

Elegy: Aplicando o Processo de Fábrica de Jogos ao Domínio Role-Playing Games

Leila S. de S. T. de Azevedo André L. de M. Santos André W. B. Furtado

Universidade Federal de Pernambuco, Centro de Informática, Brasil.

Resumo

Este paper apresenta uma solução de fábrica de jogos, denominada Elegy, para o domínio de jogos RPG, em busca de obter um maior custo/benefício durante a produção desses *softwares*. Para a elaboração da Elegy foram utilizados conceitos como *domain-specific languages* (DSLs) e uma metodologia de análise de domínio para jogos intitulada SharpLudus.

Palavras-chave: DSL, Role-Playing Games, Fábrica de Jogos.

Authors' contact:

{lssta,awbf,alms}@cin.ufpe.br

1. Introdução

A indústria de jogos eletrônicos é uma das mais expressivas no ramo do entretenimento [1], equiparado-se às indústrias de cinema e música. Porém, a indústria de software em geral ainda hoje funciona de forma artesanal, desta forma a produção não atende de forma eficiente à demanda de mercado[2].

Neste cenário, surgem opções para automatizar a produção de software. Uma abordagem interessante é a implementação de fábricas de software, baseada na produção de softwares de uma mesma família, através do uso de linguagens, *frameworks* e ferramentas [2]. Neste paradigma são montadas verdadeiras linhas de produção de software, focadas em soluções para problemas de uma mesma espécie.

Alinhado à iniciativa de Fábricas de Software, encontramos o conceito de Desenvolvimento de Domínio Específico, abordagem utilizada para resolver problemas que pode ser aplicada quando um problema em particular ocorre repetidamente [3]. Baseia-se no conceito de Linguagens de Domínio Específico (*Domain-Specific Languages* – DSLs), forma limitada de linguagens computacionais projetadas para uma classe específica de problemas que oferecem, através de notações e abstrações apropriadas, expressivo poder de foco, e usualmente restrito, a um problema do domínio.

Este trabalho apresenta uma alternativa, baseada no uso de Fábricas de Software e Desenvolvimento de Domínio Específico, para produção de jogos no domínio *Role-Playing Games*.

2. Análise de Domínio

Por quase três décadas, a área de Análise de Domínio (Domain Analysis ou DA) tem servido outras iniciativas da Engenharia de Software, como Desenvolvimento baseado em Componentes, Linhas de Produção de Software e Reuso de Software em geral [4]. Segundo Prieto-Diaz [5], DA é definida como um processo pelo qual a informação utilizada no desenvolvimento de sistemas de software é identificada, capturada e organizada com o propósito de fazê-la reusável na criação de novos sistemas.

O termo Análise de Domínio foi inicialmente introduzido por Neighbors [6] como “a atividade de identificar os objetos e operações de uma classe de sistemas similares num domínio particular de problemas”. Ainda segundo ele, “a chave para softwares reusáveis é capturada na análise de domínio na medida em que se salienta a reusabilidade da análise e *design*, não código”.

Porém, nem Neighbors nem Prieto-Diaz, dentre outros autores, resolveram o problema de como fazer a análise de domínio. É necessário desenvolver abordagens para análise de domínio, a fim de explorar verdadeiramente o reuso em ambientes industriais.

A metodologia SharpLudus proposta por Furtado [4], propõe uma abordagem pragmática de análise de domínio para a criação de fábricas de jogos, que engloba as atividades de definição de características sob análise, seleção de amostras da linha de produção, execução da análise e, finalmente, a definição das características comuns e variabilidades.

Esta metodologia foi considerada mais adequada por ser focada na automatização no desenvolvimento de jogos, e portanto escolhida para ser aplicada na criação da fábrica *Elegy*. Nas próximas seções encontra-se o resumo de cada etapa do processo no domínio RPG.

2.1 Funcionalidades do domínio sob Análise

Nesta etapa foram definidas funcionalidades e características consideradas importantes no domínio de jogos RPG eletrônicos. A Tabela 1 apresenta as variáveis escolhidas para análise.

