An extension of the graph-grammar based simulator GroIMP for visual specification of plant models using components

Michael Henke, Katarína Smoleňová, Yongzhi Ong, Winfried Kurth

Department Ecoinformatics, Biometrics and Forest Growth, University of Göttingen, Germany

Introduction

The most frequently used formalism to specify functional-structural plant models (FSPMs) are L-systems. A powerful generalization are parallel graph grammars. The programming language XL combines these with the power of the general-purpose, object-oriented language Java and gives thus support to the combination of process-based and rule-based submodels (Kniemeyer et al. 2008). However, when applied to complex tasks, models specified in XL like those in other languages still tend to grow in an intransparent manner. Methods which are commonly used in software engineering are not yet supported (this being a general problem for scientific software, see Wilson 2006). As a contribution for improvement, we are currently implementing some extensions of XL, supported by the platform GroIMP, which will facilitate a component-based modelling.

Aim

- Goals:
  - Low entry threshold: make access simpler
  - Reuse of software (parts)
    - Independently developed
    - Development by experts
    - Periodic maintenance
  - Reduction of development effort and error-proneness
  - Transparent and flexible modelling process and models
- Side effects:
  - Reduction of time for developer to get familiar with a system / language
  - Models that can be evaluated / combined
  - Comparability of models
  - Enhanced quality
  - Faster model development
  - Facilitation of communication between modeller and experimenter

Materials and Methods

- Modelling language and platform
  - Written in Java and XL as extension of modelling platform GroIMP
- Combination of two approaches: Component-based software engineering & Visual programming

The composition of a model in terms of nested and interconnected components is represented by a directed graph $(C \cup S, E)$. Its nodes stand for the components C and their slots S. Its edges have types with different semantics (Tab. 1).

Fig. 1 shows a simple example of a component structure. On the left-hand side, we see the classical graph view with the components as labelled circles, the slots as triangles and all edges as arrows. Edges marked with "/" stand for the "containment" relationship. The dotted edge sends the output of the model (the result of some calculation in "structure") to the exterior output slot at the coarsest level, it is thus a "send" connector. The right-hand side shows the same component structure in a hierarchical, nested view. The "plant" component (at the coarsest level) contains (visually) the three other components. Only edges between slots are shown as arrows here.

