The Capsis training

Computer Aided Projection of Strategies In Silviculture

A software platform to integrate forestry models

January 2020

François de Coligny – Nicolas Beudez

INRAE - UMR AMAP
bot**A**ny and **M**odelling of **P**lant **A**rchitecture and vegetation

The Capsis training - contents

- **Introduction: the Capsis project**

- **Software framework**
  - kernel
  - modules
  - simulation history
  - project manager
  - a guided tour
  - scripting
  - reuse of the Capsis framework

- **Specifics for forestry**
  - trees
  - species
  - stands
  - plots

- **The Capsis application**
  - graphical user interface
  - extension manager
  - disk organisation
  - resources
  - to go further

Introduction: the Capsis project

Objective - Targeted public - Main options

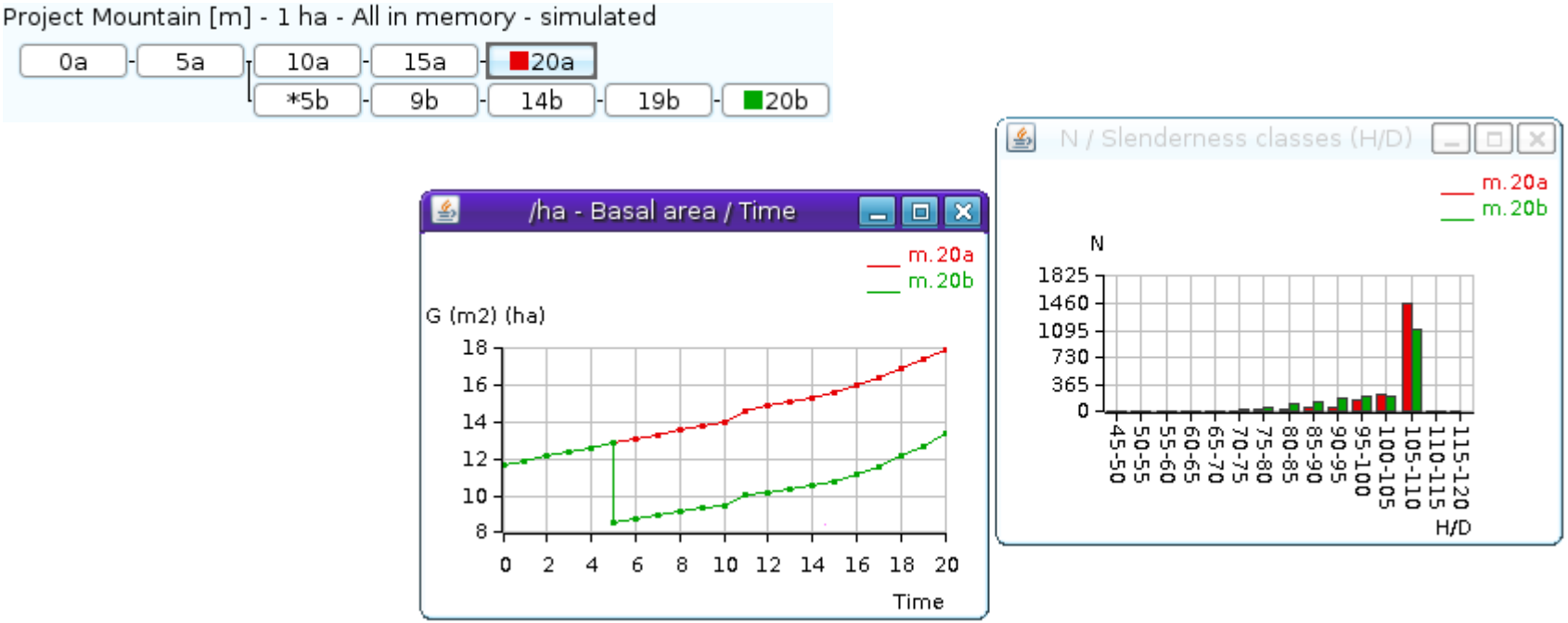
Build a generic **software** platform
to **integrate forestry** growth and yield or dynamics **models**...

...for the **scientists**, the forest **managers** and **education**

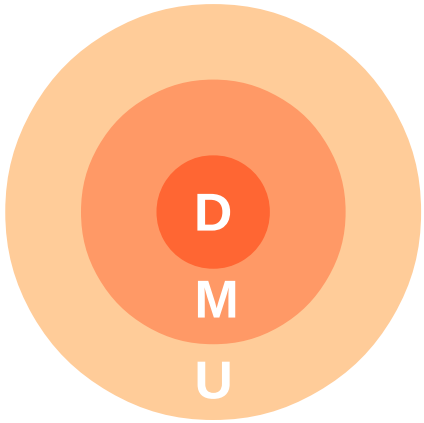
- growth **models** -> Capsis modules
- **intervention** tools (e.g. thinning)

scenarios

- technical and domain related transversal **libraries** (genetics, biomechanics...)
- **extensions** (viewers, diagrams, export...)
- based on **free** software
- **collaborative** development
- evolutions discussed by the members of the **community**



Actors - Roles - Organisation



Concentric organisation:

- **Developers**: kernel development, training, assistance...

- **Modellers**: develop their modules

- **end-Users**: of some modules, interact with the modellers

The project directions are discussed collectively by the concerned actors

The **Capsis co-development community**: developers + modellers

An annual meeting

The Capsis charter

Accepted by all members:

- **free software** (LGPL): kernel + pilots + extensions + libraries

- the **modellers develop** their modules

- they can have **support**: training, starting stage, design, assistance...

- all the **source codes are shared** by the members of the Capsis co-developement community (developers + modellers)

- mutual respect of the **intellectual property**

- developers do **technical validation**, modellers do **functional validation**

- **distribution** facilities and guidelines (with specific licenses for the modules)

- the modellers manage the **relations with their end-users**

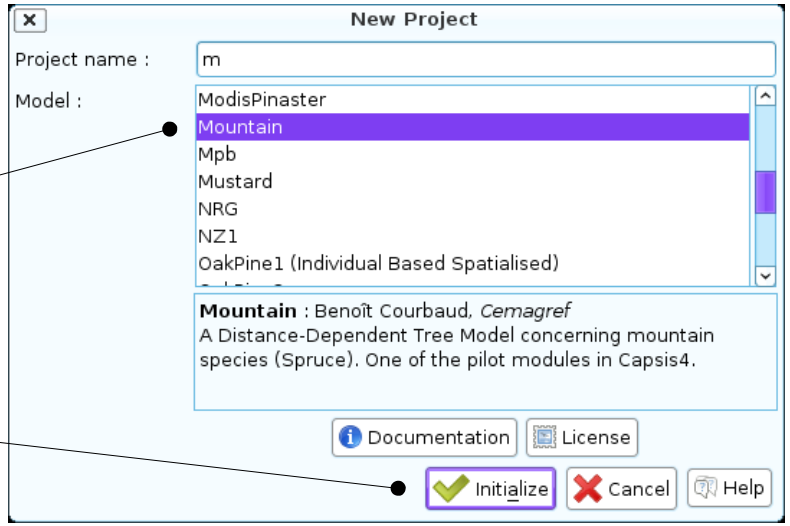
The text of the charter: http://capsis.cirad.fr/capsis/charter

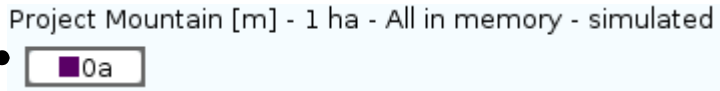
How does it work

Create a **Project**:

- a growth **module**
- a set of **initial parameters**
- an **initial state**

-> a root step in the project

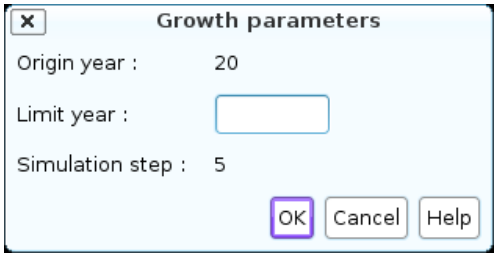




Build **silvicultural scenerios** by alterning:

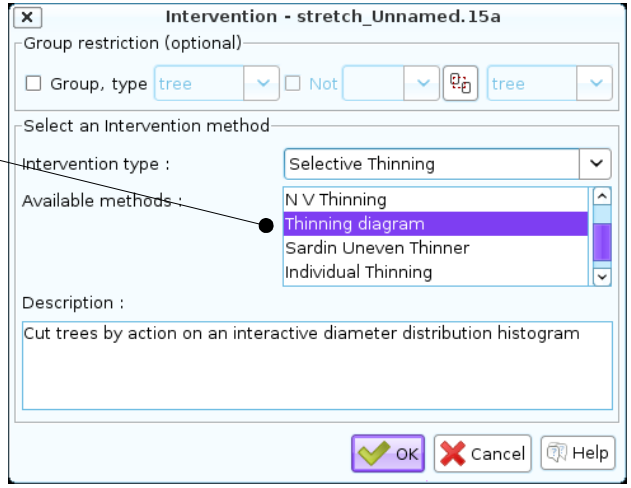
evolution stages

-> delegated to the module (with a free simulation step)

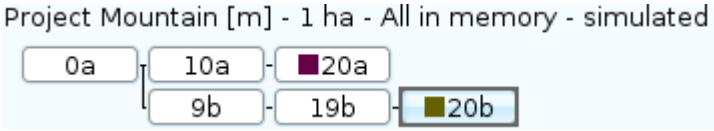


interventions

-> delegated to an extension

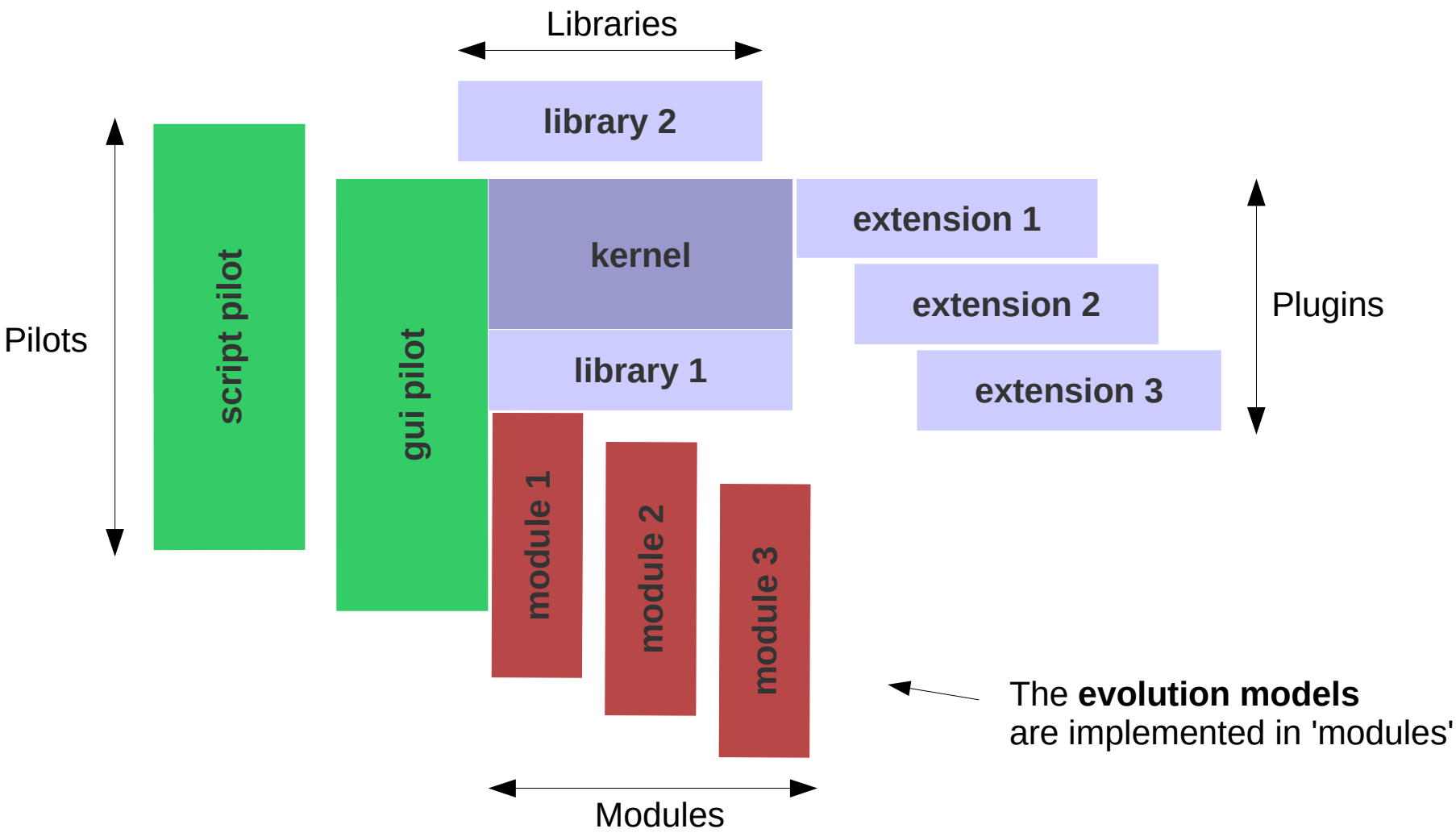


-> **New steps** in the project

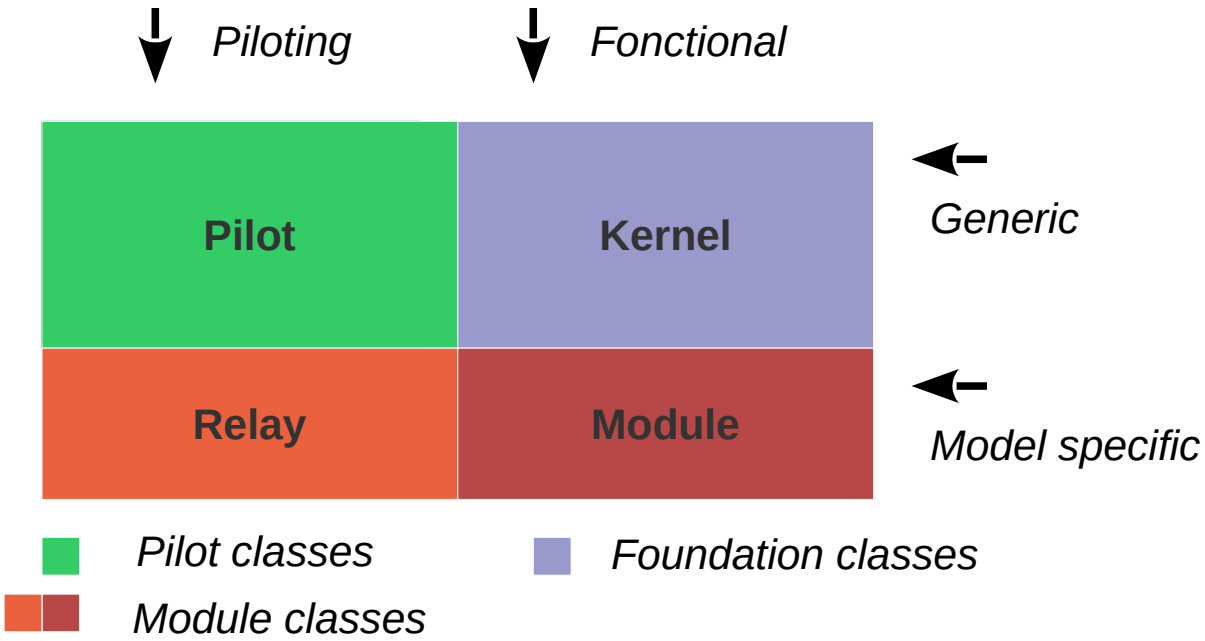


Software framework

Software diagram



Software architecture



The **pilot**: how to run the simulations

- can be a graphical user interface
- can be a script manager

The **kernel**: the main functions

- detects and loads the available models (ModelManager)
- organizes data (Session, Project, Step)
- saves on disk and reloads simulations (Engine)
- configures the application (language...)

The **relay**: between pilot and module

- answers the pilot when it knows
- relays the pilot requests to the module

The **module**: the evolution model

- creates an <u>initial scene</u>
- implements the main <u>evolution loop</u>

The Engine

The **central component** in the Capsis kernel, <u>often invisible</u>

Constructed at the very beginning of the application starting process (application Starter)

- knows the application name and main package
- knows the Pilot name and arguments
- knows the models file name
- knows the ExtensionManager if any
- knows the application version

-> creates and initializes a **ModelManager**
-> inits the **Translator** with the default Locale and loads the kernel translations
-> creates the **Pilot**
-> writes a line in the **Log**
-> tries to get the 'local max **revision** number' (accurate version number)

And knows how to:

- **load a model** (by asking the ModelManager)
- manage the **Session** (new, save, saveAs, close)
- manage the **Projects** (new, save, saveAs, close)

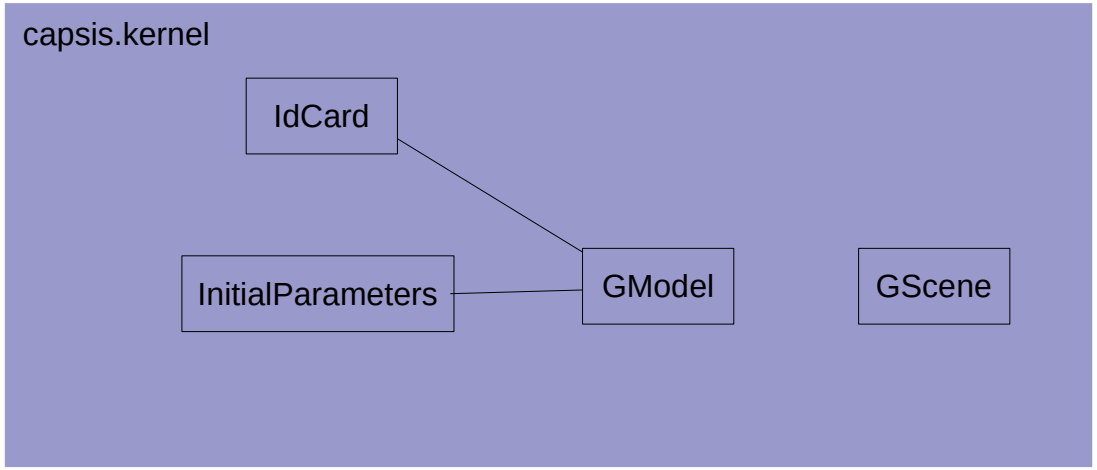
Once constructed, accessible from anywhere through **Engine.getInstance ()**

Module structure: kernel part

IdCard

model information

name
author(s)
prefix...



