UM PROGRAMA GRÁFICO INTERATIVO PARA MODELOS ESTRUTURAIS DE BARRAS

Flavio Torres da Fonseca
Roque Luiz Pitangueira
flaviotf@dees.ufmg.br
roque@dees.ufmg.br
Departamento de Engenharia de Estruturas, Universidade Federal de Minas Gerais
Av. do Contorno, 842, 30110-060, Belo Horizonte - MG - Brasil

Resumo. *O artigo apresenta os resultados do trabalho de iniciação científica desenvolvido pelo primeiro autor. Trata-se de parte de um sistema computacional que visa fomentar a pesquisa de modelos discretos de análise estrutural, relativa a modelos estruturais de barras.*

A concepção do sistema, como um conjunto de segmentos de aplicação, é apresentada, destacando-se as vantagens deste conceito.

A escolha do paradigma de programação orientada a objetos e da linguagem de programação Java, como recursos de implementação, é discutida. Nesta discussão apontam-se as principais características das tecnologias escolhidas no que se refere às plataformas de desenvolvimento, execução e distribuição, bem como à segmentação, expansão e manutenção do código.

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.

A utilização de padrões de projeto de software e das APIs Java `Swing`, `Java2D` *e* `Collections` *na implementação dos recursos de interação, bem como do processamento numérico dos modelos também é discutida.*

A formulação dos modelos estruturais de barras, como casos particulares do Método de Elementos Finitos, é brevemente revisada. As abstrações de classes adotadas para implementação da referida formulação são apresentadas, indicando-se o segmento do núcleo numérico do sistema que a contém.

Mostra-se que o programa permite a obtenção de soluções, tanto de grandezas estáticas quanto de grandezas cinemáticas. Discute-se, então, a utilização desse recurso no ensino do Método de Elementos Finitos nos cursos de graduação em engenharia.

Palavras chave: *Modelos Estruturais de Barras, Método de Elementos Finitos, Computação Gráfica, Programação Orientada a Objetos.*

1. INTRODUÇÃO

Costuma-se dividir o processo de análise estrutural em três etapas (Soriano and Lima, 1999). Orientando-se por experiência de projeto, o problema de meio contínuo da estrutura real é substituído por um modelo matemático utilizando-se hipóteses simplificadoras. Tal modelo matemático é expresso por equações diferenciais (ordinárias ou parciais) cujas soluções, ditas soluções analíticas, são conhecidas apenas em alguns poucos casos simples. Para superar as limitações de tais soluções, adota-se um modelo numérico aproximado, dito modelo discreto. Nos modelos discretos, as equações são algébricas e as grandezas são determinadas em um número finito de pontos, diferentemente das soluções analíticas cujas equações diferenciais, quando resolvidas, permitem avaliar as grandezas em um número infinito de pontos. Dentre os métodos discretos mais utilizados destacam-se o método de elementos finitos (MEF) e o método de elementos de contorno (MEC).

A pesquisa na área de métodos numéricos e computacionais para os referidos modelos discretos procura um aprimoramento das hipóteses simplificadoras dos mesmos. Isto é feito de forma a ampliar complexidades a partir dos conceitos já consolidados. Entretanto, observa-se sempre um recomeço do processo ao se recriarem as ferramentas relativas às tecnologias dominadas. Um exemplo ilustrativo deste fato é a (re)implementação computacional de algoritmos de solução de sistemas de equações algébricas lineares, toda vez que os mesmos são usados como parte do processo de aprimoramento de determinado modelo discreto.

Ao longo do tempo, algumas iniciativas de desenvolvimento de software pela comunidade acadêmica resultaram em produtos dependentes de sistema operacional, pouco amigáveis, escritos em linguagens de programação não apropriadas, de expansão, distribuição e manutenção difíceis, desenvolvidos por equipes fechadas, com documentação deficiente, entre outras limitações. Tais fracassos podem ser creditados à falta de disposição da comunidade em se apropriar das tecnologias emergentes ou mesmo à inexistência das mesmas.

Esta constatação confronta-se com o surgimento e aprimoramento de soluções tecnológicas para desenvolvimento de software, como programação orientada a objetos, linguagem Java, XML (eXtensible Markup Language), padrões de projeto de software, entre outras. Soluções estas que permitem o desenvolvimento de sistemas computacionais segmentados, amigáveis a mudanças e escaláveis em complexidade (Alvim, 2003).

Portanto, o desafio de desenvolver sistemas utilizando estes recursos é condição obrigatória para aprimoramento da agilidade e criatividade da pesquisa na área.

Este artigo apresenta parte do desenvolvimento de um sistema computacional cujo principal objetivo é fomentar a pesquisa de modelos discretos de análise estrutural.

2. O PROJETO INSANE

A utilização de modelos discretos de análise estrutural compreende três etapas principais inter-relacionadas: (1)criação do modelo, (2)montagem e resolução do modelo e (3)avaliação de resultados. Na criação do modelo, o analista informa as hipóteses simplificadoras relativas à geometria, material, carregamento e condições de contorno e estas são representadas com entidades matemáticas apropriadas, gerando assim o que se denomina malha e os atributos do modelo. Na etapa de montagem e resolução do modelo, combinam-se as informações matematicamente representadas, de modo a produzir equações algébricas que, quando solucionadas, permitem obter as diversas grandezas. Na avaliação de resultados, o analista faz uma análise crítica e verifica a adequação dos mesmos ao problema

em estudo.

Para disponibilização deste processo em computadores, normalmentente a referida divisão é adotada através de 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 das 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 disponibilizam 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.

Cada um dos três segmentos da aplicação é implementado segundo o paradigma de programação orientada a objetos (POO), adotando-se uma arquiteura em camadas e padrões de projeto de software apropriados.

2.1 Arquitetura em Camadas e Padrões de Projeto de Software

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 (Gamma et al., 1995) denominado `Modelo-Vista-Controlador` (MVC). Este padrão é bastante apropriado uma vez que preconiza a separação do processamento da informação de sua representação gráfica (Pietro, 2001), 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.

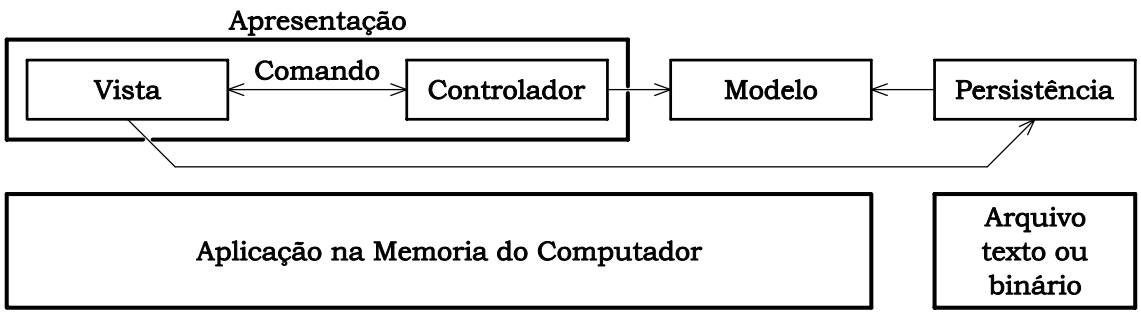


Figura 1: Arquitetura em Camadas e Padrões de Projeto adotados no INSANE

O inter-relacionamento entre as camadas é conseguido, principalmente, através da implementação do padrão de projeto de software denominado `Comando` (Gamma et al., 1995). A figura 2 exemplifica este relacionamente 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 (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`.

