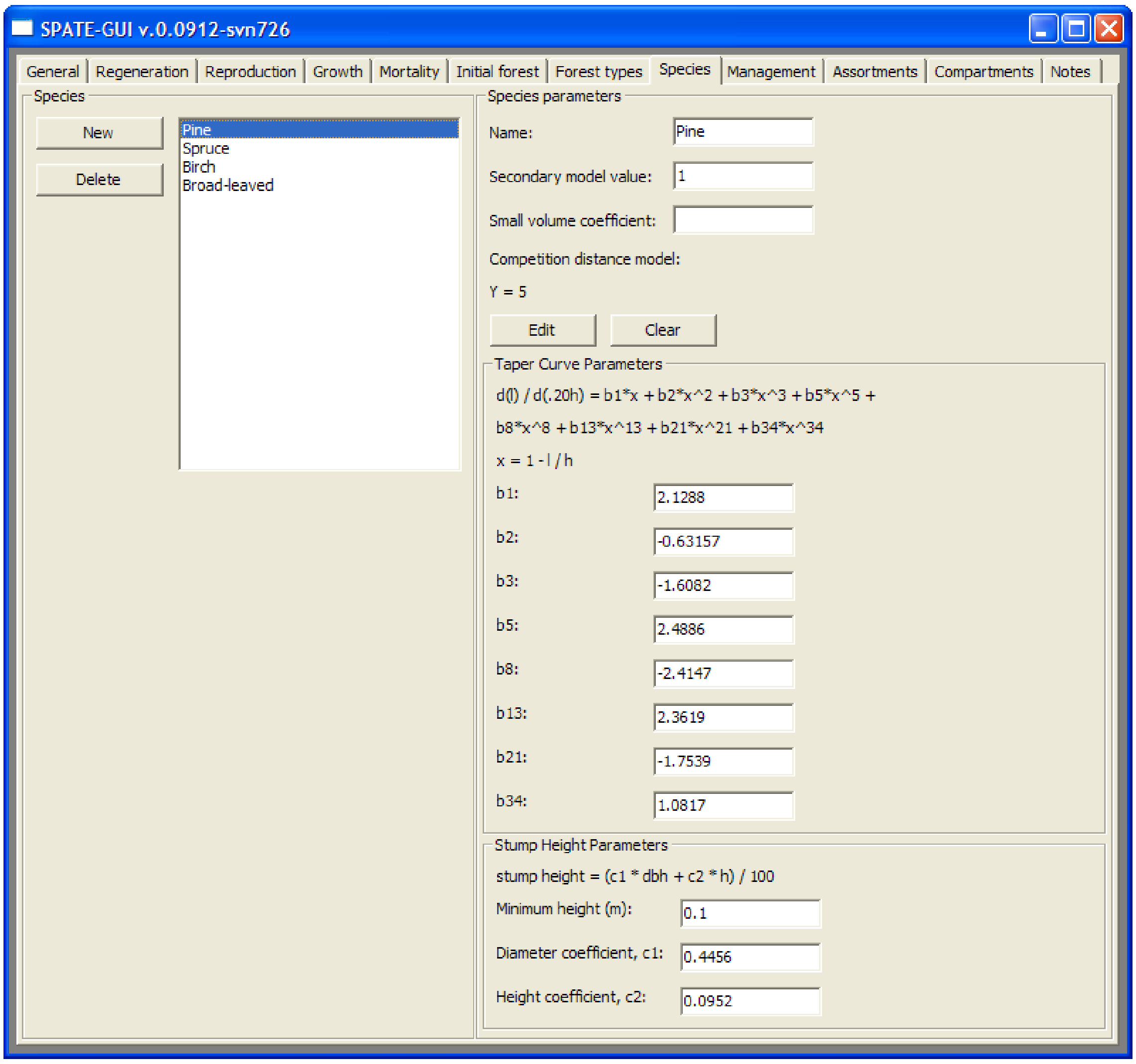
Competition distance model A model for calculating the competition distance.

Taper Curve Parameters Parameters for the taper curve functions used to estimate a trees volume.

Stump Height Parameters Parameters for the stump height model.

4.8 Management

Management templates are edited on the ‘Management’ page, see figure 4.9. Management templates and methods are described in section 3.9.


SPATE-GUI v.0.0912-svn726

General | Regeneration | Reproduction | Growth | Mortality | Initial forest | Forest types | Species | Management | Assortments | Compartments | Notes

Species

New

Delete

Pine
Spruce
Birch
Broad-leaved

Species parameters

Name: Pine

Secondary model value: 1

Small volume coefficient:

Competition distance model:

Y = 5

Edit Clear

Taper Curve Parameters

d(l) / d(.20h) = b1*x + b2*x^2 + b3*x^3 + b5*x^5 +

b8*x^8 + b13*x^13 + b21*x^21 + b34*x^34

x = 1 - l / h

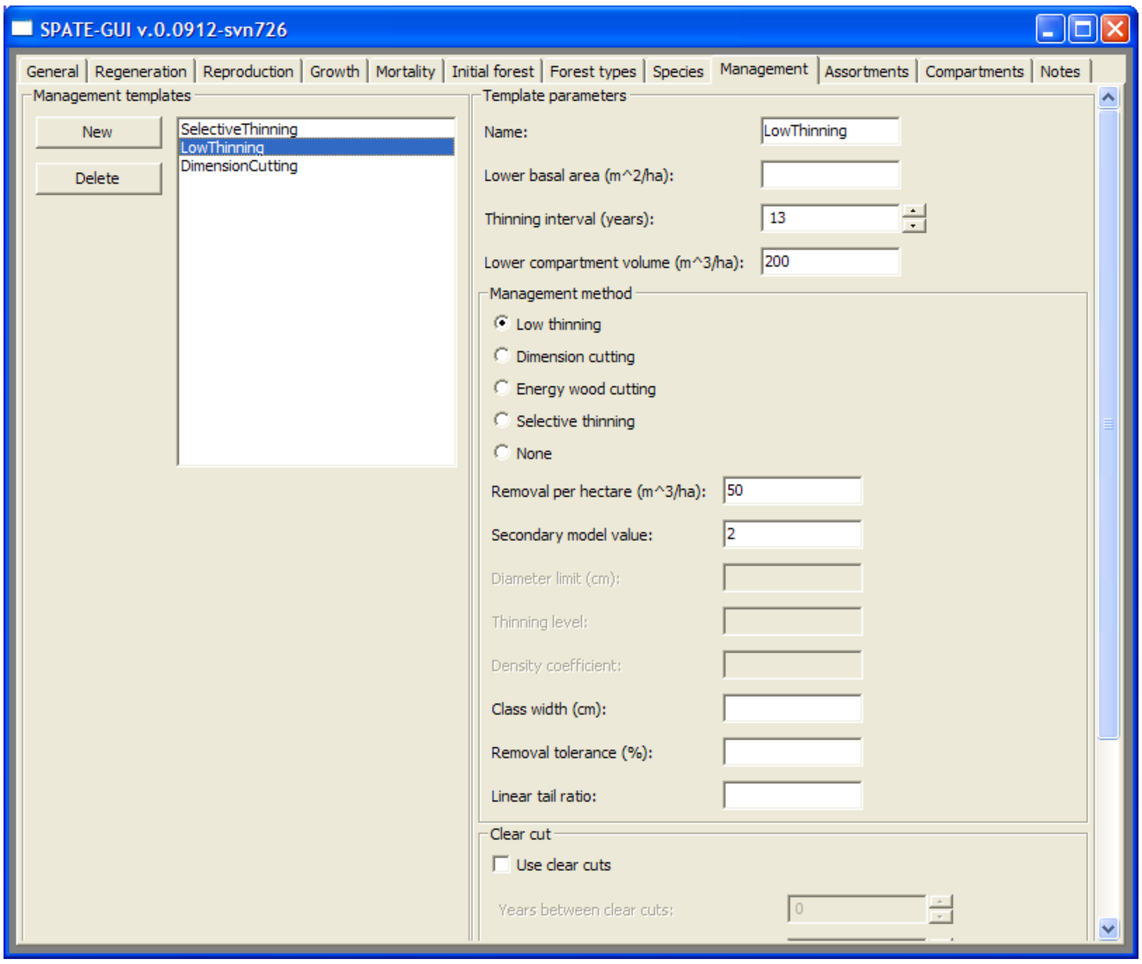
b1:	2.1288
b2:	-0.63157
b3:	-1.6082
b5:	2.4886
b8:	-2.4147
b13:	2.3619
b21:	-1.7539
b34:	1.0817

Stump Height Parameters

stump height = (c1 * dbh + c2 * h) / 100

Minimum height (m):	0.1
Diameter coefficient, c1:	0.4456
Height coefficient, c2:	0.0952

Figure 4.8: The 'Species' page. Here, the species "Pine" is selected and its settings are shown in the panel to the right.


SPATE-GUI v.0.0912-svn726

Management templates

- SelectiveThinning
- LowThinning
- DimensionCutting

Template parameters

Name:	LowThinning
Lower basal area (m^2/ha):	
Thinning interval (years):	13
Lower compartment volume (m^3/ha):	200

Management method

- [✓ Low thinning]
- Dimension cutting
- Energy wood cutting
- Selective thinning
- None

Removal per hectare (m^3/ha):	50
Secondary model value:	2
Diameter limit (cm):	
Thinning level:	
Density coefficient:	
Class width (cm):	
Removal tolerance (%):	
Linear tail ratio:	

Clear cut

- Use clear cuts

Years between clear cuts:	0

Figure 4.9: Management templates are defined on the ‘Management’ page. The selected template “LowThinning” is set to carry out low thinning removing 50m^3/ha every 13th year if the compartment volume exceeds 200m^3/ha.

On the left side of the window there is a list of the management templates that are currently defined. Selecting a management template in the list brings up its settings on the right side of the window. A new management template can be created by clicking ‘New’ and an existing management template removed by clicking ‘Delete’.

The following is a list of the settings that are configurable for a management template:

Name The management template must have a unique name, which is used to refer to the template from a compartment.

Lower basal area If specified, the management is carried out when the compartment’s basal area per hectare exceeds this parameter value.

Thinning interval Two consecutive managements cannot be carried out closer to each other than this number of years.

Lower compartment volume If specified, the management is carried out when the compartment’s total volume per hectare exceeds this parameter value.

Removal per hectare If low or selective thinning is used, this is the target volume to remove.

Secondary model value The value used for the variable “Management” in the secondary-level model, see section 3.4.1.