InitialParameters

the superclass of all
'model parameters classes'

set at simulation start time

GModel

the superclass of all
'model classes'

model initialisation and
evolution loop
(in subclass)

GScene

the superclass of all
'scene classes'

state of the system
at a given date

e.g. a plantation at age 12

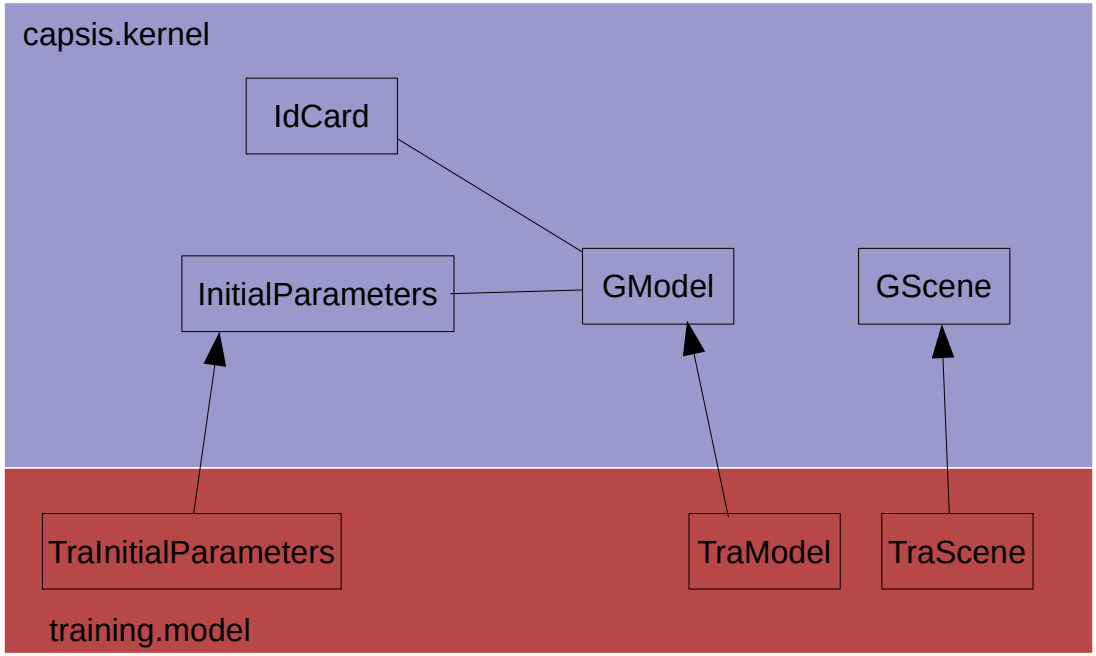
Module structure: specific part

e.g. model: **training**, prefix **Tra**

IdCard

training/
idcard.properties

training
F. de Coligny
Tra...



TraInitialParameters

training model parameters

set at simulation start time

inventoryFileName
growthP1
growthP2
growthP3
growthP4

TraModel

model class for the *training* model

model initialisation and evolution loop

TraScene

the scene description for *training*

a list of TraTree objects at a given date

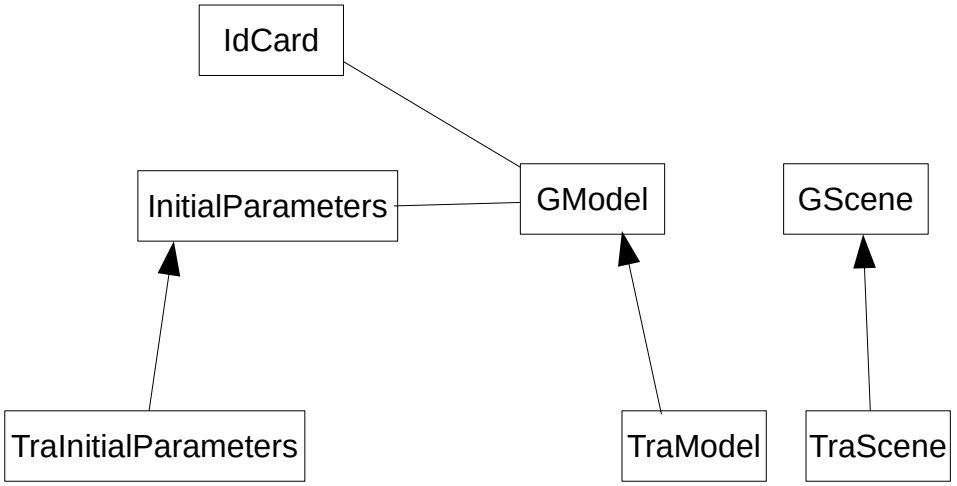
(e.g. from the list in inventory file)

Focus on module

e.g. model: **training**, prefix **Tra**

Main classes

- IdCard
- TraInitialParameters
- TraModel
- TraScene



Focus on module: IdCard

e.g. model: **training**, prefix **Tra**

The **identity card** of the module:

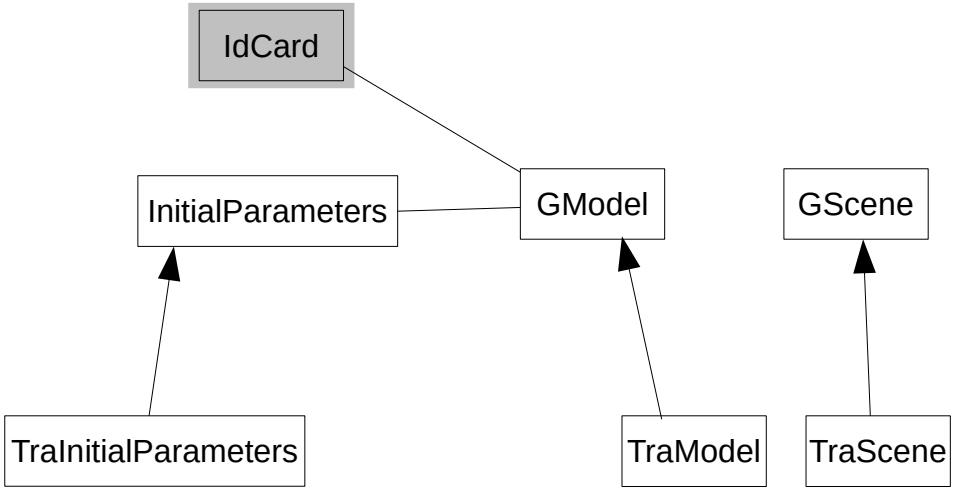
- IdCard is in capsis.kernel
- loaded from a property file named **idcard.properties** in the 'training' directory

```
protected String modelName = "-";
protected String modelAuthor = "-";
protected String modelInstitute = "-";
protected String modelType = "-";
protected String modelPrefix = "-";
protected String modelPackageName = "-";
protected String modelMainClassName = "-";
protected String modelBundleBaseName = "-";
protected String modelVersion = "-";
protected String modelDescription = "-";
```

training/idcard.properties

```
Name = Training
Version = 1.0
Author = F. de Coligny
Institute = INRA
Type = Forestry growth model
Package = training
MainClass = training.model.TraModel
BundleBaseName = training.TraLabels
Prefix = Tra
Description = The Training model by F. de Coligny (March 2014)
```

e.g. model.getIdCard().**getModelAuthor()**; // `F. de Coligny`



Focus on module: TraInitialParameters

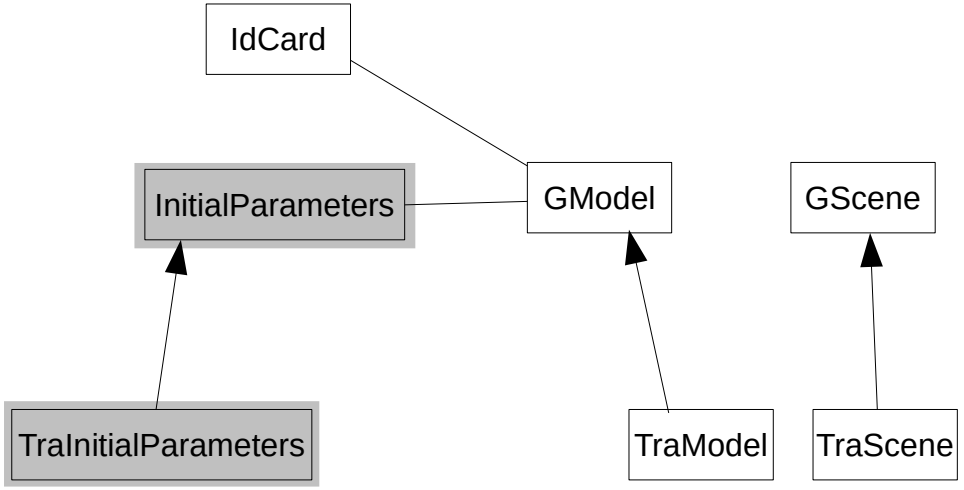
```
public class TraInitialParameters extends AbstractSettings
        implements InitialParameters {

    public String inventoryFileName;

    public double growthP1 = 4d;
    public double growthP2 = 0.2d;
    public double growthP3 = 0.2d;
    public double growthP4 = 0.1d;

    private TraScene initScene;

    /**
     * Default constructor.
     */
    public TraInitialParameters () throws Exception {}

    /**
     * A constructor for scripts.
     */
    public TraInitialParameters (String inventoryFileName)
            throws Exception {
        this.inventoryFileName = inventoryFileName;
    }

    @Override
    public GScene getInitScene () {
        return initScene;
    }

    /**
     * Builds the initial scene.
     */
    @Override
    public void buildInitScene (GModel model)
            throws Exception {
        // Load the file, set initScene...
    }

    // Other methods if needed...

}
```

[extract]

e.g. model: **training**, prefix **Tra**

1. The **parameters** of the 'training' model
2. **Builds the initial scene**

   - implements **capsis.kernel.InitialParameters**
   - **getInitScene ()**
   - **buildInitScene ()**



Focus on module: TraModel

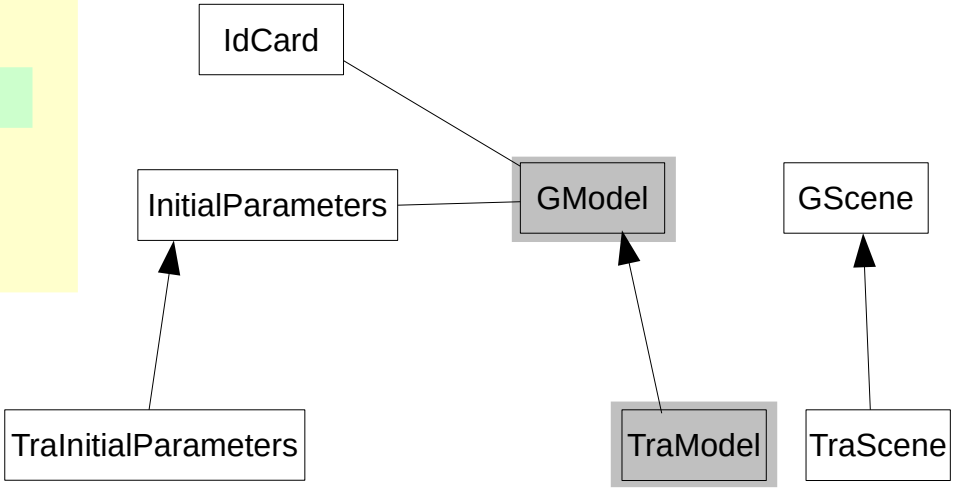
e.g. model: **training**, prefix **Tra**

```
public class TraModel extends GModel {

    /**
     * Constructor
     */
    public TraModel () throws Exception {
        super ();
        setSettings (new TraInitialParameters());
    }

    /**
     * This method is called for the first scene of the project
     * at project creation time.
     */
    @Override
    public Step initializeModel (InitialParameters p) {
        // Optional process at project creation time
        // -> nothing here
        return p.getInitScene().getStep();
    }

    /**
     * Evolution loop.
     */
    @Override
    public Step processEvolution (Step step, EvolutionParameters p)
            throws Exception {
        // Loop and create steps until the limit in p
        // Return the last Step
        return step;
    }

    // Other methods if needed...

}
```

[extract]

The **'model class'** (and the main class)

- created first when a model is chosen
- methods must be overriden



Focus on module: TraScene

e.g. model: **training**, prefix **Tra**

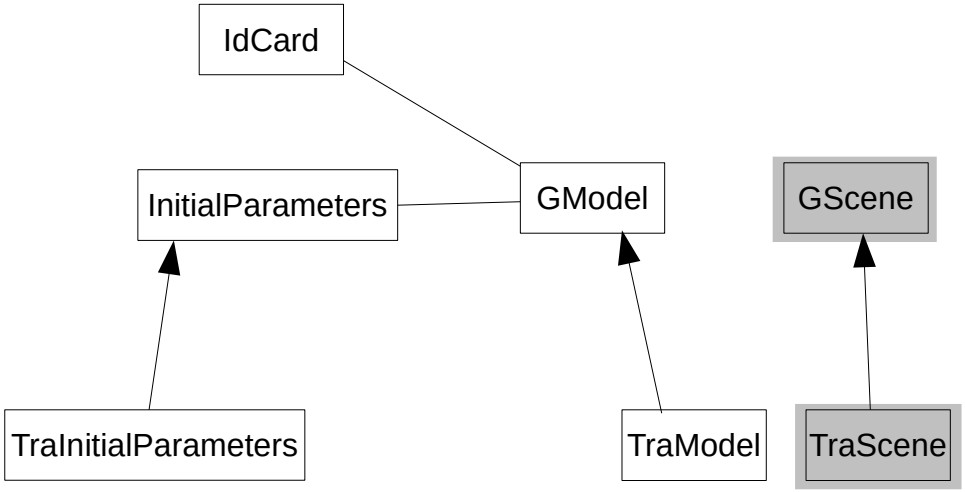
```
public class TraScene extends TreeList {

    /**
     * Constructor
     */
    public TraScene () {
        super ();
    }

    // inherited from capsis.defaulttype.TreeList
    public int getDate () {...}
    public void setDate (int d) {...}

    public void addTree (TraTree t) {...}
    public TraTree getTrees () {...}

    //...

}
```

[extract]

The **description of the scene**

e.g. a list of trees (...)

- extends GScene
- has a date
- later carried by a Step in the simulation



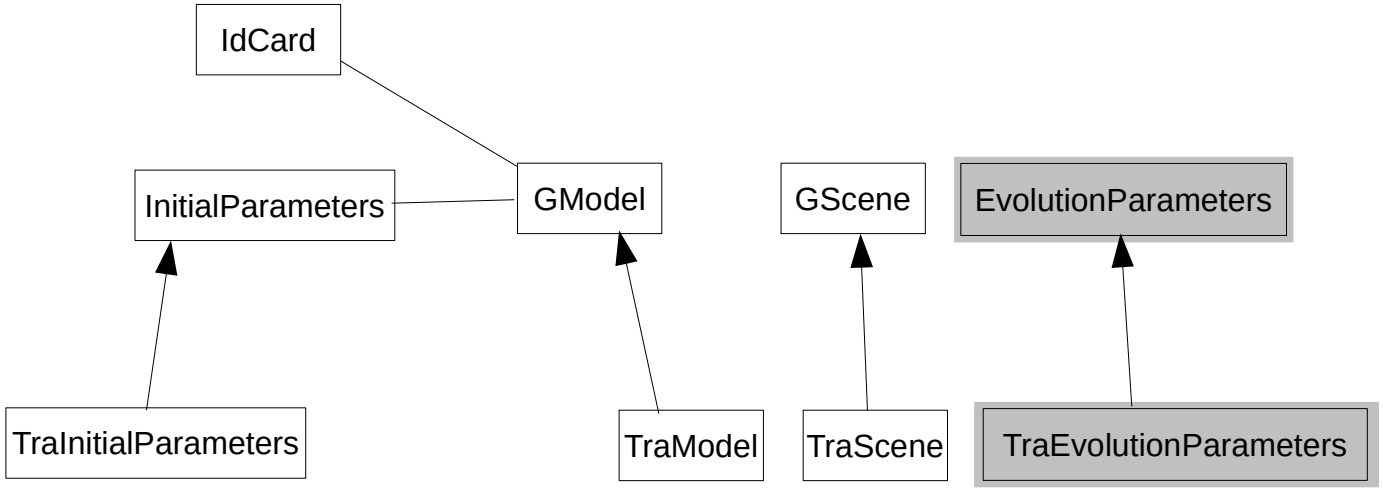
Focus on module: TraEvolutionParameters

e.g. model: **training**, prefix **Tra**

An extra class to **set an evolution target**

- for the processEvolution() method in the model class
- generally very simple (e.g. 5 years...)

```
public class TraEvolutionParameters implements EvolutionParameters {

    private int numberOfYears;  // number of years for an evolution process

    /**
     * Constructor
     */
    public TraEvolutionParameters (int numberOfYears) {
        this.numberOfYears = numberOfYears;
    }

    public int getNumberOfYears () {
        return numberOfYears;
    }

}
```



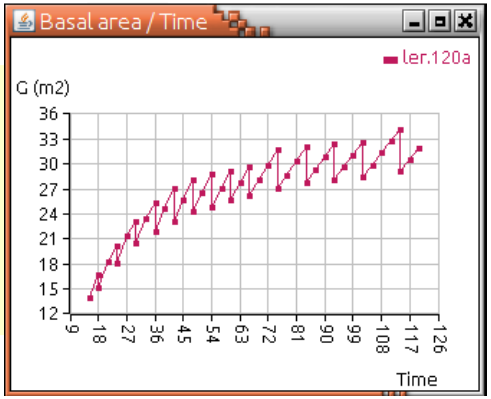
Focus on module: TraMethodProvider

e.g. model: **training**, prefix **Tra**

An **optional** way for the model to **publish calculation methods**

- implements shared interfaces
- the interfaces are in capsis.util.methodprovider
- extensions may check their compatibility with the model

