WILLIAM WAGNER MATOS LIRA

Um Sistema Integrado Configurável para Simulações em Mecânica Computacional

Dissertação apresentada ao Departamento de Engenharia Civil da PUC-Rio como parte dos Requisitos para a obtenção do título de Mestre em Ciências em Engenharia Civil: Estruturas.

Orientador: Luiz Fernando Ramos Martha
Co-orientador: Marcelo Tilio Monteiro de Carvalho

Departamento de Engenharia Civil
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 30 de Setembro de 1998

Esta dissertação é dedicada com muito amor e respeito ao meu avô Haroldo que, ao longo do desenvolvimento deste trabalho, nos deixou. Muito obrigado por tudo, vovô.

Agradecimentos

Aos meus pais Ferreira e Ana, as minhas irmãs Déia e Karllinha, a minha avó Amparo e a todos os meus tios, tias, primos e primas, pelo apoio, incentivo e paciência demonstrado ao longo do desenvolvimento deste trabalho.

Ao professor Luiz Fernando Martha e ao amigo Marcelo Tilio Monteiro de Carvalho, orientadores deste trabalho, pelo incentivo, orientação e, principamente, pela amizade e dedicação demonstrada ao longo do desenvolvimento deste trabalho.

Aos amigos Eduardo Setton e Carlos Vitor pela amizade e colaboração nas várias etapas deste trabalho.

Aos professores Roberaldo Carvalho de Souza, Severino Marques, Williams Soares Batista, Dilze Marques, Márcio Barbosa, Aline Ramos e João Barbirato, da Universidade Federal de Alagoas, pela amizade e pelos ensinamentos passados durante o curso de graduação; ao grupo PET de Engenharia Civil pelos ensinamentos e grandes momentos vividos na graduação.

Aos amigos da família Craveiro (Léo, Alexandre e Gustavo) pela grande amizade consolidada durante esses anos de convívio no Rio de Janeiro.

Aos amigos Daniella, Áurea, Evandro, Andréia Diniz, Claudinha, Denyse, Leornardo Bello, Alexandre Miranda, Marco De La Quintana, Julio Macias, Márcio, Paulão, Stoessel, Rodacki, Thadeu, João Batista, Joaquim, Camilo, Ivan e Lula pelos chopps, conversas, discursões e, principalmente, pela nossa amizade.

Aos amigos da PUC: Ângela, Araken, Deane, Regina, Isabella, Karina, Marco Aurélio, Ricardo Couto, Alexandre, Salete, Yvelline, Beto, Sandoval, Aellington, Humberto, Ana Cristina e Turco.

Aos amigos do TeCGraf: Cacá, Waldemar, João "*Acaí*" Luiz, Eduardo "*Boi*" Nobre, Raquel, Arlindo, Mônica, André Costa, Carla, Anselmo, Adelailson, Mediano, Antônio Miranda, Ana

Elisa, Renato Borges, André Derraik, Anna Hester, Cecília, Renato Cerqueira, Claúdia, Cassino, Clínio, Almendra, Neil, Sedrez, Peter, Gonzaga, Juliana, Ovídio, Scuri, Anderson, Anderson Oliveira, Becken, Paula, Flávio, Yeda Audiléia e Claudinei.

Aos engenheiros da Petrobras Álvaro Maia e Claúdio Amaral pela colaboração e paciência no desenvolvimento deste trabalho.

A Ana Roxo e a todos os funcionários do Departamento de Engenharia Civil da PUC-Rio.

Aos ilustres flamenguistas Tilio, Paulão, Márcio, Vitor, Ivan, Áurea e Andréia por ter dividido os grandes momentos proporcionados pelo Flamengo, e um agradecimento especial ao meu pai, por ter me dado a honra de ser torcedor do maior clude de futebol do mundo.

Aos amigos da torcida arco-íris Camilo, Setton, Lula, Thadeu, Joaquim, Stoessel, Evandro e Jõao Batista, por proporcionar grandes alegrias nos momentos de vitória do mengão.

À CAPES e ao CNPQ pelo apoio financeiro durante os cursos de graduação e mestrado.

Resumo

Este trabalho dá continuidade ao desenvolvimento de uma metodologia para extensão e configuração de aplicações gráficas interativas utilizadas em simulações baseadas no método dos elementos finitos (MEF). Novos atributos requisitados pelos módulos de análise podem ser facilmente incluídos nos pré- e pós-processadores. Os atributos são definidos, através de uma linguagem de extensão interpretada relativamente simples, em um arquivo de configuração. A configuração e extensão é feita a partir da criação de classes e métodos, no contexto da programação orientada a objetos, de atributos da simulação. Esta metodologia foi implementada no desenvolvimento de um sistema integrado para simulações numéricas de problemas bidimensionais em geotecnia pelo MEF.

A tese apresenta, inicialmente, uma discussão sobre a evolução dos sistemas utilizados para simulações numéricas na mecânica computacional, desenvolvidos no Departamento de Engenharia Civil da PUC-Rio, indicando os problemas existentes. A seguir, é apresentada uma extensão da arquitetura do módulo utilizado para o gerenciamento e extensão configurável dos atributos (ESAM), que, originalmente, considerava uma abordagem onde atributos só podiam ser aplicados a entidades geométricas. A nova arquitetura permite que atributos também possam ser aplicados diretamente em nós e elementos de uma malha de elementos finitos.

O sistema implementado através da integração dos módulos utilizados para pré-processamento, análise numérica e pós-processamento com o módulo de gerenciamento de atributos resultou em um sistema bastante flexível, podendo ser estendido por um usuário configurador para diversos outros tipos de simulação.

As fases envolvidas em um processo de simulação na mecânica computacional (definição da geometria, especificação dos atributos, geração da malha de elementos finitos, análise numérica e visualização dos resultados) são ilustradas em um exemplo de aplicação do sistema proposto.

Abstract

This work continues the development of a methodology for extension and configuration of interactive graphics applications utilized on a finite element simulations. New attributes necessary for the analysis modules can be easily included in pre- and post-processor modules. The attributes are defined, through a relatively simple interpreted extension language, in a configuration file. The extension and configuration is performed through the creation of classes and methods, in the context of object oriented programming, of simulation attributes. This methodology was implemented in development of an integrated system for two-dimensional numerical simulation of geotechnical problems by the finite element method.

The dissertation presents, initially, a discussion on the evolution of the systems used for numerical simulations in computational mechanics, developed in Department of Civil Engineering of PUC-Rio, pointing the existing problems. In the sequence, it is presented an extension of architecture of module that manager extension the simulation attributes (ESAM), that, originally, considerated only an approach where attributes could be applied in geometrical entities. The new architecture permits that attributes may also be applied directly to nodes and elements of a finite element mesh.

The system implemented through the integration of the pre-processing, numerical analysis and post-processing modules, with the attribute management module resulted in a very flexible system, that can be extended for several other types of simulations.

The phases involved in a simulations of a computational mechanics process (geometry defined, attributes specification, automatic mesh generation, numerical analysis and visualization of results) are illustrated in an application example of the proposed system.

Sumário

Lista de Figuras

1. Introdução

No contexto da mecânica computacional, o método numérico mais utilizado tem sido o método dos elementos finitos (MEF) [Bathe 1982]. Isto se deve à sua grande versatilidade, qualidade de resultados e, principalmente, sua relativa facilidade de implementação. O MEF se baseia em um modelo numérico obtido a partir da discretização do domínio do problema, juntamente com informações adicionais associadas a essa discretização, necessárias para uma completa definição do problema físico. Tais informações adicionais são denominadas atributos. A discretização, denominada malha de elementos finitos, consiste em um conjunto de nós ou vértices (pontos com coordenadas) e um conjunto de elementos finitos com uma topologia pré-definida (triângulos ou quadriláteros, por exemplo). Os elementos são definidos por uma lista de conectividade de seus vértices (seqüência de vértices que pertencem a cada elemento).

Geralmente, a metodologia adotada na discretização do domínio é comum para os diversos tipos de problemas que podem ser analisados pelo MEF. Os atributos, entretanto, são específicos para cada tipo de simulação. Por exemplo, em uma análise de tensões de um problema estrutural, alguns atributos se referem às cargas distribuídas, às condições de apoio e aos deslocamentos prescritos. Por outro lado, em uma análise térmica os atributos que particularizam o problema são, entre outros, o fluxo de calor, o gradiente térmico e a temperatura prescrita no contorno do modelo. A Figura 1.1 mostra dois problemas distintos da mecânica computacional que possuem a mesma geometria, porém diferentes atributos associados aos respectivos modelos.

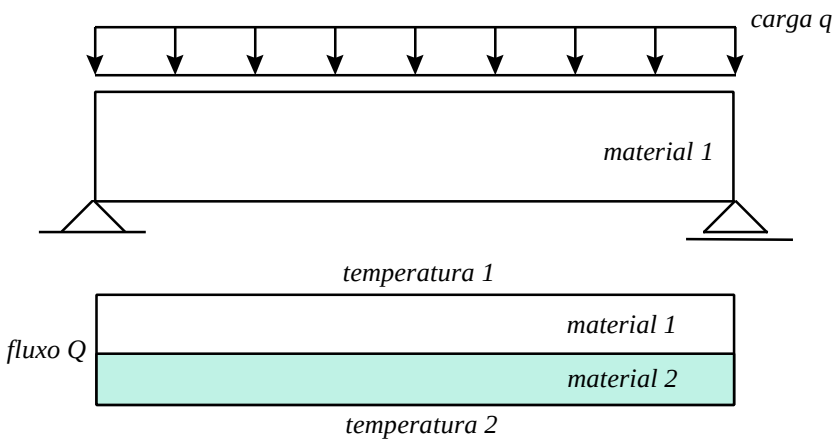

Figura 1.1 - Problemas de mecânica computacional.

Na maioria dos sistemas computacionais utilizados para simulações numéricas que utilizam o MEF, as informações referentes aos atributos estão intrinsecamente ligadas aos processos de modelagem geométrica, geração de malha, análise numérica e visualização de resultados, através da estrutura de dados implementada nos seus códigos. Nesses sistemas, a incorporação de novos tipos de atributos (por exemplo, novos tipos de materiais) não é uma tarefa trivial, pois, geralmente, os códigos computacionais existentes são grandes e complexos.

É desejável, então, que os atributos possam ser configurados pelo usuário da aplicação, de acordo com o tipo de análise desejada, sem que este usuário precise acessar as informações do código computacional que implementa as outras etapas do processo de simulação. Isso permite que um sistema possa ser facilmente adaptado para diferentes tipos de simulações. Esta tese apresenta um Sistema Integrado Configurável de Geotecnia para múltiplas análises (SIGMA) utilizado para simulações numéricas bidimensionais pelo MEF, que oferece a possibilidade de configuração, pelo usuário, dos atributos da simulação. Atualmente, existem alguns outros programas que oferecem, de alguma maneira, este tipo de funcionalidade. Os sistemas PATRAN [MSC/Patran 1996] e ANSYS [ANSYS 1997] utilizam linguagens de programação próprias que permitem a configuração dos atributos específicos de uma simulação e possibilita, de maneira fácil, a sua integração com outros *softwares*. Estes sistemas também permitem que o usuário configure a sua interface gráfica para realizar tarefas específicas referentes a um determinado tipo de simulação.

1.1 Histórico

Uma das preocupações da linha de pesquisa de Computação Gráfica do Departamento de Engenharia Civil da PUC-Rio, na qual esta dissertação está inserida, é a busca da automação real na simulação de problemas da mecânica computacional. Neste contexto, a questão referente a configuração dos atributos de uma simulação, conforme discutido na seção anterior, assume uma importância fundamental.

Neste sentido, o trabalho de Carvalho [1995] concebeu uma arquitetura de *software* que possibilita a criação de ferramentas configuráveis para simulações numéricas na mecânica computacional, no que se refere ao tratamento de atributos.

Anteriormente a este trabalho, a automação do processo de simulação baseava-se apenas na integração dos módulos de pré-, análise e pós-processamento através de um arquivo de dados com formato único padrão, denominado arquivo neutro (*NeutralFile*) [TECG3 1992], que contém informações relativas à malha de elementos finitos, tais como coordenadas dos nós e conectividade dos elementos, informações dos atributos associados a essa malha e sobre os resultados obtidos após a realização da análise numérica. Um exemplo era o sistema utilizado para simulações numéricas de problemas bidimensionais da mecânica computacional pelo MEF (Figura 1.2). Este sistema era composto pelo módulo de pré-processamento MTool (*Bidimensional Mesh Tool*) [TECG1 1992], responsável pela definição do modelo geométrico do problema e geração da malha de elementos finitos, pelo programa de análise FEMOOP (*Finite Element Oriented Object Programming*) [Guimarães 1992, Martha *et al.* 1996], e pelo módulo de pós-processamento MView (*Bidimensional Mesh View*) [TECG2 1992].

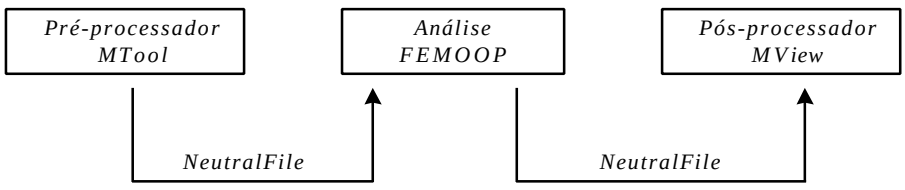

Figura 1.2 - Arquitetura de um sistema para simulações bidimensionais na mecânica computacional pelo MEF.

Neste sistema o processo de simulação se inicia com a definição do modelo geométrico do problema, formado por componentes com geometria bem definida e por relações de conectividade entre esses componentes (topologia), e a especificação dos atributos. Sobre o modelo geométrico é gerada a malha de elementos finitos. Os atributos associados às entidades geométricas são, então, mapeados, automaticamente, para os nós e elementos do modelo discretizado. Por exemplo, os atributos associados a uma curva do modelo geométrico, são mapeados para os nós gerados nesta curva, a partir da discretização do modelo (Figura 1.3).

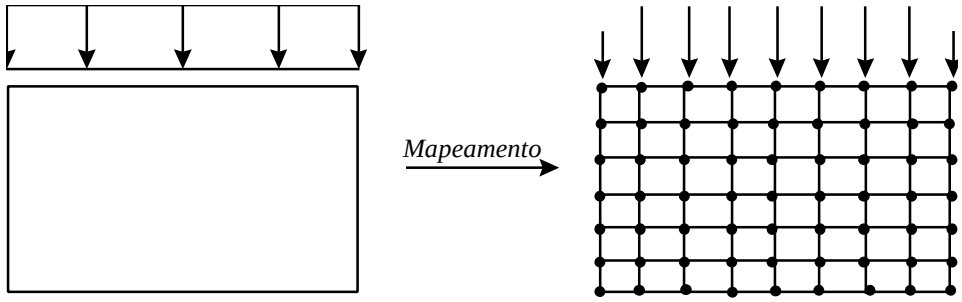

Figura 1.3 - Mapeamento de uma carga distribuída no modelo geométrico para a malha de elementos finitos associada.

Com as informações da malha e dos atributos, é gerado o arquivo de dados de entrada do programa de análise FEMOOP. Este programa é baseado em programação orientada a objetos e abrange diversos tipos de análise estrutural. Após a análise numérica é gerado um arquivo de saída com os resultados obtidos que será lido pelo pós-processador MView.

O tipo de abordagem utilizado na etapa de modelagem desse sistema, no qual os atributos são associados às entidades geométricas do modelo e a discretização herda, automaticamente, estes atributos é chamado de modelagem baseada em geometria.

Em alguns sistemas de simulação pelo MEF, principalmente os mais antigos, os atributos do modelo são aplicados diretamente às entidades da malha (nós e elementos finitos). Este enfoque, que é natural para os programas de análise (FEMOOP, por exemplo), oferece limitações, principalmente quando há necessidade de redefinição da

malha. Este trabalho refere-se a modelagem baseada em malha como sendo o processo de simulação onde atributos são aplicados diretamente às entidades de malha.

A modelagem baseada em geometria oferece diversas vantagens, dentre as quais pode-se citar um melhor suporte para a automação do processo de modelagem, incluindo geração automática de malha e análise adaptativa (os atributos são associados às entidades geométricas do modelo e não precisam ser redefinidos no processo de refinamento da malha de elementos finitos), e a simplificação da geração do modelo para a simulação [Shephard *et al.* 1988].

Uma prática bastante comum em ambientes de projeto e/ou pesquisa é que o programa de análise, no caso o FEMOOP, seja estendido a partir da adição de novos tipos de atributos para tentar resolver uma quantidade de problemas cada vez maior. Como o pré-processador, no caso o MTool, é responsável pela geração do arquivo de dados contendo as informações do modelo (malha de elementos finitos e atributos associados a ela) que será analisado pelo FEMOOP, é necessário, que esses atributos também sejam incorporados ao pré-processador. Três tarefas devem ser realizadas para incluir um novo atributo neste módulo: a primeira tarefa refere-se à necessidade de disponibilizar uma interface que possibilite a captura dos dados relacionados ao atributo; a segunda diz respeito à necessidade de se incorporar este atributo na estrutura de dados do pré-processador, permitindo que ele possa ser associado às entidades do modelo; finalmente, a terceira tarefa está relacionada com a necessidade de se inserir os atributos na rotina responsável pela geração do arquivo de dados que será lido pelo programa de análise. Como geralmente o código computacional de um pré-processador é bastante sofisticado, a realização destas tarefas é relativamente complexa.

Com o objetivo de solucionar este problema, Carvalho [1995] propôs uma arquitetura de *software* que permite a concepção de sistemas nos quais é possível *configurar* os atributos específicos a um determinado problema com um alto grau de abstração. Neste contexto, entende-se por alto grau de abstração a possibilidade de configurar os atributos sem a necessidade de conhecer o código computacional da aplicação. A configuração dos atributos é feita através de arquivos que contém instruções em uma

linguagem de programação extensível (linguagens utilizadas para estender ou configurar uma aplicação para fins particulares) bastante simples. Esta linguagem é interpretada, permitindo que a configuração dos atributos seja feita sem a necessidade de recompilação do sistema. Carvalho implementou um sistema, denominado ESAM (*Extensible System Attributes Management*), baseado nesta arquitetura. Como a motivação para desenvolvimento do sistema ESAM surgiu a partir da dificuldade de expansão (criação de atributos) do sistema para simulação descrito anteriormente (MTool/FEMOOP/MView), sua abordagem original foi de utilizar uma modelagem baseada em geometria, por ser a mesma utilizada pelo MTool.

Em princípio, qualquer linguagem de extensão poderia ser utilizada para implementar a arquitetura do ESAM, tais como Phyton [Rossum 1993], Tcl [Ousterhout 1994] ou Lua [Ierusalimschy *et al*. 1994]. No primeiro protótipo desenvolvido por Carvalho, optou-se pela linguagem Tcl. Atualmente, utiliza-se a linguagem Lua devido à simplicidade de sua sintaxe e à facilidade de incorporação dos conceitos de programação orientada a objetos.

Utilizando o sistema ESAM proposto por Carvalho, Silveira [1995] apresentou o protótipo de um sistema computacional para modelagem bidimensional configurável na mecânica computacional, especificamente para problemas de elasticidade (análise de tensões) e difusão térmica. Esse sistema oferece um ambiente para a realização de simulações numéricas, utilizando o MEF, com uma grande flexibilidade em relação a criação e a manipulação dos atributos utilizados. O processo de simulação, que utiliza uma modelagem baseada em geometria (mesmo procedimento do MTool, descrito anteriormente), é iniciado com a definição da geometria e especificação dos atributos do problema. Neste sistema, no entanto, os atributos são gerenciados pelo ESAM, de tal forma que novos atributos incorporados ao módulo responsável pela análise numérica podem ser facilmente implementados pelo usuário no pré-processador. Os resultados obtidos através da análise numérica são visualizados por um pós-processador gráfico. Este pós-processador apresenta uma característica que é comum à maioria dos pós-processadores conhecidos: não existe uma informação explícita sobre as entidades geométricas (curvas de contorno, por exemplo) do modelo. A informação geométrica é obtida através das coordenadas dos nós da malha, o que

impede a utilização do ESAM, que adota o enfoque de uma modelagem baseada em geometria, para gerenciar os seus atributos. A Figura 1.4 ilustra a arquitetura deste sistema.

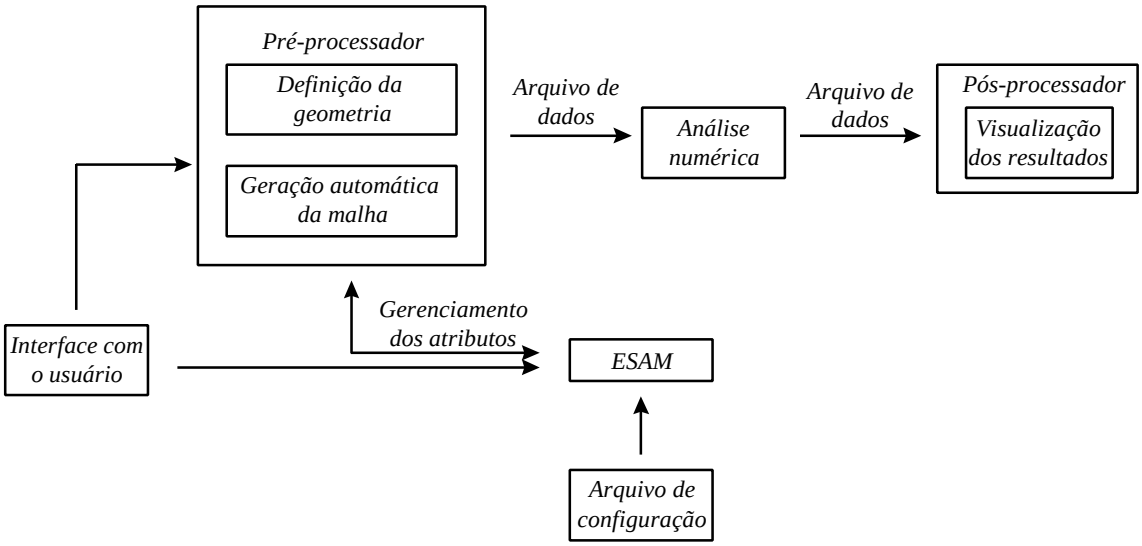

Figura 1.4 - Representação do sistema computacional para modelagem bidimensional configurável, apresentado por Silveira [1995].

