4
A instância PoliFacets-AS

Este capítulo apresenta a instância PoliFacets-AS, com suas facetas e ligações (existentes e idealizadas) demonstrando como o modelo abstrato ganha vida em casos práticos. Ao final, é relatada uma avaliação da recepção das facetas e uma prova de conceito do modelo PoliFacets.

4.1.
Facetas instanciadas e suas ligações

Analisando o PoliFacets-AS a partir do modelo PoliFacets (Figura 4.1), o sistema de significação de entrada é o conjunto de arquivos que constituem um projeto desenvolvido no AS, assim como as representações existentes e a ontologia de domínio. O analisador captura e organiza as informações que formam os signos de programa e programação.

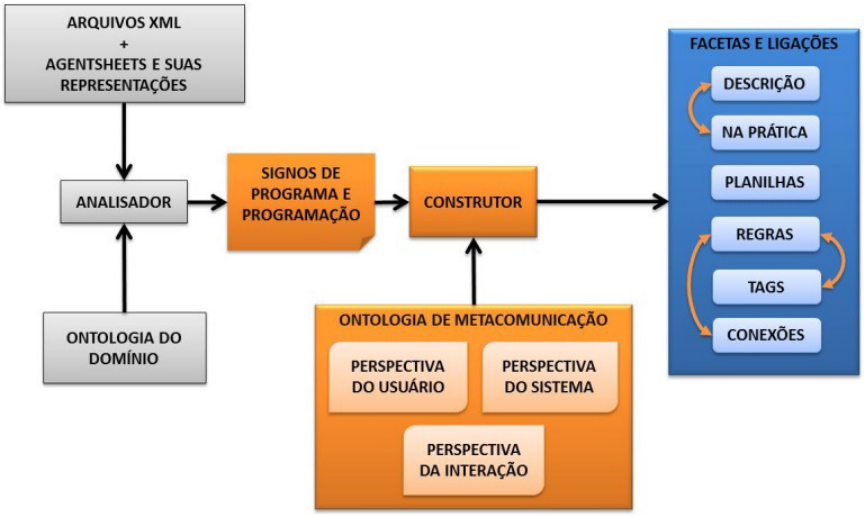

Figura 4.1 - Modelo PoliFacets-AS

A Tabela 4.1 exibe a grade da ontologia de metacomunicação preenchida com as facetas e ligações existentes no PoliFacets-AS. As facetas surgiram de necessidades observadas na avaliação do AS, nos estudos nas escolas e nos estudos empíricos realizados no projeto SGD-Br.

Tabela 4.1 - Grade da ontologia de metacomunicação no PoliFacets-AS

			Metacomunicação através de Signos ESTÁTICOS (principalmente)			**Metacomunicação através de Signos DINÂMICOS (principalmente)**		
	Operações Semióticas @ Usuário	**Operações Simbólicas @ Sistema**	LOCAL foca um elemento de programa	REGIONAL foca um conjunto de elementos de programa	GLOBAL foca todos os elementos de programa	LOCAL foca um elemento de programa	REGIONAL foca um conjunto de elementos de programa	GLOBAL foca todos os elementos de programa
Operações sobre o sistema de significação do OBJETO do Documento Ativo	RESSIGNIFICAÇÃO dá novos significados aos signos do programa	-						Na prática
	PARÁFRASE expressa os signos do programa de maneira alternativa	REFORMULAÇÃO sem alteração semântica	Regras					
		REESCRITA com alteração semântica		Conexões				Planilhas
	FIGURAS DE LINGUAGEM faz combinações inéditas com os signos do programa e revela novos significados	REFORMULAÇÃO sem alteração semântica			Tags			
		REESCRITA com alteração semântica						
Outro Sistema de Significação	EXPANSÃO LINGUÍSTICA dá novos significados aos signos do programa	REFORMULAÇÃO sem alteração semântica			Descrição			
		REESCRITA com alteração semântica						

Por enquanto, foram instanciadas as facetas: descrição, na prática, planilhas, regras, tags e conexões. Elas estão classificadas conforme seu posicionamento na grade da ontologia de metacomunicação. É importante dizer que esta classificação é apenas uma interpretação do *designer* do PoliFacets-AS. Outros *designers* poderiam classificar as facetas de modo diferente, já que o modelo, como ferramenta epistêmica, não gera as facetas, somente ajuda a pensar sobre elas. A descrição das facetas e sua classificação de acordo com a ontologia de metacomunicação do modelo PoliFacets serão apresentadas nas subseções a seguir.

Antes de descrever as facetas individualmente, vale observar como elas foram distribuídas na grade e o que essa distribuição significa. Na Figura 4.2 estão quatro gráficos que demonstram a distribuição das facetas. Na perspectiva do usuário, a ressignificação, a figura de linguagem e a expansão linguística introduzem novos significados. As facetas destes tipos devem ser projetadas com cuidado, especialmente no caso das facetas com expansão linguística, que envolve outro sistema de significação. Nós adotamos a estratégia de criar as facetas de modo que a complexidade dos significados fosse gradual. No contexto do ensino, as paráfrases foram mais utilizadas no PoliFacets-AS porque permitem reapresentar os mesmos dados de um modo diferente, elas são novas explicações dos conteúdos disponíveis, por isso aparecem em número maior no gráfico de operações semióticas.

