Software Requirements Specification for a Parallel Mesh Generation Toolbox

Wen Yu

September 2008

Computing and Software

McMaster University

Contents

1 Reference Material

1.1 Table of Symbols, Abbreviations and Acronyms

1.1.1 Symbols

Ω	a closed bounded domain in $\mathbb{R}^2$
Ω^*	a mesh covering the domain bounded by Ω
K	a simple shape, such as a line segment in 1D, a triangle or a quadrilateral in 2D, or a tetrahedron or hexahedron in 3D
M^{IN}	an input mesh
M^{OUT}	an output mesh
I	instructions on how a mesh should be refined/coarsened

1.1.2 Abbreviations and Acronyms

1D	One Dimensional Space
2D	Two Dimensional Space
3D	Three Dimensional Space
FEA	Finite Element Analysis
HPC	High Performance Computing
PDE	Partial Differential Equation
PMGT	Parallel Mesh Generation Toolbox
SHARCNET	Shared Hierarchical Academic Research Computing Network
SRS	Software Requirements Specification
AOMD	Algorithm Oriented Mesh Database

1.2 Index of Requirements

2 Introduction

This section gives an overview of the Software Requirements Specification (SRS) for a Parallel Mesh Generation Toolbox (PMGT). First, the purpose of the document is provided. Second, the scope of PMGT is identified. Third, some terminology for software engineering and mesh generation are defined. Finally, the organization of the document is summarized. The Table of Symbols, Abbreviation and Acronyms, and Index of Requirement are given at the beginning of the SRS.

2.1 Purpose of the Document

This SRS provides a black-box description of PMGT. The intended audience of the SRS is the development team and the users of PMGT.

2.2 Scope of the Software Product

PMGT provides a library that will be embedded into a larger application, such as a finite element analysis (FEA) program.

- The input of PMGT is an existing mesh M^{IN} with instructions I provided by the user on how the mesh should be refined/coarsened.
- PMGT refines/coarsens M^{IN} according to the supplied instructions I on how the mesh should be refined/coarsened.
- PMGT will take advantage of parallel computation.
- The output of PMGT is a refined/coarsened mesh M^{OUT}.

Note that depending on the given instruction, PMGT can either refine or coarsen the given mesh, but cannot do both at the same time. That is, any individual transition from M^{IN} to M^{OUT} will only do one of refining or coarsening. The embedding application will have access to reading the mesh information, such as information on the position of vertices and on the vertices that define a given element. However, the application cannot directly change any mesh data, except for the information indicating which elements should be refined/coarsened.

2.3 Terminology Definition

This subsection provides the definitions for terminology used in the SRS. There are two classes of terminology. One relates to software engineering, and the other relates to mesh generation. The definitions are listed in alphabetical order.

2.3.1 Software Engineering Related Terminology

Constraint: A statement that expresses measurable bounds for an element or function of the system. That is, a constraint is a factor that is imposed on the solution by force or compulsion and may limit or modify the design changes. (IEEE, 1998)

Context: The boundaries between the system that we intend to build and the people, organizations, other system and pieces of technology that have a direct interface with the system. (Robertson and Robertson, 2001)

Functional Requirements: Functional requirements define precisely what input are expected by the software, what outputs will be generated by the software, and the details of relationships that exist between those inputs and outputs. In short, functional requirements describe all aspects of interface between the software and its environment (that is, hardware, humans, and other software). (Davis, 1990)

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

Non-functional Requirements: Non-functional requirements define the overall qualities or attributes to be exhibited by the resulting software system. (Davis, 1990)

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. (IEEE, 2000)

Software Engineering: Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. (IEEE, 1990)

Software Requirements Specification: A Software Requirements Specification (SRS) is a document containing a complete description of what the software will do without describing how it will do it. (Davis, 1990)

System: An interdependent group of people, objects, and procedures constituted to achieve defined objectives or some operational role by performing specified functions. (IEEE, 1998)

System Context: System Context documents the relationships between the system being specified and other human and computer systems. (Sommerville, 1992)

User: The person, or persons, who operate or interact directly with the product. (IEEE, 2000)

2.3.2 Mesh Generation Related Terminology

Cell: Another name for an element, as defined in page 7.

Conformal Mesh: A conformal mesh is a mesh (defined on page 8) following the definition of a mesh, with the addition of the following property: The intersection of two elements in the mesh Ω^* is either the empty set, a vertex, an edge or a face (when the dimension is 3). (Frey and George, 2000)

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 vertices at the element level." (Frey and George, 2000)

Domain: The area or volume that is to be discretized. The domain is sometimes referred to as the computational domain. (Smith and Chen, 2004)

Edge: An edge is a line segment between two vertices.

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. (Smith and Chen, 2004)

Embedding Application: The software that uses PMGT.

Face: A face is a maximal connected subset of the plane without vertices inside the subset. In 2D, a face is a cell.(Frey and George, 2000)

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

Mesh: In Smith and Chen (2004), a mesh is defined as follows:
Let Ω be a closed bounded domain in $\mathbb{R}$ or $\mathbb{R}^2$ or $\mathbb{R}^3$ and let K be an element. A mesh of Ω, denoted by Ω^*, has the following properties:

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

The only difference between above definition and the definition given by Frey and George (2000) is that equality ($=$) had been changed to approximate equality ($\approx$).

