PIANOS requirements specifications

Group Linja

Helsinki 7th September 2005

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Joonas Kukkonen
Marja Hassinen
Eemil Lagerspetz

Client

Marko Salmenkivi

Project Masters

Juha Taina
Vesa Vainio (Instructor)

Homepage

`http://www.cs.helsinki.fi/group/linja`

Contents

1 **Preface** **1**

1.1 Overview . . . 1

1.2 Version history . . . 1

2 **Introduction** **2**

2.1 The problem domain . . . 2

2.2 Previous systems . . . 2

2.3 The problem domain in diagrams . . . 2

3 **Glossary** **4**

3.1 Probabilistic inference . . . 4

3.2 Models and related concepts . . . 6

3.3 Metropolis-Hastings algorithm . . . 6

4 **Use cases** **8**

4.1 The format of use cases . . . 8

4.2 Defining a model . . . 8

4.2.1 Creating a new model . . . 8

4.2.2 Modifying an existing model . . . 8

4.3 Defining simulation parameters . . . 8

4.4 Running the simulation . . . 9

5 **Requirements** **10**

5.1 Model requirements . . . 11

5.2 Data requirements . . . 17

5.3 Simulation requirements . . . 18

5.4 Output requirements . . . 25

5.5 General error conditions . . . 29

5.6 Non-functional requirements . . . 31

5.7 General requirements . . . 32

6 **System architecture** **34**

7 **System evolution** **35**

8 References **36**

1 Preface

This is a software requirements specification (SRS) document of the PIANOS project. The purpose of an SRS is to list and prioritize all requirements set for the software being produced, and so work as an agreement between the project personnel and the customer. This document provides a basis for future enhancements and it also reduces development effort.

1.1 Overview

1. Preface	Describes this document's purpose.
2. Introduction	Describes the problem domain.
3. Glossary	Explains the glossary used in this document.
4. Use cases	Describes the use cases of the program.
5. Requirements	Describes the requirements set for the program.
6. System architecture	This section contains a simple model of the software and its inputs and outputs.
7. System evolution	Describes which requirements are likely to change in the future.

1.2 Version history

Version	Date	Modifications
1.0	01.06.2005	The initial version
1.1	14.06.2005	The draft to be inspected
1.2	21.06.2005	The final version

2 Introduction

The project's goal is to produce a program for the University of Helsinki's spatial data analysis research group. The program analyzes spatial data by using a prior distribution and a mathematical model to calculate the posterior distribution for a set of variables. The posterior distribution could for example be a bird species' distribution in Finland. The program will not analyze how accurate the given model is, it will only calculate the posterior distribution. Model comparison will be done with other software using the posterior distributions as data.

2.1 The problem domain

The main topic of interest is the question about the most accurate model to describe a given real life phenomenon. The only way to compare the models is to compare the results. Defining a general modeling language is hard, implementing a program to process data using a general model is even harder. The PIANOS project will concentrate on producing a program for processing two given datasets. This cuts back the amount of needed features in the modeling language.

The simulation needs a large number of iterations before the results start to converge on certain points. Due to the number of iterations and large data, the calculation might take a long time to finish. This requires the program and the implementation language to be efficient.

2.2 Previous systems

The software is related to another currently in use, named Bassist: 'Bassist is a tool that automates the use of hierarchical Bayesian models in complex analysis tasks. Such models offer a powerful framework for modeling statistically complex real-world phenomena.' [Bass]

Currently the research group is using BASSIST to analyze spatial problems. BASSIST was not designed for problems with spatial dependencies. Its modeling language lacks features which would allow efficient definition of spatial problems. The goal is to concentrate on a small subset of spatial problems which cannot be solved efficiently with BASSIST.

2.3 The problem domain in diagrams

This section aims to provide a high abstraction level map of the problem domain. It is a set of ER-diagram-style maps of the domain as the project members see it.