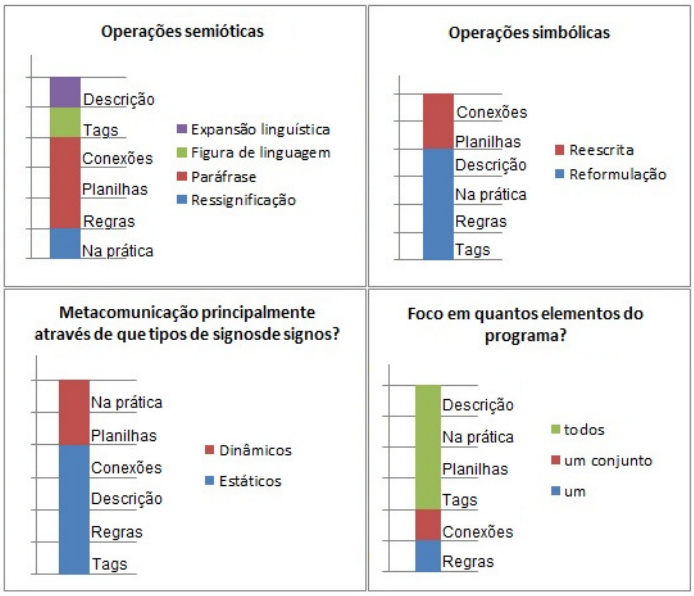

Figura 4.2 - Distribuição das facetas

Com relação às operações simbólicas, observamos um maior número de facetas de reformulação do que de reescrita. A reformulação consiste em renomear ou reordenar o que já existia. Nesse caso, apresentamos dados de outra forma para ressaltar pontos específicos. Na reescrita, o sistema não só reorganiza os dados, mas produz novas informações. Estas facetas são muito interessantes, mas devem ser introduzidas de acordo com as necessidades observadas para que os aprendizes não fiquem sobrecarregados com as novas informações.

A metacomunicação no PoliFacets-AS acontece principalmente através de signos estáticos, com quatro facetas. Os signos dinâmicos são usados apenas quando os estáticos não são suficientes. Nesse caso, o jogo ou simulação da faceta 'na prática' é naturalmente um signo dinâmico. Os signos na faceta 'planilhas' precisam ser explorados no tempo para tentar transmitir, de modo eficaz e eficiente, a mensagem que desejamos, sobre agentes empilhados, agentes invisíveis e imagens de fundo. Apesar de a planilha ser uma representação espacial, os agentes são posicionados em camadas e decidimos usar signos dinâmicos para a facilitar a visualização de todas as informações que estão escondidas no AS. De certo modo, os signos estáticos são fundamentais para a Engenharia Semiótica porque eles formam os signos dinâmicos (um conjunto de signos estáticos desdobrados no tempo) e metalinguísticos (signos estáticos ou dinâmicos fazendo referência a outros signos). Além disso, depois que o usuário entra em contato com a linguagem de interface e percebe a metamensagem comunicada pelo *designer*, quando ele visualizar um signo estático que faz parte de um signo dinâmico, ele entenderá a mensagem comunicada mesmo antes de realizar a interação.

Com relação às facetas e escopos do programa, a maioria das representações do PoliFacets-AS focam em todos os elementos do programa para complementar as representações do AS que têm o escopo mais local. Além disso, iniciamos a introdução de uma faceta com o escopo regional, focando em um conjunto de elementos do programa. Apesar de conhecermos o valor da faceta de escopo regional, decidimos iniciar a sua introdução de modo gradual para os aprendizes tenham condições de absorver essas novas formas de explorar os significados.

Pensando na classificação geral, o PoliFacets-AS é formado, em sua maioria, por facetas de paráfrase com reformulação e a metacomunicação acontece principalmente por meio de signos estáticos focando em todos os

elementos do programa. Como uma ferramenta epistêmica, o modelo instanciado fala de volta para o *designer* reportando a nossa maior atenção em determinados aspectos. A interpretação do *designer* sobre a representação que ele criou é a representação falando de volta com o *designer* (Nakakoji e Yamamoto, 2001) baseada na teoria de reflexão e ação (Schön, 1983). Entretanto, também é importante observar que a ausência de facetas de alguns tipos tem um significado. Por exemplo, não foi explorada a expansão linguística com reescrita. Isto não é por acaso, pois este tipo de faceta tem alto grau de complexidade, especialmente para aprendizes que estão fazendo parte do projeto SGD-Br e da pesquisa desta tese.

4.1.1. Faceta 'descrição'

Esta faceta corresponde a dois tipos de informações: uma descrição geral do projeto e o conjunto de instruções sobre ele. A descrição e as instruções são apresentadas na página de descrição do projeto, a página de abertura de exploração. A Figura 4.3 mostra a faceta no projeto *Pacman*.

Esta é a melhor oportunidade do criador do projeto explicar em linguagem textual a sua intenção de comunicação, ele pode contar uma história ou explicar o comportamento dos agentes de acordo com o seu desejo.

Figura 4.3 - Faceta 'descrição' do *Pacman*

No modelo

Perspectiva da interação

A faceta não apresenta signos que se modificam em diferentes instantes no tempo e, usualmente, faz referência ao projeto como um todo. Mesmo considerando a liberdade do criador do jogo de escrever sobre partes locais ou regionais do programa, o *design* dessa faceta foi projetado para uma descrição mais abrangente.

