ARCHITECTING A FAMILY OF MESHING TOOLS

MARÍA CECILIA BASTARRICA, NANCY HITSCHFELD-KAHLER, AND PEDRO O. ROSSEL

Abstract. Meshes are used for numerical modeling, visualizing and/or simulating objects or phenomena. A mesh is a discretization of a certain domain geometry that can be either composed by a unique type of element or a combination of different types of elements. Meshing tools generate and manage these discretizations.

Meshing tools are inherently sophisticated software due to the complexity of the concepts involved, the large number of interacting elements they manage, and the application domains where they are used. They need to accomplish specific functionality while still having an acceptable performance. Managing thousands and even millions of elements with a reasonable use of computational resources becomes a must. Lately, however, other qualities like modifiability have also become relevant.

There are many application domains where meshing tools are used [DCW+02]. For this reason, a variety of meshing tools have been built differing on their functionality, the algorithms used, the data representation, or the input and output format. Also depending on the application domain, it may be required to have one, two or three dimensional meshes, each one maybe using different types of basic modeling elements.

Developing any complex software from scratch is expensive, slow and error prone, but this is the way meshing tools have traditionally been built. If this development is not performed using good software engineering practices, it may easily get out of control making it almost impossible to debug and even more difficult to modify. There have been some efforts lately applying software engineering concepts in meshing tool development, mainly building general purpose libraries that facilitate reuse. Also object-orientation and design patterns have been used for enhancing software reuse at the code and design levels.

The software architecture is one of the main artifacts developed during the software life cycle [Fow03] because it determines most of the non-functional characteristics the resulting software will have. Architectural patterns [BMRS96] are used as guidelines for architectural design because they promote different non-functional characteristics.

Software product lines is a trend for planned massive reuse [CN01]. The most typical reusable assets are software components, but we can also reuse other assets as the product line architecture (PLA). The PLA is an important reusable asset because all software products in the family will share it [Bos00].

We present the product line architecture for a family of meshing tools. We intended to provide a PLA that would promote flexibility and extensibility, so that existing algorithms, data structures, data formats and visualizers could be combined in different ways to produce a variety of meshing tools appropriate for diverse application domains, sharing the same software structure. The PLA is modeled following the layered architectural pattern that promotes modifiability, reusability and portability. Sometimes it is argued that layered architectures may penalize performance, but we have found that performance does not necessarily degrade significantly using the proposed PLA [Lil06]. In [HKLC+06] it is reported that a tool implementing this layered architecture performs almost as fast as TetGen [SG05].

We show how the proposed PLA can be instantiated for generating different meshing tools. In particular we show how already implemented tools can be seen as instantiations of our product family, independently of the methods followed for generating the meshes and the dimensions of the managed mesh.

References

[BMRS96] F. Buschmann, R. Meunier, H. Rohnert, and P. Sommerlad. *Pattern Oriented Software Architecture: A System of Patterns*. John Wiley & Son Ltd., 1996.

[Bos00] J. Bosch. *Design and Use of Software Architectures. Adopting and Evolving a Product Line Approach.* Addison Wesley, 2000.

[CN01] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns.* Addison Wesley, 2001.

[DCW+02] R. W. Douglass, G. F. Carey, D. R. White, G. A. Hansen, Y. Kallinderis, and N. P. Weatherill. Current views on grid generation: summaries of a panel discussion. *Numerical Heat Transfer, Part B: Fundamentals*, 41:211–237, March 2002.

[Fow03] M. Fowler. Who Needs an Architect? *IEEE Software*, 20(5):11–13, 2003.

[HKLC+06] N. Hitschfeld-Kalher, C. Lillo, A. Cáceres, M. C. Bastarrica, and M. C. Rivara. Building a 3D Meshing Framework Using Good Software Engineering Practices. In *Proc. of the 1^{st} Int. Workshop on Advanced Software Engineering*, Chile, August 2006.

[Lil06] Carlos Lillo. Analysis, Design and Implementation of an Object-Oriented System that allows to Build, Improve, Refine and Visualize 3D Objects. Master's thesis, CS Dept, Univ. de Chile, 2006.

[SG05] H. Si and K. Gärtner. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In *Proc. of the 14^{th} International Meshing Roundtable*, 2005.

DCC, Universidad de Chile
E-mail address: cecilia@dcc.uchile.cl

DCC, Universidad de Chile
E-mail address: nancy@dcc.uchile.cl

DCC, Universidad de Chile; DCI, Universidad Católica del Maule
E-mail address: prossel@dcc.uchile.cl