Mesh Generation: The automatic mesh generation problem is that of attempting to define a set of elements to best describe a geometric domain, subject to various element size and shape criteria. (Smith and Chen, 2004)

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

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)
The topology of a mesh is the set of topologies of its constitute mesh elements.

Unstructured Mesh: The mesh whose element connectivity of the neighbouring grid vertices varies from point to point. Any mesh that is not structured is an unstructured mesh. (Smith and Chen, 2004)

Vertices: The locations that define the shape of the cells. In 1D the vertices are the end-points of the elements. For 2D and 3D elements the vertices correspond to the location in space that defines the intersection of the edges of an element. (Smith and Chen, 2004)

2.4 Organization of the Document

This SRS follows the template introduced by Lai (2004). Lai's template targets an SRS for scientific computing software. In particular, the example shown is for engineering mechanics software, such as software to analyze beams. In the current work, Lai's template is modified to fit PMGT, which is a more general purpose software. For example, the instanced model section of Lai's template is removed since PMGT is not designed for solving a specific physical problem.

Section 2 (this section) is an introduction to the SRS. The rest of the document is arranged as follows. Section 3 provides the general information about the system. Section 4 is the major part of the SRS. All functional requirements and non-functional requirements of the software are presented in this section. Section 5 discusses some other system issues. Section 6 gives a traceability matrix that summaries the association of each requirement with goals, assumptions, theoretical models and data definitions introduced in 4. This SRS also contains the list of possible changes in the requirements and values of auxiliary constants. The references are listed at the end of this document.

3 General System Description

This section describes the general information about the system. The interfaces between the system and its environment are defined first. Then the characteristics of potential users are discussed. At end of this section, some system constraints are described.

3.1 System Context

The software to be built is a library tool that will be called by other applications. There is no direct interaction between the system and the end users. Users of the embedding application, such as an FEA program, provide some parameters directly to the FEA program. Some of these parameters are passed to PMGT by the FEA program. The interface between PMGT and the embedding application should only show what PMGT can do and hide the information about how to do it. Therefore, users who are not experts in mesh generation or in parallel processing will be able to use this toolbox.

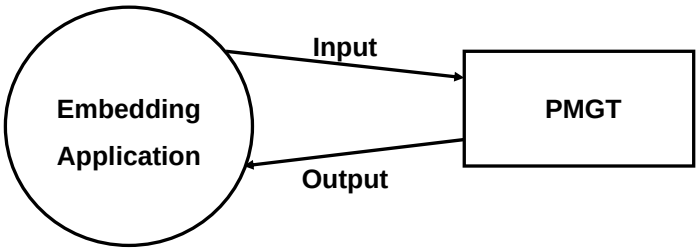

Figure 1: System Context Diagram

Figure 1 shows the context that PMGT will normally fit into. A circle represents an external entity outside the system, an embedding application in this case. The rectangle is the system itself. Arrows represent the data flows between them.

The *input*: $M^{\mathrm{IN}} \times I$

The *output*: M^{OUT}.

PMGT has the following function:

A mesh M^{IN} and some refining/coarsening instructions I are given. PMGT generates a refined/coarsened mesh M^{OUT} according to the instructions I.

3.2 User Characteristics

The target user group of PMGT includes both software designers, who intend to embed this library in their applications, and theoreticians, who are involved in parallel mesh generation. A user of PMGT is expected to be familiar with the notion/knowledge of mesh creation. PMGT is a library used by other applications. Therefore, users should not be novices in terms of software design. The prerequisite software design knowledge are equivalent to that of a senior undergraduate student in science or engineering who took an introductory course on programming. For example, they should be comfortable with compilation of the programming language in which PMGT is written, be familiar with embedding a library in their software, *etc.*

3.3 System Constraints

This system is intended to be built on the Shared Hierarchical Academic Research Computing Network (SHARCNET). SHARCNET is structured as a "cluster of clusters" across South Central Ontario, designed to meet the computational needs of researchers in a diverse number of research areas and to facilitate the development of leading-edge tools for high performance computing (HPC) grids.

Large production clusters, located at the Universities of Western Ontario, Guelph and McMaster, house over 400 HP/Compaq Alpha processors and large symmetric multiprocessor computers. Windsor and Wilfrid Laurier host smaller development clusters (8 processors), which enable researchers to develop and test code before moving to one of the larger clusters. A glance of SHARCNET systems is shown in Table 1. Note that the network is constantly being updated. Detailed information can be found at SHARCNET (Last Access: January, 2006).

4 Specific System Requirements

This section describes the system requirements in detail. After the problem is clearly and unambiguously stated, some solution characteristics are specified.

System	Make	Type	CPUs	OS
bala	Compaq	Cluster	8	Red Hat Linux 7.2
cat	Unknown	Cluster	162	Red Hat Linux 8
goblin	Sun	Cluster	56	Fedora Core 2
hammerhead	Compaq	Cluster	112	Red Hat Linux 7.2
idra	Compaq SC	Cluster	128	Tru64
mako	HP	Cluster	16	Fedora Core 2
tiger	Compaq	Cluster	8	Red Hat Linux 7.2
typhon	Compaq	SMP	16	Tru64
wobbe	Unknown	Cluster	193	Red Hat Linux 8
TOTALS			699	

Table 1: A Glance at the SHARCNET System

Non-functional requirements are also included in this section. The symbol := is used to indicate type definition. The notation for set building and expressions used in this section follows Gries and Schneider (1993). To define the notation, first let x be a list of dummies, t a type, R a predicate, E an expression, $*$ an operator, and P a predicate. Notation $\{x : t \mid R : E\}$ represents a set of values that result from evaluating $E[x := v]$ in the state for each value v in t such that $R[x := v]$ holds in that state. Expression $(*x : t \mid R : P)$ denotes the application of operator $*$ to the values P for all x in t for which range R is true.