Perspectiva do usuário

Como a descrição é formada por dois campos de texto abertos, podem ocorrer modificações em todas as dimensões semióticas. O vocabulário utilizado e a gramática da linguagem são diferentes da representação original do programa. Além disso, há nova intenção, pois esta faceta permite descrever o projeto e fornecer instruções específicas sobre o seu funcionamento.

Perspectiva do sistema

Consideramos as alterações do tipo léxico e sintático porque agregamos a descrição e as instruções de funcionamento do jogo ou simulação que surgiram como requisito nos momentos de compartilhamento dos projetos. Apesar de o sistema de significação ser outro, a faceta 'descrição' não realiza modificações semânticas, pois os dados adicionados não são interpretados pelo sistema.

4.1.2. Faceta 'na prática'

Esta é a faceta onde o jogo ou simulação pode ser executado. É o produto resultante da programação desenvolvida no AS. Na Figura 4.4, está um exemplo da faceta 'na prática' do projeto *Frogger* no PoliFacets-AS.

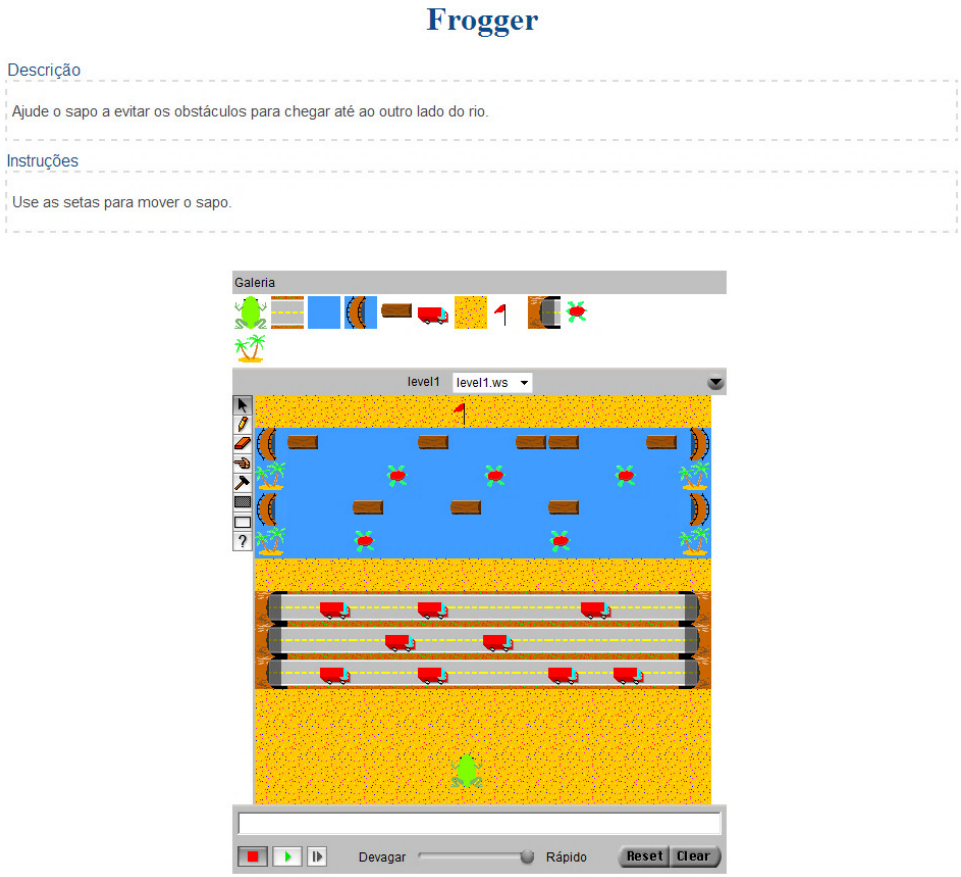

Figura 4.4 - Faceta 'na prática' do projeto *Frogger*

No modelo

Perspectiva da interação

Esta faceta é descrita como um signo dinâmico que foca todos os elementos do programa, pois retrata a execução do projeto completo do AS. Por definição, a execução de um programa é um signo dinâmico.

Perspectiva do usuário

A planilha de trabalho no AS é o local onde os agentes são posicionados. Durante a construção do projeto, a planilha é usada para simulação, testes e conferência do comportamento programado nos agentes. A faceta 'na prática' possui uma nova intenção, que é a de apresentar o resultado final do que foi construído no projeto, é uma ressignificação com o mesmo conteúdo e expressão.

Perspectiva do sistema

Ressignificações não apresentam operações simbólicas correspondentes.

Ligação

A faceta 'na prática' disponibiliza o *applet,* que executa o jogo ou simulação produzidos no AS, junto com a descrição e o campo de instruções que fazem parte da faceta 'descrição', isto concretiza a ligação entre as duas facetas.

4.1.3. Faceta 'planilhas'

Esta faceta apresenta a organização espacial do projeto. Podemos verificar quantas planilhas existem, se elas possuem imagem de fundo, como os agentes estão posicionados, se há agentes empilhados e se há agentes transparentes. A Figura 4.5 mostra a faceta com as planilhas que existem no projeto *Frogger*. Quando uma planilha é selecionada, são apresentados todos os detalhes com relação a número de classes e instâncias de agentes.

