Anais das XXXII Jornadas Sulamericanas de Engenharia Estrutural
Maio / 2006 ISBN 85-99956-01-9
Investigações Teóricas, Experimentais, Numéricas e Computacionais
Trabalho JOR0577 - p. 3261-3271

SISTEMA GRÁFICO INTERATIVO PARA MODELOS BIDIMENSIONAIS DO MÉTODO DOS ELEMENTOS FINITOS

F. T. Fonseca [1], M. L. Almeida [1], M. B. Gonçalves [1], R. L. Pitangueira [1]

[1] UFMG – Escola de Engenharia, Departamento de Engenharia de Estruturas, Avenida do Contorno 842, CEP 30110-060 – Belo Horizonte - MG – Brasil - flaviotf@dees.ufmg.br, mla@dees.ufmg.br, marco@dees.ufmg.br, roque@dees.ufmg.br

RESUMO

O artigo apresenta o resultado do trabalho colaborativo de quatro integrantes da equipe de desenvolvimento do INSANE (*Interactive Structural Analysis Environment*): um sistema computacional para modelos discretos de análise estrutural do método dos elementos finitos (MEF). A concepção do INSANE, como um conjunto de segmentos de aplicação, é apresentada, destacando-se as vantagens deste conceito. A persistência dos dados compartilhados entre os vários segmentos do sistema também é discutida, ressaltando-se o uso de arquivos XML ou objetos Java para criação dos protocolos de comunicação. O trabalho de unificação de três implementações inicialmente independentes (sistema gráfico interativo para modelos reticulados, gerador de malhas bidimensionais de elementos finitos e processador de modelos do MEF baseado na formulação paramétrica) é apresentado, discutindo-se as soluções adotadas. Nesta discussão apresentam-se as características de cada um dos três trabalhos, através de diagramas UML apropriados de maneira a caracterizar a proposta de unificação. O projeto orientado a objetos do novo sistema é detalhado, ressaltando-se a adoção do padrão Model-View-Controller no trabalho de unificação. Finalmente, os diversos recursos disponibilizados são apresentados através de exemplos e as possibilidades de novas expansões, sem repetir o trabalhoso processo de unificação, são discutidas.

Palavras-chave: Método de Elementos Finitos, Computação Gráfica, Programação Orientada a Objetos.

1 INTRODUÇÃO

Para disponibilização de modelos discretos de análise estrutural em computadores, utilizam-se programas de pré-processamento (para a criação dos modelos com recursos gráficos interativos), processamento (para a montagem e resolução numérica do modelo) e pós-processamento (para visualização gráfica de resultados). As possibilidades que os recursos tecnológicos para desenvolvimento de software oferecem para cada uma destas três etapas constituem amplo campo de pesquisa na área de métodos numéricos e computacionais aplicados à engenharia. O domínio destes recursos e a aplicação dos mesmos no aprimoramento progressivo dos modelos, sem ter que recomeçar o processo a cada novo aperfeiçoamento, requer um ambiente computacional segmentado, amigável a mudanças e escalável em complexidade.

O projeto INSANE ("*Interactive Structural Analysis Environment*") objetiva desenvolver um sistema computacional com estas características. Para isto, o ambiente possui três grandes segmentos (pré-processador, processador e pós-processador). Os pré e pós-processadores são aplicações gráficas interativas, implementadas na linguagem Java, que disponibiliza recursos diversos para diferentes modelos discretos. O processador é uma aplicação, também implementada em Java, que representa o núcleo numérico do sistema. Este núcleo é responsável pela obtenção dos resultados de diferentes modelos discretos de análise estrutural. A persistência dos dados compartilhados pelas três aplicações é alcançada através de uma interface baseada em arquivo(s) XML e/ou objetos Java.

2 O PROJETO INSANE

A figura 1 mostra a combinação da arquitetura em camadas e padrões de projeto de software adotados para o INSANE. Como pode ser visto na figura, a versão atual do sistema possui quatro camadas lógicas e duas camadas físicas. Três das camadas lógicas do sistema foram definidas utilizando-se o padrão de projeto de software [4] denominado `Modelo-Vista-Controlador` (MVC). Este padrão é bastante apropriado uma vez que preconiza a separação do processamento da informação de sua representação gráfica [9], facilitando assim os trabalhos de expansão e manutenção da aplicação. A quarta camada lógica é a camada de persistência.

Em termos físicos, o INSANE possui atualmente somente duas camadas: uma aplicação carregada na memória do computador (compreendendo as camadas lógicas Modelo, Vista e Controlador) e arquivos textos e/ou binários persistidos em disco.