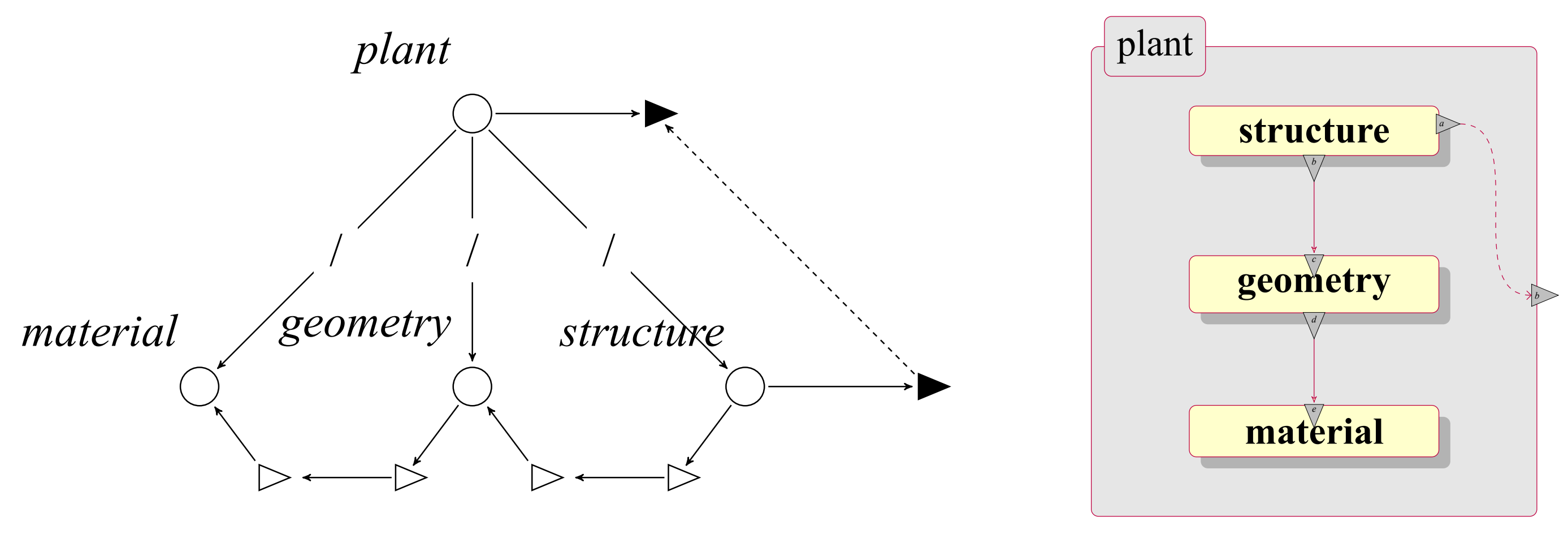


Fig. 1 A graph representing a component structure of a plant model, shown in two alternative views. Left: classical graph view. Right: hierarchical view, "containment" relationships are visualized by geometric inclusion of rectangles.

GroIMP's basic data structure is an attributed graph, with the nodes generally standing for objects (e.g., plant organs) and the edges for relationships. This graph can undergo transformations, specified in XL. The GroIMP graph offers itself in a natural way to represent the component structure of a model, expressed by the above-described graph, as well. Each node from C is then associated with a piece of code which represents the functionality of the corresponding component. Currently, all components have to be encoded in XL or Java and have to be accessible for the integrated XL compiler of GroIMP. Frequently used components can be kept permanently in an inherent library of GroIMP and are then accessible by a browser.

Tab. 1 Current types of edges:

Name	Symbol	Description
Refinement	/	"containment" relationship, only between components
Signal	s	unidirectional signal, can trigger an event or transport data
Call/receive		a method call or a request for a service

Results

- Component-based framework as extension of GroIMP software (Fig. 3)
- Visual modelling support (2D editor)
- Component developer environment
- Library of predefined components (Fig. 2)
- User manual (including background details, framework description, examples, and more)

