The Capsis Training
Computer Aided Projection of Strategies In Silviculture

A software platform to integrate forestry models
v3.1 - December 2017

François de Coligny – Nicolas Beudez
INRA - UMR AMAP
botAny and Modelling of Plant Architecture and vegetation
The training course is made of several parts:

- **Introduction:** the Capsis project

- **The Capsis framework:**
  - kernel
  - modules
  - simulation history
  - project manager
  - a guided tour
  - scripting

- **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 • scenarios
- **intervention** tools (e.g. thinning) •
- 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**
A 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](http://capsis.cirad.fr/capsis/charter)
How does it work

Create a Project:
- a growth module
- a set of initial parameters
- an initial scene
- > a root step in the project

Build silvicultural scenarios by alternating:

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

**interventions**
- > delegated to an extension
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)
The Capsis framework
Software diagram

The evolution models are implemented in 'modules'
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 initial scene
- implements the main evolution loop
The **central component** in the Capsis kernel, often invisible

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

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

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IdCard</strong></td>
<td>model information</td>
</tr>
<tr>
<td><strong>InitialParameters</strong></td>
<td>the superclass of all model parameters set at simulation start time</td>
</tr>
<tr>
<td><strong>GModel</strong></td>
<td>the superclass of all model classes</td>
</tr>
<tr>
<td><strong>GScene</strong></td>
<td>the superclass of all scene classes</td>
</tr>
</tbody>
</table>

**capsis.kernel**

- **IdCard**
  - model information
  - name
  - author(s)
  - prefix...

- **InitialParameters**
  - the superclass of all model parameters
  - set at simulation start time

- **GModel**
  - model initialisation and evolution loop (in subclass)

- **GScene**
  - 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

The Capsis training - F. de Coligny - INRA AMAP - December 2017
Focus on module

e.g. model: training, prefix Tra

Main classes

- IdCard
- TraInitialParameters
- TraModel
- TraScene
Focus on module: IdCard

The identity card of the module:

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

```java
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 = "-";
```

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...
}
Focus on module: TraModel

```java
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...
}
```

The 'model class' (and the main class)
- created first when a model is chosen
- methods must be overridden
**Focus on module: TraScene**

The description of the scene

e.g. a list of trees (…)

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

```java
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 () {...}

    //...
}
```

The Capsis training - F. de Coligny - INRA AMAP - December 2017
Focus on module: TraEvolutionParameters

An extra class to set an evolution target

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

```java
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**

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

```java
public interface GProvider extends MethodProvider {
    /**
     * Compute the basal area for the given trees of the given stand.
     * @param stand the GScene
     * @param trees the Collection of GTree
     * @return the basal area
     */
    public double getG(GScene stand, Collection trees);
}
```

```java
public class TraMethodProvider implements GProvider {
    /**
     * Basal area
     * @param stand the GScene
     * @param trees the Collection of GTree
     * @return the basal area
     */
    public double getG(GScene stand, Collection trees) {
        // Loop on the tree, sum their basal areas
        return basalArea;
    }
}
```
The history classes

A way to memorize the simulation

**Session**
- has a name
- is unique, held 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

<table>
<thead>
<tr>
<th>Project PNN [pnn_Unnamed] - 5 ha - All in memory - virtual stand</th>
</tr>
</thead>
<tbody>
<tr>
<td>10a - 11a - 12a - 13a - 14a - 15a - 16a - 17a - 18a - 19a - 20a</td>
</tr>
<tr>
<td>13b - 14b - 15b - 16b - 17b - 18b - 19b - 20b</td>
</tr>
</tbody>
</table>
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 all steps:

- 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** (capsis framework or application specific commands)

step popup was opened on step 15b
An illustration

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

- 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

```
Project Training [tra] - 277.57 m2 - All in memory - data/template/template.inv
```

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
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;
    }
}

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

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

An illustration

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
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
Focus: `TraModel.processEvolution()`

```java
/** 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;
}
```

**Loop 10 times**
- Get the limit: 10
- Loop 10 times 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 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

Focus: Thinning extension

```java
public class TraThinner implements Intervener, GroupableIntervener {
    private boolean constructionCompleted = false; // if cancel in interactive mode, false
    private GScene scene; // Reference scene: will be altered by apply()
    private GModel model;
    protected Collection<Tree> concernedTrees;
    protected double selectionDbh = 20; // cm, cut no trees above this dbh

    /** 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;
    }
}
```

An intervention is a particular type of extension

- Get a full copy of the scene \( *_{10} \)
- Optionally 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
4. **Evolution again**: Right click on step *10 > Evolution > 10 years more > 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 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...