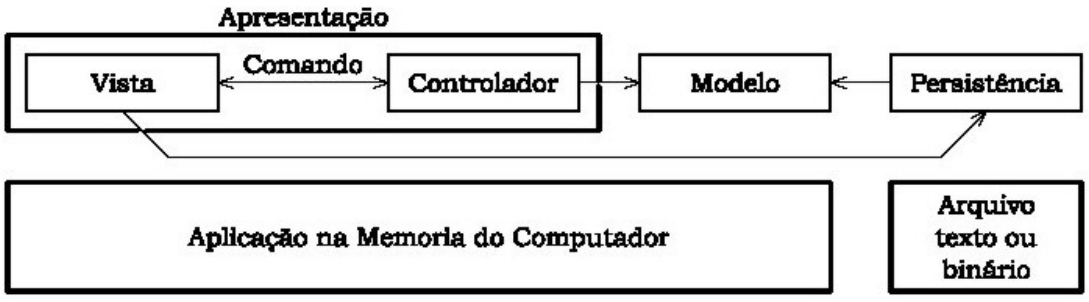


Figura 1. Arquitetura em Camadas e Padrões de Projeto adotados no INSANE.

O inter-relacionamento entre as camadas é conseguido, principalmente, através da implementação do padrão de projeto de software denominado `Comando` [4]. A figura 2 exemplifica este relacionamento para o caso da tarefa de adição de uma entidade geométrica ao modelo corrente e sua visualização. Como pode ser visto na figura, o fluxo de informações para realização de tal tarefa ocorre em quatro etapas. Na primeira etapa, o objeto `Comando`, responsável pela tarefa, aciona o `Controlador` ativo, informando a requisição. A seguir, etapa 2, o `Controlador` cria o objeto correspondente à entidade geométrica e o adiciona ao `Modelo` pertinente. Na etapa 3, o `Controlador` cria objetos de desenho representativos dos objetos do `Modelo`. Finalmente, na etapa 4, os objetos de desenho pertencentes ao `Controlador` são apresentados na área de desenho da `Vista`.

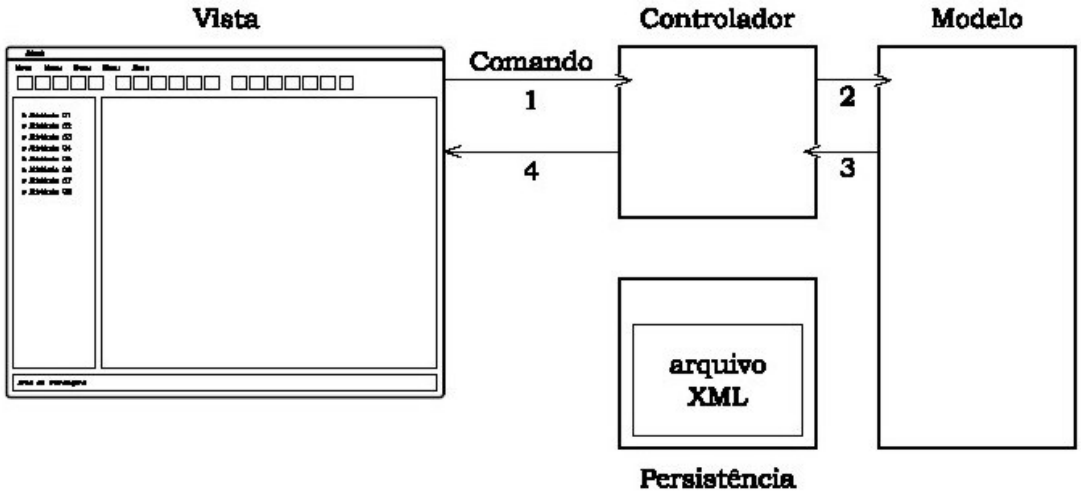


Figura 2. Relacionamento entre camadas do INSANE.

3 O PRÉ-PROCESSADOR

O primeiro gerador de malhas disponibilizado no INSANE é baseado em mapeamentos transfinitos. Conforme detalhadamente descrito em Fonseca [3], o mapeamento transfinito estabelece sistemas de coordenadas curvilíneas definidos pelo contorno de domínios arbitrários. Este método descreve uma superfície aproximada que coincide com a superfície real ou idealizada em um número não enumerável de pontos, propriedade que dá nome ao método. O método é capaz de modelar exatamente o contorno de superfícies sem a introdução de nenhum erro na geometria do mesmo.

As técnicas de mapeamentos transfinitos consistem em se obter valores das coordenadas dos pontos no interior do domínio, a partir das funções que descrevem suas fronteiras, através do uso de projetores. Neste trabalho são utilizados os projetores Lofting, Bilinear e Trilinear [5].

Existem duas alternativas para representação do contorno de regiões: a forma contínua e a forma discreta. A forma contínua representa o vetor posição de uma curva do contorno como função de alguma coordenada paramétrica. A representação discreta do mapeamento transfinito consiste em listas finitas de pontos localizados na curva, com uma única coordenada associada com cada ponto da lista. Uma posição na curva pode somente ser avaliada nos pontos contidos na lista e é indefinida nos demais pontos. A forma discreta de representação é inteiramente geral e pode ser usada em curvas de

qualquer forma. Curvas complexas podem ser criadas pela concatenação de uma série de curvas simples sem acréscimo de complexidade nas rotinas de mapeamento. Para evitar que uma grande quantidade de dados tenha que ser informada para descrever uma curva, pode-se usar um gerador de curvas que cria automaticamente descrições discretas de curvas baseadas em vários modelos contínuos [5].