4.1 Problem Description

The problems (goals) specified in this subsection represent ideal general models. The problems are simplified by introducing some assumptions, which are listed in Section 4.2.

4.1.1 Background Overview

Many physical problems of importance to scientists and engineers are modeled as a set of Partial Differential Equations (PDEs). In most practical cases, it is necessary to solve the PDEs numerally. Numerical methods to solve PDEs frequently require that the domain of interest be divided into a mesh, which is a set of small, simple elements that cover the computational domain. In some applications, a single mesh is generated and used many times; in this case the processing time spent on mesh construction is not critical and a relatively slow, sequential algorithm suffices (Ruppert, 1993). However, some applica-

tions need adaptive meshing, which requires that the meshes be generated once and then modified many times. For instance, adaptive meshing is used for reliable Finite Element Analysis (FEA) using a posterrori error estimation (Zienkiewicz et al., 2005). The increased mesh interaction for adaptive meshing means an increased need for speed of managing the mesh data which suggest employing parallel processing techniques. Although generating a mesh using multiple processors is complicated, it can offer considerable speed-up over sequential processing. In addition, some FEA applications are implemented on multiple processors. If the adaptive mesh can be generated in multiple processors as well, the mesh data can remain on the local processors. Potentially, time to be used will be significantly reduced.

4.1.2 Goal Statements

There are two related goals for PMGT.

G1: Given a mesh M^{IN} and instructions I on how to refine the mesh, PMGT should generate a refined mesh M^{OUT} according to the instructions I.

G2: Given a mesh M^{IN} and instructions I on how to coarsen the mesh, PMGT should generate a coarsened mesh M^{OUT} according to the instructions I.

4.2 Solution Characteristics Specification

The goals stated in the last section are too general to achieve. In this section, the assumptions are specified first to reduce the scope of the software. Second, the theoretical models for the goals are described. Third, data definitions are given to assist with defining the theoretical models. Finally, the system behaviour is summarized.

4.2.1 Assumptions

A1: PMGT focuses on a 2D domain.

A2: The input and output meshes are bounded.

A3: The input and output meshes are unstructured.

A4: The input and output meshes are conformal.

A5: The elements of input and output meshes are triangles.

A6: The initial mesh is valid.

4.2.2 Theoretical Model

The theoretical models corresponding to the goals given in Section 4.1 describes the relationship between the input mesh (M^{IN}) and the output mesh (M^{OUT}). The meshes are assumed to be embedded in a 2D space.

TM1: Refining Mesh

***Input*:** M^{IN}: `MeshT`, I: `RCinstructionT`
***Output*:** M^{OUT}: `MeshT`

The following behavior is specified:

- *Refined*(M^{OUT}, M^{IN})

That is, the output mesh is a refined version of the input mesh.

TM2: Coarsening Mesh

***Input*:** M^{IN}: `MeshT`, I: `RCinstructionT`
***Output*:** M^{OUT}: `MeshT`

The following behavior is specified:

- *Coarsened*(M^{OUT}, M^{IN})

That is, the output mesh is a coarsened version of the input mesh.

4.2.3 Data Definitions

The data definitions below are organized so that a definition listed in the beginning may be used to define a data item listed after it.

VertexT (D1): A vertex is represented by two real numbers, which are its x coordinate and y coordinate. More formally,
`VertexT` := tuple of ($x : \mathbb{R}$, $y : \mathbb{R}$).

EdgeT (D2): An edge is represented by a set of `VertexT`. More formally,
`EdgeT` := set of `VertexT`.

ValidEdge (D3): An edge is valid if the edge is a line segment (that is, the set has two elements). More formally,
ValidEdge: `EdgeT` $\rightarrow \mathbb{B}$
ValidEdge(e: `EdgeT`) $\equiv \#e = 2$

CellT (D4): A cell is represented by a set of $VertexT$. More formally,
`CellT` := set of `VertexT`

Area (D5): The area of a triangle whose apexes are elements of a cell. More formally,
Area: `CellT` $\rightarrow \mathbb{R}$
Area(c: `CellT`) $\equiv \Sigma v1, v2, v3$: `VertexT` $\mid v1 \in c \wedge v2 \in c \wedge v3 \in c$
$\wedge\, v1 \neq v2 \wedge v2 \neq v3 \wedge v3 \neq v1$:
$\frac{1}{12} * |v1.x * v2.y - v2.x * v1.y +$
$v2.x * v3.y - v3.x * v2.y +$
$v1.x * v3.y - v3.x * v1.y|$

ValidCell (D6): A cell is valid if the cell is a triangle (that is , the set has three elements) and the area of the triangle is greater than zero. More formally,
ValidCell: `CellT` $\rightarrow \mathbb{B}$
ValidCell(c: `CellT`) $\equiv \#c = 3 \wedge$ *Area*(c) ≥ 0

MeshT (D7): A mesh is represented by a set of cells. More formally,
`MeshT` := set of `CellT`.