```plaintext
coligny@marvin-13:/workspace/capdis4S sh capdis.sh -p script maddmodule.myscripts.SimpleScript
Script: splashscreen was deactivated
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 Capsis modellers
Capsis comes with ABSOLUTELY NO WARRANTY
The core of the Capsis platform (packages capdis.*) 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 capdis.script.Pilot: correct boot at 20 Mar 2014 12:12:02 CET
Working dir: /home/coligny/workspace/capdis4S

Launching script maddmodule.myscripts.SimpleScript...
Memorizer capdis.extension.memorizer.DefaultMemorizer was correctly set for project P
project_capdis.app.C4Script
[0->200] 1.234 5678 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200]
coligny@marvin-13:/workspace/capdis4S
```
How to write a script

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

```java
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");
    }
}
```
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...
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
Rectangular or polygonal plot

AbstractPlot
scene
origin (m)
xSize (m)
ySize (m)
area (m²)
getVertices ()
getShape ()
addCell (), getCells ()...

DefaultPlot
RectangularPlot

TreeListCell
trees

SquareCell
iGrid, jGrid
getArea (), getShape ()
x, y, z

PolygonalPlot

capsis.defaulttype

LemoineStand
SamsaPlot
SamsaStand

VtxPlot
VtxStand

capsis.kernel

lemoine.model
samsara.model
ventoux.model

The Capsis training - F. de Coligny - INRA AMAP - December 2017
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)

```java
// 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.
A script can be built for each OS to launch the application Starter:

**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

```java
/**  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