Os algoritmos de geração de malhas implementados neste trabalho disponibilizam elementos Serendípticos quadrilaterais (com 4n nós, $n \in \mathrm{N}$) e triangulares (com 3n nós, $n \in \mathrm{N}$) e elementos Lagrangeanos quadrilaterais (com n^2 nós, $n \in \mathrm{N}$) e triangulares (com $\sum_1^n i$ nós, $n \in \mathrm{N}$, n>1).

A interface do pré-processador possui uma estrutura em árvore que é responsável por ativar o Modelo, a Vista e o Controlador adequados à cada atividade, representada pelos nós desta árvore (Figura 3). Ao ativar o primeiro nó da árvore (figuras 3(a) e 3(b)), a interface torna ativo o Controlador Geométrico, o Modelo Geométrico e prepara a Vista para receber as requisições do usuário, de maneira a cumprir a primeira atividade, a definição geométrica das regiões. Esta atividade consiste em descrever o contorno das regiões onde se deseja gerar as malhas. Para cumprir essa etapa, a interface disponibiliza ao usuário uma barra de ferramenta que permite criar as primitivas (pontos, linhas, arcos, splines, etc), com as quais os contornos das regiões serão definidos.

A segunda etapa (Figura 3(c)) consiste em agrupar primitivas para gerar as curvas do contorno que definem as regiões onde as malhas serão geradas. Para essa etapa é ativado um novo par Vista-Controlador, a interface disponibiliza uma nova barra de ferramentas com botões sensíveis ao estado atual do Modelo e do Controlador. O Controlador ativo passa a ser o Controlador de Sub-regiões. A Vista, no estado atual, é capaz de gerar uma curva formada pelo grupo de primitivas selecionadas. Quando a segunda curva é gerada, ativa-se um botão que disponibiliza a opção de gerar uma região Lofting composta de duas curvas abertas definida pelas duas curvas recém geradas, tem-se ainda a opção de gerar uma terceira curva. Optando pela criação da terceira curva, é ativado o botão para geração de uma região Trilinear definida pelas três curvas, existindo ainda a opção de gerar uma quarta curva. Optando pela criação da quarta curva, é ativado o botão para geração de uma região Bilinear definida pelas quatro curvas.

A terceira etapa (Figura 3(d)) consiste em definir o número de divisões de cada primitiva que compõe as curvas. Para esta etapa é ativado o terceiro par Vista-Controlador. A Vista recebe uma nova barra de ferramentas e um Controlador de Partição é ativado.

Na quarta etapa são geradas as malhas (figuras 3(e) e 3(f)). Para formar o quarto par Vista-Controlador, a Vista recebe nova barra de ferramentas e é ativado um Controlador de Mapeamento. Os nós e elementos gerados nesta etapa são armazenados em um `modelo` de elementos finitos, que se torna ativo na próxima etapa.

A quinta etapa do pré-processador é o Editor de Atributos. Neste módulo o usuário pode especificar o material, as restrições nodais e o carregamento.

Em qualquer etapa, durante o processo de geração dos dados necessários à análise via método dos elementos finitos, o `Modelo` pode ser armazenado como um objeto Java. O seu estado atual será preservado e o pré-processamento poderá ser retomado no momento adequado.