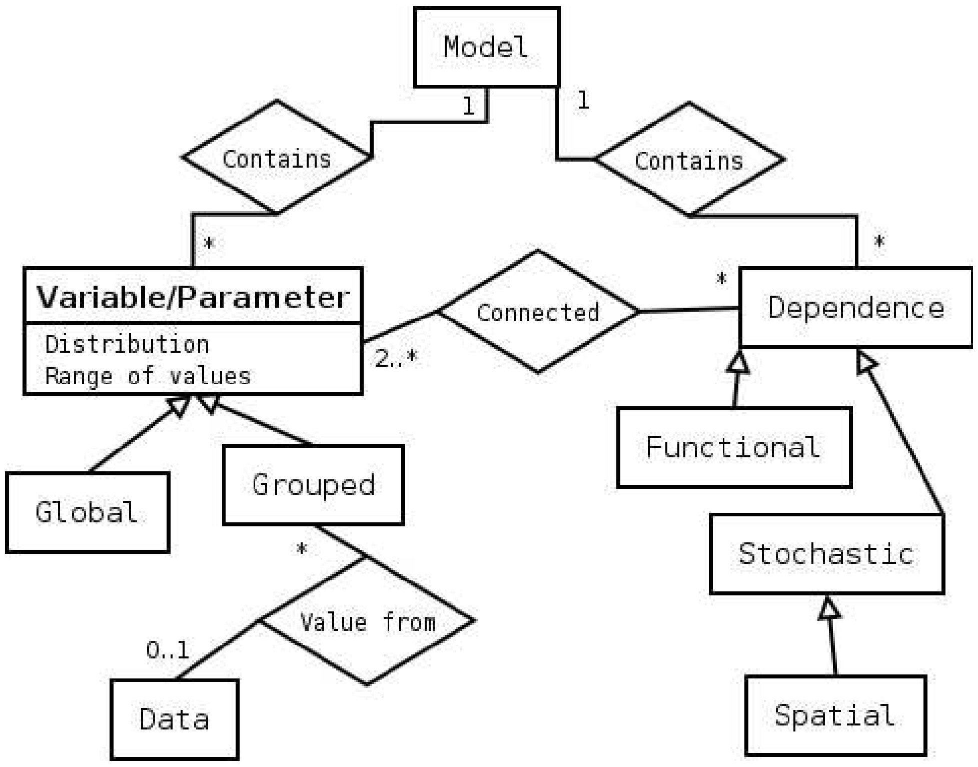

Figure 1: The model and surrounding terms

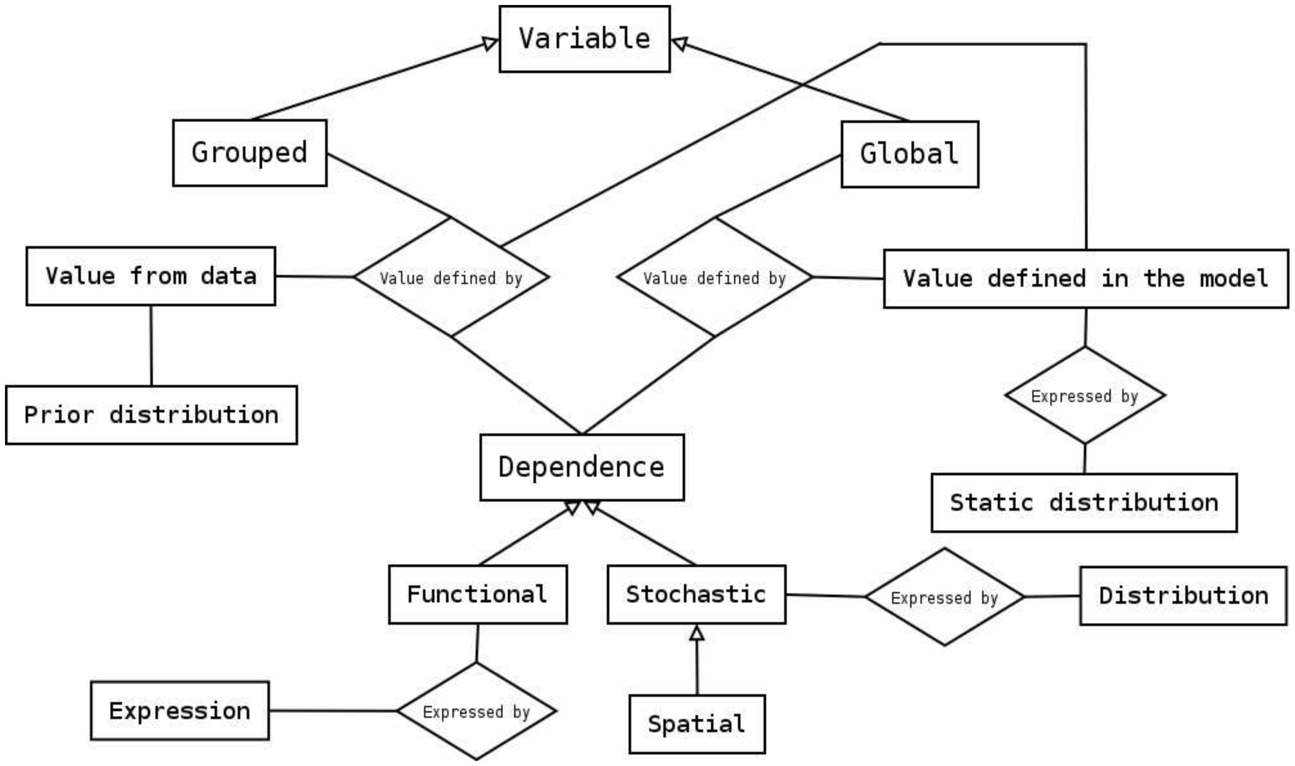

Figure 2: The variables

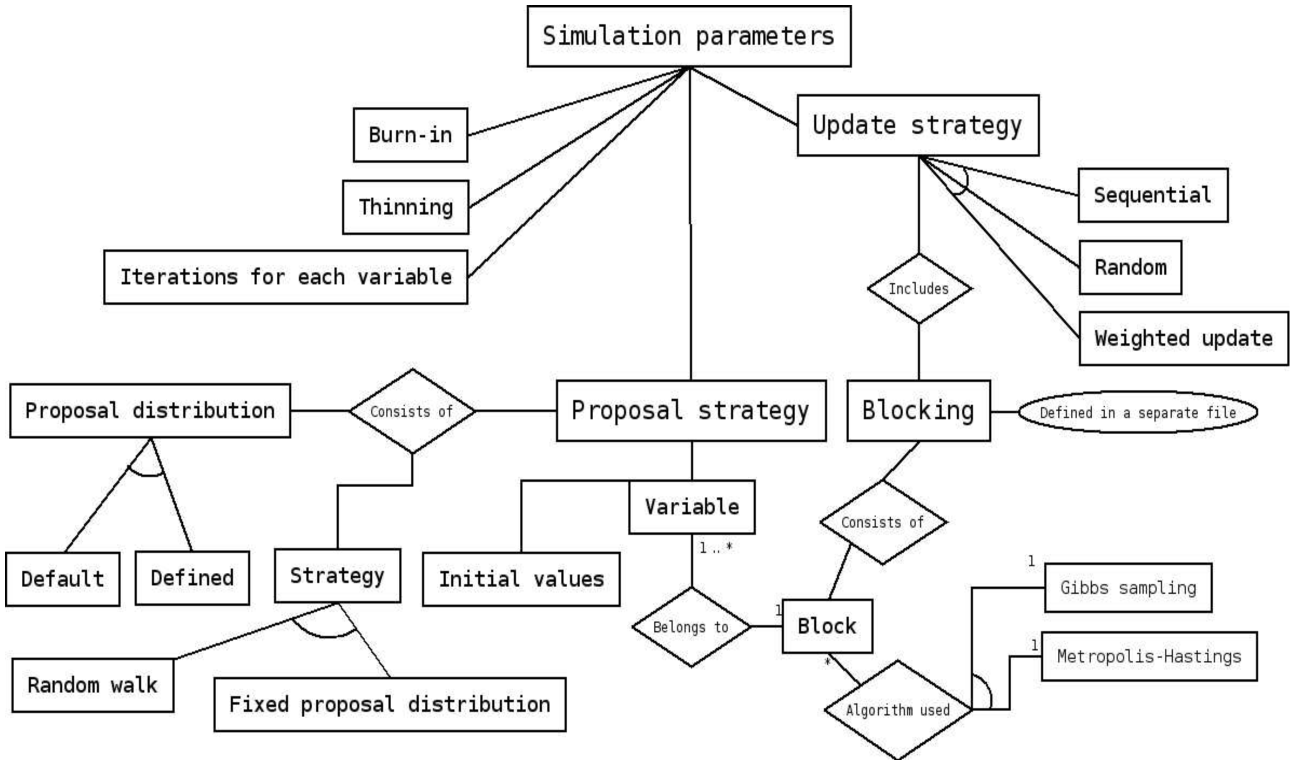

Figure 3: The simulation parameters

3 Glossary

3.1 Probabilistic inference

Random variable (synonym: stochastic variable): Function which combines events with their probabilities. A numeric variable related to a random phenomenon (for example throwing a dice). The value of the variable is determined if the result of the phenomenon is known, otherwise only the probabilities of different values are known. [Tode04]

Discrete random variable: Random variable with a discrete range.

Point probability function: (Synonym: frequency function) Function $f : R \to R$ related to discrete random variable X so that $\forall x \in R : f(x) = P\{X = x\} =$ the probability that the value of X is x. [Tode04]

Density function: $f(x)$ is a density function iff

1. $f \geq 0$
2. f is integrable in R and $\int_{-\infty}^{\infty} f(x)dx = 1$.

[Tode04]

Cumulative distribution function: The function $F : R \to R$ is the cumulative distribution function associated with random variable X iff $F(x) = P\{X \leq x\} =$ the probability that X is less than or equal to x. [Tode04]

Continuous distribution: A random variable has a continuous distribution as its density function f if $\forall a, b \in R : P\{a \leq X \leq b\} = \int_a^b f(x)dx$ [Tode04]

Joint distribution: If X and Y are random variables, their joint distribution describes how probable all the possible combinations of X and Y are.

Independence: Events A and B are independent, if the probability of B is independent on whether A has happened or not. (For example if A = “it rains”, B = “when I throw a dice the result is 6” and C=“when I throw a dice the result is even” then A and B are independent but B and C are not.) [Tode04]

Conditional probability: If X and Y are random variables, the conditional probability $P\{X = x|Y = y\}$ means the probability that the value of X is x if it is assumed that the value of Y is y.