```java
/** 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)

```java
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
common 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
util/  
scripts in java

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
...
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 (english) or the projects page for a quick overview. Have a look at the documentation page for more details.

Community news

- Priscilla Callié (FCBA BSA, Bordeaux) spent a week in AMAP to start the Economics module implementation in partnership with the ONF R&D division (see the projects page). (fc - 20.3.2014)

- Nicolas Donès (INRA PIAR, Clermont Ferrand) came to Montpellier in February for the OpenFluid training session and could work on a 2 days review on the Capsis-RReShar model by Philippe Balandier. (fc - 20.3.2014)

- Florian Dererue (UMR BioGeCo, Université de Bordeaux) came to AMAP for a whole week work session on his Woudyfor model. The Pine trees light effect has been added, the forestry management effects are now considered, the common gorse dynamics processes have been updated and reviewed. (fc - 29.1.2014)

- François de Coligny was invited on 17th October 2013 at the Faculty of Forest Science and Forest Ecology of the University of Goettingen (Germany) to present the Capsis project at the colloquium “Forest Biometrics and Computer Science”. Thierry Fourcaud was also invited to present AMAP’s tree architecture modelling works at this special occasion of the 70th birthday of Prof. Branislav Sloboda. (fc - 21.10.2013)

- Capsis and AMAPstudio have been presented at the IUFRO 5.01.04 – Wood quality modelling international conference MeMoWood, in Nancy 1-4 October 2013. The presentation is available here. (fc - 9.10.2013)

- François de Coligny was invited to present Capsis and AMAPstudio at the international summer school “Modelling of Ecosystems by Tools from Computer Sciences” co-organised at the Czech University of Life Sciences in Prague (Czech Republic) on 16-20 September 2013 by Profs. Winfried Kurth and Branislav Sloboda (University of Goettingen, Germany) and Prof. Marek Fabrika (Technical University of Zvolen, Slovakia), and coupled with the 6th meeting of the developers and users of GroIMP. (fc - 23.9.2013)
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)
- Capsis quick tutorial: create a ModisPinaster project, build silvicultural scenarios, check the results in the integrated charts
- Capsis quick tutorial: create a Mountain project, build interactively a silvicultural scenario, compare various interventions
- FAQ (frequently asked question)
- The Capsis Manual
- Help Index

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
- Code Editor (Notepad++ / Textpad / Softe / Eclipse)
- Amap-dev : development web site
- SVN update and commit
The Project management tool

http://amap-dev.cirad.fr/projects/capsis
The Project versioning tool

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Revision</th>
<th>Age</th>
<th>Author</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>QS1</td>
<td></td>
<td>5099</td>
<td>over 2 yrs</td>
<td>François de Coligny</td>
<td>Freca.Plocabies now grows, renamed GTreeDbh and...</td>
</tr>
<tr>
<td>abcedrus</td>
<td></td>
<td>6583</td>
<td>over 1 yr</td>
<td>François de Coligny</td>
<td>Upgraded classes for the new icons</td>
</tr>
<tr>
<td>afoce/pa</td>
<td></td>
<td>5635</td>
<td>over 2 yrs</td>
<td>François de Coligny</td>
<td>Reorganized classes in capsis.commongui.command...</td>
</tr>
<tr>
<td>afoce/cpp</td>
<td></td>
<td>5635</td>
<td>over 2 yrs</td>
<td>François de Coligny</td>
<td>Reorganized classes in capsis.commongui.command...</td>
</tr>
<tr>
<td>alisier</td>
<td></td>
<td>6877</td>
<td>about 1yr</td>
<td>François de Coligny</td>
<td>Moved Numbearable from capsis.util to capsis.def...</td>
</tr>
<tr>
<td>amapsim</td>
<td></td>
<td>6877</td>
<td>about 1yr</td>
<td>François de Coligny</td>
<td>Moved Numbearable from capsis.util to capsis.def...</td>
</tr>
<tr>
<td>artams</td>
<td></td>
<td>8138</td>
<td>17 days</td>
<td>Mathieu Forth</td>
<td>Implementation of ScenarioRecorder module tool, etc.</td>
</tr>
<tr>
<td>artemis2014</td>
<td></td>
<td>8174</td>
<td>7 days</td>
<td>Denis Hache</td>
<td>change the hd association</td>
</tr>
<tr>
<td>bicasos</td>
<td></td>
<td>8073</td>
<td>about 1m</td>
<td>Jacques Labonne</td>
<td></td>
</tr>
<tr>
<td>btimodal</td>
<td></td>
<td>5635</td>
<td>over 2 yrs</td>
<td>François de Coligny</td>
<td>Reorganized classes in capsis.commongui.command...</td>
</tr>
<tr>
<td>ca1</td>
<td></td>
<td>7187</td>
<td>11 months</td>
<td>Philippe Dreyfus</td>
<td>comment corrected</td>
</tr>
<tr>
<td>gui</td>
<td></td>
<td>5635</td>
<td>over 2 yrs</td>
<td>François de Coligny</td>
<td>Reorganized classes in capsis.commongui.command...</td>
