4
Estudo de Caso

Neste capítulo será apresentado um estudo de caso, com a implementação de um sistema para a geração e narração de estórias. O maior objetivo é ajudar no entendimento do problema visto no capítulo anterior e discutir algumas soluções, estando fora do escopo deste trabalho apresentar todas as abordagens possíveis ou apontar qual delas é a melhor.

A abordagem aqui proposta leva em consideração os estudos em literatura e narratologia, além dos principais guias de escritores disponíveis que foram estudados em detalhe no capítulo 2.

4.1
Descrição Geral da Arquitetura

A arquitetura do sistema é dividida em três módulos principais, cada um tendo a responsabilidade de resolver um dos subproblemas que foram apresentados no capítulo anterior. Todos os três módulos são independentes, podendo portanto ser utilizados separadamente. Na figura 4.1 é apresentado um diagrama geral da arquitetura.

O primeiro módulo é responsável por resolver o problema da geração das estórias e será apresentado com mais detalhes na seção 4.2. Apenas um gerador de estórias é usado. Ele deve ter acesso a uma base de conhecimento e receber os fatos que deverão ser verdadeiros no início da estória. Na saída do gerador, deve ser apresentada a descrição da estória gerada.

A geração da fábula é feita por um planejador baseado em redes de tarefas hierárquicas. Nesse modelo, cada tarefa executada pode ser dividida em tarefas menores. Dessa forma, a descrição final da estória contém não apenas uma seqüência de ações, mas também a hierarquia de tarefas envolvida. Os dados dessa hierarquia podem ser particularmente úteis na etapa de narração.

É proposto o uso dos métodos e técnicas de criação literária encontrados no capítulo 2. Com isso procura-se tornar o processo de geração mais parecido com o

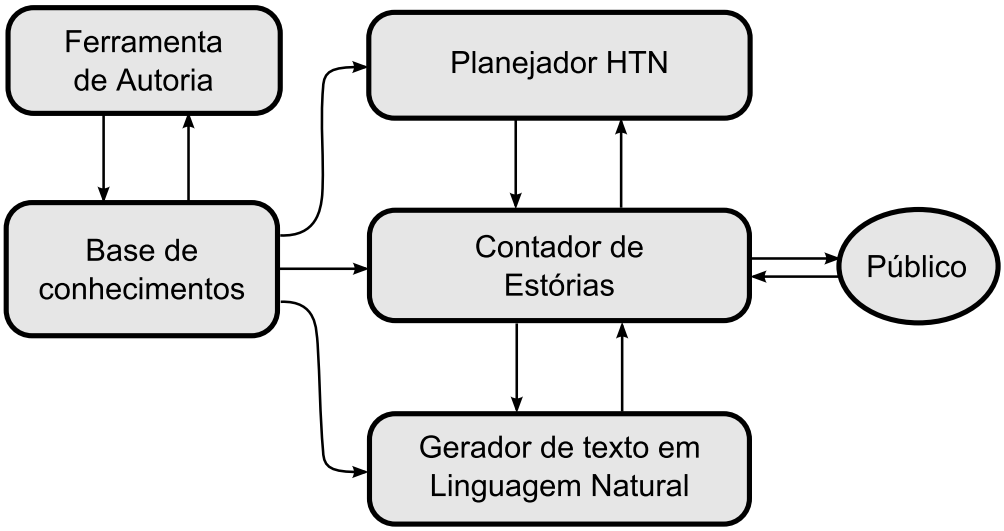

Figura 4.1: Arquitetura utilizada para o estudo de caso

tipo de criação feita por autores humanos, dado que essas técnicas são muito usadas na prática por escritores e roteiristas.

O segundo módulo consiste em um contador de estórias, que será mais bem explicado na seção 4.3. A narração da estória se dará através da geração de texto em linguagem natural. Foi implementada uma ferramenta capaz de transformar a descrição de uma fábula em um texto em linguagem natural, suportando os idiomas inglês e português. Essa ferramenta tira proveito da estrutura hierárquica das tarefas para conseguir gerar textos mais fáceis de serem lidos e entendidos.

O último módulo é uma ferramenta de autoria, construída para que o usuário consiga facilmente registrar as informações de que o computador precisa para a geração e narração das estórias.

Essa ferramenta também é capaz de se comunicar com o gerador e o contador de estórias; assim o usuário pode testar de forma rápida e eficaz a base de conhecimento que estiver sendo editada.

Sempre que possível os dados são descritos através de documentos baseados na linguagem XML. Toda a comunicação entre o gerador de estórias, o contador de estórias e a ferramenta de autoria é feita através desses arquivos em XML, ou através de uma biblioteca que também é capaz de ler e gravar estes arquivos. O formato XML foi escolhido por uma série de motivos, entre os quais podemos citar:

- É um padrão aberto, adotado internacionalmente;
- É independente de plataforma, não se limitando a alguma implementação específica, dispondo de um grande número de ferramentas para leitura, escrita e validação;
- Possui uma estrutura hierárquica que se encaixa bem no tipo de informação necessária para este estudo de caso.

Mais detalhes sobre a base de conhecimento e a ferramenta de autoria podem ser encontrados na seção 4.4.

A implementação do protótipo foi feita através da linguagem *Java*, escolhida por ser uma linguagem de alto nível que possui uma biblioteca padrão poderosa, com facilidades para a manipulação de arquivos em XML e a elaboração de interfaces gráficas com o usuário.

Foram utilizadas as bibliotecas externas: *JUnit* e *Log4J*, que auxiliaram, respectivamente, na elaboração de testes unitários e na exibição de mensagens de *log*. O ambiente de desenvolvimento utilizado foi o *Eclipse* e a ferramenta *Ant* foi usada para ajudar no processo de construção do software.

Para futuras implementações, pode ser interessante a utilização de uma linguagem de mais baixo nível, como *C* ou *Prolog*, na implementação dos módulos mais críticos em relação ao desempenho. No entanto, observa-se que o tempo necessário para a geração de uma estória depende muito mais da técnica utilizada do que da linguagem escolhida.

4.2
Geração

Nesta seção serão apresentadas algumas das principais abordagens empregadas na geração de estórias e será discutido o problema do planejamento automatizado e sua relação com a geração de estórias. Em seguida, será apresentada uma abordagem fundamentada no planejamento em redes de tarefas hierárquicas.

Alguns dos principais métodos vistos na seção 2.3 serão combinados a este algoritmo, definindo um processo de geração que se assemelhe aos processos utilizados por boa parte dos autores humanos.

4.2.1
Estórias Pré-Definidas e Estórias Geradas Dinamicamente

A abordagem atualmente mais utilizada para a criação de estórias em meios computacionais envolve o uso de *scripts*, que são roteiros previamente estabelecidos por autores humanos. Não necessita de um gerador automático de estórias, pois todas as possíveis variações da estória já estão especificadas de forma explícita pelo autor antes de a fase de narração começar.

Este processo, porém, é demorado e acarreta altos custos de produção. Além disso, o número de estórias possíveis é limitado e o grau de interação acaba sendo prejudicado, frustrando por vezes as expectativas do público. Essa abordagem também dificulta a adaptação das estórias de acordo com as preferências de cada usuário. Por outro lado, apesar dos problemas, permite um controle maior sobre o resultado final da estória e é relativamente simples de ser implementada.

Principalmente por estes motivos, a grande maioria dos jogos criados atualmente ainda se serve de *scripts* para a geração de estórias.

Uma alternativa aos roteiros pré-definidos é o uso de estórias geradas dinamicamente através de um gerador de estórias. Várias técnicas podem ser usadas com esse intuito, sendo que a principal delas se dá através da transformação do problema original em um problema de planejamento, como será explicado na próxima seção.

4.2.2 Planejamento de Estórias

O *planejamento* é um campo de pesquisa, dentro da área de Inteligência Artificial, que estuda a construção de programas capazes de resolver problemas através da geração e execução de planos. Um *plano* pode ser representado por uma seqüência de eventos, com dependências temporais entre si, que são executados por agentes (Hendler, Tate e Drummond 1990).

A representação de um plano tem muitas semelhanças com a de uma fábula. Como visto no capítulo 2, Shklovsky (1991) definiu uma fábula como sendo o conjunto de todos os eventos que ocorrem em uma estória, seguindo uma ordem cronológica. Em ambos os casos os eventos estão relacionados com as ações que podem alterar o estado do mundo.

A semelhança entre fábulas e planos possibilita a utilização destes na representação de enredos literários, uma prática desde há muito tempo introduzida por pesquisadores de Inteligência Artificial (Schank e Abelson 1975) e adotada em um grande número de SGE's (Meehan 1981, Dehn 1981, Lebowitz 1985, Riedl 2004, Pozzer 2005).

Além da semelhança entre planos e fábulas, Young (1999) aponta a relação de causalidade entre os passos dos planos como sendo um dos motivos para que o planejamento seja um bom modelo para a geração de estórias. Riedl e Young (2003) argumentam ainda que a decomposição dos planos em operações discretas facilita o trabalho do contador de estórias, que pode executar cada operação separadamente em um mundo virtual.