OnEdge (D8): Checks if a vertex is on the line segment between two vertices (exclusive) of an edge. More formally,
OnEdge: `VertexT` $\times$ `EdgeT` $\rightarrow \mathbb{B}$
OnEdge(v: `VertexT`, e: `EdgeT`) $\equiv \exists\, v1,\ v2$: `VertexT` $\mid$
$v1 \in e \wedge v2 \in e \wedge v1 \neq v2 \wedge v \neq v1 \wedge v \neq v2$:
$(v1.x < v.x \leq v2.x \wedge$
$(v.y - v1.y)/(v.x - v1.x) = (v2.y - v1.y)/(v2.x - v1.x))$

BelongToCell (D9): Checks if an edge belongs to a cell. More formally,
BelongToCell: `VertexT` $\times$ `CellT` $\rightarrow \mathbb{B}$
BelongeToCell(e: `EdgeT`, c: `CellT`) $\equiv \forall\, v$: `VertexT` $\mid v \in e : v \in c$

Inside (D10): Checks if a point (of type VertexT) is inside of a cell. The *inside* checking is false if the point is on an edge of the cell or the point is a vertex of the cell. (The algorithm to check if a point is inside a polygon is from Blackpawn (Last Access: January, 2006).) More formally,

Inside: `VertexT` $\times$ `CellT` $\rightarrow \mathbb{B}$
Inside(v: `VertexT`, c: `CellT`) $\equiv \exists\ v1,\ v2,\ v3$: `VertexT` |
$v1 \in c \wedge v2 \in c \wedge v3 \in c \wedge v1 \neq v2 \wedge v2 \neq v3 \wedge v3 \neq v1$:
$((v.y - v1.y) * (v2.x - v1.x) - (v.x - v1.x) * (v2.y - v1.y)) *$
$((v.y - v2.y) * (v3.x - v2.x) - (v.x - v2.x) * (v3.y - v2.y)) > 0 \wedge$
$((v.y - v2.y) * (v3.x - v2.x) - (v.x - v2.x) * (v3.y - v2.y)) *$
$((v.y - v3.y) * (v1.x - v3.x) - (v.x - v3.x) * (v1.y - v3.y)) > 0$

Vertices (D11): A set of all vetices of the mesh. More formally,
Vertices: `MeshT` $\rightarrow$ set of `VertexT`
Vertices(m: `MeshT`) $\equiv \{v$: `VertexT` | $(\forall\ c$: `CellT` | $c\ \in m : v\ \in c) : v\}$

Edges (D12): A set of all edges of the mesh. More formally,
Edges: `MeshT` $\rightarrow$ set of *EdgeT*
Edges(m: `MeshT`) $\equiv \{v1,\ v2$: `VertexT` | $(\forall c$: `CellT` | $c \in m :$
$v1 \in c \wedge v2 \in c \wedge v1 \neq v2) : \{v1, v2\}\}$

BoundaryEdges (D13): A set of edges are boundary edges if they form a boundary of a mesh. More formally,
BoundaryEdges: `MeshT` $\rightarrow$ set of *EdgeT*
BoundaryEdges(m: `MeshT`) $\equiv \{b$: `EdgeT` | $b \in$ *Edges*(m) $\wedge$
$(\#\{c$: `CellT` | $c \in m \wedge$ *BelongToCell*($b,\ c$): $c\} = 1)$: $b\}$

BoundaryVertices (D14): A set of boundary vertices of the mesh. More formally,
BoundaryVertices: `MeshT` $\rightarrow$ set of `VertexT`
BoundaryVertices(m: `MeshT`) $\equiv$
$\{v$: `VertexT` | $v \in$ *BoundaryEdges*(m): $v\}$

Bounded (D15): A mesh is bounded if the boundary edges form a closed polygon(all vertices of boundary edges belong to exactly two boundary edges). More formally,
Bounded: `MeshT` $\rightarrow \mathbb{B}$
Bounded(m: `MeshT`) $\equiv \forall v$: `VertexT` | $v \in$ *BoundaryVertices*(m):
$(\#\{e$: `EdgeT` | $e \in$ *BoundaryEdge*(m) $\wedge v \in e : e\} = 2)$

Conformal (D16): In 2D, a mesh is conformal if the intersection of any two cells is either a vertex or an edge or empty. More formally,
Conformal: `MeshT` $\rightarrow \mathbb{B}$
Conformal(m: `MeshT`) $\equiv \forall c1, c2$: `CellT` | $c1 \in m \wedge c2 \in m \wedge c1 \neq c2$:
$(\exists e$: `EdgeT` | $e \in$ *Eeges*(m): $(\exists v$: `VertexT` | $v \in$ *Vertices*(m):
$(c1 \cap c2 = e \vee c1 \cap c2 = v \vee c1 \cap c2 = \emptyset) \wedge (\neg$ *OnEdge(v, e))))*