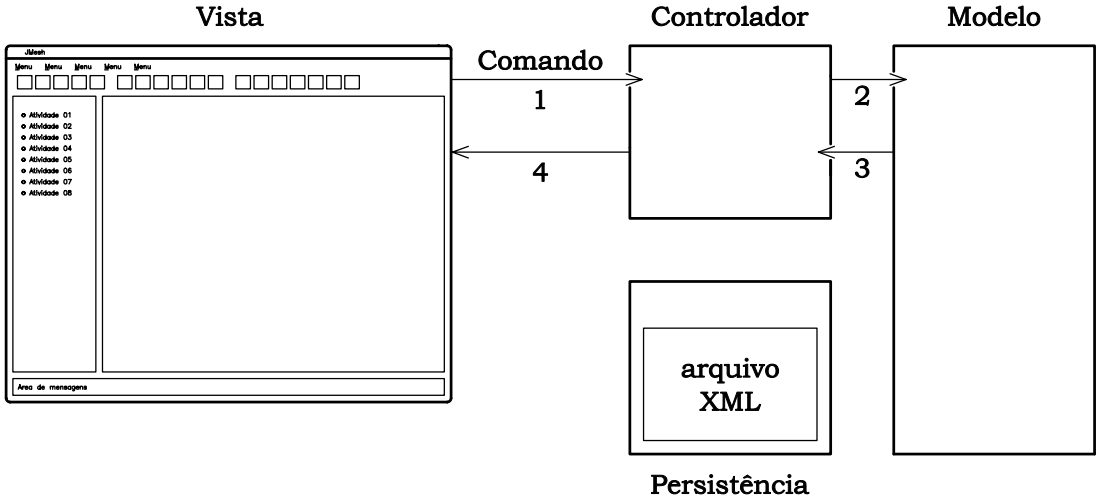


Figura 2: Relacionamento entre camadas do INSANE, para adição de uma entidade geométrica

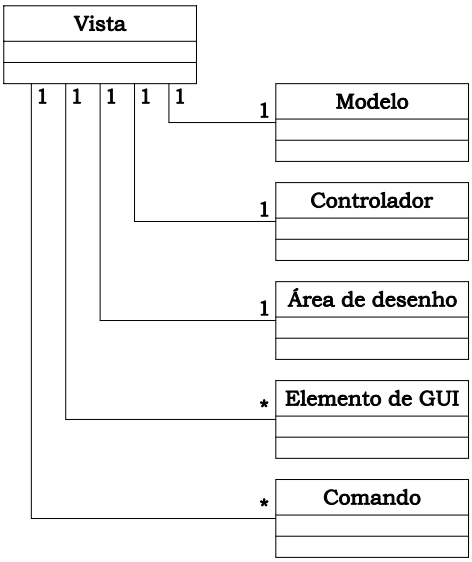
2.2 Implementação do Padrão MVC

Para implementação do Padrão MVC no sistema, foram criadas as classes `Vista`, `Modelo` e `Controlador`, conforme os diagramas UML ("Unified Modeling Language" (Booch et al., 2000)) da figura 3.

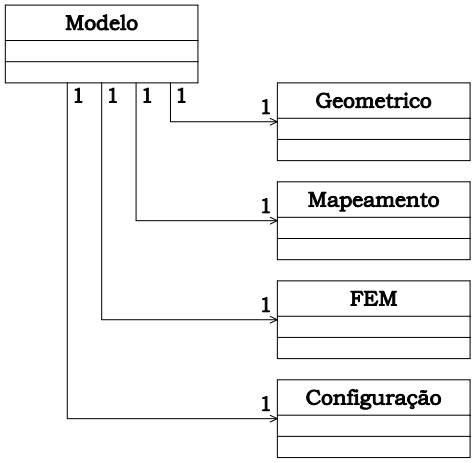
A classe `Vista` (figura 3(a)) possui um objeto da classe `Modelo`, um da classe `Controlador`, um da classe `Área de Desenho`, vários objetos da classe `Comando` e vários objetos representativos de elementos que compõem a interface gráfica com o usuário (GUI - "Graphical User Interface").

A classe `Modelo` (figura 3(b)) é composta por instâncias de classes que implementam a lógica da modelagem através do Método dos Elementos Finitos e por um objeto da classe `Configuração`, que armazena informações comuns a todas as classes de modelagem. Entre as classes de modelagem, a classe `Modelo Geométrico` cuida da estrutura de dados representativa da geometria do modelo. A classe `Mapeamento` implementa técnicas de geração de malhas de elementos finitos através de mapeamentos (transfinito, isoparamétrico, entre outros (ver Fonseca (1989))). A classe `FEM` representa o modelo do Método dos Elementos Finitos e possui objetos que representam todos os atributos de uma discretização baseada neste método.

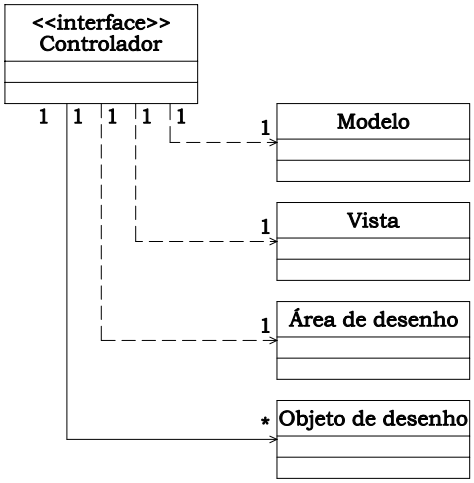
A classe `Controlador` (figura 3(c)), como recomenda Grand (1998), é uma classe totalmente abstrata (em `Java`, uma interface) que faz referência ao `Modelo` corrente, à `Vista` e à `Área de Desenho`.



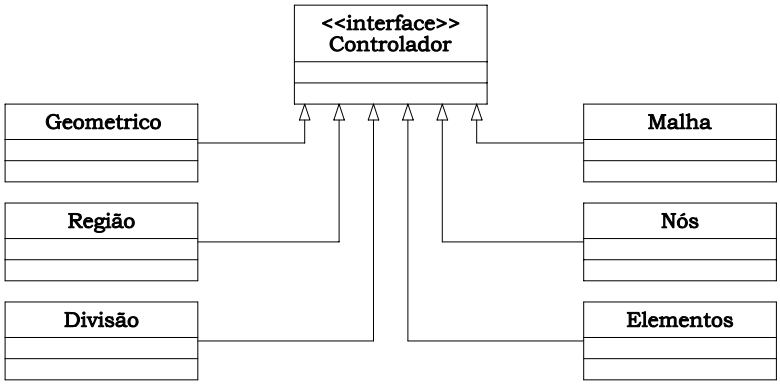
(a) Instâncias da `Vista`



(b) Instâncias do `Modelo`



(c) Instâncias do `Controlador`



(d) Herança do `Controlador`

Figura 3: Diagramas do projeto orientado a objetos do INSANE

Cada uma das classes que implemantam a interface `Controlador` (ver figura 3(d)) possui listas (instâncias de classes da `API Collections` (Horstmann and Cornell, 2000)) contendo instâncias de `Objeto de Desenho`, que são extensões de classes da `API` gráfica `Java2D` (Rowe, 2001).

Os objetos `Área de Desenho` e `Elemento de GUI` (ver figuras 3(a) e 3(c)) são intâncias de classes da `API Java Swing` (Horstmann and Cornell, 2000). O objeto `Área de Desenho` é um componente da `Swing`, onde um objeto `Graphics2D`, composto das intâncias de `Objeto de Desenho` do `Controlador` (ver figura 3(c)), é desenhado.

Como dito anteriormente, o relacionamento entre as classes criadas para implementação do `MVC` é conseguido com o uso do padrão `Comando`. A figura 4, que detalha a figura 2, exemplifica este relacionamento.