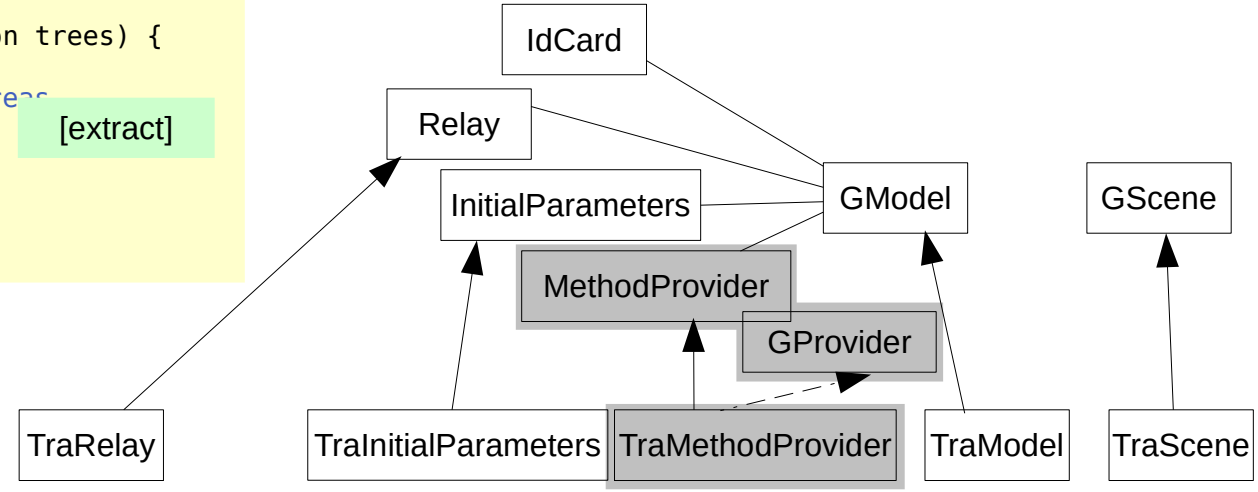
```
public interface GProvider extends MethodProvider {

    /**
     * Compute the basal area for the given trees of the given stand.
     */
    public double getG (GScene stand, Collection trees);   // restrict to the given trees

}
```



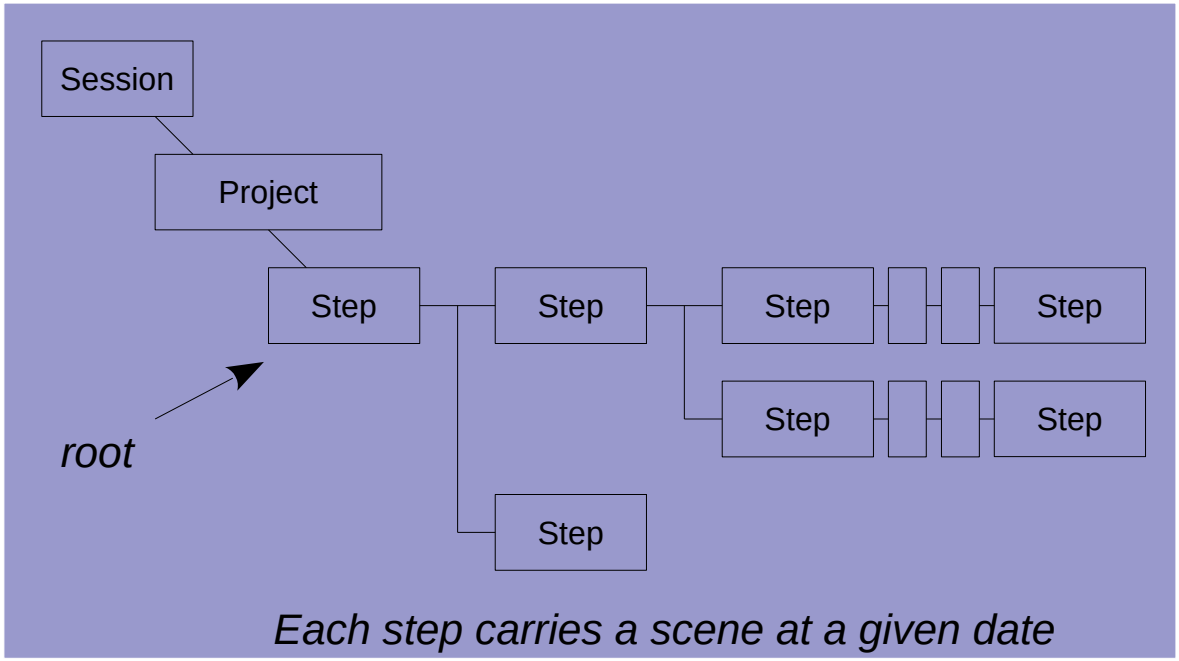
Compatible if TraMethodProvider implements GProvider

```
public class TraMethodProvider implements GProvider {

    /**
     * Basal area
     */
    public double getG (GScene stand, Collection trees) {

        // Loop on the tree, sum their basal areas
        return basalArea;

    }

}
```

[extract]



The history classes



A way to memorize the simulation

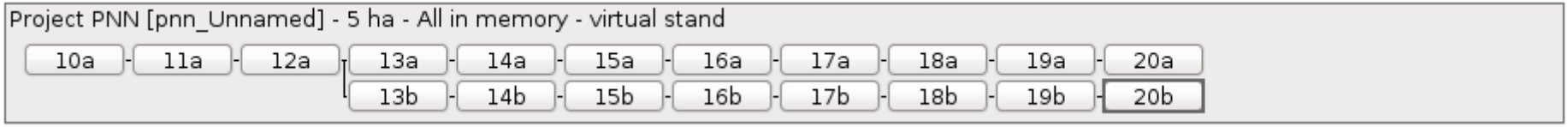
Session
- has a name
- is unique, hold by the Engine
- contains the **project list**

Projects
- have a name
- are **connected to a GModel** instance
- have a **root Step**
- contain the **method to add a new Step**

Steps
- **carry a GScene** instance with a given date
- are linked together

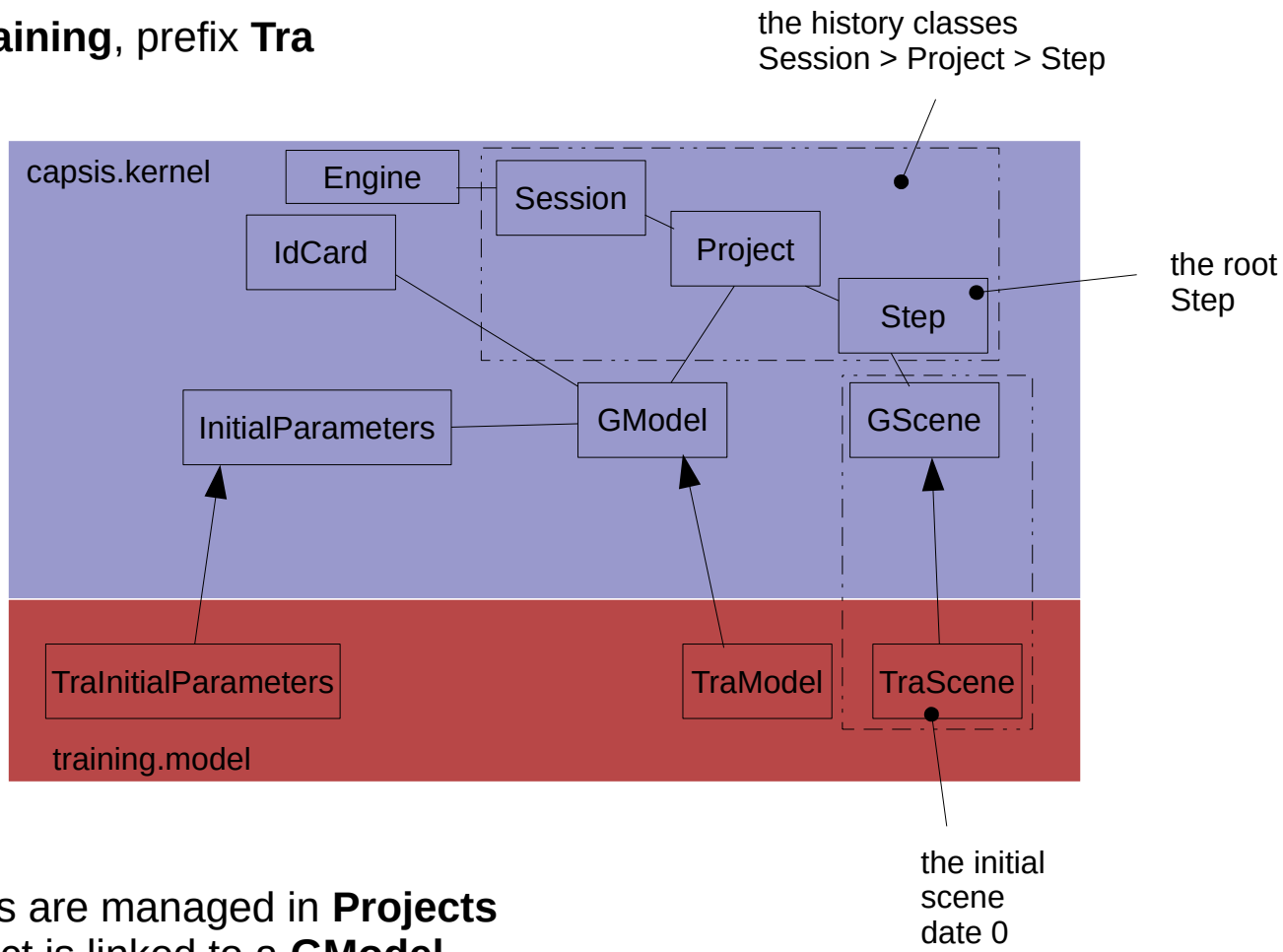
The scenes under the steps are...
calculated by InitialParamaters.buildInitScene ()
created by the processEvolution () method
resulting of the intervention

e.g. a project with 19 Steps in the project manager



A new project once created

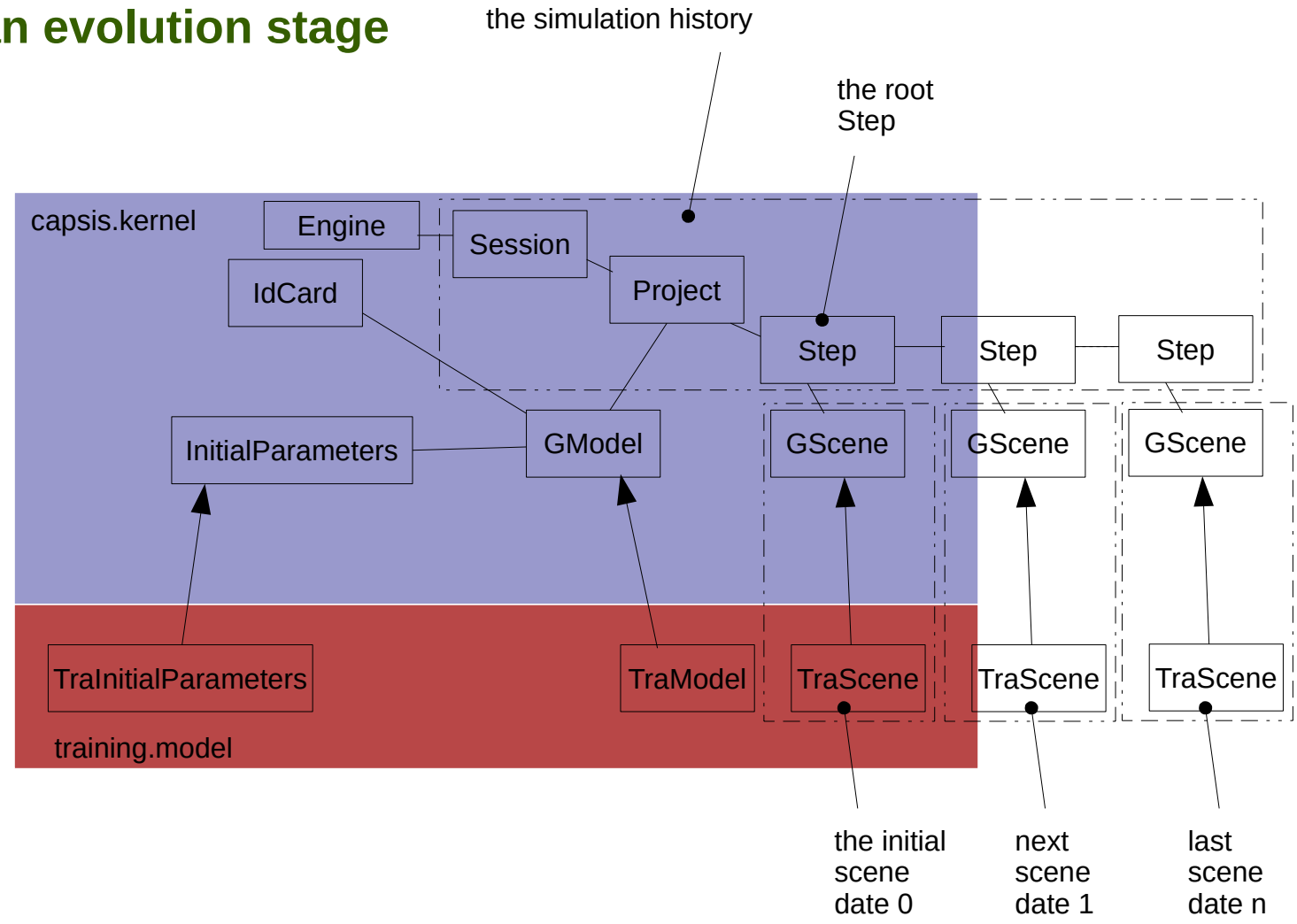
e.g. model: **training**, prefix **Tra**



Simulations are managed in **Projects**
each Project is linked to a **GModel**...
...and has a **root Step** carrying the **initial GScene**
other Steps may be linked after the root to carry the Scene resulting of evolutions...
...or interventions
the **Session** can contain several Projects

This is **the minimal possible situation** after a Project initialisation: the root Step can not be deleted

A project after an evolution stage

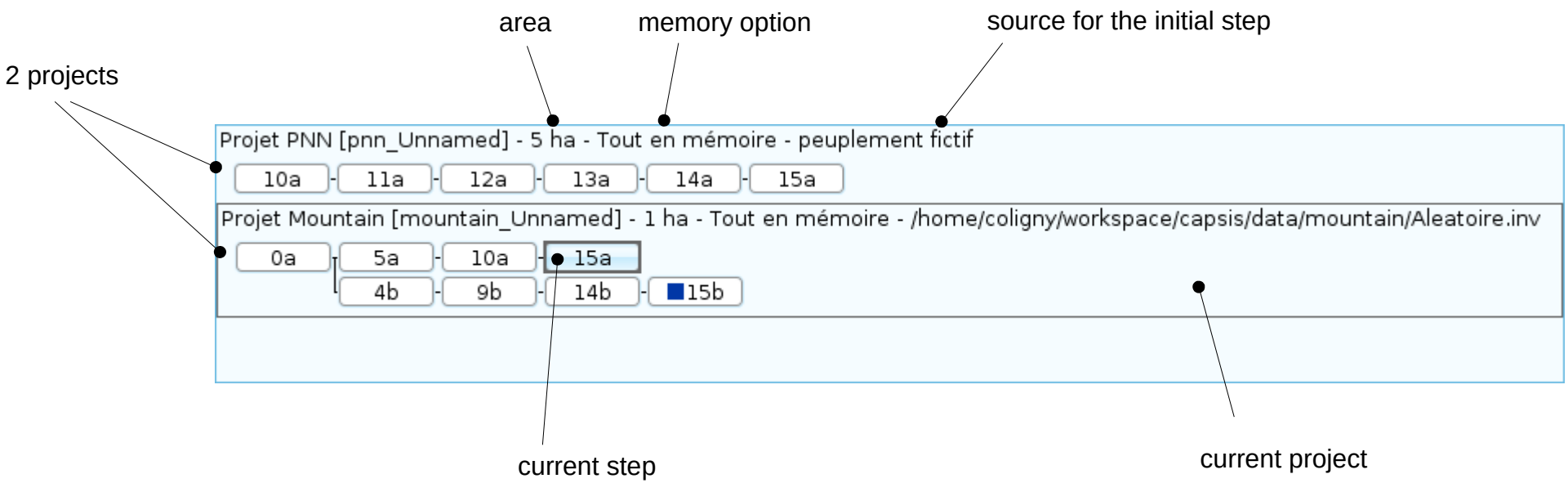


The model creates new scenes in its processEvolution() method
A new Step is created for each new Scene
The Steps are linked together in the Project

The ProjectManager

The project manager shows the **session** with all the projects included

For each project:
- each step is represented as a **button**
- each button shows the **date** of the carried GScene
- each step can have several sons if several **scenarios** were run
- a **letter** is given to each scenario: 'a', 'b'...