NoInteriorIntersect (D17): NoInteriorIntersect is true if a point in space (of type VertexT) is inside only one cell of the mesh. More formally,
NoInteriorIntersect: `MeshT` $\rightarrow \mathbb{B}$
NoInteriorIntersect(m: `MeshT`) $\equiv \forall c1, c2$: `CellT` |
$c1 \in m \wedge c2 \in m \wedge c1 \neq c2 : (\forall v$: `VertexT` | *Inside(v, c1)*: $\neg$ *Inside(v,c2))*

ValidMesh (D18): A mesh is valid if the mesh is bounded, conformal, and any point is only inside one cell. More formally,
ValidMesh: `MeshT` $\rightarrow \mathbb{B}$
ValidMesh(m: `MeshT`) $\equiv$ ($\forall e$: `EdgeT` | $e \in$ *Edges(m): ValidEdge(e))*
$\wedge$ ($\forall c$: `CellT` | $c \in m$: *ValidCell(c))* $\wedge$
Bounded(m) $\wedge$ *Conformal(m)* $\wedge$ *NoInteriorIntersect(m)*

CoveringUp (D19): True if two meshes covering up each other, that is, if all endpoints of the boundary edges of one mesh are on the boundary edges or are end points of the boundary edges of another mesh. More formally,
CoveringUp: `MeshT` $\times$ `MeshT` $\rightarrow \mathbb{B}$
CoveringUp($m1$, $m2$: `MeshT`) $\equiv \forall v1$, $v2$: `VertexT`, |
$v1 \in$ *BoundaryVertice(m1)* $\wedge$ $v2 \in$ *BoundaryVertices(m2)*:
($\exists b1$, $b2$: `EdgeT` | $b1 \in$ *BoundaryEdges(m1)* $\wedge b2 \in$ *BoundaryEdges(m2):*
(OnEdge(v1, b2) $\vee v1 \in b2) \wedge$ *(OnEdge(v2, b1)* $\vee v2 \in b1$))

InstructionT (D20): The type of instructions is defined as:
`InstructionT` := {`REFINE, COARSEN, NOCHANGE`}

CellInstructionT (D21): The type of instructions on a cell is defined as:
`CellInstructionT`:= tuple of (*cell*: `CellT`, *instr*: `InstructionT`)
(For each cell, there is an instruction for refining, coarsening, or nochange.)

RCinstructionT (D22): The type of instructions on a mesh is defined as:
`RCinstructionT`:= tuple of (*rORc*: `InstructionT`, *cInstr*: set of `CellInstructionT`)
(For each mesh, there is an instruction on whole mesh, and there are set of instruction on each cell.)

Refined (D23): True if a mesh M' is a refined mesh of a mesh M. More formally
Refined: `MeshT` $\times$ `MeshT` $\times$ `RCinstructionT` $\rightarrow \mathbb{B}$
Refined(*m', m*: `MeshT`, *rc*: `RCinstructionT`) $\equiv$
rc.rORc = `REFINE` $\wedge$ *ValidMesh(m)* $\wedge$ *ValidMesh(m')* $\wedge$
CoveringUp(m', m) $\wedge \#m' \geq \#m$

Coarsened (D24): True if a mesh M' is a coarsened mesh of a mesh M. More formally
Coarsened: `MeshT` $\times$ `MeshT` $\times$ `RCinstructionT` $\rightarrow \mathbb{B}$
Coarsened(*m'*, *m*: `MeshT`, *rc*: `RCinstructionT`) $\equiv$
rc.rORc = `COARSEN` $\wedge$ *ValidMesh(m)* $\wedge$ *ValidMesh(m')* $\wedge$
CoveringUp(m', m) $\wedge \#m' \leq \#m$

4.2.4 System Behaviour

System Behaviour, shown through functional requirements, defines what the software should do. The functional requirements, as well as nonfunctional requirements in Section 4.3, partially come from Smith and Chen (2004). Smith and Chen (2004) listed all requirements that are common for mesh generation systems. They also considered the difference between meshes in term of variabilities. However, the mesh generations analyzed by Smith and Chen (2004) are targeted at full FEA applications. PMGT only manages the geometric information about the mesh, not other FEA related information, such as boundary condition and material property. Hence, only commonalities that is meaningful for PMGT are selected. Variabilities with parameters of variation that are suitable for PMGT are also considered. Other part of the requirements are obtained from Dr. Smith.

New functional requirements, *RCInstruction* (F9) and *Help* (F16) are added. F9 is unique to PMGT and F16 facilities the non-functional requirements *Usability* (N6).

We specify both functional requirements and non-functional requirements in the tables. In each table, the field *Description* gives a brief description of this requirement. It tells what PMGT should do to fulfill this requirement. There are two potential sources, shown in the *Source* field, for each requirement. One source is from Smith and Chen (2004), and the other comes from Dr. Smith. If the requirement is from Smith and Chen (2004), then this field will show the commonality number, with a prefix C and the associated variability, shown by a prefix V. Where applicable, *Related Data Definitions* and *Related Theoretical Models* gives the numbers of related data definitions and the numbers of related theoretical models, respectively. These two field only appear for functional requirements. The *Binding Time* field either shows scope time or run time. *Scope time* means that this requirement is determined when the SRS is written. *Run time* means that this requirement is determined when the system is running. *History* records the time of creating and changing of the requirements.