No sistema antigo, no entanto, o arquivo de dados lido pelo programa de análise era gerado pelo próprio pré-processador (MTool, no caso). Isto significa que qualquer modificação neste arquivo implica em uma alteração no código computacional que implementa o MTool. No sistema apresentado por Silveira, o arquivo é gerado pelo ESAM e pode ser configurado pelo próprio usuário que define os atributos da simulação. Neste sistema, a comunicação entre o programa de análise e o pós-processador é realizada através de um arquivo de dados contendo as informações referentes à malha de elementos finitos e aos resultados obtidos da análise. Entretanto, como o ESAM utilizava uma modelagem baseada em geometria, não foi possível incorporá-lo ao pós-processador para gerenciar os seus atributos, conforme pode ser visto na Figura 1.4. Com isso, os atributos associados às entidades geométricas do modelo não são visualizados no módulo de pós-processamento do sistema.

1.2 Motivação e Objetivos

Como foi dito na seção anterior, o sistema apresentado por Silveira [1995] não permite que os atributos definidos no módulo de pré-processamento sejam visualizados no pós-processador. Entretanto, em uma simulação numérica dentro de um ambiente gráfico-interativo, é desejável que exista uma integração entre os módulos envolvidos nesta simulação. Ou seja, o modelo definido no pré-processador deve ser totalmente representado, inclusive com seus atributos, no módulo de pós-processamento juntamente com os resultados obtidos pela análise numérica.

Portanto, em função das deficiências apontadas acima, esta dissertação propõe uma extensão da arquitetura do sistema ESAM que, originalmente, permitia apenas a aplicação de atributos às entidades geométricas do modelo. A arquitetura proposta também possibilita que os atributos sejam associados diretamente às entidades da malha de elementos finitos (nós e elementos). Por esse motivo, a nova arquitetura permite que pós-processadores, que possuem informações relativas apenas à malha de elementos finitos, possam obter informações referentes aos atributos aplicados no modelo geométrico. Estas extensões do sistema ESAM viabilizaram o principal objetivo desta dissertação : a criação de um sistema integrado configurável para simulações numéricas na mecânica computacional. A Figura 1.5 ilustra a arquitetura do sistema proposto.

O sistema ESAM foi, então, incorporado ao pré-processador MTool e ao pós-processador MView, tornando-os extensíveis e configuráveis. A incorporação do ESAM ao módulo de pós-processamento MView só foi possível a partir da nova arquitetura do sistema ESAM, que utiliza uma modelagem baseada em malha. O pré-processador MTool e o pós-processador MView foram integrados aos programas de análise, desenvolvidos por engenheiros do Grupo de Geotecnia do Centro de Pesquisa da Petrobras (PETROBRAS/CENPES/DIPREX/SEDEM), originando o Sistema Integrado de Geotecnia para Múltiplas Análises (SIGMA) que é responsável por simulações numéricas bidimensionais de geotecnia (escavação de poços destinados à mineração subterrânea, análise de estacas, etc), através do MEF. Para desenvolver o SIGMA, foi necessário configurar, através do sistema ESAM, os atributos

implementados nos programas de análise. Neste trabalho, os detalhes referentes aos programas de análise numérica utilizados não são considerados.

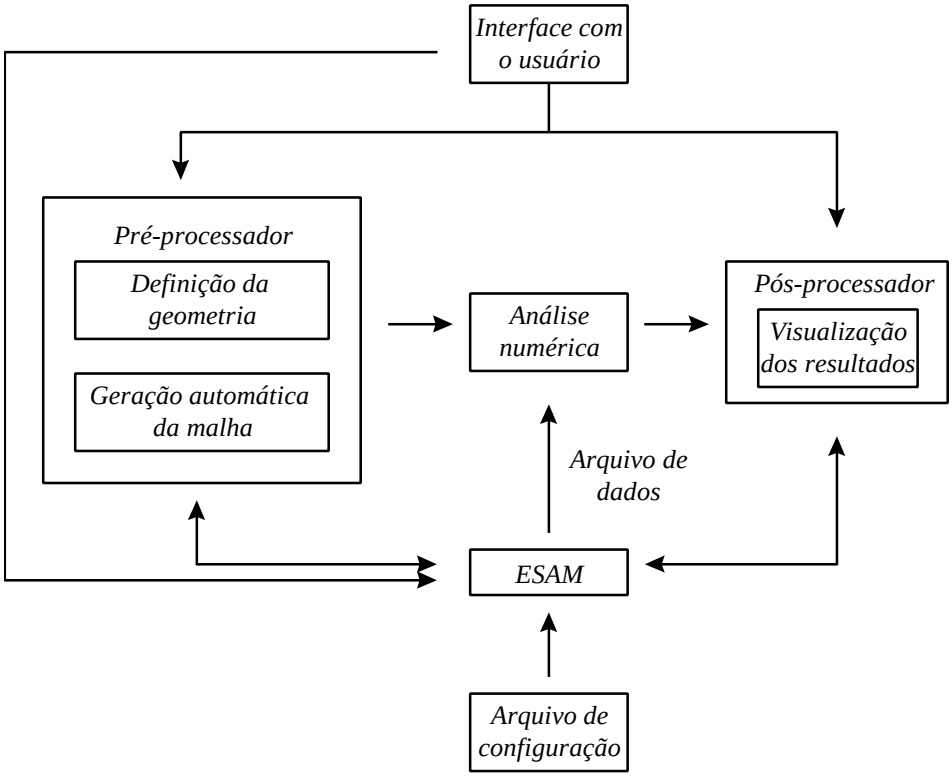

Figura 1.5 - Arquitetura do sistema integrado configurável para simulações numéricas.

1.3 Organização da Tese

Esta dissertação foi dividida em 5 capítulos. Nesta seção, é apresentado uma visão geral do trabalho, descrevendo resumidamente o conteúdo de cada capítulo.

O capítulo 2 apresenta a nova arquitetura proposta para o sistema ESAM. Apresenta-se uma descrição mais detalhada deste sistema, desenvolvido segundo o paradigma da programação orientada a objetos, ilustrando a nova organização das classes e suas funcionalidades dentro do processo de simulação numérica.

No capítulo 3, descreve-se o sistema integrado de geotecnia para múltiplas análises, desenvolvido neste trabalho. As etapas envolvidas na simulação (modelagem, geração

da malha de elementos finitos, análise e visualização) são detalhadas, dando ênfase às informações relacionadas aos atributos da simulação, que foram configurados com base nos programas de análise utilizados. Alguns exemplos de configuração dos atributos são ilustrados neste capítulo.

O capítulo 4 mostra um exemplo de uma simulação bidimensional de geotecnia utilizando o sistema proposto neste trabalho.

Finalmente, no capítulo 5 são apresentadas as conclusões deste trabalho, bem como sugestões para trabalhos futuros.

2. Arquitetura do Sistema de Configuração de Atributos

No capítulo anterior foi mencionado que a arquitetura original do sistema ESAM [Carvalho 1995, Silveira 1995] adotava apenas o enfoque de modelagem baseada em geometria, onde os atributos da simulação são associados às entidades geométricas do modelo. Como foi dito, esse tipo de abordagem oferece algumas vantagens, tais como um melhor suporte para a automação completa do processo e a simplificação da geração do modelo para a simulação.

A comunicação entre os pré- e pós-processadores que utilizam o sistema ESAM para gerenciar seus atributos é realizada através de um arquivo gerado por esse sistema. Nos pré-processadores, estes arquivos são compostos pelos atributos e pelas entidades geométricas aos quais os atributos foram aplicados. Entretanto, como a maioria dos pós-processadores não possui uma informação explícita sobre as entidades geométricas do modelo, é necessário que os pré-processadores sejam capazes de gerar um arquivo contendo os atributos associados às entidades da malha (elementos e nós) e as próprias entidades da malha.

Para resolver estes problemas, optou-se por expandir a arquitetura original do sistema ESAM, para dar suporte a uma modelagem baseada em malha e permitir aos módulos de pré-processamento gerar arquivos que possam ser interpretados pelos pós-processadores. Na verdade, a nova arquitetura do ESAM não apenas permite a sua utilização em aplicações que utilizam quaisquer um dos tipos de abordagem (baseada em geometria e baseada em malha), mas também pode ser utilizado em aplicações que façam uso dos dois tipos de modelagem simultaneamente.

2.1 Arquitetura do Ambiente de Configuração

O ambiente de configuração proposto por Carvalho [1995] considera apenas os aspectos relacionados com a definição dos atributos da simulação. Os detalhes referentes à modelagem geométrica, à geração da malha de elementos finitos e à visualização dos resultados de uma análise numérica não são considerados. Assume-se que estes serviços são fornecidos por alguma aplicação.

De modo a permitir que a especificação desses atributos seja feita de forma independente da aplicação, a comunicação entre o gerenciador de atributos e a aplicação deve ser feita através de um mecanismo que mantenha a independência dos módulos envolvidos na tarefa. Ou seja, o módulo responsável pelo gerenciamento dos atributos não deve ter conhecimento sobre as características específicas da aplicação, assim como a aplicação não deve saber da parte referente à especificação dos atributos.

Neste sentido, a arquitetura proposta divide a aplicação em três partes. Uma parte representa os serviços ou tecnologia da aplicação, composta pelos aspectos que são independentes do tipo de simulação, tais como modelagem geométrica, geração da malha e visualização dos resultados. Uma outra parte refere-se aos aspectos específicos de cada simulação, ou seja, a especificação dos atributos. A terceira parte refere-se a um módulo responsável pela comunicação entre as duas partes anteriores, que permite o acesso à base de dados da aplicação com um alto grau de abstração.

A seguir é realizada uma descrição, com as atualizações propostas neste trabalho, do sistema ESAM (*Extensible System of Attributes Managment*). Uma descrição mais detalhada da arquitetura inicial deste ambiente de configuração pode ser vista em [Carvalho 1995].

2.2 Descrição do Sistema ESAM

O sistema ESAM consiste de uma coleção de classes (*Core Classes*), no contexto da programação orientada a objetos, e de um conjunto de funções que são responsáveis pela interface entre as *Core Classes* e a aplicação. O sistema ESAM possui também um conjunto de funções que oferecem os serviços (funcionalidades do sistema) necessários para a configuração da aplicação. Estes serviços são oferecidos através de um conjunto de rotinas que devem ser incorporadas à aplicação. Os serviços, entretanto, não incorporam a semântica da aplicação. A Figura 2.1 mostra a arquitetura proposta, onde os retângulos com linhas tracejadas correspondem aos módulos que não fazem parte do sistema ESAM, ou seja, o módulo referente aos serviços da aplicação e o arquivo de configuração dos atributos. Neste arquivo é realizada a especificação dos novos tipos de atributos para uma simulação específica. As *Core Classes* representam uma abstração das entidades envolvidas no processo de simulação, ou seja, representam as entidades geométricas e da malha (*Modeling Classes*) e os atributos de simulação (*Attribute Classes*). A Figura 2.2 ilustra a organização das *Core Classes*. Estas classes estão implementadas no ESAM utilizando a linguagem de extensão Lua.

A especificação de atributos pode requisitar informações referentes à aplicação que estão armazenadas na sua base de dados. O meio como as informações são obtidas é, portanto, dependente da implementação desta base de dados. Neste sentido, é desejável que as informações sobre a aplicação sejam obtidas através de chamadas procedurais e não através de acesso explícito à sua estrutura de dados. Ou seja, deve existir uma interface entre a base de dados da aplicação e a parte configurável da aplicação. Esta interface é realizada através de um conjunto de funções, denominadas funções MIS (*Modeling Interface Specification*), que representam uma camada de abstração sobre a base de dados da aplicação (veja Figura 2.1).

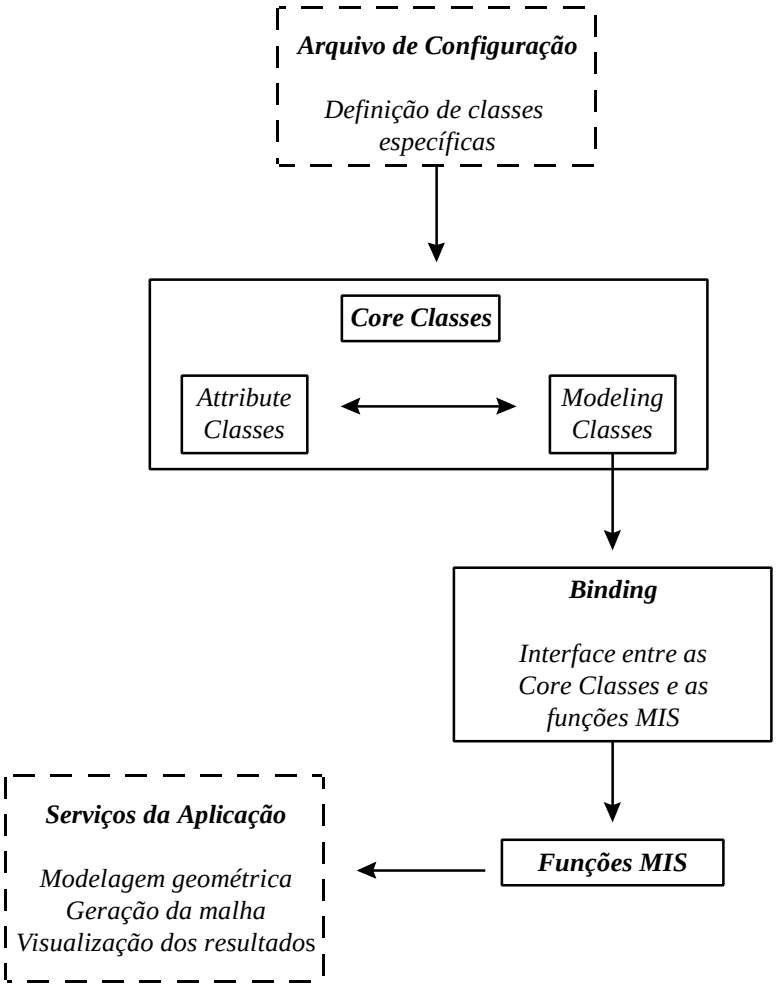

Figura 2.1 - Arquitetura do Sistema ESAM.

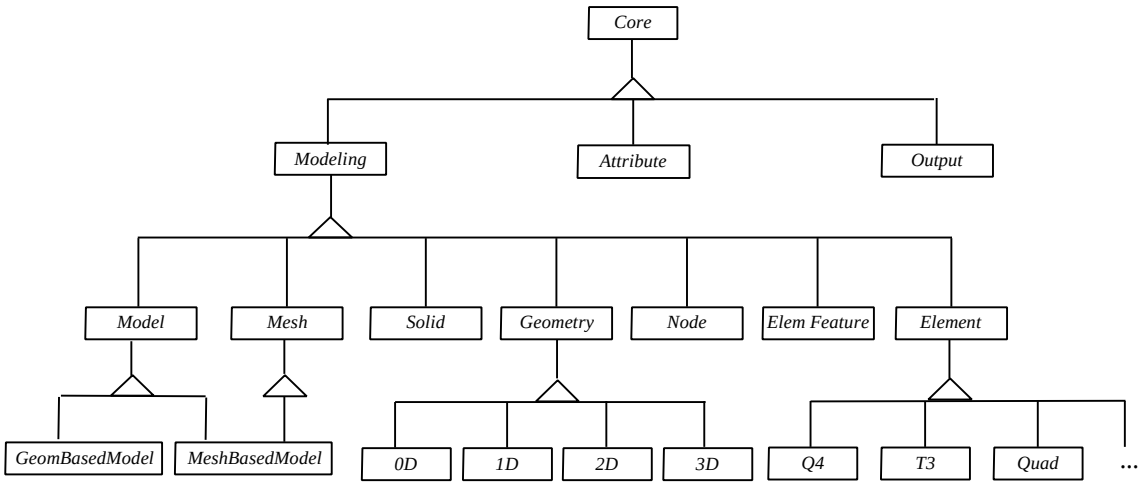

Figura 2.2 - Organização das *Core Classes*.

O módulo *Binding* é responsável pela ligação entre a parte configurável e a aplicação. Este módulo representa a interface responsável pelo acesso da parte configurável à aplicação. Na aplicação, as funções MIS disponibilizam o acesso aos objetos da simulação aos quais os atributos estão relacionados. O módulo *Binding* permite que o usuário responsável pela implementação da aplicação escreva as funções MIS sem conhecer as características da linguagem de configuração utilizada e do sistema de configuração. Por outro lado, o configurador acessa os dados da aplicação, sem conhecer o seu código, através de métodos das classes definidas no próprio ambiente de configuração.

2.3 Detalhamento das *Core Classes* do Sistema ESAM

Esta seção contém uma descrição atualizada das *Core Classes* do sistema ESAM após as contribuições deste trabalho.

Conforme mencionado na seção anterior, as *Core Classes* formam uma biblioteca de classes que representam as entidades envolvidas em uma simulação. Essas entidades dependem do tipo de abordagem que é adotada na simulação. Em um processo de simulação utilizando uma modelagem baseada em geometria, os atributos de simulação são associados diretamente às entidades geométricas do modelo. O conjunto de entidades geométricas compõem os sólidos. O modelo geométrico pode ser constituído por um ou mais sólidos. A partir da definição da geometria, é realizada a geração automática da malha de elementos finitos, criando-se elementos e nós, que associados aos seus respectivos atributos, compõem o modelo numérico. Posteriormente, é realizado o mapeamento destes atributos para a malha de elementos finitos, onde as entidades discretizadas (elementos e nós) herdam automaticamente os atributos associados ao modelo geométrico. Neste processo, as entidades envolvidas na simulação são o modelo geométrico (representado no ESAM pela classe *GeomBasedModel*), os sólidos (classe *Solid*), as entidades geométricas (classe *Geometry*), os atributos (classe *Attribute*), a malha de elementos finitos (classe *Mesh*), os elementos (classe *Element*) e os nós (classe *Node*).

Em uma modelagem baseada em malha, os atributos da simulação ficam associados diretamente às entidades de discretização (nós e elementos finitos). Neste tipo de abordagem, não existe a definição de sólidos e entidades geométricas do modelo. Neste processo, as entidades envolvidas na simulação são o modelo numérico (representado no ESAM pela classe *MeshBasedModel*), os atributos (classe *Attribute*), os elementos (classe *Element*) e os nós (classe *Node*).

A seguir, são descritas as *Core Classes* individualmente, mostrando as suas características e representações. As variáveis e os métodos citados tem como objetivo ilustrar o que cada classe representa, estabelecendo as relações entre os objetos das classes do sistema, bem como as relações de hierarquia existentes nesta arquitetura. Os diagramas mostrados nesta seção têm por finalidade estabelecer as relações entre classes e objetos definidos no ESAM. Estes diagramas de classes e objetos são baseados na técnica OMT (*Object Modeling Technique*) [Rumbaugh 1991]. Os diagramas de classes descrevem as classes e as relações estáticas entre elas, podendo descrever, também, suas estruturas. A Figura 2.3 mostra um exemplo da notação OMT para relações estáticas entre classes. Esta figura indica que as classes *GeomBasedModel* e *MeshBasedModel* são subclasses (derivadas) da classe *Model*. Além disso, mostra que a classe *Model* possui os métodos *new* e *delete* associados a ela.

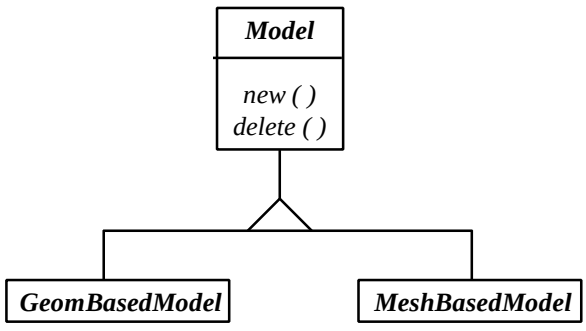

Figura 2.3 - Relações estáticas entre classes, segundo a OMT.

Outro exemplo de diagrama de classe importante é mostrado na Figura 2.4, onde um objeto da classe *Element* possui referência para um objeto da classe *Geometry*. A

adição de um ponto na extremidade da seta do diagrama indica que um objeto da classe agrega múltiplos objetos, como é ilustrado na Figura 2.5, onde os objetos da classe *Element* possuem referências para múltiplos objetos da classe *Attribute*.

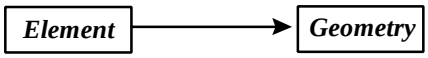

Figura 2.4 - Relações simples entre classes.

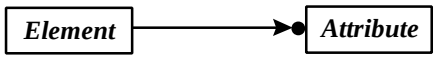

Figura 2.5 - Relações múltiplas entre classes.