Diameter limit If dimension or energy wood cutting is used, trees with larger or smaller diameter than this value, respectively, are cut.

Thinning level The parameter β_0 in formula (3.12) when selective thinning is used.

Density coefficient The parameter β_1 in formula (3.12) when selective thinning is used.

Class width The width of size classes. (The default is 4cm.)

Removal tolerance The percentage the resulting removal volume in low and selective thinning may deviate from the target volume.

Linear tail ratio If specified, the tail of the Weibull curve in low or selective thinning is linearized according to this ratio, see section 3.9.2.

Use clear cuts Whether to carry out clear cuts.

Years between clear cuts The number of years between to consecutive clear cuts.

Trees left per hectare The number of trees per hectare to spare after a clear cut.

Forced tree distance Spared trees may stand no closer than this distance.

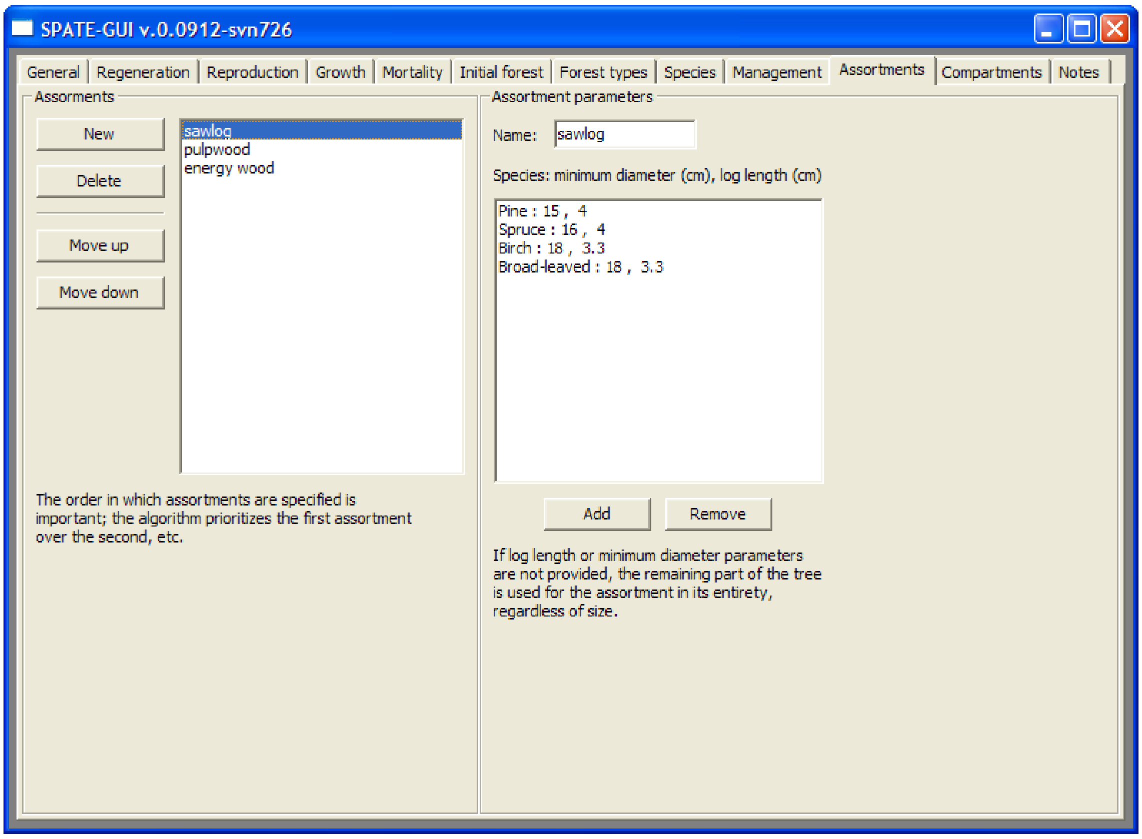
Class width The size class width. (The default is 4cm.)

Minimum diameter of spared trees Spared trees may not have a diameter smaller than this value.

Minimum diameter of spared trees Spared trees may not have a diameter larger than this value.

4.9 Assortments page

Assortments are edited on the ‘Assortments’ page, see figure 4.10. If assortments are specified, volume statistics on trees removed and on the growing stock for the different assortments are produced after a simulation has finished. Assortments are described in section 3.10.



sawlog
pulpwood
energy wood

The order in which assortments are specified is important; the algorithm prioritizes the first assortment over the second, etc.

Name: sawlog

Species: minimum diameter (cm), log length (cm)

```
Pine : 15 ,  4
Spruce : 16 ,  4
Birch : 18 ,  3.3
Broad-leaved : 18 ,  3.3
```

If log length or minimum diameter parameters are not provided, the remaining part of the tree is used for the assortment in its entirety, regardless of size.

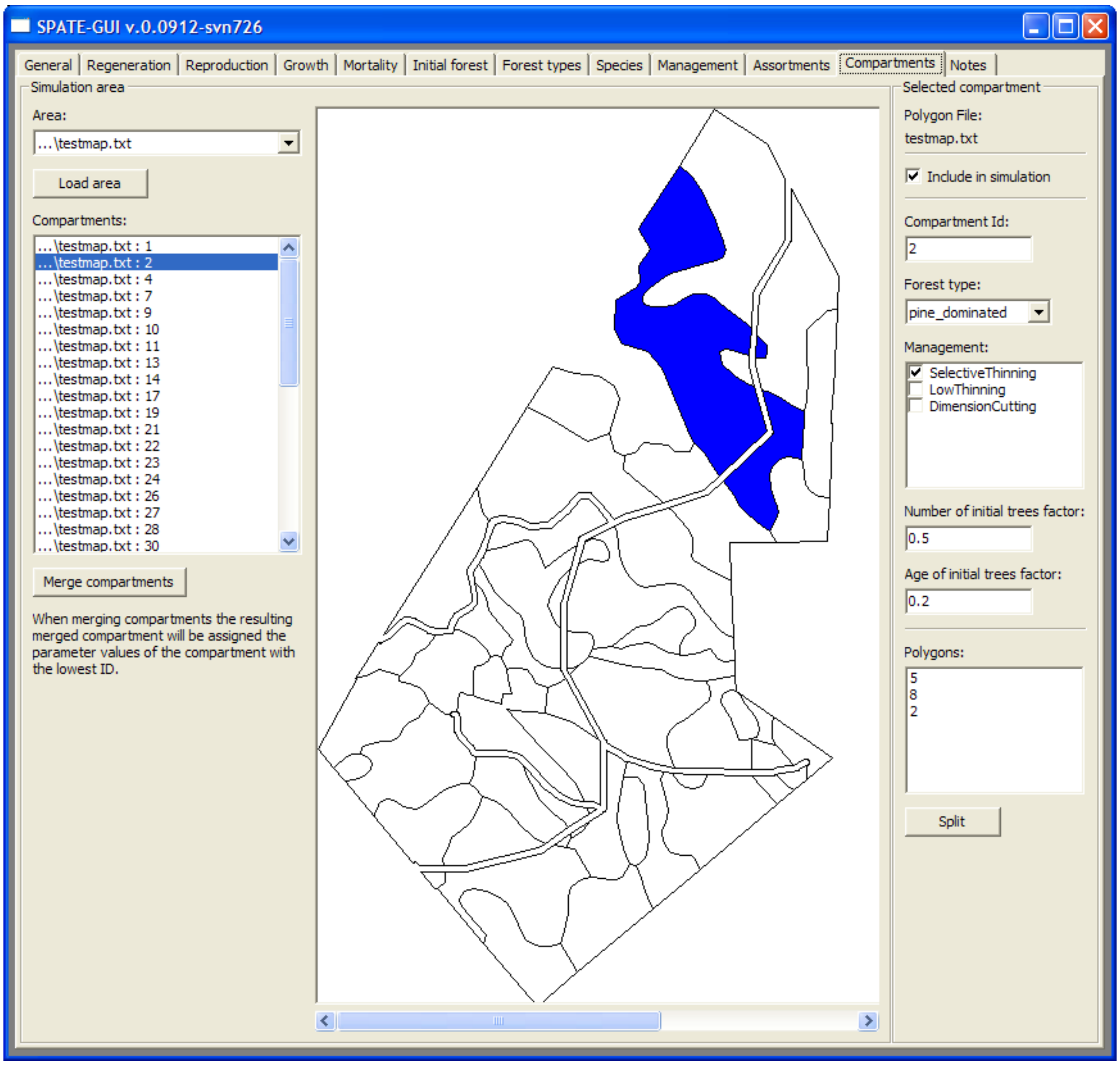
Figure 4.10: Assortments are defined on the ‘Assortments’ page. Here, the assortment “sawlog” is selected and its size criteria for different species are shown in the panel to the right.

On the left side of the window there is a list of the assortments that are currently defined. Selecting an assortment in the list brings up its settings on the right side of the window. A new assortment can be created by clicking ‘New’ and an existing assortment removed by clicking ‘Delete’.

An assortment must be given a unique name using the ‘Name’ input field and a set of size criteria. Species-specific size criteria are defined by clicking the ‘Add’ button. The size criteria that can be given are ‘Minimum diameter’ and ‘Log length’. If these fields are left empty no size criteria are enforced and the assortment can be used for the top of the tree. One can specify the species for which the criteria apply by selecting it in the list. If no species is selected the size criteria will apply to all species.

4.10 Compartments page

Compartments are set up on the ‘Compartments’ page, see figure 4.11. There must be at least one compartment defined in order to run the simulation. Compartments are described in section 3.3. First of all a map must be loaded. This is done by clicking the ‘Load area’ button and selecting a polygon file that conforms to the specification in section 5.2. Upon successfully loading a polygon file the area will be presented in the main part of the window, where polygons are enclosed with black lines. The compartments are also listed in the left part of the window.



Area:
...\testmap.txt

Compartments:
...\testmap.txt : 1
...\testmap.txt : 2
...\testmap.txt : 4
...\testmap.txt : 7
...\testmap.txt : 9
...\testmap.txt : 10
...\testmap.txt : 11
...\testmap.txt : 13
...\testmap.txt : 14
...\testmap.txt : 17
...\testmap.txt : 19
...\testmap.txt : 21
...\testmap.txt : 22
...\testmap.txt : 23
...\testmap.txt : 24
...\testmap.txt : 26
...\testmap.txt : 27
...\testmap.txt : 28
...\testmap.txt : 30

When merging compartments the resulting merged compartment will be assigned the parameter values of the compartment with the lowest ID.