Requirements Number	F1
Requirements Name	RefiningMesh
Description	PMGT should have capabilities for refining an existing mesh. $I.rORc =$ `REFINE` $\wedge$ *Refined*(M^{OUT}, M^{IN})
Source	C1, V3
Related Data Definitions	D20, D22, D23
Related Theoretical Models	TM1
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2005. Change the name from "ImprovingMesh" to "RefiningMesh". Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F2
Requirements Name	CoarseningMesh
Description	PMGT should have capabilities for coarsening an existing mesh. $I.rORc =$ `COARSEN` $\wedge$ *Coarsened*(M^{OUT}, M^{IN})
Source	C1, V3
Related Data Definitions	D20, D22, D24
Related Theoretical Models	TM2
Binding Time	Scope time
History	Created – October, 2006.

Requirements Number	F3
Requirements Name	RefiningOrCoarsening
Description	PMGT can either refine a given mesh to a refined mesh, or coarsen a mesh to a coarsened mesh. However, PMGT cannot do both refining and coarsening at the same time.
Source	C1, V3
Related Data Definitions	D20, D22, D23, D24
Related Theoretical Models	TM1, TM2
Binding Time	Run time
History	Created – October, 2006.

Requirements Number	F4
Requirements Name	MeshType
Description	The mesh generated by PMGT is unstructured.
Source	C1, V6
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F5
Requirements Name	ElmShape
Description	The shape of the elements in both input and output meshes are triangles. $\forall c1, c2$: `CellT` | $c1 \in M^{\text{IN}} \wedge c2 \in M^{\text{OUT}}$: $\#c1 = 3 \wedge Area(c1) > 0 \wedge$ $\#c2 = 3 \wedge Area(c2) > 0$
Source	C1, V9
Related Data Definitions	D4, D5
Related Theoretical Models	TM1, TM2
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Take out the requirement for generating quadrilateral meshes. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F6
Requirements Name	DomainDimension
Description	The computational domain is in 2D space.
Source	C1, V13
Related Data Definitions	N/A
Related Theoretical Models	TM1, TM2
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F7
Requirements Name	Conformal
Description	Both input and output meshes are conformal. $Conformal(M^{\mathrm{IN}}) \wedge Conformal(M^{\mathrm{OUT}})$
Source	C1, V18
Related Data Definitions	D16
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Take out the requirement for generating non-conformal meshes. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F8
Requirements Name	InputDefinition
Description	The input of PMGT should be provided by the embedding application.
Source	C8
Related Data Definitions	D7, D20, D22
Related Theoretical Models	TM1, TM2
Binding Time	Scope time
History	Created – June, 2005. Modified–October 2005. Change the name from "Input" to "InputDefinition" to clarify that this requirements is about the source of the input. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F9
Requirements Name	RCInstruction
Description	The Instruction on how to refine/coarsen a mesh includes the instruction of whether to refine or coarsen the mesh and an individual instruction for each element of the the mesh to indicate refining, coarsening, or no change.
Source	Dr. Smith
Related Data Definitions	D20, D21, D22
Related Theoretical Models	TM1, TM2
Binding Time	Scope time
History	Created – June, 2005 Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F10
Requirements Name	OutputStorage
Description	The output of PMGT is stored in memory or in files or in both memory and files.
Source	C12, Dr. Smith
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Run time
History	Created – June, 2005. Modified – October 2005. Change the name from "Output" to "OutputStorage" to clarify that this requirements is about the storage of the output. Modified – October 2006. Add the requirement of storing the output mesh in files or both memory and files. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F11
Requirements Name	VertexUniqueID
Description	Each vertex in the output file has a unique identifier.
Source	C2
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F12
Requirements Name	ElmUniqueID
Description	Each element in the output file has a unique identifier.
Source	C3
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F13
Requirements Name	ElmTopology
Description	The topology of an element in the output file is given by the connectivity of its set of vertices.
Source	C4
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F14
Requirements Name	OutElmOrder
Description	The element information in output files is listed in ascending order.
Source	C13, V34
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F15
Requirements Name	OutVertexOrder
Description	The vertex information, such as the coordinates, in output files is listed in ascending order.
Source	C14, V35
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

Requirements Number	F16
Requirements Name	Help
Description	Helps on documenting the interface and the functionality of each function should be provided.
Source	Dr. Smith
Related Data Definitions	N/A
Related Theoretical Models	N/A
Binding Time	Scope time
History	Created – June, 2005. Modified – October 2005. Add the requirement of documenting functionality of each function. Modified – October, 2006. Field for "Related Data Definitions" and "Related Theoretical Models" are added.

4.3 Non-functional Requirements

All non-functional requirements listed in Smith and Chen (2004) are selected except for C16, which is solution tolerance, since a mesh refined/coarsened by different algorithms may have different solutions, but all of these solutions can still be valid. All potential output meshes are valid as long as the output meshes are covering/covered up meshes of the original mesh, and they are refined/coarsened according to the RCInstruction. The resulting mesh is difficult to measure in terms of *solution tolerance*. Three new non-functional requirements, which are *LookAndFeel* (N5), *Usability* (N6), and *Maintainability* (N7), are added. These requirements are mentioned in Lai (2004).

PMGT is difficult to validate. One reason is that the solution for refining/coarsening a mesh is unknown, as mentioned above. The other reason is that it is difficult to write validatable requirements, especially for nonfunctional requirements. For example, what is the proper way for specifying the requirement of *Usability* (N6) of PMGT? On the one hand, that *the software should easy to use* is not validatable. On the other hand, that *a person should be able to use the software in two days* is validatable. However, the measurement, *two days*, often lacks a justifiable rationale.

