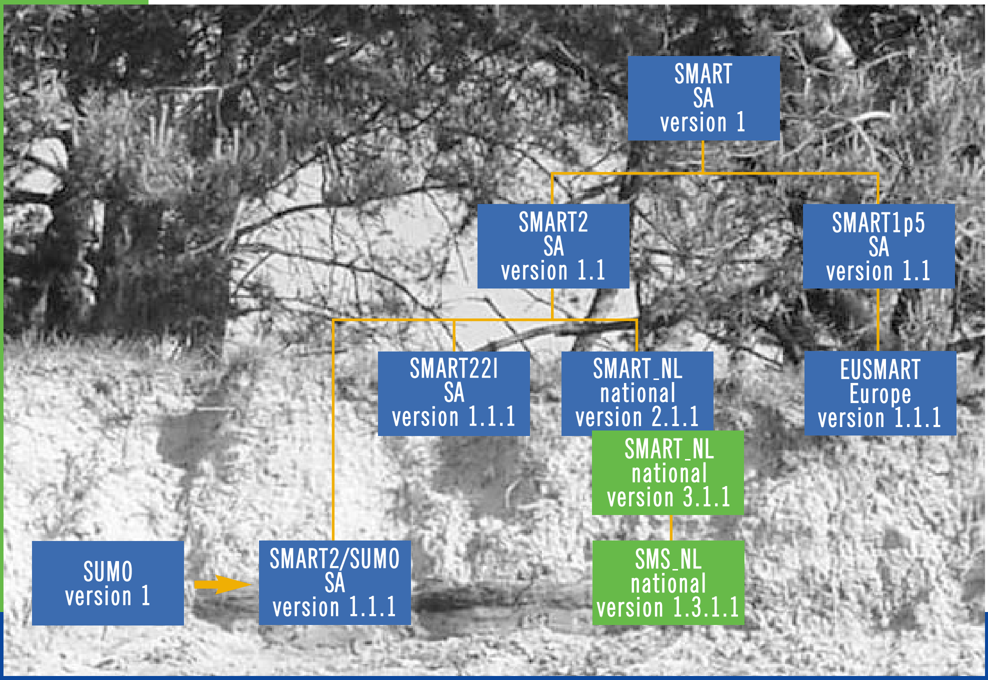
ALTERRA

SMART users guides: SMART_NL & SMS_NL

J.P. Mol-Dijkstra
M. Grobben
J. Kros
G.J. Reinds



SMART SA version 1
SMART2 SA version 1.1
SMART1p5 SA version 1.1
SMART22I SA version 1.1.1
SMART_NL national version 2.1.1
EUSMART Europe version 1.1.1
SMART_NL national version 3.1.1
SUMO version 1
SMART2/SUMO SA version 1.1.1
SMS_NL national version 1.3.1.1

Alterra-rapport 229, ISSN 1566-7197

WAGENINGEN UR

SMART users guides:

SMART_NL & SMS_NL

SMART users guides:

SMART_NL & SMS_NL

J.P. Mol-Dijkstra
M. Grobben
J. Kros
G.J. Reinds

Alterra-rapport 229

Alterra, Green World Research, Wageningen, 2001

ABSTRACT

Mol-Dijkstra, J.P., M. Grobben, J. Kros and G.J. Reinds, 2001. *SMART users guides: SMART_NL* & *SMS_NL*, Wageningen, Alterra, Green World Research.. Alterra-rapport 229. 50 pp. 3 figs.; 9 tables; 9 refs.

Instructions to install and to use the model SMART_NL are given, and how to use the model SMART_NL with SUMO included. Besides, a user manual for configuration management is given.

Keywords: SMART_NL, SUMO, users guide, configuration management

ISSN 1566-7197

This report can be ordered by paying 33,75 Dutch guilders (€ 15,-) into bank account number 36 70 54 612 in the name of Alterra, Wageningen, the Netherlands, with reference to rapport 229. This amount is inclusive of VAT and postage.

© 2001 Alterra, Green World Research,
P.O. Box 47, NL-6700 AA Wageningen (The Netherlands).
Phone: +31 317 474700; fax: +31 317 419000; e-mail: postkamer@alterra.wag-ur.nl

No part of this publication may be reproduced or published in any form or by any means, or stored in a data base or retrieval system, without the written permission of Alterra.

Alterra assumes no liability for any losses resulting from the use of this document.

Alterra is the amalgamation of the Institute for Forestry and Nature Research (IBN) and the Winand Staring Centre for Integrated Land, Soil and Water Research (SC). The merger took place on 1 January 2000.

Project 382-10972.01 [Alterra-rapport 229/HM/05-2001]

Contents

Preface

The model SMART2 has originally been developed to serve as a soil module for the vegetation effect module MOVE. As a tandem SMART/MOVE has been applied on a National scale for the Nature Outlook and the model SMART2 alone for the Environmental Outlook. In order to facilitate application on the National scale SMART2 was embedded in a framework. This framework is called SMART_NL and is developed in Fortran77. Furthermore SMART2 is also coupled with a succession module SUMO. This version is called SMS_NL. For its interaction with data it uses grid-ASCII files for spatial data and plain ASCII files for non-spatial data. The model is installed at both Alterra and the RIVM.

This version of SMART_NL uses mainly the same database tables as the previous version. The only different table is NEDNVK250.QUA. This table now also contains the seepage, and seepage quality class aside from the soil, vegetation and Gt-info. The hydrologic input for MN4 and NV97 is supplied by the LGM-model at a 250 x 250 m^2 level. This change was made, because in contrast to the previous applications of SMART2 on the National scale, where the hydrologic input was given at 1 x 1 km^2 grid, the system now uses hydrologic info (MSW and seepage) at a 250 x 250 m^2 grid. The final output of SMART_NL will be presented on a 250 x 250 km^2 scale.

The configuration- and change management is organised by Alterra b.v. A configuration manager looks after the quality control of the product, and will serve as a helpdesk for the users. The manager will also notify the users in case a new version of SMART_NL is developed, and, if so desired, takes care of the extradition of the executable of the new product. The configuration management is done with the program Visual Source Safe, which uses labels to keep track of major changes. With every new extradition a label will be given to the user, and this label has to be referred to whenever the user calls the helpdesk from Alterra (currently done by: Mireille Grobben: m.s.grobben@alterra.wag-ur.nl). Chapter 5 of this document will give a concise insight in the configuration and change management. The manual that is used by the model developers is added in the annex.

Summary

SMART_NL is the framework, that includes SMART2 in order to facilitate the application of the model on a national scale. SMART2 is a simple one-compartment soil acidification and nutrient cycling model that includes the major hydrological and biogeochemical processes in the vegetation, litter and mineral soil. Apart from pH, the model also predicts changes in aluminium, base cation, nitrate and sulphate concentrations in the soil solution and solid phase characteristics depicting the acidification status, i.e. carbonate content, base saturation and readily available Al content. The model is used for evaluation of the effect of deposition and hydrology scenarios on soil chemistry and on pH and nitrogen availability.

With the integration of the succession module SUMO, there is a feed-back between growth and N-availability. It is possible to model succession from grassland to forest and to investigate management options. SUMO is integrated in SMART2 (Stand Alone) as well as in SMART_NL, which is described in this Users Guide. SMART_NL with SUMO integrated is part of the “Natuurplanner”.

The model SMART_NL (version 2.1.1) has been developed, for the application of SMART2 on a national scale in a VMS / Oracle environment. SMART_NL (2.1.1) has been evolved into SMART_NL (3.1.1), for applications in a Windows NT-environment.

System inputs refer to a specific deposition scenario and hydrological scenario for each grid cell, whereas the model variables and parameters refer to distinguished combinations of generic soil types and generic vegetation types.

The configuration- and change management is organised by Alterra. A configuration manager looks after the quality control of the product, and will serve as a helpdesk for the users. The configuration management is done with the program Visual Source Safe, which uses labels to keep track of major changes.

1 Introduction

This document provides a brief description of the installation and operation of the system, which drives the model SMART2 (Kros et al., 1995) for an application on a national scale: SMART_NL. SMART2 is a simple one-compartment model to assess abiotic site factors in response to emission deposition reductions and reduction in groundwater abstractions on a national scale. It includes geochemical buffer processes (a.o. weathering and cation exchange) and a complete nutrient cycle.

Originally for the application at a national scale SMART2 was embedded in a database environment using ORACLE. Though this version did function adequately, the exchange of this version between RIVM and Alterra platforms was not optimal. Therefore it was decided to build a version that operates under Windows NT and that has no interaction with a certain database management system.

1. This version is flexible in evaluating different atmospheric deposition and hydrology scenarios. It is possible to choose whether a hydrology scenario is evaluated or if the hydrology remains constant. Besides, atmospheric deposition can be used at a 5 x 5 km^2 as well as at a 1 x 1 km^2 resolution.
2. SMART_NL runs 'stand-alone', i.e. without the aid of additional software such as a GIS or database, at both Alterra and RIVM. Furthermore, this version is extended with the succession module SUMO (called SMS_NL), which is also described in this document (Wamelink et al., 2000). SMS_NL is implemented in the 'Natuurplanner' (Alkemade et al., 1997)

The objective of this document is to provide a user's guide, which specifies the requirements for SMART_NL. A model description is given in chapter 2. In chapter 3, the model input en output is described and how to get started. Running the model with the integrated succession module SUMO is described in chapter 4. Chapter 5 involves a description of the configuration- and change management of SMART_NL.

2 SMART_NL

SMART_NL is the framework, that includes SMART2 in order to facilitate the application of the model on a national scale. In this chapter we provide a brief background on both SMART2 and SMART_NL.

2.1 SMART2

SMART2 (Kros et al., 1995) is a simple one-compartment soil acidification and nutrient cycling model that includes the major hydrological and biogeochemical processes in the vegetation, litter and mineral soil. Apart from pH, the model also predicts changes in aluminium (Al^{3+}), base cation (BC), nitrate (NO_3^-) and sulphate (SO_4^{2-}) concentrations in the soil solution and solid phase characteristics depicting the acidification status, i.e. carbonate content, base saturation and readily available Al content. The SMART2 model is an extension of the dynamic soil acidification model SMART (De Vries et al. 1989). The major extensions in SMART2 are the inclusion of a nutrient cycle and an improved modelling of hydrology. The SMART2 model consists of a set of mass balance equations, describing the soil input-output relationships, and a set of equations describing the rate-limited and equilibrium soil processes.