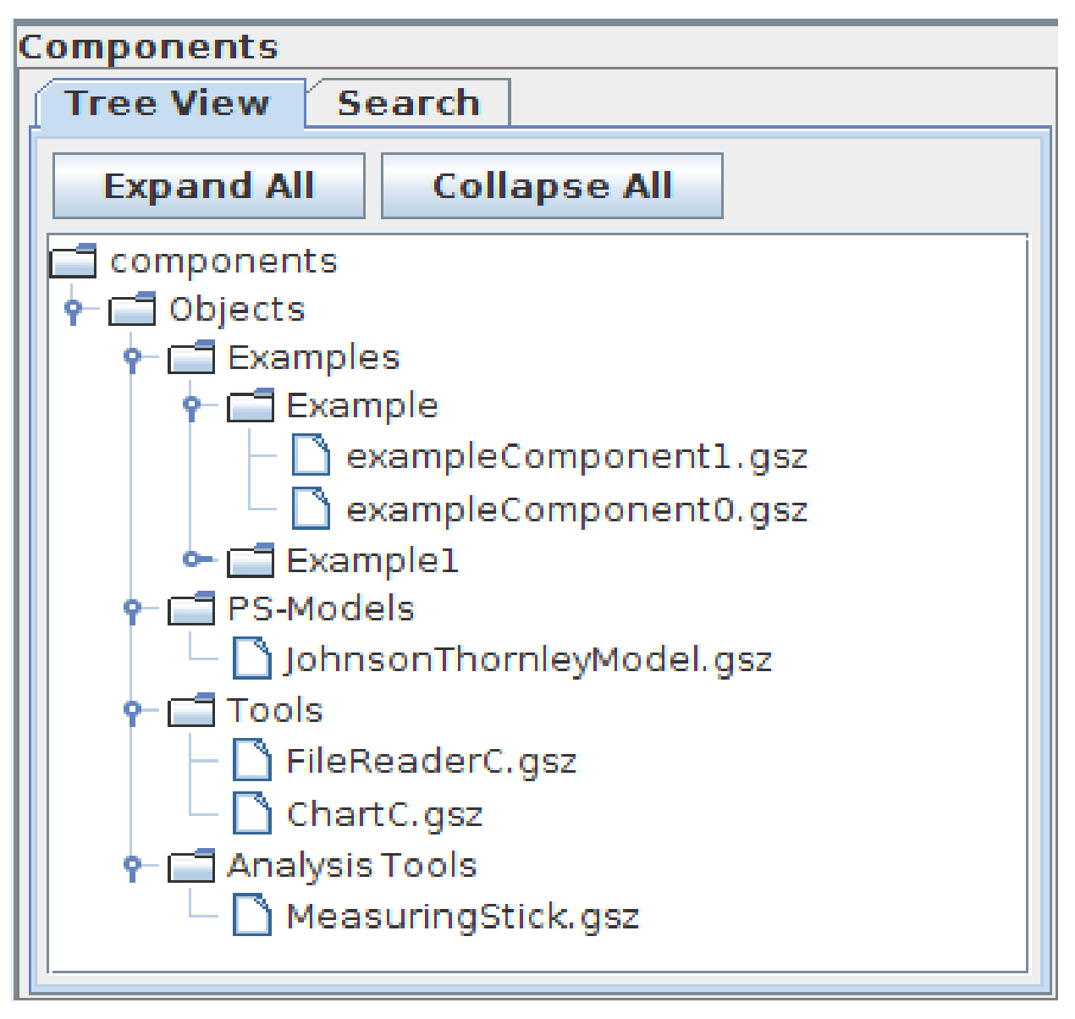


Fig. 2 Hierarchical Component Explorer of the GroIMP software; additionally a full text search mechanism will help the user to find quickly what he is searching for.

The component editor offers two alternative views, corresponding to the two graph representations in Fig. 1. Furthermore, the XL code of each component can be directly accessed in a code window. With this, we deliberately violate the "black-box principle" of strict component-oriented development, thus answering to the demands of modellers from the FSPM community who have expressed their wish to have full insight and control of the software components they are expected to use. An additional developer environment with a template component as a base allows such users to implement their own components quickly.

Components can be used in XL to realize aspect-oriented programming in the sense of Cieslak et al. (2011). Aspects are the result of decomposing an organism into sub-entities which represent functionalities rather than spatial units. In the example of Fig. 1, topological structures or rules reside in "structure", geometrical ones in "geometry", and optical properties are confined to "material".