Barros e Musse (2007) levantaram duas questões importantes em relação ao uso de planejamento em SGE's. A primeira diz respeito aos problemas causados pela interação com o usuário. Isso porque alguma ação do usuário pode alterar o estado do mundo no exato momento em que um plano é executado, causando problemas que precisam ser resolvidos de alguma forma (Barros e Musse 2007).

A segunda questão é que nem todo plano que leva a um estado final desejado resulta em uma estória de qualidade. As métricas normalmente usadas para avaliar a

qualidade dos planos (como, por exemplo, o tamanho de cada plano) não fornecem uma medição confiável para a qualidade das estórias (Barros e Musse 2007).

Uma das métricas sugeridas procura maximizar o número de tarefas paralelas, independente do número total de eventos do plano encontrado. Dessa forma a estória tende a se tornar menos previsível e é esperado que o interesse do público seja maior (Barros e Musse 2007).

Ainda assim, não há consenso na literatura sobre como melhorar a qualidade das estórias geradas. O principal motivo é a falta de uma maneira objetiva de se avaliar as estórias.

A implementação de um programa capaz de analisar e "entender" uma estória é um problema em aberto, cuja solução ainda é desconhecida com o uso da tecnologia atual (Mueller 2003). Avaliar a qualidade dessas estórias é uma tarefa talvez ainda mais difícil de ser realizada.

4.2.2.1 Planejamento Automatizado

Dentro do contexto de planejamento, um problema normalmente é caracterizado pela descrição de um *estado inicial* e de um *estado objetivo*. O primeiro descreve quais fatos são verdadeiros no mundo no momento imediatamente anterior à execução do plano, enquanto o estado objetivo descreve os fatos que devem ser verdadeiros no final da execução. Os fatos devem ser representados por um conjunto de fórmulas bem-definidas a respeito do mundo (Fikes e Nilsson 1971).

Fikes e Nilsson (1971) criaram uma terminologia para a descrição dos eventos contidos nos planos, onde cada evento é representado por um operador cuja especificação envolve:

- Uma *lista de pré-condições*, indicando os fatos que precisam ser verdadeiros para que o operador possa ser executado;
- Uma *lista de remoção*, especificando os fatos que deixarão de ser verdadeiros depois da execução do operador;
- Uma *lista de adição*, especificando os fatos que passarão a se tornar verdadeiros depois da execução do operador.

Essa terminologia, também conhecida como STRIPS, é amplamente utilizada entre planejadores automatizados, e recebeu esse nome (acrônimo de "Stanford Research Institute Problem Solver") devido ao projeto através do qual foi implementado pela primeira vez.

As listas de remoção e adição de um operador também são chamadas de pós-condições. Assume-se que os fatos não mencionados pelas pós-condições devem permanecer inalterados após a execução do evento.

Dessa forma, a maior parte dos sistemas de planejamento recebe como entrada um problema (estado inicial e objetivo) e uma lista de operadores (com suas pré e pós-condições). Há planejadores, porém, que recebem outros dados, principalmente para ajudar na especificação de domínios de aplicação.

4.2.2.2 Complexidade do planejamento STRIPS

A complexidade dos planejadores que seguem o modelo STRIPS depende de restrições que podem ser impostas às fórmulas e aos operadores utilizados na representação do problema. A complexidade varia também dependendo da obrigatoriedade em se achar uma solução ótima (Bylander 1994).

Descobrir se um problema de planejamento STRIPS possui uma solução é, em geral, um problema *PSPACE*-completo. Somente é possível transformá-lo em um problema da classe *P* ou *NP*-completo sob fortes restrições impostas a operadores e fórmulas. Para uma descrição mais detalhada da complexidade de tais planejadores é recomendada a leitura do trabalho de Bylander (1994).

A classe *PSPACE* contém os problemas de decisão que podem ser resolvidos por uma máquina de Turing, determinística ou não, usando uma quantidade polinomial de memória, sem retrições quanto ao tempo de execução. Um problema pertence à classe *PSPACE*-completo caso pertença à classe *PSPACE* e caso todos os problemas da classe *PSPACE* possam ser reduzidos a ele em tempo polinomial (Jones 1997).

Suspeita-se que a classe *PSPACE*-completo não seja igual às classes de complexidade *P* e *NP*, mas ainda não foi encontrada uma prova que justifique tal afirmação (Jones 1997). De qualquer forma, todos os algoritmos conhecidos para a resolução desses problemas precisam de um tempo exponencial para serem executados. Ou seja, não são algoritmos eficientes e tendem a ficar intratáveis dependendo do tamanho do problema.

Para o planejamento de estórias ser possível em um tempo razoável, há duas alternativas. Uma é diminuir o tamanho do problema, a outra é a inclusão de mais restrições ao planejamento.

A primeira opção é problemática, pois a qualidade da estória gerada depende muito da riqueza da base de conhecimento que define o problema. Uma base de conhecimento reduzida tende a gerar estórias simples demais para a expectativa dos usuários em boa parte das aplicações, mas pode ser uma alternativa para aplicações que efetivamente não precisem de uma base de conhecimento muito detalhada.

A outra opção consiste em criar mais restrições ao planejador, facilitando a resolução do problema. Essas restrições dependem do tipo de estória que deve ser gerada. Um exemplo desta abordagem será vista na seção 4.2.2.4.

4.2.2.3
Classificação dos Sistemas de Planejamento

Nau (2007) dividiu em três classes os sistemas de planejamento automatizado. Segundo ele, os planejadores podem ser de *domínio específico*, *independentes de domínio* ou de *domínio configurável*.

Os planejadores *independentes de domínio* são construídos visando um grande número de aplicações. Um planejador deste tipo recebe como entrada a descrição do problema e deve ser capaz de resolvê-lo independente de qual seja o domínio da aplicação, desde que seja possível descrever o problema segundo uma dada forma de representação (Hendler, Tate e Drummond 1990). O maior problema com este tipo de abordagem está relacionado à eficiência, que é sacrificada para tornar o sistema mais genérico (Nau 2007).

Segundo Nau (2007), apesar de serem chamados de independentes de domínio, a maior parte desses planejadores parte de algumas suposições restritivas, nem sempre verdadeiras, a respeito dos problemas. Por exemplo, muitas vezes parte-se da hipótese de que o sistema possui um número finito de estados e envolve um mundo totalmente observável, além de ser determinístico.

Os planejadores de *domínio específico*, ao contrário, são construídos para serem usados em um único tipo aplicação. A maior parte dos planejadores empregados em aplicações reais utiliza essa abordagem (Nau 2007, Hendler, Tate e Drummond 1990).

O maior problema com esses planejadores é que precisam ser criados (ou radicalmente refeitos) para cada tipo de aplicação. Isso pode não ser um grande problema para projetos de grande orçamento, como os *Veículos Exploradores de Marte* da NASA, que logram utilizar essa abordagem com sucesso (Estlin et al. 2003). Mas, para projetos de menor orçamento, é muito caro e arriscado refazer todo o código de planejamento a cada nova aplicação.

Já os planejadores de *domínio configurável* são semelhantes aos independentes de domínio, mas recebem o conhecimento específico do domínio em sua entrada, o que os torna mais flexíveis, ao mesmo tempo em que aproveitam o conhecimento específico do domínio da aplicação (Nau 2007).

Essas duas últimas classes de planejadores se utilizam do conhecimento de domínio específico para restringir o espaço de busca e, dessa forma, conseguir uma eficiência maior na execução.

Na próxima seção será vista uma abordagem para a construção de planejadores de domínio configurável, chamada de *Rede de Tarefas Hierárquicas* (*RTH*).

4.2.2.4
Rede de Tarefas Hierárquicas

O planejamento RTH é uma abordagem que gera planos a partir da decomposição de *tarefas*, utilizando-se do conhecimento de domínios específicos para aumentar sua eficiência.

Ao contrário do que acontece no planejamento STRIPS, a descrição de um problema não contém um estado objetivo. Ao invés disso, é dada uma lista de tarefas que precisam ser executadas. Na saída tenta-se apresentar um plano que consiga executar todas essas tarefas.

O planejamento é feito a partir da decomposição das tarefas em subtarefas cada vez menores, até que se chegue a tarefas primitivas, que possam ser executadas diretamente. As tarefas primitivas são definidas por *operadores*, semelhantes aos operadores STRIPS, definidos por suas pré- e pós-condições. As demais tarefas, por sua vez, devem ser decompostas em tarefas menores, primitivas ou não. Essa decomposição é efetuada através de *métodos* (Erol, Hendler e Nau 1994b).

Os *métodos* são responsáveis por descrever o domínio específico da aplicação e devem ser passados como parte da entrada do planejador. Cada método é representado por uma tupla <t, r>, onde t é uma tarefa e r é uma rede de tarefas, informando que a tarefa t pode ser realizada através da execução das tarefas pertencentes à rede de tarefas r (Erol, Hendler e Nau 1994b).