Bayes’s rule: $P\{X = x|Y = y\} = \frac{P\{Y=y|X=x\} \cdot P\{X=x\}}{P\{Y=y\}}$. The rule is obtained from the observation that $P\{Y = y|X = x\} \cdot P\{X = x\} = P\{X = x \text{ AND } Y = y\}$

Chaining: Using the Bayes’s rule many times consecutively.

Bayesian model: A model which connects dependent random variables to each other by defining dependencies and conditional probabilities. The model defines the joint distribution of the variables.

Likelihood function: $P\{data|explanation\}$ is called the likelihood function because it defines how likely it is to get such a data if the real conditions are known. (For example if we know that a bird resides at some area, then how likely it is to get the observations we have already got.)

Markov random field: A random field that exhibits the Markovian property:
$\pi(X_i = x_i|X_j = x_j, i \neq j) = \pi(X_i = x_i|\delta_i)$, Where δ_i is the set of neighbours for X_i. That is, only the adjacent units affect the conditional probability. [Rand]

Discrete distributions:

- **Uniform distribution**
- **Binomial distribution**
- **Geometric distribution**
- **Poisson distribution**

Continuous distributions:

- **Uniform distribution**
- **Normal distribution**
- **Multinomial distribution**
- **Exponential distribution**

- **Binormal distribution**
- **Lognormal distribution**
- **Beta distribution**
- **Gamma distribution**
- **Dirichlet distribution**

Descriptions of these distributions can be found in [Math05].

Functional dependence: A condition between two variables X and Y so that the value of X determines the value of Y unambiguously.

Stochastic dependence: A condition between two variables X and Y so that the value of X doesn't determine the value of Y but influences the probabilities of the possible values.

Spatial dependence: A special case of stochastic dependence where the dependence is related to some spatial structure. (For example towns that are adjacent to each other influence to each other.) The spatially dependent variables form a Markov random field.

Prior distribution: The prior distribution of the parameters describes their assumed joint probability distribution before inferences based on the data are made.

Posterior distribution: The posterior distribution of the parameters describes their joint probability distribution after inferences based on the data are made.

Marginal distribution: The prior or posterior distribution concerning only one parameter.

3.2 Models and related concepts

Model: Means: Bayesian model

Variable: A variable is an entity in the model that can have an assigned value from its range of values. Variables and their dependencies form the base of the problem that the software is developed to solve.

Parameter: A parameter is a variable whose value is not defined by data.

Adjacency matrix: The adjacency matrix of a simple graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position ij according to whether i and j are adjacent or not.

Floating point number: A computer representation of a real number with finite precision.

3.3 Metropolis-Hastings algorithm

Iteration: A single round of the algorithm when all the parameters have been updated once.

Burn-in-iterations: The iterations that are run before any output is produced.

Thinning factor: The thinning factor t means that every t^{th} iteration value is used in the output and the rest are discarded.

Block: A set of parameters which are defined to be updated together. That is, the proposals are generated to all of them and the acceptance of all the proposals is decided at the same time.

Update: Proposing a value to a parameter and then accepting it (the value changes) or discarding it (the value remains the same).

Proposal strategy: The proposal strategy defines how the next proposed value is generated. Possible choises are

1. Fixed proposal strategy: The next proposed values for a parameter is taken from its proposal distribution.

2. Random walk: The next proposed value for a parameter is created by adding a value taken from the proposal distribution to the current value of the parameter.

Proposal distribution:

1. The distribution from which the next proposed value for a parameter is chosen (when using the "Fixed proposal distribution" proposal strategy).

2. The distribution that is used in generating proposed values for a parameter by adding a value taken from the distribution to the parameter's current value (when using the "Random walk" proposal strategy).

Update strategy: The update strategy describes which variables belong to the same block. (See: Block) The update strategy also includes information about whether the blocks are considered for updates in sequential order, whether the next block to update is chosen at random or whether the block to update is chosen based on the block weights.

Convergence: The phenomenon that during the simulation the parameter values get closer to the posterior distribution. The speed of the convergence depends on the initial values and other simulation parameters.

4 Use cases

4.1 The format of use cases

Each use case begins with an introductory description. After this there are *Starting conditions*, *Ending conditions* and *Error conditions* listed. Some of these cases are divided into subcases. The subcases contain the information that differs from the definition of the supercase.

4.2 Defining a model

The user defines a model that contains necessary information for calculating statistical probabilities, with or without the spatial dimension. The specifics of what the model may contain are described in requirements M2-M7 and M10-M11.

The model is stored in a text file, using a model definition language which will be defined in the design phase.

4.2.1 Creating a new model

Starting conditions:	None
Ending conditions:	A new valid model has been created.
Error conditions:	If the model is invalid, there will be consequences in the simulation. This use case does not check for the validity.

4.2.2 Modifying an existing model

Starting conditions:	A valid existing model is present
Ending conditions:	A different valid model has been created.
Error conditions:	If the model is invalid, there will be consequences in the simulation. This use case does not check for the validity.