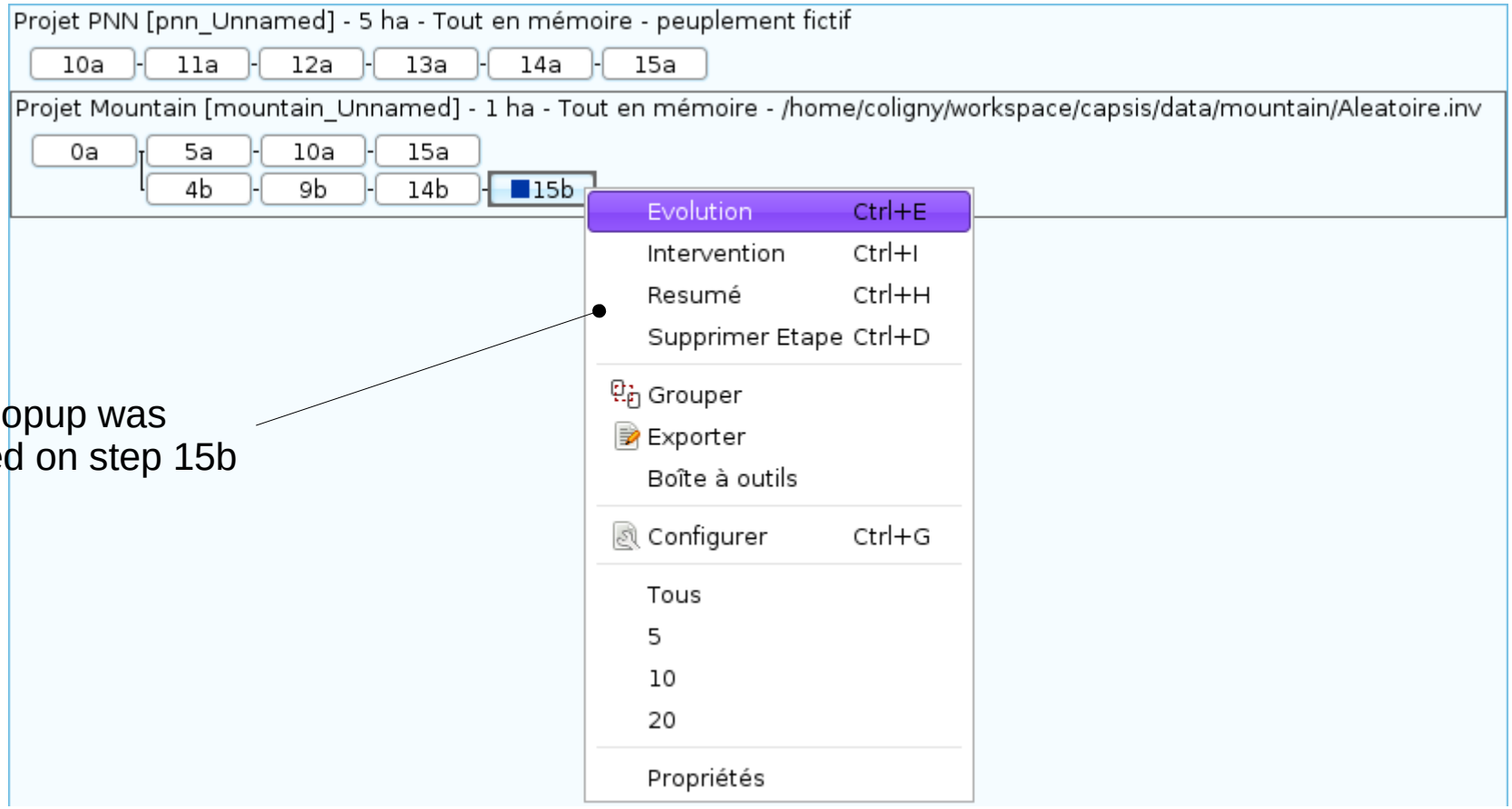


If the user clicks on buttons to change the current Step, all listening components are told

The Step Popup

It can be opened on any step:

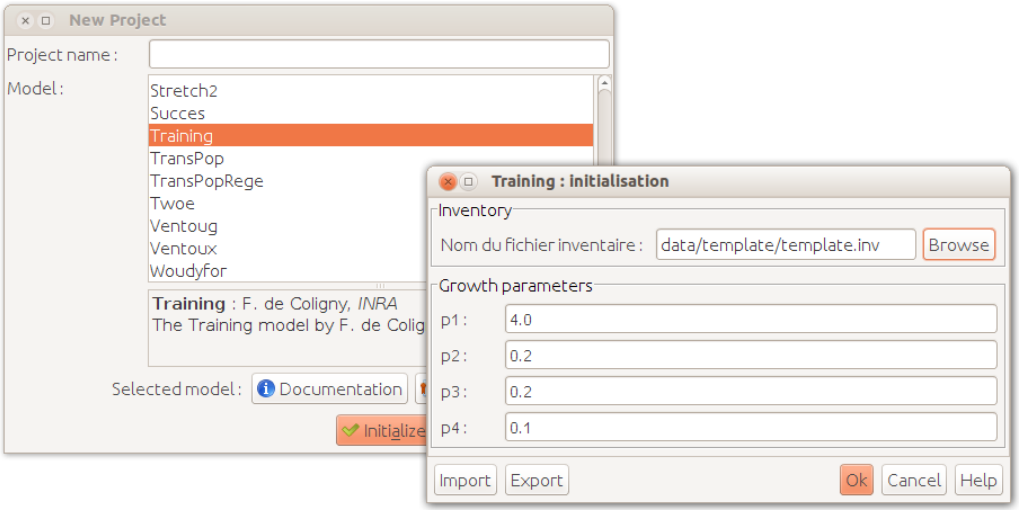
- to trigger an **evolution** from this step
- to trigger an **intervention** on this step
- to **remove** the step (only for leaf steps)
- to **view** more or less steps in the project
- to trigger **other commands**



step popup was opened on step 15b

An illustration

1. Initialisation: New project > Training > Initialize > Select a file > Ok

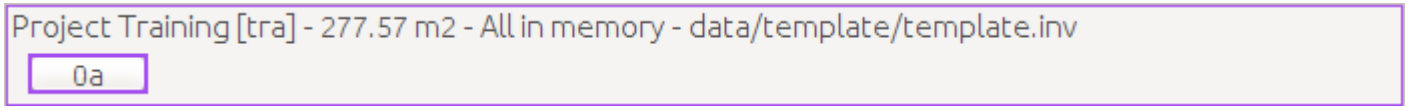


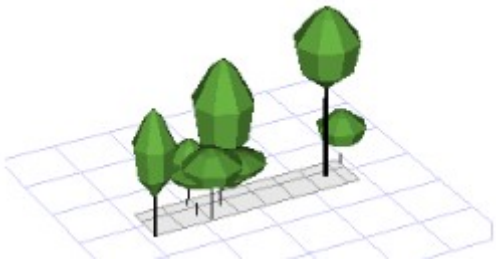

```
TraInitialParameter
      template.inv
      4, 0.2, 0.2, 0.1
      buildInitScene ()
```

```
TraScene 0
```

```
TraModel
      initializeProject ()
```

What happens:
- the file name and param values are copied in the **TraInitialParameter** fields
- buildInitScene() is called, loads the file and creates the initial **TraScene** (with **TraTree** objects inside)
- initializeProject() is called
- the project is created, with a root step carrying the initialScene





An illustration

Focus: Initialisation with a file reader

this superclass is a Capsis file loader
it contains methods to turn a list of lines in a text file
into a list of TreeLine objects

```
public class TraInventoryLoader extends StandRecordSet {
    // A tree line in the file
    @Import
    static public class TreeLine extends Record {
        public TreeLine () {super ();}
        public TreeLine (String line) throws Exception {super (line);}
        public int    id;     // must be unique
        public int    age;    // years
        public double x;      // m
        public double y;      // m
        public double z;      // m
        public double height; // m
        public double dbh;    // cm
        public double crownBaseHeight;      // m
        public double crownRadius;          // m
    }
    public TraInventoryLoader (String fileName) throws Exception {
        createRecordSet (fileName);
    }
    public GScene load (GModel m) throws Exception {
        TraModel model = (TraModel) m;

        TraScene initScene = new TraScene ();
        initScene.setSourceName (source);  // fileName
        initScene.setDate (0);
        int maxId = 0;

        for (Iterator i = this.iterator (); i.hasNext ();) {
            Record record = (Record) i.next ();

            if (record instanceof TraInventoryLoader.TreeLine) {
                // A tree line
                TraInventoryLoader.TreeLine r = (TraInventoryLoader.TreeLine) record;
                if (r.id > maxId) {maxId = r.id;}

                // Create a tree
                TraTree tree = new TraTree (r.id, initScene, r.age,
                        r.height, r.dbh, r.crownBaseHeight, r.crownRadius,
                        r.x, r.y, r.z);

                // Add the tree in the scene
                initScene.addTree (tree);

            } else {
                throw new Exception ("Unrecognized record : "+record);
            }
        }
        return initScene;
    }
}
```

create a list of TreeLine

create an initScene object

loop on the TreeLine objects

create a TraTree for each

add it in the initScene

return the initScene

```
TraInitialParameter
      template.inv
      4, 0.2, 0.2, 0.1
      buildInitScene ()
```

```
TraInventoryLoader l = new TraInventoryLoader (fileName);
TraScene initScene = (TraScene) l.load(this);
```

```
TraScene 0
```

```
TraModel
      initializeProject ()
```

```
// Optional process at project creation time
// -> nothing here
```

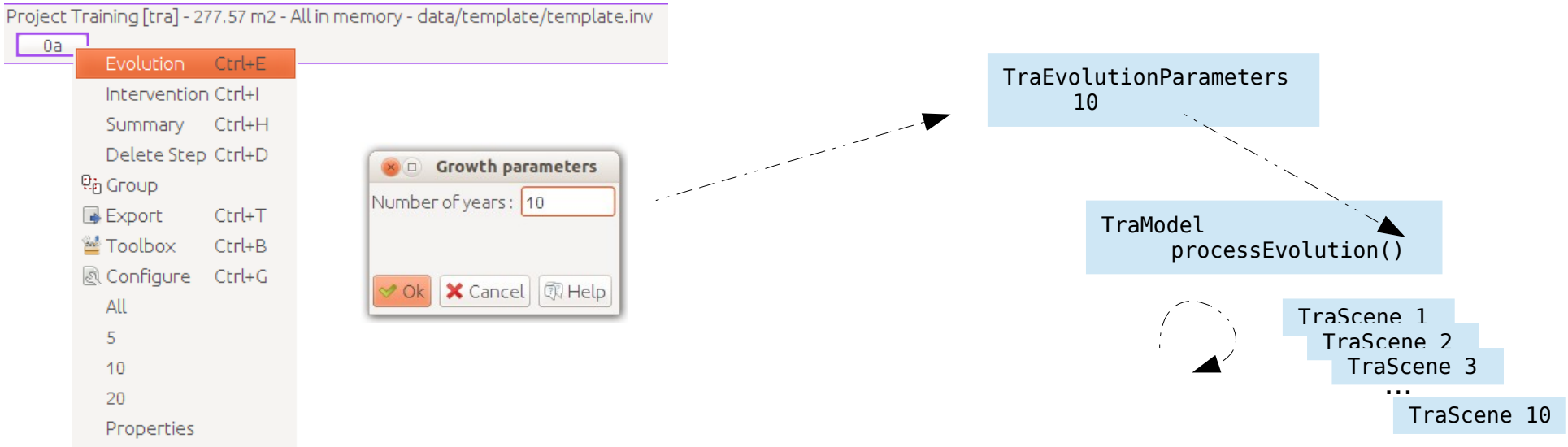
template.inv

```
##
## A sample inventory file for the Template and Training modules
##
```

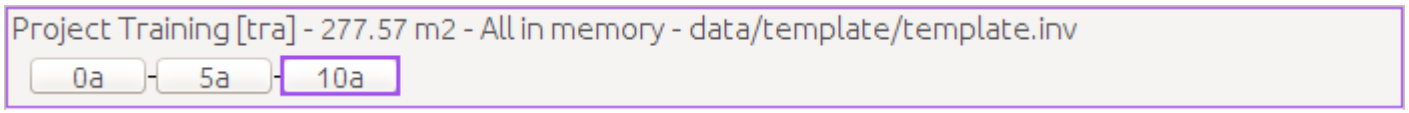
# id	age(y)	x(m)	y(m)	z(m)	h(m)	dbh(cm)	cbh(m)	cr(m)
45	34	9.53	40.70	0	22.95	46.6	7	3.5
48	35	25.68	48.17	0	24.34	51.5	5	5.2
50	15	18.27	46.70	0	10.49	12.4	8	4
52	16	18.04	45.37	0	12.99	17.8	5	3.5
53	8	22.97	44.15	0	8.85	9.3	5	6.2
54	9	18.25	43.25	0	9.88	11.2	6	5
56	17	19.82	41.20	0	12.95	17.7	6	5.2
57	15	24.80	41.79	0	10.69	12.8	6.5	4.8
59	15	46.69	46.84	0	9.93	11.3	3.2	4.2
60	40	42.47	44.51	0	30.99	77.5	16	5.6

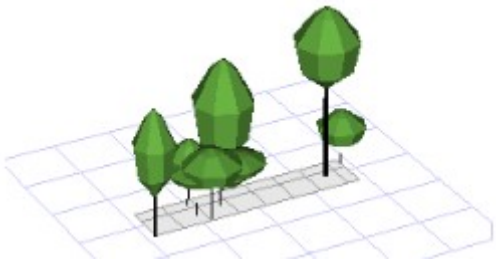
An illustration

2. Evolution: Right click on step 0 > Evolution > 10 years > Ok



What happens:
- a **TraEvolutionParameter** is created, contains the limit: '10'
- processEvolution() is called and passed the limit
- it loops 10 times to create the 10 scenes dated 1 to 10
- each time, it asks the **Engine** to create a new **Step** to carry the new scene





An illustration

Focus: TraModel.processEvolution()

```
/** Evolution loop.
 */
@Override
public Step processEvolution (Step step, EvolutionParameters p) throws Exception {

    TraEvolutionParameters ep = (TraEvolutionParameters) p;

    int originYear = step.getScene ().getDate ();
    int numberOfYears = ep.getNumberOfYears ();

    ProgressDispatcher.setMinMax (0, numberOfYears);

    for (int k = 1; k <= numberOfYears; k++) {
        int newYear = originYear + k;
        StatusDispatcher.print (Translator.swap ("TraModel.evolutionForYear") + " " + newYear);
        ProgressDispatcher.setValue (k);

        // Create the new scene by partial copy
        TraScene newScene = (TraScene) step.getScene ().getEvolutionBase ();
        newScene.setDate (newYear);

        // Make all the trees grow, add them in newScene
        Collection trees = ((TraScene) step.getScene ()).getTrees ();
        for (Iterator i = trees.iterator (); i.hasNext ();) {
            TraTree t = (TraTree) i.next ();

            TraTree newTree = t.processGrowth (getSettings ().growthP1,
                    getSettings ().growthP2,
                    getSettings ().growthP3,
                    getSettings ().growthP4);
            newScene.addTree (newTree);
        }

        String reason = "Evolution to year " + newYear;  // a free String

        Step newStep = step.getProject ().processNewStep (step, newScene, reason);
        step = newStep;
    }
    StatusDispatcher.print (Translator.swap ("TraModel.evolutionIsOver"));
    ProgressDispatcher.stop ();

    // Return the last Step
    return step;

}
public TraInitialParameters getSettings () {
    return (TraInitialParameters) settings;
}
```

10

get the limit: 10

loop 10 times

get a **partial copy** of the scene k

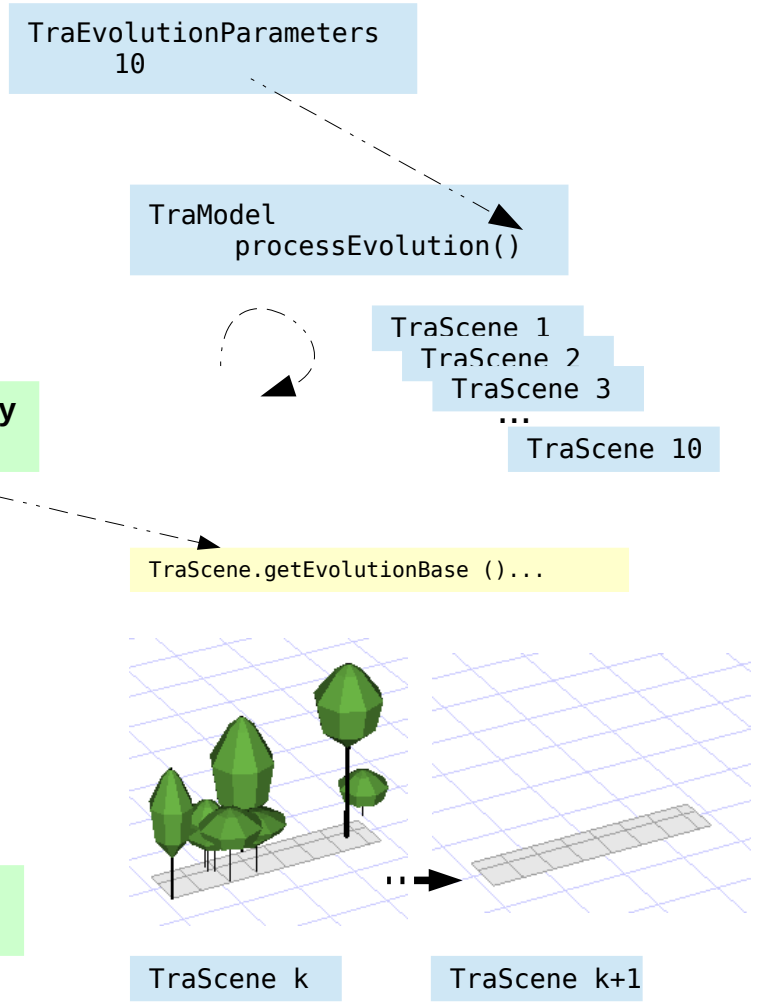
loop on the trees in scene k

create a tree at age k+1

add it in the scene k+1

create a new Step for scene k+1

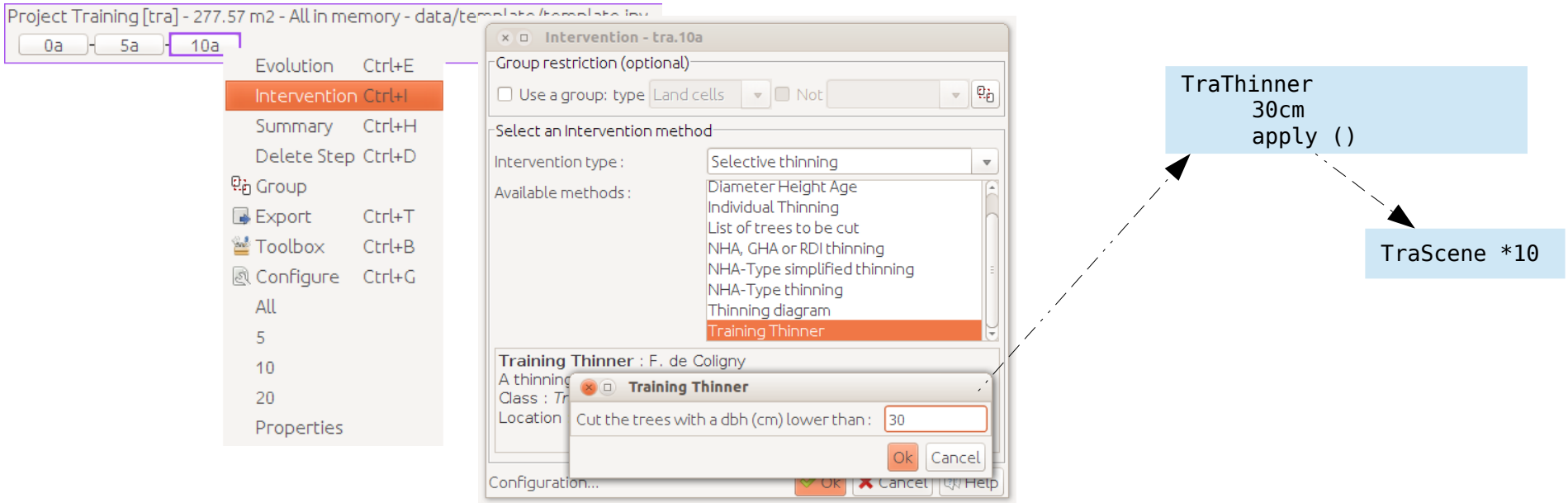
return the last Step



An illustration

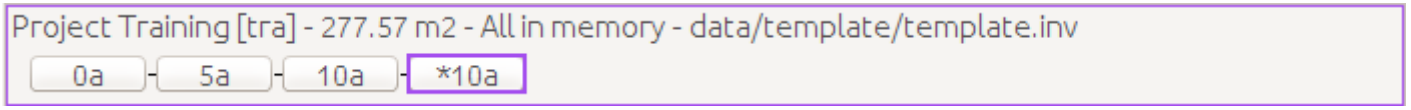
An **intervention** is a particular type of extension