Tabela 1: Variáveis de Análise.

Funcionalidade	Descrição
Descrição	Principais características do jogo em geral.
História	Breve descrição do enredo.
Fluxo da História	Descrição de como se dá o desenrolar da história, linearidade, presença ou não de missões alternativas e impacto das ações dos jogadores na história.
Personagens	Características, atributos e habilidades de personagens jogáveis, presença ou não de personagens não jogáveis, quantidade de personagens que podem ser controlados.
Desenvolvimento dos Personagens	Como os personagens evoluem no decorrer da história.
Cenário	Descrição do mundo e época em que se passa a história.
Sistema de Batalha	Descreve como as características inerentes aos personagens podem afetar no resultado das ações de batalha. Lógica e visão estratégica da batalha também são pontos cruciais neste item.
Fluxo em Batalha	Ações que podem ser tomadas dentro da batalha e como os personagens podem interagir neste ambiente.
Fluxo fora de batalha	Ações que podem ser tomadas fora da batalha e como os personagens podem interagir.
Itens	Itens e equipamentos que podem ser encontrados durante o jogo.

2.2 Seleção de Amostras

Foram escolhidos 5 jogos considerados representativos no domínio:

- Final Fantasy VI
- Chrono Trigger
- Baldur's Gate II
- Seiken Densetsu III

- Neverwinter Nights

2.3 Execução da Análise de Domínio

Nesta etapa as variáveis escolhidas anteriormente foram analisadas em relação a cada um dos jogos escolhidos.

2.4 Extração de Semelhanças e Variabilidades

Nesta etapa as funcionalidades semelhantes e variáveis foram organizadas através de *feature models*. Para tal, foi utilizada a ferramenta Feature Model DSL [7]. A Figura 1 mostra a raiz dos diagramas.

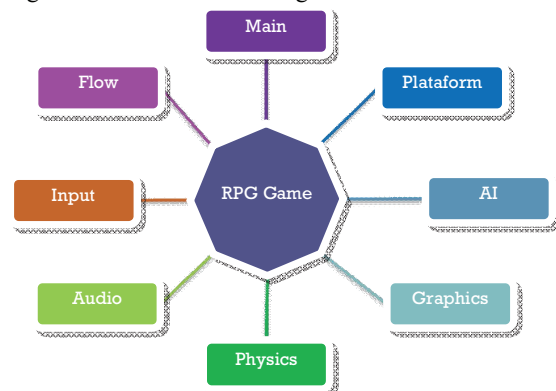


Figura 1: Variáveis de Análise.

2.5 Validação do Domínio

Nesta etapa, com o *feature model* pronto, uma amostra que não foi incluída na análise será usada para determinar a generalidade do modelo, ou seja, o jogo escolhido deverá ter suas funcionalidades mapeadas no *feature model* projetado. O domínio foi validado pelo jogo *Persona 3*, da série *Megami Tensei*, lançado em 2006 pela *Atlus* para a plataforma *Playstation 2*.

2.6 Critério de Parada para a DA

O tempo foi o critério levado mais fortemente em conta para a finalização da análise. Como os jogos RPG costumam ser bastante complexos, o número de funcionalidades mapeadas no modelo certamente representa apenas um subconjunto, ainda que bastante representativo e generalizado. Desta forma termina o processo de análise de domínio e funcionalidades, passando agora para o momento de implementação da fábrica.

3. Elegy Game Factory

O resultado obtido da análise de domínio, no sentido *top down*, e o estudo da *engine RPG Starter Kit* [8], escolhido como *domain framework* (a ser consumido pelo código gerado por DSLs da fábrica), foram utilizados para guiar a modelagem, através da combinação de abordagens *top-down* e *bottom-up*.

A abordagem top-down abrange toda aplicação da metodologia sobre o domínio, resultando nos feature models, ao mesmo tempo em que é feito um estudo em cima da engine a ser reusada, no sentido bottom-up. O resultado da aplicação destas duas abordagens em paralelo culmina na modelagem da linguagem visual, juntamente com o código a ser gerado. Desta forma uma camada de abstração surge de modo a facilitar a criação de jogos através do uso de elementos do domínio, como personagens, mapas, inimigos, quests, entre outros, sem a necessidade de programação em linhas de código.