The soil solution chemistry in SMART2 (Figure 1) depends solely on the net element input from the atmosphere (the product of deposition and filtering factor) and groundwater (seepage), canopy interactions (foliar uptake, foliar exudation), geochemical interactions in the soil (CO_2 equilibrium, weathering of carbonates, silicates and/or Al-hydroxides, SO_4^{2-} sorption and cation exchange) and a complete nutrient cycle (litterfall, mineralisation, root uptake, nitrification and denitrification). Processes that are not taken into account are: (i) N fixation and NH_4^+ adsorption, (ii) uptake, immobilisation and reduction of SO_4^{2-}, (iii) complexation of Al^{3+} with OH^-, SO_4^{2-} and $RCOO^-$ and (iv) interaction between the soil solution and the vegetation. Growth of the vegetation and litterfall are modelled by a logistic growth function, which acts as a forcing function. Nutrient uptake is only limited when there is a shortage in the soil solution.

Soil interactions are either described by simple rate-limited (zero-order) reactions (e.g. uptake and silicate weathering) or by equilibrium reactions (e.g. carbonate and Al-hydroxide weathering and cation exchange). Influence of environmental factors such as pH on rate-limited reactions and rate-limitation of weathering and exchange reactions are ignored. Solute transport is described by assuming complete mixing of the element input within one homogeneous soil compartment with a constant density and a fixed depth (at least the root zone). Since SMART2 is a single layer soil model, neglecting vertical heterogeneity, it predicts the concentration of the soil water leaving the root zone. The annual water flux percolating from this layer is taken equal to the annual precipitation, which must be specified as model input. The time step of the model is one year, so seasonal variations are not considered.

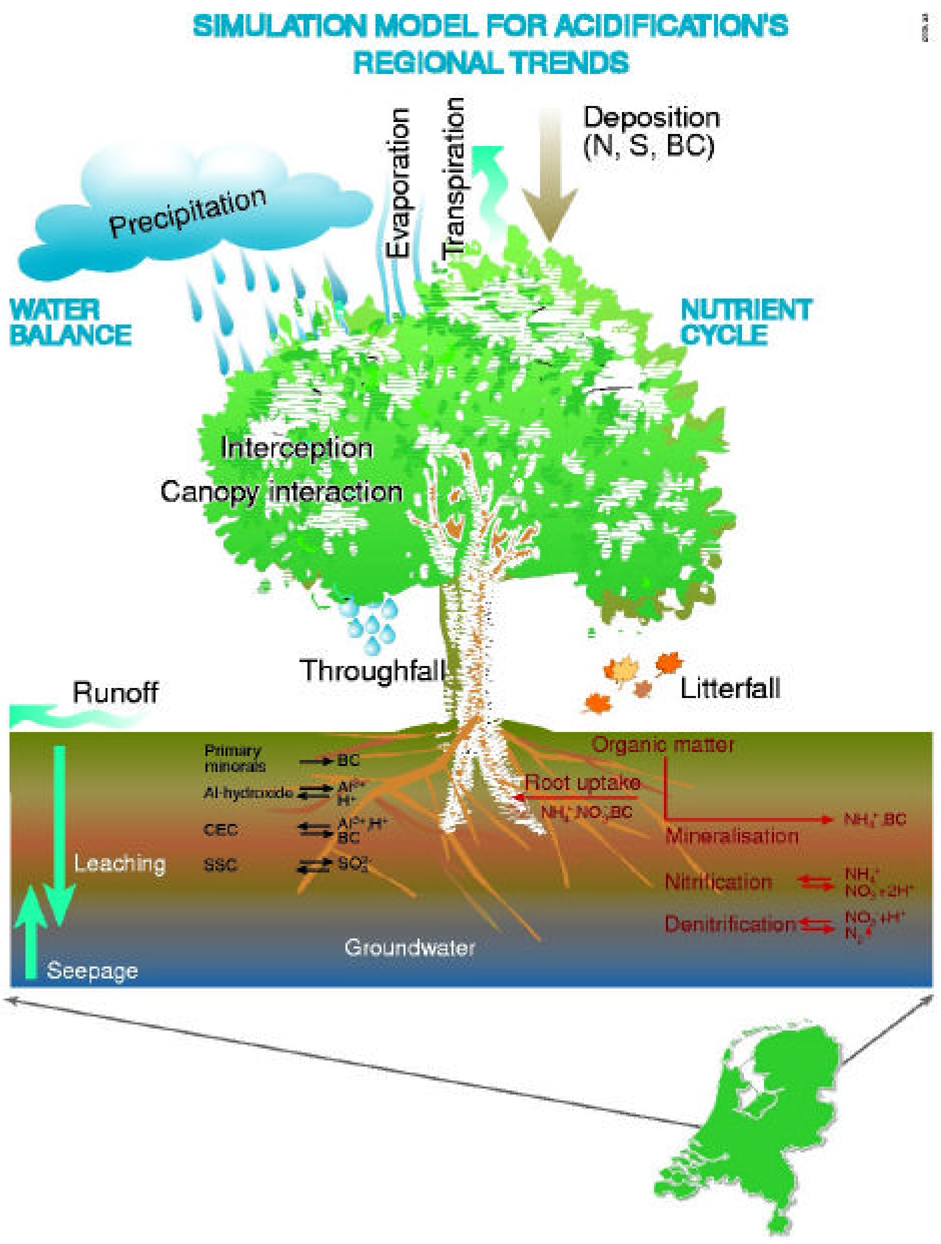


Figure 1 The SMART2 model

Input data for the SMART2 application can be divided in system inputs and initial values of variables and parameters. System inputs are the atmospheric deposition, hydrology and vegetation development.

2.2 SMART_NL

The model SMART_NL (version 2.1.1) has been developed, for application of SMART2 on a national scale in a VMS/Oracle environment. SMART_NL (2.1.1) has evolved into SMART_NL (3.1.1), for applications in a Windows NT-environment. This version operates without a database management system. All interactions will take place through plain ASCII files. For the application of SMART_NL (3.1.1), data are needed with respect to system inputs (driving variables), the initial state of model variables and model parameters. System inputs refer to a specific deposition scenario and hydrological scenario for each grid cell, whereas the model variables and parameters refer to distinguished combinations of generic soil types and generic vegetation types.

All input data are derived either as a function of location (grid cell) or soil type or vegetation type or the combination of vegetation type and soil type. Input data refer to (i) a specific deposition scenario for each grid cell, (ii) model variables and (iii) parameters which are either related to a soil type or a vegetation type or to a combination of both and (iv) a soil map and vegetation map. For the national scale application, the rasterised soil map and vegetation map, representing the dominant soil type and vegetation type for a 250 $\times$ 250-m^2 grid respectively are generalised. Seven soil classes are distinguished and four vegetation types.

In predicting of the long-term impact of atmospheric deposition and seepage on site factors on a national scale, a distinction is made in:

- geo-referenced information on model inputs, varying between each grid cell considered, i.e. (i) the area of soil vegetation combinations, Gt, and seepage fluxes (ii) the deposition of SO_4^{2-}, NO_3^-, NH_4^+, (iii) precipitation, bulk deposition of base cations and Cl^-
- generic information, i.e. average values for initial values of model variables and model parameters for each combination of vegetation type and soil type.

The ER-diagram (data model showing all the tables) is given in figure 2.

Soil, vegetation, seepage and Gt information that describes the ecosystem is available at a 250 $\times$ 250-m^2 grid (ERD: grid cell and Gt). The grid related information, the vegetation and soil-related parameter are stored in ASCII files.

- The deposition values of SO2, NOx and NH3 for a particular scenario are usually derived from a 1 × 1-km2 grid. Deposition scenario's are supplied by the RIVM using the OPS-model for future predictions (Jaarsveld, 1995) and the EDACS-model for the prediction of actual and historical deposition (Erisman, 1991). Both resolutions (1 x 1 and 5 x 5 km2 can be processed by SMART_NL) (ERD: Acid deposition 5000x5000 and 1000x1000).
- The precipitation and bulk deposition values of base cations and Cl- are derived from a 10 $\times$ 10-km2 grid (ERD: precipitation & BC deposition).
- Seepage and Gt was derived from the National Groundwater Model (LGM; Pastoors, 1993), with a resolution of 250 $\times$ 250-m2 (in the previous application

a resolution of 1 × 1 km^2 was used, Kros et al., 1995) (ERD: hydrology scenario). Seepage quality is derived from a 1 x 1 km^2 grid (based on LKW, Bolsius et. al, 1994).

– Information, concerning deposition scenarios and hydrological scenarios is available in grid-ASCII format. These files are converted into direct access files for the use in SMART_NL with the program 'conv_asc_dac.exe', which is described in section 3.4. Model output is stored in ASCII-tables (Section 3.5)

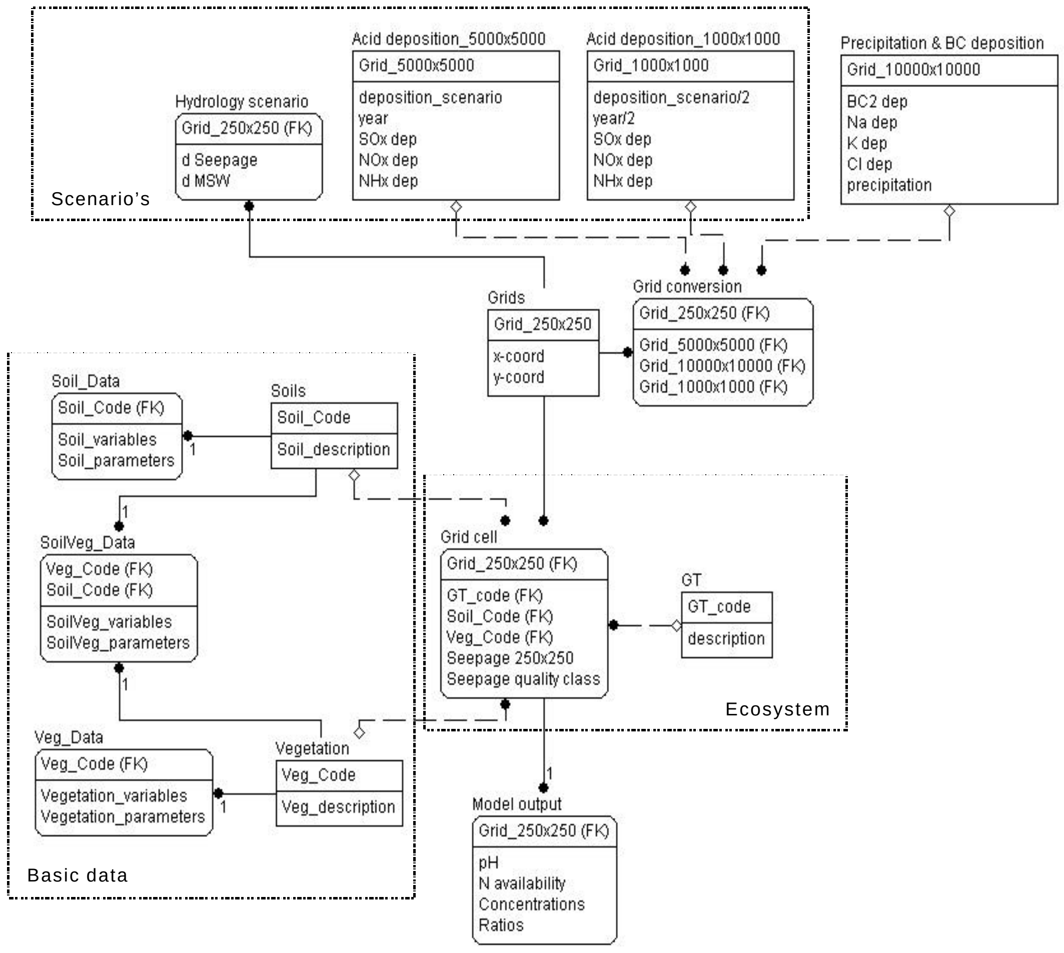


Figure 2 Entity Relation diagram of SMART_NL

3 Program use

For the installation all files need to be present in the appropriate directories (see paragraph 3.1 and conf_sel.dat). In Annex 1 an overview is given of all required files. The general dataflows in SMART_NL are described in Figure 3. The input files that are directly for SMART_NL belong to the boxes within the fat edged part of Figure 3.

