Rapidely Generating Different Meshing Tools

María Cecilia Bastarrica, Nancy Hitschfeld-Kahler, Pedro O. Rossel, César Castro
CS Department, Universidad de Chile, {cecilia,nancy,prossel,ccastro}@dcc.uchile.cl

Developing meshing tools is hard and complex. However there are several meshing tools that have been developed over the years differing in their modeling elements, the algorithms applied, the data structures used for implementing the meshes, the application domains, etc. [2]. Most of them are different but they still share similarities: generate an initial mesh from a geometry description, refine and/or improve the mesh, evaluate it, and visualize it, among others. Developing each of these features is hard, and defining the right combination is hard too. We propose to apply Model Driven Engineering [1] to encapsulate and reuse already developed software components, reducing the effort of building meshing tools to the process of choosing the features that will be included. This is achieved through a user interface that guides the configuration process, a data base (DB) of meshing tool software components, an underlying set of consistency rules that determines when a configuration is reasonable, and a generic structure that is configured with the chosen components in order to generate the tool.

Components. We extracted meshing tool code components from tools we have developed and also from open source tools, and we stored and classified them in a DB so that they can be found. For example, we have used Flexmg as a source of $2\frac{1}{2}$D meshing data structures and algorithms. Thus we obtained a *Mesh* class that represents a surface mesh and several components representing algorithms that interact with the mesh such as *Refine*, *Evaluate* and *GenerateMesh*. We also store particular implementations for these classes such as *LeppRefine* and *LongestEdgeBisection* for the *Refine* class. This component DB can be incrementally grown by incorporating new particular implementations. In this way the potential meshing tools that may be built also grows.

Structure and Constraints Once the features we want the meshing tool to exhibit, they are combined using a product line architecture (PLA), i.e. the architecture that all meshing tools share. The PLA design is embedded in the configuration tool and the interfaces of all components must conform to this PLA. Not any component combination is possible. For example, if we choose to have a *Refine* algorithm, we should choose a particular implementation for it. Also if we have a 3D representation of the mesh, the chosen algorithms for managing it must be also designed for 3D meshes. All these constraints are defined in a Feature Model [1] and the configuration tool uses it for guiding the decisions the user must make while designing the meshing tool.

References

[1] K. Czarnecki and U. W. Eisenecker. *Generative Programming. Methods, Tools, and Applications*. Addison Wesley, May 2000.

[2] S. J. Owen. http://www.andrew.cmu.edu/user/sowen/mesh.html. Accessed on January 2009.