Polygon File:
testmap.txt

[✓] Include in simulation

Compartment Id:
2

Forest type:
pine_dominated

Management:
[✓] SelectiveThinning
[ ] LowThinning
[ ] DimensionCutting

Number of initial trees factor:
0.5

Age of initial trees factor:
0.2

Polygons:
5
8
2

Figure 4.11: The compartments page showing the forest area. A compartment consisting of three polygons is selected.

Selecting a compartment from the list or clicking on the corresponding polygon in the area view will show its settings in the right part of the window. A compartment has the following configurable settings:

Include in simulation Whether the compartment should be included in the simulation.

Compartment id Each compartment should be given a unique ID number. If not explicitly changed the default is the polygon ID.

Forest type The forest type of the compartment.

Management Here, the management templates that should be implemented in the compartment can be selected.

Number of initial trees factor If specified, the number of trees per hectare in the initial stand is weighted with this factor.

Age of initial trees factor If specified, the age of trees in the initial stand is weighted with this factor.

Multiple compartments can be combined to form a multi-polygon compartment. This is done by selecting multiple compartments either in the compartment list or in the area view and clicking the 'Merge compartments' button. A multi-polygon compartment can subsequently be split by selecting a polygon in the polygon list and clicking the 'Split' button.

4.11 Notes page

The 'Notes' page contains a text box into which the user may enter arbitrary notes. The notes have not effect on the simulation and can only be used for attaching explanatory text to the XML input file.

Chapter 5

File formats

This chapter describes the file formats of the files used as input and output of SPATE-HPC and SPATE-GUI.

5.1 Parameter file

The parameter file, which is the main input file of SPATE-HPC, is in the extensible markup language (XML) format[1]. In this section the possible element, attributes and values that can be used in the XML input file are described.

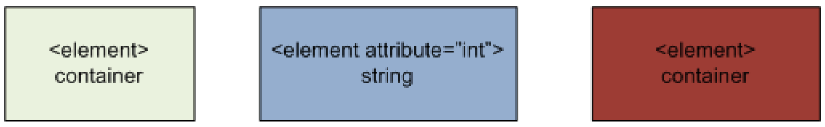


Figure 5.1: XML elements are illustrated with these types of boxes.

XML element are illustrated with boxes connected to each other with arrows. There are three color codes for the elements: light green, blue and maroon (see figure 5.2). A light green element represents a container element. It does not contain a value itself but is merely a container for other elements. A blue element contains a value. The type of the value is written on the last line within the box. Maroon elements are mutually exclusive; when a set of child elements have a maroon indicator color, only one of the siblings may be chosen. Maroon elements are either container or value elements. Elements may also have attributes. This is indicated by a "name='value type'" string within the element box. See the blue element in figure 5.2.

The vertical structure of the XML file is illustrated with arrows. An arrow from one element to another denotes that the latter element is a child of the former. There is also a cardinality label accompanying the arrowhead, which states how many instances of the child element that may occur.

[1] See http://www.w3.org/XML/ for the XML specification.

5.1.1 Simulation

The root element of the XML file is named “simulation”, see figure 5.2. Its child elements are: *note*, *inspectionParameters*, *generalParameters*, *speciesParameters*, *forestTypeParameters*, *managementParameters*, *assortmentParameters* and *compartmentParameters*. The element *note* is ignored by the simulator. It can be used by the user to attach explanatory notes to the input file. The other elements contain parameters which are described in the following sections.

5.1.2 General parameters

The *generalParameters* element contains assorted simulation parameters, see figure 5.3. Its child elements containing values are the following:

name Name of the simulation.

numberOfyears The number of years that should be simulated. The value can be zero, whereupon only the initial stand is generated.

iterationLength The length of one iteration step in number of years.

numberOfReplications The number of replications (identical simulations) to perform.

rngSeed A number, preferably very large, to seed the random number generator with.

saveTreeFiles The value of the element, which can be either “0” or “1”, indicates whether tree data files should be produced after the simulation has completed. Additionally, if the “iteration” attribute with the value “1” is given, tree data files are produced at every intermediate iteration step.

iterationLoadBalancing The value of the element, which can be either “0” or “1”, indicates whether the simulator should rebalance trees between processors between iterations. The element takes the attribute *imbalancePct* whose value is an integer greater than or equal to zero and represents the maximum allowed imbalance percentage.

outputDirectory If given, the value of this element is the name of the directory that should be used for all output files. The name may contain the character “%t”, which is replaced by the current time stamp.

standardNormalLimitingFactor Normally distributed random numbers in some parts of the program are limited to a support defined by this factor. The support is the value of this element multiplied by the standard deviation.

Size classes

The *sizeClasses* element contains a list of diameter size classes, defined by *sizeClass* elements. Each *sizeClass* element should contain a *name* and a *minimumDiameter* element. *sizeClass* elements must be ordered according to minimum diameter, and there must be at least one *sizeClass* element within *sizeClasses*.

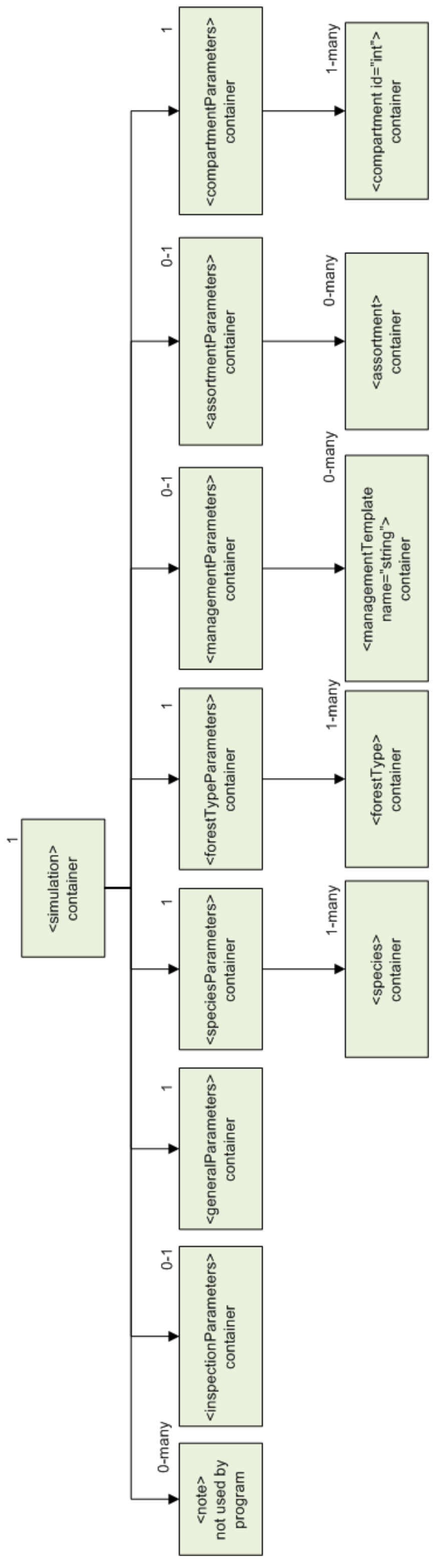


Figure 5.2: The root element, "simulation", and its direct children.

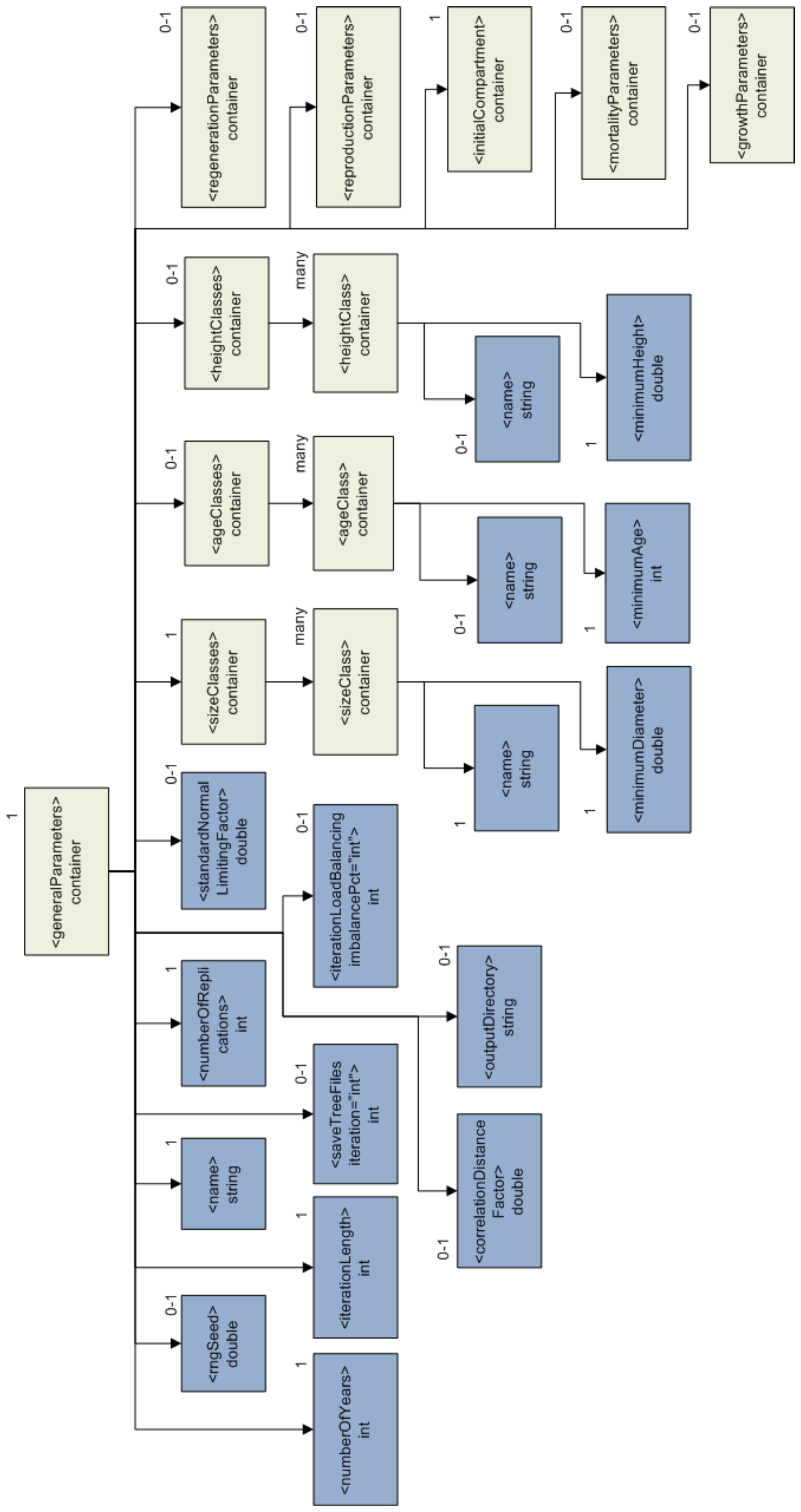

<generalParameters> container
1
<regenerationParameters> container
0-1
<reproductionParameters> container
0-1
<initialCompartment> container
1
<mortalityParameters> container
0-1
<growthParameters> container
0-1
<heightClasses> container
0-1
<heightClass> container
many
<name> string
0-1
<minimumHeight> double
1
<ageClasses> container
0-1
<ageClass> container
many
<name> string
0-1
<minimumAge> int
1
<sizeClasses> container
1
<sizeClass> container
many
<name> string
1
<minimumDiameter> double
1
<standardNormal LimitingFactor> double
0-1
<iterationLoadBalancing imbalancePct="int"> int
0-1
<outputDirectory> string
0-1
<numberOfRepli cations> int
1
<saveTreeFiles iteration="int"> int
0-1
<correlationDistance Factor> double
0-1
<name> string
1
<iterationLength> int
1
<rngSeed> double
0-1
<numberOfYears> int
1

Figure 5.3: The *generalParameters* element contains assorted simulation parameters.