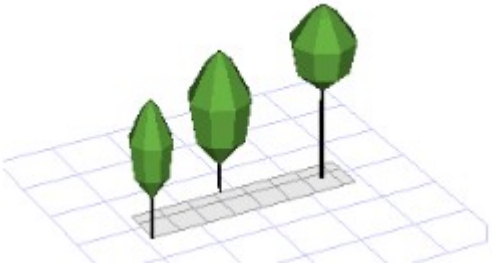
3. Intervention: Right click on step 10 > Intervention > Training Thinner > 30 cm > Ok



What happens:

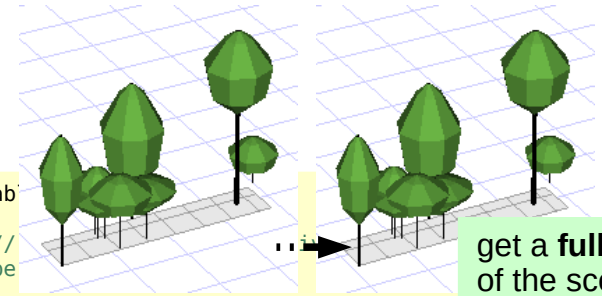
- a copy of the scene below step 10 is passed to the **TrainingThinner**
- the TrainingThinner.apply() method removes the small trees
- TraModel.postIntervention() is called
- a new Step is created to carry the new **TraScene**





An illustration

TraScene.getInterventionBase ()...



An **intervention** is a particular type of extension

Focus: Thinning extension

```
public class TraThinner implements Intervener, Groupab

    private boolean constructionCompleted = false;  //
    private GScene scene;  // Reference scene: will be
    private GModel model;
    protected Collection<Tree> concernedTrees;
    protected double selectionDbh = 20;  // cm, cut no

    /** Init the thinner on a given scene.
     */
    @Override
    public void init(GModel model, Step s, GScene scene, Collection c) {
        this.scene = scene;   // this is referentScene.getInterventionBase ();
        this.model = model;
        if (c == null) {
            concernedTrees = (Collection<Tree>) ((TreeList) scene).getTrees ();  // all trees
        } else {
            concernedTrees = c;  // restrict to the given collection
        }
        constructionCompleted = true;
    }

    /** Makes the action: thinning.
     */
    public Object apply () throws Exception {
        // Check if apply is possible
        if (!isReadyToApply ()) {
            throw new Exception ("TraThinner.apply () - Wrong input parameters, see Log");
        }

        scene.setInterventionResult (true);

        // iterate and cut
        for (Iterator i = concernedTrees.iterator (); i.hasNext ();) {
            Tree t = (Tree) i.next ();

            // do we cut this tree ?
            if (t.getDbh () <= selectionDbh) {  // yes
                // remove the tree
                i.remove ();
                ((TreeList) scene).removeTree (t);
                // remember this tree has been cut
                ((TreeList) scene).storeStatus (t, "cut");

            }

        }
        return scene;
    }
}
```

TraScene 10 dbh

TraScene 10 copy

get a **full copy** of the scene k

```
TraThinner
      30cm
      apply ()
```

TraScene *10

optionaly restrict to a given list of trees

check if dbh is correct (> 0...)

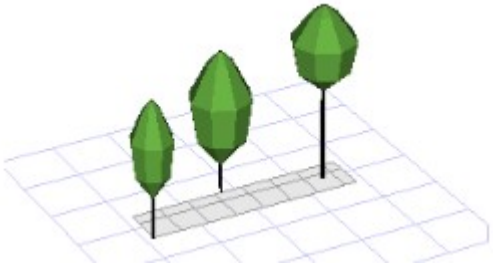
set the intervention boolean: *10

loop on trees

remove the small ones

store them in a cut list

return the *10 scene

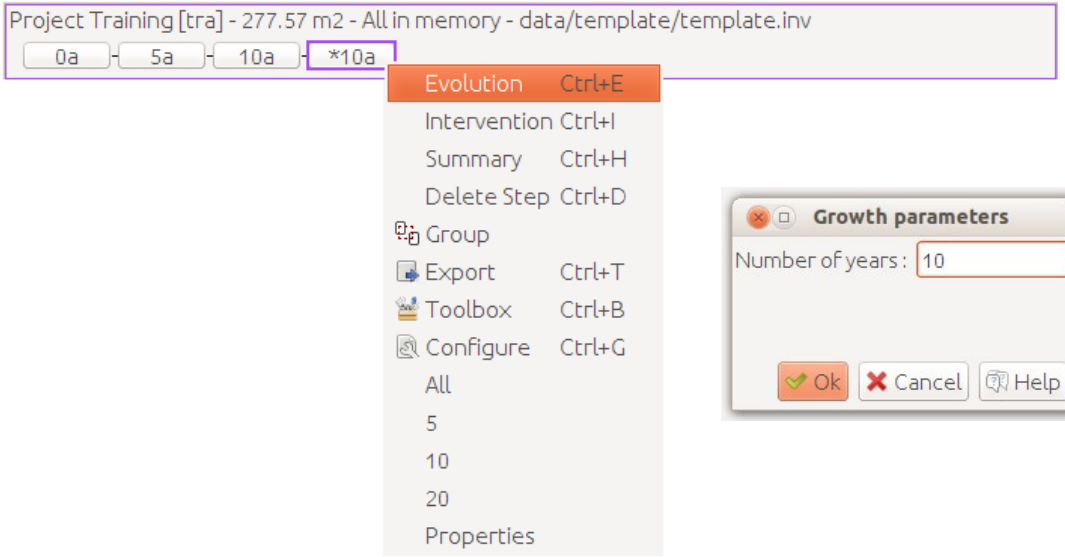


An illustration

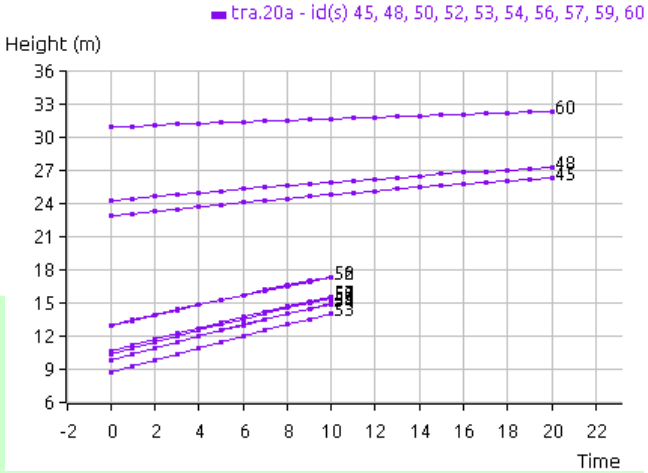
```
TraEvolutionParameters
      10
```

4. Evolution again: Right click on step *10 > Evolution > 10 years more > Ok

```
TraModel
      processEvolution ()
```

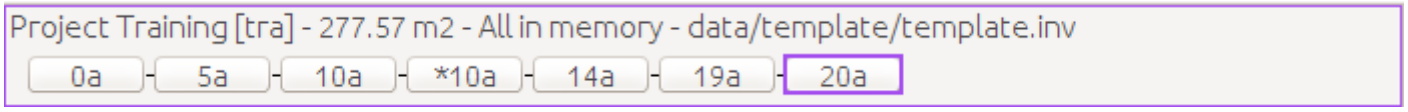


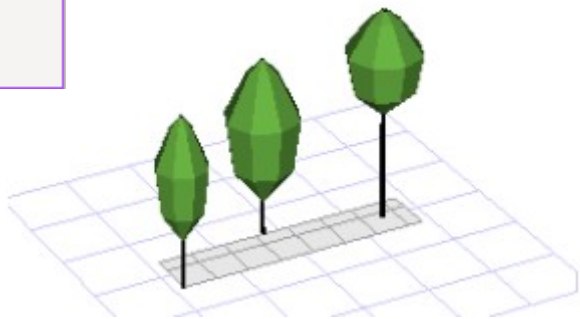

```
TraScene 11
  TraScene 12
    TraScene 13
      ...
        TraScene 20
```



What happens:

- a **TraEvolutionParameter** is created, contains the limit: '10'
- processEvolution() is called and passed the limit
- it loops 10 times to create the 10 scenes dated 11 to 20
- each time, it asks the **Engine** to create a new **Step** to carry the new scene





The Script mode

The capsis framework includes script features

- it is possible to write **Capsis scripts in java**
- they run simulations **without a graphical user interface**
- no change needed in the model implementation
- scripts rely on **C4Script**
- loops, repetitions, exports...

capsis launcher

script pilot

script name

Capsis launching

script execution

```
coligny@marvin-13:~/workspace/capsis4$ sh capsis.sh -p script maddmodule.myscripts.Si
mpleScript
Script: splashscreen was desactivated
64 bits architecture - max memory: 7000 mega bytes
Capsis 4.2.3, (c) 2000-2014 F. de Coligny, S. Dufour-Kowalski (INRA-AMAP) and the Cap
sis modellers
Capsis comes with ABSOLUTELY NO WARRANTY
The core of the Capsis platform (packages capsis.*) is free software
and you are welcome to redistribute it under certain conditions.
Some components in other packages may not be free. See licence files.

Capsis 4.2.3-8202 with pilot capsis.script.Pilot: correct boot at 20 Mar 2014 12:12:0
2 CET
Working dir: /home/coligny/workspace/capsis4

Launching script maddmodule.myscripts.SimpleScript...
Memorizer capsis.extension.memorizer.DefaultMemorizer was correctly set for project P
roject_capsis.app.C4Script
[0->200: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 2
9 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110
 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131
132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 1
53 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 17
4 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195
 196 197 198 199 200]
coligny@marvin-13:~/workspace/capsis4$
```

How to write a script

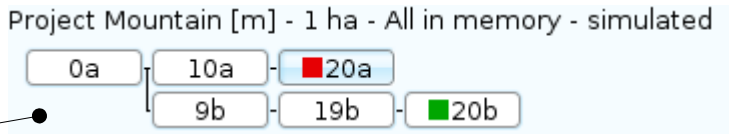
An example (see the documentation page on the Capsis web site)

```
package maddmodule.myscripts;

import maddmodule.model.MaddEvolutionParameters;
import maddmodule.model.MaddInitialParameters;
import capsis.app.C4Script;
import capsis.kernel.Engine;
import capsis.kernel.PathManager;
import capsis.kernel.Step;


/**    A Capsis script, type 1 (preferred):
 *     (a) extends nothing particular,
 *     (b) with a main method (can be launched directly with java),
 *     (c) uses an instance of C4Script.
 *
 *     Two ways to launch it:
 *     (1) sh capsis.sh -p script maddmodule.myscripts.SimpleScript
 *     (2) java -cp class:ext/* maddmodule.myscripts.SimpleScript
 *
 * @author F. de Coligny - 16.9.2010
 */
public class SimpleScript {

       public static void main(String[] args) throws Exception {

        C4Script s = new C4Script("maddmodule");

        MaddInitialParameters i = new MaddInitialParameters (PathManager.getDir ("data") + "/maddmodel/A.inv");

        // init
        s.init(i);

        // evolution
        Step result = s.evolve(new MaddEvolutionParameters (200));

        // save
        Engine.getInstance ().processSaveAsProject (s.getProject(),
                                s.getRootDir() + "/tmp.test.prj");

       }

}
```

A generic approach

The management of **evolution models** has been abstracted:

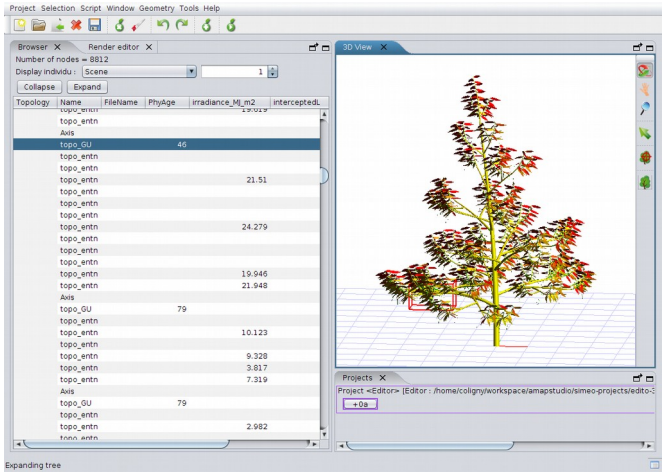
- a generic kernel
- evolution models
- intervention tools
- simulation history
- interactive / script modes
- a generic project manager...

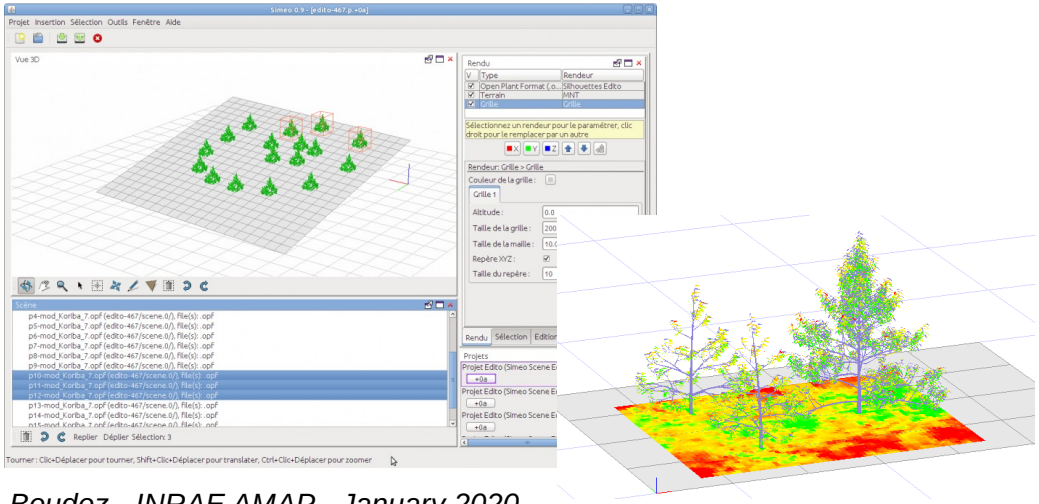


-> the Capsis Framework, may be reused in other applications
to integrate other dynamics models

Other applications: the AMAPstudio software suite for plants architecture modelling

- **Xplo**: Plant Architecture eXploration (S. Griffon, Cirad AMAP)
- **Simeo**: Scene Implantation Manager with Edition by Outline (F. de Coligny INRA AMAP)



