Department of Computing and Software

Faculty of Engineering — McMaster University

Commonality Analysis for Mesh Generating Systems

by

S. Smith and C. H. Chen

C.A.S. Report Series **CAS-04-10-SS**

Department of Computing and Software **October 2004**
Information Technology Building
McMaster University
1280 Main Street West Hamilton, Ontario, Canada L8S 4K1

Commonality Analysis for Mesh Generating Systems

Spencer Smith[*], Chien-Hsien Chen

Technical Report CAS-04-10-SS
Department of Computing and Software
McMaster University

October 25, 2004

Abstract

This report presents a commonality analysis for mesh generating systems with the intention that the commonality analysis can be used to facilitate development of meshing software as a program family. The document reviews both the methodology of commonality analysis and the details of meshing systems. The report also reviews arguments in favour of development of mesh generators as a program family. The commonality analysis itself consists of the following: *i*) terminology and definitions; *ii*) commonalities, or features that are common to all potential family members; *iii*) variabilities, or features and characteristics that may vary among family members; and, *iv*) parameters of variation, or the potential values that can be assigned to the variabilities. The documentation of the above items for meshing software is clarified by decomposing each item into subsections on mesh generation, input, output, nonfunctional requirements, and, where appropriate, system constraints.

Keywords: Commonality analysis, mesh generation, program family.

[*]Computing and Software Department, McMaster University, smiths@mcmaster.ca

Contents

1 Introduction

Mesh generating systems are well suited to development as a program family because they fit Parnas's definition of a program family: "a set of programs whose common properties are so extensive that it is advantageous to study the common properties of the programs before analyzing individual members" (Parnas, 1976). Developing mesh generating systems as a program family is advantageous because mesh generators share many common features, or commonalities. Furthermore, when there are differences between systems, the variabilities between them can be systematically considered. The purpose of this document is to record these commonalities and variabilities and show the relationships between them, and thus facilitate the specification and development of mesh generator program family members. This document will be valuable in all future phases of system development and analysis. For instance, the requirements documentation for any mesh generator will use the commonality analysis, since the requirements should refine the commonalities, which are shared requirements of all mesh generators. Moreover, the design of any future system will use this documentation to facilitate consideration of the variabilities, so that likely changes can be made to the system with a minimal amount of work.

The scope of the commonality analysis presented here can be considered both from the point of view of mesh generating systems and from the point of view of software engineering methodologies. From the mesh generator perspective, the starting point for the current document is the commonality analysis conducted by Chen (2003). However, the scope of the current document is broader than that of Chen (2003), which was restricted to two-dimensional meshes and did not consider non-conformal, hybrid or mixed meshes. The current commonality analysis is intended to cover all mesh generators that are targeted at finite element applications. Meshes for cartography or other uses are not explicitly considered, although much of the information in this document does overlap with other mesh uses. With respect to software engineering methodologies, the scope of the current report is restricted to informal methods, with the intention that the informal requirements will form a starting point for later development and refinement by formal methods.

The first section below provides an overview of the program family of mesh generators, by reviewing what is involved in a commonality analysis and the basics of mesh generating systems. After this, the basic terminology and definitions necessary for understanding the remainder of the document are provided. The definitions include terms used in describing a commonality analysis and terms that are used in defining the characteristics and properties of meshes. The next three sections consist of the lists of commonalities, variabilities and parameters of variation, respectively. These three sections form the heart of the documentation and include an extensive set of cross-references to demonstrate the relationships between the different items. The final section provides information on unresolved issues.

2 Overview

This section provides both an overview of the process of commonality analysis and of mesh generation. The subsection on commonality analysis briefly introduces this topic, along with references that can be searched for further information. The subsection on mesh generators outlines the following: *i*) the basics of mesh generation, *ii*) the scope of the current analysis, *iii*) the overall philosophy that has been adopted for the commonality analysis, and *iv*) the arguments in favour of the development of mesh generators as a program family.

2.1 Commonality Analysis

In some situations it is advantageous to develop a collection of related software products as a program family. The idea is that if the software products are similar enough, then it should be possible to predict what the products have in common, what differs between them and then reuse the common aspects and thus support rapid development of the family members. The idea of program families was introduced by Dijkstra (1972) and later investigated by Parnas (Parnas, 1976, 1979). More recently, Weiss (Weiss, 1997, 1998; Ardis and Weiss, 1997) has considered the concept of a program family in the context of what he terms Family oriented Abstraction, Specification and Translation (FAST) (Cuka and Weiss, 1997; Weiss and Lai, 1999).

In the approach advocated by Weiss, the first step is a commonality analysis. This analysis consists of systematically identifying and documenting the commonalities that all program family members share, the variabilities between family members and the terminology used in describing the family. A commonality analysis provides a systematic way of gaining confidence that a family is worth building and of deciding what the family members will be. A commonality analysis document provides the following benefits (Weiss, 1997, 1998):

1. A starting point for the design of a domain specific languages (DSL): Once a DSL, or application modelling language, is developed the program family members can be rapidly generated by specifying a given family member using the language.

2. A basis for a common design for all family members: When the software engineers come to designing the individual family members, they can take advantage of the commonalities to reuse code. Moreover, the variabilities can be considered in the design so that they can be easily accommodated. One approach to the design may be to decompose the system into components that can each be customized by specification of values for its various parameters, where the parameters correspond to the parameters of variation identified in the commonality analysis document.

3. A historical reference: This document records the important issues concerning the scope and the nature of the family (as well as some unsolved issues) to facilitate the involvement of the participants in maintaining and evolving the family.

4. A basis for reengineering a domain: Existing projects may not have been developed using software engineering methodologies. The projects can be systematically reorganized and redesigned with the aid of a commonality analysis to unify the existing products.

5. A basic training reference for new software developers: This document provides the necessary basic information for a new team-member to understand the family.

The next section will show that the above uses of the commonality analysis document will be beneficial for the development of a program family of mesh generators. The commonality analysis will benefit all subsequent stages in the software development process. For instance, as mentioned in the introduction, the commonalities will act as requirements that will be the starting point for writing a software requirements specification. The commonalities will be refined into specific requirements by fixing the value of their associated variabilities. The change in the values of the variabilities then corresponds to the change from one program family member to another.

As commonalities and variabilities are requirements, they should express “What” functionalities and qualities the system should have, and not mention “How” these requirements are to be accomplished. That is, the commonalities and variabilities should not involve design decisions. The design decisions will be made after the requirements for a family member have been specified. The one exception to this is system constraints, which are requirements that explicitly make design decisions.

Besides the “What” versus “How” test, there are other tests that can be used to review commonalities and variabilities, as proposed by Weiss (1997). One such test is the “what is ruled out” test. This test determines if a commonality or variability actually makes a decision because if no alternatives are ruled out then no decision has really been made. Another test is the “negation” test. If the negation of a decision represents a position that someone might argue for, then the original decision is likely to be meaningful. For instance, the statement that “the software should be reliable” has a negation that no one would likely argue for and thus the statement does not represent a good characterization of a goal for the system.

In Weiss (1997) the stages of a commonality analysis are described in a systematic way. The stages include the following: prepare, plan, analyze, quantify and review. The stages are completed through the aid of a moderator and a series of meeting and preliminary documents and documentation reviews. Although the systematic approach advocated by Weiss has its advantages, for the case of writing a commonality analysis document for mesh generating systems it was decided that a less structured approach is feasible. Mesh generators are simpler than other software systems in the sense that they have fewer interactions with the environment. Also, the theory of mesh generation has a solid mathematical basis that can be used to remove some of the ambiguity that Weiss’s approach is aimed at reducing. Therefore, the approach adopted here is to revise the original commonality analysis document produced by Chen (2003) and then make the new document available to others for review. The new document will be maintained in a concurrent versioning system repository so that multiple authors can work on it, and more importantly, so that the documentation's revision history can be tracked and the documentation can be rolled back to an earlier version if necessary.