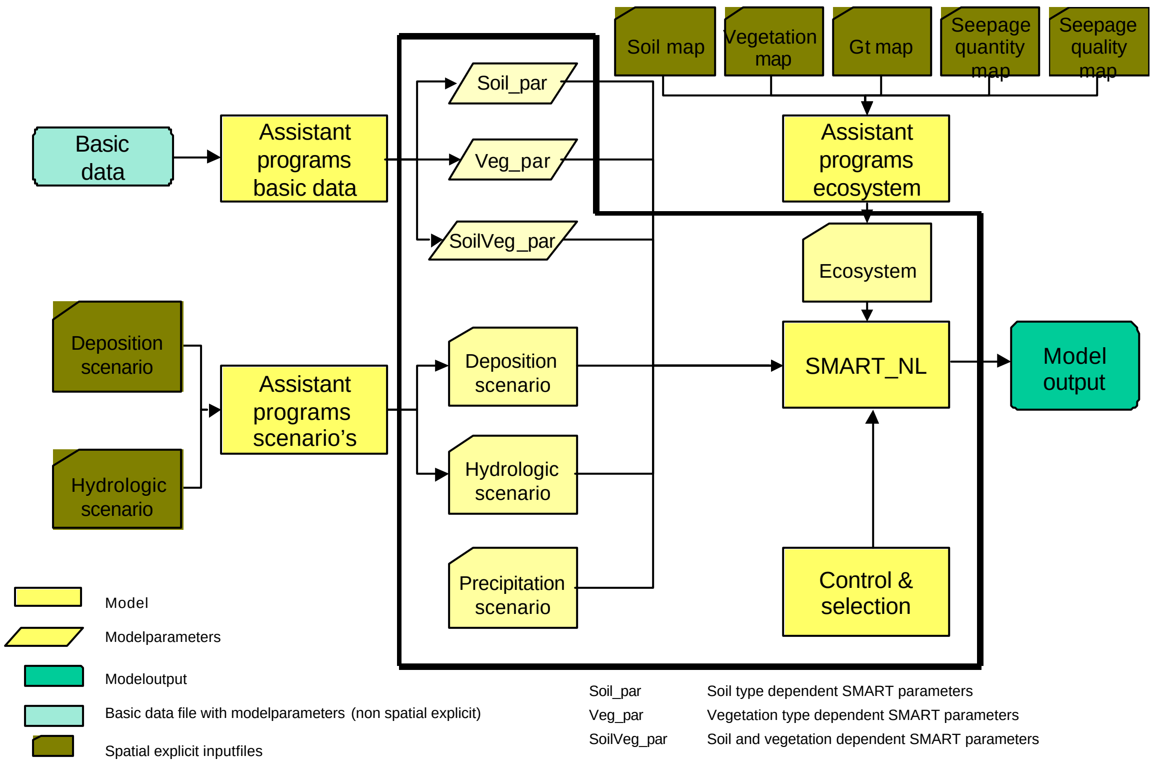

Soil map
Vegetation map
Gt map
Seepage quantity map
Seepage quality map
Soil_par
Veg_par
SoilVeg_par
Basic data
Assistant programs basic data
Assistant programs ecosystem
Ecosystem
Deposition scenario
Hydrologic scenario
Assistant programs scenario's
Deposition scenario
Hydrologic scenario
Precipitation scenario
SMART_NL
Model output
Control & selection
Model
Modelparameters
Modeloutput
Basic data file with modelparameters (non spatial explicit)
Spatial explicit inputfiles
Soil_par Soil type dependent SMART parameters
Veg_par Vegetation type dependent SMART parameters
SoilVeg_par Soil and vegetation dependent SMART parameters

Figure 3 Dataflow diagram of SMART_NL

The program-input files can be divided in:
- files with control and selection information (see paragraph 3.1)
- files with SMART2 parameters (see paragraph 3.2)
- spatial information (see paragraph s 3.3 and 3.4)

3.1 Control and Selection

SMART_NL is controlled by two ASCII-files:
- **conf_sel.dat:** is the main control file with reference to input files and control variables for the simulation (period, depth, output variables, etc.)
- The user is able to specify the following in the *conf_sel.dat* file:
  - soil-vegetation combinations to be evaluated;
  - simulation period;

  – model outputs to be stored.
- **conf_inp.dat**: configuration file with full name, path and filename of non-spatial explicit filenames

3.1.1 Conf_sel.dat

The *conf_sel.dat* is the file that will be the most frequently used file. It contains information about the users' choice regarding the selection of scenarios, simulation period, output variables etc.

The format of the *conf_sel.dat* file looks as follows:

```
 [ * ... ]*
@DATAFILES = [pathname]filename
@CODEDIR = [pathname]
@OUTPUTDIR = [pathname]filename
@LABEL = CCC
@CONSTANT_GROWTH = [YES/NO]
@SOILS = s [,s]*
@DEPTH = DRZ
@VEGETATIONS = v [,v]
@GT = n [,n]*
@ACID_SCEN = [1/2/3/4/5/6/7/8/9/0]
@HYDR_SCEN = [1/2/3/4/5/6/7/8/9/0]             (0 = no scenario will be implied)
@START_YEAR = n
@END_YEAR = n
@OUTPUT_YEARS = n [,n]*
@OUTPUT_VARS =  o [,o]*
```

Lines preceded by * are skipped (as comments).
Blank lines are skipped.

Explanation of *conf_sel.dat*:

– **@datafiles**
absolute file name (i.e., *conf_inp.dat*) of the configuration file which holds the filenames of the SMART parameter files.

- **@codedir**
pathname in which the file the input files are stored. These are the spatial data and the code files. The code_files contain the possible codes of SMART parameters.

The following code_files should be available in *codedir* directory:

soil_cod.dat	(file with codes, see table 1)
sv_cod.dat	(all possible combinations of soil- and vegcodes, e.g. SPHEA)
vars_cod.dat	(file with all the output variables, e.g. pH, SOx etc.)
veg_cod.dat	(file with codes, see table 2)
soil_par.dat	(file with the soil variables, e.g. pco2)
sv_par.dat	(file with soil-veg variables, e.g. thickrz)
veg_par.dat	(file with veg variables, e.g. iagevg, iagelt)

The following files with spatial data should be available in *codedir* directory:

nednvk250.qua	(ecosystem table)
rain.dac	(precipitation data)

bcdep.dac	(base cation deposition)
nhxdep.dac	(NH_x-deposition)
noydep.dac	(NO_y-deposition)
soxdep.dac	(SO_x-deposition)
hydrology.dac	(only if hydr_scen > 0) (seepage and water table change)
nhxdep.idx	(index file for looking up right position)
rain.idx	(index file for looking up right position)
nhxdep.hdr	(header file for checking grid size of acid deposition file)
noydep.hdr	(header file for checking grid size of acid deposition file)
soxdep.hdr	(header file for checking grid size of acid deposition file)

– **@soils**
  Soil types must always be named explicitly, using the code separated by space or comma. The selection off all soils can be achieved by using a wildcard ('*' or 'all'). See table 1 for the soil type options.

Table 1 Possible soil type options

Number	Code	Soil type
1	SP	Sand Poor
2	SR	Sand Rich
3	SC	Sand Calcareous
4	CN	Clay Non-calcareous
5	CC	Clay Calcareous
6	LN	Loess Non-calcareous
7	PN	Peat Non-calcareous
	* or all	All soil types

– **@depth**
  indication of the depth [m] at which the model should be evaluated:
  drz: depth of the root zone (as indicated in SV_DATA file)
  [0.1, 0.2, .., 1.0]: any multiple depth of 0.1 m in [0.1, 1.0]

– **@vegetation**
  Vegetation types must always be named explicitly, using the code separated by space or comma. The selection off all vegetation types can be achieved by using a wildcard ('*' or 'all'). See table 2 for the vegetation type options.

Table 2 Possible vegetation type options

Number	Code	Vegetation type
1	DEC	Deciduous
2	SPR	Spruce
3	PIN	Pine
4	HEA	Heather
5	GRP	Grass (nutrient) Poor
	* or all	All vegetation types

– **@Gt**
  groundwater classes (Gt) to be evaluated should be named explicitly by using integer numbers [1,5], separated by a space or comma.

– **@acid_scen**
acidification scenario to be evaluated. The numbers are not directly coupled with certain scenarios. The names of the deposition files are fixed, which means that alternative deposition files must be renamed to the fixed filenames by hand. The number will be found in the name of the output file.

– **@hydr_scen**
hydrology scenario to be evaluated. This uses the following codes: 0 if no scenario is used, and > 0 when a scenario will be used. The number will be found in the name of the output file.

– **@start_year**
a 4 digit indication of the starting year

– **@end_year**
a 4 digit indication of the ending year

– **@outputyears**
years (maximum is 10) at which output should be stored in the database table. Years given with 4 digits separated by space or comma.

– **@outputvars**
output variable to be stored (at time specified by outputyears) in database table, using the code separated with a space or comma.

See for the possible model outputs: table 3.