Foram modeladas 3 DSLs para compor a fábrica: Map Manager DSL, Quest Manager DSL e Entity Definition DSL. Estas DSLs foram modeladas separadamente, mas como pontos de integração entre si. O escopo deste projeto não abrange a integração entre as DSLs.

Nas próximas seções serão apresentadas as DSLs modeladas no *Microsoft DSL Tools* [9]. Cada DSL gera código XML para a plataforma XNA. Este código XML é consumido pela *engine* RPG Starter Kit para a criação de jogos.

A grande contribuição deste projeto está justamente na geração do código XML a ser consumido, pois esses arquivos muitas vezes são extensos e complexos para serem escritos manualmente, tornando desta forma a criação de jogos enfadonha e lenta.

3.1 MapManagerDSL

Esta DSL é responsável por gerenciar a criação de mapas e a transição entre eles.

Através desta DSL podem-se criar diversos mapas, com transições entre eles, como mostra a Figura 3. Para definição de *tiles* e objetos do mapa usa-se um editor, como pode ser visto na Figura 2. Este editor também exibe opções como definição de pontos de colisão no mapa e por onde se dará a transição de um mapa para outro.

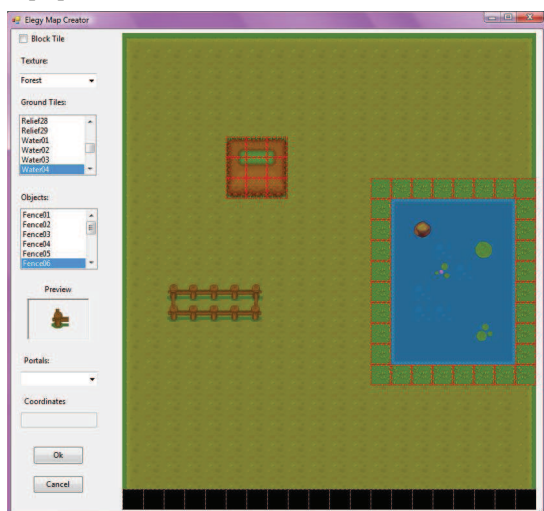


Figura 2: Editor customizado de propriedades do mapa.



Figura 3: Representação visual de mapas e transições.

3.2 QuestManagerDSL

Esta DSL é responsável por gerenciar a criação de *quests*.

Através desta DSL pode-se definir uma *quest*, como pode ser visto na Figura 4.

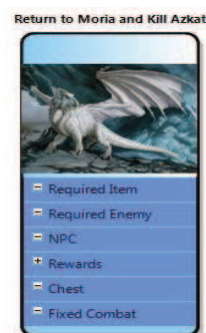


Figura 4: Representação visual de uma *quest*.

Uma *quest* apresenta características como itens e inimigos requeridos, recompensas, entre outros, que são modificados através de editores customizados.

3.3 EntityDefinitionDSL

Esta DSL é responsável pela criação de personagens, tanto personagem principal, inimigos e NPCs de *quests*.

O personagem principal, inimigos e NPC's podem ser definidos como mostra a Figura 5.

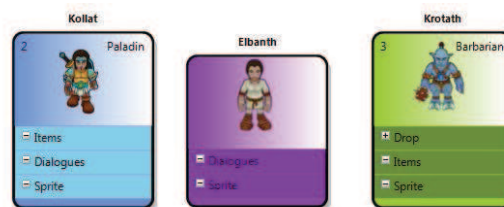


Figura 5: Representação visual dos personagens.

Editores para modificar as propriedades dos personagens também foram implementados, como o editor de escolha de *sprites*, representado na Figura 6.