Uma *rede de tarefas* contém um conjunto parcialmente ordenado de tarefas a executar e uma lista de pré-condições que precisam ser satisfeitas antes da execução (Erol, Hendler e Nau 1994b).

Erol, Hendler e Nau (1994a) provaram que os planejadores RTH são mais expressivos que os planejadores STRIPS. Ou seja, qualquer problema descrito em uma linguagem STRIPS pode ser mapeado para um problema de planejamento RTH, mas o contrário não é verdadeiro. Entre outras coisas, faltariam aos planejadores STRIPS meios de declarar objetivos e restrições aos estados intermediários, mas está fora do escopo deste trabalho apresentar uma prova a essas afirmações. Para maiores detalhes, é sugerida a leitura do trabalho de Erol, Hendler e Nau (1994a).

Ao contrário do planejamento STRIPS, o planejamento RTH não é decidível, a não ser que se imponha certo número de restrições. Por exemplo, caso seja incluída a restrição de que as listas de tarefas sejam totalmente ordenadas, o problema passa a ser decidível (Erol, Hendler e Nau 1994a). A *decomposição ordenada de tarefas* é uma versão modificada do planejamento RTH, com ordenação total. Este tipo de planejamento, também é mais expressivo que o STRIPS e é igualmente decidível.

Segundo Nau et al. (2003), muitos problemas em domínios específicos podem ser resolvidos em tempo polinomial através da decomposição de tarefas, graças ao conhecimento específico de domínio utilizado.

Essa abordagem de planejamento pode de fato ser muito útil na geração de estórias, por uma série de motivos. Segundo Cavazza, Charles e Mead (2002), o planejamento RTH tende a contribuir mais para a fase de autoria de estórias do que o planejamento STRIPS, propiciando, entre outras coisas, meios para a autoria de variações nas estórias.

Além disso, os papéis dos personagens podem ser definidos através da associação de redes de tarefas distintas aos personagens (Cavazza, Charles e Mead 2002). Na etapa de autoria da estória podem ser criadas alternativas para a ação dos personagens dependendo do estado afetivo de cada um.

Outra vantagem do modelo de decomposição ordenada de tarefas é sua facilidade na decomposição de estórias em partes, como, por exemplo, a divisão da estória em três atos ou na seqüência de passos do monomito (v. 4.2.3.5) ou de algum outro modelo que vier a ser utilizado.

4.2.3
Métodos para Criação de Estórias

No capítulo 2.3 foram apresentados alguns métodos para a criação de estórias, com destaque para os trabalhos de Aristóteles (2004), Polti (1945), Field (1982), Campbell (1968) e Vogler (1998). Nesta seção será proposta uma abordagem para a utilização desses métodos na geração automática de estórias através de RTH.

4.2.3.1
Método Proposto

Na seção 2.3.3 foi descrito um método para a criação de estórias usando cartões de papel, cada um contendo a descrição de uma cena ou seqüência da estória. Uma vez que os cartões estejam preenchidos e as cenas definidas, estas são refinadas até se chegar a sua versão final (Field 1982, McKee 1997).

Essa técnica *top-down* é muito popular entre roteiristas de cinema e é análoga ao processo seguido por um planejador de RTH, onde a construção de uma estória é feita a partir de uma seqüência de tarefas que são subdivididas em tarefas cada vez menores e mais específicas.

Existem várias versões para este método. Syd Field (1982) sugere que o primeiro passo na geração de uma estória seja definir sobre o que a estória trata. Depois, os protagonistas devem ser criados para então dividir-se a estória em atos, cenas e ações.

A abordagem utilizada no presente estudo de caso se baseia fortemente nesse método. Inicialmente será definida a situação dramática da estória, em seguida serão definidos os principais personagens e seus arquétipos.

A estória será dividida em início, meio e fim, seguindo a estrutura apresentada por Aristóteles (2004). No início da estória, os principais personagens devem ser apresentados ao público. O meio e o fim da estória serão gerados levando em consideração um modelo simplificado da jornada do herói (Campbell 1968, Field 1982).

Os principais passos da jornada são definidos em um primeiro estágio para então as tarefas serem refinadas até que as ações primitivas sejam definidas.

4.2.3.2 Personagens

Segundo Aristóteles (2004), os dois principais elementos para a criação de estórias são: enredo e personagem. A maioria dos trabalhos em geração de estórias acaba privilegiando algum desses dois elementos. Não há um consenso sobre qual é o mais importante, mas com certeza ambos os elementos devem ser bem trabalhados para que a qualidade da narrativa seja satisfatória.

No atual estágio deste trabalho, o enredo acabou sendo privilegiado, mas ainda assim os personagens têm muita importância no processo de geração. Ao contrário de outros geradores de estória (Mateas e Stern 2003, Pozzer 2005), os personagens não são definidos antes da geração da estória. Ao invés disso, eles são criados de acordo com as necessidades durante a etapa de geração.

Cada personagem existente em uma estória deve possuir pelo menos cinco atributos, são eles:

- Papel na estória;
- Tipo de personagem;
- Gênero;
- Nome do personagem;
- Residência.

O *papel* é definido de acordo com o motivo pelo qual o personagem foi criado. Por exemplo, um personagem pode ter sido criado para atrapalhar o herói, tentando impedi-lo de atingir seu objetivo e, com isso, deixar a estória mais emocionante. Este conceito de papel está de certa forma relacionado aos *arquétipos* que acabam definindo a função de um personagem dentro de uma estória.

O *tipo* está relacionado à profissão ou ocupação do personagem, ou a algum estereótipo que o qualifique. Por exemplo, entre os tipos de personagens podemos ter: princesa, cavaleiro, plebeu, andarilho, aventureiro, monge, feiticeiro, ogro, mercenário, entre outros.

Certas ações só são permitidas a alguns tipos de personagens. Por exemplo, a um mercenário é permitido seqüestrar uma princesa, mas uma princesa não pode

sequestrar um mercenário. A escolha do tipo de personagem tem ligação com o seu papel dentro de uma estória. Neste estudo de caso não é permitido, entre outras restrições, que um ogro seja o herói da estória.

O *gênero* (masculino, feminino) deve ser considerado na atribuição de nomes aos personagens, e pode também influir nas pré-condições de alguns eventos – por exemplo, a execução do operador 'casar' pode ser permitida apenas entre personagens de sexos opostos.

A escolha do gênero pode ser influenciada também pelo tipo de personagem. Se, por exemplo, for criado um personagem do tipo 'padre' em uma sociedade tradicional, esse personagem deverá obrigatoriamente ser do sexo masculino.

O *nome* é retirado de uma lista de nomes possíveis. Estes nomes podem estar vinculados ao tipo de personagem e ao seu gênero, por exemplo, a lista de nomes possíveis para um cavaleiro da corte deve ser diferente da lista de nomes possíveis para um ogro.

A *localização inicial* de um personagem é definida através de uma lista, dependendo do tipo de personagem. Uma princesa, por exemplo, deve morar em um castelo, enquanto um ogro pode morar em uma caverna ou floresta.

A forma como os atributos dos personagens são utilizados varia dependendo do tipo de estória. Os cinco atributos definidos aqui representam uma simplificação da difícil tarefa de se construir personagens.

Fica como um trabalho futuro uma definição mais completa dos atributos necessários para a criação de personagens de estórias. O ideal seria construir um modelo capaz de simular o comportamento afetivo e que leve em consideração os arquétipos do inconsciente coletivo.

4.2.3.3 Situações Dramáticas

Como visto na seção 2.3.3, Polti (1945) reuniu algumas das principais situações dramáticas encontradas nas estórias da literatura mundial. Trabalhos como esse são muitas vezes usados para auxiliar escritores a iniciar suas estórias. Os sistemas apoiados por computador também podem se beneficiar dessa pesquisa e utilizar tais situações dramáticas como um impulso inicial na criação de uma estória.

No método proposto para este estudo de caso, a primeira tarefa que deve ser executada é justamente a definição da situação dramática.

A situação dramática é usada na etapa de geração para definir os principais personagens da estória e o papel de cada um deles. Além disso, as ações iniciais e o desfecho da estória devem depender da situação dramática escolhida.

Neste estudo foram definidas apenas duas situações dramáticas, que dizem respeito às situações de *resgate* e *vingança*. Para a situação de resgate, o aconteci-

mento provocador deverá ser um rapto e a estória deverá terminar com a libertação da vítima do rapto. No caso de uma estória de vingança, o acontecimento provocador deverá ser algum tipo de crime contra a vítima e no final o herói deverá derrotar o vilão como forma de vingança.

Em uma estória de resgate os principais personagens são: o sequestrador, a vítima e o herói. Em uma estória de vingança eles são: o vilão, o injustiçado e o justiceiro.