Age classes

The *ageClasses* element contains a list of age classes, defined by *ageClass* elements. *ageClasses* is similar to *sizeClasses*, but with more lax cardinalities. Each *ageClass* element may contain a *name* and should contain a *minimumAge* element. *ageClass* elements must be ordered according to the minimum age.

Height classes

The *heightClasses* element contains a list of height classes, defined by *heightClass* elements. *heightClasses* is similar to *sizeClasses*, but with more lax cardinalities. Each *heightClass* element may contain a *name* and should contain a *minimumHeight* element. *heightClass* elements must be ordered according to the minimum height.

Growth, mortality and reproduction parameters

The models for growth, mortality and reproduction are given within the *growthParameters*, *mortalityParameters* and *reproductionParameters* elements, respectively. The corresponding parameters element may contain zero, one or two *mark* elements. The content of a *mark* element represents the growth, mortality or reproduction model for one of the marks (diameter or height). A *mark* element must have a *name* attribute with a value of either “Diameter” or “Height”. No other name attribute values are allowed, and there must be no two sibling mark elements with the same name, see figure 5.4.

A *mark* element contains a set of *modelTerm* elements. The *modelTerm* elements are described in section 5.1.3. Furthermore, if environmental effects are used in the model a *mark* element should have a *environmentalEffects* child element. The *environmentalEffects* element may have two *coefficient* child elements whose values correspond to θ_1 and θ_2 in formula (3.4); the first *coefficient* element corresponds to θ_1 and the second one to θ_2. If either one or both coefficients are left out, their default value is zero.

In the *mortalityParameters* element, a *mark* element may contain an additional element called *regularMortalityThreshold*. The value of *regularMortalityThreshold* corresponds to the regular mortality threshold described in section 3.5.3. If *regularMortalityThreshold* is not given the default value is 1.0.

If one of the *growthParameters*, *mortalityParameters* or *reproductionParameters* container elements is missing, or if it exists but contains no *mark* elements, the corresponding phase will not be carried out in the simulation. In other words, it is possible to disable a particular simulation phase by not including the corresponding parameters element in the XML input file.

Regeneration parameters

The *regenerationParameters* element contains the model for calculating regeneration, i.e., the number of new trees in a reproduction phase. The *regenerationParameters* container element may contain a set of *variable* elements, which each has a *name* attribute and a value. The value of the name attribute must be one of the variable names listed in section 3.5.4, and there must be no two *variable* elements with the same name. The value of the *variable* element corresponds to the variable’s coefficient, see figure 5.4.

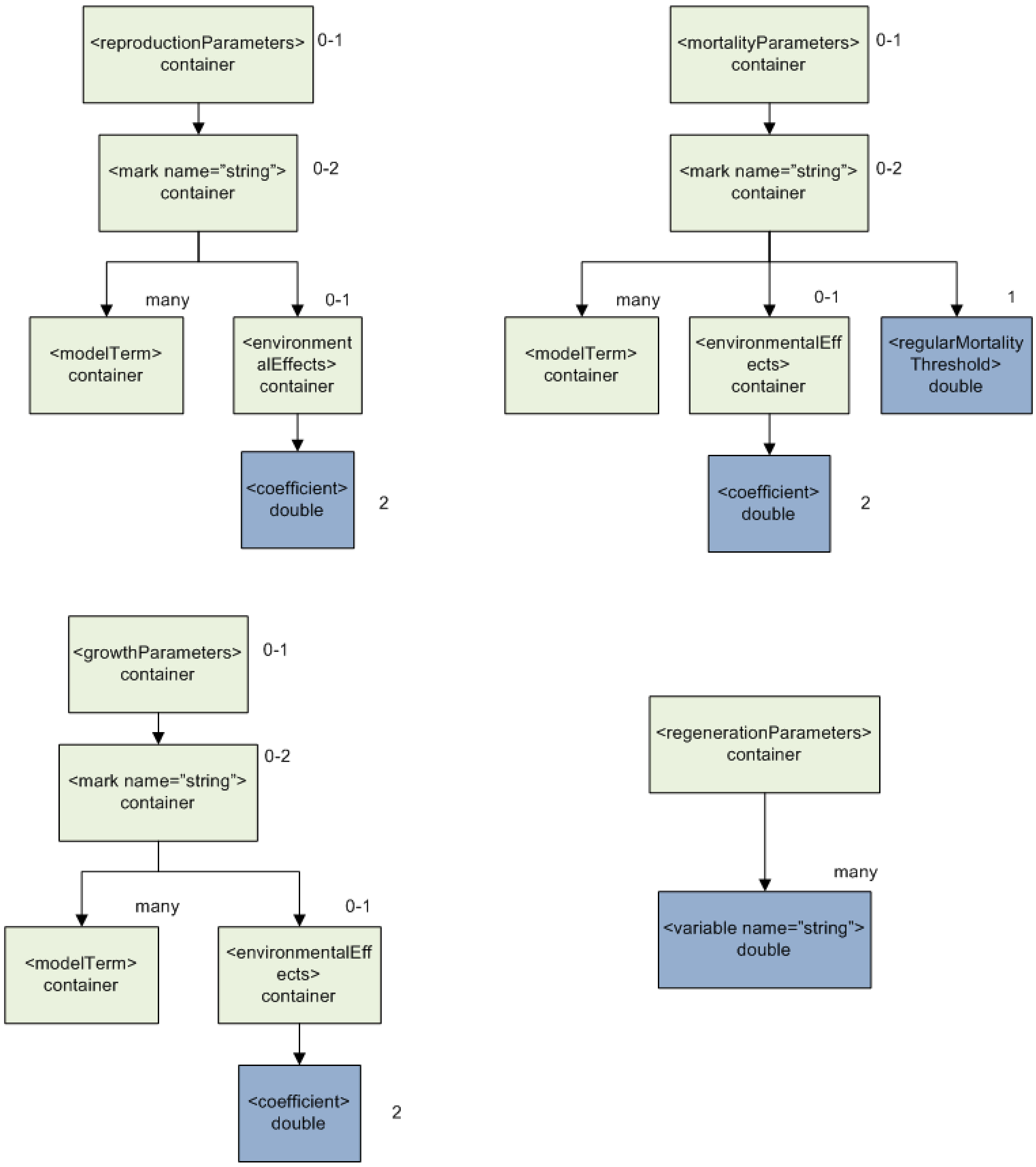

<reproductionParameters> container
0-1
<mark name="string"> container
0-2
many
<modelTerm> container
0-1
<environmentalEffects> container
<coefficient> double
2
<mortalityParameters> container
0-1
<mark name="string"> container
0-2
many
<modelTerm> container
0-1
<environmentalEffects> container
1
<regularMortalityThreshold> double
<coefficient> double
2
<growthParameters> container
0-1
<mark name="string"> container
0-2
many
<modelTerm> container
0-1
<environmentalEffects> container
<coefficient> double
2
<regenerationParameters> container
many
<variable name="string"> double

Figure 5.4: Parameters governing growth, mortality, reproduction and regeneration.

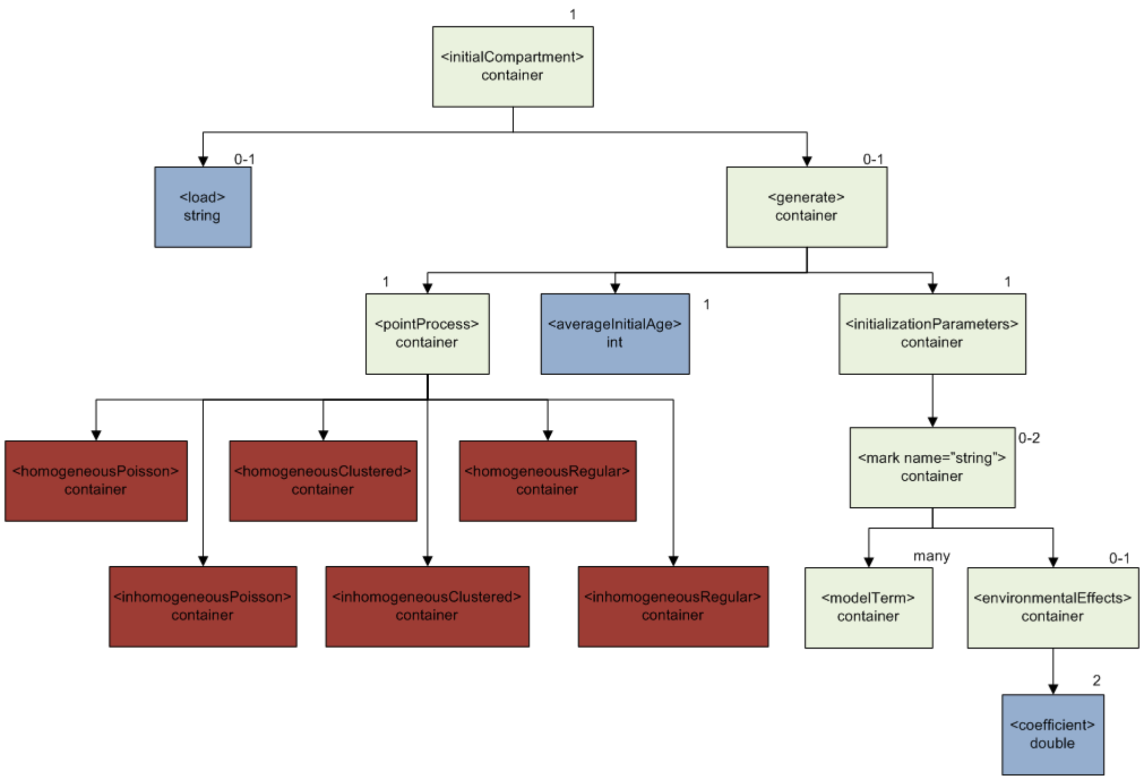


Figure 5.5: *initialCompartment* contains parameters for generating or loading trees for the initial stand.