Table 3 Possible model outputs

Code	Model output	Unit
CAL	Al concentration	$mol_c\ m^{-3}$
PH	pH	
CBC1	BC1 concentration	$mol_c\ m^{-3}$
CBC2	BC2 concentration	$mol_c\ m^{-3}$
CCL	Cl^- concentration	$mol_c\ m^{-3}$
CNH4	NH_4^+ concentration	$mol_c\ m^{-3}$
CNO3	NO_3^- concentration	$mol_c\ m^{-3}$
CSO4	SO_4^{2-} concentration	$mol_c\ m^{-3}$
CORG	$RCOO^-$ concentration	$mol_c\ m^{-3}$
ALOX	Amount of amorf Al_{ox}	$mmol_c\ kg^{-1}$
CACARB	Amount of $CaCO_3$	$mmol_c\ kg^{-1}$
FBC	Base saturation	$mol_c\ mol_c^{-1}$
FAL	Fraction Al^{3+} at CEC	$mol_c\ mol_c^{-1}$
FH	Fraction H^+ at CEC	$mol_c\ mol_c^{-1}$
CALBC2R	Al^{3+} to BC2 ratio	$mol_c\ mol_c^{-1}$
CNRATLT	C to N ratio in litter	$g\ g^{-1}$
NAVAIL	N availability	$mol_c\ ha^{-1}\ a^{-1}$
NOUT	N leaching	$mol_c\ ha^{-1}\ a^{-1}$
NMIN	N mineralisation	$mol_c\ ha^{-1}\ a^{-1}$
NDEP	N deposition	$mol_c\ ha^{-1}\ a^{-1}$

NINOUT	N_{in} to N_{out} ratio	-
THICKLT	Thickness litter layer	m
GVG	Mean spring watertable	m
SEEP	Seepage flux	m a^{-1}

Example:
```
* File name: conf_nv1
* SELECTIE FILE SMART_NL tbv Natuurverkenning 1
*
* Run met mediaan
@DATAFILES = C:\data\input_nv1.dat
@LABEL = TST
@CODEDIR = c:\data\…
@CONSTANT_GROWTH = YES
@SOILS = ALL
@DEPTH = DRZ
@VEGETATIONS = ALL
@GT = 1,2,3,4,5
@ACID_SCEN = 4
@HYDR_SCEN = 2
@START_YEAR = 1985
@END_YEAR = 2020
@OUTPUT_YEARS = 1995, 2020
@OUTPUT_VARS =  PH NAVAIL GVG SEEP
```

3.1.2 Conf_inp.dat

The following control and selection criteria will be defined in the *conf_inp.dat* file:

– datafiles
  absolute file name of the configuration file which holds the filenames of the SMART parameter files. The format of the configuration file looks as follows:

```
[ * ...]*
SOIL_DATA = [pathname]filename
VEG_DATA  = [pathname]filename
SV_DATA   = [pathname]filename
```

Example:
```
* File name: conf_nv1
* FILE INLEZEN SMART PARAMETERS tbv Natuurverkenning 1
* For default filename use '*'
SOIL_DATA = C:\data\smrt_med.out
VEG_DATA  = C:\data\smrt_nv1.out
SV_DATA   = C:\data\smrt_sv.out
```

3.2 SMART2 parameters

SMART2 parameters are parameters which are related to soil and/or vegetation. These parameters are derived from measurements by mainly GENSTAT programs that exist at Alterra. These programs bridge the gap between plain monitoring data and the model parameters and initial values of state variables in the non-spatial explicit model input files. These programs are not described in this document, but we suffice with the description of the input files with the non-spatial data, that are directly used by SMART_NL.

3.2.1 Soil parameters

Data used for the soil parameters of the seven soil types are provided in the *SOIL_DATA* file (see example below, path and filename are given in the *conf_inp.dat* file). This file contains for each soil type a block of soil parameters, which are either depth dependent or not. For the depth dependent parameter values must be specified for the whole covering soil layer between the soil surface and specified depth. This is necessary, because SMART2 can only perform calculations for one soil compartment. For the depth independent parameters, only one value must be specified in the first column. For the description of the necessarily soil parameters we refer to Kros et al. (1995).

The format of the *soil_data* file looks as follows:

[* ...]*
{@SOIl=SOIL_ID$^{1)}$
 @DEPTH$^{2)}$ **unit**$^{3)}$ **0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 1.00**
{Soil parameter$^{3)}$ **unit**$^{3)}$ **x [x x x x x x x x]}*}+**

$^{1)}$ See Table 1 for possible soil types
$^{2)}$ In the example specified required parameters and depths are mandatory
$^{3)}$ See example for units

Example:
```
 * File: soil.dat
 * TEST FILE SOIL DATA
 @SOIL=SP
 @DEPTH    [m]            0.10     0.20     0.30     0.40     ..  1.00
 frpp      [-]            0.1000
 rho       [g/cm3]        1.371    1.408    1.416    1.437    ..  1.461
 rholt     [g/cm3]        0.1300
 theta     [m3/m3]        0.1740   0.1547   0.1483   0.1422   ..  0.1399
 Cacarb    [mmolc/kg]     0        0        0        0        ..  0
 cec       [mmolc/kg]     23.35    18.06    24.00    14.18    ..  9.350
 omc       [kg/kg]        0.03333  0.02595  0.03100  0.02048  .   0.01571
 Cnrat     [kg/kg]        22.03    21.02    20.59    20.95    .   21.37
 gtal      [log(mol/l)]   0.09300  0.1132   0.5614   0.4871   .   0.7871
 gth       [log(mol/l)]   3.727    3.783    3.745    3.867    .   3.900
 Gibbs     [log(mol/l)]   6.835    7.282    7.363    7.696    .   8.174
 fbc       [-]            0.1071   0.08887  0.06000  0.07861  .   0.07267
 Rfnigvgmx [-]            0.3000
 Fnitmx    [-]            1.000
 anit      [-]            0.1250
 bnit      [1/m]          1.750
 z1nit     [m]            0.1000
 z2nit     [m]            0.5000
 Fdenmx    [-]            0.9000
 Aden      [1/m]          0.5000
 Rfdegvgmn [-]            0.2500
 alox      [mmolc/kg]     45.00    79.50    106.0    83.25    ..  101.4
 Adsmax    [mmolc/kg]     0.9000   1.590    2.120    1.665    ..  2.028
 adsh      [molc/m3]      1.000
 Coacid    [mmolc/m3]     0
 Pkpar1    [-]            2.500
 Pkpar2    [-]            0.9000
 Pkpar3    [-]            0.03900
 pco2      [-]            33.00
 bc2w      [molc/m3/a]    0.01143
 naw       [molc/m3/a]    0.00929
 kw        [molc/m3/a]    0.00929
 @SOIL=SR
 @DEPTH    [m]            0.10     0.20     0.30     0.40     ..  1.00
 frpp      [-]            0.2000
```

3.2.2 Vegetation parameters

Data used for the vegetation parameters of the five vegetation types are provided in the *VEG_DATA* file (see example below, path and filename are given in the *Conf_inp.dat* file). This file contains for each vegetation type a block of vegetation parameters. For the description of the required vegetation parameters we refer to Kros et al. (1995).

The format of the *veg_data* file looks as follows:

```
[ * ... ]*
{@VEG=VEG0_ID1)
{Vegetation parameter3) unit3)      x  }*}+
```

1) See Table 2 for possible vegetation types
2) In the example specified required parameters are mandatory
3) See example for units

Example:

```
* File: SMRT_NV1.OUT
**      961217 Hans Kros
**      gebaseerd op SMRT_VEG.OUT (960124)
**      iagevg (PIN, SPR, DEC) --> 70, 60, 80
**      ctXst(HEA) = ctXst(DEC)
**      ctXst(GRP) = ctXlv(GRP)
@VEG=PIN
Iagevg     [a]         70.00
iagelt     [a]         80.00
ffso2      [-]         1.400
ffnh3      [-]         1.300
ffnox      [-]         0.8500
fdd        [-]         2.500
fint       [-]         0.3000
fnh4fu     [-]         0.3000
fhfu       [-]         0.3000
fKfe       [-]         0.6300
amlf       [kg/m2/a]   0.4125
ncf        [-]         0.5000
fre        [-]         0.3600
frrtlt     [-]         0.2500
ctbc2lv    [%]         0.3100
ctKlv      [%]         0.6000
Ctnitlvmn  [%]         1.500
Ctnitlvmx  [%]         2.500
Ctnitst    [%]         0.1200
ctbc2st    [%]         0.1100
ctKst      [%]         0.05000
fmifl      [-]         0.8000
kmilt      [1/a]       0.05000
@VEG=SPR
iagevg     [a]         60.00
iagelt     [a]         80.00
ffso2      [-]         1.600
ffnh3      [-]         1.500
ffnox      [-]         1.000
```

3.2.3 Soil/Vegetation parameters

Model parameters that depend on both soil type and vegetation type refer to the depth of the root zone, transpiration rate and growth parameters. Values used for

each combination of soil type and vegetation type are required in the *SV_DATA* file. For the description of the required vegetation parameters we refer to Kros et al. (1995).
The format of the *SV_DATA* file looks as follows:

```
[ * ... ]*
{@SV=SOIL_VEGETATION_ID1)
 SV parameter3) unit3)      x  }*
```

[1)] See Table 1and Table 2 for possible soil vegetation combinations. The SOIL_VEGETATION_ID consist of a concatenation of SOIl_ID and VEGETATION_ID.
[2)] In the example specified required parameters are mandatory
[3)] See example for units

Example:

```
* File: SMRT_SV.OUT
* TEST FILE SOIL VEGETATION DATA
 @SV=SPPIN
 thickrz     [m]              0.7000
 tr          [m/a]           0.2760
 krgl        [1/a]          0.06700
 t05         [a]              40.00
 amstmx      [kg/m2]           22.24
 @SV=SPSPR
 thickrz     [m]              0.7000
 tr          [m/a]           0.2960
 krgl        [1/a]          0.07200
 t05         [a]              38.00
 amstmx      [kg/m2]           25.02
 @SV=SPDEC
 thickrz     [m]              0.7000
 tr          [m/a]           0.3260
 krgl        [1/a]          0.08800
 t05         [a]              50.00
 amstmx      [kg/m2]           28.77
 @SV=SPHEA
 thickrz     [m]              0.2000
 tr          [m/a]           0.3350
 krgl        [1/a]           0.1500
 t05         [a]              10.00
 amstmx      [kg/m2]           1.400
 @SV=SPGRP
 thickrz     [m]              0.2000
 tr          [m/a]           0.4000
 krgl        [1/a]           0.1500
 t05         [a]              5.000
 amstmx      [kg/m2]          0.5000
```

3.3 Ecosystem table

The generation of this file is described in annex 2. Here we suffice with the description of the input file with the spatial data, describing the ecosystem, that is directly used by SMART_NL.

Table 4 Content of the file nednvk250.qua

Variable	Unit
x-coordinate (x)	m
y-coordinate (y)	m
Soil type (soil)	-
Reference groundwater table class (Gt.)	-
Vegetation type (veg)	-
Reference seepage[1)] (seep)	mm·day^{-1}

Seepage quality class (qual)	-

1) negative value refers to upward seepage

Example:
```
x        y        soil   gt   veg   seep     qual
193750   310250   CN     5    DEC   0        0
182000   323250   LN     5    DEC   0        2
163250   369250   SP     4    SPR   0        0
165750   370750   SR     3    DEC   -0.068   2
14750    377750   SC     5    GRP   0        4
15000    378000   SC     5    GRP   0        2
167250   384000   SP     3    PIN   0        1
85000    388250   SR     3    DEC   0        1
```

3.4 Deposition and hydrology scenarios

The temporal trends and soil solution concentrations and fluxes predicted by SMART2 are driven by scenarios for quantity and quality of the atmospheric deposition, seepage and related changes in phreatic water level. The scenarios considered only related to the quality of atmospheric deposition and the quantity of seepage (Table 5).