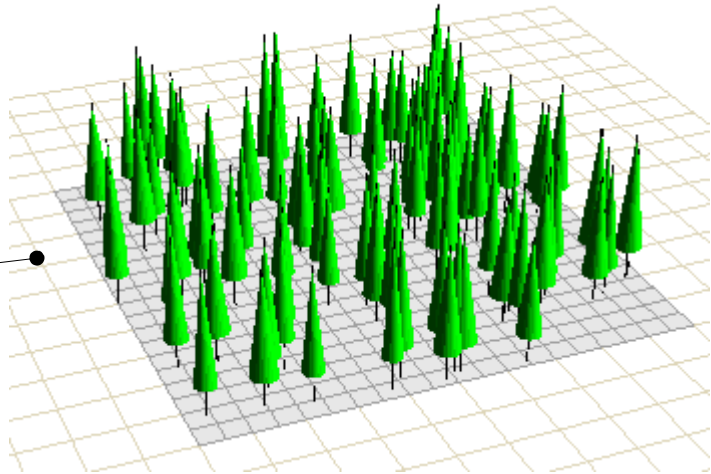


Specifics for forestry

Forestry classes proposals

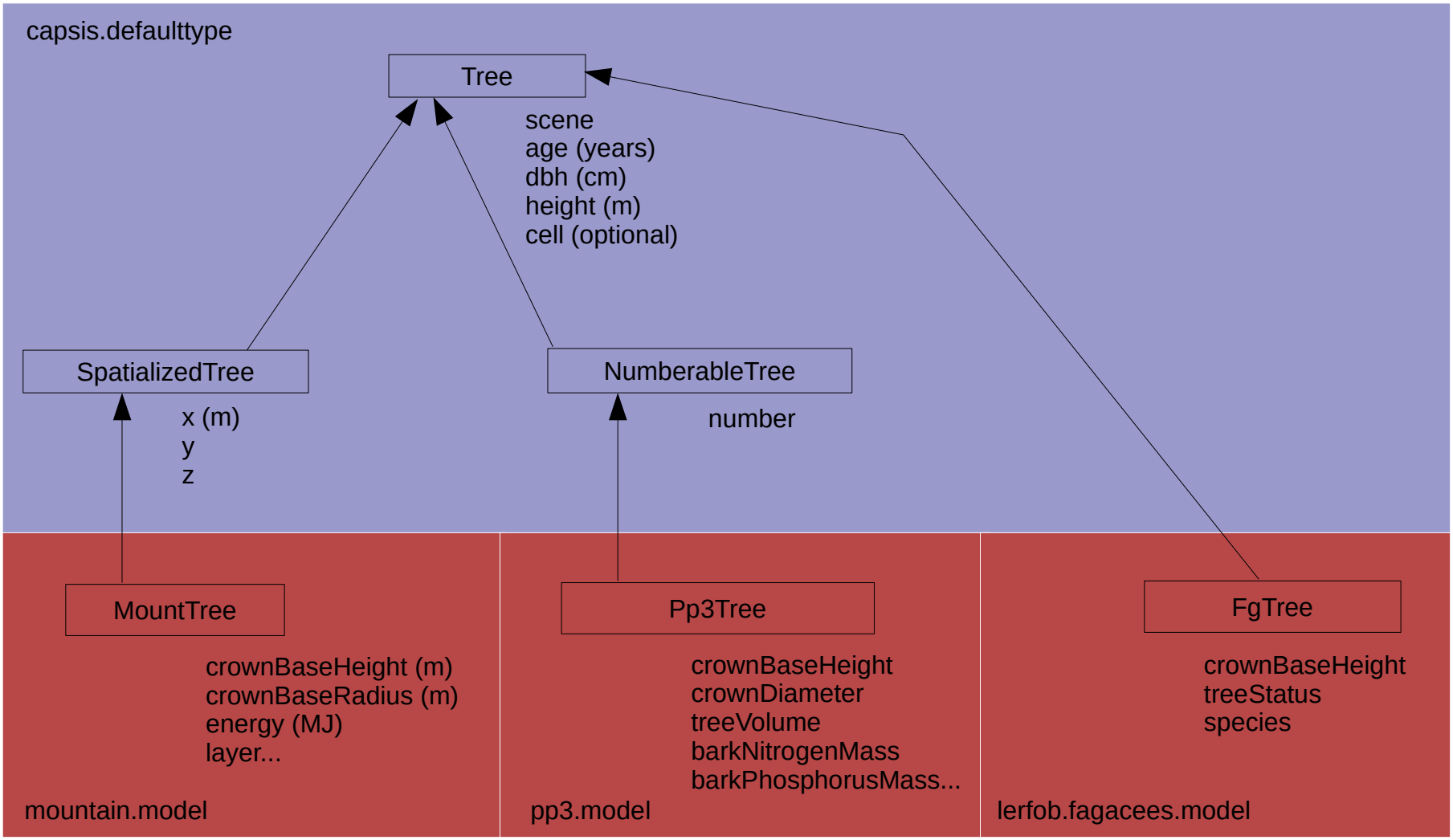
Capsis **proposes** some domain dependent descriptions
-> to be used / subclassed in the modules

- package capsis.defaulttype
- **optional** (can be completely ignored)
- **useful**: some tools may become compatible directly (viewers, interveners...)
- few classes only (on purpose)
- enough to naturally **structure the developments in simple families e**.g. individual based models, spatialised models, with geometry for the plot...



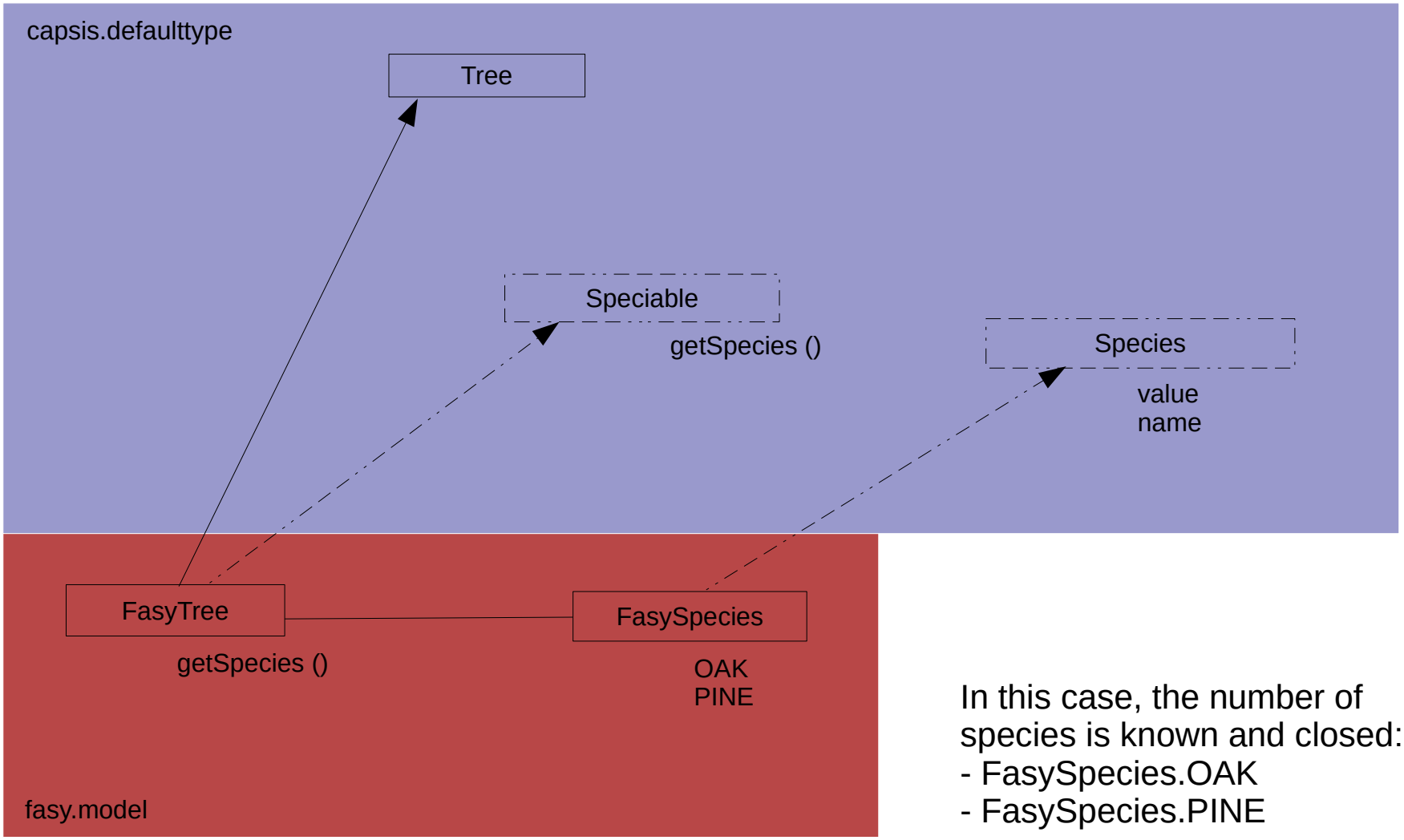
Samsara (B. Courbaud, IRSTEA)

Trees



	mountain	pp3	fagacees
Individual based...	yes	no	yes
Spatialised...	yes	no	no

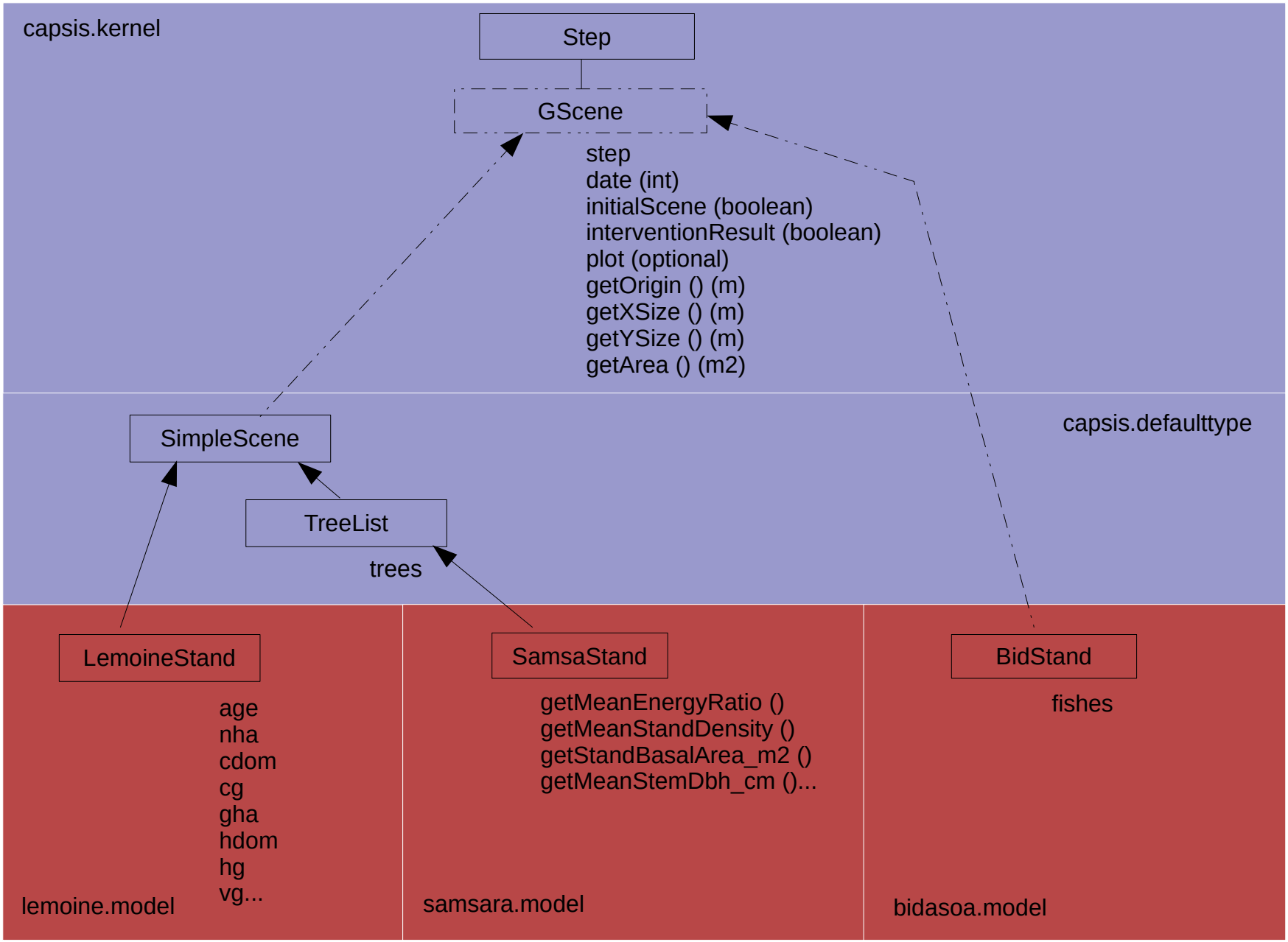
Species



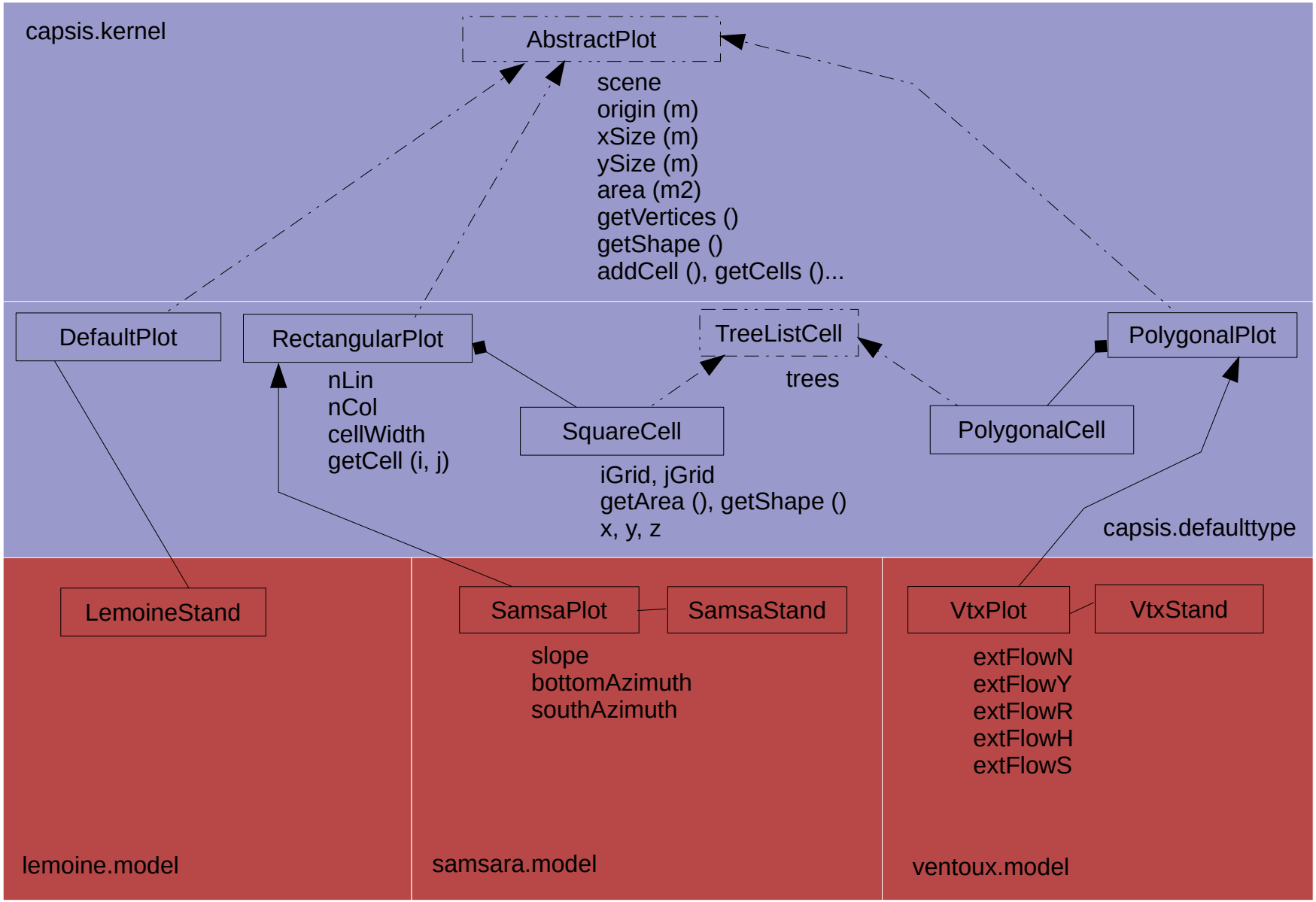
In this case, the number of species is known and closed:
- FasySpecies.OAK
- FasySpecies.PINE

In other cases, a list of species may be created at simulation start from a file

Scenes / stands



Rectangular or polygonal plot



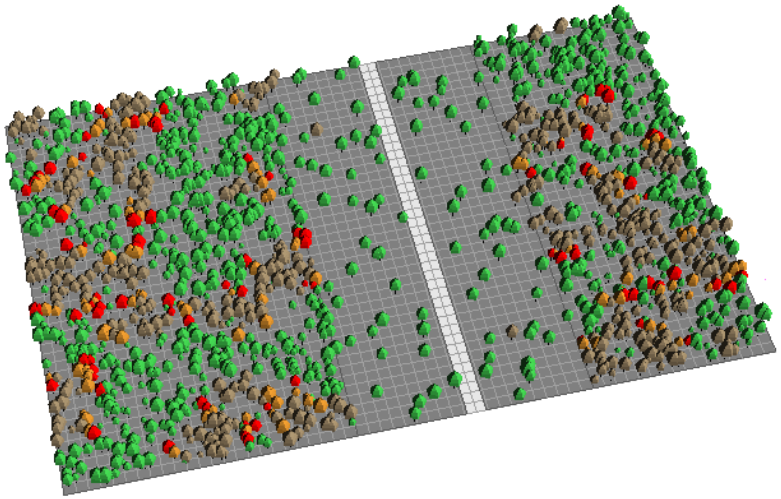
Rectangular plot - neighbourhoods

A feature of the Rectangular plots
- based on a matrix -> implicit neighbourhood
- RoundMask: a convenient method to get the neighbours within a given distance
- may consider the plot as a torus (optional)

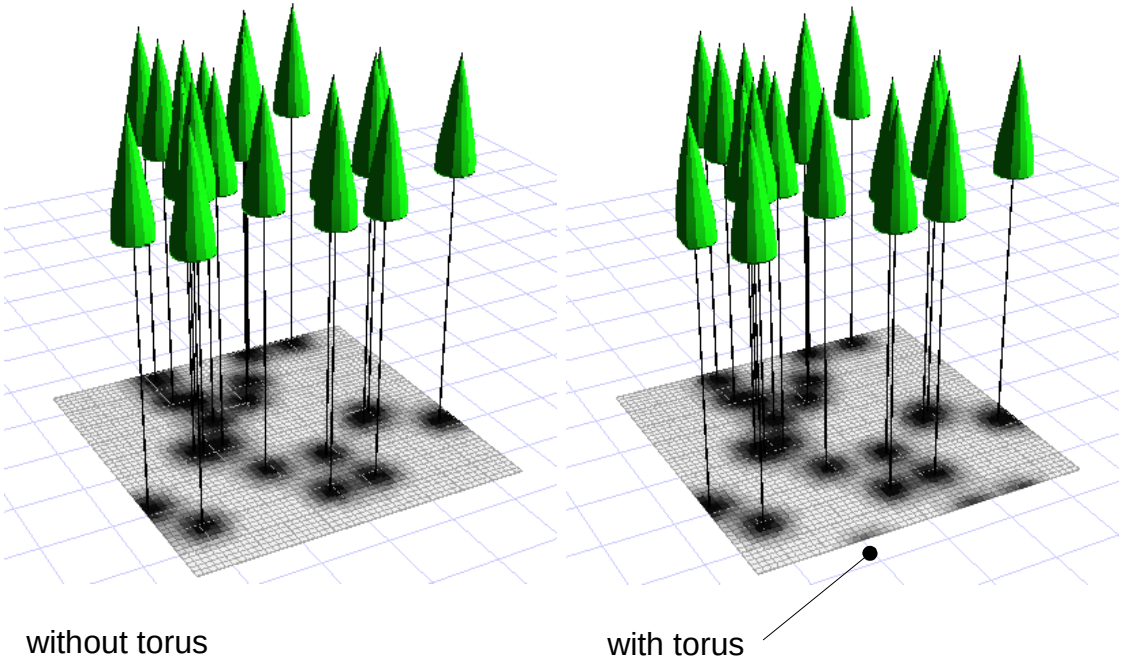
```
// for each source tree we will identify trees within the radius that
// may be potential targets. These trees are currently status 0.
// get the radius
double mr = newStand.getMaxSpreadDistance();
// make a circular search object that will identify trees in range
boolean torus = false;// torus true enables wrapping at boundaries
FiPlot plot = newStand.getPlot();

RoundMask m = new RoundMask(plot, mr, torus);
```