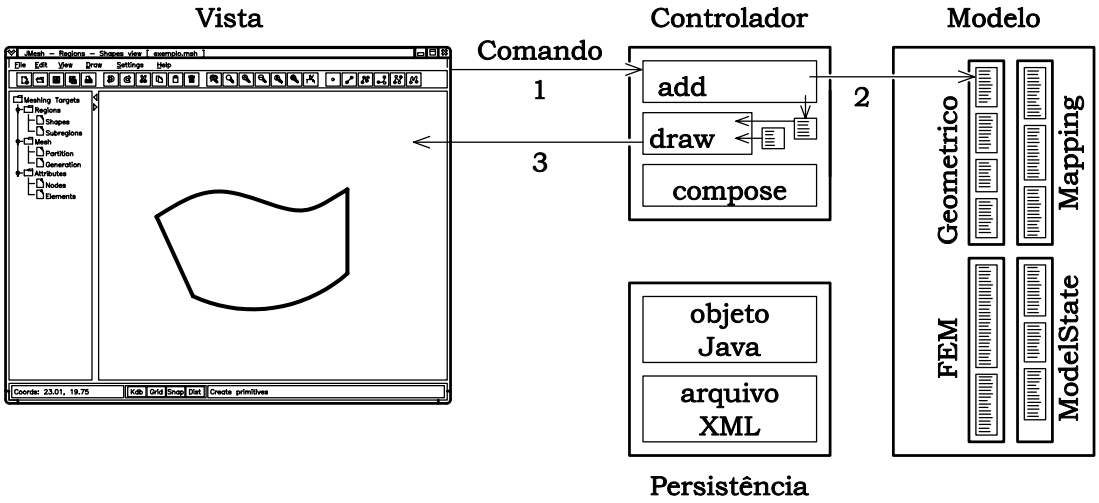


Figura 4: Relação entre camadas para operação de adição de objeto no INSANE

Tendo o Usuário requisitado a introdução de uma entidade geométrica ao modelo, o `Controlador` correspondente (ver figura 3(d)) é acionado (etapa 1 das figuras 2 e 4).

Com informações relativas à entidade a ser adicionada, o `Controlador` solicita a adição da mesma no `Modelo` (etapa 2 das figuras 2 e 4).

Ainda possuidor do controle da operação, o objeto `Controlador` adiciona uma instância de `Objeto de Desenho`, correspondente à entidade geométrica, em sua lista (ver método `add` na figura 4 equivalente à etapa 3 da figura 2) e repassa o controle para o `Comando`.

Tal `Comando` solicita à `Vista` que atualize sua `Área de Desenho`. Para tanto, o `Controlador` corrente é acioando para que construa o objeto `Graphics2D` correspondente, a ser desenhado (etapa 3 da figura 4 e etapa 4 da figura 2).

A figura 5 ilustra o diagrama `UML` de seqüência para a requisição de adição de um ponto no `Modelo Geométrico`.

3. INSANE PARA MODELOS ESTRUTURAIS DE BARRAS

A versão do INSANE que aqui se apresenta contempla modelos estruturais de barras. A figura 6 mostra a tela principal do programa, onde se visualiza o diagrama de momentos fletores de um modelo de viga contínua. Como pode ser visto na figura, as tarefas de modelagem são gerenciadas através da árvore posicionada à esquerda da área de desenho. Os recursos de pré e pós-processamento disponibilizados na versão atual do programa

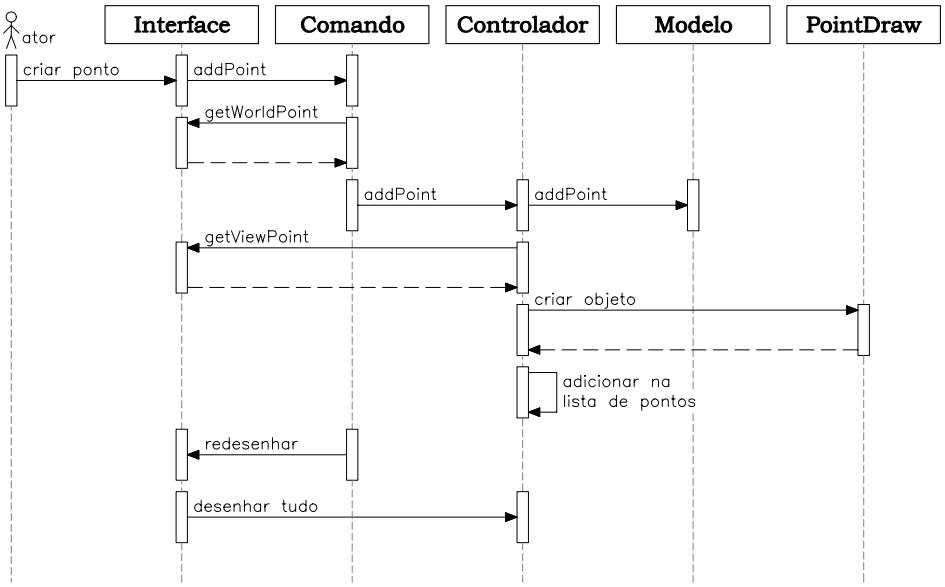


Figura 5: Diagrama de seqüência para operação de adição de objeto no INSANE

serão apresentados na seção 4. Esta seção discute o projeto orientado a objetos para o processamento numérico de modelos estruturais de barras.

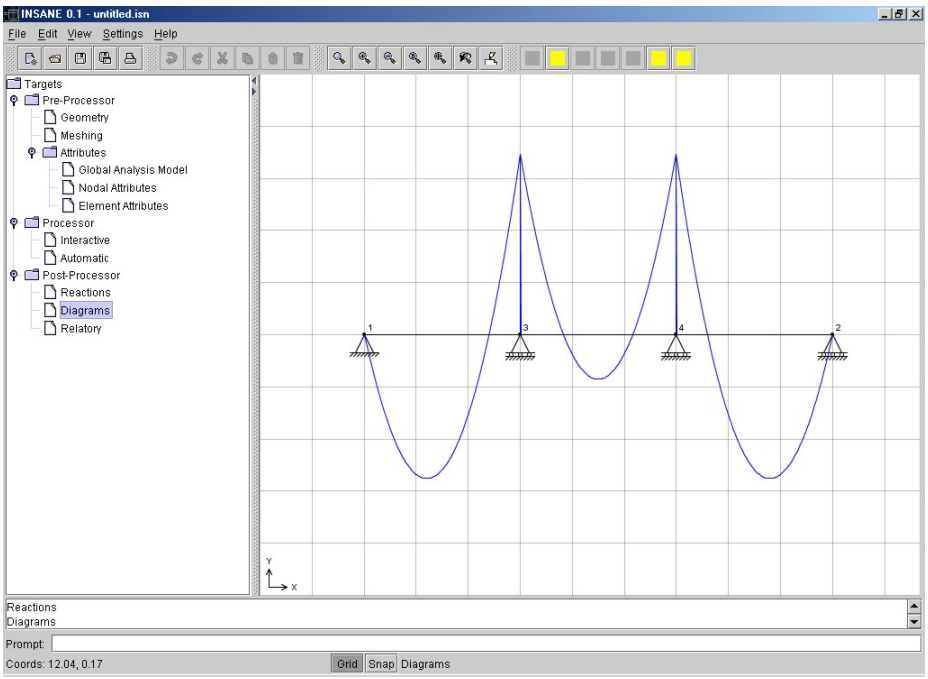


Figura 6: Tela principal do INSANE

3.1 Modelos Estruturais de Barras como casos Particulares do MEF

Uma das etapas mais difíceis do projeto de software orientado a objetos é estabelecer qual a decomposição do problema em sub-problemas mais adequada. Também não é nada fácil decidir quais categorias ou classes de objetos compõem o problema a ser resolvido.

Para modelos discretos de análise estrutural, esta tarefa pode ficar menos árdua se a formulação matemática dos mesmos for observada com cuidado. No caso do Método

dos Elementos Finitos, formulado em termos cinemáticos, os deslocamentos em qualquer ponto (armazenados no vetor $\underline{u}$) são obtidos a partir dos valores nodais (armazenados no vetor $\underline{d}$), através da aproximação

$$\underline{u} = \underline{N}\ \underline{d} \tag{1}$$

onde $\underline{N}$ é a matriz que contém as chamadas funções de forma do elemento, uma para cada deslocamento permitido.

Utilizando a aproximação dada na equação 1, as componentes de deformação (vetor $\underline{\epsilon}$) podem ser calculadas através da relação

$$\underline{\epsilon} = \underline{B}\ \underline{d} \tag{2}$$

onde $\underline{B}$ é uma matriz de derivadas das funções de forma.

Adotando-se hipóteses relativas ao comportamento do material, as tensões (vetor $\underline{\sigma}$) podem ser calculadas por

$$\underline{\sigma} = \underline{C}\ \underline{\epsilon} \tag{3}$$

onde $\underline{C}$ é uma matriz contendo combinações de propriedades do material.