Os diagramas de objetos possuem relações exclusivas entre objetos. Segundo a OMT, os objetos são denominados por "(*class*)", onde *class* representa o nome da classe do objeto. A Figura 2.6 mostra um exemplo de diagrama de objetos. O objeto (*Element*) contém variáveis, *Attribute*[0] e *Attribute*[1], que referenciam os objetos, (*attr_1*) e (*attr_2*), da classe *Attribute*. Além disso, o objeto (*Element*) possui o método *attach* associado a ele.

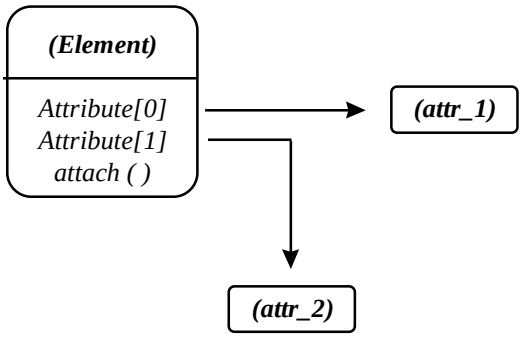

Figura 2.6 - Relações entre objetos, segundo a OMT.

2.3.1 Classe *Model*

A classe *Model* representa a abstração do modelo utilizado em uma simulação em mecânica computacional (*GeomBasedModel*, para uma modelagem baseada em geometria, ou *MeshBasedModel*, para um modelagem baseada em malha). A classe

Model é definida como sendo uma classe virtual, isto é, uma classe que não pode ser instanciada. Os objetos das classes *GeomBasedModel* ou *MeshBasedModel* contêm referências para objetos da classe *Attribute*, representando os atributos do modelo. Esses atributos são separados em dois grupos distintos. O primeiro grupo contém referências para os atributos que são associados às entidades (geométricas ou de malha) do modelo. O segundo, contém referências para os atributos que são associados ao modelo (tipo de análise, por exemplo). A classe *Model* possui métodos que permitem a manipulação destes atributos, tais como métodos para criar (*createattrib*) ou modificar (*modifyattrib*) atributos. A Figura 2.7 mostra uma representação gráfica da classe *Model*.

Como a classe *Model* não possui informações relativas às entidades geométricas e de malha do modelo, ela não contêm métodos que permitem associar atributos criados pela aplicação a essas entidades. As informações sobre as entidades do modelo estão localizadas nas subclasses (classes *GeomBasedModel* e *MeshBasedModel*) desta classe, onde esses métodos são definidos.

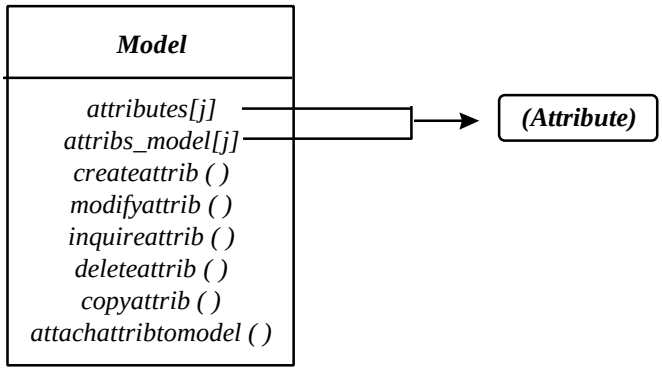

Figura 2.7 - Representação gráfica da classe *Model*.

2.3.2 Classe *Mesh*

Os objetos da classe *Mesh* representam abstrações da malha de elementos finitos de uma simulação. Um objeto desta classe possui uma lista de objetos da classe *Element* e uma lista de objetos da classe *Node* que representam, respectivamente, abstrações dos nós e elementos da malha. Os objetos desta classe não possuem informações sobre os

atributos associados as entidades da malha. Estes atributos estão diretamente associados aos objetos da classes *GeomBasedModel* ou *MeshBasedModel*, que são descritos a seguir. A classe *Mesh* possui métodos que respondem informações relacionadas à malha de elementos finitos, tais como o método *getnodelist*, que fornece a lista de nós da malha, ou o método *writemeshtofile*, que escreve as informações relativas à malha em um arquivo de dados. A Figura 2.8 mostra a representação gráfica dos objetos desta classe.

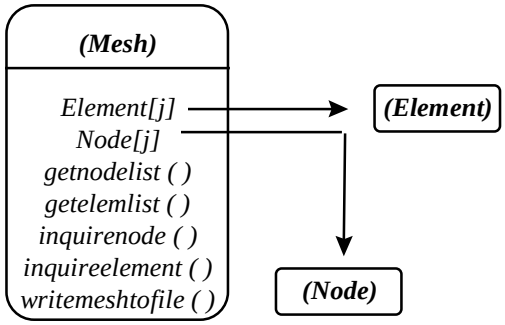

Figura 2.8 - Representação gráfica dos objetos da classe *Mesh*.

2.3.3 Classe *GeomBasedModel*

A classe *GeomBasedModel* representa uma abstração do modelo utilizado em uma modelagem baseada em geometria. Um objeto desta classe contém referências para objetos da classe *Solid*, representando os sólidos que compõem o modelo e objetos da classe *Mesh*, representando as malhas de elementos finitos associadas ao modelo geométrico, bem como métodos que manipulam esses objetos, tais como o método para criar um objeto da classe *solid* (*createsolid*) ou o método para criar um objeto da classe *Mesh* (*createnummodel*). As variáveis que representam os atributos do modelo são herdadas da classe base *Model*. Os métodos responsáveis pela associação de atributos às entidades geométricas ou de malha são definidos nesta classe (por exemplo, o método *attachattrib*, que associa um atributo já criado às entidades do modelo). A Figura 2.9 mostra a representação gráfica dos objetos da classe *GeomBasedModel*.

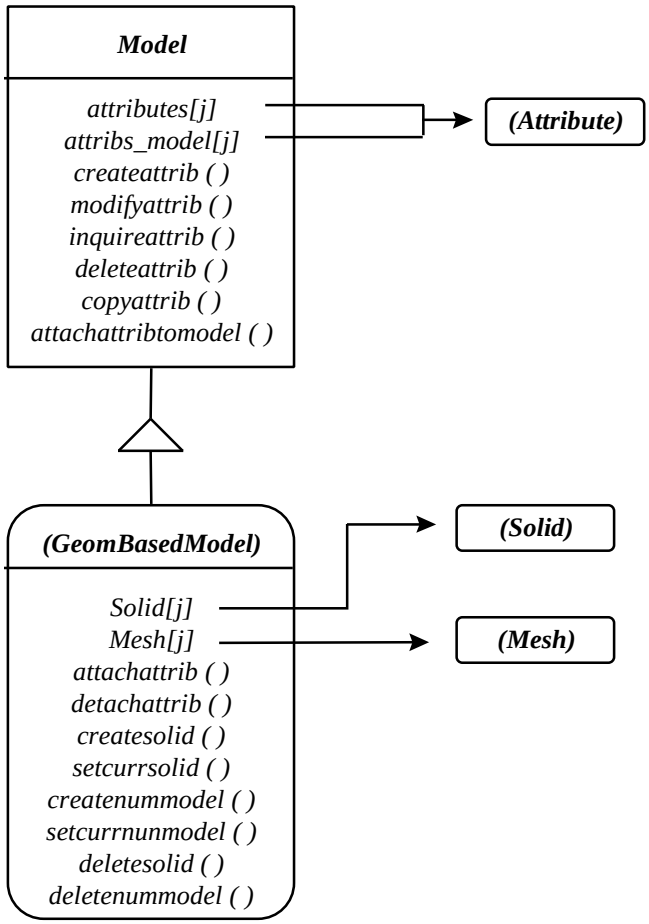

Figura 2.9 - Representação gráfica dos objetos da classe *GeomBasedModel*.

Como foi dito acima, o método *attachattrib* da classe *GeomBasedModel* é responsável pela associação de atributos às entidades geométricas do modelo. Este método repassa para o objeto da classe *Solid*, o identificador da entidade ao qual o atributo será associado e o objeto da classe *Attribute*, que representa o atributo que será aplicado à entidade. Por sua vez, o método *attachattribtoentity* da classe *Solid* repassa o atributo para o objeto da classe *Geometry*. Por fim, o método *attach* da classe *Geometry* associa o atributo ao objeto geométrico. Esta cadeia está representada na Figura 2.10.

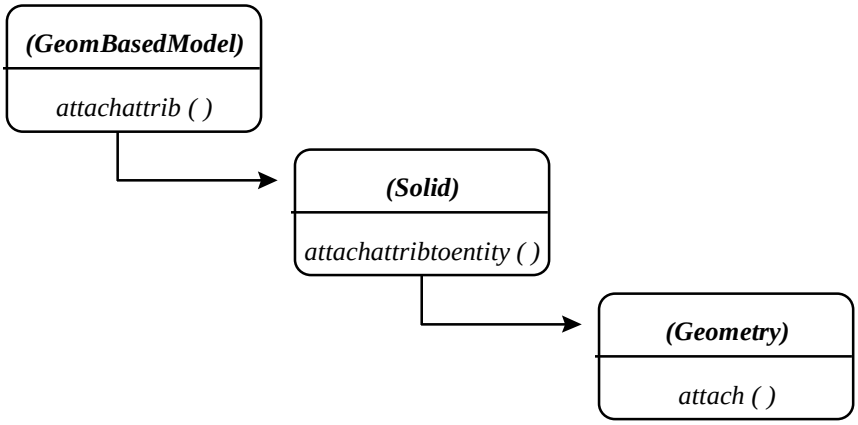

Figura 2.10 - Fluxo mostrando a associação de atributos às entidades geométricas.

2.3.4 Classe *MeshBasedModel*

A classe *MeshBasedModel* representa uma abstração do modelo utilizado em uma modelagem baseada em malha de elementos finitos. Um objeto desta classe contém informações sobre os atributos, os nós e os elementos do modelo discretizado. Esta classe possui herança múltipla (ver Figura 2.2), onde as informações relacionadas aos atributos do modelo são obtidas da classe base *Model*, enquanto que as informações relativas aos nós e elementos da malha são herdadas da classe base *Mesh*. Os métodos responsáveis pela associação de atributos às entidades de malha são definidos nesta classe (por exemplo, o método *modifyattrib*, que modifica os valores referentes as propriedades de um atributo já criado). A Figura 2.11 mostra a representação gráfica dos objetos desta classe.

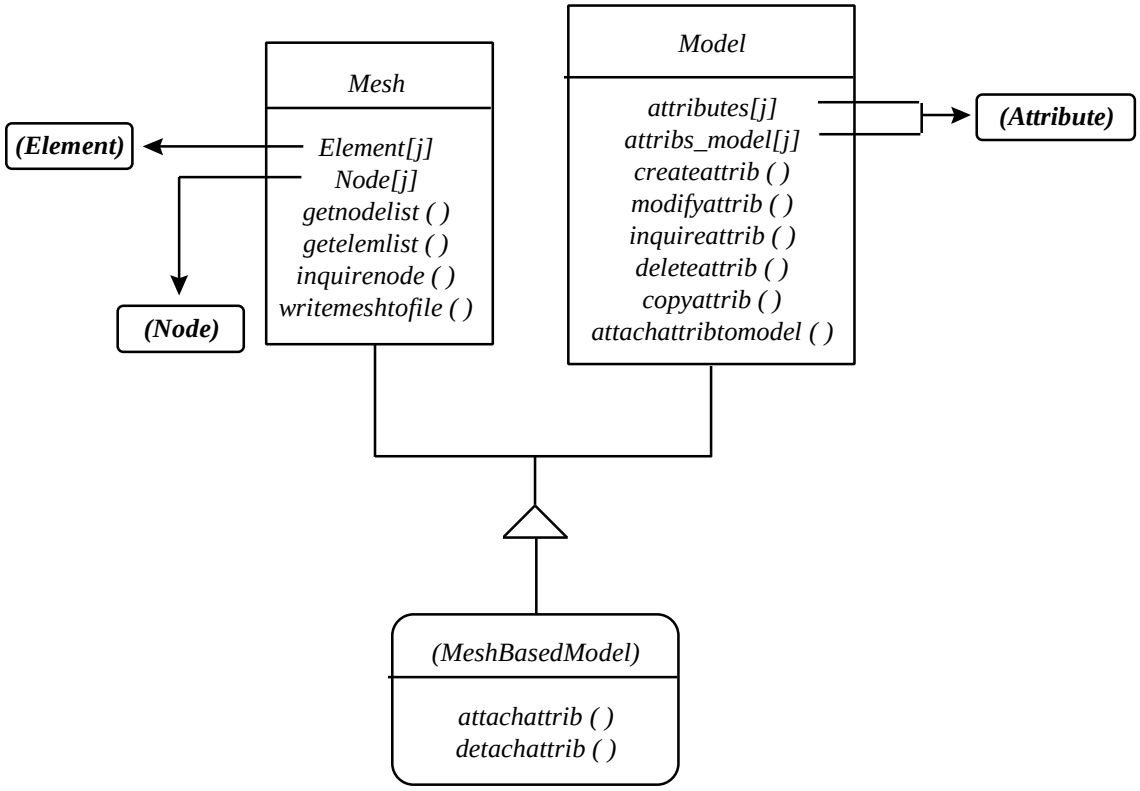

Figura 2.11 - Representação gráfica dos objetos da classe *MeshBasedModel*.

2.3.5 Classe *Solid*

Os objetos da classe *Solid* representam abstrações dos sólidos que compõem o modelo geométrico de uma simulação. Cada objeto possui uma lista de objetos da classe *Geometry* que representam abstrações das entidades geométricas de um determinado sólido do modelo geométrico. Os métodos definidos nesta classe respondem por informações relativas às entidades geométricas associadas ao sólido, tais como o método *removetop*, que remove um objeto da classe *Geometry* deste sólido, o método *attachattribtoentity*, que associa um atributo a uma entidade geométrica do sólido, ou o método *readtopfromfile*, que lê de um arquivo de dados informações referentes às entidades geométricas deste sólido. A Figura 2.12 mostra a representação gráfica dos objetos desta classe.

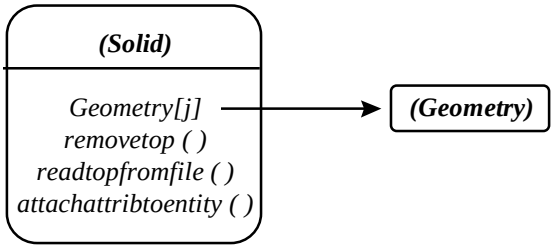

Figura 2.12 - Representação gráfica dos objetos da classe *Solid*.

2.3.6 Classe *Geometry*

Um objeto da classe *Geometry* representa uma abstração das entidades geométricas do modelo. As instâncias da classe *Geometry* contêm referências para os objetos da classe *Attribute* que estão associados à entidade geométrica. Os métodos desta classe respondem por informações relacionadas à entidade, como, por exemplo, o método *inquire*, que mostra os atributos associados a esta entidade, ou o método *attach*, que associa um atributo a esta entidade. A Figura 2.13 mostra a representação gráfica dos objetos da classe *Geometry*.

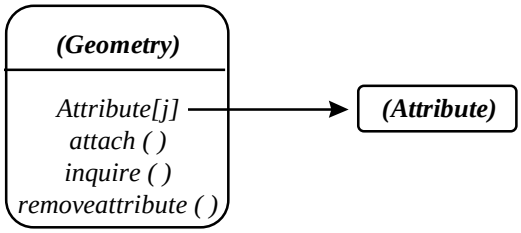

Figura 2.13 - Representação gráfica dos objetos da classe *Geometry*.

A classe *Geometry* possui subclasses que se referem aos tipos de entidades geométricas definidas na aplicação. Os métodos da classe *Geometry* possuem uma implementação padrão (*default*) que são herdados por todas as suas subclasses. Entretanto, estes métodos podem ser redefinidos pelo usuário configurador com o objetivo de controlar o comportamento de cada tipo de entidade geométrica da aplicação. Por exemplo, o método *attach* da classe *Geometry* simplesmente adiciona um novo atributo na lista de atributos da entidade. O usuário pode redefinir este

método nas subclasses da classe *Geometry* para verificar se já existe um atributo do mesmo tipo associado à entidade correspondente.

2.3.7 Classe *Node*

Um objeto da classe *Node* representa uma abstração do nó da malha correspondente na aplicação. Para cada nó da malha de elementos finitos, é criado um objeto da classe *Node*. Em uma modelagem baseada em geometria, o objeto desta classe possui uma referência para a entidade geométrica ao qual o nó está associado (essa informação não está disponível em uma modelagem baseada em malha). Por exemplo, se um nó da malha está sobre uma curva do modelo geométrico, o objeto da classe *Node* correspondente a este nó referencia o objeto geométrico relativo à curva. Além disso, o objeto desta classe possui referência para objetos da classe *Attribute* que estão associados diretamente ao nó. Esta classe implementa métodos que respondem informações referentes aos atributos associados ao nó (*getattribslist*, que fornece todos os atributos associados ao nó), bem como métodos que fornecem informações relativas ao nó, tal como o método *getcoords*, que fornece as coordenadas do nó, no contexto da aplicação. A Figura 2.14 mostra a representação gráfica dos objetos da classe *Node*.

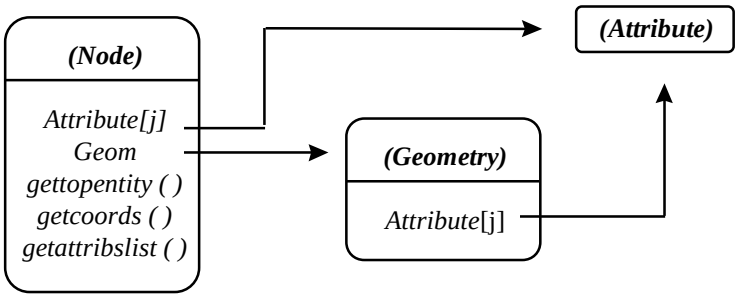

Figura 2.14 - Representação gráfica dos objetos da classe *Node*.

2.3.8 Classe *ElemFeature*

A classe *ElemFeature* foi criada para facilitar a identificação dos atributos associados a partes do elemento finito. Por exemplo, cada lado de um elemento finito bidimensional corresponde a um objeto *ElemFeature*. Da mesma forma, em um elemento finito tridimensional, cada face e cada aresta do sólido que representa este elemento finito corresponde a um objeto *ElemFeature*. Se a aplicação estiver utilizando uma modelagem baseada em geometria, o objeto desta classe possui uma referência para a entidade geométrica ao qual o objeto *ElemFeature* está associado (essa informação não está disponível em uma modelagem baseada em malha). Além disso, o objeto desta classe possui uma lista de objetos da classe *Attribute* que estão associados diretamente ao *ElemFeature*. Esta classe implementa métodos que respondem informações referentes aos atributos associados a uma parte do elemento finito (por exemplo, o método *attach*, que associa um atributo ao objeto *ElemFeature*), bem como métodos que fornecem informações referentes ao objeto *ElemFeature*, tais como o método *getconnect*, que retorna a conectividade do objeto *ElemFeature* ou o método *getelement*, que retorna o objeto da classe *Element* referente ao elemento finito associado ao objeto *ElemFeature*. A Figura 2.15 mostra a representação gráfica dos objetos desta classe.

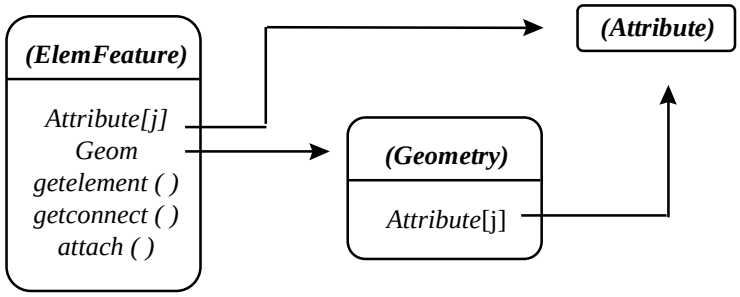

Figura 2.15 - Representação gráfica dos objetos da classe *ElemFeature*.

2.3.9 Classe *Element*

Um objeto da classe *Element* representa uma abstração dos elementos finitos da discretização do modelo. Para cada elemento da malha é criado um objeto desta classe. As subclasses da classe *Element* se referem aos tipos de elementos definidos na aplicação. Existe um mecanismo que cria automaticamente estas subclasses. A Figura 2.16 mostra um exemplo de organização das *Element Classes* em uma aplicação específica.

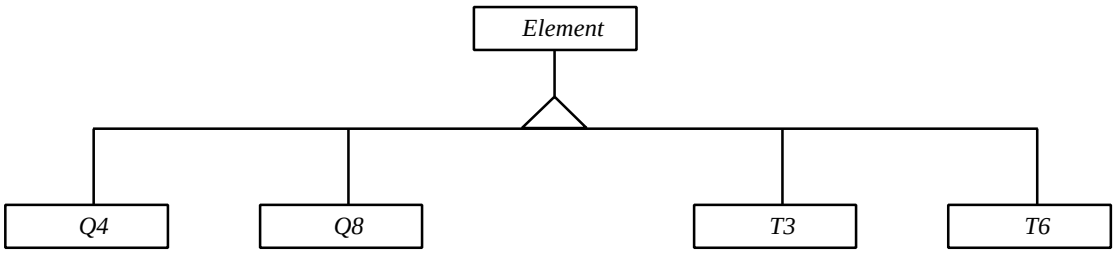

Figura 2.16 - Organização das *Element classes* em uma aplicação específica.