Get the trees near a given tree t

```
Collection ns = m.getTreesNear(t);
```



Beetle spreading (R. Parsons, USDA Forest Service)



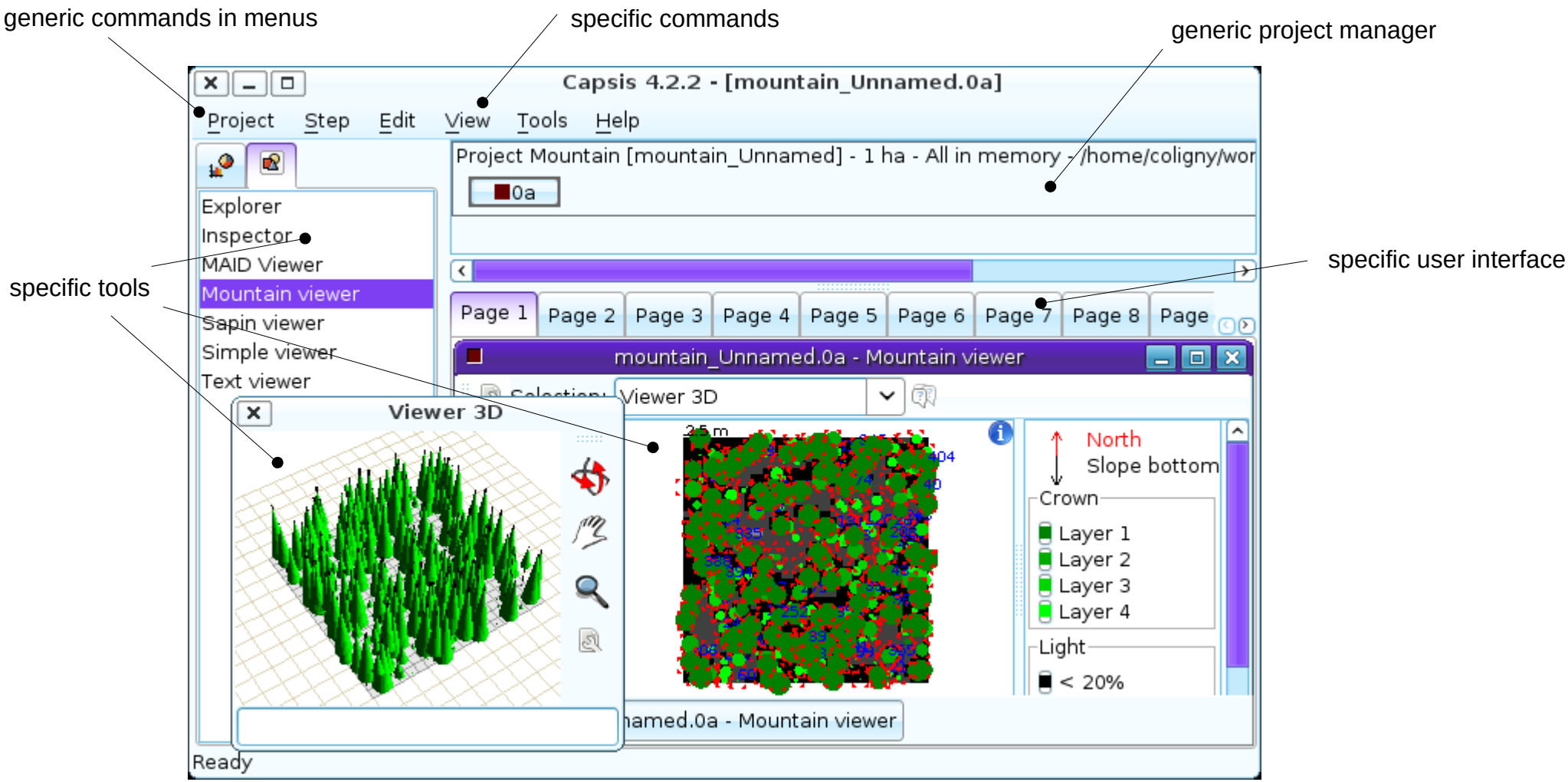
Radiative balance (G. Vincent, IRD AMAP, M. Laurans, Cirad AMAP)

The Capsis application

Graphical user interface

It was built with generic components of the framework (e.g. project manager)...
... and specific components for forestry modelling (e.g. forest stand viewers).

It can be run in french or english.



Mountain model in Capsis (B. Courbaud, IRSTEA)

The application starter

Capsis launchers are provided to launch Capsis under various OS

Windows: capsis.bat
java (...) -cp ./class;./ext/* capsis.app.Starter %1 %2 %3 %4 %5 %6 %7 %8 %9

Linux / Mac: capsis.sh
java (...) -cp ./class:./ext/* capsis.app.Starter $1 $2 $3 $4 $5 $6 $7 $8 $9

capsis.app.Starter (extract): contains the main() method

```
/**   Capsis Starter.
 *    To launch Capsis: new Starter (commandLineArgs).start ();
 *    @author F. de Coligny - september 2001
 */
public class Starter {

  static public final String CAPSIS_VERSION = "4.2.3";

  static private boolean wasCalled;


  /** Deals with main options, then creates the capsis.kernel.Engine.
  *   start () must be called just afterward.
  */
  public Starter (String[] args) {
    if (Starter.wasCalled) {return;}  // can be called only once (could happen in script mode)
    Starter.wasCalled = true;
    ...
```

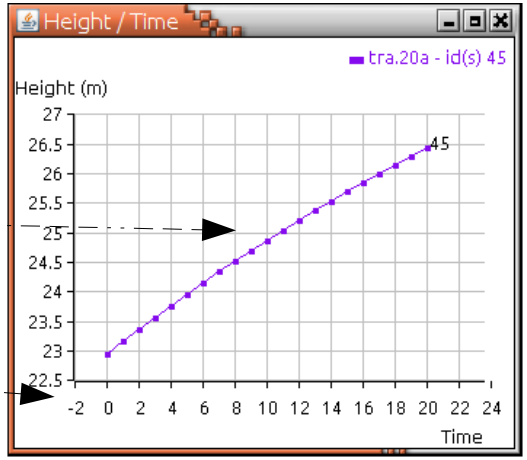
The extension manager (optional)

Capsis uses an extension manager:

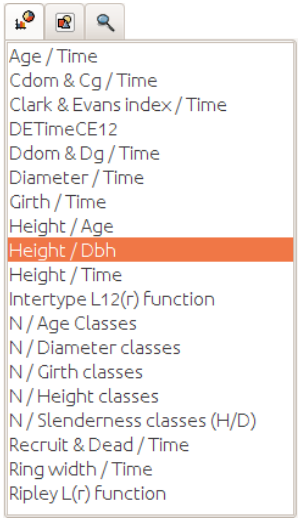
- the init() method declares the **application specific extension types**
- init() **loads the extension file**
- the extension manager is **created in the Starter** constructor and passed to the Engine

extension types for CapsisExtensionManager

```
/**    Declaration of the extension types
 */
public static void initTypes(ExtensionManager em) {

  em.declareType(CapsisExtensionManager.GENERIC_TOOL, GenericTool.class);

  em.declareType(CapsisExtensionManager.DATA_EXTRACTOR, DataExtractor.class);
  em.declareType(CapsisExtensionManager.IO_FORMAT, IFormat.class);
  em.declareType(CapsisExtensionManager.IO_FORMAT, OFormat.class);

  em.declareType(CapsisExtensionManager.STAND_VIEWER, StandViewer.class);
  em.declareType(CapsisExtensionManager.DATA_RENDERER, DataRenderer.class);
  em.declareType(CapsisExtensionManager.INTERVENER, Intervener.class);

  em.declareType(CapsisExtensionManager.MODEL_TOOL, ModelTool.class);
  em.declareType(CapsisExtensionManager.FILTER, Filter.class);
  em.declareType(CapsisExtensionManager.TREELOGGER, TreeLogger.class);


  em.declareType(CapsisExtensionManager.ECONOMIC_FUNCTION, EconomicFunction.class);
  em.declareType(CapsisExtensionManager.MEMORIZER, Memorizer.class);

  em.declareType(CapsisExtensionManager.LOLLYPOP, Lollypop.class);
  em.declareType(CapsisExtensionManager.SPATIALIZER, Spatializer.class);
  em.declareType(CapsisExtensionManager.WORKING_PROCESS, WorkingProcess.class);

  em.declareType(CapsisExtensionManager.OBJECT_VIEWER, ObjectViewer.class);

  em.declareType(CapsisExtensionManager.GROUPER_DISPLAY, GrouperDisplay.class);

  em.declareType(PaleoExtension.class.getSimpleName(), null);

}
```



Special components in the GUI propose the user to run extensions



Organisation on disk

Installation directory: **somewhereOnDisk/capsis4/**

capsis4/	Linux / Mac / Windows scripts to launch **ant** and **capsis** License.txt: main **license file** Readme.txt
src/	all **source codes** for capsis and all the modules and extensions
class/	all **binaries** (.class)
data/	**sample** data to run demonstrations
etc/	**parameters** (capsis.models, extension.list...)
ext/	**additional libraries** in .jar files (3D, maths...)
tmp/	convenient
var/	**logs** and other files with a variable length

A way to get directory paths whatever the Operating System (Windows, Linux, Mac)

```
String installDir = PathManager.getInstallDir ();
String inventoryFile = PathManager.getDir ("data")+"/mountain/Aleatoire.inv";
```

Main packages

All source codes (.java) are in **src/** ...

capsis/ app/	capsis Starter, ExtensionManager and Script superclass
commongui/	gui common components of the capsis framework
defaulttype/	domain dependent proposals: Tree, TreeList...
extension/	generic extensions: compatible with more than one module
extensiontype/	the type of extensions Capsis knows
gui/	the gui pilot: Capsis graphical user interface
kernel/	the capsis kernel
lib/	integrated domain dependent libraries: genetics...
script/	the script pilot of the capsis framework: to write simulations scripts in java
util/	some utilities

also in **src/**: all the modules source codes...

e.g. the stretch module:

stretch/ extension/	Stretch specific extensions
gui/	specific dialogs for the module
model/	the implementation of the Stretch model
myscripts/	scripts to run simulations at night
...	

The Capsis web site

http://capsis.cirad.fr
http://www.inra.fr/capsis

Capsis login

Capsis

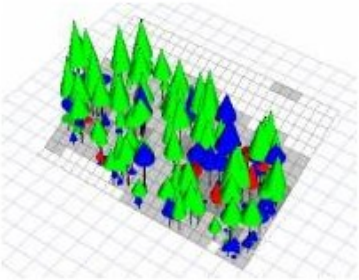
Computer-aided projection of strategies in silviculture

Sitemap Login

- Home
- Presentation
- Download
- FAQ
- Screenshots
- Charter
- Publications
- Documentation
- Projects
- Development
- Contact

Capsis is a simulation platform for forestry growth / dynamics models. It is a tool for forest scientists, forest managers and education. It has been developed in the AMAP laboratory since 1999.

See the Capsis presentation page, download the Capsis brochure (fr) or go to the projects page for a quick overview. Have a look at the documentation page or the Capsis impacts 10 minutes video (fr) for more details.



Community news

- Nicolas Martin, Julien Ruffault and Francois Pimont (INRAE URFM Avignon) came 3 days in december to work on the **introduction of the Sureau model in Capsis. A prototype was built together.**. *fc-6.1.2020*
- Manon Helluy (IRSTEA UR RECOVER, Aix-en-Provence) came 1 day to AMAP to add a **water stress index in soil** and add a **new competition index**. *nb-6.12.2019*
- Benoit Courbaud (IRSTEA Grenoble) came 3 days to AMAP to add **an ecological index based on micro dendro habitats** to the trees in Samsara2. *fc-3.12.2019*
- Gilles Le Moguedec (INRA AMAP) spent **one week at UTAD in Vila Real, Portugal**, to work on the optimisation part of ModisPinaster with Teresa Fonseca and Adelaide Cerveira. *fc-4.11.2019*
- Manon Helluy (IRSTEA UR RECOVER, Aix-en-Provence), Philippe Balandier (IRSTEA UR EFNO, Nogent-sur-Vernisson) and Nicolas Donès (INRA UMR PIAF, Clermont-Ferrand) came 5 days in AMAP to **add a water transfer package in the Regeneration library**. This package contains the **water transfer process in soil** previously implemented in **RReShar module** and the **water transfer in aerial plant of vegetation** has been added. *nb-21.10.2019*
- François Pimont (URFM, Avignon) spent 2 days in AMAP to work on a **general upgrade of the Fuel Manager**, with simpler starting options and a new 3D viewer. *fc-4.10.2019*

The Documentation page

Capsis login

Capsis
Computer-aided projection of strategies in silviculture

Sitemap Login

- Home
- Presentation
- Download
- FAQ
- Screenshots
- Charter
- Publications
- Documentation
- Projects
- Development
- Contact

The Capsis documentation

1. Using Capsis: documentation for the end-users

This section is for the **Capsis end-users** who were given a Capsis installer (Zip or *IzPack* jar version) by a Capsis modeller (see the Capsis charter). Your installer generally contains only few models (the other models in Capsis are not all free). For a complete and up to date list of projects, see the models page.

- **Installation Guide**
- Tutorial(fr) / Tutorial(en)
- A Capsis quick tutorial: create a ModisPinaster project, build silvicultural scenarios, check the results in the integrated charts
- A Capsis quick tutorial: create a Mountain project, build interactively a silvicultural scenario, compare various interventions
- FAQ (frequently asked question)
- Diagrams and Viewers
- **The Capsis Manual**

2. Developing in Capsis: documentation for the modellers

This section is for the **Capsis modellers**. You have the complete version of Capsis including all the models thanks to the Capsis charter. You can find here documentations about Capsis, Java and various tools used in the Capsis project as well as reports, training courses and various other documents.

2.1 Getting started

- **Who can join the Capsis project ?**
- **Developing in Capsis with SVN and Ant**
- Troubleshooting: Check the 'Capsis trouble' forum
- Code editors: Eclipse, Notepad++, TextPad, SciTE
- Amap-dev : development web site
- SVN update and commit
- Ant commands
- Capsis now runs under Java 1.8

- The Capsis documentation
  - 1. Using Capsis: documentation for the end-users
  - 2. Developing in Capsis: documentation for the modellers
    - 2.1 Getting started
    - 2.2 Developing modules
    - 2.3 Extensions
    - 2.4 Scripting
    - 2.5 Documenting and distributing your model
    - 2.6 Good practices
    - 2.7 Libraries
    - 2.8 Specific concerns
    - 2.9 Technical topics for skilled developers
  - 3. Training lessons
    - Last version
    - Former training versions
  - 4. Presentations and other documentations
  - 5. Reports of Capsis annual meetings
  - 6. Conventions, norms
  - 7. References
  - 8. Useful links

The up to date project list

Capsis login

Capsis

Computer-aided projection of strategies in silviculture

Sitemap Login

- Home
- Presentation
- Download
- FAQ
- Screenshots
- Charter
- Publications
- Documentation
- Projects
- Development
- Contact

Capsis projects

This page contains the list of active projects in the Capsis platform. You can find here about 80 forest growth or dynamics models (i.e. Capsis modules) and several applicative libraries.

Some of these components are distributed with an **open-source licence**, they are packaged in an installer you can download from the Download page. To get more details regarding a given project, please mail to the contact person.