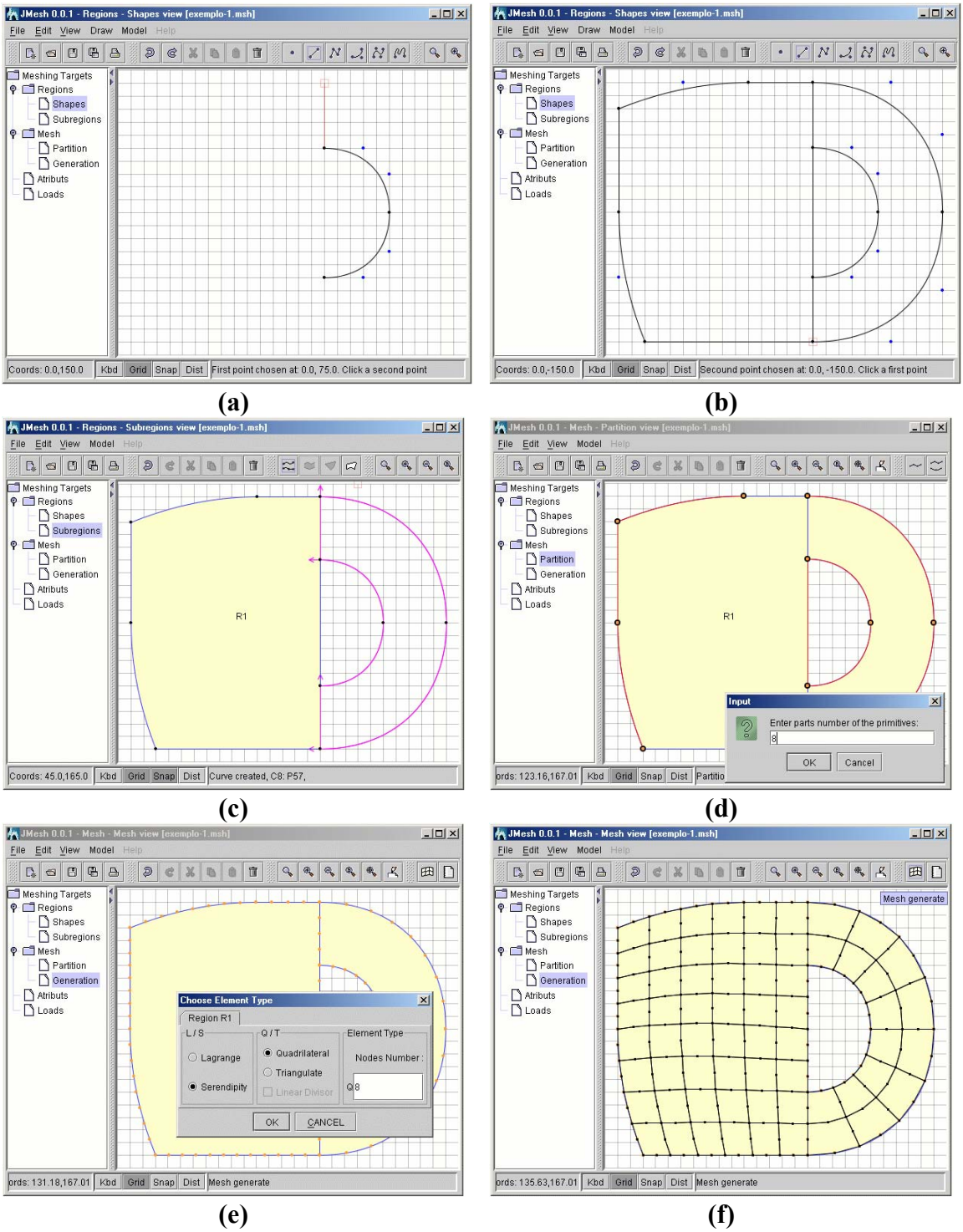


Figura 3. Funcionamento do Pré-Processador.

4 O NÚCLEO NUMÉRICO

Pode-se creditar à formulação paramétrica do MEF o seu grande desenvolvimento e aceitação como ferramenta de engenharia. Esta formulação permite que as entidades matemáticas do modelo discreto possam ser calculadas de uma única

maneira, quaisquer que sejam as características físicas, geométricas, de material e de carregamento do modelo.

Este potencial de generalização da formulação paramétrica do MEF a torna propícia ao paradigma de programação orientada a objetos (POO), conforme discussão a seguir.

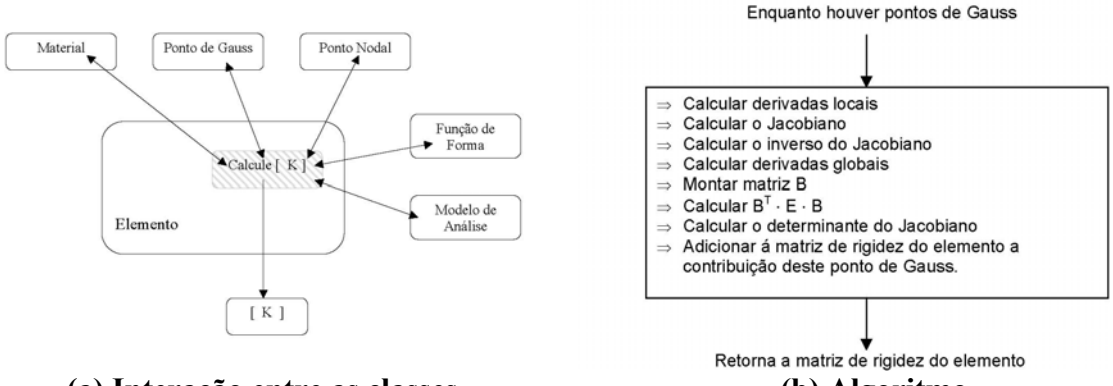
O conceito mais significativo do Método dos Elementos Finitos é, como o próprio nome sugere, o conceito de elemento. Assim é natural a existência da *classe elemento*. Lembrando-se que um elemento possui pontos nodais, um material e funções de forma, pode-se evoluir a análise, criando-se outras classes correlatas. Entretanto, para maior clareza e enriquecimento da análise, é relevante referir-se à Eq. 1, que permite calcular a matriz de rigidez de um elemento finito.

$$[K]^e = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1}[B]^T [E][B]\underline{J}|d\xi \cdot d\eta \cdot d\zeta \qquad (1)$$

A equação (1) revela que a obtenção da matriz de rigidez de um elemento depende de propriedades do material (para montagem da matriz [E]) e de derivadas das funções de forma (para montagem da matriz [B]) que, por sua vez, dependem dos pontos nodais, justificando assim a criação das *classes material, função de forma e ponto nodal*.