Os objetos destas subclasses representam abstrações dos elementos finitos correspondentes na aplicação. Em uma modelagem baseada em geometria, os objetos destas subclasses referenciam o objeto geométrico ao qual o elemento está associado (essa informação não está disponível em uma modelagem baseada em malha). Por exemplo, se um elemento da malha está sobre uma região do modelo geométrico, o objeto da classe *Element* correspondente a este elemento referencia o objeto geométrico relativo à região. Além disso, existe referência para os objetos da classe *ElemFeature*, que representam abstrações das partes associadas ao elemento finito. Os objetos das subclasses da classe *Element* possuem ainda referências para os objetos da classe *Attribute* que estão associados diretamente ao elemento. Esta classe implementa métodos que retornam informações necessárias para a criação do modelo discretizado para uma análise numérica (por exemplo, o método *getconnect*, que retorna a conectividade do elemento), bem como métodos que fornecem informações relativas aos atributos associados ao elemento finito, tais como o método *inquire*, que mostra os atributos associados ao elemento, ou o método *removeattribute*, que remove atributos associados ao elemento ou a partes dele. A Figura 2.17 ilustra a representação gráfica dos objetos da classe *Element*.

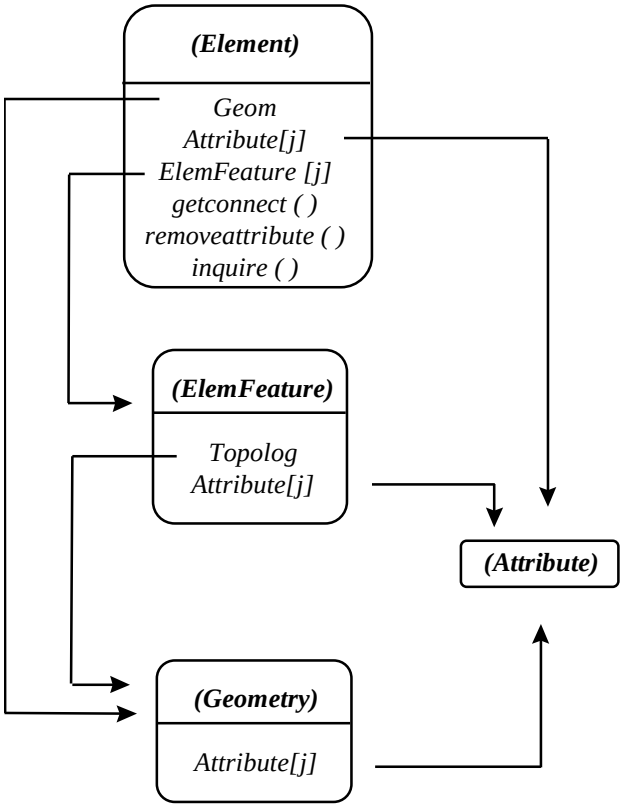

Figura 2.17 - Representação gráfica dos objetos da classe *Element*.

2.3.10 Classe *Attribute*

A classe *Attribute* representa a classe base dos atributos de uma simulação. Para cada tipo de simulação, deve-se definir subclasses derivadas desta classe com o objetivo de atender as necessidades específicas da simulação. As instâncias destas subclasses possuem referências para os objetos da classes *Geometry*, *Element* ou *Node*, ao qual o atributo está associado. Os métodos da classe *Attribute* respondem por informações relativas às entidades geométricas ou de malha ao qual o atributo está associado, tais como o método *removeentity*, que remove um objeto das classes *Geometry*, *Node*, ou *Element* deste atributo, ou o método *showentities*, que mostra todas as entidades (geométricas e de malha) associadas ao atributo.

Os objetos desta classe estão associados a um identificador. Este identificador representa um rótulo (*label*) implementado através de uma cadeia de caracteres, que pode ser definida pelo usuário ou construída automaticamente pelo sistema. As novas

classes são criadas em um arquivo de configuração, que é lido pela aplicação e que pode ser alterado sem a necessidade de recompilação do sistema. A Figura 2.18 mostra uma representação gráfica dos objetos da classe *Attribute*.

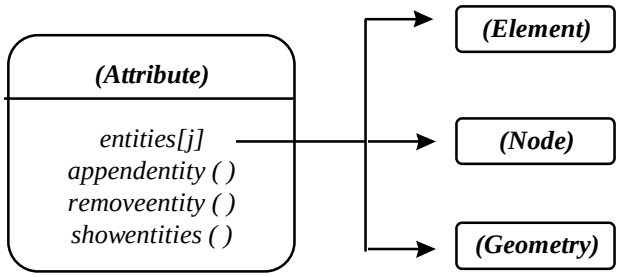

Figura 2.18 - Representação gráfica dos objetos da classe *Attribute*.

Muitas vezes é desejável que classes de atributos, referentes aos tipos de atributos de uma simulação, sejam agrupadas em conjuntos individuais baseado nas características referentes a esses tipos de atributos. Deseja-se, por exemplo, agrupar os tipos de atributos que só podem ser associados às arestas do modelo em um mesmo conjunto, restringindo quais os atributos do grupo inicial que devem fazer parte deste conjunto. O sistema ESAM permite, então, que subclasses da classe *Attribute* (relativas aos tipos de atributos de uma simulação) sejam agrupadas em um mesmo conjunto definido pelo usuário configurador da aplicação. A definição de grupos relacionadas às classes de atributos é bastante importante, principalmente quando se deseja elaborar uma interface gráfica para a aplicação como, por exemplo, controlar a associação de atributos às entidades do modelo, em uma aplicação específica. Inicialmente, todas as subclasses da classe *Attribute* estão definidas em um único grupo *default*.

A criação das classes de atributos é uma tarefa do usuário configurador e independe da arquitetura do ESAM. Isto é feito com alto grau de abstração, utilizando a linguagem de extensão Lua.

2.3.11 Classe *Output*

Como as informações relacionadas aos atributos de uma simulação são gerenciadas pelo sistema ESAM, é desejável que este sistema ofereça ao usuário um mecanismo para que ele possa configurar arquivos de dados relativos ao programa que é utilizado para fazer a análise por elementos finitos. O sistema ESAM provê, então, a classe *Output*, da qual pode-se derivar subclasses, cujo métodos, definidos pelo usuário, permitem a configuração destes arquivos de dados. Dois métodos desta classe podem ser definidos pelo usuário configurador da aplicação: *writetofile*, que é responsável pela escrita dos dados referentes à malha de elementos finitos e dos atributos associados a ela, no formato específico para cada programa de análise; e o método *check*, que pode ser utilizado para verificar a validade dos dados relacionados à malha, antes de estes serem escritos. Os objetos desta classe representam abstrações de formatos de arquivos de dados relacionados aos programas de análise por elementos finitos. A Figura 2.19 mostra a representação gráfica dos objetos da classe *Output*.

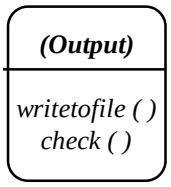

Figura 2.19 - Representação gráfica dos objetos da classe *Output*.

2.4 Serviços Oferecidos pelo ESAM

O ESAM possui um conjunto de funções, implementadas em C, que servem para incorporar o ambiente de configuração de atributos à aplicação e têm como objetivo permitir ao usuário final acessar às funcionalidades do ambiente de configuração, tais como criar um atributo, associar um atributo a uma entidade do modelo, etc. Como uma aplicação que utiliza o sistema ESAM pode ter módulos configuráveis, é desejável que esses serviços do ESAM possam ser utilizados nesses módulos. Portanto, os serviços disponíveis em C também estão disponíveis na linguagem de configuração Lua. Esses serviços podem ser acionados, por exemplo, através de *callbacks* de

eventos de interface. A seguir são descritos os serviços disponibilizados pelo sistema ESAM.

2.4.1 Serviços de Inicialização e Finalização

Estes serviços são responsáveis pelas tarefas de inicialização e finalização necessárias ao ambiente de configuração, tais como a inicialização do sistema ESAM e a criação de objetos das classes *GeomBasedModel*, em uma modelagem baseada em geometria, ou *MeshBasedModel*, em uma modelagem baseada em malha. O sistema ESAM adota o conceito de objeto corrente nas operações realizadas sobre algumas entidades envolvidas no processo de simulação (atributos, sólidos, modelo geométrico e modelo numérico). Existem, portanto, serviços de inicialização que selecionam objetos como correntes, tal como o serviço *EsamSetCurrGeomBasedModel*, responsável por selecionar um objeto criado para representar uma modelagem baseada em geometria como corrente. Este serviço deve ser chamado sempre após a criação de um objeto que irá representar o tipo de modelagem utilizada pela aplicação. Abaixo, segue uma descrição destes serviços:

EsamOpen ()

Inicializa o sistema ESAM, realizando todos os procedimentos para a sua utilização.

EsamClose ()

Encerra a utilização do sistema ESAM.

EsamCreateGeomBasedModel()

Cria um objeto para representar uma modelagem baseada em geometria.

EsamDeleteGeomBasedModel(objmodel)

Destroi um objeto que representa uma modelagem baseada em geometria.

EsamCreateMeshBasedModel()

Cria um objeto para representar uma modelagem baseada em malha.

EsamDeleteMeshBasedModel(objnmodel)

Destroi um objeto que representa uma modelagem baseada em malha.

EsamSetCurrGeomBasedModel(objmodel)

Seleciona um objeto que representa uma modelagem baseada em geometria como corrente.

EsamSetCurrMeshBasedModel(objnmodel)

Seleciona um objeto que representa uma modelagem baseada em malha como corrente.

EsamGetCurrGeomBasedModel()

Retorna o objeto corrente que representa uma modelagem baseada em geometria.

EsamGetCurrMeshBasedModel()

Retorna o objeto corrente que representa uma modelagem baseada em malha.

2.4.2 Serviços de Especificação de Atributos

Estes serviços são responsáveis por tarefas de manipulação relacionadas aos atributos, tais como criação ou destruição de um atributo, modificação de atributos já existentes e associação ou remoção de atributos das entidades geométricas ou de malha do modelo. Existe um serviço (*EsamSetCurrAttrib*) que permite selecionar um objeto, entre aos atributos do modelo, como corrente. Este serviço deve ser sempre chamado antes que qualquer operação relacionada com a manipulação de atributos seja realizada. Além disso, existem serviços (*EsamSelectAttrib* e *EsamSelectClass*) que permitem ao usuário da aplicação selecionar, a partir de diálogos pré-definidos pelo sistema, atributos referentes a uma determinada classe definida no arquivo de configuração, ou classes de um determinado grupo pré-estabelecido pela aplicação. Segue abaixo uma descrição destes serviços:

EsamCreateAttrib (classname)

Cria um objeto representando um atributo.

EsamAttachAttrib ()

Associa objeto corrente representando um atributo às entidades do modelo.

EsamDetachAttrib ()

Remove o objeto corrente das entidades do modelo.

EsamModifyAttrib ()

Modifica as variáveis que representam as propriedades de um objeto corrente.

EsamDeleteAttrib ()

Destroi o objeto corrente representando um atributo.

EsamCopyAttrib (attlabel, newlabel)

Faz uma cópia de um objeto representando um atributo.

EsamSpecifyAttrib (groupname)

Mostra um diálogo contendo todas as funcionalidades relacionadas com a manipulação dos atributos.

EsamAttachAttribToModel ()

Associa um objeto representando um atributo ao modelo corrente.

EsamSetCurrAttrib (attobj)

Seleciona um objeto representando um atributo como corrente.

EsamGetCurrAttrib ()

retorna o objeto selecionado como corrente.

EsamGetAttrib (attlabel)

Retorna o objeto representando um atributo associado a um label.

EsamSelectClass (groupname)

Exibe um diálogo mostrando todas as classes relacionadas a um determinado grupo de atributos.

EsamSelectAttrib (classlabel)

Exibe um diálogo mostrando todos os objetos de uma determinada classe de atributos.

2.4.3 Serviços de Cliente

Estes serviços devem ser chamados pela aplicação sempre que operações relacionadas à manipulação de dados referentes à geometria, tais como a criação de sólidos ou a eliminação de entidades do modelo, sejam realizadas pela aplicação. Estas operações atuam sobre o modelo corrente definido na aplicação. Os serviços responsáveis por operações sobre as entidades geométricas do modelo (*EsamCreateGeomEntity*, por exemplo) devem atuar sobre o sólido corrente definido pela aplicação. O sistema ESAM oferece um serviço que permite à aplicação selecionar um objeto da classe *Solid* como corrente (*EsamSetCurrSolid*). Existem, ainda, serviços que realizam operações sobre as entidades geométricas do modelo, tais como a divisão de uma entidade gerando duas novas entidades (*EsamSplitGeomEntity*) ou a união de duas entidades geométricas em uma única entidade (*EsamJoinGeomEntity*). Além disso, existe um serviço que permite a criação de modelos numéricos relacionados a um modelo geométrico corrente definido pela aplicação (*EsamCreateNumModel*). Como os serviços que realizam operações sobre a malha de elementos finitos atuam sobre um modelo numérico corrente, existe um serviço no ESAM que possibilita a aplicação selecionar um objeto que representa uma abstração do modelo numérico como

corrente (*EsamSetCurrNumModel*). O serviço *EsamInitNumModel* é responsável pelo mapeamento dos atributos associados às entidades geométricas do modelo para as entidades (nós e elementos) da malha de elementos finitos, em uma modelagem baseada em geometria. Segue abaixo uma descrição destes serviços:

EsamCreateNumModel (addr)

Cria um objeto representando uma malha associada a um modelo baseado na geometria.

EsamDeleteNumModel (addr)

Destroi um objeto representando uma malha associada a um modelo baseado na geometria.

EsamSetCurrNumModel (addr)

Seleciona um objeto representando uma malha de elementos finitos como corrente.

EsamInitNumModel ()

Inicializa a malha de elementos finitos. É responsável pelo mapeamento dos atributos associados às entidades geométricas para a malha de elementos finitos.

EsamCreateSolid (addrsolid)

Cria um objeto representando um sólido associado a um modelo baseado em geometria.

EsamDeleteSolid (addrsolid)

Destroi um objeto representando um sólido associado a um modelo baseado em geometria.

EsamSetCurrSolid (addrsolid)

Seleciona um objeto representando um sólido como corrente.

EsamCreateGeomEntity (addr, type)

Cria um objeto representando uma entidade geométrica do modelo.

EsamDeleteGeomEntity (addr, type)

Destroi um objeto representando uma entidade geométrica do modelo.

EsamSplitGeomEntity (addrold, addrnew1, addrnew2, type)

Divide um objeto representando uma entidade geométrica em dois outros objetos.

EsamJoinGeomEntity (addrold1, addrold2, addrnew, type)

Une dois objetos representando entidades geométricas em um único objeto.

2.4.4 Serviços de Validação

Estes serviços devem ser chamados pela aplicação sempre que operações que atuam sobre a geometria, tais como a divisão ou a eliminação de entidades do modelo, são realizadas pela aplicação. Existem casos em que as operações são geometricamente válidas, mas podem depender dos atributos associados às entidades correspondentes.

Como o sistema ESAM gerencia os atributos do modelo, ele oferece serviços para verificar a validade destas operações. Os serviços estão relacionados a uma modelagem baseada em geometria e suas operações atuam sobre o modelo e o sólido correntes definidos na aplicação. Segue abaixo uma descrição destes serviços:

EsamChkCreateGeomEntity (addr, type)

Verifica se um objeto representando uma entidade geométrica pode ser criado.

EsamChkDeleteGeomEntity (addr, type)

Verifica se um objeto representando uma entidade geométrica pode ser destruído.

EsamChkSplitGeomEntity (addr, type)

Verifica se um objeto representando uma entidade geométrica pode ser dividido.

EsamChkJoinGeomEntity (addr1, addr2, type)

Verifica se dois objetos representando entidades geométricas podem ser unidos.

2.4.5 Serviços de Entrada e Saída de Dados

Estes serviços são responsáveis por tarefas como carregar o arquivo de configuração da aplicação ou leitura e gravação de arquivos contendo informações relativas aos atributos da aplicação. Os serviços responsáveis pela leitura e gravação dos dados referentes às entidades geométricas do modelo devem atuar sobre um modelo geométrico corrente definido pela aplicação. Já os serviços responsáveis pela leitura e gravação dos dados referentes à malha de elementos finitos atuam sobre um modelo numérico corrente também definido pela aplicação. Abaixo segue uma descrição destes serviços:

EsamLoadAtrribFile (arqname)

Carrega um arquivo de configuração de atributos.

EsamOpenReadFile (arqname)

Abre um arquivo para leitura dos dados referentes aos atributos.

EsamCloseReadFile ()

Fecha um arquivo aberto para leitura dos dados referentes aos atributos.

EsamOpenWriteFile (arqname)

Abre um arquivo para gravação dos dados referentes aos atributos.

EsamCloseWriteFile ()

Fecha um arquivo aberto para gravação dos dados referentes aos atributos.

EsamReadAttribModel ()

Lê os atributos do modelo.

EsamWriteAttribModel ()

Salva os atributos do modelo.

EsamReadModelFile ()

Lê as informações referentes a um modelo baseado em geometria.

EsamWriteModelFile ()

Escreve as informações referentes a um modelo baseado em geometria.

EsamReadMeshFile ()

Lê as informações referentes a malha de elementos finitos.

EsamWriteMeshFile ()

Salva as informações referentes a malha de elementos finitos.

EsamWriteNumModelFile ()

Salva um arquivo contendo as informações referentes a uma análise específica (subclasses da classe Output).

EsamImportAttrib (arqname)

Importa atributos de um arquivo de dados cujo formato permite que ele seja editado pelo usuário.

EsamExportAttrib (arqname)

Exporta atributos de um arquivo de dados cujo formato permite que ele seja editado pelo usuário.

EsamSetFileFormat (format)

Seleciona um formato corrente referente a uma análise específica (subclasses da classe Output).

EsamCheckNumModel (void)

Verifica a consistência dos dados referentes a uma análise específica (subclasses da classe Output).

2.4.6 Serviços de Consulta

Estes serviços são responsáveis pelas tarefas de consultas realizadas sobre os atributos do modelo, as entidades geométricas e de malha, os sólidos do modelo e os modelos geométrico e numérico da aplicação. Exemplos são a consulta às variáveis que representam as propriedades físicas de um atributo ou a consulta sobre os atributos

associados a uma determinada entidade do modelo. Os serviços relacionados com consultas a informações referentes à geometria atuam sobre o modelo geométrico e o sólido corrente definidos pela aplicação. Já os serviços relacionados com consultas às entidades da malha atuam sobre o modelo numérico corrente, também definido pela aplicação. Abaixo segue a descrição destes serviços:

EsamInquireFromAttrib (attlabel)

Mostra todas as entidades relacionadas a um determinado atributo.

EsamInquireAttrib (attlabel)

Mostra as propriedades referentes a um atributo.

EsamInquireEntity ()

Mostra os atributos associados a uma determinada entidade.

EsamHasGeomBasedModel ()

Verifica se um objeto representando uma modelagem baseada em geometria já foi criado.

EsamHasMeshBasedModel ()

Verifica se um objeto representando uma modelagem baseada em malha já foi criado.

EsamHasNumModel ()

Verifica se existe objetos representando as malhas de elementos finitos do modelo.

EsamHasSolid ()

Verifica se existe objetos representando sólidos do modelo.

EsamHasElemAttrib (id, ninc, inc)

Verifica se um determinado elemento possui atributo associado a ele.

EsamHasNodeAttrib (id)

Verifica se um determinado nó possui atributo associado a ele.

EsamHasGeomEntity (addr, type)

Verifica se uma determinada entidade possui atributo associado a ela.

---

3. Sistema Integrado de Geotecnia para Múltiplas Análises - SIGMA

Este capítulo tem por objetivo descrever o Sistema Integrado de Geotecnia para Múltiplas Análises (SIGMA), desenvolvido neste trabalho, utilizado no Cenpes/Petrobras para simulações numéricas de problemas bidimensionais de geotecnia pelo MEF (escavação de poços para mineração subterrânea, por exemplo). Este sistema integra, em um único ambiente, os módulos de pré-processamento MTool e pós-processamento MView, responsáveis pela geração do modelo numérico para análise pelo MEF e visualização de resultados, com os programas de análise numérica para problemas de geotecnia AEEPECD [Costa 1984] e ANVEC [Costa 1984]. Este ambiente é extensível e configurável, permitindo que novas funcionalidades sejam facilmente implementadas. Na versão anterior do SIGMA, a comunicação entre os módulos era realizada via arquivo neutro (incluindo os atributos de simulação - vide seção 1.1). Nesta versão, as informações relativas aos atributos foram retiradas dos módulos MTool e MView, e passaram a ser gerenciadas, exclusivamente, pelo sistema ESAM (*Extensible System of Attributes Management*), descrito no capítulo 2. Desta forma, quando um novo tipo de atributo for incorporado ao módulo de análise, será facilmente incluído nos módulos MTool e MView.

Para desenvolver o SIGMA, dois procedimentos foram realizados. O primeiro diz respeito à integração do sistema ESAM com os módulos de pré-processamento (MTool) e pós-processamento (MView); o segundo procedimento refere-se à configuração do SIGMA através da incorporação dos atributos utilizados pelos programas de análise AEEPECD e ANVEC. Este passo consiste na implementação de rotinas que viabilizaram a comunicação entre o sistema ESAM e os programas de análise e na criação dos atributos.

Neste trabalho, os detalhes referentes aos programas de análise utilizados não foram considerados. Maiores informações sobre eles podem ser encontradas em [Costa 1984] ou em [Mello 1994].