Initial compartment

The *initialCompartment* element contains parameters for generating trees for the initial stand, see figure 5.5. The element should contain one of the elements *load* or *generate*, or both. If a *load* element is given its text content will be interpreted as a filename of a tree data file in the format described in section 5.5. The file's content will be read and used as the initial stand. If a *generate* element is given trees will be generated based on the parameters provided therein. If both a *load* and a *generate* element are given, data available in the file will be loaded whereas data not available will be generated based on the generation parameters.

The *generate* element should contain all of the following three elements:

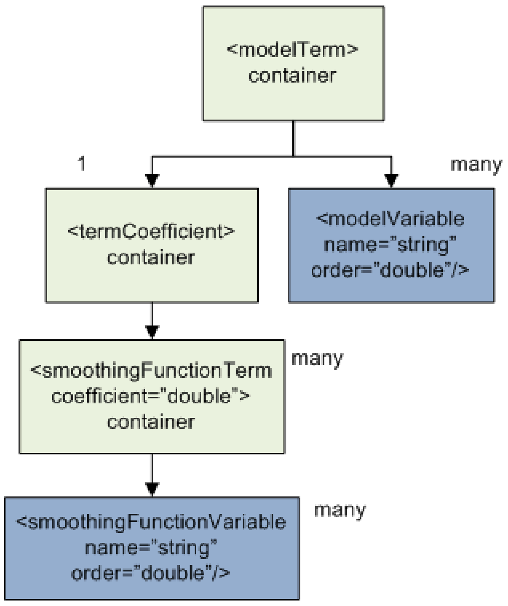
pointProcess This is a container element which should contain one of the following elements: *homogeneousPoisson, inhomogeneousPoisson, homogeneousClustered, inhomogeneousClustered, homogeneousRegular* or *inhomogeneousRegular*. The child element determines the point process used to place new trees. The point process elements and their child elements are listed in section 5.1.4.

averageInitialAge The element's value should be an integer stating the average age of trees in the initial stand.

initializationParameters This element should contain *mark* elements that provide the models for calculating mark values of trees in the initial stand. Each *mark* element should contain a set of *modelTerm* elements and optionally an *environmentalEffects* container.

5.1.3 Model terms

A *modelTerm* element is a child of a *mark* or a *competitionDistanceModel* element. A *modelTerm* element corresponds to a term in a model function, which contains variables and a coefficient. There may be any number of sibling *modelTerm*s inside a *mark* or *competitionDistanceModel* container, whereupon each *modelTerm* element represents a separate term, see figure 5.6.



<modelTerm> container
1
many
<termCoefficient> container
<modelVariable name="string" order="double"/>
<smoothingFunctionTerm coefficient="double"> container
many
<smoothingFunctionVariable name="string" order="double"/>
many

Figure 5.6: A *modelTerm* element corresponds to a term in a model function.

modelTerm elements correspond to the terms in formula (3.2). A *modelTerm* must contain a *termCoefficient* element, which represents the term's coefficient, and a set of *modelVariable* elements, which represent variables. A *modelVariable* element has a *name* attribute whose value should be one of the variable names listed in section 3.4.1. Additionally, a *modelVariable* may have a *order* attribute whose value is the order to which the variable is to be exponentiated. Moreover, variables can be multiplied by each other by having several sibling *modelVariable* elements.

A *termCoefficient* element corresponds to a secondary-level model, as in formula (3.3). A *termCoefficient* element has a set of *smoothingFunctionTerm* elements that represent secondary-level terms, and each term contains a set of *smoothingFunctionVariable* elements that represent secondary-level variables. A *smoothingFunctionTerm* has a *coefficient* attribute that represents the term's coefficient.

A *smoothingFunctionVariable* has a *name* attribute whose value should be one of the secondary-level variables listed in section 3.4.1. Secondary-level variables can also be exponentiated by providing the *smoothingFunctionVariable* with an *order* attribute, and they can also be multiplied by each other by providing several sibling *smoothingFunctionVariable* elements.

5.1.4 Point processes

The elements *homogeneousPoisson*, *inhomogeneousPoisson*, *homogeneousClustered* and *inhomogeneousClustered* and their child elements are shown in figure 5.7 The elements *homogeneousRegular* and *inhomogeneousRegular* and their child elements are shown in figure 5.8.

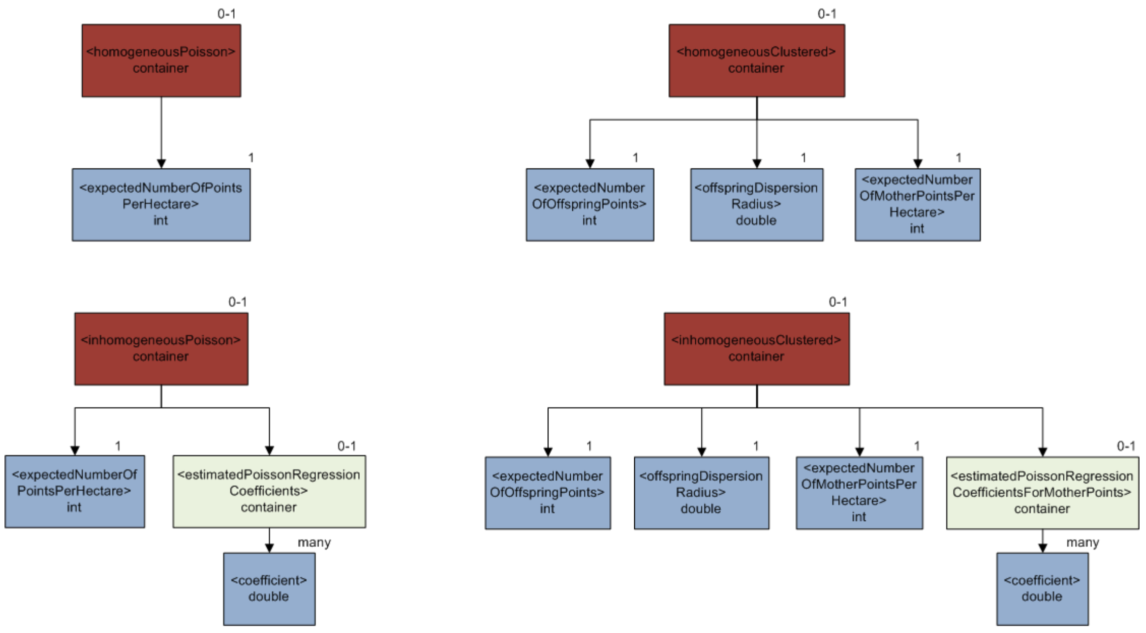

0-1
<homogeneousPoisson> container
1
<expectedNumberOfPoints PerHectare> int
0-1
<homogeneousClustered> container
1
<expectedNumber OfOffspringPoints> int
1
<offspringDispersion Radius> double
1
<expectedNumber OfMotherPointsPer Hectare> int
0-1
<inhomogeneousPoisson> container
1
<expectedNumberOf PointsPerHectare> int
0-1
<estimatedPoissonRegression Coefficients> container
many
<coefficient> double
0-1
<inhomogeneousClustered> container
1
<expectedNumber OfOffspringPoints> int
1
<offspringDispersion Radius> double
1
<expectedNumber OfMotherPointsPer Hectare> int
0-1
<estimatedPoissonRegression CoefficientsForMotherPoints> container
many
<coefficient> double

Figure 5.7: The elements *homogeneousPoisson*, *inhomogeneousPoisson*, *homogeneousClustered* and *inhomogeneousClustered*.

5.1.5 Species parameters

The *speciesParameters* element contains a set of *species* elements. The content of a *species* element defines characteristics that are specific to a certain species, see figure 5.9. A *species* element may contain the following elements:

name The name element is obligatory. It should contain a unique name identifying the species.

secondaryModelValue If given, it should contain a numerical value that is subsequently assigned the "Species" variable in the secondary-level model.

taperCurveParameters This is a container element which if present must contain all of the following elements: *coefficient1*, *coefficient2*, *coefficient3*, *coefficient5*, *coefficient8*, *coefficient13*, *coefficient21* and *coefficient34*. The child elements should contain numerical values for the corresponding taper curve coefficients.

competitionDistanceModel This is a container element that contains a model for the competition distance. Its child elements are *modelTerm* elements.

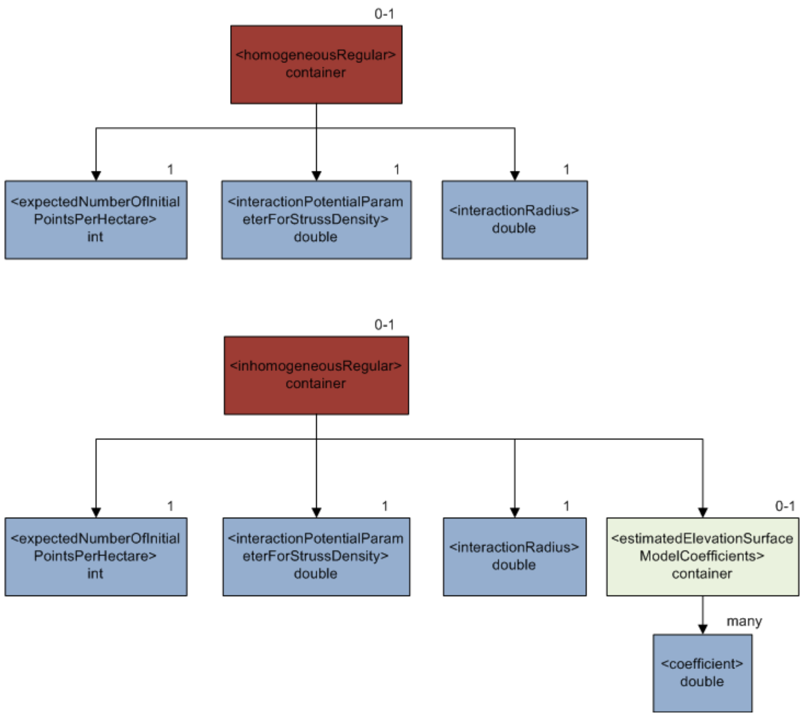

0-1
<homogeneousRegular> container
1
<expectedNumberOfInitialPointsPerHectare> int
1
<interactionPotentialParameterForStrussDensity> double
1
<interactionRadius> double
0-1
<inhomogeneousRegular> container
1
<expectedNumberOfInitialPointsPerHectare> int
1
<interactionPotentialParameterForStrussDensity> double
1
<interactionRadius> double
0-1
<estimatedElevationSurfaceModelCoefficients> container
many
<coefficient> double

Figure 5.8: The elements *homogeneousRegular* and *inhomogeneousRegular*.

stumpHeightParameters This element contains parameters used for calculating the height of the stump. It should contain each one of the following elements: *minimumHeight*, *heightCoefficient* and *diameterCoefficient*.

smallVolumeCoefficient When a tree is too small the program forgoes taper curves and uses a cylindrical model to calculate the volume. The cylindrical volume is multiplied by this weight coefficient. If not given its default value is 0.6.