2.2 Mesh Generators

A mesh is a discretization of a geometric domain into small simple shapes, such as line segments in 1D, triangles or quadrilaterals in 2D, and tetrahedral or hexahedra in 3D. Meshes are employed in many application areas. For instance, in geography and cartography, meshes are used to give compact and precise representations of terrain data (Bern and Plassmann, 2000). In computer graphics, most objects are first reduced to meshes before being rendered to the screen. As previously mentioned the principal application of interest for the current study is the finite element method, where meshes are essential in the numerical solution of partial differential equation arising in physical simulation (Bern and Plassmann, 2000). Typically, the first step in a finite element analysis is the generation of a finite element mesh. The output of the mesh generation program becomes the input to a finite element program.

A simple 2D mesh of quadrilateral elements, which could be used for analysis of a solid mechanics problem, is illustrated in Figure 1. The files created by a mesh generator must describe the following: how the domain is decomposed into cells (or elements); the material properties; and the boundary conditions on the domain, with respect to applied tractions (*e.g.* τ_x), prescribed displacements (*e.g.* Δ_y) and fixity (*e.g.* roller versus pinned versus free). The mesh shown in Figure 1 is an example of a structured mesh for a rectangular domain. In many practical problems the domain does not have such a simple geometry and the cell topology is no longer regular, which means an unstructured mesh must be adopted. Figure 2 shows an example of a 3D unstructured mesh for a magnetron. (A magnetron is a diode-type electron tube that can be used to produce microwave energy.) The magnetron mesh, along with other examples, can be found at the following web-page: http://www.geuz.org/gmsh/gallery/

In the majority of cases, the tedious preparation and checking of a mesh are too demanding to do manually, especially when the model description contains several thousand or more elements. Therefore, automatic generation of meshes, using a mesh generator, is of obvious practical value for reducing the workload. As a result, the user will only need to concentrate on a few input parameters and rely on a mesh generator to produce a corresponding mesh. The occurrence of human errors can thus be greatly diminished (Zienkiewicz and Phillips, 1971). The quality of the mesh, in terms of the size, shape and placement of the elements, is critical for the success of any finite element analysis; therefore, careful thought should occur before one proceeds to the implementation of a mesh generating system.

The suitability of the concept of a program family for mesh generating systems is argued in Chen (2003) and Smith and Chen (2004) by showing that mesh generating systems meet the three hypotheses for a program family as proposed by Weiss (1997):

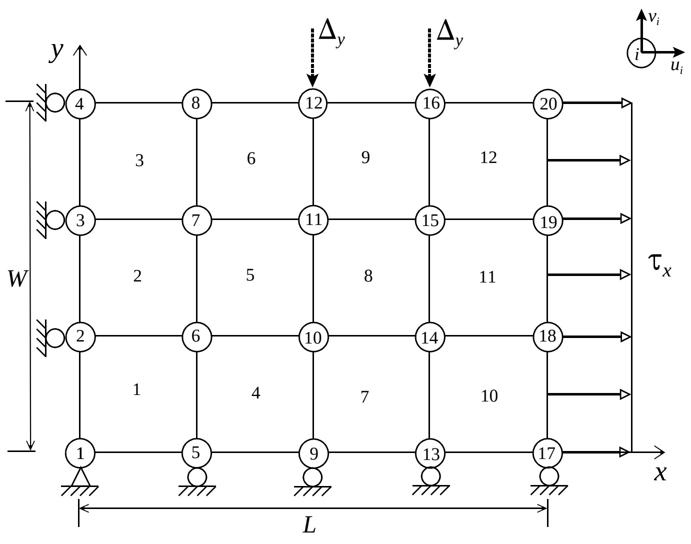

Figure 1: Example structured finite element mesh

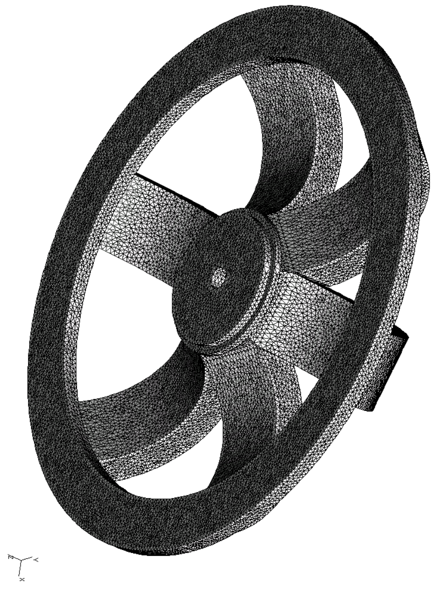

Figure 2: Example unstructured finite element mesh by P. Lefèvre

The Redevelopement Hypothesis: This hypothesis requires that most software development involved in producing the family should be redevelopement, which means that there should be a significant portion of common requirements, design, and code between the family members. For mesh generators there are examples of large general purpose codes that are constantly redeveloped over their lifetime, such as QMESH (Jones, December 1975) and CUBIT (Blacker et al., May 1994; Tautges et al., 1995) from Sandia National Laboratories. In the case of small mesh generator codes redevelopment is also common, as most small codes are based on modification of existing code. Although different in the specific details, all mesh generators can be abstracted as: input information then calculate a mesh discretization and finally output the results.

The Oracle Hypothesis: This hypothesis requires that the types of changes that are likely to occur during the system's lifetime are predictable. This is certainly the case for a mesh generators where one can determine likely changes by consulting the large body of literature and mathematical theories on the topic. Moreover, there are many example systems that show how the software can evolve over time.

The Organizational Hypothesis: According to the organizational hypothesis, the program to be developed using the program family approach should be one that allows designers and developers to organize the software, as well as the development effort, in a way that the predicted changes can be made independently. If this assumption holds, then a predicted change will require changing only a few modules in the system. This hypothesis, however, is challenging for mesh generating systems.

For some of the likely changes, such as the changes in the user interface, visualization, and output format, the changes can be dealt with in an elegant way. However, for other types of the changes, like the use of different mesh generating algorithms and the use of different optimization and smoothing algorithms, the goal of restricting the change within one module is difficult to meet because a mesh data structure is inevitably assumed by these algorithms. However, research on organizing the system so that predicted changes can be made independently has begun (Berti and Bader, 1998; Chen, 2003; ElSheikh et al., 2004).

The above discussion suggests that it is possible to use the program family strategy for mesh generators, but before investing effort into the commonality analysis, it is worthwhile to justify why the analysis is worthwhile. Five general advantages of a commonality analysis are mentioned in the previous section: a starting point for the design of a domain specific languages (DSL); a basis for a common design for all family members; a historical reference; a basis for reengineering a domain; and, a basic training reference for new software developers. In particular, the idea of reengineering mesh generating systems has appeal. It should be possible to learn from the many existing mesh generation systems to produce something that can reproduce the functionality of the existing systems, but have the advantage of being designed so as to avoid the pitfalls of previous attempts and to have the advantage of complete and unambiguous documentation.

In addition to sharing the general advantages that a commonality analysis brings to software projects, there are also advantages that may not apply to all systems, but which do apply in the special case of mesh generating systems. For instance, a commonality analysis would provide a convenient framework for summarizing the existing literature on mesh generation and for summarizing the existing systems. The existing work could be characterized by specifying the values for the parameters of variation that would produce this existing system as a program family member. Other attempts to classify existing meshing software, such as Teng and Wong (2000), do not follow as systematic an approach and focus a significant portion of the discussion on distinguishing meshing software based on implementation details. Classification of meshing software using the commonality analysis would also provide the benefit of testing the commonality analysis document. If an existing system is found that cannot be characterized by the commonality analysis, then the document can be fixed to accommodate the oversight.