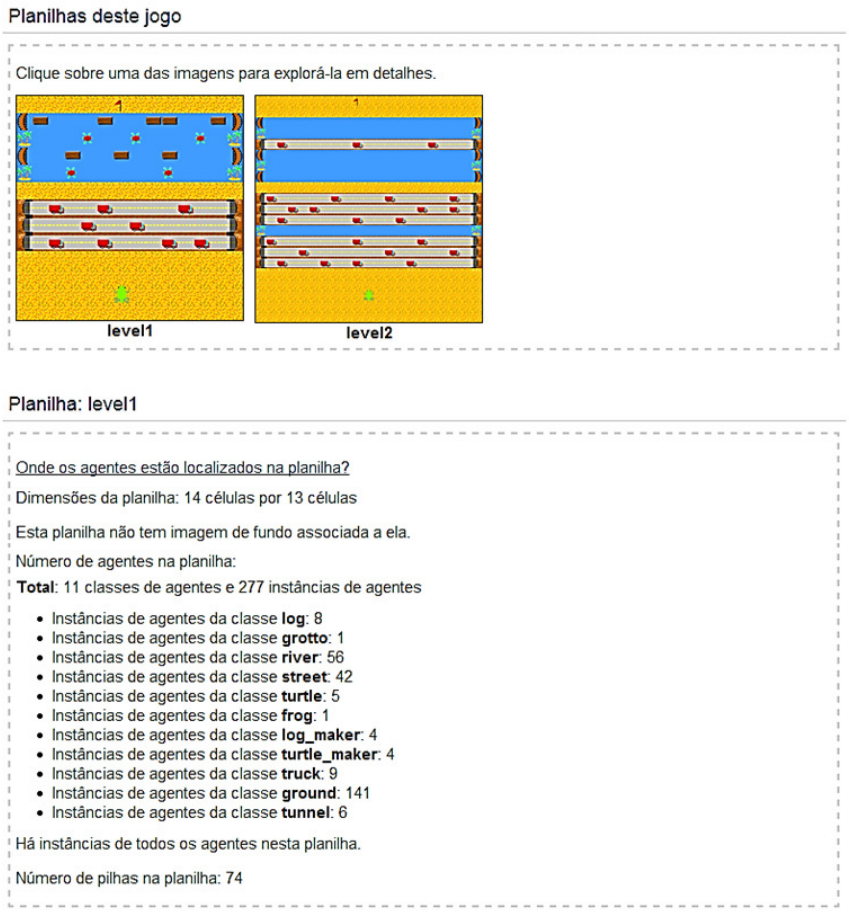

Figura 4.5 - Faceta 'planilhas' do *Frogger*

Esta faceta, também disponibiliza uma grade, onde é possível esconder e mostrar os agentes individualmente, identificando o posicionamento exato de cada

um deles (Figura 4.6) nas planilhas. Note que, na Figura 4.7, os agentes *river* e *ground* foram escondidos (desmarcados na lista da Figura 4.6).