5.1.6 Forest type parameters

The *forestTypeParameters* element contains a set of *forestType* elements. The content of a *forestType* element defines a particular forest type, see figure 5.10. A *forestType* element may contain the following elements:

name The name element is obligatory. It should contain a unique name identifying the forest type.

species There should be one or several *species* elements—one for each species that occurs in the forest type. The mandatory *name* attribute is used to refer to the species in question. The element's content is the relative species ratio, which is used when generating trees for the initial stand. The optional attribute *reproductionAmount* is the relative species ratio used in

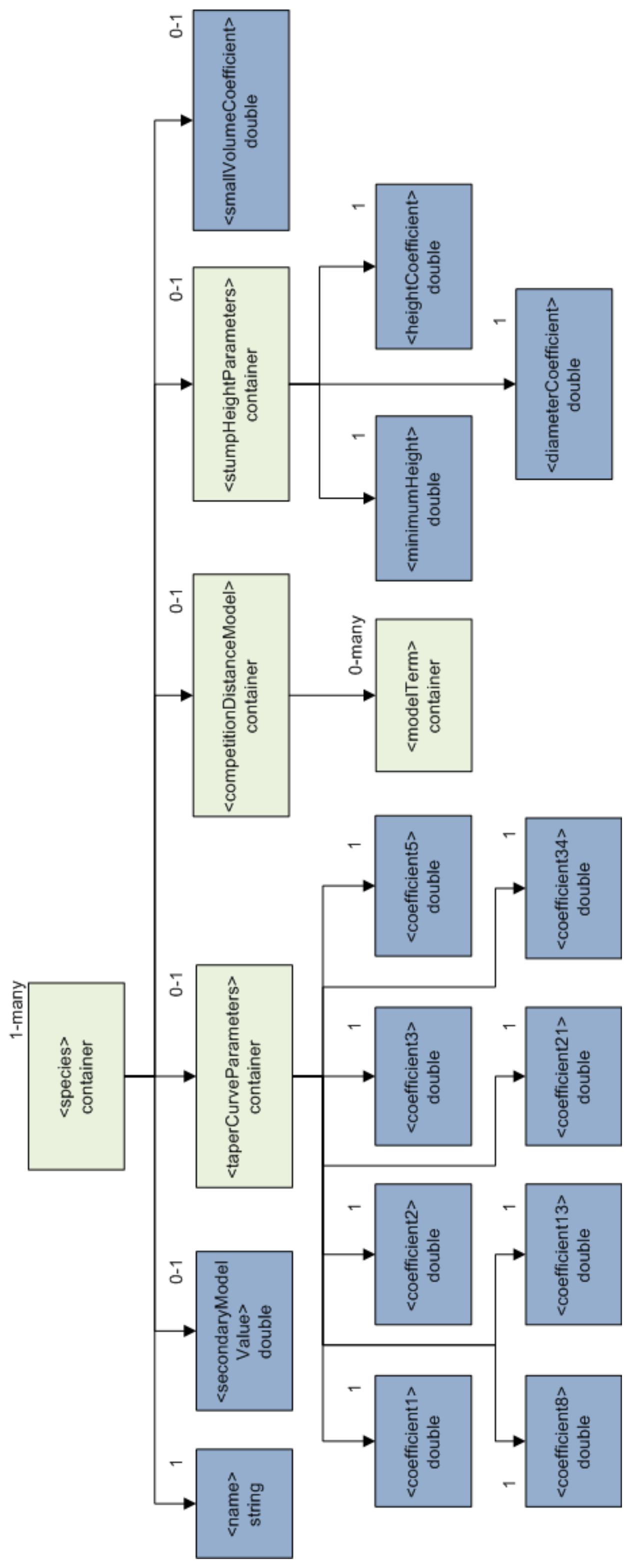


Figure 5.9: A *species* element contains parameters specific to a certain species.

the reproduction phase. The optional attributes *initialTreeDiameterFactor* and *newTreeDiameterFactor* are factors that the trees' diameters are multiplied by in the initial stand and reproduction phase, respectively.

initialStandTransformations This element may contain a *transformation* element for each mark that is to be transformed in the initial stand phase.

reproductionTransformations This element may contain a *transformation* element for each mark that is to be transformed in the reproduction phase.

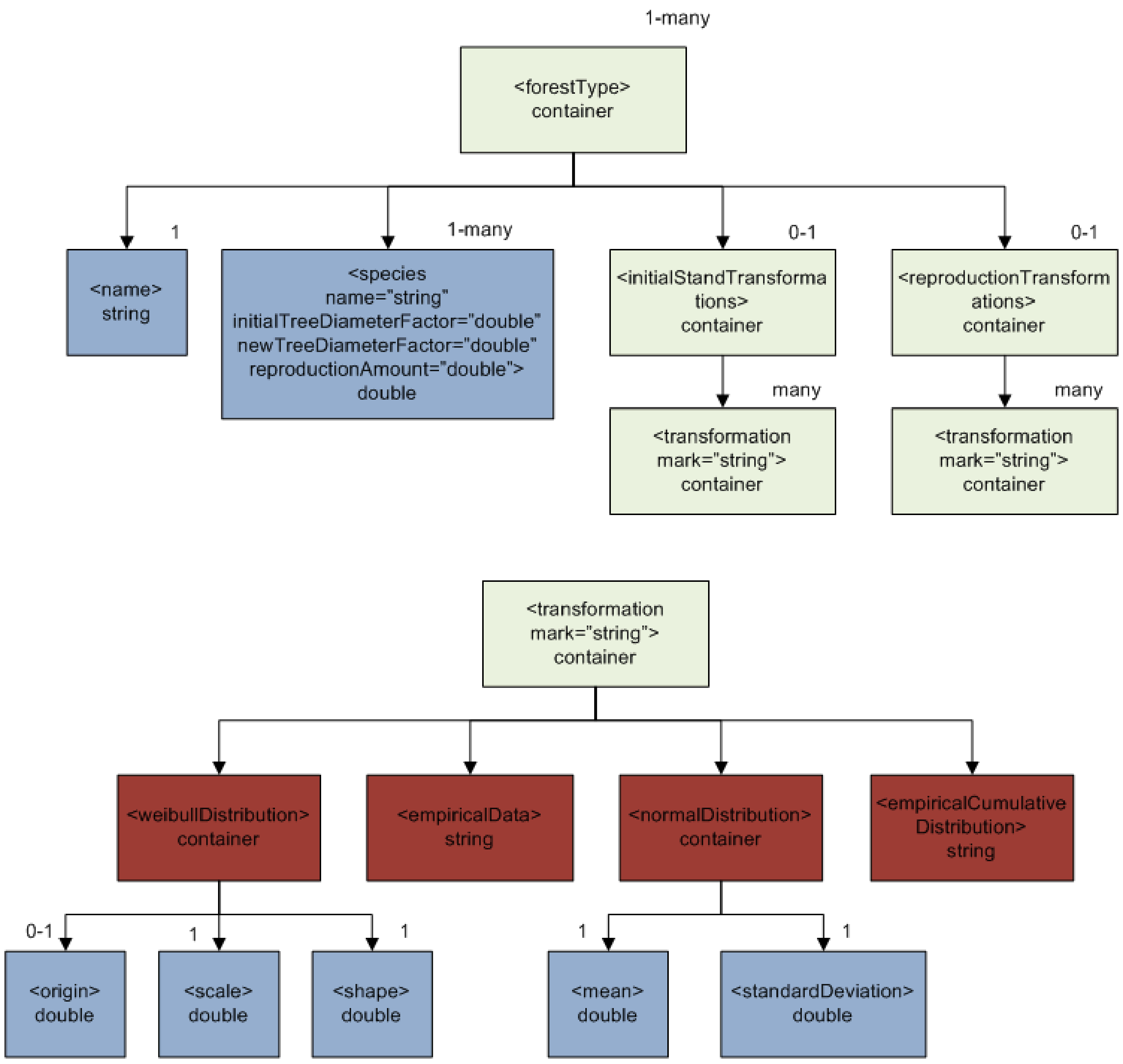

1-many
<forestType> container
1
<name> string
1-many
<species name="string" initialTreeDiameterFactor="double" newTreeDiameterFactor="double" reproductionAmount="double"> double
0-1
<initialStandTransformations> container
0-1
<reproductionTransformations> container
many
<transformation mark="string"> container
many
<transformation mark="string"> container
<transformation mark="string"> container
<weibullDistribution> container
<empiricalData> string
<normalDistribution> container
<empiricalCumulativeDistribution> string
0-1
<origin> double
1
<scale> double
1
<shape> double
1
<mean> double
1
<standardDeviation> double

Figure 5.10: A *forestType* element defines a particular forest type.

A *transformation* element, see figure 5.10, contains the parameter of a target distribution of a transformation. It should contain one of the following child elements: *weibullDistribution*, *normalDistribution*, *empiricalData* or *empiricalCumulativeDistribution*. The *weibullDistribution* element, which designates a theoretical Weibull distribution, should contain the elements *scale* and *shape* and may contain the element *origin*. The *normalDistribution* element, which designates a theoretical normal distribution, should contain the elements *mean* and *standardDeviation*. The elements *empiricalData* and *empiricalCumulativeDistribution* should contain a filename that refers to a file containing empirical data or an empirical cumulative distribution, respectively. The file format of these files are described in sections 5.3 and 5.4.