Another advantage of commonality analysis that applies for mesh generating systems is the support that the analysis provides for the generation of special purpose systems. When considering existing mesh generating systems, it is apparent that many are tailored to specific problems. For instance, a mesh generator may target a certain problem domain, or it may be restricted to generate meshes for a simple domain. Although general purpose mesh generating systems exist, the extra details required for interacting with a general purpose system can distract engineers if their intention is solve a relatively simple problem. For instance, if they know that for their particular problem they will always need to discretize a rectangular domain, then a package that supports an arbitrary closed shape is more complicated than they require. Special purpose mesh generators are also popular because they can be targeted to the specific physics of the problem of interest and they can produce output files in the format required by the engineers preferred finite element processor. Given the predominance of special purpose systems, engineers would greatly benefit by having a program family generator that allows rapid production of customized mesh generators.

Proper documentation of the commonality analysis has the advantage that it will allow open communication between different domain experts. Those in the field of mesh generation, or computational geometry, will be able to learn about the software engineering methodologies that they can employ to reduce their workload. For instance, computational scientists can learn something about the language of software engineering by consulting the terminology section, which includes definitions from software engineering field so that the non-software engineers can understand the commonality analysis exercise. Furthermore, the documentation will allow computer scientists and software engineers to more easily learn about mesh generation and thus contribute their ideas to the development of mesh generating systems. Proper documentation can assist the software engineering in designing a system with a module decomposition that satisfies the principles of separation of concerns and information hiding because clear guidance is provided about the likely changes to the system.

In the previous section, it was pointed out that a commonality analysis document should focus on “What” as opposed to “How”. It should be clarified that the notion of “How” is a relative one. Although in the current analysis the commonalities and variabilities focus on "What" is required, in some contexts the choice to use a mesh generator would be considered a "How" decision. For instance, in the context of solving a partial differential equation in a scientific computing problem, the decision to use a finite element analysis, and thus a mesh generator, would be considered a design decision, not a requirement. As a consequence of this difference in context, a requirements template for scientific and engineering computation problems, such as that proposed by Lai (2004) and Smith et al. (2004), cannot be directly employed to document mesh generating systems.

Since the commonality analysis for mesh generators focuses on "What" is required and not "How" to do it, the different mesh generating algorithms should not be considered as variabilities. Instead, the mesh quality attributes, such as element aspect ratios, minimum angles, maximum angles, *etc.*, are variabilities. The choice of meshing algorithm should only be made after the commonality analysis and associated requirement specification are complete. Once it is known what qualities are required for the mesh, then during the design phase a meshing algorithm can be selected that can provide these qualities. A review of meshing algorithms and some insight into how they influence mesh quality can be found in Frey and George (2000); Teng and Wong (2000); Bern and Plassmann (2000); Bern and Eppstein (1992).

3 Terminology and Definitions

This section is divided into two subsections. The first discusses the terminology that comes from the software engineering field, while the second presents the definitions used in mesh generation. Common acronyms are also listed in this section. The lists are not intended to be read sequentially, but rather to be consulted for reference purposes; therefore, the terms are ordered alphabetically, with the consequence that some terms that appear early in the list depend on the definitions of later terms.

3.1 Software Engineering Related Definitions and Acronyms

Commonality: A requirement or goal common to all family members.

Goal: "Goals capture, at different levels of abstraction, the various objectives the system under consideration should achieve." van Lamsweerde (2001)

Program Family: A set of programs that are analyzed and designed together starting from the initial stages of the software development life-cycle.

Requirements: A software requirement is: *i*) a condition or capability needed by a user to solve a problem or achieve an objective; *ii*) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document; or, *iii*) a documented representation of a condition or capability as in the above two definitions. (Thayer and Dorfman, 2000)

Variability: A requirement or goal that varies between family members.

3.2 Mesh Generation Related Definitions and Acronyms

1D: One Dimensional

2D: Two Dimensional

3D: Three Dimensional

Conformal mesh: A conformal mesh is a mesh following the definition of a mesh, with the addition of the following property (Frey and George, 2000):

1. the intersection of two elements in τ is either the empty set, a vertex, an edge or a face (when the dimension is 3)

Cell: Another name for an element, as defined below.

Connectivity: There are two types of connectivity, one for the mesh and one for a mesh element:

1. "The connectivity of a mesh is the definition of the connection between its vertices." (Frey and George, 2000)
2. "The connectivity of a mesh element is the definition of the connections between the verticies at the element level." (Frey and George, 2000)

Degree of Freedom (dof) : In the finite element method the dependent variables that are being solved for are referred to as degrees of freedom. They may correspond to displacements, velocities, pressures, stresses, temperatures, *etc.*

Domain : The area or volume that is to be discretized. The domain is sometimes referred to as the computational domain.

Element: The original domain is discretized into smaller, usually simpler, shapes called elements. The typical shapes for elements in 1D is a line, in 2D is a triangle or a quadrilateral, and in 3D a tetrahedron or a hexahedron. Elements are also called cells.

Grid: A structured mesh is sometimes referred to as a grid, if it is constructed of quadrilateral elements in 2D or hexahedral elements in 3D.

Hybrid mesh: "A mesh is said to be hybrid if it includes some elements with a different spatial dimension." (Frey and George, 2000)

M.G.s: Mesh Generators

Manifold: "A surface mesh is called manifold if its internal edges are shared by exactly two elements (one element in the case of a boundary edge for an open surface)." (Frey and George, 2000)

Mesh: Let Ω be a closed bounded domain in $\mathbb{R}$ or $\mathbb{R}^2$ or $\mathbb{R}^3$ and let K be a simple shape, such as a line segment in 1D, a triangle or a quadrilateral in 2D, or a tetrahedron or hexahedron in 3D. A mesh of Ω, denoted by τ, has the following properties:

1. $\Omega \approx \cup(K | K \epsilon \tau : K)$, where $\cup$ is first closed and then opened
2. the length of every element K, of dimension 1, in τ is greater than zero
3. the interior of every element K, of dimension 2 or greater, in τ is nonempty
4. the intersection of the interior of two elements is empty

The above definition of a mesh is the same as that from Frey and George (2000), but the equality ($=$) has been changed to approximate equality ($\approx$) because it will not always be possible for the mesh discretization to exactly match the boundary of the given domain. For instance, if the boundary of a 2D domain is curved, it cannot be exactly matched with a non-infinite number of straight-edged triangles. The definition also differs from Frey and George (2000) by allowing for 1D elements.

Mesh Generation: The automatic mesh generation problem is that of attempting to define a set of nodes and elements to best describe a geometric domain, subject to various element size and shape criteria.

Mixed mesh: "A mesh is said to be mixed if it includes some elements of a different geometric nature." (Frey and George, 2000)

Multi-block mesh: A region can be divided, without holes or overlaps into a set of contiguous subdomains/blocks, which may be considered as an unstructured grid. Within each subdomain, a separate structured grid is then generated. This type of mesh is referred to as a block-structured or multi-block grid.

Node: In the finite element method the degrees of freedom are located at the nodes. Nodes are often located at the verticies, but they can also be placed at locations such as the midpoints of edges or the centroid of the element. If there are more nodes than vertices and if the nodes are used to interpolate the geometry, then the element can have curved boundaries.