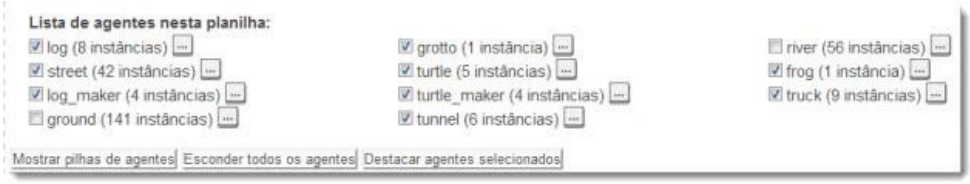

Figura 4.6 - Opções de visualização da planilha

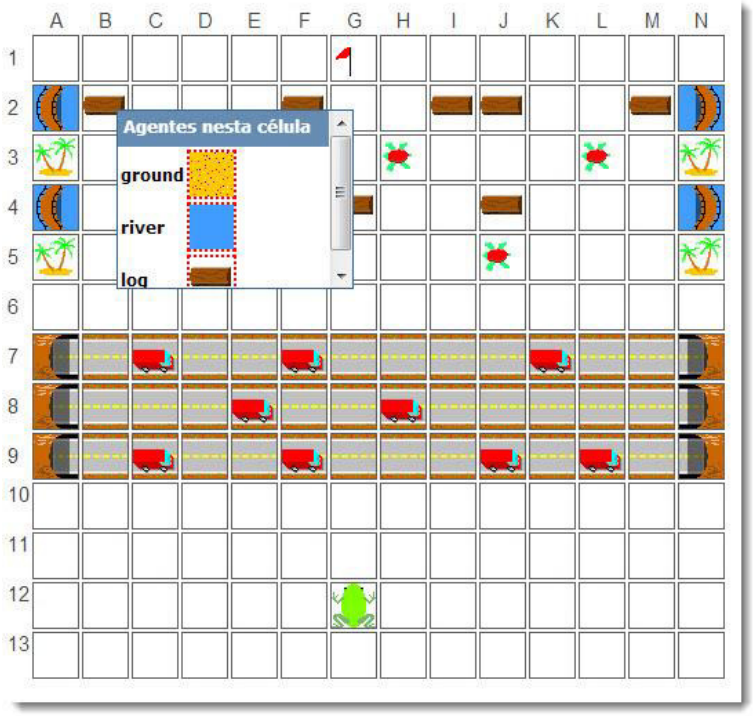

Figura 4.7 - Grade com os agentes *river*, *ground* e *street* escondidos

Na interface do AS, não é fácil (e nem sempre possível) saber se os agentes estão empilhados um por cima do outro. Dependendo do modo como a planilha foi construída, o empilhamento pode se dar por razões distintas: pode ser parte do *design* do projeto; devido a agentes esquecidos sobrepostos; ou podem ser pilhas geradas pelo comportamento de outros agentes de forma indesejada. Essa visualização reforça a lembrança de que se não há uma função para esses agentes no projeto, eles deveriam ser removidos, pois podem interferir negativamente no comportamento de outros agentes ou, simplesmente, causar um desperdício de recursos computacionais se houver muitos agentes "inúteis" escondidos ou espalhados pela planilha. Em alguns casos, o excesso de agentes sem função pode até levar ao travamento do programa. Na grade, podemos ver os agentes

empilhados em cada célula. Ainda na Figura 4.7, ao clicar no agente *log* (célula B2), notamos que há outros agentes naquela célula, na ordem: *ground*, *river* e *log*. Como escondemos os agentes *river* e *ground*, nesta célula, estamos vendo somente o *log*.

Os exemplos apresentados nesta subseção sobre a faceta planilha destacam o valor da documentação ativa que propomos. O aprendiz interage com os signos da interface para explorar a planilha dos jogos e simulações, compreendendo detalhes que não estavam disponíveis em outras visualizações. Essa interação como uma 'conversa' com sistema é o que caracteriza a documentação ativa.

No modelo

Perspectiva da interação

Consideramos que a metacomunicação nesta faceta é essencialmente através de signos dinâmicos porque a maior parte dos significados emerge da interação, no decorrer do tempo, com os elementos da interface, especialmente na grade da faceta 'planilhas'. Além disso, o foco está em todos os elementos do programa porque a planilha é inteiramente reapresentada fazendo referência ao projeto como um todo.

Perspectiva do usuário

Há mudanças nas dimensões de expressão e intenção da perspectiva do usuário. Todas as informações disponibilizadas existem no AS, mas elas não estão bem comunicadas. Na faceta, elas são reapresentadas em um novo formato para permitir o destaque de possíveis problemas, criando assim uma paráfrase (ou seja, a comunicação do mesmo conteúdo, mas com outra forma).

Perspectiva do sistema

Da perspectiva do sistema, há mudanças nas dimensões léxica, sintática e semântica. Há novo vocabulário e nova gramática, pois a faceta apresenta conceitos não discutidos explicitamente no AS como classes, instâncias e pilhas. Para explicitar esse tipo de conceito foi preciso elaborar novos algoritmos para processar os dados existentes e gerar novas informações, por isso consideramos que houve mudança na dimensão semântica, caracterizando a reescrita na perspectiva do sistema.

4.1.4. Faceta ‘regras’

Esta faceta apresenta as regras de comportamento de cada agente em linguagem pseudonatural e em linguagem de código do AS. Na Figura 4.8 está a tradução das regras do agente *frog* para o português. A numeração das regras indica a ordem em que cada uma delas é verificada durante a execução do jogo ou simulação.

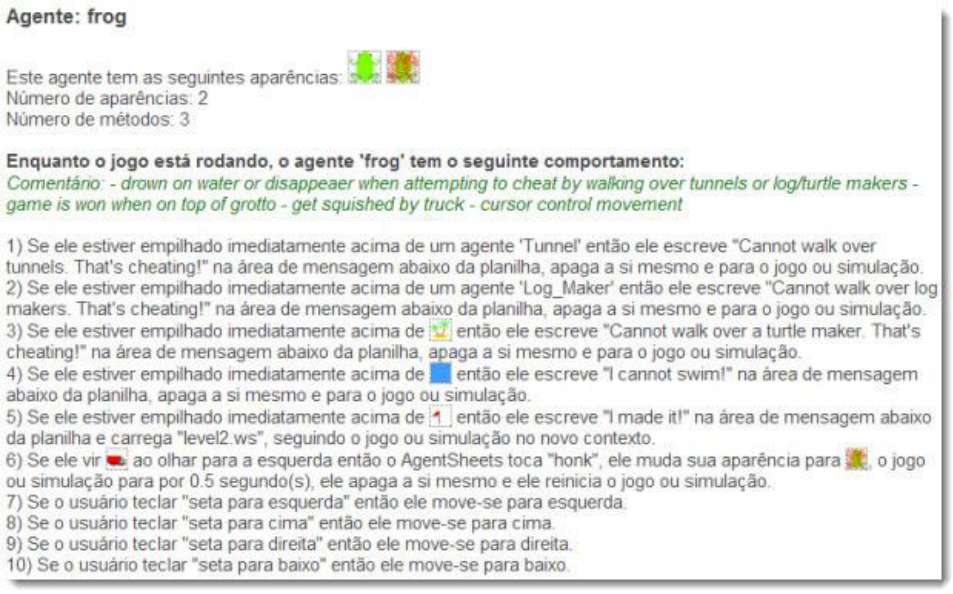

Figura 4.8 - Regras de comportamento do *Frogger*

As regras de comportamento do agente *turtle_maker*, nas duas opções de representações, podem ser vistas na Figura 4.9. O texto da representação em linguagem pseudonatural ainda é apenas uma tradução literal das regras, mas colhemos indícios de como o texto já pode ajudar aprendizes a entenderem mais sobre o funcionamento dos seus projetos (subseção 3.2.3). A linguagem de código do AS é bastante icônica (refere-se diretamente aos comandos e imagens do AS), pode ser vista como um ponto de mediação entre as regras construídas através da linguagem visual do AS e o texto em linguagem pseudonatural do PoliFacets-AS.