4.3 Defining simulation parameters

The user defines some of the simulation parameters defined in requirements S1-S12. Some of these, except the initial values, have defaults that are used in case that their corresponding parameter values are not set.

Starting conditions:	A valid model has been chosen for simulation.
Ending conditions:	The simulation parameters are set. The simulation can be run with these parameters.
Error conditions:	If parameters set incorrectly, problems may arise during the simulation.

4.4 Running the simulation

The user has specified a model to the program, the parameters for the simulation and a file for output. The program starts the simulation with these parameters. The program reports on the progress of the run on-screen, while the actual values of the simulated parameters are directed to the files specified.

Starting conditions:	A model with either default simulation parameters or set simulation parameters is present. Files to direct the output to have been given.
Ending conditions:	The simulation is run. Output is produced to the given files.
Error conditions:	Invalid model or simulation parameters can cause inconsistent results or error messages. If the program is not able to read or write to a file, it will prompt the user for another file.

5 Requirements

The requirements are classified in five groups:

- Requirements related to the statistical model. (5.1)
- Requirements related to data files. (5.2)
- Requirements related to the simulation algorithm and its parameters. (5.3)
- Requirements related to output the program must produce. (5.4)
- Requirements related to error conditions. (5.5)
- Non-functional requirements (5.6)

Each requirement has an *id*, *name*, a longer *description*, *rationale*, *priority*, *stability* and *source*.

The id:s are used when referring to the requirements from other documents (for example from the system design document). The name of the requirement states shortly what the requirement is about. The description is a longer explanation of the requirement and the rationale includes reasons why the requirement is important. The source tells who suggested including the requirement.

The requirements are divided into the following categories by priority:

- Essential: Implies that the software will not be acceptable unless these requirements are provided in an agreed manner.
- Conditional: Implies that these are requirements that would enhance the software product, but would not make it unacceptable if they are absent.
- Optional: Implies a class of functions that may or may not be worthwhile. This gives the supplier the opportunity to propose something that exceeds the SRS.

[IEEE98]

The requirements are divided into the following categories by stability:

- Stable: The requirement is unlikely to change.
- Unstable: The requirement may change in the future. The software architecture could be designed to be flexible enough to adapt to the possible changes.

5.1 Model requirements

These are the requirements related to the statistical models. This section contains the requirements that define the expressive power of the model definition language.

Id:	M1
Name:	Using models
Description:	The program must run simulations on models which are described with the model description language (specified later).
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M2
Name:	Defining variables whose values are taken from data
Description:	The model description language must be able to express variables whose values are taken from the data and specify where the values are found (the file name of the data file and the location in the data file). It must also be possible to define a distribution for such a variable (M12).
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M3
Name:	Defining parameters whose values are not taken from data
Description:	The model description language must be able to express that there are parameters whose values are not taken from the data. If the parameter is stochastic, it must also be possible to define a distribution for such a variable (M12). If the parameter is deterministic, it must be possible to define an equation that determines the value of the parameter.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M4
Name:	Defining dependencies
Description:	The model description language must be able to express dependencies between two entities, each either a variable or a parameter. The dependencies can be functional or stochastic (special case: spatial dependencies (M7)).
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M5
Name:	Equations
Description:	The model description language must be able to express functional dependencies as equations. The left side of the expression is a single parameter but the right side must be able to include basic arithmetic operations (+, -, *, /), logarithm, sum, product and power expressions.
Rationale:	Equations are a natural way to express functional dependencies.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M6
Name:	Defining variable/parameter repetition structures
Description:	The model description language must be able to express a structure of variables/parameters which are related to the same entity. This entity structure must then be automatically repeated for a specified number of times, for example to span the length of a data file.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M7
Name:	Defining spatial relations
Description:	The model description language must be able to express the meaning “all adjacent units” in mathematical expressions used in dependencies (M4). An adjacency matrix defines the entities that are neighbours to each other and form a network of spatial relationships. It must be possible to specify the file name of the adjacency matrix.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M9
Name:	Reading models from text files
Description:	The program must be able to read a model (described with the specific description language) from a text file.
Rationale:	It’s reasonable to store the models into text files because they can then be easily edited without any special tools.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M10
Name:	The distributions used
Description:	The model must be able to express at least the following distributions: Discrete distributions: • Uniform distribution • Binomial distribution • Geometric distribution • Poisson distribution Continuous distributions: • Uniform distribution • Normal distribution • Multinomial distribution • Exponential distribution • Binormal distribution • Lognormal distribution • Gamma distribution • Beta distribution • Dirichlet distribution
Rationale:	These distributions are frequently used in statistical models.
Priority:	Essential
Stability:	Unstable
Source:	Marko Salmenkivi

Id:	M11
Name:	Distributions defined by the user
Description:	The program should be able to use a distribution defined by the user. The distributions are defined as Fortran modules which include all the subroutines which are needed in the calculation.
Rationale:	
Priority:	Conditional
Stability:	Stable
Source:	Marko Salmenkivi