Combinando-se as relações acima (equações 1, 2 e 3) e informações relativas ao carregamento, obtém-se a equação matricial de equilíbrio de cada elemento, dada por

$$\underline{k}\ \underline{d} = \underline{p} + \underline{f} \tag{4}$$

onde

$$\underline{k} = \int_V \underline{B}^T\ \underline{C}\ \underline{B}\ dV \tag{5}$$

é a matriz de rigidez do elemento,

$$\underline{f} = \int_V \underline{N}^T\ \underline{b}\ dV \tag{6}$$

é o vetor que contém o carregamento nodal equivalente a cargas não nodais (vetor $\underline{b}$) e $\underline{p}$ é o vetor de cargas nodais.

A combinação das rigidezes e carregamentos dos vários elementos leva à equação de equilíbrio do modelo

$$\underline{K}\ \underline{d} = \underline{P} + \underline{F} \tag{7}$$

Para modelos estruturais de barras, tratados como casos particulares do Método dos Elementos Finitos, as equações 5 e 6 ficam simplificadas nas formas

$$\underline{k} = \int_L M\ S\ \underline{B}^T\ \underline{B}\ dx \tag{8}$$

e

$$\underline{f} = \int_L \underline{N}^T\ \underline{b}(x)\ dx \tag{9}$$

onde M é um escalar, propriedade do material, S é um escalar, propriedade geométrica da seção transversal e as integrais são realizadas ao longo do comprimento do elemento (L).

3.2 Análise e Projeto Orientado a Objetos

A partir da observação da formulação acima, é razoável supor, para o caso de modelos de barras, algumas categorias de objetos. Um nó é caracterizado por suas coordenadas, seus deslocamentos e suas cargas. Uma barra, genericamente tridimensional, é composta por nós, material, seção transversal e pode estar submetida a forças e momentos distribuídos em seu corpo. Quando parte de um modelo estrutural, cada barra tem seu estado deformado descrito por uma função de aproximação.

Estas categorias ou classes de objetos devem conter informações suficientemente gerais para caracterizá-los. Entretanto, é comum a particularização dos modelos estruturais de barras nas categorias: Treliça Plana, Viga, Pórtico Plano, Grelha, Treliça Espacial e Pórtico Espacial.

Para subdividir a tarefa de modelagem e resolução de modelos discretos, optou-se, com base na proposta feita por Martha et al. (1996) e Pitangueira (1998), pela criação de três classes abstratas: `Driver`, `Model` e `Solution`. A classe `Driver` contém todos os métodos e atributos necessários para a montagem e resolução do modelo desejado. Um objeto `Driver` contém uma instância da classe `Solution` e uma da classe `Model` (figura 7).

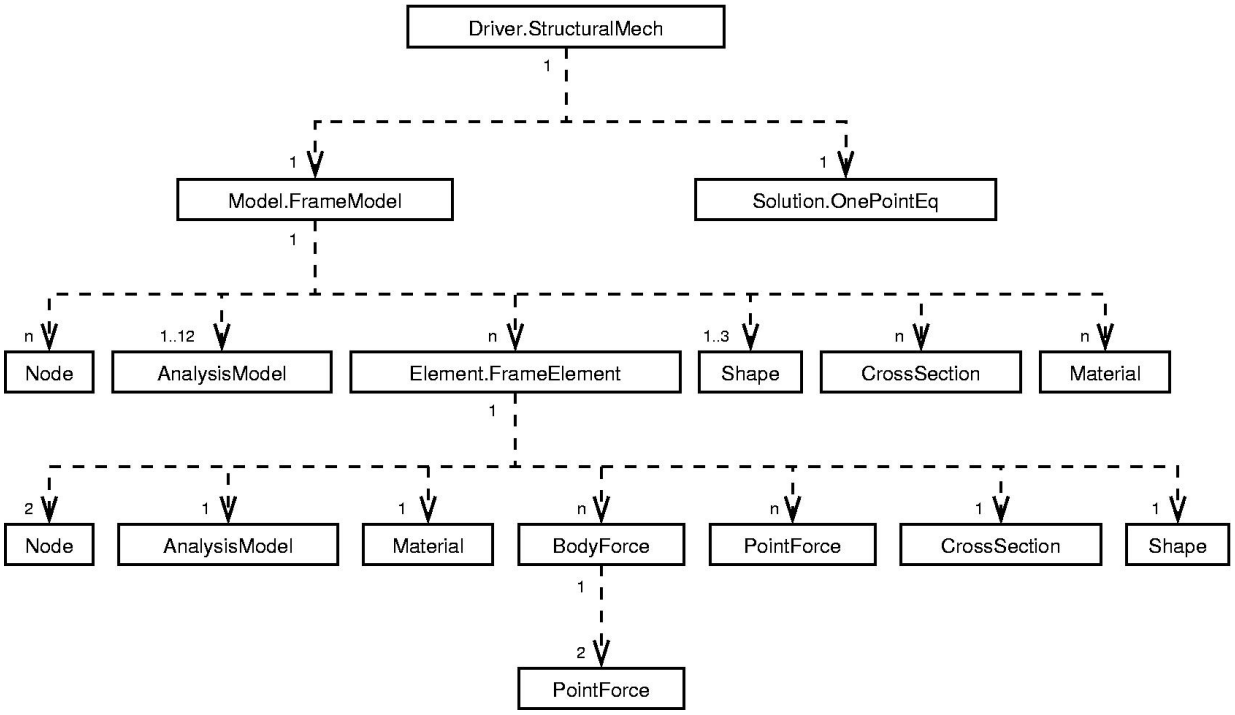


Figura 7: Intâncias de classes do programa

A classe `Model` contém os dados relativos ao modelo que deve ser analisado, como sua geometria, propriedades dos materiais e o tipo de análise a ser realizada. `Solution` fornece as ferramentas matemáticas necessárias para a resolução do modelo.

`Driver` contém métodos que, a partir dos dados contidos em `Model` e utilizando as ferramentas de `Solution`, montam e resolvem o problema. Para o caso particular de modelos estruturais de barras, foram implementadas subclasses destas classes principais. Foram criadas as classes `StructuralMech`, `OnePointEq` e `FrameModel`, como subclasses de `Driver`, `Solution` e `Model`, respectivamente (figura 7).

`FrameModel`, como dito anteriormente, é a classe em que são armazenados os dados

do modelo de barras. Estes dados são armazenados em listas, mais especificamente em instâncias de classes da `API Java Colections`. As listas contidas em um `FrameModel` são as listas de nós, elementos, materiais, seções transversais, tipos de análise e funções de forma (ver figura 7).

A classe abstrata `AnalysisModel` representa o tipo de análise escolhido para o modelo estrutural e para cada elemento específico. Seus atributos são o número de graus de liberdade de um elemento, as equações válidas e um identificador. Para o caso de elementos de barra, foi implementada uma subclasse de `AnalysisModel` denominada `FrameElmAnalysis`, que também é abstrata. Esta classe tem como subclasses classes que representam os seis tipos de análise de elementos de barra, denominadas `Beam`, `Grid`, `PlaneTruss`, `PlaneFrame`, `SpaceTruss` e `SpaceFrame`. Cada uma destas subclasses implementa os métodos abstratos definidos na classe base. Estes métodos retornam matrizes e vetores necessários para o cálculo das matrizes de rigidez e dos vetores de força do modelo.

A classe abstrata `Element` representa um elemento finito qualquer (ver figura 7). Seus atributos são um identificador, sua incidência (uma lista de objetos `Node`) e seus carregamentos, tanto concentrados quanto distribuídos. A classe `FrameElement`, subclasse de `Element`, representa um elemento de barra (ver figura 7). Seus atributos são seu comprimento, sua seção transversal, suas liberações nas extremidades, suas variações de temperatura (nas fibras superior e inferior e na altura do centro de gravidade), suas pré-deformações, as ações em suas extremidades e um vetor contendo forças nodais equivalentes para ser usado quando se desejar analisar uma estrutura submetida a um carregamento não previsto por esta implementação.