4.2.3.4
Apresentação dos Personagens

Aristóteles (2004) definiu a divisão das estórias em três partes: início, meio e fim. Uma divisão similar foi apresentada por Field (1982), que deu o nome de paradigma à sua estrutura de três atos.

Em ambas as divisões, o início ou primeiro ato tem como incumbência fazer a apresentação da estória. Nela deve ser mostrado ao público quem são os personagens principais e qual é a situação dramática da estória. Se o público não conhecer nada sobre os personagens dificilmente terá simpatia por algum deles e a experiência com a estória será prejudicada.

Há duas formas de proceder. A primeira é através de um narrador externo que apresenta cada um dos personagens, e a outra é exibindo ações dos próprios personagens. Esta última é a mais recomendada, pois, segundo Field (1982), as escolhas assumidas por um personagem é que definem de fato como ele é, e a partir da análise de suas atitudes é que o público sabe o que dele pode esperar.

Neste estudo, porém, será utilizada a primeira abordagem, que é mais simples de implementar e serve bem aos propósitos do trabalho. Dessa forma, sempre que um personagem é criado dentro da estória ele é apresentado ao público. A apresentação é bastante simples: basicamente são passadas ao público as informações sobre quem é o personagem, seu nome e tipo e onde mora.

4.2.3.5
Monomito

Uma vez definida a situação dramática, os protagonistas e respectivos arquétipos, o personagem principal deve seguir caminho em busca de seus objetivos.

Essa jornada, também chamada de monomito, é descrita em detalhes por Campbell (1968), com ênfase maior nas estórias de mitologia. Vogler (1998) adaptou e estendeu o modelo criado por Campbell (1968) para aplicá-lo na construção de roteiros de cinema. Maiores detalhes sobre o monomito podem ser encontrados na seção 2.3.3.

A aplicação desse método é muito popular entre escritores de obras de ficção e roteiristas de cinema. Contudo, não foi observado na literatura nenhum SGE que utilizasse o monomito para a criação das estórias.

Será proposto aqui um modelo simplificado em relação aos modelos de Campbell (1968) e Vogler (1998) para a geração de estórias. Baseia-se em apenas seis passos, descritos a seguir:

1. *Chamado à aventura*: O protagonista presencia ou fica sabendo do acontecimento provocador da estória. De algum modo ele resolve assumir a responsabilidade de resolver o impasse para restabelecer o equilíbrio no mundo da estória.

2. *Encontro com o Mentor*: O mentor treina o protagonista para enfrentar as adversidades e atingir o objetivo, passa informações importantes ao protagonista (local do esconderijo, ponto fraco do antagonista, etc.), presenteia o protagonista com item importante (arma, feitiço, transporte, dinheiro, etc.) e cria motivação para o protagonista continuar em sua jornada.

3. *Testes, aliados, inimigos*: O protagonista luta contra aliados do antagonista (inimigos) e enfrenta armadilhas criadas por ele. Os aliados do protagonista o auxiliam a atravessar os obstáculos que aparecerem no caminho.

4. *Aproximação à caverna mais profunda*: O protagonista se prepara para atingir seu objetivo final e chega ao local onde está seu maior inimigo.

5. *Climax*: Ponto alto da estória, quando o protagonista deve enfrentar o maior inimigo. No final do embate ele deve atingir seu objetivo.

6. *Retorno com o elixir*: O protagonista volta para casa são e salvo, com algum ganho em relação ao que tinha antes, e a estória atinge seu desfecho.

A tarefa principal de gerar estórias é dividida em duas subtarefas. A primeira corresponde à escolha da situação dramática e apresentação dos personagens. A segunda diz respeito à tarefa associada ao monomito, que possui seis subtarefas, cada uma delas correspondendo a um dos passos da versão simplificada descrita acima. Cada uma destas últimas pode ser executada de inúmeras formas, dependendo principalmente da situação dramática escolhida e do estado inicial fornecido.

4.2.3.6
Estado Inicial

O estado inicial de uma estória descreve os fatos que são verdadeiros antes de começar a execução de um plano. Isso não quer dizer que o início da estória tem que estar pré-definido, já que há operadores para a criação de personagens e estes não precisam estar especificados antes da geração.

O estado inicial pode ser utilizado para instanciar parâmetros na geração da estória. Por exemplo, é através do estado inicial que é passada a lista de possíveis nomes para os personagens e os tipos de residência para cada tipo de personagem. Através do estado inicial podem ainda ser definidas quais situações dramáticas são permitidas e também as restrições sobre como a estória deve ser mostrada.

4.3
Narração de Estórias

A narração é a fase onde o público entra em contato com os personagens e vivencia os principais acontecimentos de uma estória.

O contador é responsável por contar ao público uma estória já elaborada. Neste estudo, cada estória é representada por um documento em XML que deve ter informações suficientes para que o contador possa narrá-la ao público.

4.3.1
Meios de Comunicação para Narração de Estórias

Entre as formas de se narrar uma estória pode-se citar textos em prosa, poesias, filmes, jogos digitais, além, é claro, da forma oral. Esta dissertação se concentra na narração automática de estórias através de computadores.

Os meios nos quais se pode usar um computador para transmitir uma estória ao público também são diversos. Muitas vezes pode-se preferir a geração e exibição de um texto através de um console textual, em outras situações pode ser útil o uso de desenhos animados em duas dimensões.

Recentemente os jogos tridimensionais têm-se tornado um meio expressivo para narrativas, pois seus recursos gráficos realistas ajudam a estimular o sentimento de imersão nos jogadores. Há ainda experimentos na área de realidade aumentada, onde se misturam elementos tridimensinais com imagens reais capturadas por câmeras.

4.3.2
Geração de Texto em Linguagem Natural

Neste estudo optou-se pela narração de estórias através de textos. A Geração de Linguagem Natural (GLN) é mais um tópico de pesquisa dentro da área de

Inteligência Artificial. A utilização de linguagens naturais, tais como o português e o inglês, facilita, de um modo geral, a comunicação entre computadores e usuários para os mais diversos tipos de aplicação. Em particular, uma ferramenta capaz de gerar textos a partir de uma seqüência de ações é um instrumento utilizável para a narração de estórias.

4.3.3 Sintaxe

Para este trabalho, foi usado um pequeno subconjunto das regras de sintaxe das línguas portuguesa e inglesa, de tal forma que sintaticamente estes dois subconjuntos se correspondem, apesar de as línguas originais serem diferentes. Uma implementação mais fiel às linguagens originais deveria poder acomodar diferenças consideráveis, mas para os fins deste estudo isso não foi necessário. Nesta seção será analisada a estrutura lingüística.

Dentro dessa estrutura, os textos são divididos em segmentos chamados de sentenças, que representam frases e orações. Cada sentença por sua vez é constituída de unidades menores que são organizadas de forma hierárquica.

As sentenças são divididas em duas partes chamadas de: sintagma nominal e sintagma verbal representando, respectivamente, sujeito e predicado. O *sintagma nominal* é constituído opcionalmente de um artigo definido ou indefinido e obrigatoriamente de um substantivo que pode ser próprio ou comum. Já o *sintagma verbal* é formado por um verbo seguido de um complemento, além de zero ou mais modificadores.

Um *complemento* pode ser definido por um sintagma nominal ou por um *sintagma preposicional*, que por sua vez é composto por uma preposição e um sintagma nominal. Já os *modificadores* são subdivididos da mesma forma que os complementos, a diferença é que os modificadores são opcionais e não há limite para o uso de modificadores.

Os elementos sintáticos da gramática podem ser observados através da figura 4.2:

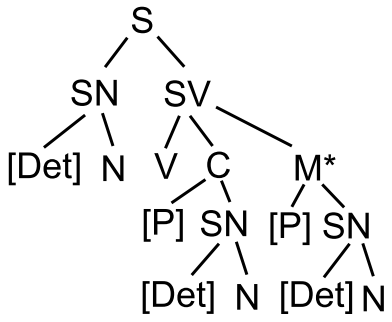

Figura 4.2: Estrutura Sintática

Onde:

S é uma sentença,

SN é um sintagma nominal,

SV é um sintagma verbal,

Det é um artigo,

N é um substantivo,

V é um verbo,

C é um complemento,

M é um modificador,

P é uma preposição

Os colchetes foram utilizados para dizer que um elemento é opcional e o asterisco foi usado para indicar um elemento que pode ser repetido zero ou mais vezes.

4.3.4
Descrição de Uma Fábula

Como visto no capítulo 2.2.2 uma fábula representa o conjunto de todos os eventos que acontecem em uma estória. Como a quantidade de informação necessária para descrever todos esses eventos é muito grande, a descrição feita aqui representa apenas um subconjunto de uma fábula.

Neste estudo de caso, cada ação que acontece na estória é representada pela instanciação de uma operação. Além das tarefas primitivas, também deve ser armazenada toda a hierarquia de tarefas envolvida.

Um exemplo da descrição de uma fábula pode ser visto no anexo D.