Figura 4.9 - Linguagem pseudonatural e Linguagem de código do AS

No modelo

Perspectiva da interação

O foco das verbalizações é apenas em um agente, de modo que a tradução das regras de cada agente é exibida separadamente no contexto local. A faceta é classificada como um signo estático porque é formada por textos e imagens, cuja interpretação pode ser atingida desconsiderando-se desdobramentos de relações temporais ou causais ao longo da interação.

Perspectiva do usuário

Há mudanças apenas na expressão e na intenção. Já que nenhuma informação adicional é fornecida, a faceta é uma paráfrase de um conteúdo que existe no AS, mas está representado de modo diferente, textualmente.

Perspectiva do sistema

As mudanças estão na dimensão léxica e sintática, com mudança no vocabulário e na gramática da linguagem. Desse modo, a faceta é uma reformulação, pois os elementos foram renomeados e reordenados.

4.1.5. Faceta 'tags'

A faceta 'tags' exibe uma nuvem de *tags* com os comandos utilizados no projeto. É uma representação que mostra quais comandos são mais utilizados ou menos utilizados. O tamanho da expressão que representa o comando corresponde à quantidade de vezes que o comando foi utilizado no projeto. Veja os exemplos a

seguir, a Figura 4.10 é a faceta ‘tags’ do projeto *Frogger* e a Figura 4.11 mostra a faceta ‘tags’ do projeto *Pacman*.

Figura 4.10 - *Tags* do *Frogger*

Figura 4.11 - *Tags* do *Pacman*

Note que no *Frogger*, os comandos “ver” e “uma vez a cada” são utilizados várias vezes, isso mostra como a verificação de proximidade entre os agentes e a questão temporal são importantes. Enquanto isso, no *Pacman*, o comando “mudar” é muito utilizado, já que o agente *Pacman* muda de aparência sempre que se move em direções diferentes.

No modelo

Perspectiva da interação

Os comandos ilustrados representam o projeto completo, então a faceta foca em todos os elementos do programa. Esta faceta é classificada como um signo estático, pois a maior parte da mensagem do *designer* é percebida sem interação e em um único instante de tempo.

Perspectiva do usuário

Há mudanças no conteúdo que foi exibido e na intenção, pois a faceta é uma nuvem de *tags* com a intenção de destacar os comandos e quantidade de utilização

deles no projeto. O tamanho dos comandos aumenta conforme o número de vezes que ele aparece no programa, fazendo uma analogia entre o tamanho e a quantidade. Além disso, podemos pensar no comando como uma parte do programa (a parte pelo todo) que constituiria uma figura de linguagem (metonímia).

Perspectiva do sistema

Há uma modificação do vocabulário que foi reduzido, pois, nesse caso, apenas os comandos aparecem. Também há alteração na estrutura sintática para compor a nova organização dos elementos, o que caracteriza uma reformulação.

Ligação

A faceta 'tags' está conectada a faceta 'regras' porque é possível acessar as regras dos agentes que usam os comandos destacados. Por exemplo, no jogo *Frogger*, se clicarmos em "ver" uma lista de agentes que usam esse comando será apresentada e nós poderemos ir até as regras de determinado agente selecionado.

4.1.6. Faceta 'conexões'

A faceta 'conexões' exibe um conjunto de diagramas de dependências com um diagrama para cada agente do projeto. A ideia é mostrar que o agente principal (em vermelho) se relaciona com outros agentes (em preto) de forma direta (linhas verdes) ou indireta (linhas cinzas), como pode ser visto na Figura 4.12, que apresenta as conexões do agente *frog* do jogo *Frogger* (Figura 4.4).

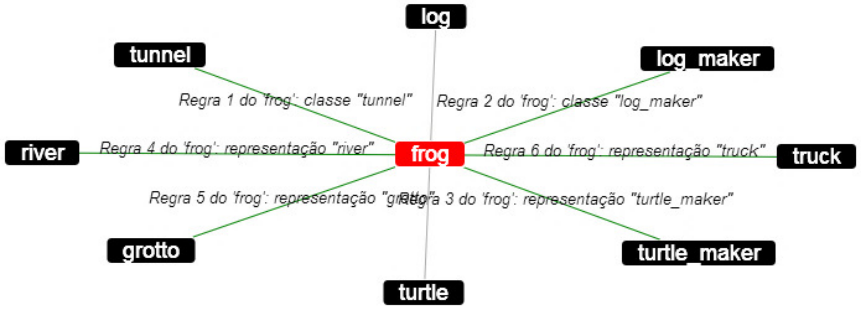

Figura 4.12 - Diagrama de conexões do agente *frog*

Por enquanto, estão sendo consideradas as relações que envolvem os comandos: *ver*, *ver um*, *empilhado*, *empilhado em um*, *próximo a*, *novo*, *anunciar* e *em cima de*. Além dos comandos, identificamos se a relação acontece pela

representação (aparência) ou pela classe do agente. Também é possível observar as relações que ocorrem através do compartilhamento de propriedades ou atributos do projeto, pois quando uma variável local ou global é consultada ou alterada por dois agentes, isso significa que eles se relacionam.