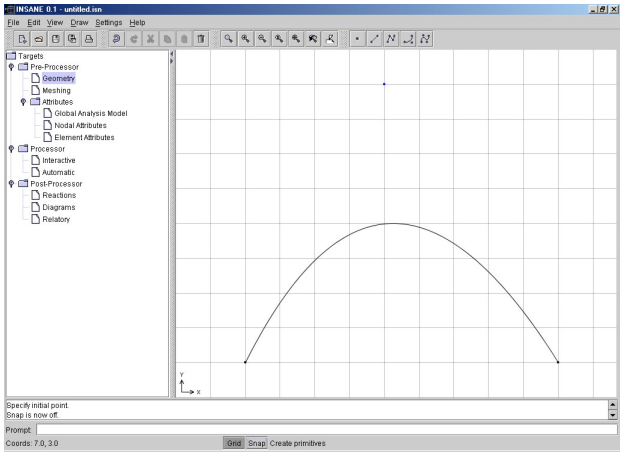
Detalhes dos atributos e métodos de cada uma das classes criadas para o processamento numérico de modelos estruturais de barras podem ser encontrados em Fonseca et al. (2004).

4. RECURSOS DISPONIBILIZADOS NA VERSÃO ATUAL

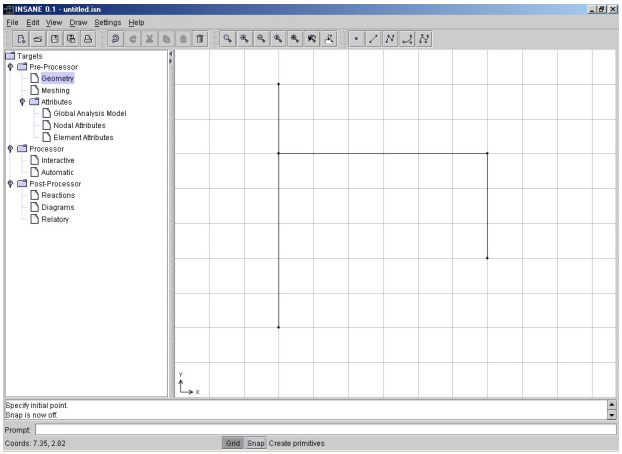
Na versão atual do sistema, vários recursos foram disponibilizados de maneira a facilitar o trabalho de modelagem. A figura 8 ilustra as três principais tarefas da etapa de pré-processamento: definição da geometria, criação da malha de elementos finitos e definição do modelo de análise. Para a definição da geometria, o usuário pode criar entidades como ponto, segmento de reta, seqüência de segmentos de reta, curvas quadráticas ou cúbicas (ver figuras 8(a) e 8(b)). Na criação da malha, cada entidade geométrica anteriormente definida pode ser subdividida e uma malha de elementos finitos de barra é gerada (ver figura 8(c)). O modelo de análise global (se pórtico plano, viga, treliça plana ou grelha) é uma informação fundamental, uma vez que define de quais tipos de elementos a malha pode ser formada. Na figura 8(d) define-se o modelo de grelha para a malha da figura 8(b).

Definido o tipo de análise global, segue-se informando os atributos dos nós e das barras. A figura 9 ilustra alguns dos atributos nodais possíveis, tais como restrições (modelo de grelha da figura 9(a)), forças (modelos de treliça da figura 9(b) e de pórtico da figura 9(c)), deslocamentos prescritos (modelo de pórtico da figura 9(d)), apoios elásticos (modelo de pórtico da figura 9(e))e apoios inclinados (modelo de pórtico da figura 9(f)).

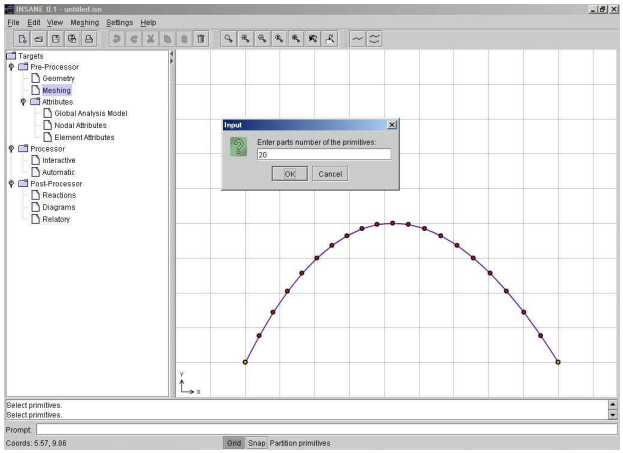
A figura 10 mostra os diálogos para atribuição de cargas distribuídas (modelos de pórtico da figura 10(a) e de grelha da figura 10(b)), forças concentradas (modelo de grelha da figura 10(c)), deformações prévias (modelo de treliça da figura 10(d)), variação de temperatura (modelo de pórtico da figura 10(e)) e liberação de extremidades (modelo de pórtico da figura 10(f)) . Além destes atributos, o programa também permite a prescrição



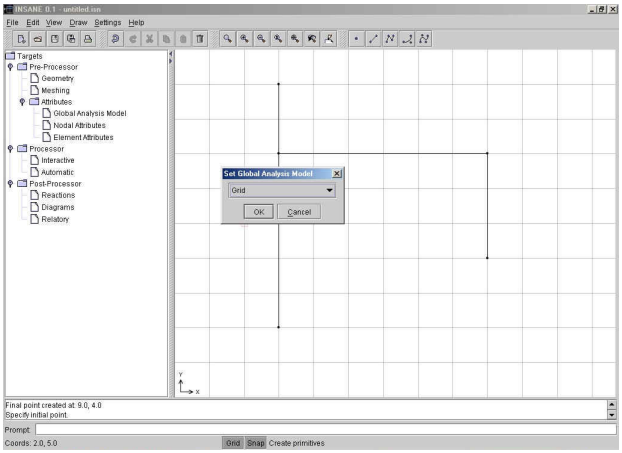
(a) Definição da geometria



(b) Definição da geometria



(c) Criação da malha



(d) Definição do modelo de análise

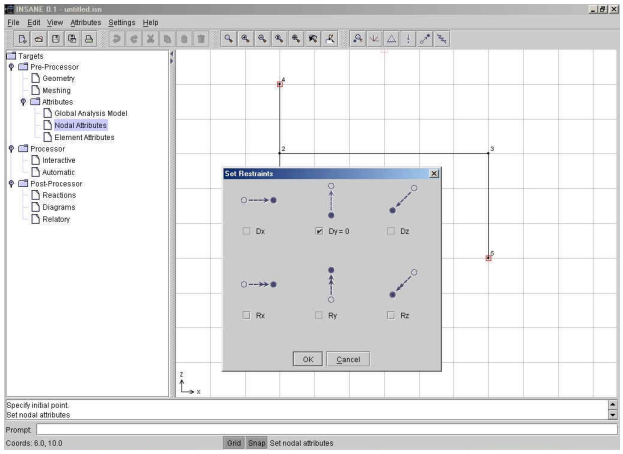
Figura 8: Definição da geometria e criação da malha de elementos finitos de barra no INSANE

de diferentes tipos de análise por barra, características das seções transversais, materiais, ligações elásticas e carregamentos nodais equivalentes oriundos de efeitos não tratados no programa.

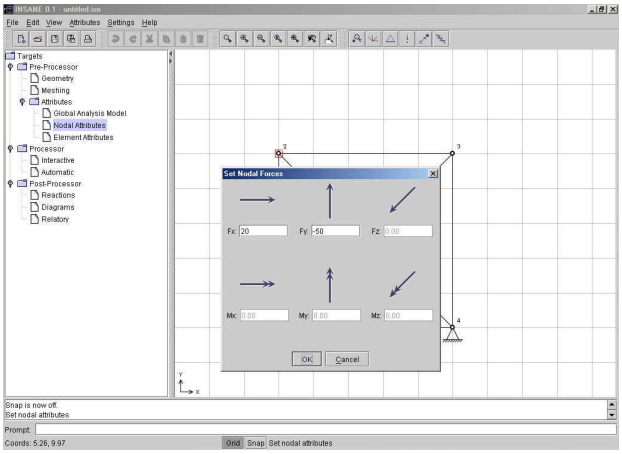
Definida a discretização, passa-se para a etapa de processamento. Nesta versão do programa somente o processamento automático, sem interferência do usuário, está implementado. Entretanto, para uso do programa como ferramenta de ensino, esta interferência durante o processamento numérico do modelo é fundamental.