Table 5 Considered scenarios with respect to quality and quantity

Aspect	Deposition	Seepage
Quantity	Constant[1)]	Variable
Quality	Variable[2)]	Constant

1) Refers to precipitation

2) Refers to SO_x, NO_x and NH_3. Atmospheric deposition of base cations and chloride was assumed to be constant

Scenarios for the quantity of seepage were generated with the National Groundwater Model for the Netherlands (LGM, Pastoors, 1993). The LGM was developed and used for the National Policy Plan on Drinking Water and Industrial Water Supply to assess the effect of changes in groundwater abstractions on the geo-hydrological system. These effects are expressed in terms of hydrological variables: changes in phreatic water level, groundwater heads in deeper aquifers, fluxes between the upper boundary of the geo-hydrological system and the first aquifer, and fluxes across the aquitard.

SMART_NL uses deposition scenarios for NH_x, NO_x and SO_x calculated by the OPS-model (van Jaarsveld1995),that are provided by RIVM on a national scale for 1×1 km^2 or 5×5 km^2 grids. The input of deposition scenarios occurs by integration of The files delivered by OPS-output, are in grid-ascii format. These deposition files are integrated and converted into direct access files, for immediate use by SMART_NL.

The direct access files have fixed names (Table 6) Files for precipitation and deposition of base cations have the same format, although there are no scenarios for rainfall and base cation deposition. To construct these direct access files, a conversion program called 'conv_asc_dac' is used (see 3.4.1).

Table 6 Names of the direct access files for SMART_NL

Description	Filenames
Precipitation	rain.dac
Deposition of base cations	bcdep.dac
Hydrology scenario	hydrology.dac
NOx-deposition	noydep.dac
NH3-deposition	nhxdep.dac
SOx-deposition	soxdep.dac

3.4.1 Conversion of grid-ASCII files into direct access files

The conversion will be executed by the Fortran-program conv_asc_dac.exe and runs under MS-DOS.

input files

listfile (*.lst)	file with names of the grid-ASCII files to be converted

outputfiles

filename.dac	direct access file with fixed names, include deposition data
filename.hdr	header file, required for identification of years of deposition
filename.idx	indexfile, required for searching the right deposition

The conversion command is: conv_asc_dac [listfile] [outputfile] [x],
where:
[listfile] is the name of the listfile,
[outputfile] is the name of the outputfile,
[x] is the type of data (h/o):
[h]= the hydrologic scenario (data offered in a 250 x 250 m^2 grid)
[o]= other (data offered in a 1 x 1 or 5 x 5 km^2 grid)

Precipitation

Listfile for precipitation:
RAIN.ASC

? The conversion command is: conv_asc_dac [rain.lst] [rain.dac] [o]

Base cations deposition

The listfile for base cation deposition has the following format, with a fixed order of grid-ASCII files :

ca_dep.asc
mg_dep.asc
k_dep.asc
na_dep.asc
cl_dep.asc

? The conversion command is: conv_asc_dac [bcdep.lst] [bcdep.dac] [o].

Acid deposition
The required names of the deposition files are given in Table 6.

The names of all the grid-ASCII-files are free, except for the last four digits of the filenames for NOx, NHx and SOx, which have to represent the deposition year. The program has to perform separate runs for NOx, NHx and SOx, so each parameter needs his own lst-file. All available years will be put into the direct access file per component.

Example of a listfile:
```
dep_mv5\nox-1980.asc
dep_mv5\nox-1990.asc
dep_mv5\nox-1997.asc
dep_a\nox-2010.asc
dep2020\nox-2020.asc
dep2030\nox-2030.asc
dep2030\nox-2050.asc
```

? The conversion command is: conv_asc_dac [depnox.lst] [noydep.dac] [o]

Hydrology
If hydr_scen > 0, a file with a hydrology-scenario, named hydrology.dac, is required.

The listfile has the following format, with a fixed order of grid-ASCII files:
```
NEDDKWEL.ASC
NEDDGVG.ASC
```

? The conversion command is: conv_asc_dac [hydrology.lst] [hydrology.dac] [h]

3.5 SMART_NL output

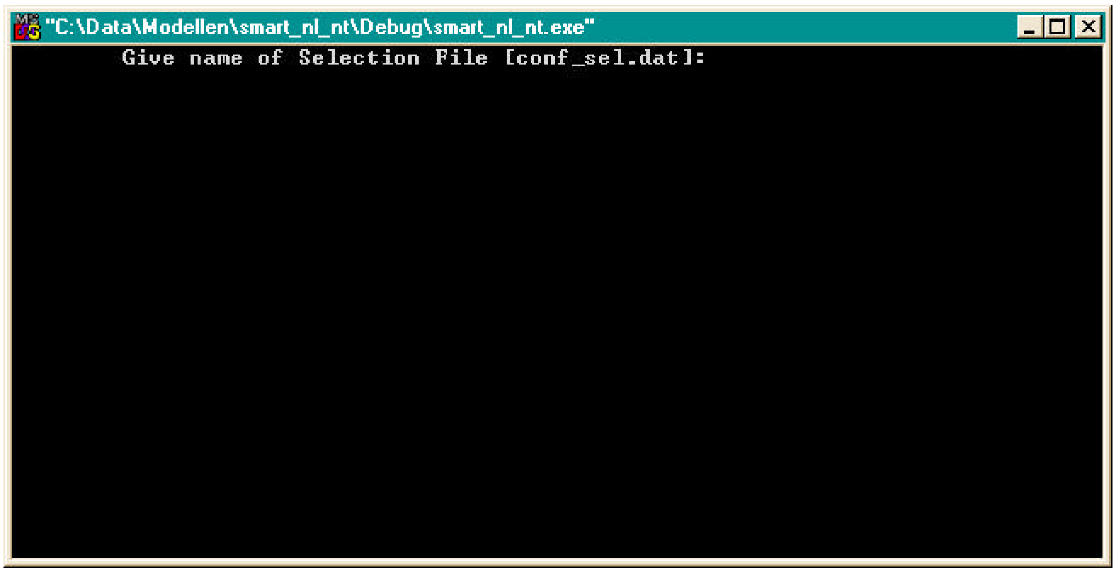
The following table is a representation of the output of SMART2.

```
X_coord  y_coord  Soil_code  gt_code   veg_code    PH_1995    PH_2020    SEEP_1995
239000   571250   SP         4         DEC         4.18       4.20       0.00
239250   571250   SR         4         DEC         4.18       4.31       0.00
238750   571000   SR         4         GRP         3.93       3.95       0.00
239000   571000   SR         5         GRP         3.93       3.95       0.00
239250   571000   SR         5         DEC         4.18       4.31       0.00
238000   570750   PN         2         GRP         6.66       6.66       2.28
238750   570750   SR         4         PIN         7.30       7.32       1.42
239000   570750   SR         5         PIN         4.10       4.13       0.00
239250   570750   SR         5         PIN         4.10       4.13       0.00
238750   570500   SR         4         PIN         7.30       7.32       1.42
239000   570500   SR         5         PIN         4.10       4.13       0.00
239250   570500   SR         5         DEC         4.20       4.26       0.00
239500   570500   SR         4         GRP         3.91       3.95       0.00
241250   570500   SR         5         DEC         4.21       4.37       0.00
238250   570250   PN         2         GRP         6.44       6.45       0.713
```

The output can be used for several other programs, such as Excel or Arc-view.

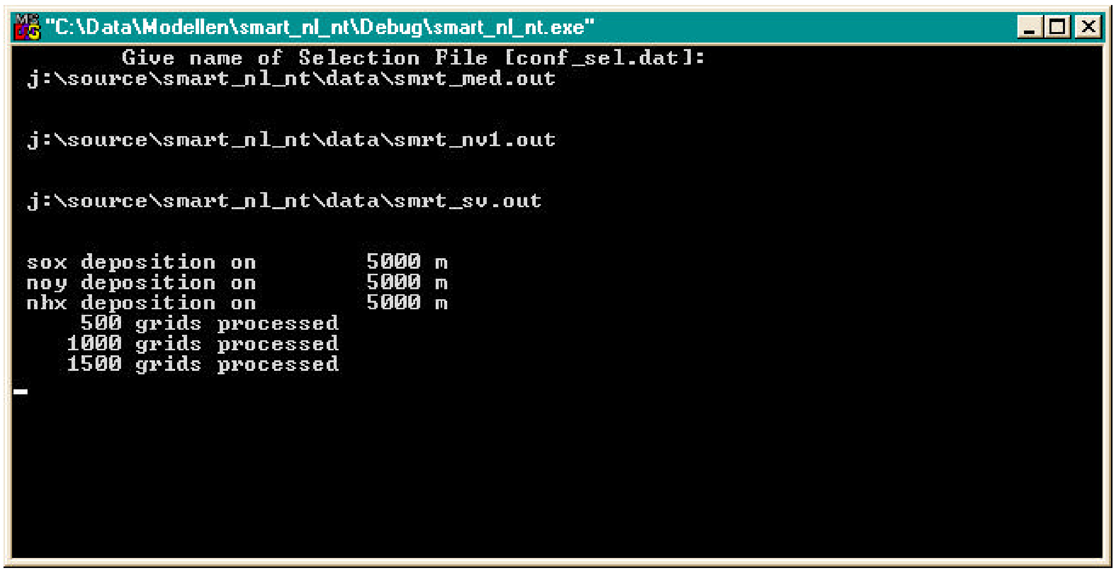
3.6 Running the model

To start the model-run, you can double-click the name of the model or type the name of the executable in a DOS-box. After doing this the next screen will appear:



```
        Give name of Selection File [conf_sel.dat]:
```

The program now interactively asks for the file with control and selection information. Default is *conf_sel.dat*. If ENTER is pressed, the program will use this input file and echoes the filenames with the SMART2 parameters and the grid size of the deposition files. During the run the number of processed grid cells appears on your screen:



```
        Give name of Selection File [conf_sel.dat]:
 j:\source\smart_nl_nt\data\smrt_med.out


 j:\source\smart_nl_nt\data\smrt_nv1.out


 j:\source\smart_nl_nt\data\smrt_sv.out


 sox deposition on         5000 m
 noy deposition on         5000 m
 nhx deposition on         5000 m
     500 grids processed
    1000 grids processed
    1500 grids processed
_
```

At the end, the model gives the following message:

```
*** smart_nl, normal successful completion ***
```

4 SMART_NL with integrated succession module SUMO: SMS_NL

In 1998 the succession module SUMO was developed in order to model succession dependent on N availability (Wamelink et al., 2000). SUMO was integrated in SMART_NL, which resulted in the version of SMART_NL, which is called SMS_NL. SMS_NL (1.3.1.1) is implemented in the 'Natuurplanner', a system that makes it possible to evaluate nature development in response to, for instance deposition and/or (water) management. SUMO is only suitable for the Netherlands and similar situations, since the parameterisation is performed for Dutch situations.