4.3.5
Descrição do Léxico

O léxico é uma estrutura de dados que armazena um acervo de palavras, formando um vocabulário. Cada item do vocabulário representa um conjunto de palavras ligadas ao mesmo significado. Por exemplo: *pular*, *pularam* e *pulou* são diferentes palavras que representam a mesma ação.

Os itens do léxico são classificados em tipos como, por exemplo: adjetivos, verbos, preposições, substantivos próprios, entre outros. Cada item pode possuir várias entradas indicando formas diferentes para sua representação.

No léxico usado, cada entrada pode representar a palavra em uma língua ou gênero diferente. Por exemplo, as palavras *menino*, *menina*, *boy* e *girl* fazem parte do mesmo item de vocabulário, mas com variações quanto ao gênero e à língua utilizada.

No anexo B encontra-se o documento que representa o léxico utilizado neste estudo de caso. Um exemplo de descrição de item de vocábulario é mostrado a seguir:

```
<lexicon>
  <lexeme type="noun">
    <entry language="en"    word="boy"    gender="male"/>
    <entry language="en"    word="girl"   gender="female"/>
    <entry language="pt-br" word="menino" gender="male"/>
    <entry language="pt-br" word="menina" gender="female"/>
  </lexeme>
   ...
   ...
</lexicon>
```

4.3.6
Descrição da Conjugação de Verbos

Para gerar sentenças corretamente, o sistema precisa ter conhecimento sobre como podem ser flexionados os verbos para a língua desejada.

A conjugação de cada verbo é definida através de um outro documento que deve conter a lista de conjugações dos verbos do vocabulário.

Os verbos são flexionados quanto ao tempo, modo, número e pessoa. Para cada verbo devem ser indicadas também suas formas no infinitivo, gerúndio e particípio passado.

No anexo A encontra-se o documento que descreve a conjugação dos verbos utilizados neste estudo de caso. Um exemplo simples que mostra a conjugação do verbo *libertar* é mostrado a seguir:

```
<conjugation>
  <language name="pt-br">
    <verb name="free" infinitive="libertar"
          gerund="libertando" pastparticiple="libertado">
      <tense name="present" mood="indicative">
        <conjugation person="1" number="singular" value="liberto"/>
        <conjugation person="2" number="singular" value="libertas"/>
        <conjugation person="3" number="singular" value="liberta"/>
        <conjugation person="1" number="plural"   value="libertamos"/>
        <conjugation person="2" number="plural"   value="libertais"/>
        <conjugation person="3" number="plural"   value="libertam"/>
      </tense>
      <tense name="past" mood="indicative">
        <conjugation person="1" number="singular" value="libertei"/>
        <conjugation person="2" number="singular" value="libertaste"/>
        <conjugation person="3" number="singular" value="libertou"/>
        <conjugation person="1" number="plural"   value="libertamos"/>
        <conjugation person="2" number="plural"   value="libertastes"/>
        <conjugation person="3" number="plural"   value="libertaram"/>
      </tense>
    </verb>
   ...
```

```
   ...
  </language>

  <language name="en">
   ...
   ...
  </language>
</conjugation>
```

4.3.7
Descrição do Mapeamento de Eventos

Para transformar a descrição de uma estória em um texto é preciso ter acesso a mais um documento, contendo informações sobre como mapear as tarefas executadas na estória em estruturas sintáticas que possam ser transformadas em linguagem natural.

Como visto na seção 4.2, as tarefas possuem parâmetros os quais, quando instanciados, passam a conter valores específicos. Neste estudo de caso, esses valores sempre estão na forma de cadeias de caracteres, diretamente utilizável na geração dos textos.

Para cada tarefa deve ser estipulado o verbo que será usado e sua estrutura sintática. É definido ainda como cada argumento será utilizado dentro dessa estrutura. Um exemplo simples, mostrando como gerar texto a partir do verbo ‘dar’, segue abaixo.

```
<phrase task="give">
  <subject>
    <noun_phrase>
      <noun var="from"/>
    </noun_phrase>
  </subject>
  <verb value="give"/>
  <complement>
    <noun_phrase>
      <noun var="item"/>
    </noun_phrase>
  </complement>
  <modifier>
    <prepositional_phrase>
      <preposition value="to"/>
      <complement>
        <noun_phrase>
          <noun var="to"/>
        </noun_phrase>
      </complement>
    </prepositional_phrase>
  </modifier>
</phrase>
```

Neste exemplo, o operador ‘give’ recebe três parâmetros, chamados de ‘from’, ‘item’ e ‘to’, que indicam, repectivamente, quem está dando um objeto, qual é este

objeto e para quem será entregue. Quando este operador for executado, os parâmetros respectivos serão instanciados com valores a serem usados na geração do texto.

4.3.8 Implementação

Para auxiliar na tarefa de geração de textos foi criada uma biblioteca responsável por cuidar de todos os detalhes do processo. Ela foi construída não apenas para ser usada por geradores de estórias, mas para qualquer aplicação que precise gerar textos a partir de uma seqüência de eventos, estejam eles organizados de modo hierárquico ou não.

Essa biblioteca, que suporta tanto o idioma português quanto o inglês, está provida de facilidades para ler a descrição de uma estória através de um documento em XML e apresentar em sua saída um texto descrevendo a estória.

Para ser capaz de gerar o texto, a ferramenta precisa ter acesso a um léxico, contendo algum vocabulário básico, além de informações básicas sobre a sintaxe da língua em questão e sobre como formar frases a partir de cada operador ou método. Essas informações também são passadas através de documentos no formato XML.

Além da biblioteca, também foi criado um aplicativo que recebe como parâmetro o caminho para os documentos com as informações necessárias para a geração, e mostra em sua saída, a qual pode ser um console textual ou um arquivo de texto, o texto final que foi gerado.

4.3.9 Uma Arquitetura para a GLN

Reiter e Dale (1997) defendem uma arquitetura para a GLN dividida em três estágios: planejamento do documento, microplanejamento e realização sintática. Segundo eles, esses estágios devem estar conectados através de um *pipeline* assim como mostrado na figura 4.3.

Para a elaboração deste trabalho foi utilizada essa arquitetura. Nas próximas seções cada um desses estágios será apresentado com mais detalhes.

4.3.9.1 Planejamento do Documento

O planejamento do documento consiste nas tarefas de determinação de conteúdo e na estruturação do documento.

A *determinação de conteúdo* é a tarefa de decidir quais informações devem estar presentes no documento final para satisfazer aos objetivos da comunicação. Podem envolver a filtragem e a sumarização dos dados de entrada (Reiter e Dale 1997).

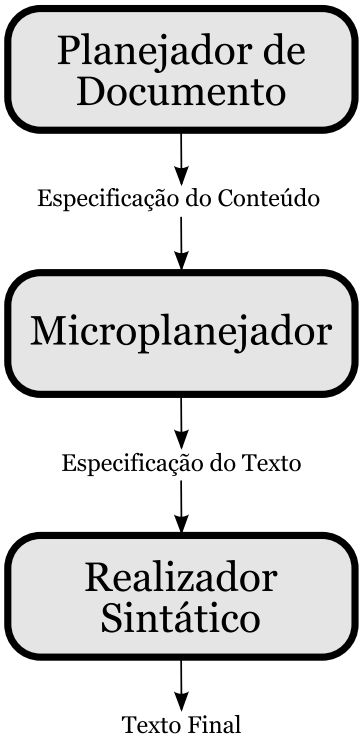

Figura 4.3: Arquitetura para a GLN

O conteúdo deve ser expresso através de entidades e relações. No caso da narração de estórias, por exemplo, as relações dizem respeito às ações que acontecem nas estórias e as entidades indicam quem ou o que gerou a ação e a quem ou a que se aplica a ação.

Neste estudo, a determinação de conteúdo é em parte feita na etapa de geração da estória, pois é reponsabilidade do planejador definir quais os eventos que deverão fazer parte da fábula. Ele entrega em sua saída uma hierarquia de tarefas e cabe ao gerador de textos decidir como irá utilizar os dados contidos nessa árvore.

Uma possibilidade é apenas utilizar as tarefas representadas nas folhas da árvore, mas muitas delas podem ter pouca importância no contexto da estória e deixar o texto desnecessariamente longo. À medida que se sobe na hierarquia através dos nós intermediários, obtem-se uma sumarização das tarefas dos personagens. No entanto, caso se suba muito nessa hierarquia, muitas informações importantes podem ser perdidas.

Na abordagem adotada, a responsabilidade de decidir quais nós utilizar foi delegada ao criador da base de conhecimento, mais especificamente através do documento de mapeamento de eventos, que está descrito na seção 4.3.7. Nesse documento nem todas as tarefas precisam ser mapeadas. A idéia é que apenas as tarefas que possuem informações suficientes para geração de texto sejam mapeadas. Nelas estão incluídas todas as tarefas primitivas.