Na etapa de pós-processamento (figura 11), é possível a visualização e consulta aos valores numéricos das reações de apoio (modelos de viga da figura 11(a) e de pórtico da figura 11(b)), diagramas de esforços internos (modelo de pórtico das figuras 11(c), 11(d) e 11(e)) e estado deformado do modelo (modelo de pórtico da figura 11(f)).

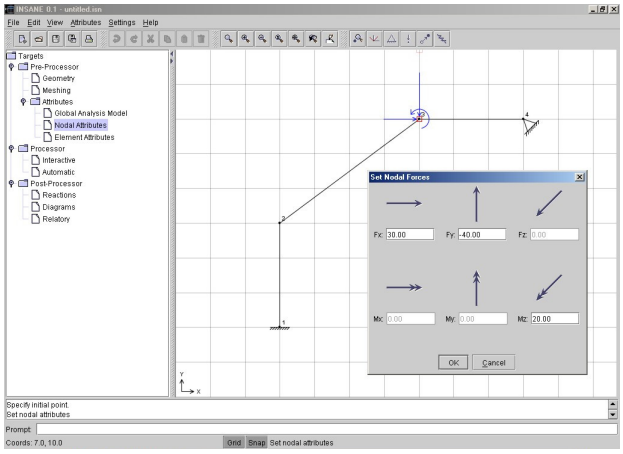
Também é possível visualizar e/ou exportar para arquivo `PDF` e `XML`, um relatório que contém todos os dados e resultados da análise (figura 4.). Tais relatórios foram gerados utilizando as tecnologias sugeridas em Tidwell (2001) e Lozano (2004).



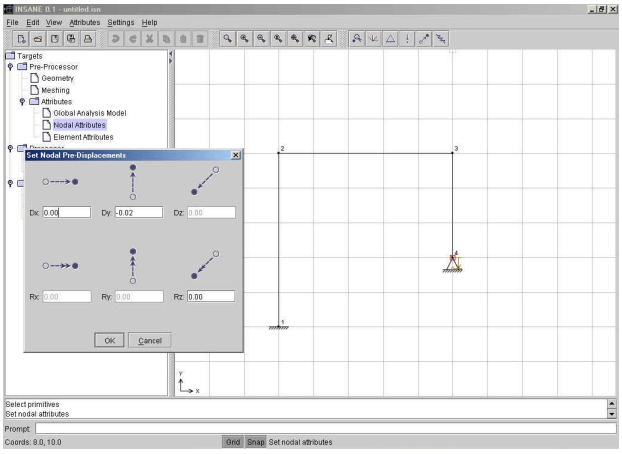
(a) Restrições nodais



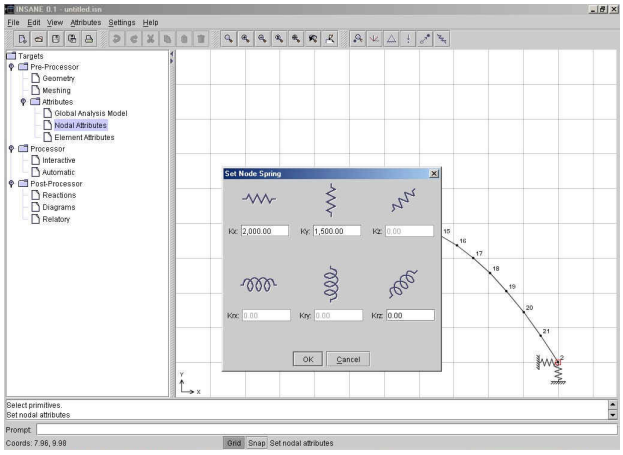
(b) Forças nodais



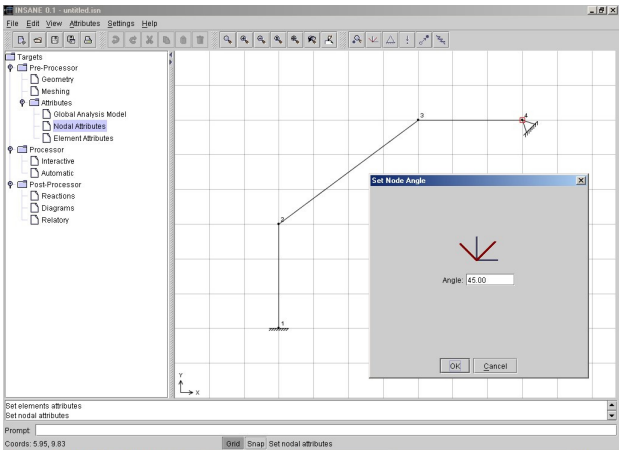
(c) Forças nodais



(d) Deslocamentos prescritos

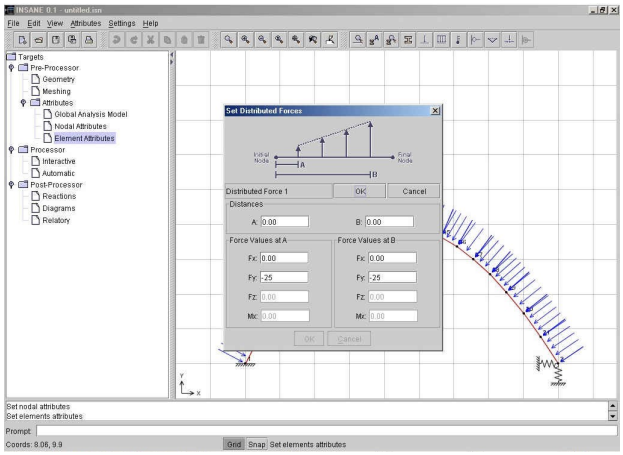


(e) Apoios elásticos

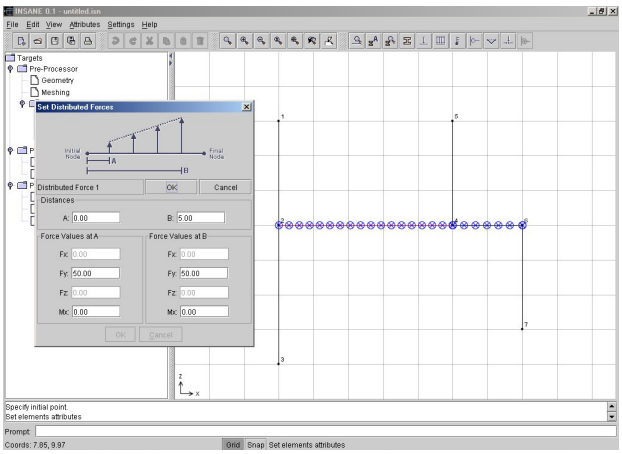


(f) Apoios inclinados

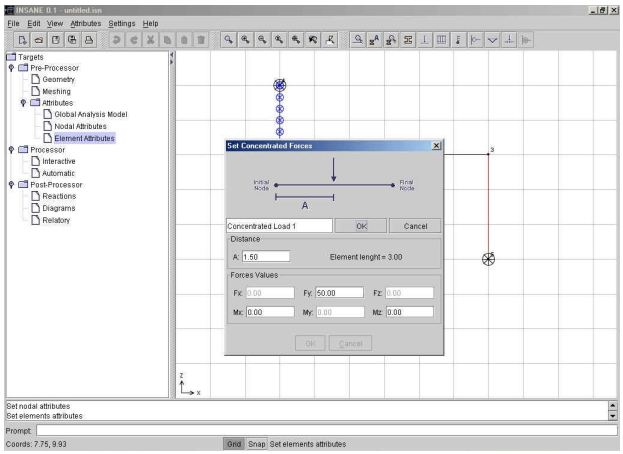
Figura 9: Informação de atributos nodais no INSANE