The existing growth function in SMART_NL was replaced by SUMO, which calculates biomass production, litter production and N uptake. The biomass production is calculated for the functional vegetation types herbs, dwarf shrubs, shrub and forest. The forest is divided in pioneer trees and climax trees (per tree species). The succession stage is defined by the amount of biomass per functional vegetation type. In SUMO 12 vegetation structure types exist, which all have its own composition of functional vegetation types and their own growth parameters.

Running SMS_NL

To run SMS_NL, the same conf_sel.dat can be used. The possible model outputs (Table 3) are extended by 2 output variables: 'VEG' and 'BIOM'. VEG is the SUMO-vegetation structure type and BIOM is the total above ground biomass (tons.ha^{-1}). These two variables need to be present in vars_cod.dat (see section 3.1.1).

Input files

Extra input files are needed when running SMS_ NL (see Table 7). These files have to be in the directory that is given in *conf_sel.dat* behind *@codedir* (see Section 3.1.1).

Table 7 Extra input files for SMS_NL

File name	Content
Vegout.txt	vegetation and management map
Biomini.txt	initial biomass and N content per organ per functional vegetation type for 54 combinations of age classes and vegetation types (corresponding the file numbers in vegout.txt)
Paramete.txt	vegetation structure type related parameters
Terugtre.txt	re-allocation of nitrogen per organ and per N content class

Table 8 gives the content of the vegetation and management map *vegout.txt*. **This map must have the same order of grid cells as in the file grid_s_v.dat!** The vegetation and management is available at 250 × 250 m^2 grid. However, within each grid cell several SUMO vegetation structure types may occur, which means that in the map several lines with the same x- and y-coordinates may occur. Since the ecosystem map and the vegetation map must have the same order of grid cells, *grid_s_v.dat* needs to be synchronized to the vegetation map *vegout.txt*.

Table 8 Content of vegout.txt

Code	Description	Unit
XCOORD	x-coordinate	-
YCOORD	y-coordinate	-
OPP	area	ha*10
BEM	fertilisation (not operational yet)	ton N·ha^{-1}·a^{-1}
VEGTYPE	SUMO vegetation structure type	-
BEHEER	management in first year: 0 = no management 1 = mowing 2 = sod cutting 3 = grazing (simulated as mowing) 4 = natural forest management (not operational yet) 5 = clearcut management (forest) 6 = coppice management (forest)	-
PLAGGEN	frequency of sod cutting	A
STROOISEL	sod cutting [1] or not [0]	-
MAAIEN	frequency of mowing	times·a^{-1}
LEEFTIJD	age of the vegetation	A
FILENR	number of file with parameters per age class per vegetation type [1...50]	-
PBOOM	pioneer tree species	-
CBOOM	climax tree species	-

Example of vegout.txt

```
xcoord  ycoord  opp  bem  veg  beh  pla  str  x  leeftijd  filenr  pbo  cbo  larch
193750  310250   2    0    4    6   30    0   1     30        51    BER  ELS  B225
182000  323250   3    0    4    5   30    0   1    100        28    BER  EIK  B225
163250  369250   1    0    3    5   30    0   1     30        22    BER  DOU  B203
165750  370750   8    0    4    5   30    0   1     60        31    BER  ELS  B226
 14750  377750   2    0   11    0   30    0   1     50         3    GEE  GEE  B004
 15000  378000  62    0    2    2   30    0   1     50         9    GEE  GEE  B012
167500  384000   1    0    5    5   30    0   1     30        34    BER  GRO  B206
 85000  388250   3    0    1    0   30    0   1     25        12    GEE  GEE  B111
```

Biomini.txt, *paramete.txt* and *terugtre.txt* are input files with fixed parameter values. Wamelink et al. (2000) gives a more detailed description of these input parameters.

5 Configuration management for SMART_NL

SMART_NL has a configuration and change management, which is organised with the program Visual Source safe from Microsoft. The details are described in a concise manual in annex 5.

The place of SMART_NL in the whole development of several SMART-models, can be seen in Table 9 and Figure 4 (the green box).

Table 9, The difference between the models.

Model	Version	Application
Stand alone (single point)		
SMART	1	The original parent model
SMART 1p5	1.1	Rather simple nutrient cycle, parent model for EUSMART
SMART2	1.1	With nutrient cycle and improved hydrology modelling
SMART22L	1.1.1	A two soil-layered version of the SMART2 model
SMART2/SUMO	1.1.1	SMART2 with integrated SUMO
Larger scale ((multiple points)		
EUSMART	1.1.1	SMART 1p5 embedded in a framework for application on a European scale
SMART_NL	2.1.1	SMART2, embedded in a Oracle-framework for application on a larger scale
SMART_NL	3.1.1	The model of this users guide, operates in Window-NT environment
SMS_NL	1.3.1.1	SMART_NL with integrated SUMO

With the integration of the succession module SUMO, there is a feed-back between growth and N-availability. It is possible to model succession from grassland to forest and to investigate management options. SUMO is integrated in SMART2 as well as in SMART_NL.

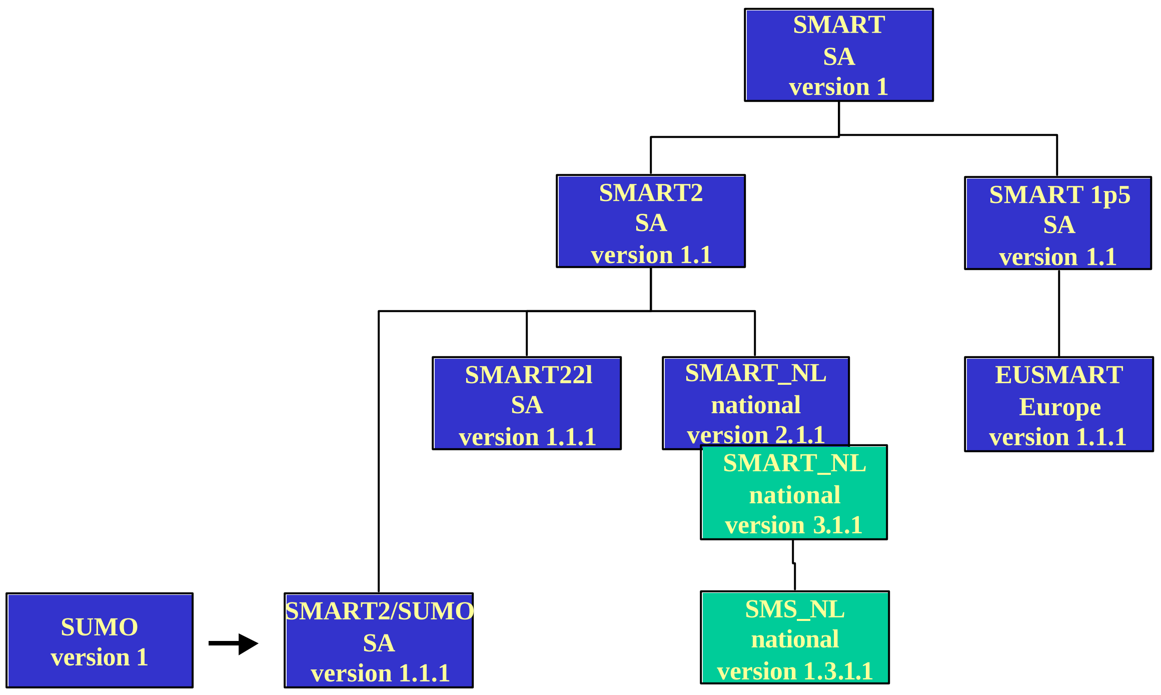


Figure 4 Model development (SA = stand alone: operates for one point only)

Version control numbering

The version control numbering is constructed in such a way that the development of the model can be derived from the version number. The parent model has just one number. This number comes back as the last number in each version number of the extended models. The most extended model till now, SMART_NL with integrated SUMO, has 4 numbers. The first number is the version number of SUMO, the second number is the number of the model where SUMO is integrated in. The third number is the version number of SMART2. For example, when SMART2 is updated, the version number of SMS_NL changes in 1.3.2.1, but when just SUMO is updated, the number will change into 2.3.1.1.

References

Alkemade, J.R.M., Latour J.B.I. , Staritsky, G. & Wiertz, J (1997) De natuurplanner: decision support system natuur en milieuversie 1.1, RIVM rapport 711901019, Bilthoven

Bolsius, E.C.A, J.H.M. Eulderink, C.L.G. Groen, W.B. Harms, A.K. Bregt, M. van der Linden, B.J. Looise, G.J. Maas, E.P. Querner, W.L.M. Tamis, R.W. de Waal, H.P. Wolfert en M. van ‘t Zelfde (1994). Eén digitaal bestand voor de landschapsecologie van Nederland. LKN-rapport 4, Rijksplanologische Dienst, Den Haag.

De Vries, W., M. Posch and J. Kämäri (1989) Simulation of the long-term soil response to acid deposition in various buffer ranges. Water Air and Soil Poll. 48: 349-390.

Erisman, J.W. (1991) Acid deposition in the Netherlands. RIVM report nr. 723001002 Bilthoven

Jaarsveld, H.J.A. van (1995) Modelling the long-term atmospheric behaviour of pollutants on various spatial scales. Thesis, Universiteit Utrecht.

Kros, J., Reinds, G.J., De Vries, W., Latour, J.B. and Bollen, M.J.S. (1995) Modelling of soil acidity and nitrogen availability in natural ecosystems in response to changes in acid deposition and hydrology, Alterra Report 95, 90 pp. Wageningen

Kros, J. (1997) Verbetering, verfijning en toepassing van het model SMART2 – De modellering van de effecten van verzuring, vermesting en verdroging voor bossen en natuurterreinen ten behoeve van de Milieubalans, Milieuverkenning en Natuurverkenning, Alterra MBP rapport 3 Wageningen.

Pastoors, M.J.H. (1993) Landelijk grondwater model; conceptuele modelbeschrijving - Onderzoek effecten grondwaterwinning 10, RIVM rapport 714305004 Bilthoven.

Wamelink, G.W.W., J.P. Mol-Dijkstra, H.F. van Dobben, J. Kros & F. Berendse (2000) Eerste fase van de ontwikkeling van het successie model SUMO 1; verbetering van de vegetatiemodellering in de Natuurplanner.Wageningen, Alterra, 2000. Alterra-rapport 045.