Durante a geração de texto, o sistema vai tentar sempre gerar texto a partir dos

nós mais altos da hierarquia; caso não seja possível, ele vai descendo na hierarquia até chegar às tarefas primitivas.

Já a *estruturação do documento* é o processo responsável por fazer o agrupamento e a ordenação das informações. O conteúdo deve ser expresso de forma a facilitar sua leitura e entendimento por parte das pessoas, dividindo o texto em partes, seções e parágrafos quando necessário.

Também foram usadas as informações da hierarquia de eventos. No documento de mapeamento de eventos é possível indicar quando o texto relativo a uma tarefa específica deve ser exibido em um parágrafo separado.

4.3.9.2 Microplanemento

O microplanejador recebe como entrada a *especificação do conteúdo*, definindo as informações que deverão ser mostradas e a forma como elas serão agrupadas. O microplanejamento é responsável por tomar decisões sobre a estrutura do texto que será gerado. Esse estágio é dividido nas etapas de lexicalização, aglutinação e geração de expressões referentes (Reiter e Dale 1997).

A *lexicalização* é o processo de decidir quais palavras devem ser escolhidas para expressar as entidades e relações que aparecem nas mensagens. É a partir daí que a linguagem escolhida para a geração do texto (inglês ou português) passa a ser considerada.

Para cada verbo e cada entidade pode haver mais de uma forma de realizar a lexicalização. A escolha pode ser feita segundo critérios diversos. Por exemplo, para melhorar a fluência de um texto, deve-se evitar o uso de palavras repetidas. Além disso, a escolha pode ser feita dependendo do grau de formalidade exigido. As características linguísticas regionais também podem ser levadas em consideração.

A *aglutinação de orações* é o processo de agrupar em uma única oração complexa, certo número de mensagens que poderiam estar em orações distintas. Nem sempre um aglutinador de orações é necessário, pois cada mensagem pode ser expressa através de uma oração separada. No entanto, na grande maioria dos casos esse processo é capaz de melhorar a fluência na leitura de um texto.

Neste trabalho foram definidos três tipos de aglutinações. O primeiro tipo ocorre quando duas ou mais frases possuem o mesmo sujeito, mas os predicados são diferentes. Neste caso, essas orações podem ser unidas através de outra frase, onde a segunda oração tenha um sujeito oculto. Por exemplo, as orações '*Maria é uma princesa*' e '*Maria vive em um castelo*' podem ser transformadas em uma só frase '*Maria é uma princesa e vive em um castelo*'.

O segundo tipo de aglutinação ocorre quando os sujeitos são diferentes, mas o verbo e o predicado são iguais. Por exemplo, as mensagens relativas às frases:

'*Carlos vai para a caverna*' e '*João vai para a caverna*', podem ser reunidas um uma única oração: '*Carlos e João vão para a caverna*'.

O último tipo de aglutinação acontece quando apenas um objeto, direto ou indireto, muda na oração. Por exemplo, as orações '*Antônio entrega a espada para João*' e '*Antônio entrega o escudo para João*' podem se transformar em '*Antônio entrega a espada e o escudo para João*'.

A *geração de expressões referentes* é a tarefa de substituir referências a frases nominais por expressões referentes que sejam capazes de indentificar essa referência ao leitor.

Muitas vezes as expressões referentes são dadas por pronomes, como *isso*, *aquilo*, *ele* e *ela*. Por exemplo, na frase: '*José e Maria foram ao parque e eles se divertiram muito*', as expressões '*José e Maria*' e '*eles*' foram ambas utilizadas para se referir às entidades relacionadas a José e a Maria.

Há casos em que uma mesma entidade pode ser referida de várias formas diferentes. Por exemplo, '*João da Silva*', '*Joãozinho*', '*João*' e '*o menino de camisa azul*' podem estar se referindo à mesma pessoa. A melhor forma de escolher qual expressão utilizar depende do contexto na estória.

Em geral, quando um personagem é apresentado pela primeira vez, sua descrição deve ser mais detalhada, enquanto no desenrolar da estória é normal que se use expressões menores para facilitar a fluência da leitura por parte do leitor.

4.3.9.3
Realização Sintática

A realização sintática é a etapa responsável pela geração final do texto, que deve ser correto do ponto de vista sintático e expressar todas as informações contidas na especificação do texto.

Essa especificação geralmente é feita através de uma árvore, cujos nós intermediários podem representar, por exemplo, seções, parágrafos, frases ou orações; enquanto as folhas representam os elementos mais elementares como pronomes ou substantivos.

O realizador sintático deve saber transformar os nós em palavras organizadas de acordo com a sintaxe da língua especificada. Deve ser capaz de flexionar os verbos, inserir pontuação, além de realizar a diagramação do texto possibilitando a identificação de títulos, parágrafos e seções.

Para a geração de um mesmo texto em línguas diferentes e com sintaxes diferentes é necessário o uso de realizadores sintáticos distintos ou configuráveis.

Neste trabalho, foi utilizada a estrutura sintática descrita na seção 4.3.3. Essa estrutura simplificada serve tanto para o português quanto para o inglês. Durante a realização sintática o sistema deve consultar o documento de léxico e conjugação

de verbos para escolher a forma correta de flexionar os pronomes, artigos e verbos na língua desejada.

Também é tarefa do realizador armazenar o texto em algum formato onde possa ser lido, como, por exemplo, na forma de uma página na internet ou em um documento no formato PDF. Para este estudo o formato de texto simples foi suficiente para exibir a estória.

Um exemplo de texto gerado por esta ferramenta é mostrado a seguir.

```
  Maria é princesa e vive no castelo. João é cavaleiro e vive no castelo. Pedro
é feiticeiro e vive na floresta.

  Pedro vai para castelo, sequestra Maria, força Maria para ir para caverna e
prende Maria dentro da caverna.

  Antonio é mestre, vive no castelo, dá espada magica para João e diz boa-sorte
para João.

  Ethan é andarilho e vive na estrada. João encontra Ethan, Ethan oferece ajuda
para João e João aceita ajuda do Ethan. Samuel é ogro e vive na floresta. João
encontra Samuel, luta com Samuel e derrota Samuel.

  Ethan e João vão para caverna.

  João liberta Maria da prisao.

  João casa com Maria.
```

4.4
Base de Conhecimento

A base de conhecimento deve armazenar as informações necessárias para a representação de estórias e domínios literários. Essas informações devem ser armazenadas em um documento utilizando uma linguagem baseada em XML que foi criada para este fim.

O padrão XML foi escolhido, entre outros motivos, pelo grande número de ferramentas e bibliotecas disponíveis para a manipulação de documentos nesse formato e pela possibilidade de extensão futura da linguagem.

Para auxiliar o processo de geração e narração de estórias, foi implementada uma biblioteca que possibilita manipulação, leitura e escrita sobre a base de conhecimento.

O acesso à biblioteca facilita o trabalho do programador, permitindo abstrair os detalhes específicos da linguagem utilizada. Ela foi bastante utilizada na implementação do protótipo, tanto no processo de geração quanto no de narração. Foi utilizada também na elaboração de uma ferramenta de autoria.

A ferramenta de autoria é responsável por facilitar a elaboração da base de conhecimento por parte do usuário ou programador.

A biblioteca admite as seguintes abstrações de dados:

- **Fábulas**: uma fábula é representada na forma de uma árvore. Nela cada nó intermediário representa a execução de uma tarefa. Cada tarefa pode ser decomposta em subtarefas ordenadas, representadas pelos filhos do nó correspondente. As folhas representam as ações primitivas que não podem ser decompostas. Também está presente a descrição do estado inicial da estória, enquanto os demais estados intermediários podem ser calculados a partir do estado inicial e das ações que forem sendo executadas;
- **Domínios literários**: um domínio permite descrever as restrições de um gênero literário. Consiste na especificação dos atributos e axiomas existentes, além dos métodos e operadores que podem ser executados;
- **Estados**: um estado consiste basicamente de uma lista com os fatos que são verdadeiros em um dado instante de tempo;
- **Fatos**: cada fato representa a instanciação de um atributo com argumentos contendo valorações para cada parâmetro do atributo;
- **Atributos**: os atributos representam todas as grandezas que podem ser valoradas em um domínio literário. No estudo de caso em questão somente são suportados valores do tipo *string*, mas futuramente o exemplo pode ser expandido para aceitar outros tipos de valores como números inteiros e fracionários, listas, etc. Cada atributo possui uma lista de parâmetros que precisam ser todos preenchidos quando o atributo é instanciado;
- **Condições lógicas**: condições lógicas servem para testar os valores dos atributos. Uma condição lógica pode ser definida a partir de um teste de igualdade sobre o valor de um atributo, ou então a partir de conjunções, disjunções e negações envolvendo outras condições lógicas;
- **Axiomas**: os axiomas representam implicações lógicas. Cada axioma é composto por uma pré-condição, dada por uma expressão lógica, e uma conseqüência que representa um fato que será considerado verdadeiro caso a pré-condição seja válida;
- **Operadores**: um operador representa uma ação que pode ser executada no mundo. É descrita segundo a formalização de operadores STRIPS. Sendo assim, sua descrição contém uma lista das pré-condições que precisam ser válidas para sua execução, e também de listas de pós-condições, que podem ser de adição ou de remoção. Além disso, os operadores também podem possuir parâmetros que devem ser preenchidos para sua instanciação;
- **Métodos**: a descrição de um método deve possuir uma referência à tarefa a executar. Deve conter também uma lista com os parâmetros de entrada

do método e uma lista ordenada de tuplas. As tuplas devem armazenar uma condição de execução e uma lista de subtarefas que precisam ser executadas.