The approach to validate this kind of requirements are to compare it with other software with similar functionality. Phrases that are in italics and capitalized, such as *MANPROP*, represent constant defined in Section 8. Usu-

ally, these constants come from other applications with similar functionalities. For example, the *Usability* requirement of PMGT is presented as follows:

> This system should be easy to use. Users with the background specified in Section 3.2 should take *LEARNTIME* to reproduce an example mesh, which is specified by the test case TC**??** in the Appendix **??**.

First, more general requirement is given. Then, a suggestion to reproduce an example mesh is specified. The constant *LEARNTIME* is defined as the time to produce the same mesh for users with the same background using AOMD.

Requirements Number	N1
Requirements Name	Performance
Description	Refining/coarsening a mesh using multiple processors should be faster than when using a single processor. In addition, the performance of PMGT should be comparable with that of similar applications. The execution time to refine an example mesh, which is specified by the test case TC**??** in the Appendix **??**.
Source	C15, V39
Binding Time	Scope time
History	Created – June, 2005.

Requirements Number	N2
Requirements Name	Precision
Description	The number of decimal digits should agree with the IEEE standard for floating-point numbers.
Source	C17, V41
Binding Time	Scope time
History	Created – June, 2005.

Requirements Number	N3
Requirements Name	Exception
Description	Run-time exception handling should check at least the following exceptions: division by zero, redundant vertices, redundant edges, redundant cells.
Source	C18, V42
Binding Time	Scope time
History	Created – June, 2005

Requirements Number	N4
Requirements Name	Portability
Description	PMGT should build on a platform with access to SHARCNET or on a the system that has similar architecture to SHARCNET. The memory capacity should be *MEMCAP*.
Source	C19, V37, V38
Binding Time	Scope time
History	Created – June, 2005

Requirements Number	N5
Requirements Name	LookAndFeel
Description	PMGT should follow the programming conventions of the language in which the application is coded in.
Source	Dr. Smith
Binding Time	Scope time
History	Created – June, 2005

Requirements Number	N6
Requirements Name	Usability
Description	This system should be easy to use. Users with the background specified in Section 3.2 should take *LEARNTIME* to reproduce an example mesh, which is specified in the Appendex **??**.
Source	Dr. Smith
Binding Time	Scope time
History	Created – June, 2005

Requirements Number	N7
Requirements Name	Maintainability
Description	The system should be developed in the way that the effort spent to maintain the system or to add in features would be minimum. The redevelopment time to add a new algorithm to coarsen meshes in PMGT should be *MANPROP*.
Source	Dr. Smith
Binding Time	Scope time
History	Created – June, 2005

5 Other System Issues

This section includes some other supporting information that might contribute to the success or failure of the system development. The following factors are considered:

- Open issues are statements of factors that are uncertain and might make significant difference to the system.
- Off-the-shell solutions are existing systems and/or components bought or borrowed. They could be the potential solutions.
- Waiting rooms provide a blueprint of how the system will be extended.

5.1 Open Issues

There are no open issues for PMGT at this stage.

5.2 Off-the-shelf Solutions

The following programs may be used in PMGT.

- AOMD: a mesh management library (or database) that is able to provide a variety of services for mesh users (SCOREC, Last Access: January, 2006).

5.3 Waiting Rooms

Here, we list the possible changes that can affect the extension of the system. These changes are related to the assumptions specified in Section 4.2.

1. PMGT may produce both structured and unstructured meshes.
2. PMGT may produce both conformal and nonconformal meshes.
3. The elements of input and output mesh may be of a shape other than triangles.
4. The system may deal with invalid input mesh.
5. The system may accommodate a mixed mesh.
6. The system may accommodate a hybrid mesh.
7. The system may deal with a 3D problem domain.

6 Traceability Matrix

The traceability matrix defined in this section gives a big picture of the associations among goals, assumptions, data definitions, theoretical models, and functional requirements. Goals are ideal general models. After assumptions are applied, these goals are restricted to problems that can be solved by PMGT. Data definitions and theoretical models are used to describe the requirements. The matrix is too big to fit one page. For the sake of clarity, it is split into three parts in five tables, which are Table 2, Table 3, Table 4, Table 5, and Table 6. In addition, only items that have a relation with items in the same

part are listed. If there is a ✓in a cell, it means that if the goal, or the assumption, or the theoretical model, or the data definition, or the requirement in the corresponding column changes, the assumption, or the data definition, or the theoretical model, or the requirement in the corresponding row should also change.

7 List of Possible Changes in the Requirements

The system might evolve to accommodate the following changes in the future. These changes will add additional goals to the software library.

1. The input of PMGT may include material properties.

2. The input of PMGT may include boundary conditions.

8 Values of Auxiliary Constants

The constants given in this section are used to validate some nonfunctional requirements. The compatible software chosen is AOMD. However, other software can also be used as long as the other software has the required functionalities to validate the given requirement.

LEARNTIME	The time that reproduce the same example as that specified in nonfunctional requirement N6 using AOMD.
MANPROP	The redevelopment time to add the same algorithm as that specified in the nonfunctional requirement N7, using AOMD. If the algorithm is already in AOMD, the the time that AOMD took to add it.
RSPTIME	The execution time to refine the same mesh as that specified in nonfunctional requirement N1 using AOMD.
MEMCAP	The typical memory capacity of a machine on SHARCNET.