3.1 Módulos do SIGMA

O SIGMA é composto por quatro módulos responsáveis pelas diversas tarefas envolvidas em um processo de simulação (modelagem geométrica, especificação dos atributos, geração da malha de elementos finitos, análise numérica e visualização dos resultados), e por um módulo principal responsável pelo gerenciamento e integração dos outros módulos (Figura 3.1).

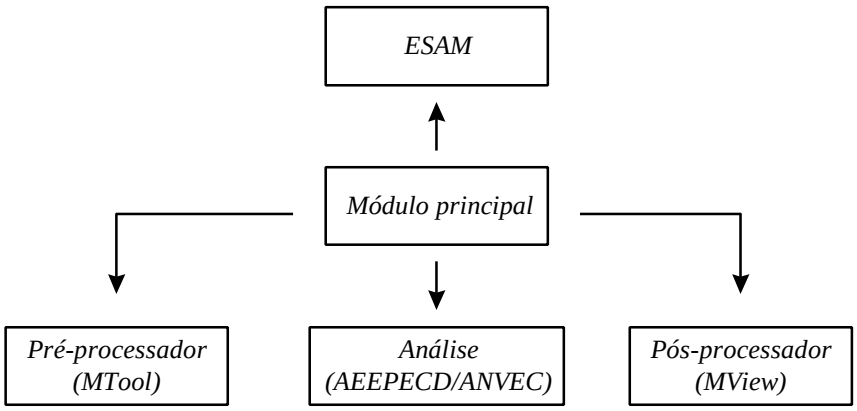

Figura 3.1 - Módulos utilizados pelo Sistema Integrado de Geotecnia para Múltiplas Análises.

3.1.1 MTool

O MTool (*Bidimensional Mesh Tool*) é um programa gráfico interativo para a geração de malhas de elementos finitos bidimensionais. Ele possui dois ambientes bem distintos. O primeiro é responsável pela definição da geometria e topologia do modelo. Este ambiente é suportado pela estrutura de dados topológica *half-edge* [Mantyla 1988], que permite assegurar a consistência do modelo durante a sua criação. É possível, por

exemplo, identificar interseções entre as arestas, determinar se uma face foi definida ou dividida, verificar os efeitos indiretos de uma determinada atribuição (por exemplo, a troca do número de subdivisões de uma aresta elimina as malhas adjacentes, caso contrário o modelo ficaria inconsistente), etc. A implementação desta estrutura de dados topológica é complexa e ocupa um espaço de memória relativamente grande para a sua realização. Entretanto, o MTool utiliza a biblioteca de rotinas HED (*half-edge data structure*) [Martha *et al.* 1993], baseada na estrutura de dados *half-edge*, que oferece uma interface de alto nível (nível abstrato), encapsulando a complexidade desta estrutura de dados.

O segundo ambiente do MTool apresenta uma representação única de malhas de elementos finitos convencional, obtida a partir da conversão das malhas do modelo de regiões, definido no ambiente de edição do modelo. No segundo ambiente, o usuário pode realizar operações típicas de modelos de elementos finitos, como, por exemplo, consultar a incidência nodal de cada elemento finito gerado na conversão. No entanto, não existe funções para edição da malha, pois não se tem o suporte de uma estrutura de dados topológica. Para isso, o usuário pode retornar ao ambiente de edição do modelo, realizar as alterações necessárias e fazer uma nova conversão para o modelo de elementos finitos.

3.1.2 AEEPECD e ANVEC

O AEEPECD e o ANVEC são programas que realizam análises numéricas em simulações de problemas bidimensionais de geotecnia pelo MEF. O programa AEEPECD realiza análises elásticas, com não-linearidade física, de modelos de estado plano de tensão, estado plano de deformação e assimétricos; enquanto o programa ANVEC realiza análises numéricas quasi-estáticas dos mesmos tipos de modelos. A entrada de dados destes programas é realizada através de arquivos com formatos específicos. Os detalhes referentes a estes formatos podem ser obtidos em [Mello 1994].

Os resultados obtidos a partir de uma análise numérica realizada por esses programas são gravados em vários arquivos de dados com as seguintes extensões: DSL, contendo os resultados referentes aos deslocamentos dos nós da malha; TNS, contendo os resultados referentes as tensões nos pontos de Gauss dos elementos da malha; DEF, contendo os resultados referente as deformações nos pontos de Gauss dos elementos da malha e extensão; e EFE, que contém os resultados referentes as tensões efetivas nos pontos de Gauss dos elementos da malha.

3.1.3 MView

O MView (*Bidimensional Mesh View*) é um programa gráfico interativo para visualização de resultados de uma análise por elementos finitos. Este módulo fornece informações qualitativas e quantitativas sobre a malha e os resultados referentes à mesma. A análise quantitativa é feita através de gráficos de resposta e funções de consulta, que fornecem informações tais como número de nós ou incidência do elemento. A análise qualitativa consiste basicamente na representação visual de resultados, que podem ser de dois tipos: resultados escalares, como componentes de tensão ou campo de temperatura, e resultados vetoriais, tais como campo de deslocamentos ou campo de velocidades. Os resultados de campos escalares podem ser fornecidos nos pontos nodais, de forma suavizada ou não, ou nos pontos de Gauss. De maneira a auxiliar na visualização do modelo, existem funções para manipular a vista, tais como a possibilidade de mudar os limites da janela de visualização ou fazer um *zoom* em todo o modelo ou em alguma face do mesmo. O programa ainda permite visualizar a animação do modelo com a resposta de um determinado caso e campo ao longo de diversos passos. A entrada de dados referente à malha de elementos finitos e aos resultados obtidos a partir da análise numérica é feita através do arquivo neutro (*NeutralFile*).

3.1.4 ESAM

O sistema ESAM é o módulo responsável pelo gerenciamento dos atributos do SIGMA. Ele permite a especificação dos atributos no pré-processador MTool (criação de atributos, associação de atributos às entidades do modelo, etc.) e a consulta destes atributos no pós-processador MView. O ESAM possibilita que uma interface seja definida para a captura dos dados referentes aos atributos do SIGMA. Existe, ainda, um mecanismo que permite ao usuário do SIGMA implementar a rotina responsável pela geração do arquivo de dados, referente a comunicação entre o MTool e os módulos de análise. Assim, os novos atributos incorporados ao programa de análise podem ser facilmente inseridos no pré-processador MTool, a partir da definição de uma interface para capturar os seus dados, bem como a inclusão, pelo próprio usuário do sistema, dos dados referentes a esses atributos na rotina que gera o arquivo de dados lido pelo programa de análise. Os detalhes referentes ao módulo ESAM foram descritos no capítulo 2.

3.1.5 Módulo Principal

O módulo principal é responsável pelo gerenciamento e integração dos demais módulos do SIGMA (MTool, MView, ESAM, AEEPECD e ANVEC).

Neste módulo são chamadas as funções de inicialização referentes aos outros módulos do sistema. Outra tarefa realizada por este módulo consiste na conversão dos arquivos gerados pelos programas AEEPECD e ANVEC contendo os resultados da análise, que possuem formatos próprios, para que possam ser interpretados pelo pós-processador MView, que utiliza o formato do arquivo de dados *NeutralFile*. A Figura 3.2 ilustra o fluxo de dados referentes à conversão dos dados gerados pelos programas de análise para serem visualizados pelo pós-processador MView.

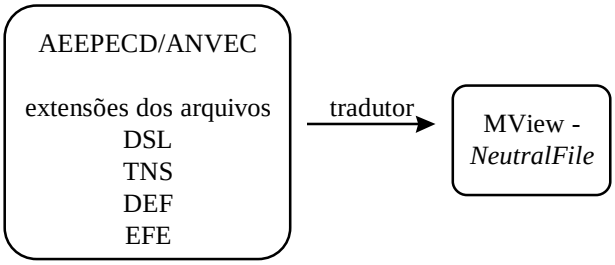

Figura 3.2 - Fluxo de dados referentes à conversão dos dados gerados pelos programas de análise para serem visualizados pelo pós-processador MView.

3.2 Integração do ESAM com o Pré-processador MTool e o Pós-Processador MView

Conforme mencionado anteriormente, um dos procedimentos necessários para o desenvolvimento do SIGMA consiste na integração do ESAM com o pré-processador MTool e o pós-processador MView. Para isso, duas tarefas básicas tiveram que ser realizadas. A primeira, consistiu na implementação das funções MIS (seção 2.2) responsáveis pela ligação dos módulos MTool e MView com o sistema ESAM, pois existem informações armazenadas na base de dados destes módulos que são necessárias para a especificação dos atributos como, por exemplo, as entidades do modelo ao qual um atributo deve ser associado. A segunda tarefa consistiu em acoplar os serviços oferecidos pelo ESAM (seção 2.4) aos módulos MTool e MView, permitindo que o usuário do SIGMA tenha acesso às funcionalidades do ambiente de configuração de atributos, tais como, criação de um novo atributo ou modificação das variáveis que representam as propriedades dos atributos já existentes. A Figura 3.3 mostra a integração do ESAM com os módulos MTool e MView do sistema SIGMA.

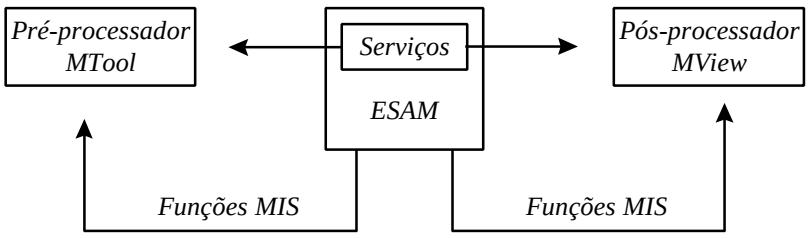

Figura 3.3 - Integração do ESAM com os módulos MTool e MView.

Para ilustrar estas atividades, dois exemplos são considerados. O primeiro refere-se a utilização do serviço *EsamOpen* (seção 2.4.1), que deve ser chamado no início do programa para a inicialização do sistema ESAM. Este serviço, implementado no módulo principal, entre outras tarefas, registra as entidades geométricas e de malha definidas nos módulos MTool e MView, para que classes referentes a essas entidades possam ser criadas pelo ESAM. Como as informações referentes à definição destas entidades estão localizadas na base de dados dos módulos MTool e MView, a maneira que o ESAM utiliza para acessar estas informações é através das funções *MisGeomRegEntities*, que registra os tipos de entidades geométricas, e *MisRegElemTypes*, que registra os tipos de elementos finitos. No MTool, a função *MisGeomRegEntities* foi implementada registrando seis tipos de entidades diferentes: *vértice*, *curva*, *elemento de interface, contorno infinito*, *contorno* e *face*. A entidade face, por exemplo, foi registrada com o nome "*Face*", seu identificador no contexto do MTool é o número 3 e sua dimensão (classe base) é 2. O código computacional em C com a implementação da função *MisGeomRegEntities* no MTool está mostrado na Figura 3.4. A Figura 3.5 mostra o código computacional que implementa a função *MisRegElemTypes* no módulo MTool.

```
int MisRegGeomEntities ( void (*register_ent)(char *name, int type, int dim) )
{
  /* chama a função recebida como parâmetro com os valores
  ** nome, tipo e dimensão.  */

  register_ent( "Vertex",        3, 0);
  register_ent( "Curve",         2, 1);
  register_ent( "Infinite",      5, 1);
  register_ent( "Interface",     4, 1);
  register_ent( "Boundary",      6, 1);
  register_ent( "Face",          1, 2);

  return 1;
}
```

Figura 3.4 - Código computacional em C da função *MisRegGeomEntities*, que registra os tipos de entidades geométricas utilizadas no MTool.

```
int MisRegElemTypes ( void (*register_elem) (char *name, int type, int nnos, int interp ) )
{
  /* chama a função recebida como parâmetro com os valores
  ** nome, tipo, numero de nos, ordem de interpolação.  */

  register_elem( "Q4",          0, 4, 1);
  register_elem( "Q8",          1, 8, 2);
  register_elem( "T3",          3, 3, 1);
  register_elem( "T6",          4, 6, 2);
  register_elem( "QUAD",        6, 8, 2);
  register_elem( "INTERFACE",   7, 6, 2);
  register_elem( "INFINITE",    8, 5, 2);

  return 1;
}
```

Figura 3.5 - Código computacional em C da função *MisRegElemTypes*, que registra os tipos de elementos finitos utilizados no MTool.

O segundo exemplo refere-se à associação de atributos às entidades do Mtool. O ESAM oferece o serviço *EsamAttachAttrib* (seção 2.4.2) que associa o atributo corrente (definido pelo serviço *EsamSetCurrAttrib*) ao conjunto de entidades selecionadas no pré-processador, disponibilizadas através da função *MisGetNSelect*. A Figura 3.6 mostra o código computacional com a implementação a desta função.

```
int MisGetNSelect( void (*endaction) ( int nent, char **name, int *type, void ** ent) )
```

```
{
  SelLst = sel;

  /* atribui a variável nent o número de entidades selecionados. */
  nent    = GetEntitiesSelect ( );

  /* aloca os vetores. */
  name  = (char **) calloc ( nent, sizeof ( char * ) );
  type    = (int *)      calloc ( nent, sizeof ( int ) );
  ent     = (void **) calloc ( nent, sizeof ( void * ) );

  /* retorna a lista de faces selecionadas */
  sel = GetBuildSelFace ( );

  /* para cada face */
  while ( sel != NULL)
  {
    /* atribui o identificador da face a um campo do vetor ent. */
    ent[cont] = sel.edgeno;

   /* atribui o tipo do face, registrado na função MisRegGeomEntities, a um campo do  vetor type. */
    type[cont] = GetFaceType ( sel);

   /*atribui o nome do face, registrado na função MisRegGeomEntities, a um campo do vetor
name.*/
    name[cont] = GetFaceName ( sel );

   sel = sel->nxt;
  }

  /* mesmo procedimento para as arestas e vértices. */
   ....

  /*  chama a função recebida como parâmetro */
  endaction (nent, name, type, ent);

 return 1;
}
```

Figura 3.6 - Código computacional em C que implementa a função *MisGetNSelect*.

3.3 Participação do ESAM nas Diversas Etapas de uma Simulação

Esta seção descreve a utilização do ESAM nas etapas envolvidas no processo de simulação do SIGMA (definição da geometria do modelo, especificação dos atributos, geração da malha de elementos finitos, análise numérica e visualização dos resultados).

3.3.1 Definição da Geometria e Especificação dos Atributos do Modelo

Sempre que um novo modelo vai ser criado pelo pré-processador MTool, o serviço de inicialização *EsamCreateGeomBasedModel* (seção 2.4.1) deve ser chamado para criar um objeto representando este modelo para o ESAM. A Figura 3.7 ilustra a parte do código computacional do MTool que utiliza este serviço.

```
void MtoolCreateModel ( void )
{
 /* Declara um variável do tipo EsamObject, que vai representar o objeto criado
 ** pelo sistema ESAM.*/
 static EsamObject model;

 …

 /* Verifica se um objeto referente a um modelo já existe. Em caso arfirmativo, destroi este objeto.*/
 if ( EsamHasGeomBasedModel ( model ) )
 {
   EsamDeleteGeomBasedModel ( model );
 }

 /* Cria um objeto representado um modelo baseado em geometria.*/
 model = EsamCreateGeomBasedModel ( );

 /*  Coloca o objeto criado como corrente. Todas as operações realizadas sobre o objeto
 ** vão atuar sobre este objeto corrente.*/
 EsamSetCurrGeomBasedModel ( model );

  …

}
```

Figura 3.7 - Instruções em C para criação de um objeto representando um modelo baseado em geometria no ESAM.

Associado ao modelo, o MTool cria sólidos que contêm as informações relativas às entidades geométricas. O ESAM deve, então, criar objetos para representar estes sólidos chamando o serviço *EsamCreateSolid* (seção 2.4.3). A Figura 3.8 mostra a parte do código computacional do MTool que utiliza este serviço.

O pré-processador MTool possui entidades geométricas, tais como, vértices, curvas e regiões, que são responsáveis pela definição da geometria do modelo. Quando operações de eliminação (*delete*), divisão (*split*) e união (*join*) são realizadas sobre essas entidades, alguns serviços do ESAM (*EsamChkDeleteGeomEntity*,

EsamChkSplitGeomEntity e *EsamChkJoinGeomEntity*) devem ser chamados para verificar se estas operações podem ser realizadas sobre os seus objetos. Por exemplo, a união de entidades geométricas que possuem materiais diferentes associados a ela não deve ser realizada, pois não é possível definir qual destes materiais será predominante na nova entidade criada. Se a realização destas operações for possível, deve-se, então, chamar serviços do ESAM (*EsamDeleteGeomEntity*, *EsamSplitGeomEntity* e *EsamJoinGeomEntity*) que efetuam essas operações sobre os objetos correspondentes. A Figura 3.9 mostra uma parte do código computacional que efetua a operação de união de duas entidades geométricas do modelo.

```
void MtoolCreateSolid ( void )
{
 /* Declara um variável do tipo EsamObject, que vai representar o objeto criado
 ** pelo sistema ESAM.*/
 static EsamObject solid;

 …

 /* verifica se o objeto representando este sólido ainda não foi criado. Em caso arfimativo, ele
 ** é criado. Recebe como parâmetro o endereço do sólido, no contexto do MTool. */
 if ( !EsamHasSolid ( addrsolid ) )
 {

    /* Cria um objeto representando o solido. Recebe como parâmetro o endereço do
    ** sólido, no contexto do MTool. */
    solido = EsamCreateSolid (addrsolid);

    /*  Coloca o objeto criado como corrente. Todas as operações realizadas sobre o objeto
   ** vão atuar sobre este objeto corrente.*/
   EsamSetCurrSolid ( solido );
 }

 …

}
```

Figura 3.8 - Instruções em C para criação de um objeto representando um sólido no ESAM.

Em relação aos atributos, O ESAM possui serviços que permitem a especificação dos atributos do SIGMA. O serviço *EsamCreateAttrib*, por exemplo, permite que atributos de tipos pré-definidos no arquivo de configuração possam ser criados pelo usuário. O usuário pode, por exemplo, criar vários atributos sem associá-los imediatamente às entidades do modelo. Esses atributos ficam armazenados em uma lista interna do

sistema ESAM, podendo ser acessados em qualquer instante pelo usuário do SIGMA. A Figura 3.10 mostra um diálogo utilizado para a criação de atributos do tipo *Constant Load Distributed*.

```
/* União de duas entidades geométricas do modelo. */
void MtoolJoinGeomEntity ( void )
{

…

 /* Chama serviço que verifica se as entidades geométricas podem ser unidas.
 ** Passa como parâmetros os endereços e o tipo das entidades. */
 if ( EsamChkJoinGeomEntity ( addr1, addr2, type ) )
 {

   /* Chama serviço que une as entidades geométricas. Passa como parâmetros
   ** os endereços das entidades antigas, o endereço da nova entidade  e o tipo destas entidades. */
   EsamJoinGeomEntity ( addrold1, addrold2, addrnew, type )
 }

…

}
```

Figura 3.9 - Exemplo de manipulação de entidades geométricas que tem seus atributos gerenciados pelo ESAM.

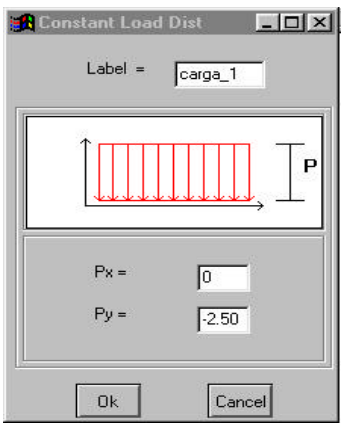

Figura 3.10 - Diálogo utilizado para a criação de atributos do tipo *Constant Load Distributed*.

Os outros serviços de especificação do sistema ESAM utilizados no SIGMA são: *EsamAttachAttrib*, que associa um atributo às entidades selecionadas do modelo; *EsamModifyAttrib*, responsável pela modificação das variáveis referentes às propriedades de um atributo já criado; *EsamDettachAttrib*, responsável pela retirada de atributos associados às entidades do modelo; *EsamDeleteAttrib*, responsável pela

eliminação de um atributo previamente criado (esta operação só é realizada se o atributo não estiver associado a entidades do modelo); *EsamCopyAttrib*, que faz uma cópia de um determinado atributo. Além destes serviços de especificação, o SIGMA utiliza um serviço de consulta, *EsamInquireAttrib*, que permite ao usuário obter informações sobre as variáveis relativas a propriedades de atributos já criados.

Além disso, pode-se obter a relação dos atributos associados às entidades geométricas do modelo, através de serviços de consultas do sistema ESAM. O serviço *EsamInquireEntity* informa quais atributos estão associados a uma determinada entidade, enquanto que o serviço *EsamInquireEntityFromAttrib* fornece quais são as entidades associadas a um determinado atributo.

Conforme foi dito no capítulo anterior, é possível estabelecer grupos de atributos definidos no arquivo de configuração, ou seja, agrupar em um mesmo conjunto tipos de atributos diferentes. Como no pré-processador MTool os atributos são associados às entidades previamente selecionadas do modelo, no SIGMA foram criados grupos de atributos referentes aos tipos de entidades disponíveis no MTool (atributos que podem ser associados às faces do modelo estão em um grupo diferente dos atributos que são associados às arestas, por exemplo). Com isso, foi possível estabelecer *menus* dinâmicos contendo o grupo de atributos referente ao tipo das entidades selecionadas do modelo. A Figura 3.11 mostra um *menu* dinâmico contendo os tipos de atributos que podem ser aplicados às entidades do tipo *Face*, definida no MTool.

