An overview of software platforms and tools for model integration, collaborative development and resource sharing

S. Dufour

UMR AMAP

23/10/2009
1. Software platform challenges
2. Modeling approaches
3. Tools to co-develop
4. Community
5. Conclusion
<table>
<thead>
<tr>
<th>Software platform challenges</th>
<th>Modeling approaches</th>
<th>Tools to co-develop</th>
<th>Community</th>
<th>Conclusion</th>
</tr>
</thead>
</table>

Software platform challenges
Pluri-disciplinary approach

Plant/Tree modeling is based on different domains
⇒ Need of collaboration

- Forestry
- Visualization
- Simulation
- Computer sciences
- Biology
- Biophysics
- Mathematics
- Measures
- Modelisation
- Analysis
Modeling strategy

1. Construct the best model (efficient & simple) for each new situation

2. A general unified model

3. Defining common phenomenon, concepts and methods:
   - Common to different situations
   - Extensible
Development method

**Experimentation vs Production**
- Research model $\neq$ Industrial products
  $\Rightarrow$ models should be usable and modifiable at the same time
- Adapt development methodology

**Limited development resources**
- The same tools/libraries are often re-develop
- **Do not reinvent the wheel**, but reuse existing tools
- Exemple : the 3D viewer
Platform challenges

Goals

- Rapid and **easy development** of model
- **Reuse** softwares & tools
- **Share** development and exchange experiences
- **Synergy** in the community
- Ensure **software quality** (portability, installation, support...)
- Improve **diffusion** and valorisation

Expected results

- Perenity
- “Standardisation”
- Cost reduction
Modeling approaches
Type of platform

Platform = an approach + organisation

**Different approach**
- Generic model
- Data structure
- Methodology
- Software libraries
- Component integration
- Simulation formalism

**Different organisation**
- Centralized vs Distributed development
- One developer or modeler vs a Community
- Open vs Closed license
Around a model

Model : Sylva
Theme : Forestry

- Forest growth simulator
- Multi-species : *Picea abies*, *Abies alba*, *Pinus sylvestris*, *Fagus sylvatica*, *Sessile Oak*, *Quercus petraea*, *Alnus glutinosa*
- Flexible : simulate diverse intermingling and structured stands, as well as tending regimes and regeneration methods.

**Platform** : AmapMod  
**Data Structure** : MTG (Multi Scale Tree graph)  
**Theme** : Plant architecture analysis  
- Query Language (AML) + Statistic tools + Analysis tools  

Methodology

**Platform**: Capsis

**Theme**: Forestry growth

- **Multi-Model**: stand level, individual, spatialized, ...
- **Software architecture**: kernel, models, plugins, ...
- Scenario and intervention
- Common **charter**, common methodology
Set of libraries

**Platform:** Jeeb  
**Theme:** plant architecture modeling

- A set of **shared libraries** and application
- **Common tools**, data structure, algo (ligth, sketching...)
- Plugin architecture, shared code
- Dedicated applications (Xplo, Simeo...)
Integration and composition

**Platform**: OpenAlea  **Theme**: Multi-scale plant modeling

- **Python** as a glue Language (multi-language integration)
- Package based & **hierarchical component architecture**
- **Dataflow** and visual programming
- Development and deployment tools
Formalism

**Platform** : VLE/Record  **Theme** : cultural system

- Discret event system specification (**DEVS**)
- **Complex system** : Dynamic system / dynamic structure
- **Modular & hierachical** model construction
- Event based + state + transition function
Tools to co-develop

...
Share information

- Bugs
- Features
- Roadmap
- Technical documentation
- Activity

![Project management (Forge)](image-url)
Sharing code

Versioning system

- All developer are **synchronised**
- All changes are **recorded**
- ⇒ Improve traceability, error recovery, bug finding...
- Exemple : SVN, Mercurial, Git, Bazaar...
Build / Deployment

Developer task should be automated and **standardised**
- Build complex and large projects (ant, cmake, scons...)
- Manage dependencies between subprojects (maven, ivy, ...)
- Portability

Deployment, diffusion and update
- Installer creation
- Web diffusion
- Package Manager (ex: apt)
Software quality: Providing tests

**Functional Testing**
- Outputs are correct for a given input
- Tests are automated ⇒ test reports
- Refactoring & improvements are possible

**Continuous integration**
- Model are tested continuously on a server
- Ensure validity after modifications
- Detect indirect errors (integration problem)
Sharing information

Documentation / wiki

- Documentation is **dynamic**
- Documentation is **up to date**
- Everyone can add/correct pages
- Build technical documentation (javadoc, sphinx...)
Roles

Standard model

- **Developer**: create & diffuse the software
- **User**: install & use the software

Scientific platform model (cf Capsis)

- **Developer**: develop platform kernel and shared libraries
- **Modeler**: develop models, reused shared tools
- **User**: use the model with its own data
Charter (Capsis)

- **Free kernel** (free license)
- **Modelers are responsible** of their model and their validation
- They can have **support from developers**: lib, training, tools...
- Models and source code are **accessible** in the modeler community
- Respect of **intellectual property**: License / Copyright
- Distribution & **Decentralisation**: modeler can distribute their own models and provide support on them
- ...
Methodology: Agile development

The Agile Manifesto [Fowler, Highsmith - 2001]

Principles

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Collaboration** over contract negotiation
- **Responding to change** over following a plan

In practice

- Small iteration & Immediate feedback
- Coding / Modeling meetings
- Good practice (code review...)
- **Keep it simple!**
License / Open Source

- Published models are accessible ⇒ verification
- Code is perennial, models & lib can be reused and improved...
- **Open source is not public domain (Copyright)**

**Strategy**

- Non viral Free license for core modules and shared libs (LGPL like)
- Open or commercial or dual licensing for models
Conclusion

A software platform?

- **Common framework** / paradigm / data structure...
- **Shared** tools and libraries (mutualisation & reuse)
- Developer / Modeler / User **community**
- Common **rules** (charter, license...)

![LEGO blocks](image)