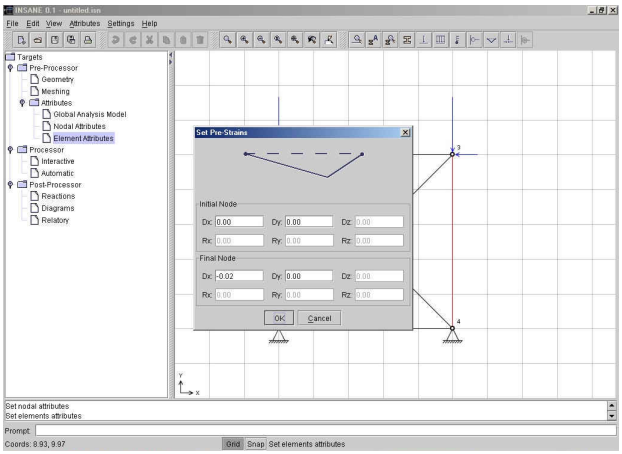
(a) Cargas distribuídas



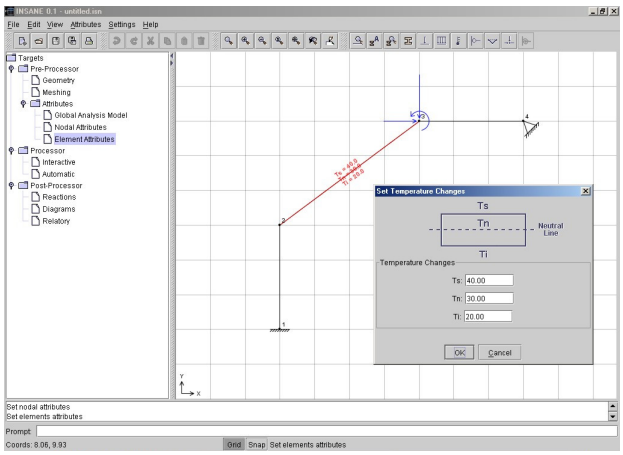
(b) Cargas distribuídas



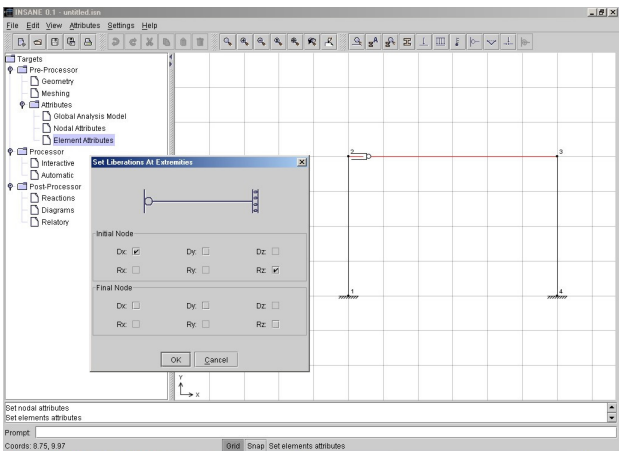
(c) Cargas concentradas



(d) Deformações prévias

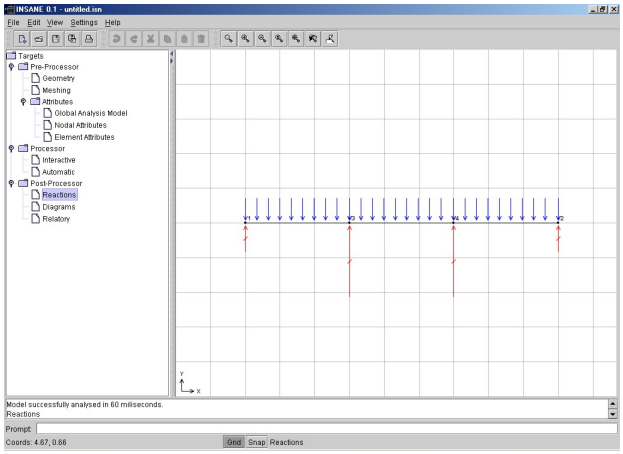


(e) Variação de temperatura

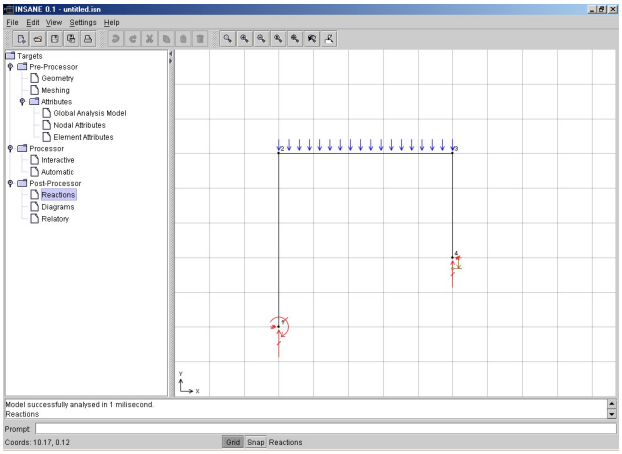


(f) Liberação de extremidades

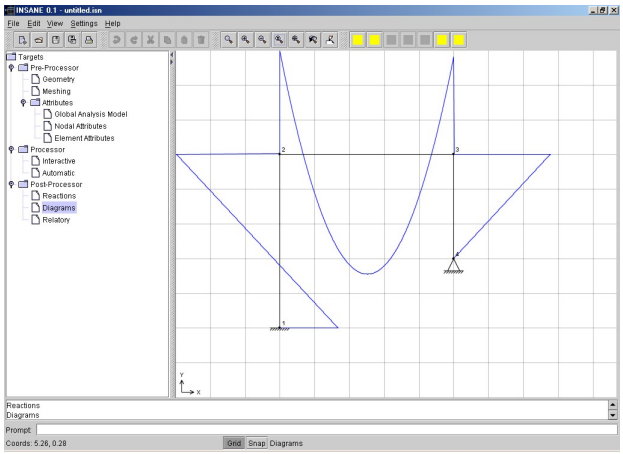
Figura 10: Informação de atributos das barras no INSANE