Id:	M12
Name:	Defining distributions
Description:	The model description language must be able to express distributions given in M10. It must be possible to use these for parameters and data variables.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

5.2 Data requirements

Id:	D1
Name:	The general data format
Description:	The program must be able to read data from files, where data is stored like a matrix, that is, the lines contain integer and floating point numbers[1]separated by spaces and the file may contain several lines. The lengths of the lines must then be equal. This format must include characters denoting that the corresponding data is missing.
Rationale:	A considerable amount of the data can be expressed in the matrix format.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	D2
Name:	Data not available
Description:	If some variable instances (for example the observation grade for each square) are not defined in the data, the variables not in the data must be treated like parameters. They are used in the simulation like any other unknown parameters (proposals of their values are made etc).
Rationale:	It is possible that all information for each entity (square, bird species etc) isn't provided in the data.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

[1]The syntax for floating point numbers is defined in [Fort01]

Id:	D3
Name:	Invalid data
Description:	If the data is invalid (for example doesn't correspond to the specified format) the program must print an error message.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Anni Kotilainen

5.3 Simulation requirements

This section lists requirements related to the calculation of the program and the simulation parameters.

Id:	S1
Name:	The algorithm used
Description:	The program must be able to run simulations using the Metropolis-Hastings algorithm.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S2
Name:	Choice of algorithm
Description:	The user could be able to choose to use the Gibbs sampling algorithm for desired blocks instead of the Metropolis-Hastings.
Rationale:	
Priority:	Conditional
Stability:	Unstable
Source:	Marko Salmenkivi

Id:	S3
Name:	Setting the number of updates
Description:	The user must be able to define the number of the updates for each block. That is, the simulation ends when every parameter has been updated at least that many times. (Note that update doesn't necessarily mean that the value changes.) It must also be possible to define a default number of updates which is used for each block if no other value is given.
Rationale:	The user wants to regulate how long the simulation will take.
Priority:	Essential
Stability:	Unstable
Source:	Marko Salmenkivi

Id:	S4
Name:	Setting the number of burn-in iterations
Description:	The user must be able to set the number of burn-in iterations: That is, the number of iterations that are run before any output is produced.
Rationale:	It takes a while until the iteration values actually conform to the posterior distribution.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S5
Name:	Setting the thinning factor
Description:	The user must be able to set the thinning factor. The thinning factor t means that every t^{th} iteration value is used in the output and the rest are discarded.
Rationale:	Even if the proposals aren't accepted and the iteration values remain the same for many iterations the thinning factor ensures that the output contains more variance and describes the posteriori distribution better.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S6
Name:	Setting the blocks
Description:	The user must be able to define which variables belong to the same block; that is, the likelihood of their values is considered as a whole for an update, rather than considering each variable separately. The blocks must be read from a text file.
Rationale:	The user might want to regulate which parameters are updated at the same time.
Priority:	Conditional
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S7
Name:	Setting the update strategy
Description:	The user must be able to set the update strategy. The update strategy defines whether the parameters / blocks are considered for updates in sequential order or whether the next parameter / block to update is chosen at random. If random update is chosen, block weights are used to choose a block to be updated (see S8).
Rationale:	
Priority:	Conditional
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S8
Name:	Setting the weight of the blocks
Description:	It could be possible to define the importance of the blocks. In the update phase the next parameter / block to update could be chosen according to the importance values.
Rationale:	Some parameters might be more important to update than the others.
Priority:	Optional
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S9
Name:	Setting the proposal strategies for variables
Description:	The user must be able to choose the proposal strategy for each parameter. Possible choices are 1. Fixed proposal strategy: The next proposed values for a parameter is taken from its proposal distribution. 2. Random walk: The next proposed value for a parameter is created by adding a value taken from the proposal distribution to the current value of the parameter.
Rationale:	
Priority:	Essential
Stability:	Unstable
Source:	Marko Salmenkivi

Id:	S10
Name:	Proposal distributions
Description:	The user must be able to set the proposal distributions for each parameter group. If the proposal distribution for a parameter is not given, the prior distribution of the parameter is used as fixed proposal distribution (see S9).
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S11
Name:	Setting initial values
Description:	The user must be able to set the initial values for the parameters whose values are not fixed in data. If the initial values are not given, the program must display an error message. (The program doesn't have to generate the initial values by itself.)
Rationale:	The initial values affect the convergence of the simulation run.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S12
Name:	Defining parameters to output
Description:	The user must be able to define which parameters are output during the simulation.
Rationale:	It’s not reasonble to output all parameters. The user might want to gain information about a small subset of all parameters.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S14
Name:	Informing the user about the progress
Description:	While simulating, the program must display information about the progress; that is, the number of the current iteration, or a percentage of iterations done.
Rationale:	The simulations can last for long time, so the user would possibly like to know how much of the simulation is already done and estimate how much time the rest of the simulation might take.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	S15
Name:	Soft stop
Description:	The user could be able to stop the simulation so that the remaining iteration is run and the output files are written so that it's possible to continue the simulation.
Rationale:	If the user stops the appication (for example by pressing Ctrl + c), it might not be possible to continue the simulation and the output files might be only partially written.
Priority:	Optional
Stability:	Stable
Source:	Marja Hassinen

Id:	S16
Name:	Parameters in random walk
Description:	The parameters of proposal distribution can depend on the variable's previous value in random walk proposal strategy.
Rationale:	
Priority:	Optional
Stability:	Stable
Source:	Marko Salmenkivi

5.4 Output requirements

This section lists requirements related to the output of the program.