Overlaid mesh: Overlaid meshes occur when two meshes are overlaid on top of each other, with no requirement to have the verticies between the two meshes match. This kind of mesh may be used when one object is on top of another and the two objects are best modelled using different coordinate systems. (Thompson et al., 1985)

Physical Attribute: Information on the properties of a mesh that are related to the physical problems. Some examples include material properties, boundary conditions, areas, *etc.*

Structured mesh: The mesh in which the local organization of the grid points and the form of the grid cells do not depend on their position but are defined by a general rule. There is a pattern to the topology that repeats. Frey and George (2000) say, "a mesh is called structured if its connectivity is of the finite difference type." They go on to remark, "Peculiar meshes other than quad or hex meshes could have a structured connectivity. For instance, one can consider a classical grid of quads where each of them are subdivided into two triangles using the same subdivision pattern."

Topology: "The topology of a mesh element is the definition of this element in terms of its faces and edges, these last two being defined in terms of the element's vertices." (Frey and George, 2000)

Unstructured mesh: The mesh whose element connectivity of the neighbouring grid verticies varies from point to point. Any mesh that is not structured is an unstructured mesh.

Vertices: The locations that define the shape of the cells. In 1D the verticies are the end-points of the elements. For 2D and 3D elements the verticies correspond to the location in space that defines the intersection of the edges of an element.

4 Commonalities

This section lists all the common features among all the potential family members. The commonalities are organized using the following abstraction of the system, which can be used to describe all mesh generators: input information, then generate the mesh and finally output the results. Section 4.1 describes the commonalities for the mesh generation step, which includes the discretization of the domain, as well as other information on the problem such as the boundary conditions, material properties, *etc.* Section 4.2 highlights the input information that is required for all mesh generators, such as the geometry of the domain that is going to be discretized. The next section, Section 4.3, shows the common features for the output of mesh generators, such as the requirement that mesh information be written to files. (Although the mesh information could simply be written to the computer's memory, in all practical applications it is desirable to have a persistent record of the mesh that was created.) The final section covers qualities of the system that cannot be classified as input, mesh generation or output. These commonalities are termed nonfunctional requirements of the system. For instance, all systems will have the goal that the response time to a user's request is small enough to allow the user to focus on his/her problem and to maintain his/her train of thought, without being distracted by excessive waiting time. The commonality in this case is refined by a later variability because the specific requirement on the response time will depend on the intended usage of the mesh generating system.

Each commonality below uses the same structure. All of the commonalities are assigned a unique item number, which takes the form of a natural number with the prefix "C". Following this, a description of the commonality is provided along with a list of related variabilities, which are given as hyperlinks that allow navigation of the document to the text describing the variability. Finally, each commonality ends with a summary of the history, including the date of creation and any dates of modification, along with a brief description of the modification. If necessary, a previous version of the document can be obtained by using the concurrent versioning system where the files are stored.

The commonalities listed in this section are for a mesh generator that is acting as a preprocessor for a finite element analysis program, which means that the mesh generator will produce data files with information on the discretization of the domain, the boundary conditions, the material properties, the system properties, *etc.* In some cases the mesh generating systems only focus on the discretization of the domain and leave it to another system to provide the other information necessary for the finite element analysis. For systems that only focus on the discretization of the domain the following subset of the commonalities applies: C1, C2, C3, C4, C9, C12, C13, C14, C15, C16, C17, C18, and C19.

4.1 Mesh Generation

Item Number	C1
Description	A mesh generator discretizes a given computational domain (closed boundary Ω) into a covering up of a finite number of simple shapes, as described in the definition of a mesh on page 11.
Related Variability	V3, V4, V5, V6, V9, V13, V16, V17, V18, V19, V20
History	Created - May 7, 2004

Item Number	C2
Description	Each vertex has a unique identifier.
Related Variability	none
History	Created - October 21, 2002; Modified April 29, 2004 ("a unique integer" was changed to "a unique identifier" to make the commonality more generic.); Modified June 14, 2004 ("node" was changed to "vertex" to be consistent with the distinction between nodes and verticies as introduced in the terminology section.)

Item Number	C3
Description	Each element has a unique identifier.
Related Variability	none
History	Created - October 21, 2002; ; Modified April 29, 2004 (“a unique integer” was changed to “a unique identifier” to make the commonality more generic.)

Item Number	C4
Description	An element’s topology is given by the connectivity of its set of verticies.
Related Variability	V8
History	Created - October 21, 2002

Item Number	C5
Description	Information on the created meshes includes material properties.
Related Variability	V24, V25
History	Created - October 24, 2004

Item Number	C6
Description	Information on the created meshes includes boundary conditions.
Related Variability	V26, V28, V29
History	Created - October 24, 2004

Item Number	C7
Description	Information on the created meshes includes system parameters, such as the number of elements in the domain and numerical parameters needed by the finite element analysis program.
Related Variability	V27
History	Created - October 24, 2004

4.2 Input

Item Number	C8
Description	A mesh generator requires that information be input by the user to define his/her meshing problem.
Related Variability	V21, V22, V27, V28, V29
History	Created - October 21, 2002; Modified - June 14, 2004 (the wording was changed to improve the flow of the sentence.)

Item Number	C9
Description	The user defines the geometric domain of the problem by a closed boundary.
Related Variability	V23
History	Created - October 21, 2002; Modified - Jun 14, 2004 (A related variability has been added.)

Item Number	C10
Description	The user needs to specify the physical attributes, such as the material properties, the boundary conditions, *etc.*
Related Variability	V24, V25, V26, V27, V28, V29
History	Created - October 21, 2002

Item Number	C11
Description	When boundary conditions are specified, a maximum of one condition may be given for each degree of freedom (dof). For instance, a dof cannot have both a prescribed displacement and a prescribed force.
Related Variability	V26
History	Created - October 21, 2002; Modified - October 24, 2004 (The wording was modified to clarify the meaning.)

4.3 Output

Item Number	C12
Description	Mesh generators write mesh information to output file(s).
Related Variability	V31, V32, V33
History	Created - October 21, 2002

Item Number	C13
Description	The element information of a mesh is listed in the output file in some order.
Related Variability	V34
History	Created - October 21, 2002; Modified - June 16, 2004 (The sentence was changed to clarify that it is the element information that is listed.)

Item Number	C14
Description	The vertex information, such as the coordinates, for a mesh is listed in output file(s) in some order.
Related Variability	V35
History	Created - October 21, 2002; Modified - June 16, 2004 (The sentence was changed to clarify that it is the vertex information that is listed.)

4.4 Nonfunctional Requirements

Item Number	C15
Description	The response time to a user's request is small enough to allow the user to focus on his/her problem and to maintain his/her train of thought, without being distracted by excessive waiting times.
Related Variability	V39
History	Created - October 22, 2002; Modified - June 14, 2004 (the concept of a reasonably small response time was made less ambiguous.); Modified - October 24, 2004 (a related variability was added.)

Item Number	C16
Description	The mesh generator provides the accuracy required for the particular problems it is intended to help solve.
Related Variability	V40
History	Created - October 22, 2002; Modified - June 14, 2004 (The wording was changed to emphasize that the accuracy requirements are problem dependent.); Modified - October 24, 2004 (A related variability was added.)

Item Number	C17
Description	The mesh generator provides the precision required for the particular problems it is intended to help solve.
Related Variability	V41
History	Created - June 14, 2004; Modified - October 24, 2004 (A related variability was added.)

Item Number	C18
Description	The mesh generator is robust enough to handle the types of users and the types of problems that the system is expected to encounter.
Related Variability	V42
History	Created - June 16, 2004; Modified - October 24, 2004 (A related variability was added.)