4.5
Ferramenta de Autoria

Tendo em vista que a qualidade do processo de geração e narração de estórias é fortemente ligada à qualidade da base de conhecimento, observa-se a necessidade de se dispor de ferramentas cada vez mais poderosas para apoiar sua construção.

Com o uso de uma ferramenta de autoria, o autor[1] não precisa estar familiarizado com os detalhes relativos à linguagem utilizada, além de poder contar com facilidades para a visualização das informações e para acertar as dependências entre elas.

Além disso, a ferramenta de autoria implementada permite a geração de estórias e sua visualização. No curso da visualização, podem ser observadas a seqüência de ações, a rede de tarefas envolvidas e os estados intermediários, o que pode ser muito útil numa etapa de depuração.

Essa ferramenta foi criada utilizando o *Swing*, que é uma API Java para a criação de interfaces gráficas. A tela principal do aplicativo é mostrada na figura 4.4.

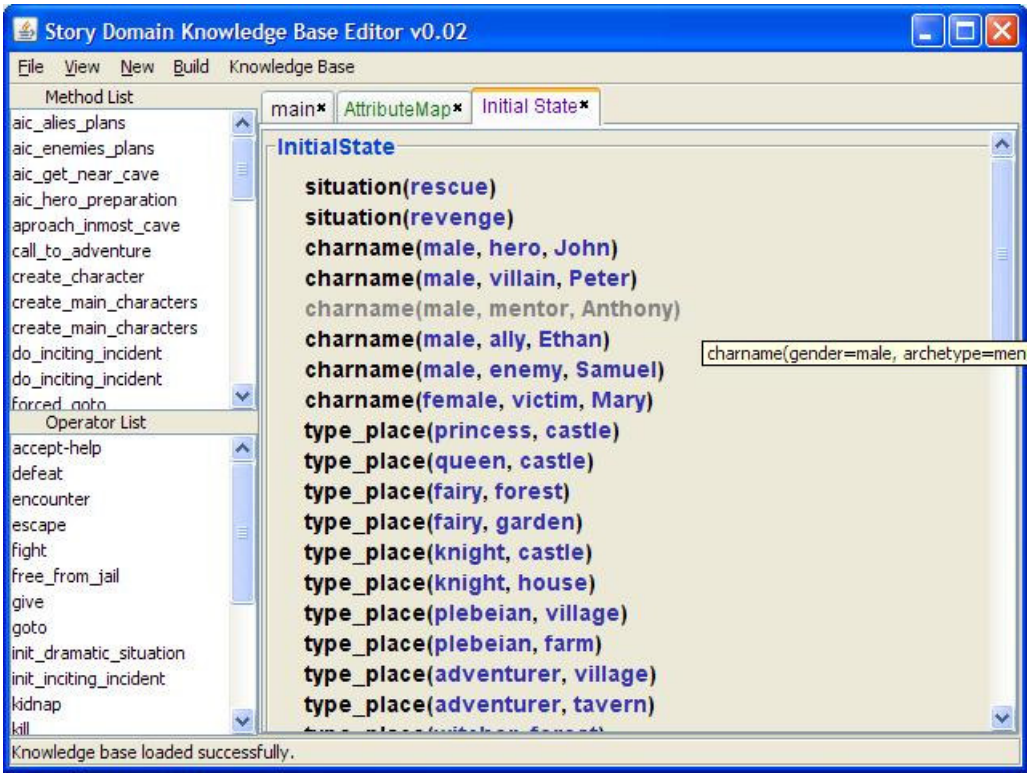

Figura 4.4: Tela inicial da ferramenta de autoria

[1]Neste texto estamos chamando de autor, o usuário da ferramenta de autoria que insere as informações na base de conhecimento

À esquerda são apresentadas duas listas, uma contendo a lista de todos os métodos da base de conhecimento e a outra os operadores. Quando o usuário executa um duplo clique, o método ou operador correspondente irá aparecer em uma nova aba que será selecionada para exibição.

Foi utilizada uma estrutura de abas, semelhante às usadas pelos principais sistemas de navegação pela internet. Essa estrutura foi escolhida por facilitar a edição de várias estruturas de dados utilizando uma única janela. Novas abas podem ser abertas e fechadas pelo usuário, facilitando o acesso à edição de suas respectivas informações.

Nas abas podem ser editados métodos, operadores e estados iniciais, além dos tipos de atributos da base de conhecimento.

A tela de edição de um método, que pode ser vista na figura 4.5, contém as informações sobre as pré-condições e a lista de tarefas de cada tupla existente. Através de um clique com o botão direito do mouse o usuário pode abrir um conjunto de opções, onde é possível adicionar uma nova tupla ou remover uma já existente. Também é possível adicionar, remover ou editar elementos da lista de tarefas ou da lista de pré-condições.

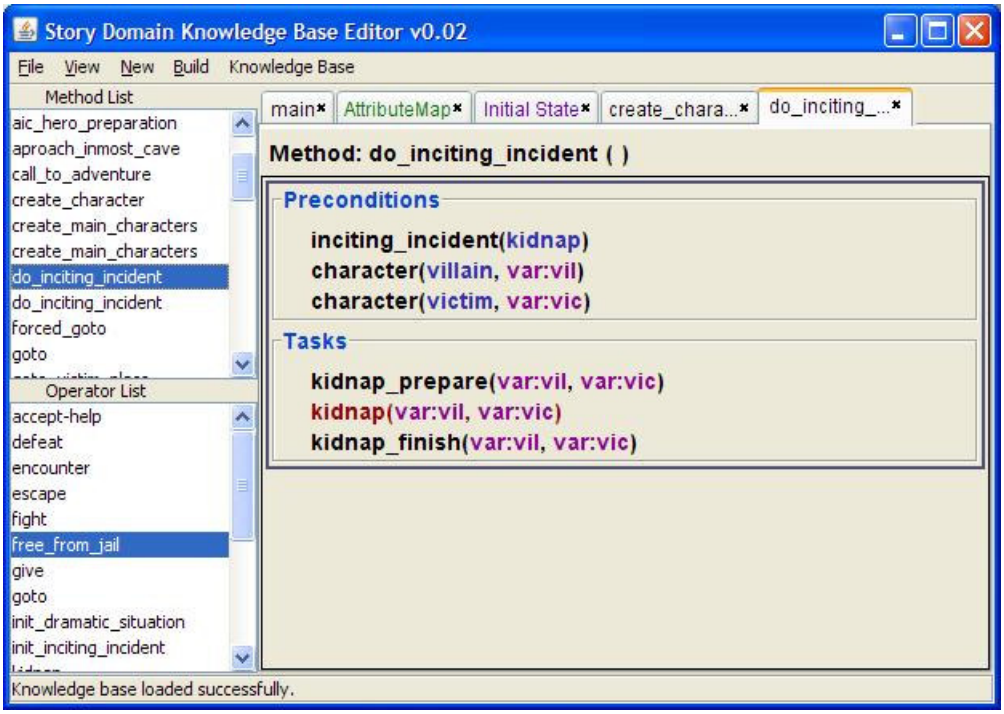

Figura 4.5: Edição de um método

A lista de tarefas contém referências a eventos, que são instanciações de operadores ou métodos. Com o intuito de ajudar na navegação entre tarefas, ao clicar em uma das tarefas uma nova aba se abrirá, possibilitando a edição da tarefa correspondente.

Para facilitar a identificação, operadores e métodos são mostrados em cores diferentes. Os argumentos podem ser valores ou variáveis, que também são exibidos com cores diferentes para facilitar a identificação.

A edição de eventos, fatos e atributos é feita através de janelas ao invés de abas, porque a quantidade de informações é menor. Em geral essas janelas devem ser abertas uma de cada vez, o que não seria conveniente para a exibição de métodos e operadores.

A janela de edição de eventos contém uma *combobox*, onde deve ser escolhida a tarefa correspondente ao evento. Dependendo da tarefa escolhida, o campo de argumentos será preenchido com os parâmetros correspondentes ao evento, podendo corresponder a variáveis ou a valores, assim como é mostrado na figura 4.6.

Figura 4.6: Edição de um evento

Os fatos são editados através de uma janela bem semelhante. Deve ser escolhido um atributo e devem ser atribuídos valores ou variáveis a seus parâmetros. Também existe a opção de negar a existência de um fato, o que é especialmente útil para a construção de pré-condições.