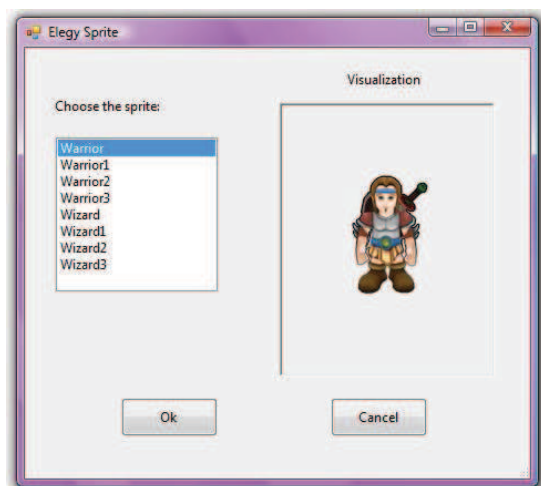


Figura 6: Editor de escolha dos sprites do personagem.

4. Conclusão

A importância deste trabalho não se resume à visualização de uma ferramenta consolidada e com grande potencial. O grande desafio na implementação produtiva de *games* através da abordagem de Fábricas de Software está na escolha de um processo adequado, confiável e que traga resultados satisfatórios. Este processo deve abranger todas as etapas de elaboração do projeto, desde sua idealização, design, até sua implementação.

O alvo do projeto Elegy esteve justamente no estudo e aplicação da metodologia SharpLudus durante a etapa de análise de domínio e funcionalidades, trazendo contribuições para esta metodologia, que ainda está em produção.

Para validar o projeto de DSL é necessário um framework, que consumirá o código gerado. A *engine RPG Starter Kit* foi escolhida para esta finalidade, com a vantagem de trazer portabilidade para a plataforma XNA. Desta forma a criação da Elegy não foi feita de forma linear. Enquanto a análise era processada, um estudo sobre a engine foi feito e culminaram na produção das três DSLs apresentadas na seção 3.

Muitos desafios foram encontrados devido à engine não ter sido produzida propriamente para a Elegy. Alguns deles puderam ser contornados, fazendo com que a criação de jogos RPG com esse motor de jogos se tornasse mais produtiva e interessante. Outros, como a necessidade de integração de DSLs, não puderam ser resolvidos nesta versão.

5. Trabalhos Relacionados

Este projeto teve como iniciativa a fábrica de jogos SharpLudus, proposta por Furtado [4]. A Elegy visou complementar a fábrica SharpLudus, que foi inicialmente focada em jogos do domínio *arcade*. Assim como a SharpLudus, a Elegy utiliza o conceito

de análise de domínio e desenvolvimento de domínio específico, com uso de DSLs para gerar o código a ser consumido pela *engine* escolhida.

Referências

- [1]ESA. (2007). Essential Facts About the Computer Game Industry. Acesso em 1 de Março de 2009, disponível em www.theesa.com/facts/pdfs/ESA_EF_2008.pdf.
- [2]GREENFIELD, J., ET AL(2004). Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. Wiley & Sons, 2004.
- [3] FOWLER, M. Language Workbenches: The Killer-App for Domain Specific Languages? Acesso em 26 de Março de 2009, disponível em martinfowler.com/articles/languageWorkbench.html;
- [4]FURTADO, A. W. (2006). SharpLudus: Improving Game Development Experience through Software Factories and Domain-Specific Languages.Dissertação de Mestrado. Recife: UFPE.
- [5]PRIETO-DIAZ, R. (1990). Domain Analysis: An Introduction. ACM SIGSOFT .
- [6]NEIGHBORS, J. M. (1980). Software Construction Using Components Software Construction Using Components.
- [7]FURTADO, A. W. (2008). Feature Model DSL. Acesso em 22 de Abril de 2009, disponível em <http://www.codeplex.com/FeatureModelDSL>.
- [8]RPG STARTER KIT. Acesso em 1 de Março de 2009, disponível em <http://creators.xna.com/en-US/starterkit/roleplayinggame>.
- [9]MICROSOFT DSL TOOLS. Disponível em <http://msdn.microsoft.com/en-us/library/bb126235.aspx>.