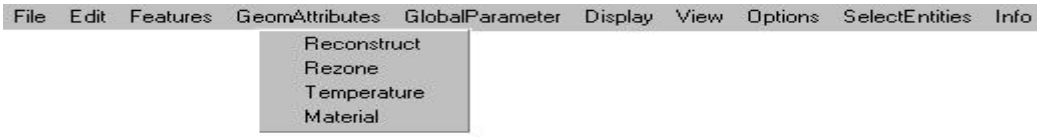

Figura 3.11 - *Menu* dinâmico contendo os atributos que podem ser associados às entidades do tipo *Face*.

3.3.2 Conversão da Malha de Elementos Finitos

No instante da conversão da malha de elementos finitos pelo pré-processador MTool, o serviço *EsamCreateNumModel* (seção 2.4.3) deve ser chamado para criar um objeto representando esta malha para o ESAM. Além disso, através do serviço

EsamInitNumModel (seção 2.4.3), o ESAM faz o mapeamento dos atributos associados às entidades geométricas do modelo para a malha de elementos finitos. A Figura 3.12 mostra a parte do código computacional que usa o ESAM para criar um objeto representando a malha de elementos finitos e para fazer o mapeamento dos atributos.

```
void MtoolCreateMesh ( void )
{

 /* Declara um variável do tipo EsamObject, representando o objeto que será
 **  criado pelo sistema ESAM.*/
 static EsamObject nummodel;

 …

 /* Verifica se o objeto referente a malha já existe. Em caso arfirmativo, destroi este objeto.*/
 if ( EsamHasNumModel ( nummodel ) )
 {
   EsamDeleteNumModel (nummodel );
 }

 /*  Cria um objeto representando uma malha de elementos finitos associada modelo.
 ** O endereço da malha de elementos finitos é passado como parâmetro.*/
 nummodel = EsamCreateNumModel (addrnummodel);

 /*  Coloca o objeto criado como corrente. Todas as operações realizadas sobre o modelo
  ** referentes a malha de elementos finitos vão atuar sobre este objeto corrente.*/
 EsamSetCurrNumModel ( nummodel );

  /*  Inicializa os dados referentes a esta malha, para o ESAM. Aqui é realizado
  ** o mapeamento dos atributos das entidades geométricas para a malha de elementos finitos. */
 EsamInitNumModel ( );

 …

}
```

Figura 3.12 - Instruções em C para a criação de um objeto para o ESAM, representando a malha de elementos finitos do MTool, e para o mapeamento dos atributos das entidades geométricas para a malha de elementos finitos.

3.3.3 Malha de Elementos Finitos

Como foi dito anteriormente, as alterações realizadas no sistema ESAM possibilitou a associação de atributos diretamente à malha de elementos finitos. Com isso, todas as funcionalidades referentes a manipulação de atributos também estão disponíveis no

ambiente de malha de elementos finitos. Por exemplo, pode-se a associar um atributo a elementos previamente selecionados da malha através do serviço de especificação *EsamAttachAttrib* (seção 2.4.2).

É importante ressaltar que os atributos associados diretamente às entidades de malha têm preferência sobre os atributos aplicados às entidades geométricas nas quais essas entidades de malha são associadas. Por exemplo, se um atributo do tipo *Material* foi aplicado a uma face do modelo geométrico e um outro atributo, deste mesmo tipo, foi aplicado a um elemento associado a esta face, apenas este último é considerado.

Da mesma forma como foi dito na seção 3.3.1, a relação dos atributos associados às entidades da malha pode ser obtida através de serviços de consultas do sistema ESAM (*EsamInquireEntity* e *EsamInquireEntityFromAttrib*).

3.3.4 Análise Numérica

A comunicação entre as etapas de pré-processamento e análise é realizada através de arquivos de dados, em formatos específicos dos programas de análise. Como os atributos da simulação são gerenciados pelo sistema ESAM, é possível que classes e métodos sejam definidos pelo usuário com o objetivo de configurar estes arquivos de dados. Assim, duas novas classes (*Aeepecd* e *Anvec*), subclasses da classe *Output*, com seus respectivos métodos foram criadas no arquivo de configuracão para especificar o formato dos arquivos de dados referentes aos programas de análise AEEPECD e ANVEC.

No SIGMA, o programa de análise numérica que será utilizado depende do tipo da análise estabelecido pelo usuário. Este tipo é definido pelo atributo *Analysis Type*. Para os casos onde o usuário define que a análise será elástica ou elasto-plástica, o programa numérico utilizado será o AEEPECD. Se for definido que a análise será visco-elasto-plástica, o programa utilizado para a análise numérica será o ANVEC. Esta informação deve ser passada para o ESAM através do serviço

EsamSetFormatFile (seção 2.4.5) que é responsável pela estipulação do formato corrente referente ao tipo de análise que será realizada. As operações realizadas sobre a análise, tais como a verificação e a escrita dos dados referentes a ela, atuam sobre este formato corrente.

Para salvar um arquivo de dados para a análise definido pelo usuário, o serviço *EsamWriteNumModelFile* (seção 2.4.5) deve ser chamado passando o nome do arquivo como parâmetro (nome do projeto adicionado da extensão DAT). Este serviço dispara o método *writetofile* referente ao formato corrente (AEEPECD e ANVEC) pré-definido pelo usuário, que escreve os dados relativos a este formato. Para verificar a compatibilidade dos dados relativos à malha e aos atributos associados a ela, deve-se chamar o serviço *EsamCheckNumModelFile* (seção 2.4.5), que dispara o método *check* referente ao formato corrente. A Figura 3.13 mostra a parte do código computacional que usa o ESAM para gerar o arquivo de dados que serão lidos pelo módulo de análise.

```
int MtoolWriteFormat ( void )
{
  …

  /* verifica a compatibilidade dos dados referentes ao tipo de análise definido pelo usuário. */
  if ( EsamCheckNumModelFile ( ) )
  {

    /* escreve os dados referentes ao tipo de análise definido pelo usuário. Passa o nome
    ** do arquivo de dados. Este nome é composto pelo nome do projeto adicionado da
    ** extensão dat. */
     EsamWriteNumModelFile ( filename );
   }
   return 1;
}
```

Figura 3.13 - Parte do código computacional em C que usa o ESAM para gerar os dados que serão lidos pelos programas de análise numérica.

A Figura 3.14 mostra a parte do código computacional em Lua implementado no método *writetofile* da classe *Aeepecd*, responsável pela escrita dos dados relacionados aos atributos do tipo *Support*, associados aos nós da malha de elementos finitos, no arquivo de dados que será lido pelo programa AEEPECD.

```
function Aeepecd:writetofile ( filename )
```

```
-- abre o arquivo de dados para gravação.
writeto ( filename)
-- retorna o modelo corrente.
 local objmodel = esam_getcurrentmodel ( )

 ...

--  retorna todos os atributos do tipo Support associados aos nós da malha.
 local attrsupp =objmodel:getnodeattribstype ("Support")

-- verifica se algum atributo do tipo Support foi associado aos nós.
if not attrsupp then
  return nil
end

-- laço percorrendo todos os atributos do tipo support.
local i = 1
while attrsupp[i] do

  -- retorna os nós que possuem o atributo attrsupp[i].
  local nodes = objmodel:getnodeswithattribs (attrsupp[i])
  -- laço percorrendo estes nós.
  local j = 1
  while nodes[j] do

    -- escreve os ids dos nós.
    write ( format ( "%5i", nodes[j]:getid( ) ), "s")
    -- escreve as restrições em x e y associadas a esse nó.
    write ( format ( "%5i",  attrsupp[i]:getrest_x( ) ), "s")
    write ( format ( "%5i", attrsupp[i]:getrest_y ( ) ), "s")
    write ( "\n")

    j = j + 1
  end
  i = i + 1
end

...

-- fecha o arquivo de dados aberto para gravação.
writeto( )
end
```

Figura 3.14 - Parte do código computacional em Lua implementado no método *writetofile* da classe *Aeepecd*, responsável pela escrita dos dados relacionados aos atributos do tipo *Support*.

Assim como todas as instruções de configuração do ESAM que são escritas na linguagem de extensão Lua, a definição do formato de entrada do programa de análise pode ser alterada pelo usuário configurador. Isto significa que, para o SIGMA utilizar um outro programa de análise, é preciso apenas substituir o arquivo de configuração com a definição do formato do outro programa. Isto é feito sem a recompilação ou geração de outro executável do SIGMA.

3.3.5 Visualização dos Resultados

O módulo MView, responsável pela visualização dos resultados no SIGMA, não possui informações da geometria do modelo que originou a malha de elementos finitos. Entretanto, como foi dito anteriormente, a nova arquitetura do sistema ESAM permite que os atributos definidos na fase de pré-processamento possam ser consultados e visualizados neste módulo. Para isso, o serviço *EsamCreateMeshBasedModel* (seção 2.4.1) deve ser chamado para criar um objeto que vai representar um modelo baseado em malha no ESAM. A Figura 3.15 mostra a parte do código computacional que usa o ESAM responsável pela criação de um objeto representando um modelo baseado em malha.

As informações referentes aos atributos associados às entidades da malha podem ser obtidas através dos serviços de consultas *EsamInquireEntity* (seção 2.4.6) e *EsamEntityInquireFromAttrib* (seção 2.4.6) que permite, por exemplo, visualizar todos os elementos que possuem um determinado tipo de atributo.

```
void CreateMeshBasedModel ( void )
{
 /* Declara um variável do tipo EsamObject, que vai representar o objeto criado
 ** pelo sistema ESAM.*/
 static EsamObject nummodel;
 …
 /* Verifica se o objeto referente ao modelo já existe. Em caso arfirmativo, destroi este objeto.*/
 if ( EsamHasMeshBasedModel ( nummodel ) )
 {
   EsamDeleteMeshBasedModel ( nummodel );
 }
```

```
/* Cria um objeto para representar o modelo baseado em malha.*/
nummodel = EsamCreateMeshBasedModel ( );

/*  Coloca o objeto criado como corrente. Todas as operações realizadas sobre o modelo
** vão atuar sobre este objeto corrente.*/
EsamSetCurrMeshBasedModel (num model );
…
}
```

Figura 3.15 - Instruções em C responsáveis pela criação de um objeto representando um modelo baseado em malha no ESAM.

3.4 Leitura e Gravação dos Dados Referentes aos Atributos do Modelo

O sistema ESAM permite, através de serviços de entrada e saída de dados, que os atributos criados e associados às entidades geométricas e de malha (MTool e MView), e os objetos referentes a essas entidades possam ser armazenados de forma consistente com o modelo, para que eles sejam recuperados quando este modelo for lido. Os serviços *EsamReadAttribModel* e *EsamWriteAttribModel* são utilizados para leitura e gravação dos dados referentes aos atributos, enquanto que os serviços *EsamReadModelFile, EsamWriteModelFile*, *EsamReadMeshFile* e *EsamWriteMeshFile* são utilizados para leitura e gravação dos objetos referentes às entidades geométricas e de malha criados pelo ESAM.

No SIGMA, quando um modelo geométrico ou numérico for salvo, um arquivo gerado pelo ESAM, contendo os atributos e as entidades ao qual estes atibutos foram associados, também será gravado. Da mesma forma, quando um arquivo de dados contendo as informações do modelo for lido, o arquivo do ESAM correspondente também o será. O nome do arquivo que contém os dados referentes aos atributos e aos objetos que representam as entidades geométricas do modelo é formado pelo nome do projeto criado no MTool adicionado da extensão *top*, enquanto que o nome do arquivo contendo os dados referentes aos atributos e aos objetos representando as entidades da malha é formado pelo nome do projeto adicionado da extensão *msh*. A Figura 3.16 mostra a parte do código computacional que usa o ESAM para ler e salvar os dados

referentes aos atributos e aos objetos correspondentes às entidades geométricas do modelo.

```
/* Parte de código responsável pela gravação dos dados referentes aos atributos. */
void MtoolWriteModel ( void )
{
 …
 /*  abre um arquivo de dados para gravação. Recebe como parâmetro o nome do arquivo,
 ** que deve ser formado pela combinação do nome do projeto com a extensão top. */
 EsamOpenWriteFile ( filename );
 /* grava os dados relacionadas aos atributos do modelo.*/
 EsamWriteAttribModel ( );
 /*   grava as informações relacionadas aos objetos correspondentes às entidades
 ** geométricas do modelo. */
 EsamWriteModelFile ( );
 /* fecha o arquivo aberto para gravação dos dados referentes aos atributos. */
 EsamCloseWriteFile ( );
 …
}
/* Parte do código responsável pela leitura dos dados referentes aos atributos. */
void MtoolReadModel ( void )
{
 …
 /*  abre um arquivo de dados para leitura. Recebe como parâmetro o nome do arquivo,
 ** que deve ser formado pela combinação do nome do projeto com a extensão top. */
 EsamOpenReadFile ( filename );
 /* lê os dados relacionadas aos atributos do modelo.*/
 EsamReadAttribModel ( );
 /*  lê as informações relacionadas aos objetos que correspondem às entidades
 ** geométricas do modelo. */
 EsamReadModelFile ( );
 /* fecha o arquivo aberto para leitura dos dados referentes aos atributos. */
 EsamCloseReadFile ( );
 …
}
```

Figura 3.16 - Instruções em C responsáveis pela leitura e gravação dos dados relacionados aos atributos e aos objetos referentes às entidades geométricas do modelo.

Além dos serviços de leitura e gravação dos atributos (*EsamReadAttribModel* e *EsamWriteAttribModel*) relacionados a um determinado modelo, é possível ler e escrever atributos sem que eles estejam associados a um modelo, através dos serviços *EsamImportAttrib* e *EsamExportAttrib*. Os arquivos gerados pelo serviço *EsamExportAttrib* possui um formato simples que permite, por exemplo, a criação de atributos pelo usuário através da edição destes arquivos. A Figura 3.17 ilustra um exemplo do formato deste arquivo contendo um atributo do tipo *Material*, definido no arquivo de configuração.

```
crtattrib {
        classname = 'Material',
        label     = 'mat1',
        epls      = 'OBJ11',
        mass_esp  = 1.00,
        ko        = 0.78,
        kxt       = 0,
        kyt       = 0,
        alft      = 0,
        e         = 2100.00,
        nu        = 0.3,
        inda      = 1,
        color     = '0 0 1',
}

crtattrib {
        classname = 'Drained',
        label     = 'OBJ11',
        c         = 55.00,
        fu        = 0.00,
}
```

Figura 3.17 - Formato de arquivos exportados pelo sistema ESAM.

Neste exemplo, o atributo *Material* possui características elasto-plásticas, sendo essas características definidas através de um objeto da classe *Drained*, referenciado pela variável *epls*. As variáveis descritas no atributo correspondem às variáveis estabelecidas na definição da classe de atributos no arquivo de configuração.

3.5 Configuração do SIGMA

Definidos os passos utilizados para a integração do sistema ESAM com os módulos de pré-processamento MTool e pós-processamento MView, são considerados, nesta seção, os procedimentos utilizados para a configuração do SIGMA (redefinição de métodos referentes as subclasses da classe *Geometry*, que representam as entidades geométricas do modelo e a configuração dos atributos utilizados no SIGMA).

3.5.1 Redefinição de Métodos Associados às Entidades Geométricas

A *Core Classe Geometry*, definida no ESAM, implementa métodos relacionados aos atributos associados às entidades geométricas do modelo. Conforme foi dito anteriormente (seção 2.3.6), o ESAM possui um mecanismo que cria automaticamente subclasses da classe *Geometry* de acordo com a dimensão da entidade definida no pré-processador (no caso, o MTool). O ESAM não cria, no entanto, métodos para estas subclasses que herdam, a princípio, todos os métodos da classe *Geometry*. Deste modo, estas subclasses, que representam as entidades geométricas do MTool, apresentam o mesmo comportamento, pois utilizam os mesmos métodos (definidos na classe *Geometry*) no que diz respeito aos atributos. Assim, o tratamento dado a uma curva do modelo é o mesmo dado a uma face, por exemplo. Entretanto, muitas vezes é desejável que um determinado tipo de entidade tenha características próprias, com relação a manipulação de atributos. Por exemplo, a maioria dos programas de análise permite que apenas um atributo do tipo *Material* seja associado a cada elemento do modelo numérico. Assim, como os métodos referentes aos objetos da classe *Geometry* não possuem restrições relativas aos tipos de atributos associados às entidades, é necessário que esses métodos possam ser sobrepostos, de forma que essas restrições sejam inseridas no contexto de cada entidade do modelo. Para a redefinição destes métodos, é necessário que o usuário tenha conhecimento sobre os nomes das entidades (que, para o ESAM, representam o nome das subclasses da classe *Geometry*) correspondentes que são definidos pela aplicação. A Figura 3.18 ilustra a parte do código computacional, escrito em Lua, referente a redefinição do método *attach* da entidade *Face*, definida no MTool.

Este exemplo mostra que apenas um objeto de uma mesma classe de atributos pode ser associado a cada objeto geométrico do tipo *Face*. O método, entretanto, permite que um objeto da classe de atributos já associado à entidade seja substituído por outro da mesma classe.

-- Retorna o objeto referente ao tipo Face, registrado pelo MTool.
_FACE = esam_getclass (“Face”)

--Redefinição do método attach referente ao tipo Face. Recebe como

```
--parâmetro o atributo que será associado à entidade..
function _FACE:attach ( attrib )
    -- Retorna a classe do atributo passado como parâmetro.
    local attrclass = attrib:getclass( )
    -- Retorna todos os atributos de um determinado tipo, associado à entidade.
    local attrtype = self:getattribstype (attrclass)
    -- Se não existe atributo deste tipo, então coloca o atributo passado como
    -- parâmetro na lista de atributos da entidade e encerra-se o procedimento.
    if not attrtype then
      GEOMETRY.attach (self,attrib)
      return 1
    end

    -- Se um atributo deste tipo já existe, pergunta se o usuário deseja substituí-lo.
    -- Em caso afirmativo, realiza-se o procedimento de substituição.
    if ( esam_popupmessage("Warning", " Attribute type existing!\n Do you want replace?") ) then
      -- Percorre a lista de atributos do tipo desejado.
      local i = 1
      while attrtype[i] do
        -- Verifica se o tipo é igual ao do atributo passado como parâmetro.
        if ( attrtype[i]:getclass( ) == attrclass ) then
            -- remove o atributo da entidade.
            if self:removeattribute ( attrtype[i] ) then
              -- remove a entidade do atributo.
              attrtype[i]:removeentity(self)
            end
        end
        i = i + 1
      end
      -- coloca o atributo passado como parâmetro na lista de atributos da entidade.
      GEOMETRY.attach ( self, attrib )
      -- Encerra-se o procedimento.
      return 1
    end
    -- Caso o usuário não deseja fazer a substituição, encerra-se o procedimento.
    return nil
end
```

Figura 3.18 - Exemplo de redefinição do método *attach* da classe *Face*, no SIGMA.

3.5.2 Criação dos Atributos do SIGMA

Outro passo utilizado na configuração do SIGMA diz respeito à definição dos atributos necessários para a realização da análise numérica utilizando os programas AEEPECD e ANVEC. As classes e métodos referentes aos atributos foram definidos em um arquivo de configuração denominado *attsigma.lua*, que é carregado pelo sistema, contendo todas as informações necessárias para a configuração destes atributos.

Os atributos definidos no SIGMA podem ser divididos em dois grupos distintos. O primeiro grupo considera os atributos que podem ser associados diretamente às entidades geométricas (vértices, curvas, etc.) ou às entidades de malha (nós e elementos). Exemplos desses atributos são *Material*, que representam os materiais que podem ser associados às regiões do modelo ou aos elementos da malha, e *Pressure*, que representam um carregamento associado às curvas, na direção normal a elas, no plano que contém o modelo (Figura 3.19). A Figura 3.20 mostra o diálogo utilizado na captura dos dados referentes ao atributo *Pressure*.

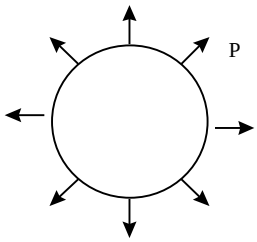

Figura 3.19 - Pressão associada a uma curva.

Figura 3.20 - Diálogo para a captura de dados referentes ao atributo *Pressure*.

O segundo grupo de atributos considera os atributos que possuem informações globais relacionadas a análise numérica que será realizada. Entre os atributos que fazem parte deste grupo estão *Printer Control*, que contêm variáveis referentes as impressões de dados nos arquivos de saída gerados pelos programas AEEPECD e ANVEC, e *Analysis Type*, que fornecem informações sobre o tipo de análise que deve ser considerado pelos programas de análise (no SIGMA só é considerado estado plano de tensão, estado plano de deformação ou modelo assimétrico). A Figura 3.21 mostra o diálogo para captura de dados do atributo *Analysis Type*.

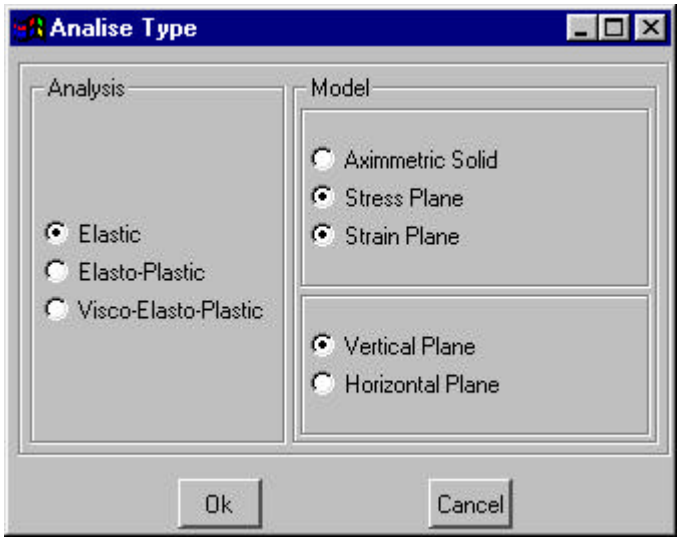

Figura 3.21 - Diálogo para captura de dados referente ao tipo de análise.