	G1	G2	A1	A2	A3	A4	A5	A6	TM1	TM2
A1	✓	✓	✓							
A2	✓	✓		✓						
A3	✓	✓			✓					
A4	✓	✓				✓				
A5	✓	✓	✓				✓			
A6	✓	✓						✓		
D1	✓	✓	✓							
D2	✓	✓								
D3	✓	✓								
D4	✓	✓	✓				✓			
D5	✓	✓	✓				✓			
D6	✓	✓	✓				✓			
D7	✓	✓								
D8	✓	✓	✓							
D9	✓	✓	✓							
D10	✓	✓	✓				✓			
D11	✓	✓								
D12	✓	✓	✓					✓		
D13	✓	✓	✓							
D14	✓	✓	✓							
D15	✓	✓	✓	✓						
D16	✓	✓	✓			✓				
D17	✓	✓	✓	✓						
D18	✓	✓								
D20	✓	✓								
D21	✓	✓								
D22	✓	✓								
D19	✓	✓	✓	✓						
D23	✓									
D24		✓								
TM1	✓		✓						✓	
TM2		✓	✓							✓

Table 2: Traceability Matrix (PART I): Goals, Assumptions, Theoretical Models, Data Definitions, and Requirements (I)

References

Blackpawn. Point in triangle test, Last Access: January, 2006. URL `http://www.blackpawn.com/texts/pointinpoly/default.html`.

	G1	G2	A1	A2	A3	A4	A5	A6	TM1	TM2
F1	✓								✓	
F2		✓								✓
F3	✓	✓							✓	✓
F4					✓					
F5	✓	✓	✓				✓		✓	✓
F6	✓	✓	✓						✓	✓
F7	✓	✓				✓				
F8	✓	✓							✓	✓
F9	✓	✓							✓	✓
F10	✓	✓								
F16	✓	✓								

Table 3: Traceability Matrix (PART I): Goals, Assumptions, Theoretical Models, Data Definitions, and Requirements (II)

Alan M. Davis. *Software Refquirements: Analysis and Specification*. Prentice Hall Inc., 1990.

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

David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math*. Springer-Verlag New Yourk, Inc., 1993.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Computer Society, Washington, DC, USA, 1990.

IEEE. *IEEE Guide for Developing System Requirements Specifications*. IEEE Computer Society, Washington, DC, USA, 1998.

IEEE. *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society, Washington, DC, USA, 2nd edition, 2000.

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

James Robertson and Suzanne Robertson. Volere requirements specification template, 2001.

Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 83–92, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0-89871-313-7.

SCOREC. Algorithm oriented mesh database, Last Access: January, 2006. URL `http://www.scorec.rpi.edu/AOMD/`.

SHARCNET. Shared hierarchical academic research computing network, Last Access: January, 2006. URL `www.sharcnet.ca`.

S. Smith and C. H. Chen. Commonality analysis for mesh generation system. Technical Report CAS-04-10-ss, Department of Computing and Software, McMaster University, 2004.

Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, 1992.

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

O. C. Zienkiewicz, R. L Taylor, and J. Z. Zhu. *The Finite Element Method Its Basis and Fundamentals*. Elsevier Butterworth-Heinemann, 6th edition, 2005.

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
D1	✓											
D2	✓	✓										
D3		✓	✓									
D4	✓			✓								
D5	✓				✓							
D6				✓	✓	✓						
D7				✓			✓					
D8	✓	✓						✓				
D9	✓	✓		✓					✓			
D10	✓			✓						✓		
D11	✓			✓			✓				✓	
D12	✓	✓		✓			✓					✓
D13		✓		✓					✓			✓
D14	✓						✓					
D15	✓	✓					✓					
D16	✓	✓		✓			✓				✓	✓
D17	✓			✓			✓			✓		
D18			✓			✓	✓					
D19	✓	✓					✓	✓				
D20												
D21				✓								
D22												
D23												
D24												
TM1							✓					
TM2							✓					
F1												
F2												
F3												✓
F5				✓		✓						
F7												
F8							✓					
F9												

Table 4: Traceability Matrix (PART II): Data Definitions and Requirements (I)

	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	D23	D24
D13	✓											
D14	✓	✓										
D15	✓	✓	✓									
D16				✓								
D17					✓							
D18			✓	✓	✓	✓						
D19	✓	✓					✓					
D20								✓				
D21								✓	✓			
D22								✓	✓	✓		
D23						✓	✓	✓		✓	✓	
D24						✓	✓	✓		✓		✓
TM1										✓	✓	
TM2										✓		✓
F1								✓		✓	✓	
F2								✓		✓		✓
F3								✓		✓	✓	✓
F5												
F7				✓								
F8								✓		✓		
F9								✓	✓	✓		

Table 5: Traceability Matrix (PART II): Data Definitions and Requirements (II)

	F1	F2	F6	F8	F10	N6
F3	✓	✓				
F5			✓			
F9				✓		
F11					✓	
F12					✓	
F13					✓	
F14					✓	
F15					✓	
F16						✓

Table 6: Traceability Matrix (PART III): Requirements