5.1.7 Management parameters

The *managementParameters* may contain a set of *managementTemplate* elements, see figure 5.11. Each *managementTemplate* contains a set of rules for carrying out management and can be referred to by a compartment. A *managementTemplate* element contains the following child elements:

thinningIntervalInYears If specified, the value of this element dictates the minimum number of years between two consecutive thinning instances. This rule only applies to non-clear cut methods.

lowerBasalArea If specified, the compartment must have basal are larger than this value. The value is given in square meters per hectare and the rule applies to all management methods.

managementMethod Contains one and only one of the following elements: *lowThinning*, *selectiveThinning*, *dimensionCutting* or *energywoodCutting*.

lowerCompartmentVolume If specified, the compartment must have a volume larger than this value. The value is given in cubic meters per hectare and the rule applies to all management methods.

yearsBetweenClearCuts This value dictates the minimum number of years between to consecutive clear cuts. This value must be specified if a clear cut is used.

clearCut If this element is given a clear cut is performed.

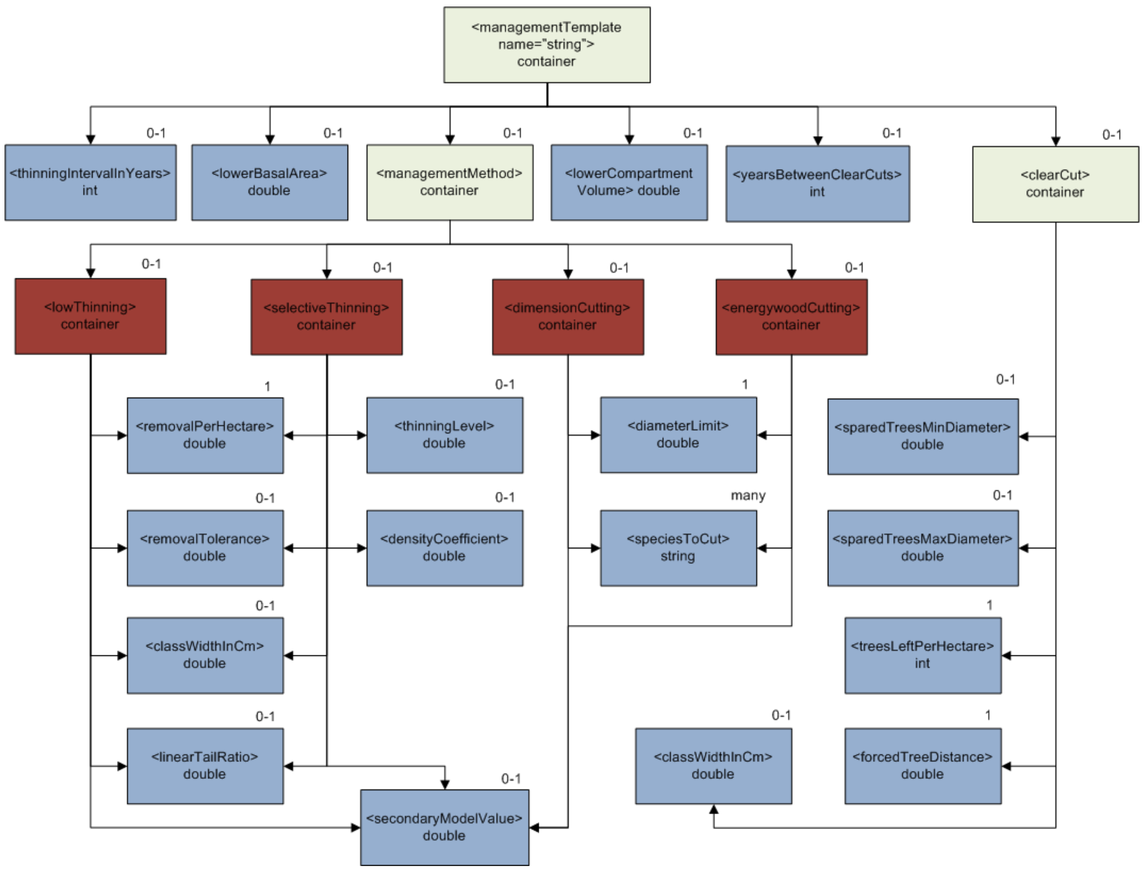
Dimension and energy wood cutting

The elements *dimensionCutting* and *energywoodCutting* may contain the following child elements:

diameterLimit For dimension cutting trees larger than this diameter are cut, whereas for energy wood cutting trees smaller than this diameter are cut. This element is mandatory.

speciesToCut There may be zero, one or several elements of this type. Each element must contain the name of a species. If there are one or more *speciesToCut* elements only trees of those species are cut.

secondaryModelValue If given, it should contain a numerical value that is subsequently assigned the “Management” variable in the secondary-level model.



<managementTemplate name="string"> container
0-1
<thinningIntervalInYears> int
<lowerBasalArea> double
<managementMethod> container
<lowerCompartment Volume> double
<yearsBetweenClearCuts> int
<clearCut> container
<lowThinning> container
<selectiveThinning> container
<dimensionCutting> container
<energywoodCutting> container
1
<removalPerHectare> double
<thinningLevel> double
<diameterLimit> double
<sparedTreesMinDiameter> double
<removalTolerance> double
<densityCoefficient> double
many
<speciesToCut> string
<sparedTreesMaxDiameter> double
<classWidthInCm> double
<treesLeftPerHectare> int
<linearTailRatio> double
<forcedTreeDistance> double
<secondaryModelValue> double

Figure 5.11: A *managementTemplate* element defines a set of management rules.

Low and selective thinning

The elements *lowThinning* and *selectiveThinning* may contain the following child elements:

removalPerHectare This mandatory element indicates the volume that is to be removed. Its unit is cubic meters per hectare.

removalTolerance The default removal tolerance is five percent, but it can be overridden with this element.

classWidthInCm The default size class width is 4cm, but it can be overridden with this element.

linearTailRatio If given, the tail of the estimated Weibull curve will be linearized according to this ratio.

secondaryModelValue If given, it should contain a numerical value that is subsequently assigned the “Management” variable in the secondary-level model.

Additionally, the *selectiveThinning* element may also contain the following child elements:

thinningLevel This element specifies the β_0 coefficient in formula (3.12). If not given the default value is 0.5.

densityCoefficient This element specifies the β_1 coefficient in formula (3.12). If not given the default value is 100.

Clear cut

The element *clearCut* may contain the following child elements:

treesLeftPerHectare The number of trees per hectare to spare while the rest are cut.

forcedTreeDistance The trees that are spared must stand no closer than this distance.

sparedTreesMinDiameter The minimum diameter of trees to spare.

sparedTreesMaxDiameter The maximum diameter of trees to spare.

classWidth The width of size classes when classifying trees. If not given the default value is 4cm.

5.1.8 Assortment parameters

The *assortmentParameters* element may contain one or several *assortment* elements. An *assortment* element must have a *name* element and its value must be unique. There may be zero, one or several *constraints* elements, which each one specifies constraints for a particular species. The species is indicated by the *species* attribute. If no *species* attribute is given the constraints are applicable to all species. Figure 5.12 shows the *assortment* element and its child elements.

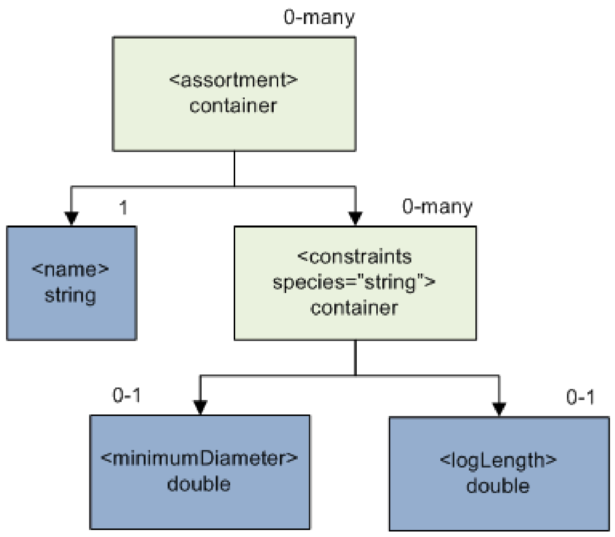


Figure 5.12: An *assortment* element defines a specific assortment.

The *constraints* element may contain the *minimumDiameter* and *logLength* elements that dictate the minimum log diameter and the log length, respectively. If none of the elements are present no constraints will be imposed and the assortment can be used for the top of the tree.

5.1.9 Compartment parameters

The *compartmentParameters* element must contain one or several *compartment* elements. Each *compartment* element uniquely identifies a particular compartment. A *compartment* element should have a *id* attribute that contains a unique identifier, which is a positive integer. Furthermore, a *compartment* element may contain the following child elements:

forestTypeName This mandatory element should contain the name of the forest type of which the compartment is made up.

managementTemplateName There may be none or several elements of this type which each contains the name of a management template that is to be applied to the compartment.

polygonFile There should be one or several *polygonFile* elements. The elements contain the polygon file's name and must have an *id* attribute that indicates the polygon to use.

numberOfInitialTreesFactor If given, the number of trees in the initial stand is multiplied by this factor.

ageOfInitialTreesFactor If given, the age of the trees in the initial stand is multiplied by this factor.

The *compartment* element and its child elements are shown in figure 5.13.

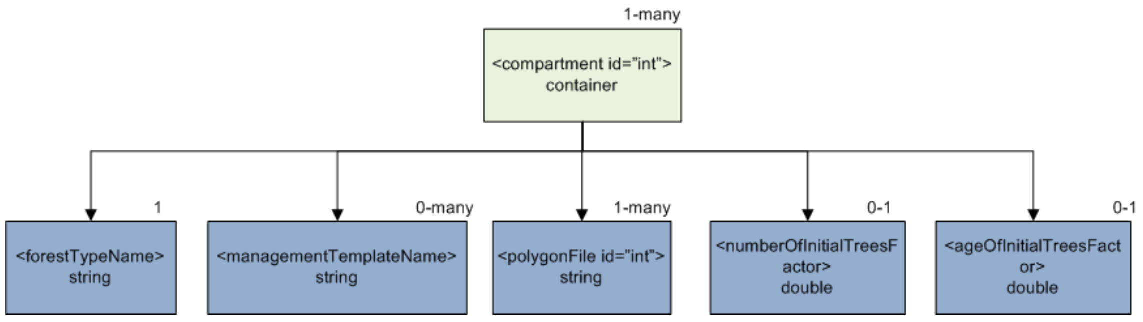


Figure 5.13: A *compartment* element contains compartment-specific parameters.