Uma descrição detalhada dos atributos (*Material*, *Rezone*, *Support*, *Pressure*, *Numerical Control*, etc.) utilizados pelos programas de análise numérica AEEPECD e ANVEC pode ser obtida no manual do SIGMA [TECG5 1998].

3.5.2.1 Exemplo Ilustrando a Criação de Atributos

Esta seção mostra um exemplo ilustrando a criação de atributos pelo ESAM. Cada atributo é representado por uma classe contendo variáveis (referentes as propriedades do atributo) e métodos que respondem pelas informações referentes a classe (ou aos objetos da classe). Para ilustrar a criação de atributos, o atributo *Load Concentrated* (definido no SIGMA), representando um tipo de carregamento que pode ser associado aos vértices do modelo, é tomado como exemplo. A classe criada para representar este atributo possui as variáveis *px* e *py* que representam as componentes horizontal e vertical da carga. A Figura 3.22 mostra a classe *Load Concentrated* sendo criada.

```
LOADCONCENTRATED = crtclass {
                        name  = "Load Concentrated",      -- nome da classe
                        parent= ATTRIBUTE,                -- superclasse desta
classe
                        vars   = { "label", "px", "py" },  -- parâmetros da classe
                        type   = {"s", "f", "f"},          -- tipo dos parâmetros
                        group  = VERTEX                    -- grupo do atributo
```

```
}
```

Figura 3.22 - Criação da classe referente ao atributo *Load Concentrated*.

A variável *label* do campo *vars*, estabelece um nome para referenciar cada objeto desta classe que vai ser criado, sendo responsável pela identificação do atributo para o sistema ESAM. O campo *type*, mostrado no exemplo, indica o tipo correspondente a cada parâmetro definido no atributo, que pode ser "*f*" (real), "*i*" (inteiro), "*s*" (cadeia de caracteres), "*t*" (vetor) e "*o*" (instância para outro objeto). O campo *group* indica que todos os atributos desta classe estão agrupados no grupo *VERTEX* (conjunto dos atributos que podem ser associados aos vértices do modelo).

Criada a classe referente ao atributo deve-se, então, definir os métodos relacionados a essa classe. Alguns métodos devem ser, obrigatoriamente, descritos pelo usuário configurador do sistema, pois eles são utilizados internamente pelo ESAM para a manipulação do tipo de atributo que está sendo criado. Nestes métodos, a variável *self* representa o objeto corrente de cada classe, enquanto que as variáveis *Lx* e *Ly* correspondem aos parâmetros do diálogo utilizados para a captura dos dados do objeto. Segue uma descrição de cada um destes métodos obrigatórios utilizados na definição da classe *Load Concentrated*.

- *initialize* - este método, mostrado na Figura 3.23, é responsável pela inicialização das variáveis da classe, onde todos os objetos criados, referentes a esta classe, terá suas varáveis inicializadas com estes valores. No exemplo ilustrado, as variáveis *px* e *py* são inicializadas com os valores 0.00 e 15.00, respectivamente.

```
-- Este método inicializa as variáveis da classe. Desta forma, todos os atributos desta classe
-- que vai ser criado são inicializados com os valores descritos neste método.

function LOADCONCENTRATED:initialize()
  self.px = 0.00
  self.py = 15.00
end
```

Figura 3.23- Descrição do método *initialize* referente ao atributo *Load Concentrated*.

- *dlgdescriptor* - este método, mostrado na Figura 3.24, define o diálogo para a captura dos dados referentes ao atributo. Neste exemplo, são utilizados alguns

elementos de interface disponíveis no *toolkit* IupLua [TECG4 1995]. A Figura 3.25 mostra o diálogo definido neste método, utilizado para a captura dos dados referentes ao atributo.

```
-- Este método define o diálogo que será utilizado para a captura dos dados relativos aos atributos
-- do tipo Load Concentrated. retorna o diálogo criado.

function LOADCONCENTRATED:dlgdescriptor()
  self.Lb = edfield { prompt = "Label = ", value = self.label }
  self.Lx = edfield { prompt = "Px = ",    value = self.px }
  self.Ly = edfield { prompt = "Py = ",    value = self.py }
  return vdialog {
            self.Lb,
            self.Lx,
            self.Ly;
            title = "Load Concentrated"
          }
end
```

Figura 3.24- Descrição do método *dlgdescriptor* relacionado ao atributo *Load Concentrated*.

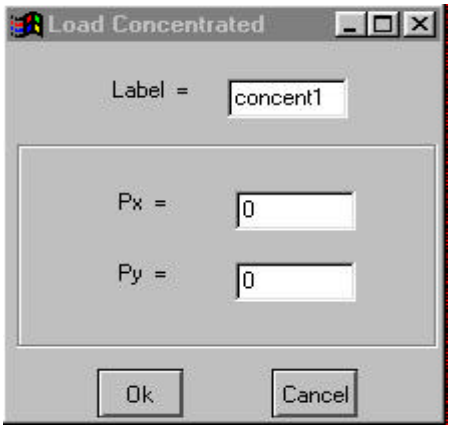

Figura 3.25 - Diálogo, definido no método *dlgdescriptor*, criado para a captura dos dados referentes ao atributo *Load Concentrated*.

- *validate* - este método (Figura 3.26) verifica se os dados fornecidos no diálogo utilizado na captura de dados do atributo são válidos. No exemplo mostrado, se uma cadeia de caracteres for inserida em qualquer um dos campos utilizados para captura dos dados do atributo, o objeto referente a este atributo não será criado.

```
-- Este método verifica se os dados fornecidos no diálogo são válidos. Recebe como parâmetro
-- um flag que indica se o botão cancel do diálogo foi acionado.
```

```
function LOADCONCENTRATED:validate(flag)
  if not flag then
    return nil
  end
  if  type(self.Lx) ~= "number" or type(self.Ly) ~= "number" then
     return nil
  end
  return 1
end
```

Figura 3.26 - Descrição do método *validate* referente ao atributo *Load Concentrated*.

- *valuetodlg* - este método, ilustrado pela Figura 3.27, transporta os dados do objeto para o diálogo utilizado na captura de dados do atributo. Neste exemplo, os valores armazenados nas variáveis do objeto *px* e *py* são atribuídos aos elementos de interface do diálogo *Lx* e *Ly*, respectivamente.

```
-- Este método transporta os valores das variáveis do objeto corrente  px e py para os objetos de
-- interface referenciados por Lx e Ly, ou seja, coloca nos respectivos campos do diálogo os
-- valores das variáveis do atributo.

function LOADCONCENTRATED:valuetodlg()
  self.Lb = self.label
  self.Lx = self.px
  self.Ly = self.py
end
```

Figura 3.27 - Descrição do método *valuetodlg* referente ao atributo *Load Concentrated*.

- *valuefromdlg* - este método (Figura 3.28) transporta os dados do diálogo utilizado para a captura dos dados do atributo para as variáveis do objeto que representa este atributo. Neste exemplo, os valores contidos nos elementos de interface *Lx* e *Ly* são atribuídos as variáveis do objeto *px* e *py*, respectivamente.

```
-- Este método transporta os valores dos objetos de interface do diálogo Lx e Ly, para as variáveis
-- px e py do objeto corrente.

function LOADCONCENTRATED:valuefromdlg()

  self.Label = self.Lb
  self.px = self.Lx
  self.py = self.Ly
end
```

Figura 3.28 - Descrição do método *valuefromdlg* relacionado ao atributo *Load Concentrated*.

Além dos métodos obrigatórios descritos acima, outros métodos podem ser definidos com o objetivo, por exemplo, de fornecer informações referentes as variáveis definidas em cada objeto criado. A Figura 3.29 mostra os métodos *getload_x* e *getload_y* referentes ao atributo *Load Concentrated* que retornam, respectivamente, os valores da carga nas direções x e y.

```
function  LOADCONCENTRATED:getload_x ()   -- retorna o parâmetro px do objeto, referente
                                              -- a componente horizontal da carga.
  return self.px
end
-------------------------------------------------------
function LOADCONCENTRATED:getload_y()    -- retorna o parâmetro py do objeto, referente
                                              -- a componente vertical da carga.
  return self.py
end
```

Figura 3.29 - Exemplo de métodos que retornam as variáveis definidas nos objetos representando os atributos do tipo *Load Concentrated*.

4. Exemplo

Neste capítulo é demonstrada a aplicação da metodologia desenvolvida neste trabalho através de um exemplo da utilização do Sistema Integrado de Geotecnia para Múltiplas Análises (SIGMA). São mostradas as fases envolvidas na simulação e ilustrados alguns exemplos dos atributos especificados no arquivo de configuração referentes aos tipos de análise que este sistema se propõe a realizar.

O exemplo apresentado simula a escavação de um poço no espaço bidimensional. A Figura 4.1 mostra uma seção transversal do poço que será analisado. A região onde o poço será escavado é constituída basicamente de arenito. Os dados referentes ao arenito e ao poço são mostrados a seguir:

- Características do arenito:
    - Módulo de Elasticidade (E): 2.85×10^3 MPa
    - Coeficiente de Poisson (n): 0.36
    - Coesão (c'): 4.99 MPa
    - Ângulo de Atrito (f'): 38.44°
    - Coeficiente de Empuxo no Repouso (K_0): 0,42
    - Coeficiente de Biot: 0.7
    - Coeficiente de Expansão Térmica (a): 2.947×10^{-8} °C^{-1}
    - Permeabilidade nas Direções x e y: 130 milidarcy

- Características do poço:
    - Profundidade: 2393 m
    - Peso Específico Médio: 25.0 kN/m^3.

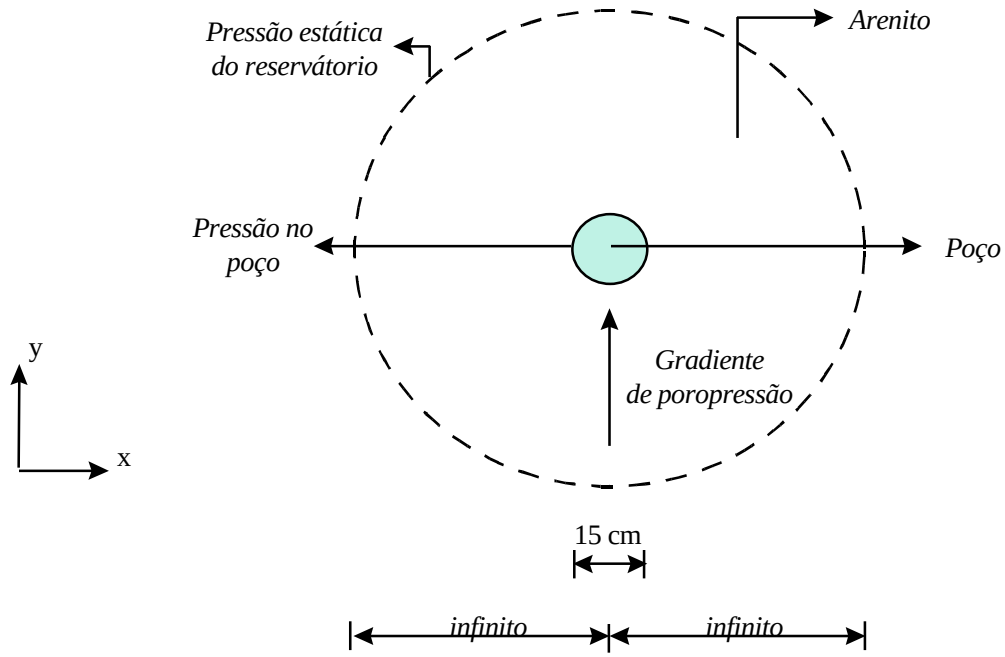

Figura 4.1 - Poço que será escavado.

A Figura 4.2 mostra a variação entre a pressão estática do reservatório P_E e a pressão no poço P_w (depleção). Esta depleção (DP = 9.80 kg/cm^2) provoca um gradiente de poropressão na direção radial do poço.

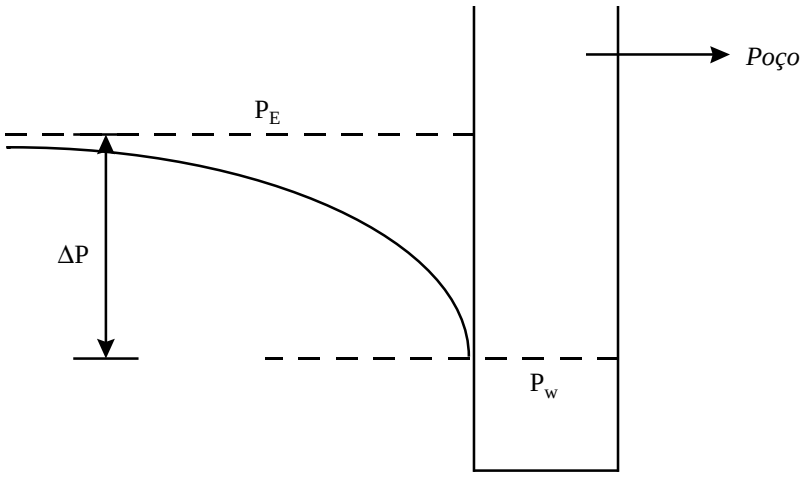

Figura 4.2 - Depleção do reservatório.

O processo de análise é realizado em três etapas distintas: na primeira considera-se o efeito das tensões iniciais associados ao modelo (peso próprio, por exemplo); na segunda etapa é realizada a escavação do poço. Nesta etapa são consideradas forças de

escavação atuando sobre o modelo; finalmente, na terceira etapa considera-se o gradiente de poropressão referente à aplicação da depleção do reservatório.

Os atributos envolvidos nesta simulação são:

- Material elasto-plástico definido pelos valores do módulo de elasticidade, coeficiente de Poisson, ângulo de atrito, coesão, empuxo lateral, coeficiente de expansão térmica e coeficientes de condutibilidade térmicas nas direções *x* e *y*.
- Escavação definida pelo valor que indica o instante em que a camada de solo será escavada.
- Poropressão aplicada na borda do poço e em pontos infinitamente distantes desta borda. Este atributo é definido pelo valor correspondente à temperatura associada ao modelo.
- Dados referente ao modelo que será analisado, tais como profundidade ou peso específico do poço que será escavado.
- Informações referentes à análise que vai ser realizada, tais como o tipo de análise que será realizada (neste exemplo, elasto-plástica) ou o tipo de modelagem utilizada (estado plano de tensão, estado plano de deformação ou modelo assimétrico).

A definição destes atributos deve ser realizada no arquivo de configuração como foi mostrado na seção 3.5.2.1. Com os atributos da simulação já configurados para este tipo análise, inicia-se o processo de simulação propriamente dito com a fase de definição do modelo geométrico e da associação de atributos às entidades geométricas deste modelo. A Figura 4.3 ilustra a definição da geometria deste modelo. Já a Figura 4.4 mostra o diálogo utilizado para a captura dos dados referentes ao material que será associado às regiões do modelo.

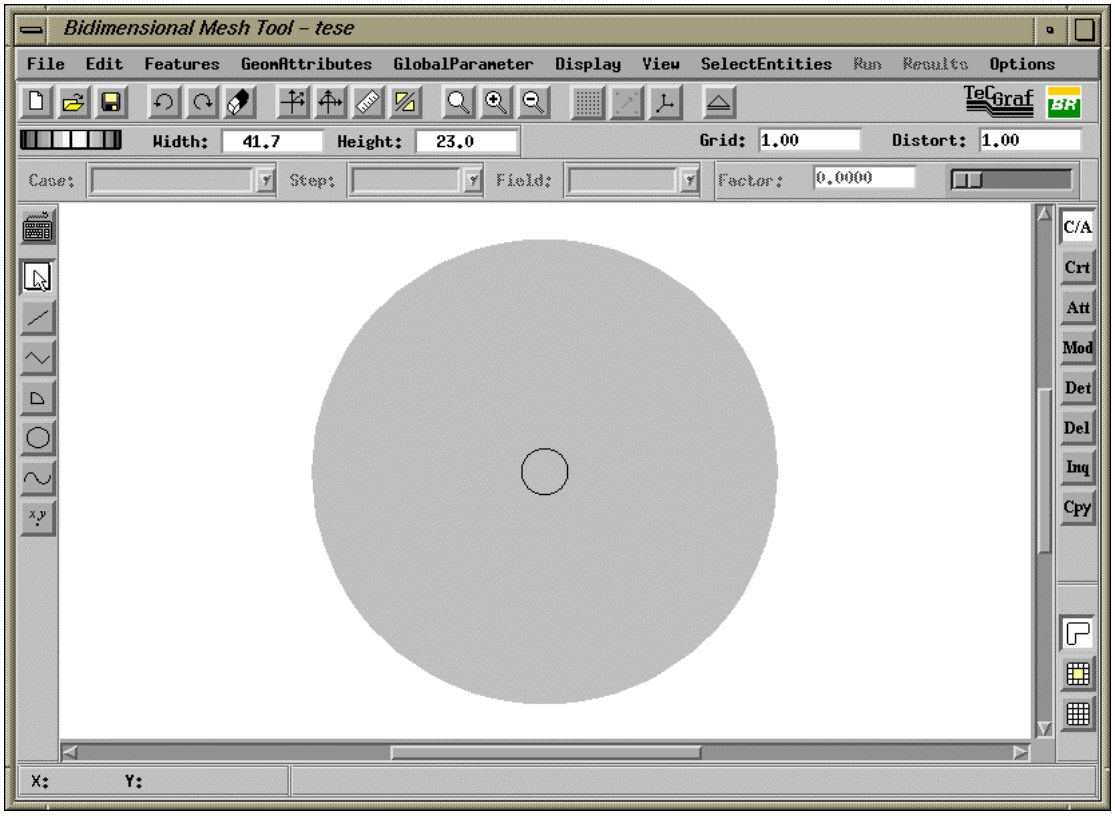

Figura 4.3 - Geração da geometria.

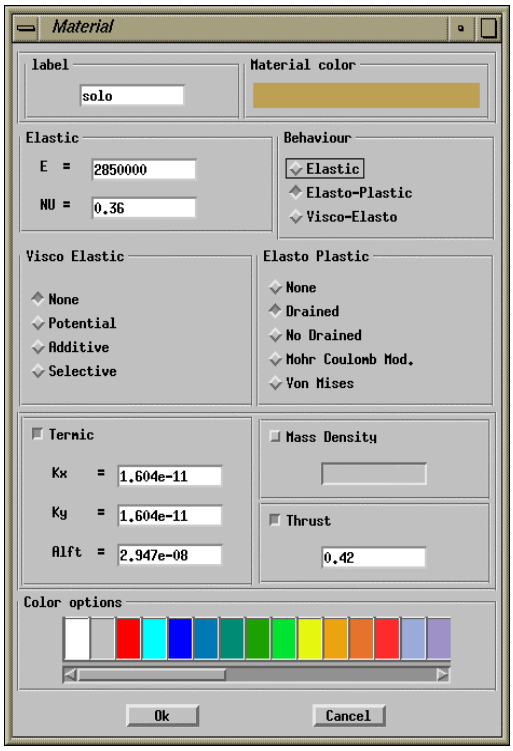

Figura 4.4 - Diálogo para a captura de dados referentes ao material elasto-plástico.

Definido este modelo, o processo continua com a geração automática da malha de elementos finitos a partir da geometria do modelo. Neste instante, os atributos associados ao modelo geométrico são mapeados para a malha de elementos finitos. A Figura 4.5 mostra a malha gerada.