Annex 1 Overview of SMART_NL and SMS_NL files

The necessary files are included in several files (see table 6)
1: files with all non-spatial data (*.out) and steering (*.dat)
2: tables with spatial eco-system data
3: files with deposition and hydrological data
4: complete source of SMS_NL (which includes all relevant SMART2 and SUMO modules, *.for, *.inc, *.lib)
5: routine for generating files with hydrological and deposition scenarios

Table 9 Necessarily files for SMART_NL and SMS_NL

DATA			SOURCE	
Parameters and steering data	Table ecosystem	Scenarios	Model	Generating scenario files
SMART_NL				
Smrt_med.out	* Nednvk250.qua1)	SOxdep.dac	Ass_code.for	conv_asc_dac.for
Smrt_nv1.out		NHxdep.dac	Ass_int.for	strip.for
Smrt_sv.out		Noydep.dac	ass_valu.for	
Conf_sel.dat		BCdep.dac	brentp.for	
Conf_inp.dat		rain.dac	Chbal.for	
Prov_cod.dat		Hydrology.dac	clear.for	
Soil_cod.dat		SOxdep.hdr	dernum.for	
Soil_par.dat		NHxdep.hdr	Getgridpointer.for	
Sv_cod.dat		Noydep.hdr	Getkwelrec.for	
Sv_par.dat		NHxdep.idx	Get_soil_veg.for	
Vars_cod.dat		rain.idx	* Influx2.for1)	
Veg_cod.dat			Initdep.for	
Veg_par.dat			Interpdep.for	
			Interpfac.for	
			* Paralloc.for1)	
			Readinp.for	
			Selcomp.for	
			Skip_grid.for	
			Skipl.for	
			Sma2subs.for	
			* smart_nl.for1)	
			Splword.for	
			Strip.for	
			Write_results.for	
			Smart_i.inc	
			Smart_p.inc	
			Ttutill.lib	
Extra files for SMS_NL				
Biomini.txt	* Grid_s_v.dat1)		* Influx2_sumo.for1)	
Paramete.txt	Vegout.txt		Inisumo.for	
Terugtre.txt			Lengte.for	
			* Parall_b.for1)	
			* Sms_nl.for1)	
			Sumosm.for	
			Vegtype.for	

1) files marked with '*' must be substituted when SMS_NL is built instead of SMART_NL

☛ **Programmer remarks:** Before building the Fortran-project, adjust the Project settings: At tabpage 'Fortran', change the category to Fortran data, then choose under 'data options': 'use bytes as RECL'.

Annex 2 Generation of spatial data files

2.1 The ecosystem table

The joining of the files happens in two stages. First the LGM output and the reference files (delivered as ArcGrid workspace) are exported to grid-ASCII files (step 1). Then the gridfiles are integrated with the soil/gt/vegetation map (step 2)

1 Export to grid-ASCII (250´250 m2) files and the regulation of the correct 'window' occurs with the following AMLs:
NLKWEL.AML (input: seepage0, MSW0; output: seepage.asc, MSW.asc)

Input files

Soil/gt/vegetation:
nednvk2.asc grid-ASCII (250×250 m^2) with the SMART soil/gt/vegetation types
Reference files:
MSW0 ArcInfo-grid (250×250 m^2) with reference MSW (m-mv)
kwel0 ArcInfo-grid (250×250 m^2) with reference seepage (mm day^{-1}; upwards flux: < 0)

These files have to be converted to files with a maximum record length of 500 characters, because the with step 1 generated grid-ASCII files are processed by GENSTAT, which has a input restriction of 500 characters. This is done under DOS with the program PSFIX.

Outputfiles

nednvk2.dat:	Grid-ASCII (250×250 m^2) with the SMART soil/gt/vegetation types with record length = 500
seepage.dat:	seepage.asc with record length = 500
gvg.dat:	gvg.asc with record length = 500

2 The soil/gt/vegetation file and the file with the hydrologic scenario are joined together as a 6 column ASCII file with GENSTAT procedure NEDNVK250.G5.

Inputfiles

nednvk2.dat:	Grid-ASCII (250×250 m^2) with the SMART soil/gt/vegetation types with record length = 500
seepage.dat:	seepage.asc with record length = 500
gvg.dat:	gvg.asc with record length = 500
kwelkwal.out:	LKN output, see Bolsius et al. (1994)

Outputfiles

nednvk250.qua:	ASCII file with 6 columns.

2.2 The hydrologic scenario

Necessary files

LGM output:

– nlqh0-mv4 ArcInfo-grid (250×250 m^2) with seepage alteration (LGM output) (mm day^{-1}; defined as seepage-mv4 – seepage0; in other words: a value < 0 indicates an increased seepage flux)
– nldh0-mv4 ArcInfo-grid (250×250 m^2) with MSW-alteration (LGM output) (cm-mv; defined as MSW-mv4 – MSW0*100; in other words: a value < 0 indicates an increased MSW (groundwaterlevel closer to the surface level))

Method

The joining of the files happens in two stages. First the LGM output and the reference files (delivered as ArcGrid workspace) are exported to grid-ASCII files. Then the grid-ASCII file is converted to a direct access file (see 2.3).

Export to grid-ASCII (250×250 m2) files and the regulation of the correct 'window' occurs with the following AMLs:
NLDKWEL.AML (input: nlqh0-mv4, nldh0-mv4; output: dseepage.asc, dMSW.asc)

Annex 3 Error messages and warnings in SMART_NL and SMS_NL

The file smart_err.lst lists all recognised errors and warnings that can occur in SMART_NL during runtime.
Errors cause a program stop, warnings do not.
Parts of the error message that are variable, are indicated by xxx

ERRORS

Errors in ass_code:
(1)
error in Ass_Code
Code xx is not standard
cause: wrong code used for Province, Soil, Vegetation or Output_Var
action: use right codes in CONF_SEL.DAT file

(2)
error in Ass_Code
Code xx is used twice
cause: code for xx is used twice
action: use codes in CONF_SEL.DAT file only once

Errors in ass_int:
(1)
Error in ass_int
Wildcard not allowed for xx
cause: incorrect use of wildcard
action: use codes instead of wildcards in CONF_SEL.DAT

(2)
Error in ass_int
Only one value allowed for xx
cause:
action: use only one value for xx in CONF_SEL.DAT
(3)
Error in ass_int
yy is not valid for xx
cause: wrong code for xx
action: user correct code for xx in CONF_SEL.DAT

Errors in dernum:
(1)
Error in DerNum
Code xx is not standard

cause: xx is not valid for Soil_Code, Veg_Code or SV_Code
action: use correct codes (as given in Code_Files)

Errors in get_soil_veg:
(1)
Error in Get_soil_veg
Syntax error in ecosystemfile 'linenumber'
cause: Type error in ecosystem table

Errors in initdep
(1)
inconsistent grid-sizes in acid deposition files
cause : different grid-sizes in acid deposition files
action : use the same grid-sizes for all acid deposition files

(2)
years do not match for acid deposition
cause : different years in acid deposition files
action : use the same years for all acid deposition files

Errors in paralloc:
(1)
Error in ParAlloc
Depth in Conf_Sel.dat is not known
cause: wrong indication for depth to consider
action: use either 'DRZ' or 0.1, 0.2, .., 1.0 in CONF_SEL.DAT

Errors in readinp:
(1)
Error in ReadInp
End of file met in xx before all soils were met
End of file met in xx before all vegetation types were met
End of file met in xx before all soil vegetation combinations were met
cause: parameterfile is not complete
action: use all codes as given in the Code_Files

(2)
Error in ReadInp
@SOIL is missing in xx
@DEPTH is missing in xx
@VEG is missing in xx
@SV is missing in xx
cause: missing indicator in file xx
action: add indicator in file xx

(3)
Error in ReadInp

yy in xx is wrong unit for zz
cause:
action: use right unit for parameter zz in file xx (cf. Code_files)

(4)
Error in ReadInp
Code yy in xx not known
cause: wrong code is used
action: use right code for parameter zz in file xx (cf. Code_files)

(5)
Error in ReadInp
Too much (i.e. double) parameters met in xx for soil code yy
Too much (i.e. double) parameters met in xx for vegetation code yy
Too much (i.e. double) parameters met in xx for soilvegetation code yy
action: use code for parameter zz in file xx only once

Errors in selcomp:
(1)
Label xx is not assigned
action: assign value to item @LABEL

(2)
Code xx is too long, maximum length = 3
action: reduce length of item @LABEL in CONF_SEL.DAT

(3)
Code xx is not standard
action: use correct code in CONF_SEL.DAT (cf. Code_Files)

(4)
xx is not valid for yy
action: use correct code in CONF_SEL.DAT (cf. Code_Files)

(5)
Depth must be given in [m], maximum depth = 1 m
cause: wrong indication for depth to consider
action: use either 'DRZ' or 0.1, 0.2, .., 1.0 in CONF_SEL.DAT

(6)
Output year xx is outside range byear - eyear
action: use year in <byear,eyear>

(7)
ii items used for xx 10 is the limit
action: use less than 10 in CONF_SEL.DAT
(8)

xx in yy not known
cause: item xx in CONF_SEL.DAT not valid
action: use correct item

(9)
xx in yy not found
cause: item xx in CONF_SEL.DAT was not encountered
action: add item xx to CONF_SEL.DAT

Errors in sms_nl:
(1)
x- or y-coordinates in grid_s_v.dat and vegout.txt do not match
grid_s_v.dat: xco,yco
vegout.txt: xcoord,ycoord
cause: different order of grid cells or different co-ordinates
action: make order and x- and y-co-ordinates equal

Errors in sumosm:
(1)
foute verdeling voor structuurtype xx ,Cverd
cause: distribution of C in vegetation does not totalise to 1 (error in paramete.txt).
action: contact Alterra

(2)
foute verdeling voor structuurtype xx Nverd
cause: distribution of N in vegetation does not totalise to 1 (error in paramete.txt).
action: contact Alterra

WARNINGS

Warnings in SMART_NL
(1)
[H+] zero
For receptor xxx,xxx,xxx,xxx,xxx
cause : the [H^+] concentration became zero for a receptor (this sometimes occurs if the system shifts from the exchange bufferrange to the calcium carbonate bufferrange)
action : this needs to be solved in next version of SMART

(2)
net bc1-flux negative
cause : the sum of all input fluxes of monovalent base cations minus all losses was less than zero
action : check input data and contact Alterra.

(3)

net bc2-flux negative
cause : the sum of all input fluxes of divalent base cations minus all losses was less than zero
action : check input data and contact Alterra.

Warning in brentp
(1)
root must be bracketed for brentp
action : check input data (mistake in the ranges) and contact Alterra.

(2)
brentp exceeding maximum iterations
action: quit the program and check input data (contact Alterra)

Warning in inisumo:
(1)
fout vegetatietype; berekenen als natuurlijk terrein
cause: FILENR in vegout.txt > 50.
action: change FILENR

Note: if this warning occurs, SUMO changes FILENR in 12 and VEGTYPE in 8.