Search the list: *the most recent projects are at the top of the list. You can sort the table by clicking on the headers.*

Project	Description	Contact, starting date
Ibasam	A metapopulation model to study salmon species population eco-evolutionary dynamics, accounting for river and marine ecosystems. This is the metapopulation version of the Ibasam software written by Cyril Piou and Etienne Prévost. Keywords : fish, dynamics, genetics, climate	Mathieu Buoro, Jacques Labonne (INRA Ecobiop) June 2019
EcoAF	An Agroforestry model. Keywords : Agroforestry, Genetics	Frédérique Santi , Tina Meziani (INRA BioforA), Christophe Sotteau (AGRO-ECO Expert) May 2018
Luberon2	A distance-dependent tree model for Cedrus with genetics. It is a new version of the **Luberon** model developed in the context of the RMT-IGS project: "Assessment of the genetic impacts of adaptation-oriented sylviculture practices". Keywords : Genetics, Tree localisation model, Cedrus	François Lefèvre , Claire Godineau (INRA URFM) July 2017
Salem	Salem is a StAnd LEvel Model for various species in pure or mixed stands. Salem is a combination of models based on NFI data for a good robustness and models based on tree cores to include climate effect. It allows to simulate stand growth according to IPCC climate projection. Keywords : Growth model, Climate effect	Patrick Vallet (IRSTEA Grenoble) June 2017
AlloStand3D	A tree localisation algorithm from dbh inventories and crown allometries based on 3D crown overlap constraints Keywords : Allometry, Tree localisation model, Voxelized tree	Philippe Verley , Nicolas Barbier, Pierre Ploton (IRD AMAP)

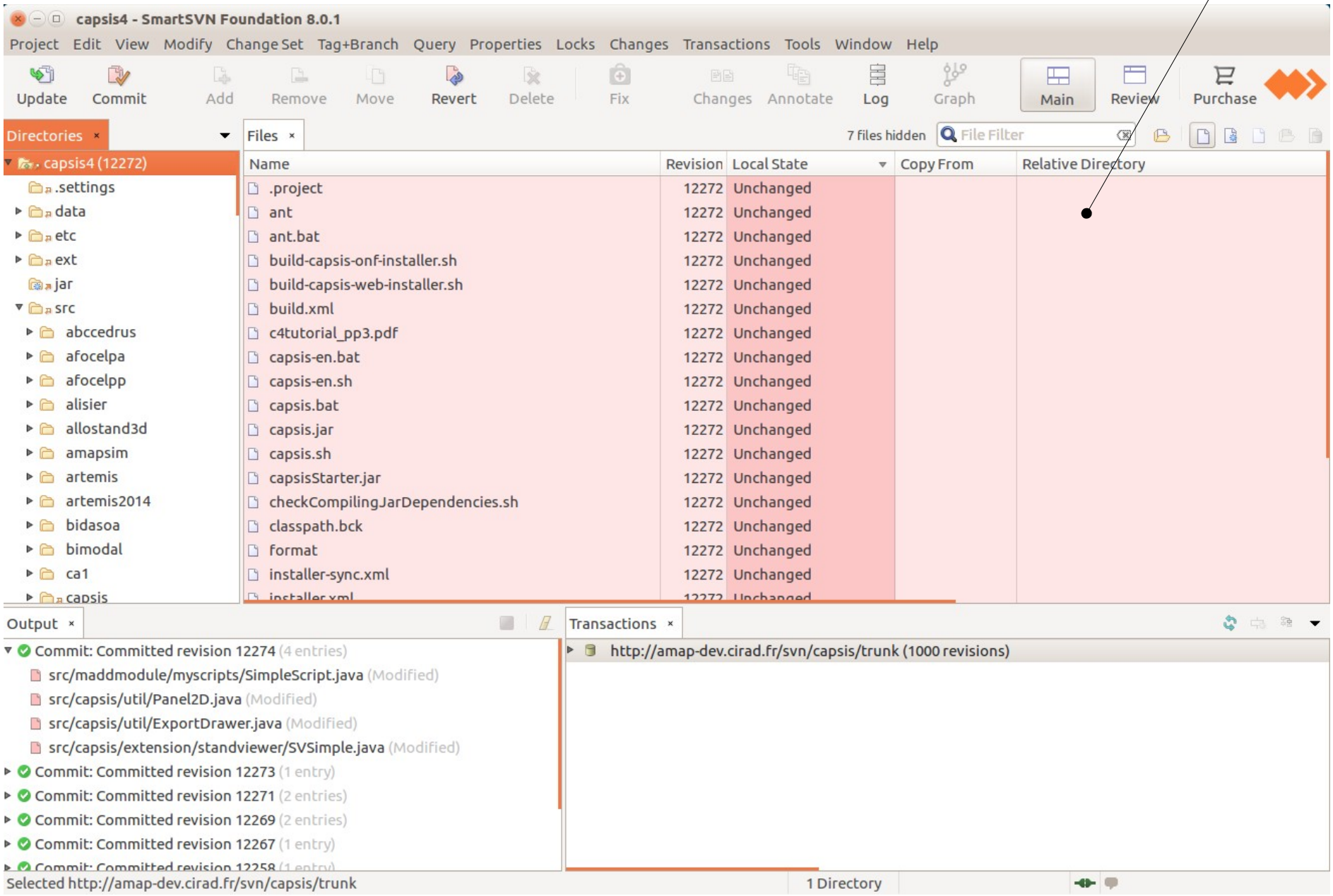
Click to create

Your pages if you like

Versioning

SVN server / SmartSVN client to get / share changes with other modellers

Capsis login



The Project management tool

http://amap-dev.cirad.fr/projects/capsis

Management of the unique Capsis login, for Capsis modellers only

Capsis login

Home My page Projects Administration Help

Logged in as coligny My account Sign out

Capsis

Search: Capsis

Overview Activity Roadmap Issues New issue News Wiki Forums Files Repository Settings

Overview

New subproject Close

Computer-Aided Projection of Strategies In Silviculture

- Homepage: http://capsis.cirad.fr
- Subprojects: JackPine, Lerfob, QuebecMRNF, Stretch

Issue tracking

- Bug: 28 open / 280
- Feature: 19 open / 165
- Support: 0 open / 9

View all issues

Members

Manager: François de Coligny

Developer: Nicolas Beudez, Raphaël Martin, Samuel Dufour-Kowalski, Sébastien GRIFFON

Modeler: alexandre Piboule, Alexis Achim, Alice Duquesnoy-Mitjavila, Ambre LE FERREC, Amit Verma, Annabelle Amm, Arnaud Caudron, Astrid de Pierpont, Aurélie Querné, Barry Gardiner, Benoit Courbaud, Bernard Prévosto, Braeden Young, Brian Chantomaud, Brieuc Cornet, Bérangère Leys, Cathleen Petit, Celine Blitz Frayret, Charles Ward, Christian Dupraz, christian pichot, Christian Pilaar, Christine Deleuze, Christophe Orazio, Christophe Pradal, Claire Cornillier, Claire Godineau, Cécile Madelaine-Antin, Céline Meredieu, Daniel Mailly, Dauzat Jean, degi harja, Denis Hache, Emmanuel Duchateau, Eric Beaulieu, Eric Mermin, Eric Rigolot, Erwan LE TEXIER, Estelle Vial, Etienne Klein, Filip Havreljuk, Florian Delerue, Francois Cerisier, Francois de Coligny (tester), Francois Lefevre, Francois NINGRE, Francois Pimont, Frank Berninger, François Courbet, François Guillemette, François Guiton, François GUITON, François Herman, Frédéric André, Frédéric Mothe, gauthier boaglio, Gauthier Ligot, Gaël HUBERT, Ghislain Vieilledent, Gianni Boris Pezzatti, Gilles Le Moguédec, grégoire talbot, Grégoire Vincent, Guillaume CORNU, Guillaume Lagarrigues, Haizhou LI, haythem ben touhami, hendrik davi, Holger Wernsdorfer, Hong Guo , Hugues Power, Isabelle Auger, Isabelle Chuine, Jacques Labonne, Jean-Baptiste Pichancourt, Jean-François Lavoie, Jean-Noel Candau, Jeremy Andre, Jordane Gavinet, Jordi Gil, Julie Gauzere, Julien Blanco, Julien Sainte-Marie, Jérôme Maurice, Kana Kamimura, Katalin Csillery, Kenneth Herdy, Kunstler Georges, Kyle Doyle, Laureline Claessens, Laurent Dedry, Laurent Saint-Andre, LECOMTE Isabelle, Leonardo Gallo, lingxia hong, Louis de Wergifosse, louise chevalier, Lucas Wells, Malek Haddad, Manon Helluy, Marc Du Sablon, Margot REGOLINI, Maria Adelaide da Cruz Cerveira, mariane alleaum-benharira, Marianne Bernard, marie ange ngo bieng, Marie Gosme, marilyne laurans, Marion Jourdan, Mathieu Fortin, Mathieu Jonard, Michel Chartier, Mickael Chauvet, Mikhail Pitchugin, Minna Pulkkinen, Nathaniel Osborne, Nicolas BILOT, Nicolas Donès, Nicolas Lambert, Nicolas Martin, Nicolas Robert, Noemie Gaudio, Oana VIGY, Olivier De Thier, Pascal Obstétar, Patrick Vallet, Paul Geoffroy, Perin Jérôme, Philippe Balandier, Philippe Dreyfus, Philippe VERLEY, priscilla cailly, Quentin Molto, Raul Garcia, Robert Schneider, Rubén Manso, Russell Parsons, Rémi CARDINAEL, sadi aid, Samuel Quevauvillers, Sandrine Brèteau-Amores, sandrine perret, Sergio de Miguel Magaña, sidonie artru, Simon Martel, simon pastorello, Sitraka Andrianarisoa, Stephanie Ewen, Susy Domenicano, Sylvie Oddou-Muratorio, Sébastien Cavaignac, Sébastien Roux, Teresa Fidalgo Fonseca, test hudson, Thierry Faure, Thierry Fourcaud, Thierry LABBE, Thomas Bronner, Thomas Cordonnier, Thomas CORDONNIER, Thomas Damestoy, Thomas Perot, Tony Franceschini, Valentin Journé, Valentine Lafond, Venceslas Goudiaby, Véronique CUCCHI, wu ma, Xavier Morin, xiangdong lei, Yassine MOTIE, Zhun Mao, Élise Filotas, 琛琛 沈

User: de Coligny Francois, Jan Thiele, JM SARAT, john price, Marius Teodosiu, Souleymane DICKO, web download

The Capsis javadoc

http://amap-dev.cirad.fr/projects/capsis

Capsis login

Home My page Projects Administration Help Logged in as coligny My account Sign out

Capsis Search: Capsis

Overview Activity Roadmap Issues New issue News Wiki Files Repository Javadoc Settings

Upload documentation

Overview Package Class Tree Deprecated Index Help

Package capsis.kernel

Interface Summary	
Cell	Interface for plot cells.
DateCorrectable	Relative to Capsis date correction: possibility to add an int offset date for root scene and a int coefficient for step duration.
EvolutionParameters	Interface for the parameters needed for an evolution stage for all modules.
GScene	GScene is the superclass for the modules main data structure (e.g. a forest OR a plantation in Capsis / an ArchiTree in Xplo / a composite scene in Simeo...).
InitialParameters	This interface describes the initial parameters of a model in the capsis.kernel.
MethodProvider	MethodProvider: interface of method providers (mp) for modules.
Plot<T extends Cell>	A plot is a geometric description of the terrain associated to the GScene.

Class Summary	
AbstractMethodProvider	AbstractMethodProvider: an abstract superclass for the method providers.
AbstractPilot	AbstractPilot is a superclass for all Capsis pilots.
AbstractPlot<T extends Cell>	An abstract superclass for the plot subclasses.
AbstractPlot.Immutable	This inner class describes the properties that do not change when considering several instances of AbstractPlot in the Project time history.
AbstractSettings	AbstractSettings is a Serializable superclass for all the model settings.
Engine	Engine of the application.
GModel	GModel is an abstract superclass for all the 'model classes' of the modules.
IdCard	IdCard is an identity Card of the model, given for each model, used by the Engine to instanciate the model's main class.
JavaAgent	This Agent helps getting information at runtime on the jvm.
MemorizerFactory	Memorizer factory creates memorizers.
ModelManager	Manages models.
Options	Evaluates the options of the capsis.kernel.
PathManager	PathManager can find the installDir.
Project	A Project in the Capsis kernel.
ProjectIdCard	Used to mark the project files when they are saved.

The reference publication

Annals of Forest Science (2012) 69:221–233
DOI 10.1007/s13595-011-0140-9

ORIGINAL PAPER

Capsis: an open software framework and community for forest growth modelling

Samuel Dufour-Kowalski · Benoît Courbaud · Philippe Dreyfus · Céline Meredieu · François de Coligny

Including a list of 50 integrated models (2012)

Table 2 Modules in Capsis: scope and main characteristics (see also Online resource 1)

Climatic zone	Stand composition[a]	Stand structure	Simulation levels[b]	Main processes[c]	Specific features[d]	Species[a]	Module name
Temperate	Pure	Regular	Average tree, stand	Gr (DI), Mo, Re, Di		*Cytisus scoparius*	Cytisus
			Average tree, stand	Gr (DI), Mo	Phy	*Abies alba, Fagus sylvatica*	Dynaclim
			Average tree, stand	Gr (DI), Mo	Phy, Ge	*A. alba, F. sylvatica*	PhysioDemoGenetics
			Average tree, stand	Gr (DI)	Th	*Pinus pinaster*	Lemoine
			Average tree, stand	Gr (DI)		Many species	Natura
			Average tree, stand	Gr (DI)		*Eucalypt* ssp., *Populus* ssp.	Regix
			Tree, stand	Gr (DI), Mo	Th	*A. alba*	Abial
			Tree, stand	Gr (DI), Mo		*Picea alba*	Afocelpa
			Tree, stand	Gr (DI), Mo		*P. pinaster*	Afocelpp
			Tree, stand	Gr (DI), Mo		*Cedrus atlantica*	CA1
			Tree, stand	Gr (DI), Mo		*Pseudotsuga menziensii, Larix* spp., *Picea abies*	Douglas
			Tree, stand	Gr (DI), Mo	Th	*Quercus petraea, F. sylvatica*	Fagacees
			Tree, stand	Gr (DI), Mo	Br	*Pinus halepensis*	Fiesta/NRG
			Tree, stand	Gr (DI), Mo		*Pinus nigra laricio*	Laricio
			Tree, stand	Gr (DI), Mo		*Abies balsamea*	Matapedia
			Tree, stand	Gr (M/D), Mo	Th	*P. pinaster*	ModisPinaster
			Tree, stand	Gr (DI)	Br	*Pinus radiata*	NZ1
			Tree, stand	Gr (DI), Mo	Br	*Pinus nigra nigra*	PNN/Pnn2
			Tree, stand	Gr (DI)		*P. pinaster*	PP3
			Tree, stand	Gr (DI), Mo		*Q. petraea*	QS1
			Tree, stand	Gr (DI), Mo		*Pinus sylestris*	Sylvestris
			Average tree, stand, forest	Gr (DI)	Th	2 virtual species	Mustard
			Tree, stand, forest	Gr (M/D)	Th	Many species	IFNCA
			Tree, stand, forest	Gr (DI), Mo, Re, Di	Ge	*C. atlantica*	Luberon
			Tree, stand, forest	Gr (DI), Mo		*Pinus contorta*	MPB
			Tree, stand, forest	Gr (DD), Mo, Re		Many species	Presage
			Tree, stand, forest	Gr (DI), Mo, Re	Ge	*Q. petraea*	Quercus
			Tree, stand, region	Gr (DI), Mo		*Pinus sylvestris, Quercus* sp.	Simmen
			Tree, stand, region	Gr (DI)	Th	*P. pinaster*	Sylvogene
		Irregular	Tree, stand	Mo, Re, Di	Ge	*Prunus mahaleb*	Prunus
			Tree, stand, forest	Re, Di		*C. atlantica*	Abccedrus
			Tree, stand, forest	Gr (DI), Mo, Re, Di		*A. alba*	Migration
			Tree, stand, forest	Gr (DD), Mo, Re		*P. abies*	Mountain
	Mixed	Regular	Average tree, stand	Gr (DI)		*P. abies, A. alba*	Melies
			Tree, stand	Gr (DI), Mo		*Q. petraea, F. sylvatica, P. sylvestris*	Fagacees–Sylvestris
			Tree, stand	Gr (DD)		*Q. petraea, F. sylvatica, P. sylvestris*	Oakpine1
			Tree, stand	Gr (DI)		*Q. petraea, F. sylvatica, P. sylvestris*	Oakpine2
			Tree, stand	Gr (M/D), Mo	Th	*Larix olgensis, Picea jezoensis, Abies nephrolepis*	LSFMGM
			Tree, stand, forest	Gr (DI), Mo		*Pinus banksiana, Picea glauca, Picea mariana*	JackPine
			Tree, stand, region	Re		Many species	Succes
		Irregular	Tree, stand	Gr (DD)		*Populus* spp., *Juglans nigra x regia, Prunus avium*, crops	Hi-sAFe
			Tree, stand	Re		Broadleaved species of north-eastern France	Regelight
			Tree, stand	Gr (DD), Mo, Re		*Quercus* sp., *P. sylvestris*	RReShar
			Tree, stand	Gr (DD), Mo, Re		*Acer saccharum, Betula alleghaniensis, Fagus grandifolia*, other broadleaves	Samare
			Tree, stand	Gr (DD), Mo, Re		*P. abies, A. alba*, other species	Samsara
			Tree, stand, forest	Gr (DI), Mo, Re, Di	Ge	*Sorbus torminalis, Q. petraea, F. sylvatica*	Alisier
			Tree, stand, forest	Gr (DI), Mo, Re, Di	Ge	*F. sylvatica*, virtual species	TranspopRege
			Tree, stand, forest/landscape	Gr (DI), Mo, Re, Di	Ge, Th	*A. alba, F. sylvatica, P. nigra nigra, P. sylvestris, Pinus uncinata*	Ventoux/VentouG
			Tree, stand, forest	Gr (DI), Mo	Th	*P. halepensis, P. sylvestris, P. pinea, P. pinaster, P. nigra laricio, P. nigra nigra*	FireParadox
			Tree, stand, region	Gr (DI), Mo, Re	Th	Many species	Artemis
Subtropical	Pure	Regular	Average tree, stand	Gr (DI)	Th	*Pinus massoniana*	ISGM
Tropical	Pure	Irregular	Tree, stand	Gr (DI)		*Eucalyptus* spp.	Eucalypt
			Tree, stand	Gr (DD)		*Rhizophora* spp.	Mangrove
			Tree, stand	Gr (DD), Re		*Avicennia* spp., *Rhizophora* spp.	Paletuviers
			Tree, stand, forest	Gr (DI), Re	Ge	*Vitellaria paradoxa*	Karite
	Mixed	Irregular	Tree, stand	Gr (DD), Mo, Re		Many species	Selva
			Tree, stand	Gr (DD), Mo, Re	Br	Many species	Stretch

[a] "Pure" with a list of species means that the model either works with a pure stand of one of these species at a time or can simulate concurrently several pure stands of different species in the same forest

[b] Scale levels at which the model works or the module gives outputs (with more or less details depending on the model)

[c] Forest dynamics processes: growth (diameter and, possibly, height) (Gr), mortality (Mo), recruitment/regeneration (Re), dispersal (seeds/seedlings) (Di). Details for growth submodel: distance-independent tree growth (Gr (DI)), distance-dependent tree growth (Gr (DD)), growth submodel relying on a transition matrix or a diameter distribution curve (Gr (M/D))

[d] Genetics (Ge), ecophysiological processes (Phy), branching model (Br), automated thinning (in addition to Capsis' interactive intervention tools) (Th)

To go further

Become a Capsis modeller

accept the charter
join the community and integrate your forest model in Capsis

Get a starting stage for your project

with a developer (2 or 3 days)
get your Capsis login
set together the structure of your model

Go back home with the survival kit

a simple editor
a simple versioning tool
the skeleton of your Capsis module

And continue the integration by yourself

with the remote support from the developers
and occasional meetings when needed