Item Number	C19
Description	The mesh generator will be as portable to other operating systems as required by the users of the system.
Related Variability	V37
History	Created - June 16, 2004

5 Variabilities

This section provides a list of characteristics that may vary among family members. As in Section 4, the first three subsections on variabilities are organized into the following sublists: Mesh Generation, Input and Output. The final two subsections list variabilities that can be characterized as system constraints and as nonfunctional requirements.

As for the commonalities, each variability is labelled with a unique item number. In this case the numbers are prepended with the letter "V". The other four headings provided for each variability are: Description, Related Commonality, Related Parameter and History. The related commonalities and parameters are given as a set of identifiers that respectively refer back to the previous section on commonalities or refer forward to the next section on parameters of variation.

5.1 Mesh Generation

Item Number	V1
Description	Different mesh generators will be able to accommodate the creation of meshes for different problem domains.
Related Commonality	none
Related Parameter	P1
History	Created - October 22, 2002; Modified - June 15, 2004 (The wording was changed to provide a better description of the variability.)

Item Number	V2
Description	The degree of generality of the mesh generator. Some mesh generators are general purpose programs, while others are tailored to a specific application domain.
Related Commonality	none
Related Parameter	P2
History	Created - June 15, 2004

Item Number	V3
Description	Some mesh generators have automated capabilities for improving/changing an existing mesh, including techniques for mesh smoothing.
Related Commonality	C1
Related Parameter	P3
History	Created - October 22, 2002; Modified - June 15, 2004 (changed the wording to clarify that the capabilities apply to changing the mesh.)

Item Number	V4
Description	Some mesh generators allow manual mesh editing. That is the user is allowed to tweak such information as the vertex location.
Related Commonality	C1
Related Parameter	P4
History	Created - October 22, 2002; Modified - June 15, 2004 (Added an example of mesh editing.)

Item Number	V5
Description	In the mesh generating stage each vertex is assigned a unique ID. The order of these IDs impacts the bandwidth of the stiffness matrix in the finite element analysis program; therefore, some mesh generators have algorithms and heuristics for modifying the order of the verticies to reduce the bandwidth.
Related Commonality	C1, C2
Related Parameter	P5
History	Created - May 12, 2003; Modified - June 15, 2004 (Made the connection between bandwidth and the system of equations in the finite element program.)

Item Number	V6
Description	There exists variety in the degree of structure exhibited by a mesh.
Related Commonality	C1
Related Parameter	P6
History	Created - October 22, 2002; Modified - June 15, 2004 (Clarified that the variability is with respect to structure.)

Item Number	V7
Description	For structured meshes different templates for the local patterns in the element topology are possible.
Related Commonality	C1
Related Parameter	P7
History	Created - October 24, 2004

Item Number	V8
Description	Different mesh generators can assume a different ordering for the local numbering of an element's vertices and nodes.
Related Commonality	C4
Related Parameter	P8
History	Created - October 22, 2002; Modified - June 15, 2004 (The variability now mentions both nodes and verticies.)

Item Number	V9
Description	The shape of the elements generated by the mesh generator as defined by their vertices.
Related Commonality	C1
Related Parameter	P9
History	Created - October 22, 2002; Modified - June 5, 2004 (Changed the wording to clearly say that it is the shape of the elements that is the variability.)

Item Number	V10
Description	The number of nodes for an element and the location of those nodes.
Related Commonality	none
Related Parameter	P10
History	Created - June 5, 2004

Item Number	V11
Description	The number of degrees of freedom at a node and the meaning of each of those degrees of freedom.
Related Commonality	none
Related Parameter	P11
History	Created - June 5, 2004

Item Number	V12
Description	The pattern of the number of degrees of freedom and the meaning of these degrees of freedom can vary between the nodes of an element.
Related Commonality	none
Related Parameter	P12
History	Created - June 5, 2004

Item Number	V13
Description	The dimensionality of the computational domain.
Related Commonality	C1
Related Parameter	P13
History	Created - June 5, 2004

Item Number	V14
Description	The shape allowed for the computational domain.
Related Commonality	C1, C9
Related Parameter	P14
History	Created - October 24, 2004

Item Number	V15
Description	The quality of the resulting mesh.
Related Commonality	C1
Related Parameter	P15
History	Created - June 5, 2004

Item Number	V16
Description	The ability of the system to accommodate a mixed mesh.
Related Commonality	C1
Related Parameter	P16
History	Created - June 5, 2004

Item Number	V17
Description	The ability of the system to accommodate a hybrid mesh.
Related Commonality	C1
Related Parameter	P17
History	Created - June 5, 2004

Item Number	V18
Description	The ability of the system to accommodate a conformal versus a nonconformal mesh.
Related Commonality	C1
Related Parameter	P18
History	Created - June 5, 2004

Item Number	V19
Description	The dimension of the coordinate system used to describe the geometry (coordinates) of the vertices and possibly of the nodes.
Related Commonality	C1, C9
Related Parameter	P19
History	Created - June 5, 2004

Item Number	V20
Description	The type of the coordinate system used to describe the geometry (coordinates) of the vertices and possibly of the nodes.
Related Commonality	C1, C9
Related Parameter	P20
History	Created - June 5, 2004

5.2 Input

Item Number	V21
Description	Some mesh generators provide a graphical user interface while others provide a text based interface.
Related Commonality	C8
Related Parameter	P21
History	Created - October 22, 2002; Modified - June 15, 2004 (Changed user friendly interface to graphical interface.)

Item Number	V22
Description	The interface for specifying the closed boundary of the computational domain (Ω).
Related Commonality	C8, C9
Related Parameter	P22
History	Created - October 22, 2002; Modified - June 15, 2004 (The description was simplified.)

Item Number	V23
Description	The mathematical form used to specify the closed boundary of the computational domain (Ω).
Related Commonality	C9
Related Parameter	P23
History	Created - June 15, 2004

Item Number	V24
Description	The number of material properties, their names and their types.
Related Commonality	C5, C10
Related Parameter	P24
History	Created - June 15, 2004

Item Number	V25
Description	The number of different materials allowed in the specification of the physical problem.
Related Commonality	C5, C10
Related Parameter	P25
History	Created - June 15, 2004

Item Number	V26
Description	The types of boundary conditions accommodated by the system.
Related Commonality	C6, C10, C11
Related Parameter	P26
History	Created - June 15, 2004

Item Number	V27
Description	The number and type of different system parameters input to the system. These parameters will be passed on to the finite element program.
Related Commonality	C7, C8, C10
Related Parameter	P27
History	Created - June 15, 2004

Item Number	V28
Description	The system may allow the user to specify that two degrees of freedom will be constrained to have the same value. As far as the finite element analysis is concerned the two dof will be solved for as one dof.
Related Commonality	C8, C10
Related Parameter	P28
History	Created - June 15, 2004; Modified - October 25, 2004 (The wording was changed to clarify this variability.)

Item Number	V29
Description	Constraints on the mesh may be inputs for the system.
Related Commonality	C8, C10
Related Parameter	P29
History	Created - June 15, 2004

5.3 Output

Item Number	V30
Description	Some mesh generators provide visualization of the meshes produced.
Related Commonality	none
Related Parameter	P30
History	Created - October 22, 2002

Item Number	V31
Description	Mesh information is output in either text or binary format.
Related Commonality	C12
Related Parameter	P31
History	Created - October 22, 2002

Item Number	V32
Description	The number of files that are output by the mesh generator.
Related Commonality	C12
Related Parameter	P32
History	Created - June 15, 2004

Item Number	V33
Description	The format of the information in the file(s) output by the mesh generator.
Related Commonality	C12
Related Parameter	P33
History	Created - June 15, 2004

Item Number	V34
Description	The element information is written to the file(s) following a different ordering.
Related Commonality	C13
Related Parameter	P34
History	Created - October 22, 2002; Modified - June 16, 2004 (The description was reworded to clarify the meaning.)