A equação (1) também mostra a necessidade de integração nas coordenadas adimensionais ξ, η e ζ. Lembrando que a quadratura de Gauss é a técnica mais empregada para proceder a referida integração e que a mesma se baseia na existência de pontos de integração internos ao elemento e pesos a estes associados, é razoável criar-se a *classe ponto de integração*.

Outro aspecto relevante, não tão explícito na Eq. (1), é o processo de montagem das matrizes [E] e [B]. Os tamanhos destas matrizes, bem como o arranjo dos parâmetros do material e das derivadas das funções de forma para a formação das mesmas dependem do modelo de análise do elemento finito. Diferentes arranjos das propriedades do material originam diferentes matrizes [E], se o modelo de análise é de estado plano de tensão ou estado plano de deformação. Diferentes arranjos das derivadas das funções de forma originam diferentes matrizes [B], se o modelo de análise é axissimétrico ou sólido. Assim, como sugerido por [7], é fundamental a criação da *classe modelo de análise*. A figura 4 ilustra a interação entre as classes concebidas, bem como o algoritmo para montagem da matriz de rigidez de um elemento paramétrico.



(a) Interação entre as classes **(b) Algoritmo**

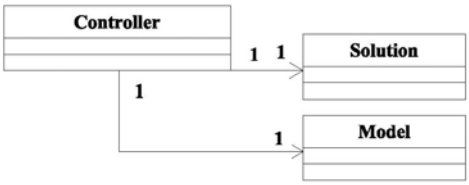
Figura 4. Montagem da matriz de rigidez de um elemento paramétrico.

Uma análise semelhante à feita acima pode ser adotada para o cálculo do carregamento nodal equivalente.

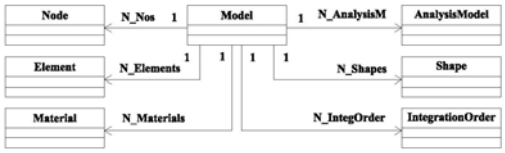
Além das classes necessárias à caracterização de um elemento finito, há que se preocupar com aquelas relativas ao modelo discreto como um todo. Assim, uma classe responsável pelas diversas coleções de objetos de um modelo discreto (elementos, nós, materiais, etc.) é necessária. Esta classe, denominada *modelo*, cria os diversos objetos da discretização através da interação com os dados persistidos em arquivos XML. Com o modelo discreto gerenciado pela *classe modelo*, a solução do mesmo precisa ser adequadamente tratada, justificando assim a criação de uma classe específica para este fim (*classe solução*). Finalmente, resta criar uma classe responsável pela definição do problema a ser resolvido. Esta classe faz requisições apropriadas às classes *modelo* e *solução* de maneira a produzir os resultados desejados para determinado problema (mecânica estrutural, transferência de calor, entre outros). A classe com estas características será denominada *classe controladora do problema*.

Além do algoritmo mostrado na figura 4, vários outros algoritmos genéricos, independentes do tipo de elemento, aparecem em qualquer modelo do MEF. São os algoritmos de montagem das matrizes globais a partir das matrizes dos elementos, baseados na técnica da rigidez direta. Tais algoritmos, na análise orientada a objetos que aqui se discute, são de responsabilidade da *classe modelo*.

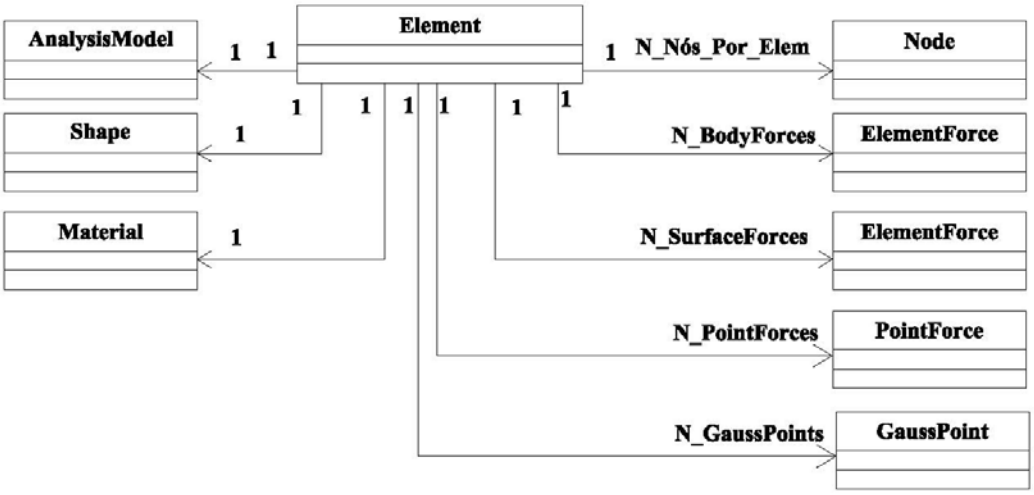
A figura 5 mostra os diagramas UML [2] de composição das principais classes criadas para implementação do núcleo numérico do sistema.