Id:	OP1
Name:	Writing output into a file
Description:	The program must be able to write output into a file.
Rationale:	Files are much easier to use than standard output. The program output may be thousands of lines.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	OP2
Name:	Output file names
Description:	The user must be able to specify the file names of the output files.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	OP3
Name:	The output
Description:	The output of the program consists of points of the given variables' posterior distributions. The columns contain values of the parameters and the lines represent different iterations.
Rationale:	This is the primary aim of the program.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	OP4
Name:	Information written to output files
Description:	There should be some information about the simulation in the beginning of the output file: the number of iterations and burn-in iterations, the thinning factor and the name of the model (if specified). There should also be information about the names of the variables whose values the output describes. The lines that contain this information must begin with a #-sign.
Rationale:	The user would like to compare the outputs of different simulations. This information helps the user to identify different simulations.
Priority:	Essential
Stability:	Unstable - More or different information might be desired.
Source:	Marko Salmenkivi

Id:	OP5
Name:	Summary of the simulation
Description:	The program should write a summary about the simulation into a separate file. The summary contains the number of proposed changes and the frequency of successful changes for each parameter.
Rationale:	The user migh want to gain information about which parameters are difficult to update.
Priority:	Conditional
Stability:	Stable
Source:	Marko Salmenkivi

Id:	OP6
Name:	File access check
Description:	The program must check if the output file is available to be written before beginning the simulation. If not, the program must print an error message.
Rationale:	It would be very frustrating if instead of results after running a simulation the user would get nothing.
Priority:	Conditional
Stability:	Stable
Source:	Joonas Kukkonen

Id:	OP8
Name:	Continuing the simulation
Description:	The last values of the simulation could be saved into a separate file and it could be possible to automatically use them as initial values for another run of the simulation. (That means that the initial values could be read from a separate file.)
Rationale:	This makes it possible to run a simulation easily in multiple parts.
Priority:	Optional
Stability:	Stable
Source:	Marko Salmenkivi

5.5 General error conditions

Id:	E1
Name:	File not found
Description:	If a file specified as input isn't available (cannot be found or cannot be read), the program must display an error message.
Rationale:	
Priority:	Essential
Stability:	Stable
Source:	Vesa Vainio

Id:	E2
Name:	Reporting syntax errors
Description:	If the file defining the model or a file including simulation parameters is syntactically invalid, the program must display an informative error message describing the nature of the error.
Rationale:	Makes the user's work easier.
Priority:	Optional
Stability:	Stable
Source:	Anni Kotilainen

Id:	E3
Name:	Reporting semantic errors
Description:	If the file defining the model or a file including simulation technical parameters is semantically invalid, the program could display an error message.
Rationale:	
Priority:	Optional
Stability:	Unstable
Source:	Anni Kotilainen

Note: Requirements related to error conditions in output are defined in the chapter "Output requirements" and requirements related to error conditions related to data are defined in the chapter "Data requirements".

5.6 Non-functional requirements

Id:	N1
Name:	Working on Linux
Description:	The program must function correctly on a Linux-based operating system.
Rationale:	Linux is widely used by the intended users of the software.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	N2
Name:	The implementation language
Description:	The efficiency of implementation must be a primary concern in the choice of the implementation language. As a result, Fortran 90/95 has been chosen for the implementation.
Rationale:	The simulations require a large amount of computation and as such require vast amounts of time. This demands efficiency.
Priority:	Essential
Stability:	Stable
Source:	Marko Salmenkivi

Id:	N3
Name:	Parallel computation
Description:	The program could be able to utilize multiple processors for distributed computation.
Rationale:	This allows for greater efficiency.
Priority:	Optional
Stability:	Stable
Source:	Marko Salmenkivi

Id:	N4
Name:	Graphical user interface
Description:	The program could include a graphical user interface.
Rationale:	This would make the program easier to use.
Priority:	Optional
Stability:	Unstable
Source:	Anni Kotilainen

5.7 General requirements

This section contains requirements that do not fall under the other categories, or should be in many of them.

Id:	G1
Name:	Adding comments to definition files
Description:	It should be possible to add comments into the definition files. (Definition files = input files - data files)
Rationale:	
Priority:	Conditional
Stability:	Stable
Source:	Marko Salmenkivi

6 System architecture

This section contains a simple model of the software and its inputs and outputs, followed by a more elaborate model.