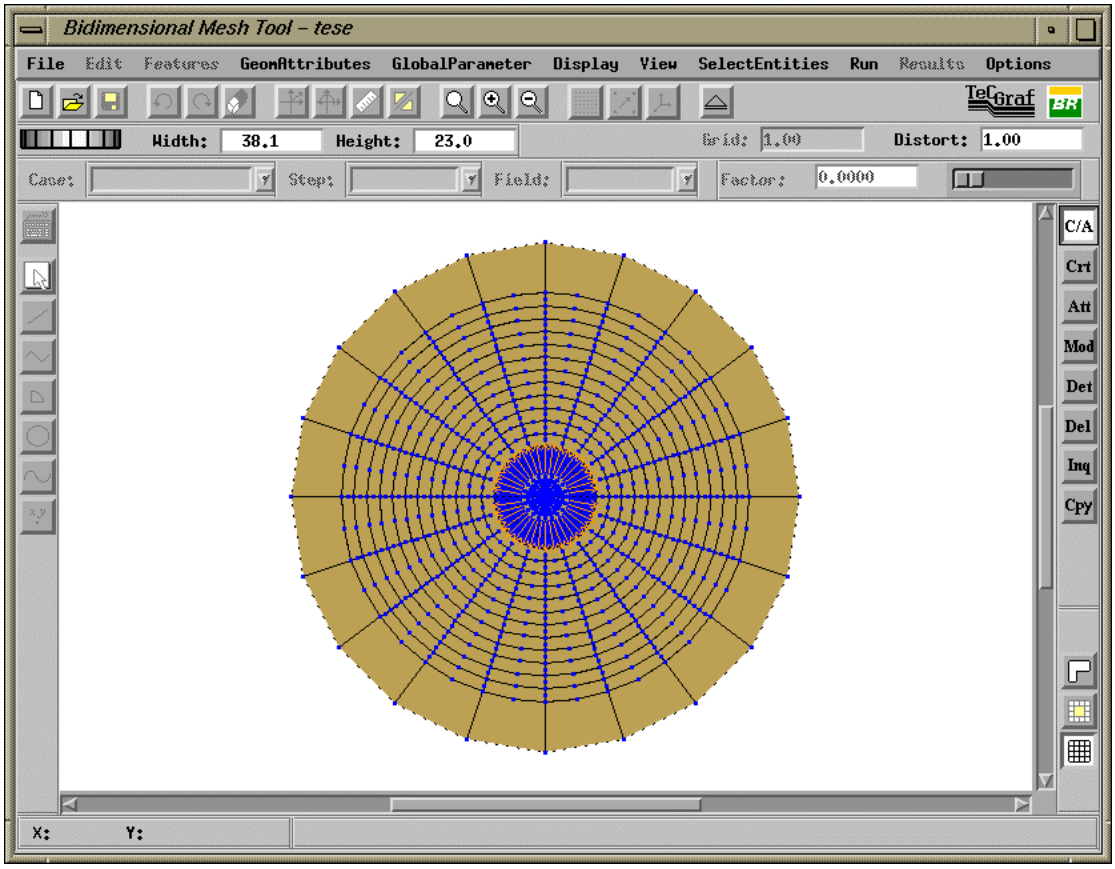

Figura 4.5 - Malha de elementos finitos.

Após a geração da malha de elementos finitos realiza-se, então, a análise numérica do problema proposto. Esta análise é realizada pelo programa AEEPECD, responsável por análises numéricas em geotecnia de modelos elasto-plásticos.

Os resultados obtidos a partir da análise numérica podem ser visualizados neste mesmo ambiente de simulação. A Figura 4.6 mostra os resultados referentes à distribuição do coeficiente de plastificação *Ratio* (relação entre a tensão atuante e a tensão de ruptura) decorrente da perfuração do poço. A Figura 4.7 mostra um gráfico que indica a

relação entre o coeficiente de plastificação e a distância ao longo da seção crítica do modelo.

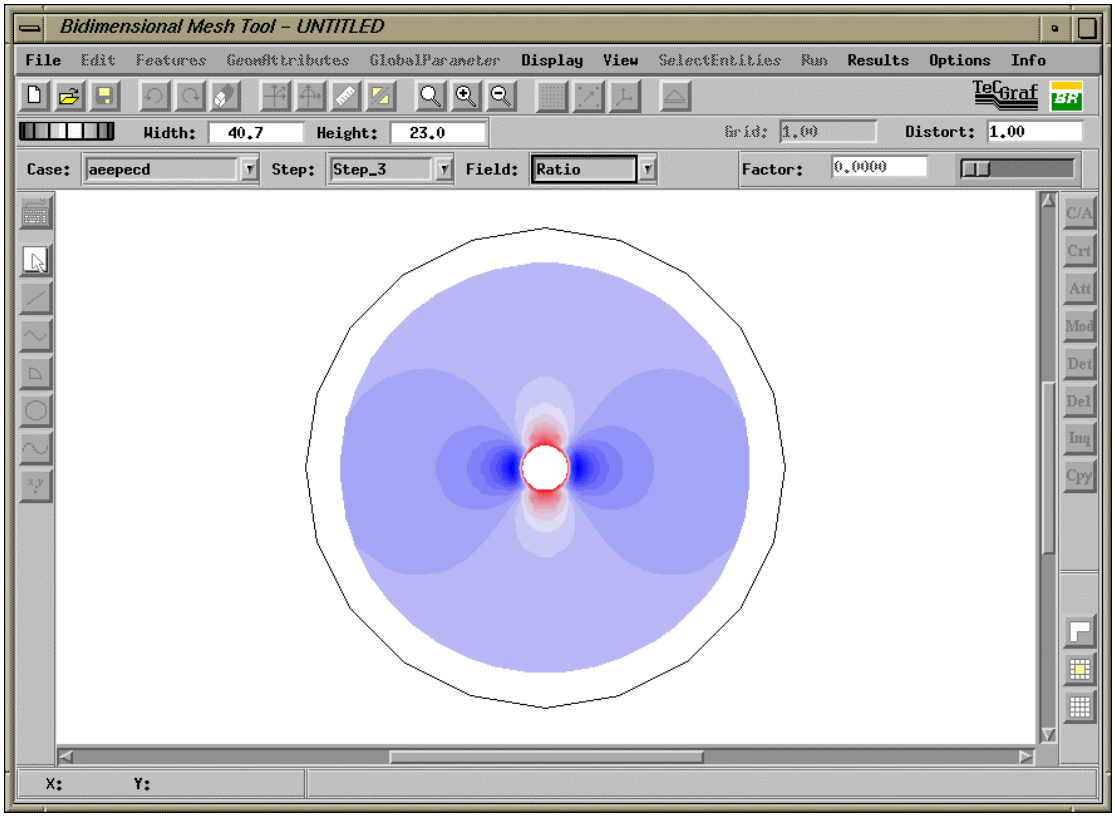

Figura 4.6 - Visualização dos resultados.

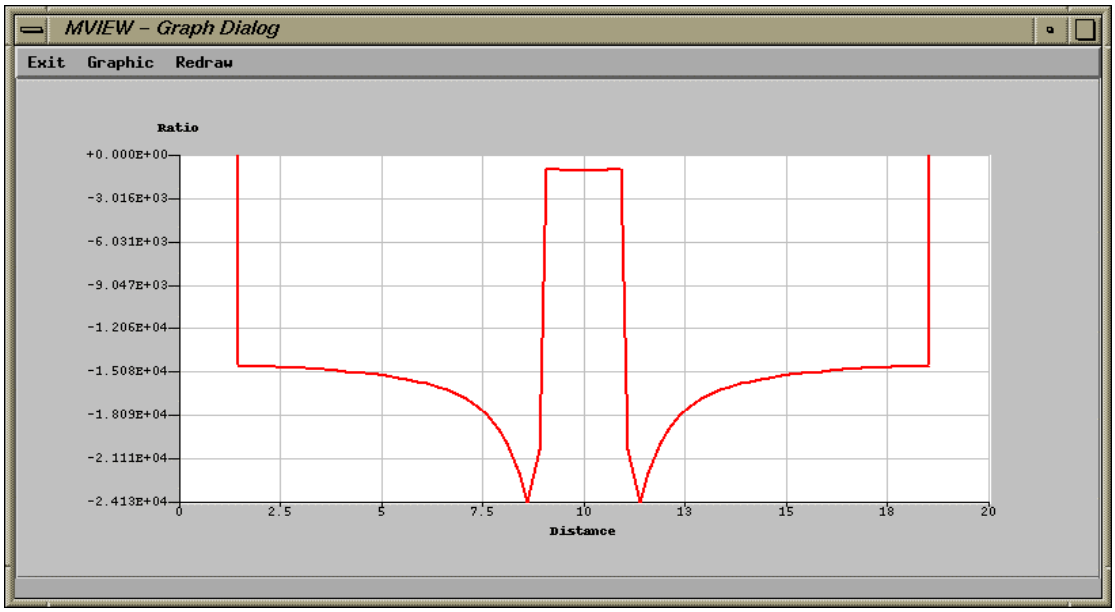

Figura 4.7 - Relação entre o coeficiente de plastificação e a distância ao longo da seção crítica do modelo.

Observa-se na Figura 4.7 que a região do modelo que foi escavada (poço) não é mostrada no desenho do modelo. Isso só foi possível porque as informações referentes ao atributo de escavação (*Rezone*), definido na etapa de modelagem geométrica, pode ser consultado neste módulo de visualização. Isso possibilitou ao MView controlar quais os elementos e os nós da malha de elementos finitos que devem ser desenhados, permitindo-o excluir estas entidades escavadas durante o processo de análise da lista de entidades que devem ser desenhadas pela aplicação.

5. Conclusão

Este trabalho dá continuidade ao desenvolvimento de uma metodologia para a extensão e configuração de aplicações gráfico-interativas que fazem uma simulação baseada no método dos elementos finitos. Esta metodologia foi implementada no desenvolvimento do Sistema Integrado de Geotecnia para Múltiplas Análises (SIGMA), utilizado para simulações numéricas de problemas bidimensionais em geotecnia pelo MEF. O SIGMA foi desenvolvido a partir da incorporação do sistema para gerenciamento de atributos ESAM, aos módulos MTool, responsável pela definição da geometria do modelo e geração da malha de elementos finitos, e MView, responsável pela visualização dos resultados gerados a partir de uma análise numérica, tornando-os extensíveis e configuráveis. Este sistema integra, em um único ambiente, os módulos MTool e MView com os programas de análise numérica para problemas de geotecnia AEEPECD e ANVEC. Os atributos implementados nestes programas de análise foram definidos em um arquivo de configuração a partir da criação de classes e métodos referentes a esses atributos.

A comunicação entre o pré-processador MTool e os programas análise do SIGMA é realizada através de arquivo de dados, onde as rotinas responsáveis pela geração deste arquivo são implementadas pelo próprio usuário que configura os atributos da simulação. Com isso, um novo atributo incorporado ao programa de análise pode ser facilmente incluído nos módulos MTool e MView, a partir da definição destes atributos no arquivo de configuração e da sua implementação na rotina que gera os arquivos de dados lidos pelos programas de análise. Isso tudo é feito sem necessidade de recompilação do sistema, visto que se utiliza a linguagem de extensão Lua.

No SIGMA, a comunicação entre os módulos MTool e MView é realizada através de um arquivo Lua gerado pelo ESAM. Em sua arquitetura original o sistema ESAM adotava uma modelagem baseada em geometria. Portanto, os arquivos gerados pelo

ESAM eram compostos apenas pelos atributos e pelas entidades geométricas aos quais os atributos foram aplicados. Entretanto, o MView não possui informação explícita sobre as entidades geométricas do modelo, isto é, no MView os atributos devem ser associados aos elementos e nós da malha de elementos finitos. Para resolver estes problemas, optou-se por expandir a arquitetura original do ESAM, permitindo-o suportar uma modelagem baseada na malha. A nova arquitetura possibilita aos pré-processadores, normalmente com recursos de modelagem geométrica como o MTool, gerar arquivos, com os atributos e as entidades da malha ao qual esses atributos estão associados, que possam ser interpretados pelos pós-processadores.

A incorporação do ESAM aos módulos MTool e MView possibilitou, ainda, a simplificação e diminuição dos códigos computacionais referentes a esses módulos, facilitando o seu entendimento, visto que o gerenciamento de atributos que era realizado pelos próprios módulos passou a ser inteiramente realizado pelo sistema ESAM.

As fases envolvidas em um processo de simulação em mecânica computacional que utilizam o MEF (definição da geometria, especificação dos atributos, geração da malha de elementos finitos, análise numérica e visualização dos resultados) foram ilustradas em um exemplo de aplicação do sistema proposto. Este exemplo trata da escavação de um poço no espaço bidimensional, considerando também a associação de gradiente de poropressão ao problema que foi analisado. Os atributos associados às entidades geométricas do modelo foram visualizados no módulo MView com os resultados obtidos a partir da análise numérica realizada pelo programa AEEPECD.

O SIGMA representa um avanço alcançado no desenvolvimento de um sistema integrado para a automação real do processo de simulação em mecânica computacional. Porém, a comunicação entre os módulos de pré- e pós-processamento e o programa de análise ainda é realizada através de arquivos de dados. A automação total do processo poderá ser alcançada com a incorporação do ESAM aos programas de análise, permitindo que todos os atributos definidos no MTool também possam ser representados nestes programas sem ser via arquivo de dados. Atualmente, pode-se criar um atributo no pré-processador MTool, mesmo que ele não esteja inserido nos

programas de análise. Entretando, este atributo não vai produzir nenhum efeito dentro do processo de simulação.

5.1 Sugestões para Trabalhos Futuros

A expansão do sistema ESAM apresentado nesta dissertação apenas considerou os aspectos referentes a arquitetura deste sistema. Alguns detalhes referentes às funcionalidades do sistema, tais como um tratamento formal sobre o *feedback* (exibição gráfica) do atributo ou definição de uma taxonomia de classes de atributos que englobem a maior parte dos atributos utilizados em mecânica computacional, não foram abordadas neste trabalho. Neste contexto, algumas idéias estão mais amadurecidas e podem ser implementadas em trabalhos futuros. Outras sugestões para a continuação deste trabalho são descritas a seguir.

Como foi citado no corpo deste trabalho, os métodos da classe *Geometry* que representa uma abstração das entidades do modelo, possuem uma implementação única para todas as entidades definidas na aplicação, ou seja, o tratamento dado a uma aresta é o mesmo dado a um vértice do modelo, por exemplo. Subclasses desta classe são criadas automaticamente pelo ESAM a partir dos nomes das entidades correspondentes (representam, para o ESAM, os nomes da subclasses da classe *Geometry*), registrada pela aplicação. O sistema ESAM permite que esses métodos sejam redefinidos pelo usuário configurador da aplicação com o objetivo de proporcionar um controle específico sobre cada tipo de entidade da aplicação. A arquitetura do ESAM, no entanto, propõe que o usuário responsável pela configuração da aplicação não deve ter acesso a sua estrutura de dados. Para resolver este problema, sugere-se que seja implementado um mecanismo no sistema ESAM que possibilite ao usuário configurador acesso aos nomes da entidades definidas na aplicação, sem que este usuário tenha conhecimento do código computacional que implementa a aplicação. Este mecanismo pode ser baseado em um processo que forneça, sempre que uma aplicação for chamada pela primeira vez, os nomes das entidades geométricas disponíveis na aplicação. Estes nomes devem ser gravados em arquivos que possam ser

acessados pelo usuário. Assim, este usuário vai ter conhecimento sobre os nomes das entidades e pode, então, redefinir os métodos referentes as subclasses da classe *Geometry* correspondentes as entidades geométricas da aplicação.

Geralmente, os modelos criados pelo SIGMA são muitos grandes, principalmente quando existem um grande número de informações sobre os atributos do modelo. Estas informações são armazenadas em arquivos de dados e o seu acesso, normalmente, é bastante lento. É desejável, então, que estes dados possam ser armazenados em bancos de dados, onde a sua recuperação, pela aplicação, seja realizada com acesso direto ao banco de dados. Esse acesso é bem mais rápido do que a leitura e interpretação de arquivos de dados. Deve-se, então, implementar mecanismos no ESAM que possibilitem o armazenamento, em bancos de dados, das informações referentes aos dados dos modelos utilizados.

O pré-processador MTool possui uma estrutura de dados que armazena todas as operações realizadas sobre as entidades geométricas do modelo. Existem mecanismos de *undo* e *redo*, implementados no MTool, que permitem desfazer as operações realizadas sobre o modelo ou recuperar a última operação desfeita pelo mecanismo de *undo*. No entanto, as informações referentes aos atributos estão definidas pelo ESAM, que não suporta estes mecanismos de *undo* e *redo*. Por exemplo, se uma operação de *undo* for efetuada sobre o modelo, recuperando uma entidade anteriormente destruída, os atributos associados a esta entidade não serão recuperados. Sugere-se, então, implementar no sistema ESAM mecanismos para operações de *undo* e *redo*. Isso pode ser feito, por exemplo, explorando o armazenamento dos atributos no formato de um banco de dados.

Uma outra sugestão refere-se a incorporação do sistema ESAM a módulos de pré- e pós-processamento de modelos tridimensionais, possibilitando o desenvolvimento de sistemas integrados configuráveis para simulações numéricas desses modelos.

6. Referências Bibliográficas

[ANSYS 1997] ANSYS 5.4, "*Complete User's Manual Set*", 1997.

[Autodesk 1995] Autodesk, "*Autocad User's Guide*", Autodesk Inc., USA, 1995.

[Bathe 1982] Bathe, K. J., "*Finite Element Procedures in Engineering Analysis*", Pretice Hill, 1982.

[Carvalho 1995] Carvalho, M. T., M., "Uma estrátegia para o Desenvolvimento de Aplicações Configuráveis em Mecânica Computacional", Tese de Doutorado, PUC-Rio, Departamento de Engenharia Civil, 1995.

[Celes 1995, Celes *et al*. 1995] Celes Filho, W., Figueiredo L. H., Gattass, M., "EDG: Uma Ferramenta para Criação de Interfaces Gráficas Interativas", artigo submetido ao VIII SIBGRAPI, 1995.

[Chambers 1997] Chambers, C., "*The Cecil Language Specification and Rationale - Version 2.1*", Department of Computer Science and Engineering, University of Washington, USA, 1997.

[Costa 1984] Costa, A. M., "Uma Aplicação de Métodos Computacionais e Princípio de Mecânica das Rochas no Projeto e Análise de Escavações Destinadas a Mineração Subterrânea", Tese de Doutorado - COPPE - UFRJ, 1984.

[Courtney 1995] Courtney, A., "*Phantom - Language Reference Manual*", Department of Computer Science, Trinity College Dublin, Ireland, 1995.

[Guimarães 1992] Guimarães, L. G. S., "Disciplina de Orientação a Objetos para Análise e Pós-processamento Bidimensional de Elementos Finitos", Dissertação de Mestrado, PUC-Rio, Departamento de Engenharia Civil, 1992.

[Ierusalimschy *et al*. 1994] Ierusalimschy, R., Figueiredo, L. H., "*Reference Manual of the Programming Language Lua*", Monografias em Ciências da Computação 4/94, Departamento de Informática, PUC-Rio, 1994.

[Levy 1993] Levy, C. H., "IUP/LED: uma Ferramenta Portátil de Interface com o Usuário", Dissertação de mestrado, PUC-Rio, Departamento de Informática, 1993.

[Mantyla 1988] Mantyla M., "*An Introdution to Solid Modeling Computer*", Science Press, 1998.

[Martha *et al.* 1993] Martha, L.F., Carvalho, P.C., Celes Filho, W. e Ferraz, M., "Gerenciamento de Subdivisões do Plano Usando HED", Caderno de Ferramentas do VII Simpósio Brasileiro de Engenharia de Software, PUC-Rio/SBC, 51-52, 1993.

[Martha *et al.* 1996] Martha, L. F., Menezes, I. F. M., Lages, E. N., Parente Jr., E. e Pitangueira, R. L.S., "*An OOP Class Organization for Materially Nonlinear Finite Element Analysis*", XVII CILAMCE, Pádova, Itália, 229-232, 1996.

[Mello 1994] Mello. L. F. N., "Manuais Provisórios dos Módulos Independentes AEEPECD, ANLEET, ANLVPEC, ANVEC, ANVECT, AXINON e DINEXP3D", 1994.

[MSC/Patran 1996] "*MSC/Patran - Release Notes - Version 5.0*", *The MacNeal-Schwendler Corporation*, 1996.

[Ousterhout 1994] Ousterhout, J. K., "*Tcl and Tk ToolKit*", Addison-Wesley, 1994.

[PCL 1990] *"Patran Command Language (PCL) Guide - Release 2.5"*, *PDA Engineering PATRAN Division*, 1990.

[Pohl 1993] Pohl, I., "*C++ for C Programmers*", Benjamin/Cummings Company, 1993

[Rossum 1993] Rossum, G. V., "*An introduction to Python for UNIX/C programmers*", *Dutch UNIX users group*, 1993.

[Rossum 1997] Rossum, G. V., "*Python Reference Manual*", Corporation for National Research Initiatives, USA, 1997.

[Rumbaugh 1991] Rumbaugh J., "Object-Oriented Modeling and Design", Prentice-Hall, Englewood Cliffs, 1991.

[Sheppard 1988] Sheppard, M. S., "*The Specification of Phisical Attribute Information for Engineering Analysis*", Engineering with Computers 4 (1988) 145-155.

[Sheppard *et al.* 1988] Sheppard, M. S., Finnigan, P. M., "Integration of Geometric Modeling and Advanced Finite Element Preprocessing", Finite Elements in Analysis and Design 4 (1988) 147-162.

[Silveira 1995] Silveira, E. S. S., "Um Sistema Configurável para Simulação Adaptativa Bidimensional de Mecânica Computacional", Tese de Mestrado, PUC-Rio, Departamento de Engenharia Civil, 1995.

[TECG1 1992] Grupo de Tecnologia em Computação Gráfica, "*Bidimensional Mesh Tool* - Manual do Usuário", PUC-Rio, 1992.

[TECG2 1992] Grupo de Tecnologia em Computação Gráfica, "*Bidimensional Mesh View* - Manual do Usuário", PUC-Rio, 1992.

[TECG3 1992] Grupo de Tecnologia em Computação Gráfica, "*Neutral File Format* - Versão 11.0", PUC-Rio, 1992.

[TECG4 1995] Grupo de Tecnologia em Computação Gráfica, "Uma Interface Lua para o Sistema IUP - Manual do Usuário - versão 1.0", PUC-Rio, 1995.

[TECG5 1998] Grupo de Tecnologia em Computação Gráfica, "Sistema Integrado de Geotecnia para Múltiplas Análises - Manual do Usuário - versão 1.0", PUC-Rio, 1998.

[Wong 1994] Wong, V. S., "*Qualification and Management of Analysis Attributes with Application to Multi-Procedural Analysis for Multichip Modules*", *Master Thesis*, Rensselaer Polytecnic Institute, Troy, New York, 1994.