5.1.10 Inspection parameters

The optional *inspectionParameters* element enables additional logging and allows one to inspect model calculations and thinning procedures more closely as they are carried out, see figure 5.14. The *inspectionParameters* element may contain the following child elements:

inspectModel There may be none or multiple *inspectModel* elements which each may contain one of the following values: “initial stand”, “growth”, “mortality”, “reproduction”, “regeneration” or “competition distance”. An element with one of the aforementioned values enables calculation inspection of the corresponding model.

inspectThinning There may be none or multiple *inspectThinning* elements which each may contain one of the values “weibull iteration” or “weibull thinning probability”.

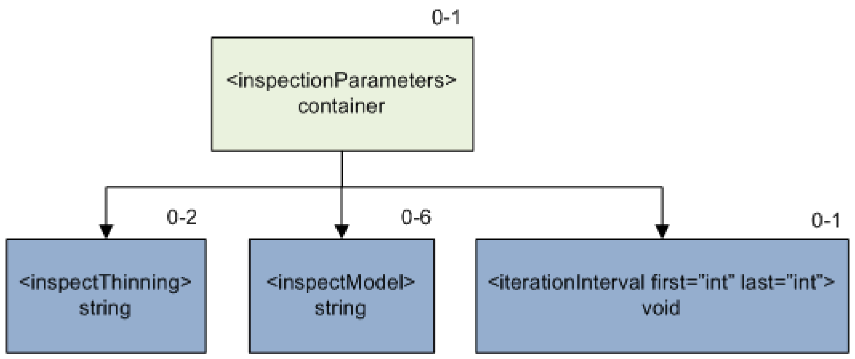


Figure 5.14: The *inspection* element enables more detailed inspection of the simulation process.

iterationInterval If given, the element should have a *first* and a *last* attribute. Inspection is then enabled only for the inclusive iteration interval given by the attributes.

5.2 Polygon file

SPATE-HPC uses polygon files to identify the polygonal areas that constitute compartments. There must exist at least one polygon file in a simulation. A polygon file is a plain text file containing the coordinates of one or several polygons. The coordinate system is a two-dimensional Cartesian coordinate system where the units are meters.

A polygon has a unique ID number that is used to associate it with a compartment, and a list of vertices (points). The vertices must be given in order so that a coherent polygonal shape can be outlined by drawing consecutive straight lines between the vertices. The last vertex must coincide with the first one. This way, the last line will end exactly where the first line started, thus enclosing the polygon. The minimum number of vertices of a polygon is hence four, which would correspond to triangle. It does not matter whether the vertices are given in clockwise or counterclockwise direction. Polygons may be conjoined by sharing borders, and there may also be regions within the forest area that are not covered by polygons. However, polygons *must not* overlap.

The following is an example of a polygon file containing two polygons: a square and a triangle.

```
1
0.0 0.0
100.0 0.0
100.0 100.0
0.0 100.0
0.0 0.0
END
2
100.0 0.0
200.0 0.0
100.0 100.0
100.0 0.0
END
END
```

A polygon definition begins with a *unique* ID number. Thereafter follows a list of vertex coordinates where each line contains the coordinates of one vertex. Coordinates are given as a pair of real-valued numbers where the first number is the x coordinate and the second one is the y coordinate. The numbers must be separated by white space, i.e., one or several space or tabular characters. After the polygon's last vertex follows a line with the word "END". An additional "END" follows the last polygon.

5.2.1 Cut-out regions

Polygons may contain cut-out regions (holes). Cut-out regions are defined by providing an additional polygon with the same ID number as a previously defined polygon. Instead of becoming a separate polygon, the second polygon will constitute a hole in the previously defined polygon. One is also free to put another, new polygon within the hole. In other words, it is possible to have a polygon that is completely surrounded by another polygon. The following is an example of a square polygon with a square cut-out region:

```
1
0.0 0.0
100.0 0.0
100.0 100.0
0.0 100.0
0.0 0.0
END
1
40.0 40.0
80.0 40.0
80.0 80.0
40.0 80.0
40.0 40.0
END
END
```

5.3 Empirical data

Empirical data can be used to provide a target distribution for marks in the initial stand and the reproduction phase. Empirical data are provided as a set of real-valued numbers in an external file. There should be one value per row and there should not be no empty rows or rows containing any other kinds of data. The numbers may appear in any order. Example:

```
1.54
14.29
2.93
25.01
19.37
10.45
9.83
```

5.4 Empirical cumulative distribution

An empirical cumulative distribution can be used to provide a target distribution for marks in the initial stand and the reproduction phase. An empirical cumulative distribution is given as a list of pairs of numbers, with the first number being x and the second number $F(x)$, where $F(x)$ is a cumulative distribution function. The values of x must appear in increasing order and the values of $F(x)$ must be monotonically increasing and span from 0 to 1. There should be one pair per row and there should be no empty rows or rows containing any other kinds of data. The two values within a pair should be separated by white space, i.e., one or several space or tabular characters. Example:

```
5.0     0.00
8.0     0.04
11.0    0.16
14.0    0.40
17.0    0.69
20.0    0.89
23.0    0.98
26.0    1.00
```

5.5 Tree data file

The tree data file is table of tree data in the comma-separated values (CSV) format. Each row in the table corresponds to an individual tree and the columns contain specific tree characteristics. The columns are the following, in order:

X coordinate The tree's x coordinate. The unit is meters.

Y coordinate The tree's y coordinate. The unit is meters.

DBH Diameter at breast height in centimeters.

Height Total heigh in meters.

Initial heterogeneity The Gaussian random number that was assigned the tree as its initial heterogeneity.

Age The age in years.

Species The name of the species.

5.6 Compartment statistics file

The compartment statistics file is a plain text file containing various figures and tables pertaining to a compartment's development. All values are averaged over all replications. If the user has run several replications some of the values will be accompanied by the lower and upper limits of a 95% Monte Carlo confidence interval. A compartment statistics file contains the following sections:

Simulation summary This section contains general information, such as the number of years, iteration steps and replications simulated, as well as compartment-specific information, such as compartment area, forest type and initial species composition.

Statistics summary Here, tables that show how the stand has developed throughout the simulation are presented. The section contains the following tables:

Age frequency table The age frequency of the final stand.

Iteration statistics The basal area and volume development in the different simulation phases and for each iteration step.

Stem statistics The total number of trees in the different simulation phases, of different age class and for each iteration step.

Assortment volumes The growing stock volumes of different assortments and species for each iteration step.

Removal assortment volumes The removal volumes of different assortments and species for each iteration step.

Stand summary The stand summary section contains mean values of tree characteristics and the final species composition.

5.7 Statistics table file

The statistics table file contains a table in the comma-separated values (CSV) format with the final characteristics of each compartment. Each row in the table corresponds to a compartment polygon and the columns contain specific polygon or compartment characteristics. The columns are the following, in order:

Polygon id The polygon's ID number.

Compartment id The corresponding compartment's ID number.

Polygon area The polygon's area in hectares.

Compartment area The compartment's area in hectares.

Forest type The name of the forest type.

Basal area growth The total basal area growth per hectare.

Final basal area The final basal area per hectare.

Volume growth The total volume growth per hectare.

Dead tree volume The total volume of dead trees, per hectare.

New tree volume The total volume of new trees, per hectare.

Removal The total volume of trees removed, per hectare.

Removal "species" The total removal volume of the species "species" per hectare.

Removal "assortment" The total removal volume of the assortment "assortment" per hectare.

Removal "species" "assortment" The total removal volume of the species "species" and assortment "assortment" per hectare.

Final volume The final volume per hectare.

Growing stock "species" The final volume of the species "species" per hectare.

Growing stock "assortment" The final volume of the assortment "assortment" per hectare.

Growing stock "species" "assortment" The final volume of the species "species" and assortment "assortment" per hectare.

Mean diameter The mean diameter in centimeters.

Mean height The mean height in meters.

Mean volume The mean trunk volume in liters.

Number of trees in compartment The total number of trees in the compartment.

"age class" The total number of trees in the compartment in age class "age class".

"species" The total number of trees in the compartment of species "species".

Polygon file The filename of the corresponding polygon file.

Index

ACML, 3
allowed imbalance percentage, 6
assortment, 21, 36

calculation inspection, 25
Cholesky decomposition, 13
clear cut, 20, 35
command line, 4
compartment, 9, 37
competition distance, 9
    model, 17, 32
competition index, 9
competitor, 9
compiling, 3, 6
correlated random numbers, 9, 13
covariance function, 13
CppUnit, 3
CSV, 58, 59
cut-out region, 56

density coefficient, 19, 35
diameter limit, 18, 35
dimension cutting, 18

editor, 23
empirical cumulative distribution, 16
empirical data, 16
energy wood cutting, 18
environmental effects, 4, 13

forest type, 17, 29

GCC, 3, 6
growth, 8, 14, 25
GSL, 3

initial stand, 13, 29
input file, 39
inspection, 4
iteration length, 25

linear tail, 35
load balancing, 5, 25
local correlation factor, 13
log file, 21
log length, 21, 36
logistic regression, 14, 19
low thinning, 18, 19
lower basal area, 20, 35
lower compartment volume, 20, 35

makefile, 3, 6
management, 8
    method, 17, 32
    template, 18, 32
minimum diameter, 21, 36
model, 10, 26
    secondary-level, 26
    value, 16
mortality, 8, 14, 25
    threshold, 14
MPI, 3, 5
MPICH, 3

output directory, 23
output file, 21

PGI, 3
point process, 13
Poisson random number, 13, 14, 19
polygon, 9, 37, 56

regeneration, 14, 25
removal per hectare, 19, 35
removal tolerance, 19, 35
replication, 25
reproduction, 8, 14, 25

secondary model value, 32, 35
secondary-level model, 10
selective thinning, 18, 19
size class, 23, 35
small volume coefficient, 17
smoothing function, 11
species, 16, 32
SPRNG, 3
standard normal limiting factor, 15
statistics, 21
stump, 16, 32

taper curve, 32
thinning interval, 20, 35
thinning level, 19, 35
thinning rate, 18
tree data file, 13, 21

volume, 16, 32

Weibull thinning, 18
wxWidgets, 6

Xerces, 3
XML, 39

year, 25
years between clear cuts, 20