(a) Classe `Controller`



(b) Classe `Model`



(c) Classe `Element`

Figura 5. Diagramas de composição das principais classes do núcleo numérico do sistema.

A classe `Controller` (fig. 5(a)) possui uma instância da classe `Model` e uma da classe `Solution`. As classes `Comtroler` e `Solution` são abstratas e, através do mecanismo de herança [10], foram implementadas as classes `StructuralMech` e `OnePointEq`, derivadas, respectivamente de `Controller` e `Solution`, para tratar da obtenção das grandezas estáticas e cinemáticas associadas a um ponto de equilíbrio de um problema de mecânica estrutural. A figura 5(b) mostra a composição da classe `Model` que representa um modelo de elementos finitos. Esta classe possui listas de objetos do tipo `Node`, `Element`, `Material` `AnalysisModel`, `Shape` e `IntegrationOrder`, cada uma contendo tantos objetos quanto os necessários para caracterizar completamente um modelo do MEF. A classe `Element`, cuja composição destaca-se na fig. 5(c), referencia um objeto do tipo `AnalysisModel` representando seu tipo de análise, um objeto do tipo `Shape` representando suas funções de forma, um objeto do tipo `Material` representando seu material e objetos do tipo `Node` representando seus nós e possui objetos do tipo `ElementForce` representando as forças por unidade de comprimento, área ou volume, objetos do tipo `PointForce` representando as forças concentradas e objetos do tipo `GaussPoint` representando seus pontos de integração.

Para generalizar o modelo quanto ao domínio do elemento finito paramétrico, às funções de forma e aos modelos de análise, o mecanismo de herança foi utilizado para disponibilizar elementos finitos unidimensionais, bidimensionais quadrilaterias e triangulares e tridimensionais hexaédricos e tetraédricos. O mecanismo de herança também permitiu disponibilizar elementos com funções de aproximação diversas e diferentes modelos de análise. A Tabela 1 mostra os diversos recursos disponibilizados.

TABELA 1 – Recursos disponibilizados na aplicação.

Recurso	**Tipos disponíveis**		
Elemento	Linha	2 nós	*L2*
		3 nós	*L3*
		4 nós	*L4*
	Quadrilátero	4 nós	*Q4*
		8 nós	*Q8*
		9 nós	*Q9*
	Quadrilátero Axissimétrico	4 nós	*AxiQ4*
		8 nós	*AxiQ8*
		9 nós	*AxiQ9*
	Triângulo	3 nós	*T3*
		6 nós	*T6*
		10 nós	*T10*
	Triângulo Axissimétrico	3 nós	*AxiT3*
		6 nós	*AxiT6*
		10 nós	*AxiT10*
	Hexaédrico	8 nós	*H8*
		20 nós	*H20*
Modelo de Análise	Unidimensional		*LineAnalysisM*
	Bidimensional	Estado plano de tensões	*PlaneStressAnalysisM*
		Estado plano de deform.	*PlaneStrainAnalysisM*
		Axissimétrico	*AxiSymetricAnalysisM*
	Tridimensional		*SolidAnalysisM*
Ordem de Integração	Coordenadas naturais cartesianas		1 a 8 pontos
	Coordenadas naturais de área		1, 3, 4 e 6 pontos
Carregamento	Linha		*LineElementForce*
	Área		*SurfaceElementForce*
	Volume		*VolumeElementForce*
Solução	Equilíbrio em um ponto		*OnePointEq*
Controle	Mecânica Estrutural (Análise de Tensões)		*StructuralMech*

4.1 Exemplo

Para ilustrar os recursos disponibilizados no núcleo numérico, apresenta-se aqui um exemplo que tem por objetivo obter as tensões radiais σ_r atuantes em uma cunha de espessura unitária submetida à força concentrada *P* no seu vértice. As configurações geométricas e de cargas da cunha estão mostradas na fig. 6, sendo L = 3 m, P = 20 kN, módulo de elasticidade E = 1 kN/m^2 e coeficiente de Poisson υ = 0,3. Neste exemplo utilizou-se o modelo de análise de estado plano de deformações e o elemento triangular *T3* nas malhas de 25, 100 e 400 elementos, mostradas na fig. 7. Os valores da tensão σ_r obtidos nos pontos de Gauss próximos de θ = 0.0, para r variando ao longo da altura da cunha, estão representados no gráfico da fig. 8.