Item Number	V35
Description	The vertex information is written to the file(s) following a different ordering.
Related Commonality	C14
Related Parameter	P35
History	Created - October 22, 2002; Modified - June 16, 2004 (The description was reworded to clarify the meaning.)

Item Number	V36
Description	The degree to which the user can customize the output file formats.
Related Commonality	C12
Related Parameter	P36
History	Created - October 24, 2004

5.4 System Constraints

Item Number	V37
Description	The operating systems on which the mesh generating system is intended to run.
Related Commonality	C19
Related Parameter	P37
History	Created - June 15, 2004

Item Number	V38
Description	The amount of persistent storage available for storing the generated data files.
Related Commonality	none
Related Parameter	P38
History	Created - June 16, 2004

5.5 Nonfunctional Requirements

Item Number	V39
Description	The response time required for user interaction with the system varies. The user will expect a faster response time for simple input operations, and will tolerate longer wait time when the system is calculating the mesh and generating the output files.
Related Commonality	C15
Related Parameter	P39
History	Created - October 24, 2004

Item Number	V40
Description	The tolerance allowed for each output produced by the system, where the tolerance is a relative quantity defined as $(calculated\ value - true\ value)/(true\ value)$.
Related Commonality	C16
Related Parameter	P40
History	Created - October 24, 2004

Item Number	V41
Description	The number of decimal digits of precision allowed for each input and displayed for each output value.
Related Commonality	C17
Related Parameter	P41
History	Created - October 24, 2004

Item Number	V42
Description	Systems will have different degrees of input error checking and runtime exception handling.
Related Commonality	C18
Related Parameter	P42
History	Created - October 24, 2004

6 Parameters of Variation

This section specifies the parameters of variation for the variabilities listed in Section 5. They are organized into the same five subcategories as employed previously: Mesh Generation, Input, Output, System Constraints, Nonfunctional Requirements.

Each parameter of variation is given a unique identifier of the form “P” followed by a natural number. The corresponding variability is listed and a hyperlink is provided that allows navigation back to the appropriate item in Section 5. The final entry for each parameter of variation is the binding time, which is the time in the software lifecycle when the variability is fixed. The binding time could be during specification, or during building of the system (compile time), or during execution of the system (run time). It is possible to have a mixture of binding times. For instance, a parameter of variation could have a binding time of “specification or building” to represent that the parameter could be set at specification time, or it could be postponed until the given family member is built. The choice of postponing the decision until the build would be associated with the presence of a domain specific language that would allow postponing decisions on the values of the parameter of variation.

6.1 Mesh Generation

Item Number	P1
Corresponding Variability	V1
Range of Parameters	Mesh generating systems can build meshes for a large range of problem domains corresponding to the large range of problems that can be solved via finite element analysis. For instance, the mesh data files can be targeted toward the following problem domains: solid mechanics, fluid mechanics, heat transfer, seepage, electrostatics, *etc.*
Binding Time	specification or run time

Item Number	P2
Corresponding Variability	V2
Range of Parameters	A continuum exists from the most specialized systems to arbitrarily general systems. For instance, a special purpose system may involve quadilateral elements on a 2D rectangular domain for the purpose of solving for the temperature in a heated plate. On the other hand, a general purpose system would handle an arbitrary geometry for the domain, provide a choice of element shapes, allow for 1D, 2D or 3D domains and provide meshes for a variety of physical problems.
Binding Time	specification or build time

Item Number	P3
Corresponding Variability	V3
Range of Parameters	Mesh generators may have optimization features such smoothening, and refinement/coarsening.
Binding Time	specification or build time

Item Number	P4
Corresponding Variability	V4
Range of Parameters	The mesh generators with graphical user interfaces can incorporate some degree of mesh-editing features. Some mesh generators do not have this feature.
Binding Time	specification or build time

Item Number	P5
Corresponding Variability	V5
Range of Parameters	Some mesh generators will provide options for bandwidth reduction. The requirements for the mesh generating systems will differ in how small a bandwidth they must achieve.
Binding Time	specification time or build time

Item Number	P6
Corresponding Variability	V6
Range of Parameters	The options are as follows: structured mesh, unstructured mesh, multi-block mesh, overlaid mesh.
Binding Time	specification or build or run time

Item Number	P7
Corresponding Variability	V7
Range of Parameters	Seven (7) potential local topology templates are possible, as shown in Appendix A.
Binding Time	specification or build or run time

Item Number	P8
Corresponding Variability	V8
Range of Parameters	The local vertex numbering can be either clockwise or counterclockwise. By convention it is nearly always counterclockwise. The local nodes also need to be numbered. If the nodes coincide with the verticies, then the numbering is usually the same. If there are more nodes than verticies, then a new numbering convention needs to be adopted. Frey and George (2000) and Zienkiewicz (1977) discuss the usual numbering scheme convention.
Binding Time	specification or build or run time

Item Number	P9
Corresponding Variability	V9
Range of Parameters	In 1D there are line segments; in 2D there are triangles and quadrilaterals; in 3D there are tetrahedras and hexahedras
Binding Time	specification or build or run time

Item Number	P10
Corresponding Variability	V10
Range of Parameters	The element can have fewer nodes than verticies, the same number of nodes as verticies or more nodes than verticies. The nodes can be located at the vertices, on the element edges, or inside the element. Some examples of the large variety of elements that are used can be found in Zienkiewicz (1977)
Binding Time	specification or build or run time

Item Number	P11
Corresponding Variability	V11
Range of Parameters	The number and type of degrees of freedom at the nodes can vary between different types of elements and within an element. The dof for an element represent the dependent variable that will be solved for. Some example dof are as follows: displacements, velocities, temperatures, voltages, pressures, *etc.* Some examples of the large variety of elements that are used can be found in Zienkiewicz (1977)
Binding Time	specification or build or run time

Item Number	P12
Corresponding Variability	V12
Range of Parameters	If the geometry is interpolated at fewer nodes than the interpolation of the dof, then the element is subparametric. If the geometry is interpolated at the same number of nodes as the interpolation for the dof, then the element is isoparametric. If the geometry is interpolated at more nodes than the interpolation for the dof, then the element is superparametric.
Binding Time	specification or build or run time

Item Number	P13
Corresponding Variability	V13
Range of Parameters	1D, 2D, or 3D
Binding Time	specification or build or run time

Item Number	P14
Corresponding Variability	V14
Range of Parameters	The computational domain in 1D can be either a straight line or a curve. For 2D and 3D the domain can consist of simple shapes, such as triangles (tetrahedra), rectangles (boxes), parallelograms (parallelepipeds), *etc.* or more complex domains constructed from polygons and polyhedra, or the domain can constructed using curved boundaries or surfaces. The domain could be a surface, which is a 2D domain located in a 3D coordinate system. The computational domain can allow for holes, or disallow holes. The computational domain can be either connected or unconnected. The computational domain may or may not allow for internal constraints within the domain that require verticies to follow a prescribed internal boundary.
Binding Time	specification or build time

Item Number	P15
Corresponding Variability	V15
Range of Parameters	There are many potential mesh quality measures, such as the aspect ratio, the minimum angle, the maximum angle, *etc.* The quality measure can change if the problem in question is anisotropic versus isotropic.
Binding Time	specification or build or run time