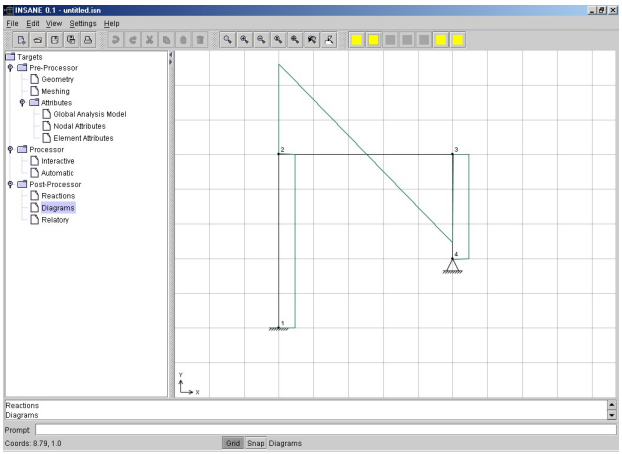
(a) Visualização das reaçoes



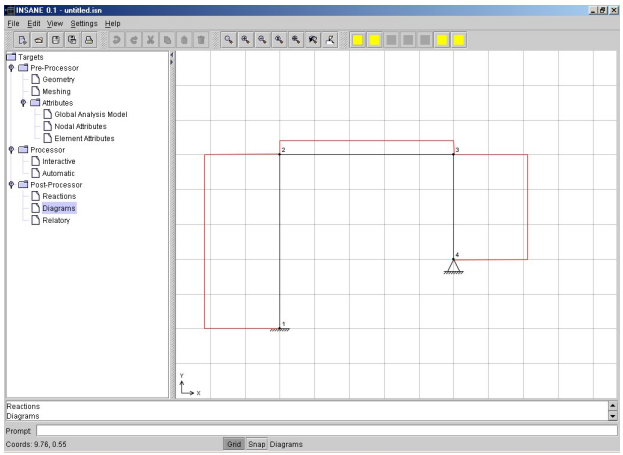
(b) Visualização das reaçoes



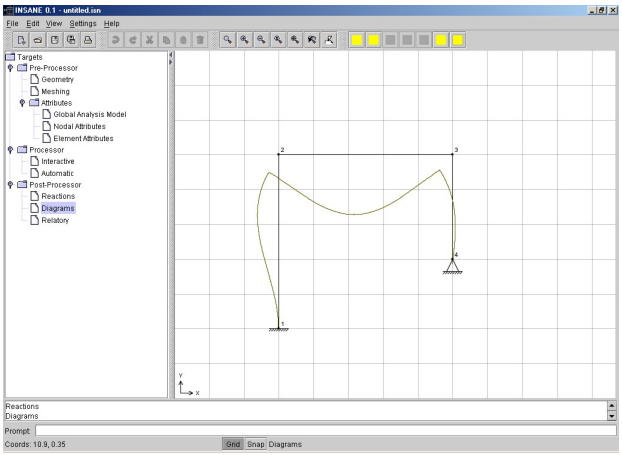
(c) Diagrama de momentos



(d) Diagrama de esf. cortante



(e) Diagrama de esf. normal



(f) Visualização da deformada

Figura 11: Visualização de resultados no INSANE

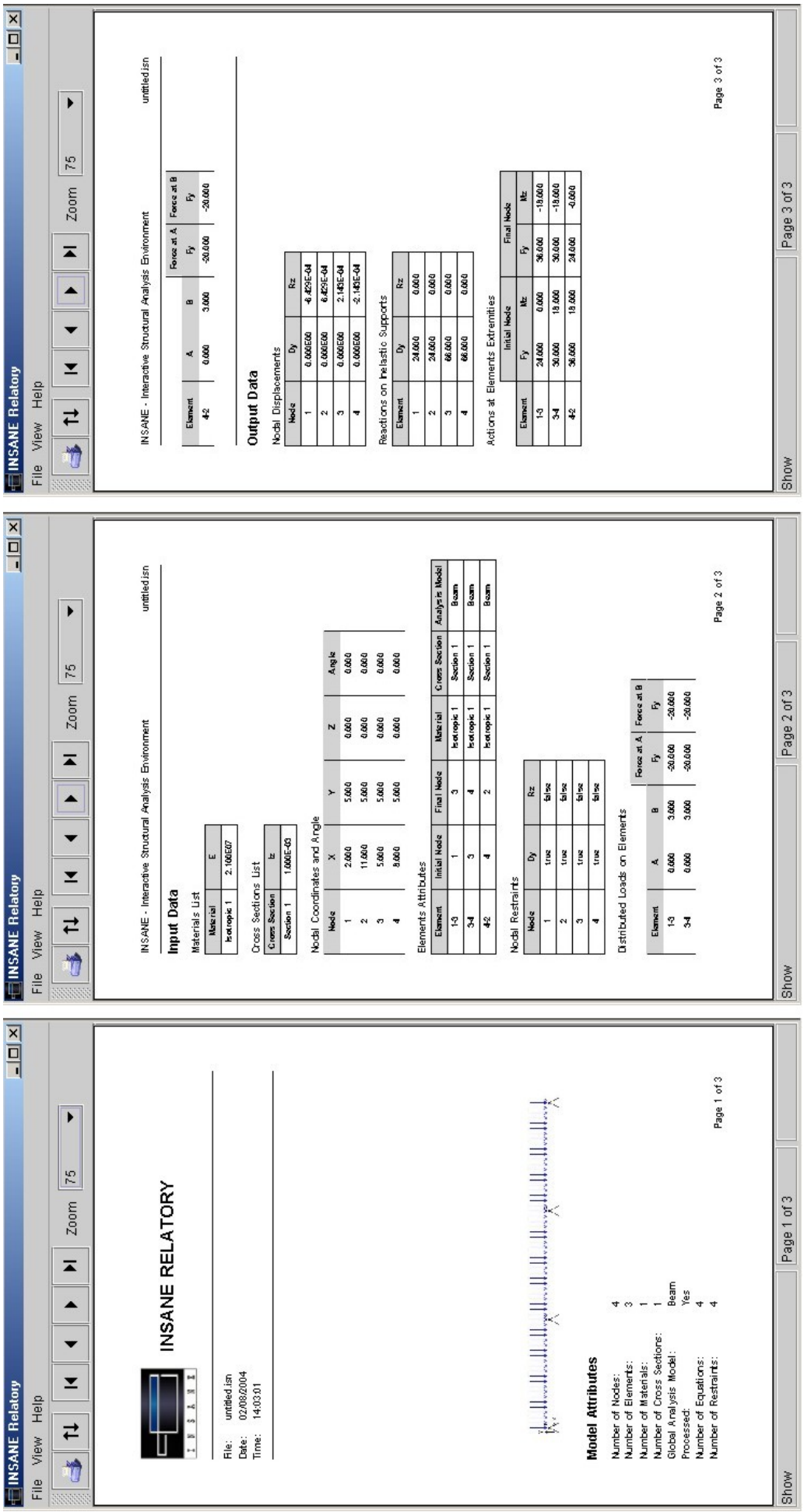


Figura 12: Visualização do relatório

5. CONSIDERAÇÕES FINAIS

O ensino do método dos elementos finitos (MEF) nos cursos de graduação de Engenharia contempla, inicialmente, os modelos estruturais de barras, para depois avançar na descrição de contínuos bi ou tridimensionais. O computador é normalmente utilizado como ferramenta auxiliar, através de programas que automatizam os procedimentos numéricos dos modelos. O estágio atual de evolução da computação gráfica permite dar saltos qualitativos relevantes no ensino do MEF, se forem utilizados programas computacionais que possibilitem a descrição dos dados de entrada (pré-processamento) e a visualização dos resultados de saída (pós-processamento) dos modelos, através de recursos gráficos interativos.

No estágio atual do desenvolvimento do sistema aqui apresentado, a parte da aplicação relativa a modelos estruturais de barras (discutida neste artigo) já está implementada e exaustivamente testada e está disponível em `http://www.dees.ufmg.br/insane`.

A implementação de um gerador de malhas que contempla elementos finitos bidimensionais já está em andamento. Outra tarefa, também já em andamento, é a expansão do núcleo numérico do sistema para contemplar elementos finitos paramétricos bi e tridimensionais.

Espera-se que o desenvolvimento do sistema aqui apresentado diminua as barreiras existentes entre o desenvolvimento teórico de modelos discretos de análise e sua aplicação. Também deseja-se que o sistema seja fomentador do desenvolvimento de novos modelos, evitando o recomeço do processo de implementação e permitindo maior agilidade e criatividade da pesquisa na área.

Agradecimentos

Ao apoio financeiro em forma de bolsa dado pela Pró-Reitoria de Graduação da UFMG, através do programa PAD, e pelo CNPq, através do programa PIBIC.

Referências

Alvim, P., 2003. Open source: Os novos desafios de negócios e a indústria de ti. *Developers Magazine*, , n. 80, pp. 13–15.

Booch, G., Rumbaugh, J., & Jabobson, I., 2000. *UML - Guia do Usuário*. Editora Campus.

Fonseca, F. T., Pitangueira, R. L., & Filho, A. V., 2004. Implementação de modelos estruturais de barras como casos particulares do método de elementos finitos. *VI SIMMEC*.

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.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J., 1995. *Design Paterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Grand, M., 1998. *Patterns in Java - Volume 1*. Jonh Wiley and Sons.

Horstmann, C. S. & Cornell, G., 2000. *Core Java 2 Volume II - Advanced Features*. Editora Prentice Hall.

Lozano, F., 2004. Relatórios corporativos. *Java Magazine*, , n. 13, pp. 20–32.

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.

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.

Pitangueira, R. L. S., 1998. *Mecânica de Estruturas de Concreto com Inclusão de Efeitos de Tamanho e Heterogeneidade*. PhD thesis, PUC - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

Rowe, G. W., 2001. *Computer Graphics with Java*. Palgrave.

Soriano, H. L. & Lima, S. S., 1999. *Método de Elementos Finitos em Análise de Estrutura*. Universidade Federal do Rio de Janeiro.

Tidwell, D., 2001. *Mstering XML Transformations - XSLT*. O'Reilly and Associates.