</tr>
<tr>
<td>model</td>
<td></td>
<td>7187</td>
<td>11 months</td>
<td>Philippe Dreyfus</td>
<td>comment corrected</td>
</tr>
<tr>
<td>CalLabels_en.properties</td>
<td>2.05 KB</td>
<td>378</td>
<td>about 5 yrs</td>
<td>durour</td>
<td>* Remove duplicated entry in properties * Fix s...</td>
</tr>
<tr>
<td>CalLabels_fr.properties</td>
<td>2.46 KB</td>
<td>378</td>
<td>about 5 yrs</td>
<td>durour</td>
<td>* Remove duplicated entry in properties * Fix s...</td>
</tr>
<tr>
<td>idcarts.properties</td>
<td>243 Bytes</td>
<td>974</td>
<td>almost 5 yrs</td>
<td>François de Coligny</td>
<td>Corrected an error in idcarts.properties and up...</td>
</tr>
<tr>
<td>capsis</td>
<td></td>
<td>8203</td>
<td>2 days</td>
<td>François de Coligny</td>
<td>RReShap fixed the crowns creation with new crow...</td>
</tr>
<tr>
<td>cassanaoany</td>
<td></td>
<td>8195</td>
<td>10 days</td>
<td>hendrik davi</td>
<td>add rootnode to inventory loader</td>
</tr>
<tr>
<td>cebas</td>
<td></td>
<td>3182</td>
<td>almost 4y</td>
<td>François de Coligny</td>
<td>Refactored commongui to remove dependencies and...</td>
</tr>
<tr>
<td>croreplant</td>
<td></td>
<td>8173</td>
<td>7 days</td>
<td>Denis Hache</td>
<td>error handling improvement</td>
</tr>
<tr>
<td>ctyssus</td>
<td></td>
<td>5635</td>
<td>over 2 yrs</td>
<td>François de Coligny</td>
<td>Reorganized classes in capsis.commongui.command...</td>
</tr>
<tr>
<td>dreeses</td>
<td></td>
<td>8038</td>
<td>about 1m</td>
<td>Patrick Vallet</td>
<td>How do we say &quot;coquille insigniaume&quot; in Engli...</td>
</tr>
<tr>
<td>dynaclim</td>
<td></td>
<td>7715</td>
<td>5 months</td>
<td>hendrik davi</td>
<td>new wood init, change in castaneaonly and PDG, ...</td>
</tr>
<tr>
<td>dynet</td>
<td></td>
<td>6583</td>
<td>over 1 yr</td>
<td>François de Coligny</td>
<td>Upgraded classes for the new icons</td>
</tr>
<tr>
<td>economics</td>
<td></td>
<td>8178</td>
<td>6 days</td>
<td>François de Coligny</td>
<td>Added translations</td>
</tr>
<tr>
<td>eucalypt</td>
<td></td>
<td>6877</td>
<td>about 1 yr</td>
<td>Patrick Vallet</td>
<td>initialization bug fixes for Simmen</td>
</tr>
<tr>
<td>fasy</td>
<td></td>
<td>7425</td>
<td>9 months</td>
<td>François de Coligny</td>
<td>Worked on Heteroflor growth with M. Jonard</td>
</tr>
<tr>
<td>fbca</td>
<td></td>
<td>7813</td>
<td>4 months</td>
<td>François de Coligny</td>
<td>Moved Numbearable from capsis.util to capsis.def...</td>
</tr>
<tr>
<td>fiesta</td>
<td></td>
<td>6877</td>
<td>about 1yr</td>
<td>François de Coligny</td>
<td>Moved Numbearable from capsis.util to capsis.def...</td>
</tr>
</tbody>
</table>
The Capsis javadoc

http://amap-dev.cirad.fr/projects/capsis
The reference publication

Including a list of 50 integrated models (2012)
### Capsis projects

**Last update: March 2014**

This page contains the list of active projects in the Capsis platform. You can find here about 60 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 archive 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.

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
<th>Contact, starting date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economics</td>
<td>The Economics Module reads a dendrometry files and a cost/income file and creates an economic balance. Keywords: Economics, Ploca abies, Pseudosuga menziesii, Pinus pinaster, Volume, Crown volume</td>
<td>Priscilla Caillie, (FCBA BSA, Bordeaux), Hanitra Rakotoarison (ONF R&amp;D) March 2014</td>
</tr>
<tr>
<td>Pinus pinaster</td>
<td>A Distance-Independent Tree Model for Maritime Pine (Pinus Pinaster). An individual-based rewriting of PP3 (see below), a basis for integration of spatialized processes. Keywords: Pinus pinaster, Growth, Temperate. Stand level, Tree level, Pure, Regular</td>
<td>Céline Meredieu, Thierry Labbé (INRA BIOGEOC, Bordeaux) December 2013</td>
</tr>
<tr>
<td>Deessses</td>
<td>Deessses, which stands for “DEux ESSEnceS” is a module that groups several 2 species models, developed with National Forest inventory data. These models allow evaluating the mixture effect on basal area growth according to environmental variables. Deessses includes a previous model, Melies, developed for mixed Norway spruce – silver fir mixtures. It also contains equations sets for mixed Norway spruce – common beech and for mixed common beech – silver fir stands.</td>
<td>Patrick Vallet, Maud Toigo, Thomas Pérot (Iresta Negent sur Vermisson) October 2013</td>
</tr>
<tr>
<td>Runaway</td>
<td>This module investigates the evolution of correlation between traits under sexual selection in a theoretical population of organisms with a special attention on genome dynamics.</td>
<td>Jacques Labonne (UMR Ecobiop, Saint-Pée sur Nivelle)</td>
</tr>
</tbody>
</table>
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 unique 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