Item Number	P16
Corresponding Variability	V16
Range of Parameters	A mesh generator will or will not be able to accommodate mixed meshes. The combinations of different element types also provides a range of parameters.
Binding Time	specification or build time

Item Number	P17
Corresponding Variability	V17
Range of Parameters	A mesh generator will or will not be able to accommodate hybrid meshes. The combinations of spatial dimensions that the problem can handle also provides a range of parameters.
Binding Time	specification or build time

Item Number	P18
Corresponding Variability	V18
Range of Parameters	A mesh generator will or will not be able to accommodate conformal meshes.
Binding Time	specification or build time

Item Number	P19
Corresponding Variability	V19
Range of Parameters	The dimension of the spatial coordinate system that is used to express the geometric coordinates. The options are 1D, 2D or 3D. The dimensionality of the shape of the elements does not have to be the same as the coordinate system used for the geometry. For instance, a surface mesh uses 2D element shapes, but is embedded in a 3D space.
Binding Time	specification or build or run time

Item Number	P20
Corresponding Variability	V20
Range of Parameters	The type of coordinate system can be Cartesian, polar, spherical, or another type.
Binding Time	specification, or build or run time

6.2 Input

Item Number	P21
Corresponding Variability	V21
Range of Parameters	Some mesh generators provide a graphical user interface, which helps to minimize the user's effort in defining the seed information, while other mesh generators instead use a text-based interface that requires the user to type the seed information, often in a given file format.
Binding Time	specification or build time

Item Number	P22
Corresponding Variability	V22
Range of Parameters	Multi-block mesh generators let the user define subregions, while unstructured mesh generators let the user define boundary entities. Structured mesh generators ask the user to specify the number of subdivisions along each direction, while unstructured mesh generators ask for the size of the elements.
Binding Time	specification or build time

Item Number	P23
Corresponding Variability	V23
Range of Parameters	The options include parametric representations, explicit representations and implicit representations, as presented in Frey and George (2000). A parametric representation is given by a function $\gamma(t)$, where t is a parameter and $\gamma(t) \in \mathbb{R}^2$ or $\gamma(t) \in \mathbb{R}^3$. Implicit curves are given by relations like $f(x, y) = 0$ and $f(x, y, z) = 0$ in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively, where x, y and z denote the coordinates. Explicit curves take the form $y = f(x)$ in $\mathbb{R}^2$.
Binding Time	specification or build or run time

Item Number	P24
Corresponding Variability	V24
Range of Parameters	The number of material properties is variable and can include such properties as elastic modulus, viscosity, relaxation time, thermal conductivity, *etc.*
Binding Time	specification or build or run time

Item Number	P25
Corresponding Variability	V25
Range of Parameters	The entire domain can consist of one material or there may be any finite number of different materials.
Binding Time	specification or build or run time

Item Number	P26
Corresponding Variability	V26
Range of Parameters	The boundary conditions may be of the Dirichilet, Neumann or of a mixed type. If the boundary conditions are for prescribed values (Dirichilet type) that are zero, they may be specified in a different manner from other prescribed values. For instance, in solid mechanics problems a boundary may be fixed in one or more directions so that it cannot move in that direction and it will be free in the remaining directions. The input may specify this kind of fixity information.
Binding Time	specification or build time

Item Number	P27
Corresponding Variability	V27
Range of Parameters	The number and meaning of the system parameters can vary from one mesh generator to the next. System parameters may include global numerical parameters, such as the degree of implicitness for a time marching scheme.
Binding Time	specification or build or run time

Item Number	P28
Corresponding Variability	V28
Range of Parameters	In some mesh generators it may be possible to set the problem so that two, or more, degrees of freedom will have the same value after the numerical solution has been calculated.
Binding Time	specification or build time

Item Number	P29
Corresponding Variability	V29
Range of Parameters	In some cases it may be possible to specify internal boundaries that the mesh will be required to follow.
Binding Time	specification or build time

6.3 Output

Item Number	P30
Corresponding Variability	V30
Range of Parameters	Some mesh generators display the resulting mesh on the screen.
Binding Time	specification or build time

Item Number	P31
Corresponding Variability	V31
Range of Parameters	Some mesh generators produce binary output file(s), some produce text (ASCII) output file(s) and some may produce a mix of both types of files.
Binding Time	specification or build or run time

Item Number	P32
Corresponding Variability	V32
Range of Parameters	The number of files can range from 1 to many. In the case of many files the data can be split between files, possibly so that geometry data, topology data, material properties data, *etc.* are separated.
Binding Time	specification or build or run time

Item Number	P33
Corresponding Variability	V33
Range of Parameters	Different mesh generators organize mesh information into file(s) in different orders. The data structure that is output can change between mesh generators, or it may be something that the user can customize within a given mesh generator.
Binding Time	specification, build or run time

Item Number	P34
Corresponding Variability	V34
Range of Parameters	Some mesh generators list elements in an increasing order (implicity), while other explicitly output the element identifier and list them in an arbitrary order.
Binding Time	specification or build or run time

Item Number	P35
Corresponding Variability	V35
Range of Parameters	Some mesh generators list nodal information in ascending order (implicitly), but others explicitly output the node's identifier and list them in an arbitrary order.
Binding Time	specification or build or run time

Item Number	P36
Corresponding Variability	V36
Range of Parameters	Some mesh generators will have a fixed file format, while others will allow the user to customize the output. The customization can range from modifying file names, to changing the order of blocks of data, to splitting the data between files, to changing the data structure, to changing from text to binary, *etc.*
Binding Time	specification or build or run time

6.4 System Constraints

Item Number	P37
Corresponding Variability	V37
Range of Parameters	Most operating systems that are in common use will have mesh generators that will run on the system. Some examples include: Windows, Unix, Mac OS X, Linux, *etc.*
Binding Time	specification, build or run time

Item Number	P38
Corresponding Variability	V38
Range of Parameters	The physical systems that the mesh generator is built on will have a limit to the amount of storage that is available. The amount could vary between kilobytes to Gigabytes, or more.
Binding Time	specification, build or run time

6.5 Nonfunctional Requirements

Item Number	P39
Corresponding Variability	V39
Range of Parameters	The maximum amount of time that the user will be expected to wait for the system to perform each of its functions. The amount of time will range from essentially instantaneous for entering user input, to seconds while waiting for graphics to display, to minutes or longer while waiting for the mesh to be generated and output files to be created.
Binding Time	specification

Item Number	P40
Corresponding Variability	V40
Range of Parameters	The tolerance for each output will be provided as a real number.
Binding Time	specification, build or run time

Item Number	P41
Corresponding Variability	V41
Range of Parameters	Each input and each output will have an associated integer that specifies the precision (number of decimal digits) with which the real number is stored and/or calculated.
Binding Time	specification, build or run time

Item Number	P42
Corresponding Variability	V42
Range of Parameters	Input error may be checked for one or more of the following: valid type, valid range, completeness, consistency, *etc.* Run-time exception handling may be checked for one or more of the following: division by zero, square root of a negative number, lack of convergence, maximum iterations exceeded, *etc.*
Binding Time	specification or build time

7 Issues

In the course of composing and revising the above documentation, the following unresolved issues arose:

- The meshes that are considered here only discretize space, there is no discretization of time; therefore, there are no "4D" meshes.
- The documentation and the definition of a mesh should be made more formal.
- A figure to show some examples of potential element, with the various options for verticies and nodes, would likely be helpful.
- The mesh quality measures should be explained in greater detail.
- The mathematics of how the closed boundary is specified could be improved by introducing the appropriate equations.
- Add some examples of general purpose mesh generators versus special purpose mesh generators in the section on the overview of mesh generators.