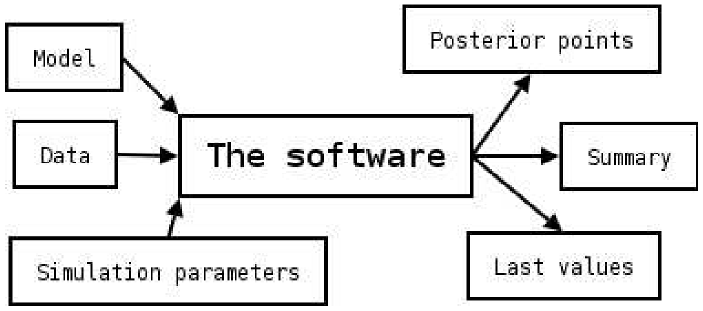

Figure 4: The software, simply put

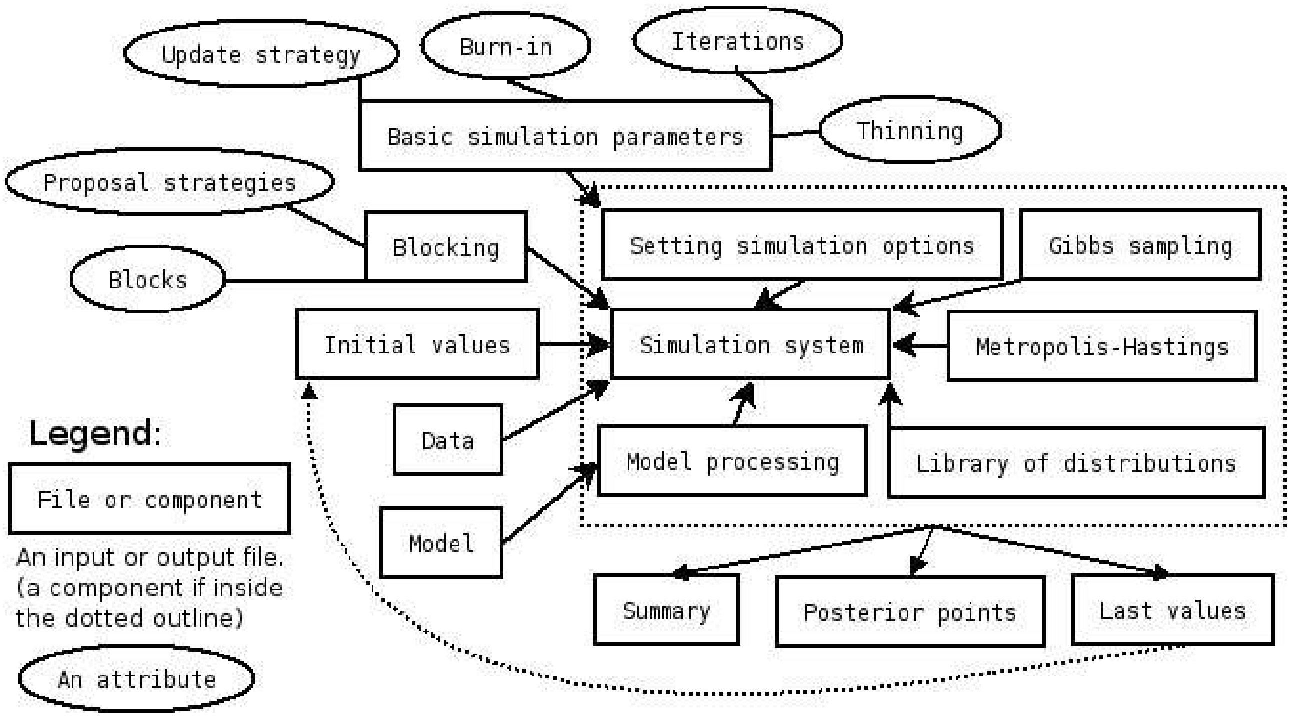

Figure 5: A more elaborate picture of the software

7 System evolution

In the future, the desired format of the output files might change. Also the requirements of the program might change to include:

- More distributions to choose from.
- More algorithm choices in addition to the Metropolis-Hastings and Gibbs sampling.
- More ways to define the number of updates for parameters.
- More proposal strategies for variables.
- Better detection of semantic errors in input files.
- Graphical user interface.

8 References

Bass	*Bassist: A Tool for MCMC Simulation of Statistical Models* http://www.cs.helsinki.fi/research/fdk/bassist/
Fort01	Haataja, J., Rahola, J. and Ruokolainen, J., *Fortran 90/95*. Picaset Oy, Helsinki, 2001. http://www.csc.fi/oppaat/f95/f95.pdf
IEEE98	IEEE Recommended Practice for Software Requirements Specifications
Rand	*Definition of Random Fields in Encyclopedia* http://encyclopedia.laborlawtalk.com/Random_fields
Tode04	Pekka Tuominen: Todennäköisyyslaskenta I
Math05	http://mathworld.wolfram.com/