Quanto mais regras estabelecem relações entre os agentes, mais espessa é a aresta que liga esses agentes no diagrama de conexões. Observe a Figura 4.13, em que duas regras ligam o agente *log* ao agente *river*.

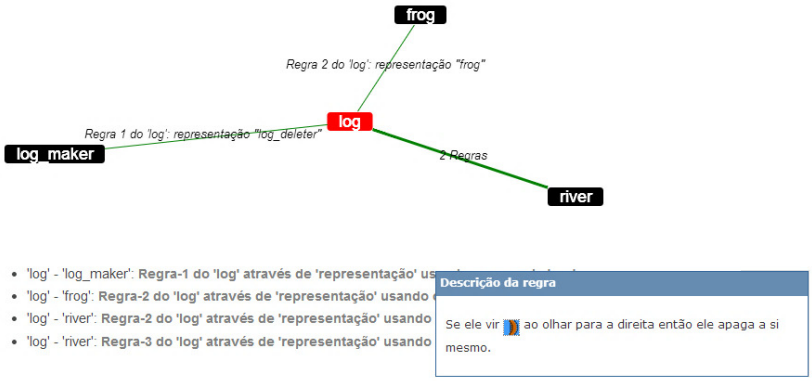

Figura 4.13 - Diagrama de conexões do agente 'log'

O diagrama de conexões, além de explicitar relações entre os agentes, mostra como elas acontecem. É possível clicar na descrição da relação para visualizar a regra do comportamento do agente traduzida para o texto em linguagem pseudonatural (Figura 4.13). Este texto, que especifica como ocorre a relação, é o mesmo presente na faceta 'regras', mas exibindo apenas a regra que é o foco da relação.

Os diagramas de conexões entre os agentes são importantes porque apoiam a exploração dos limites de influência que um agente tem sobre o comportamento de outros agentes. Usando essa faceta, os estudantes e professores têm mais opções para "dividir e conquistar", no que diz respeito à compreensão da complexidade lógica dos jogos e simulações. Além disso, a significação e comunicação das relações entre os agentes na estrutura do programa são essenciais para o domínio da complexidade cognitiva nas tarefas de programação. No AS, há uma funcionalidade de importação de agentes, esse recurso só poderá ser bem utilizado caso o aprendiz tenha consciência sobre os agentes que se relacionam ao agente importado, seja para também importá-los ou para realizar as

modificações necessárias para o funcionamento do projeto de acordo com o esperado.

No modelo

Perspectiva da interação

Esta faceta é um signo estático que explora a relação entre os agentes. Isto significa que estamos tratando da visão regional do projeto através de signos essencialmente estáticos, com o foco em um conjunto de elementos do programa.

Perspectiva do usuário

A faceta é uma paráfrase, pois há mudanças na expressão e na intenção. A forma como as relações foram representadas é totalmente distinta do que ocorre do AS e o objetivo é ressaltar as dependências existentes entre os agentes.

Perspectiva do sistema

Podemos dizer que a informação apresentada nos diagramas de conexões existe nos projetos do AS, o problema é a grande dificuldade de localizar e agregar o conteúdo que resulte em conclusões corretas sobre o modo como os agentes estão conectados. Desse modo, foi preciso criar um algoritmo que organize as relações entre os agentes que consideramos importantes para criar essa faceta, ou seja, foi preciso realizar mudanças léxicas, sintáticas e semânticas, caracterizando operação simbólica de reescrita.

Ligação

Há uma ligação entre a faceta 'conexões' e a faceta 'regras' porque é possível acessar as regras de comportamento que estabelecem as conexões entre os agentes.

4.2. Facetas idealizadas e suas ligações

Neste momento, vamos partir da grade da ontologia de metacomunicação (Tabela 4.1) e projetar, conceitualmente, algumas facetas para demonstrar a aplicação do modelo PoliFacets em uma possível expansão do PoliFacets-AS (Tabela 4.2). As facetas de A até E estão no contexto das facetas existentes e as facetas de F a J são ideias sobre outros contextos.