Uma das vantagens de usar uma ferramenta de autoria é a possibilidade de testar a base de conhecimento, gerando estórias com rapidez. A ferramenta de autoria implementada neste estudo permite visualizar a estória de três formas diferentes.

A primeira forma consiste em exibir a estrutura hierárquica das tarefas executadas. A exibição é feita utilizando a estrutura *JTree* da *API Swing* de *Java*. Essa estrutura permite, por exemplo, que os itens de uma árvore sejam expandidos ou ocultados, facilitando a visualização por parte do usuário. Está representada na figura 4.7.

A segunda forma de exibição é especialmente útil para a depuração, pois exibe ao mesmo tempo as tarefas que foram executadas e o estado do mundo antes da execução da tarefa selecionada, como se mostra na figura 4.8.

A última forma corresponde à simples exibição do texto correspondente à estória e é útil para se ter uma visão do resultado final da geração.

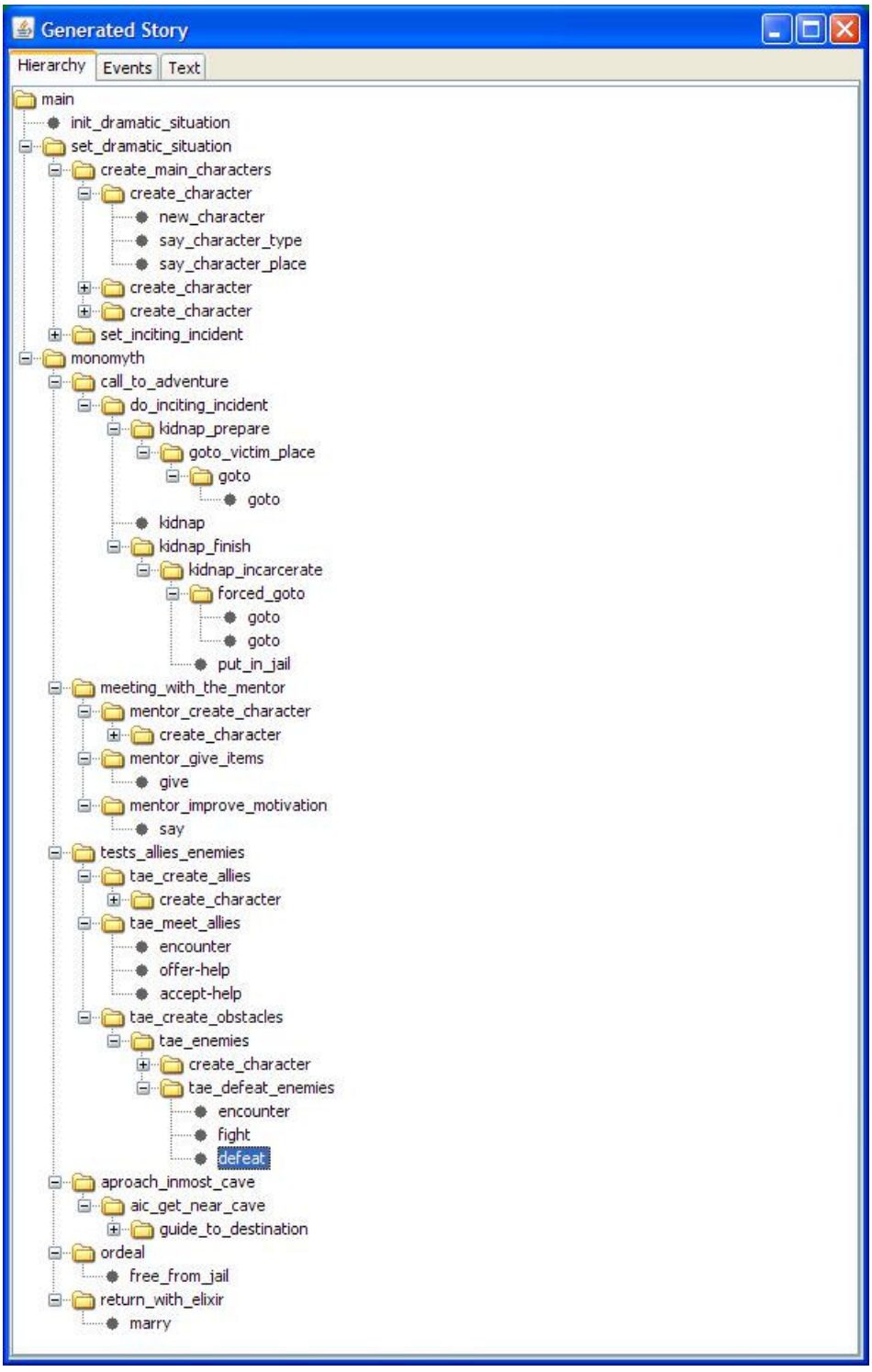

Figura 4.7: Exibição da hierarquia de tarefas

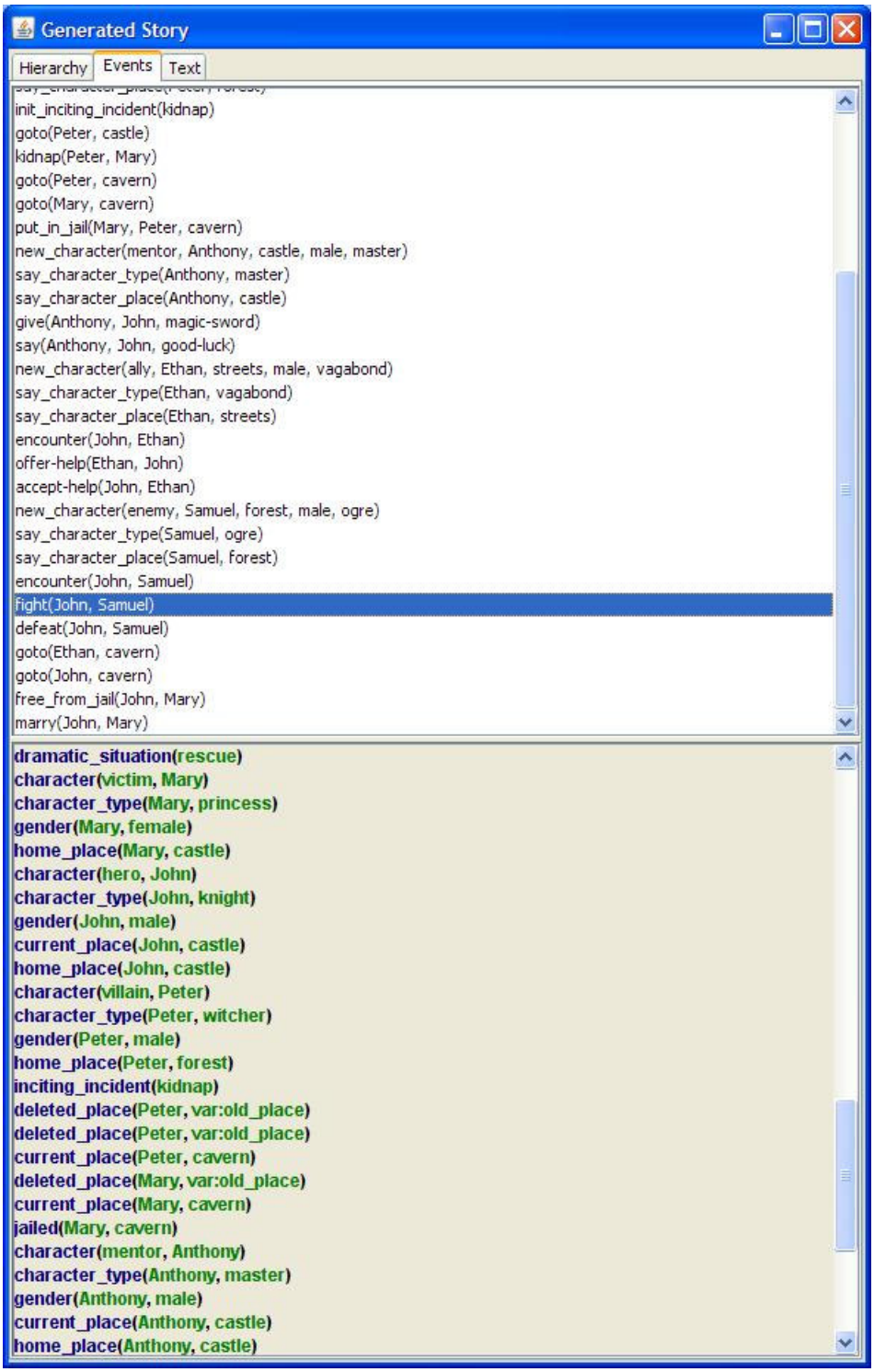

Figura 4.8: Exibição da lista de tarefas e estados correspondentes