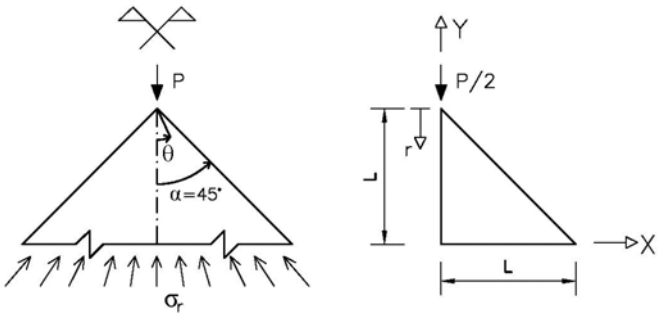
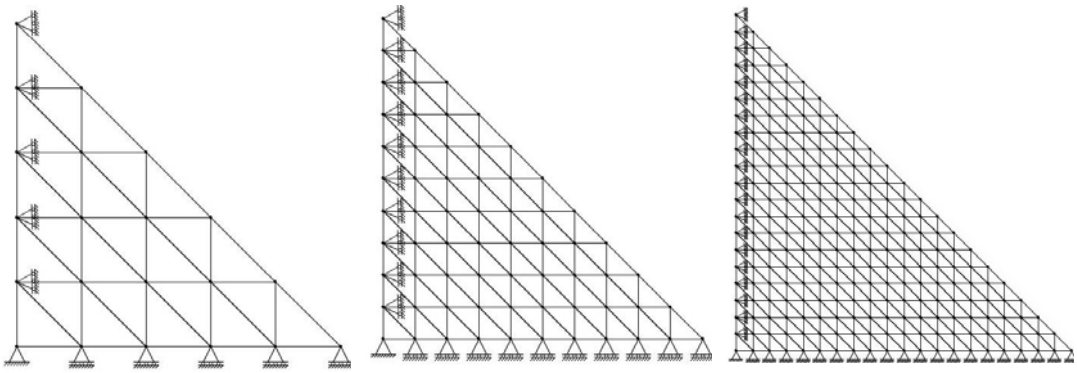


Figura 6. Cunha submetida à força concentrada.



(a) 25 T3 **(b) 100 T3** **(c) 400 T3**

Figura 7. Discretizações para a cunha (malhas obtidas com pré-processador do sistema).

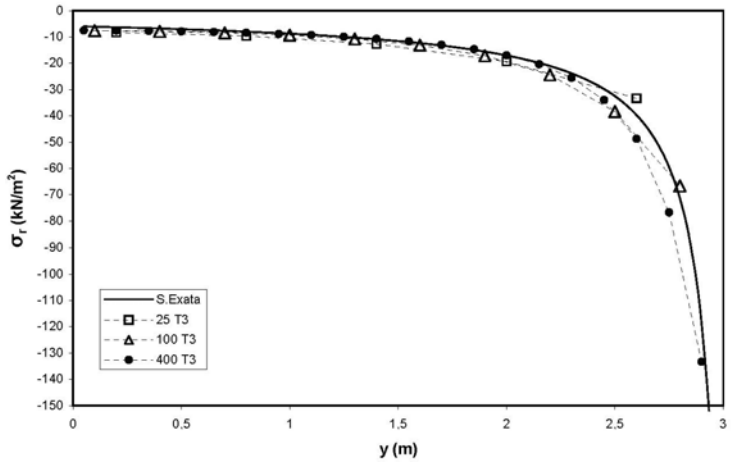


Figura 8. Tensões σ_r para a cunha proposta.

5 PÓS-PROCESSADOR

A versão atual do pós-processador contempla somente a representação de resultados de elementos finitos unidimensionais. A figura 9 ilustra esta representação para um modelo de pórtico plano. A figura 9(a) mostra a configuração geométrica e de cargas do pórtico. As reações estão mostradas na figura 9(b). Os diagramas de esforço normal, cortante e momento fletor estão mostrados nas figuras 9(c), 9(d) e 9(e), respectivamente. A figura 9(f) mostra a configuração deformada do pórtico.