Acknowledgements

The financial support of the Natural Sciences and Engineering Research Council (NSERC) and of Material and Manufacturing Ontario (MMO) are gratefully acknowledged.

References

Mark Ardis and David M. Weiss. Defining families: The commonality analysis. In *Proceedings of the Nineteenth International Conference on Software Engineering*. ACM, Inc., 1997.

Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In D.-Z. Du and F.K. Hwang, editors, *Computing in Euclidean Geometry*. World Scientific, 1992.

Marshall Bern and Paul Plassmann. *Mesh Generation*. Handbook of Computational Geometry, Elsevier Science, 2000.

Guntram Berti and Georg Bader. Design principles of reusable software components for the numerical solution of pde problems. *presented at the the 14th GAMM-Seminar Kiel on Concepts of Numerical Software*, January 1998.

T. D. Blacker, Robert A. Kerr, Patrick Knupp, Robert W. Leland, Darryl J. Melander, Scott A. Mitchell, Steven J. Owen, Jason F. Sheperd, Timoshy J. Tautges, David R. White, Steve Benzley, Michael J. Borden, Steven R. Jankovich, Jason Kraftcheck, Yong Lu, Ray J. Meyers, Michael Stephenson, Steve Storm, Eric Nielsen, and Rammagy Yoeu. Cubit mesh generation environment, volume 1: User's manual. Technical Report SAND94-1100, Sandia National Laboratories, Albuquerque, New Mexico, May 1994.

Chien-Hsien Chen. A software engineering approach to developing mesh generators. Master's thesis, McMaster University, Hamilton, Ontario, Canada, 2003.

David A. Cuka and David M. Weiss. Specifying executable commands: An example of fast domain engineering. *Submitted to IEEE Transactions on Software Engineering*, pages 1 – 12, 1997. URL `http://www.research.avayalabs.com/user/weiss/Publications.html`.

E. W. Dijkstra. *Structured Programming*, chapter Notes on Structured Programming. Academic Press, London, 1972.

Ahmed H. ElSheikh, W. Spencer Smith, and Samir E. Chidiac. Semi-formal design of reliable mesh generation systems. *Advances in Engineering Software*, accepted, 2004.

Pascal Jean Frey and Paul-Louis George. *Mesh Generation Application to Finite Elements*. Hermes Science Europe Ltd., 2000.

Rondall E. Jones. The QMESH mesh generation package. *Association for Computing Machinery SIGNUM Newsletter, vol. 10, no. 4, pp. 31-34*, December 1975.

Lei Lai. Requirements documentation for engineering mechanics software: Guidelines, template and a case study. Master's thesis, initial draft for McMaster University, Hamilton, Ontario, Canada, 2004.

David Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering*, SE-2(1):1–9, 1976.

David L. Parnas. Designing software for ease of extension and contraction. *IEEE Transactions on Software Engineering*, pages 128–138, March 1979.

W. Spencer Smith and Chien-Hsien Chen. Commonality and requirements analysis for mesh generating software. In *Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 2004)*, pages 384–387, Banff, Alberta, Canada, June 2004. Knowledge Systems Institute Gradute School, KSI, Skokie, IL, 60076, USA.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation. In *Proceedings of the NSF Workshop on Reliable Engineering Computing, September 15–17*, Savannah, Georgia, 2004. Center for Reliable Engineering Computing.

T. Tautges, T. Blacker, and S. Mitchell. The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes, 1995. URL `citeseer.ist.psu.edu/tautges95whisker.html`.

Shang-Hua Teng and Chi Wai Wong. Unstructured mesh generation: theory, practice and perspectives. *International Journal of Computational Geometry and Applications*, 10(3): 227–266, June 2000. URL `http://www-sal.cs.uiuc.edu/~steng/mesh97_changed.ps`.

R. H. Thayer and M. Dorfman, editors. *IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, Washington, DC, USA, 2nd ed. edition, 2000.

Joe F. Thompson, Z. U. A. Warsi, and C. Wayne Mastin. *Mesh Generation.* Elsevier Science Publishing Co., Inc., 1985. URL `http://www.erc.msstate.edu/publications/gridbook/index.html`.

Axel van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, pages 249–263. IEEE, IEEE Computer Society, Washington, DC, USA, August 2001.

D. Weiss and C.T.R. Lai. *Software Product Line Engineering.* Addison-Wesley, 1999.

David M. Weiss. Defining families: The commonality analysis. *Submitted to IEEE Transactions on Software Engineering*, 1997. URL `http://www.research.avayalabs.com/user/weiss/Publications.html`.

David M. Weiss. Commonality analysis: A systematic process for defining families. *Lecture Notes in Computer Science*, 1429:214 – 222, 1998. URL `citeseer.ist.psu.edu/13585.html`.

O.C. Zienkiewicz. *The Finite Element Method.* McGraw-Hill Publishing Company, 1977.

O.C. Zienkiewicz and D. V. Phillips. An automatic mesh generation scheme for plane and curved surfaces by 'isoparametric' co-ordinates. *International Journal for Numerical methods in Engineering*, 3:519–528, 1971.

8 Appendix: Topology Patterns for Structured Meshes

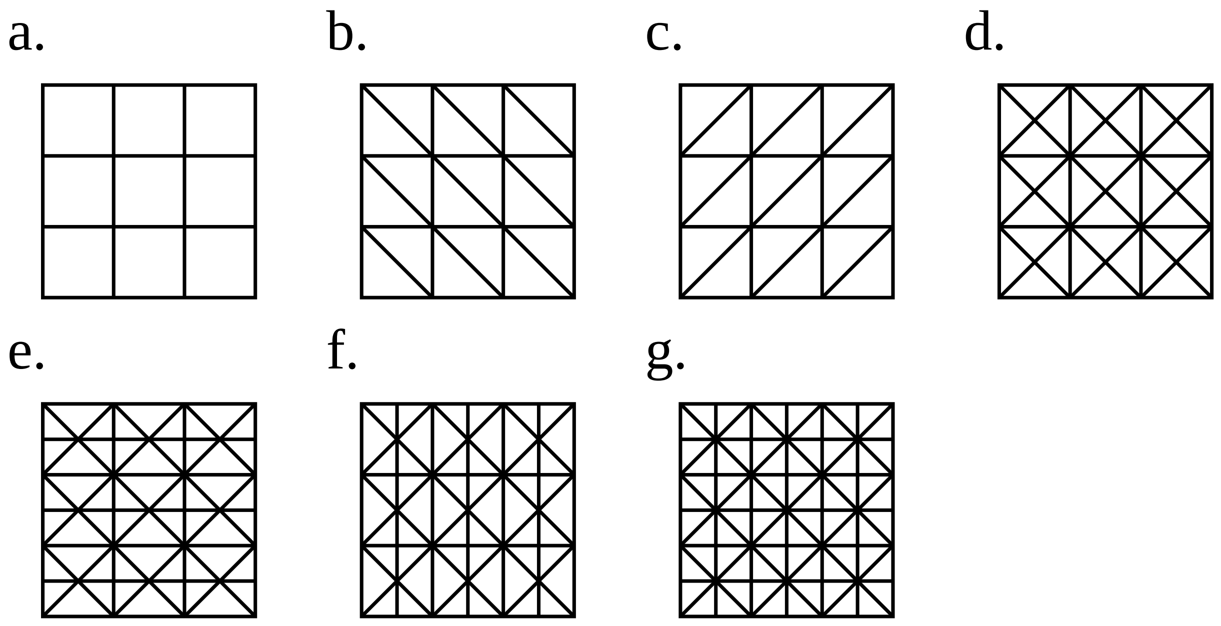

Figure 3: Parameters of variation for patterns for structured mesh discretizations