Annex 4 Manual for configuration and change management

Introduction

This guide is meant as an aid for the developers of SMART-projects. It explains the use of the program Microsoft Visual Source safe (VSS), for configuration and change management.[1] VSS helps to manage projects, regardless of the file type (fortran files, binary files, text files, etc.) by saving them to a database. This database will be maintained by the configuration manager, currently M. Grobben, who can be addressed in case of questions or problems (here also refered to as administrator).

Version control addresses the following areas:

- Team coordination — making sure, by default, that only one person at a time is modifying a file. This prevents files from accidentally being replaced by another user's changes. The administrator can change this default to allow multiple simultaneous checkouts of a single file, while still preventing overwrites of other changes. Since the development of SMART_NL is often done by several people at the same time, the use of VSS project organisation makes team coordination rather easy.
- Version tracking — archiving and tracking old versions of source code and other files, which can be retrieved for bug tracking and other purposes. When a file is added to VSS, the file is backed up on the database and made available to other people. Changes that have been made to the file are saved; so, an old version can be recovered at any time. Members of the team can see the latest version of any file, make changes, and save a new version in the database. VSS can maintain multiple versions of a file, including a record of the changes to the file from version to version.
- Reusable or object-oriented code — tracking which programs uses which modules so that code can be reused. When a files needs to be shared between two or more projects, it is possible to share them very quickly with the VSS program.

For more information on projects or files, see the help-option of the VSS program. When a project is ready to deliver to another person, institution, or Web site, VSS makes it easy to share and secure different versions of the selected set of files.

Version control

VSS provides version control and history services, to ensure that each version of a file is recoverable. The date/time stamp is used to record when files are checked out or changed. The configuration management of SMART uses three methods to track versions of files and projects:

Version numbers. These are internal numbers maintained by VSS. The user, has no control over these numbers. Every version of every file and project in VSS has a version number. The version number is always a whole number and increases every time.

[1] Recursive remarks stand for options in the VSS program

Labels. These are strings that the programmer can apply to any version of a project orfile. A label is a free-form string of up to 31 characters (e.g.: "1.1.0 ", "2.1.1 test 3", and "1.1 approved by Hans".

Date/Time stamps. These tell when a file was last modified, or when a file was checked in. SS supports a 24-hour format, which is used for this project.

Table 10 Compares and contrasts version numbers and labels.

Version number	Label
Assigned automatically by VSS	Assigned by user, using the Label command from the File menu or the History dialog box
Always a numeric value	Any combination of letters, numbers, symbols, and spaces up to 31 characters
Always increases to next whole number	Anything the user assigns
Increases each time when an action that affects storage is taken on a file or project, such as adding, checking in, or branching	Assigned when the user feels that a significant milestone has been reached
Displayed in history, paths, links, share, and file properties dialog boxes and in file pane of VSS Explorer	Displayed in history dialog boxes as a user-supplied string. Indicated by a label icon next to the project name in place of a version number. The user-supplied label string is displayed in the Action column of the History dialog box.
Does not create a new version, simply identifies a new version	Creating a label can create a new version of the file or project and the label is associated with the new version
Cannot be edited or changed by user	Can be edited in the History Details dialog box

Note It is not necessary to label a project with every build. You can get equivalent functionality and less database clutter by doing a history, get at a specific date-time stamp.

Check In, Check Out, and Edit Files and Projects

To make changes to a file you must first check it out of the VSS database. When you *Check Out* an item, VSS places a writable copy in your working folder. A file that is checked out cannot be checked out by anyone else unless your installation of VSS has been set up to allow multiple check-outs (which is the case with the SMART-projects) . You can see who an item is checked out to in the User column of the file pane. You can check out one file, multiple files, or all files in a project depending on what item(s) you have selected when you perform the checkout operation. In addition to checking out an item, you can also *View* or *Get* the item. When a file is checked out, the following options are available:

Check In your updated file or project, thus storing your changes in the current project in VSS.

Undo your checkout, canceling your changes both in VSS and in your working folder — the file or project returns to the way it was before you checked it out.

View a file or project

When you want to view a file or project, but don't want to modify it, use the *Get* or *View* command. *Get* copies the file or project from the current project into your working folder. The file placed in your working folder is read-only — any modifications made to it at this point cannot be saved in the VSS project. If you do not have a working folder, you can still retrieve a read-only copy of a file with the *View* command.

The working folder

VSS is a tool for storing and managing files, but editing or compiling files is done in a folder that you specify for VSS to use. This folder is called a working folder. A working folder can be an existing folder or a new folder that VSS creates. VSS Explorer displays your working folder path above the list view. The working folder is the place where you actually work on a file. When you *Check Out* or *Get* a file, VSS copies the item into your working folder for that project. After you make changes to the file and check it in, VSS copies it from your working folder back into the database. From then on, VSS manages your working folder by creating folders as needed on your computer when files are checked out. A working folder can be set or changed at any time. A working folder is set per user, per project, per machine. When a file or project is exclusively checked out, the VSS Explorer icon for that item has a red box around it.

When you set a working folder for a project, you can set it for the entire project, including all subprojects under that project. It is possible, however, to explicitly set a working folder for any subproject. A working folder must be specified to perform any action that copies a file out of VSS, including the *Check Out* and *Get Latest Version* commands. By any attempts to use any of these commands without a working folder, VSS displays a message asking if you would like to set a working folder. If a working folder is not set, VSS stops the command (then you can only view a file).

Working with a file

To make changes to a file you must first check it out of the VSS database. When *you Check Out* an item, VSS places a writable copy in your working folder. A file that is checked out cannot be checked out by anyone else unless your installation of VSS has been set up to allow multiple checkouts; (which is the case for the SMART project). You can see to whom an item is checked out in the User column of the file pane. One file or multiple files can be checked out, or all files in a project depending on what item(s) you have selected when you perform the checkout operation. In addition, to check out an item, you can also *View* or *Get* the item. When a file is checked out, the following options are available:

– *Check In* your updated file or project, thus storing your changes in the current project in VSS.
– Undo your checkout, canceling your changes both in VSS and in your working folder — the file or project returns to the way it was before you checked it out.

Viewing the file history

Every VSS file keeps a detailed record of its history. From the *History* dialog box, you can view version information, comments, and general details of a file's history, as well as get an old version of a file or check it out.

Note Only files can be checked out from the History dialog box; projects cannot be checked out

Get Latest Version Command
Retrieves a read-only copy of the selected VSS items. To get the most recent version, click *Get Latest Version* on the VSS menu. To get an older version, click *Get* in the *History of File* (if you want just one file) or *History of Project* dialog box (if you want to retrieve a whole project).The *Get Latest Version* command retrieves the most recent version of a file, a group of files, or an entire project from VSS and creates read-only copies for viewing or compiling in your working folder.

Note You can choose a different working folder for the current Get operation by typing a new path in the *Get Latest Version* dialog box. Click *Browse* to navigate your folders.

Getting a file does not replace the file that you currently have checked out in your working folder, unless you set the Replace advanced option.

Results of a Get Operation:
Using the *Get Latest Version* command has four possible outcomes:
When the file you are retrieving is not already in your working folder, VSS simply gets the file.
When the file in your working folder is identical to the file you are retrieving, VSS does not retrieve or modify the file in any way.
When the file in your working folder is different from the VSS master copy, and your local copy is read-only, VSS replaces your local copy.
When the file in your working folder is not read-only, VSS assumes that you have the file checked out, and doesn't replace the file. You can change the way VSS behaves in this case with the Replace Writable option in the *Local Files* tab of the *Options* dialog box.

Shared files
In VSS, one file can be shared among multiple projects. Changes to the file from one project are automatically seen by other projects sharing the file. This encourages code reuse. Use the *Links* tab in the *Properties* command on the *File* menu to see a list of projects sharing a file. In the SMART-project files are often linked between different programs.

Note You have to be aware that once you’ve shared a file to another project, and this project gets a high label, the version label of that file will increase as well.

The construction of the version control number holds a reference to that. This means that if major changes are made in the basic program (eg. 1 becomes 2), this will result in an increase of the version numbers of all the programs (eg. 1.1 becomes 1.2).

Branching shared files
Branching is the process of taking a file into two directions (that is, branches or paths) at once. VSS keeps track of these branches by tracking the different paths as a project. Branching a file breaks the shared link, making the file in that project independent of all other projects. The changes you make in the file are not reflected elsewhere, and vice versa. A branch has been created: two files (the file in the project, and its counterpart in other projects) have a shared history up to a certain point, and divergent histories after that time. After you branch a file, the *Links* tab does not show the link that has been broken. Instead, you use the *Paths* tab (also available from the *Properties* command on the *File* menu) to view the branch history of the file. So far, this option has not been used for the SMART-project.

Multiple Checkouts
When multiple checkouts are allowed, VSS keeps track of all users who have the file checked out at any given time, and compares changes made to the file when each user checks in changes. In most cases, users have changed different lines of code in the file, so VSS simply merges the changes into the master copy of the file. However, if VSS detects that two users have both modified the same line of the file, VSS displays a message alerting the user checking in changes to resolve conflicting line changes before proceeding. When this happens, no check in occurs. The user receiving the message should edit the local copy of the file in the user's working folder to resolve the conflicts

Merging with the Get Latest Version Command
In a multiple checkout situation, you can use the *Get Latest Version* command to retrieve other users' changes before checking in your version of the file. You can tell VSS to merge the changes into your local copy of the file, or you can specify that you want to verify each change before permitting the merge. To do this, set the Replace Checked Out Files option to Merge on the *Local Files* tab in the *SourceSafe Options* dialog box

Rollback to Previous Versions
Rollback returns a file to a previous version, erasing all the versions and changes since then. All changes made after the version rolled back to are lost. For example, if you click on version 5 and then click *Rollback*, all changes since version 5 are lost. If the file you roll back to is being shared by more than one project, the rollback affects only the project you specify (the current project, by default). To accomplish this, rollback performs an implicit branch, that is, it branches the file in the specified project away from the same file in all other projects, and creates a new development path. To bring the disparate development paths back together, if you make changes to the specified project, you have to merge the files.

Note You can also perform a Virtual Rollback, which doesn't erase all versions and changes of a file. This is the preferred method for performing a rollback!
To perform a virtual rollback on a file:
Select the file in VSS Explorer.
Check out the file using the *Check Out* command on the *SourceSafe* menu.

On the *Tools* menu, click *Show History* to display the *History Options* dialog box.
Click *OK* to display the *History of File* dialog box.
Select a previous version of the file, then click *Get*.
Click *Close* to quit the *History of File* dialog box.
Check the file back into VSS Explorer using the *Check In* command on the *SourceSafe* menu.

When there are any questions or problems, please contact : Mireille Grobben (0317-474263 or m.s.grobben@alterra.wag-ur.nl).