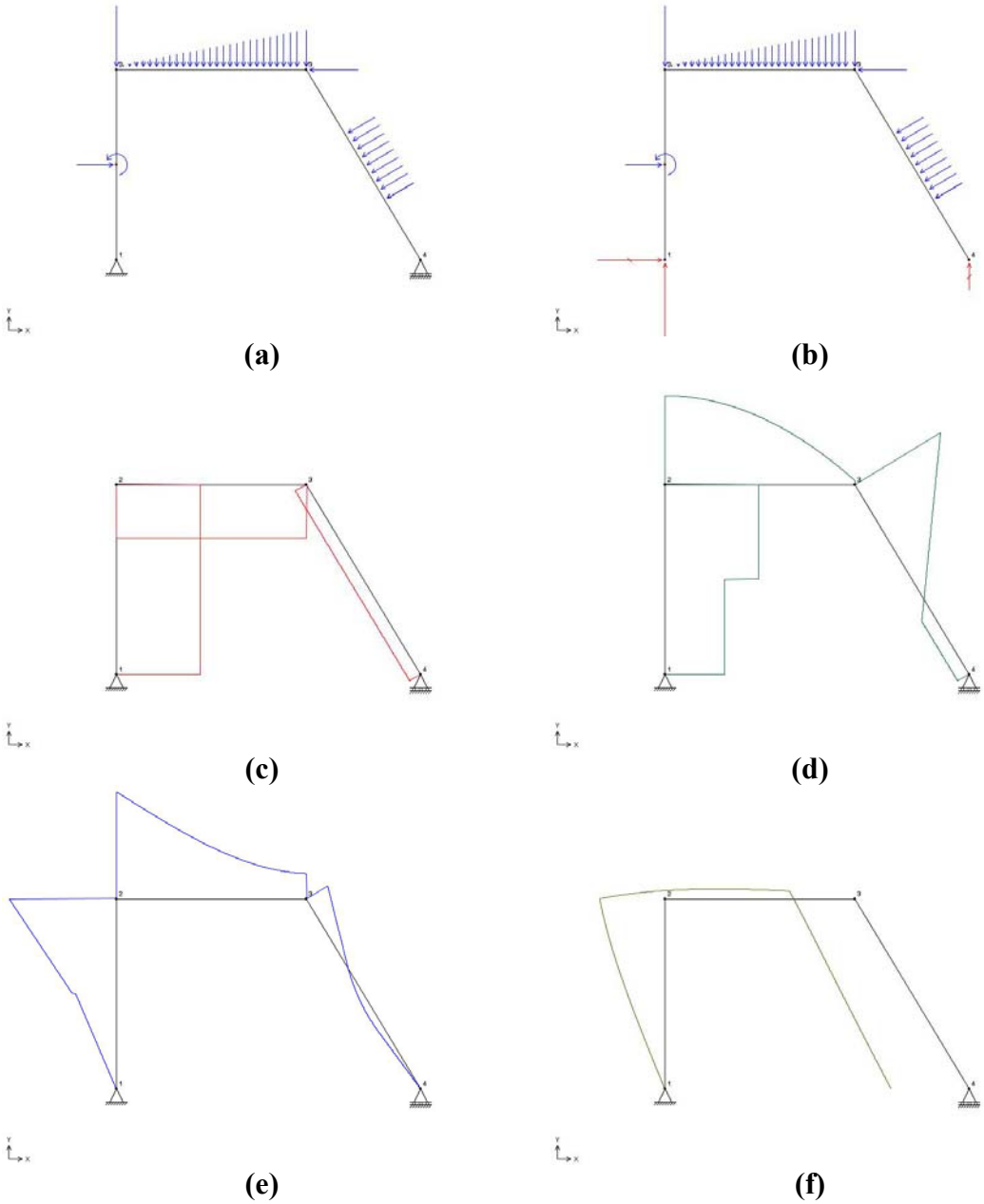


Figura 9. Funcionamento do Programa.

6 CONSIDERAÇÕES FINAIS

O artigo apresentou parte de um sistema computacional segmentado, amigável à mudanças e escalável em complexidade, que tem por objetivo fomentar a pesquisa de modelos discretos de análise estrutural. O sistema, ainda em desenvolvimento, permite um trabalho colabarativo que aglutina vários alunos, pesquisadores e idéias na área de métodos numéricos e computacionais aplicados à engenharia. Partindo-se da implementação aqui apresentada, outros trabalhos já foram iniciados mostrando o poder do sistema na ampliação da criatividade da pesquisa na área, uma vez que evita o recomeço do processo, tão presente nos sistemas computacionais tradicionais.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ALMEIDA, M. L., 2005. Elementos Finitos Paramétricos Implementados em Java. Dissetação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.

[2] BOOCH, G., RUMBAUGH, J., & JABOBSON, I., 2000. UML - Guia do Usuário. Editora Campus.

[3] FONSECA, G. L., 1989. Editor gráfico de malhas transfinitas tridimensionais para elementos finitos. Master's thesis, Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro, PUC/Rio, Rio de Janeiro, RJ, Brasil.

[4] GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J., 1995. Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley.

[5] GONÇALVES, M. A. B., 2004. Geração de malhas bidimensionais de elementos finitos baseada em mapeamentos transfinitos. Dissetação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.

[6] LICHAO, Y. & ASHOK, V. K., 2001. An object-oriented modular framework for implementing the finite element method. Computers and Structures, vol. 79, pp. 919–928.

[7] MARTHA, L., MENEZES, I. F., LAGES, E. N., JR., E., & PITANGUEIRA, R. L., 1996. An oop class organization for materially nonlinear finite element analysis. XVII CILAMCE, pp. 229–232.

[8] OÑATE, E., 1995. Cálculo de Estructuras por el Método de Elementos Finitos - Analisis Estático Lineal. CIMNE, Barcelona.

[9] PIETRO, G. A., 2001. Utilização de padrões de projeto na reengenharia de sistemas. Master's thesis, Universidade Federal de São Carlos, São Carlos, SP, Brasil.

[10] SANTOS, R., 2003. Introdução à Programação Orientada a Objetos Usando Java. Campus, Rio de Janeiro.

[11] TIMOSHENKO, S. & GOODIER, J. N., 1980. Teoria da Elasticidade. Editora Guanabara Dois, Rio de Janeiro.