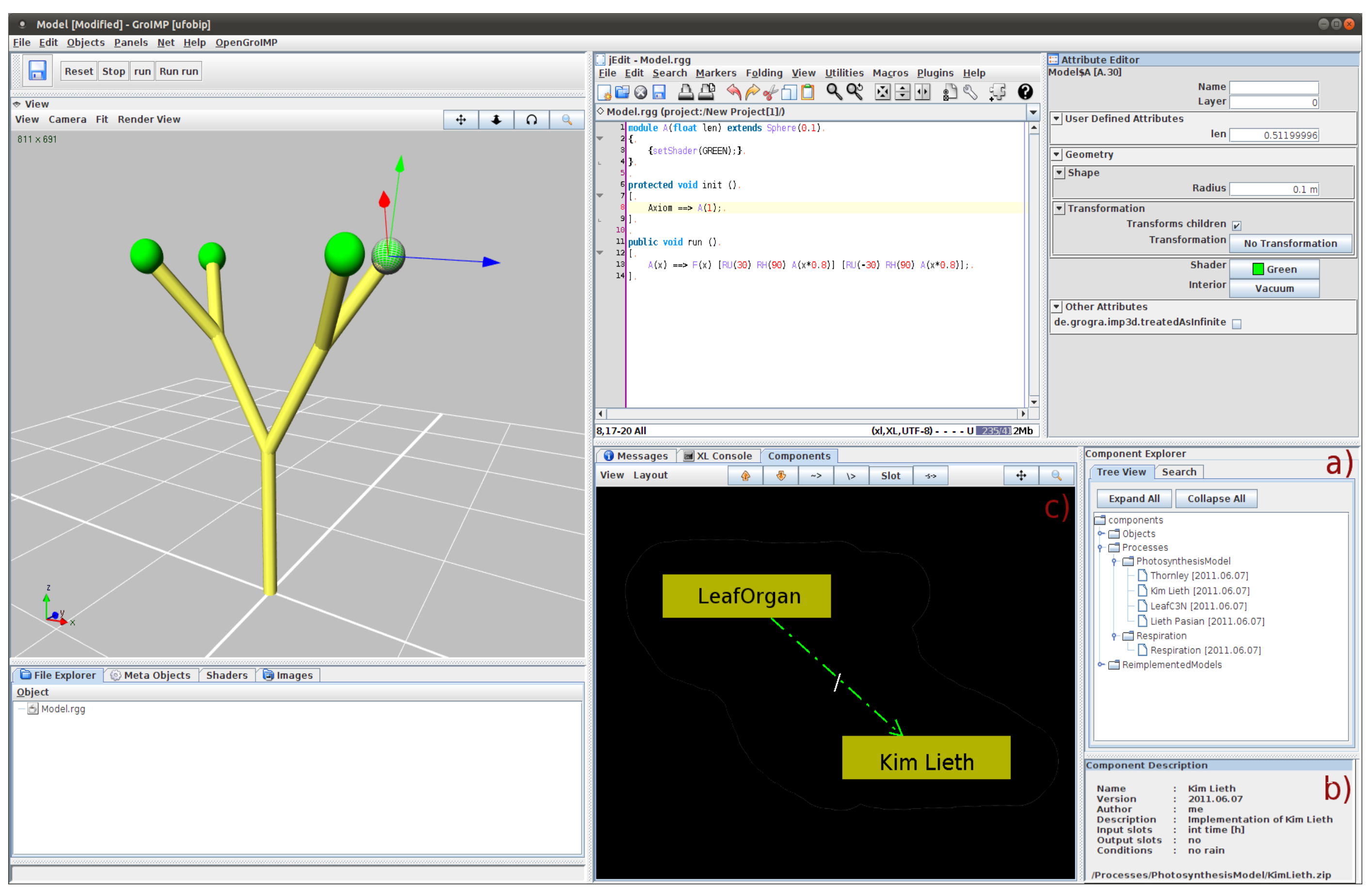


Fig. 3 Screenshot of the GroIMP software with the new introduced panels: a) Component Explorer, b) Component Description, and c) Component View

- 2D editor is used to assemble predefined components to the final model
- Modeller does not need to care about internal implementation details

  → relieves the modeller form the burden of low level programming work

  → facilitates to focus on modelling

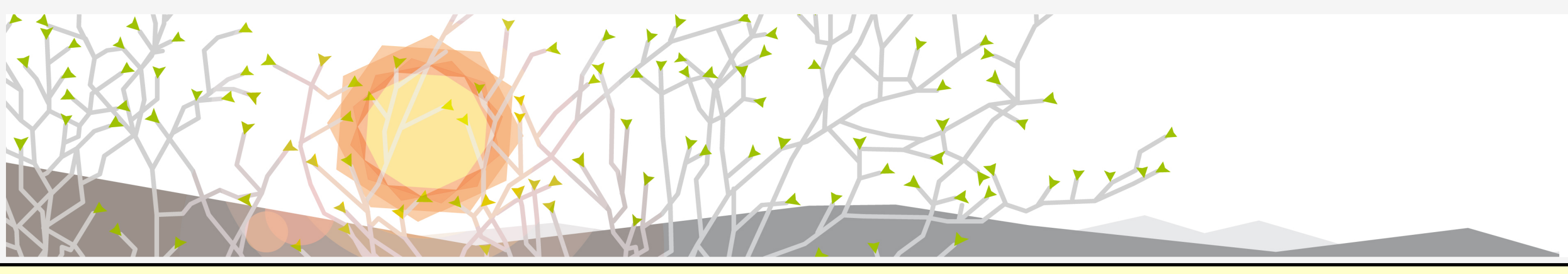
Literature

- **Cieslak M, Seleznyova AN, Prusinkiewicz P, Hanan J.** 2011. Towards aspect-oriented functional-structural plant modelling. Annals of Botany **108**(6):1025-1041.
- **Kniemeyer O, Barczik G, Hemmerling R, Kurth W.** 2008. Relational Growth Grammars -- a parallel graph transformation approach with applications in biology and architecture. In: Applications of Graph Transformations with Industrial Relevance AGTIVE '07 (eds.: A. Schürr, M. Nagl, A. Zündorf), LNCS 5088, Springer, Berlin etc., 152-167.
- **Wilson G.** 2006. Where's the real bottleneck in scientific computing? American Scientist **94**(1):5.

Acknowledgement & Contact

German Research Foundation (DFG) under project identifier KU 847/8-1. All assistances are gratefully acknowledged.

- GroIMP 1.4.2 (free, open source software): `http://www.grogra.de`
- Department: `http://www.uni-goettingen.de/en/67072.html`
- E-Mail: info@grogra.de



FSPM 2013 7th International Conference on FUNCTIONAL–STRUCTURAL PLANT MODELS 9–14th June 2013, Saariselkä, Finland