Tabela 4.2 - Grade da ontologia de metacomunicação no PoliFacets-AS com facetas idealizadas

			Metacomunicação através de Signos ESTÁTICOS (principalmente)			**Metacomunicação através de Signos DINÂMICOS (principalmente)**		
	Operações Semióticas @ Usuário	**Operações Simbólicas @ Sistema**	LOCAL foca um elemento de programa	REGIONAL foca um conjunto de elementos de programa	GLOBAL foca todos os elementos de programa	LOCAL foca um elemento de programa	REGIONAL foca um conjunto de elementos de programa	GLOBAL foca todos os elementos de programa
Operações sobre o sistema de significação do OBJETO do Documento Ativo	RESSIGNIFICAÇÃO dá novos significados aos signos do programa	-						Na prática
	PARÁFRASE expressa os signos do programa de maneira alternativa	REFORMULAÇÃO sem alteração semântica	Regras	*Faceta G*			*Faceta H*	
		REESCRITA com alteração semântica		Conexões *Faceta D*				Planilhas
	FIGURAS DE LINGUAGEM faz combinações inéditas com os signos do programa e revela novos significados	REFORMULAÇÃO sem alteração semântica	*Faceta E*		Tags			
		REESCRITA com alteração semântica						
Outro Sistema de Significação	EXPANSÃO LINGUÍSTICA dá novos significados aos signos do programa	REFORMULAÇÃO sem alteração semântica	*Faceta B*	*Faceta A*	Descrição			*Faceta C*
		REESCRITA com alteração semântica	*Faceta F*	*Faceta I*			*Faceta J*	

As possibilidades de facetas idealizadas são ilimitadas, é preciso equilibrar os níveis de complexidade para ajudar no processo de ensino e aprendizado. Algumas facetas requerem um conhecimento maior para que o conteúdo seja bem absorvido; por isso, os momentos seguintes de *design* e desenvolvimento das interfaces das facetas podem ser críticos.

4.2.1. Faceta A

Considerando a faceta 'descrição', suponha a situação em que fosse possível fornecer descrições e instruções sobre conteúdos locais do projeto considerando apenas um agente.

No modelo

Perspectiva da interação

Esta faceta seria classificada um signo estático com o escopo local. O foco da descrição seria em apenas um elemento do programa.

Perspectiva do usuário

Assim como a faceta 'descrição', a faceta A seria uma expansão linguística. Um texto sobre o programa em outro sistema de significação.

Perspectiva do sistema

A operação simbólica é de reformulação, pois a descrição e instrução não produzem alterações semânticas do ponto de vista do sistema.

4.2.2. Faceta B

Ainda considerando a faceta 'descrição', a faceta B é similar, mas a descrição e a instrução seriam sobre um conjunto de agentes ou sobre relações entre eles.

No modelo

Perspectiva da interação

Esta faceta seria classificada um signo estático com o escopo regional. O foco da descrição seria em um conjunto de elementos do programa.

Perspectiva do usuário

Assim como a faceta 'descrição' e a faceta A, a faceta B seria uma expansão linguística. Um texto sobre elementos do programa em outro sistema de significação.

Perspectiva do sistema

A operação simbólica é de reformulação, pois a descrição e a instrução não produzem alterações semânticas do ponto de vista do sistema.

4.2.3. Faceta C

Na faceta 'na prática', é possível visualizar os agentes disponíveis na "galeria" e editar a planilha, adicionando, movendo ou removendo agentes. Essas opções poderiam ser configuráveis quando o *applet* está sendo carregado. É interessante perceber como o *redesign* da planilha pode modificar a ideia do projeto sem a mudança do comportamento dos agentes. Imagine um projeto com agentes e comportamentos previamente criados e uma planilha vazia. No *applet,* é possível inserir os agentes e posicioná-los de modo diferente, livremente, isso pode alterar a expressão do jogo ou simulação, em algumas vezes, até a mensagem comunicada. Um exemplo dessa ideia está na Figura 4.14, a planilha do jogo *Frogger* modificada em algumas etapas com a mensagem diferente da original.

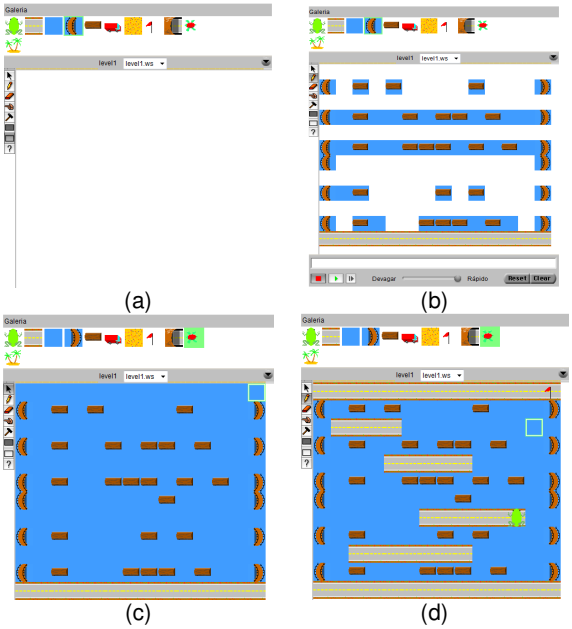

Figura 4.14 - Planilha do jogo *Frogger* modificada

No modelo

Perspectiva da interação

A faceta C é classificada como um signo dinâmico, assim como a faceta 'na prática', com foco em todos os elementos do programa.

Perspectiva do usuário

Como devem ocorrer mudanças na expressão, no conteúdo e na intenção, esta é uma faceta de expansão linguística.

Perspectiva do sistema

Não ocorrerá modificação semântica do ponto de vista do computador, logo esta é uma faceta de reformulação.

4.2.4. Faceta D

Considerando o contexto da faceta 'planilhas', poderíamos criar uma representação com os agentes envolvidos na troca de planilhas. Uma grade que destacasse os agentes que fazem parte da transição seria uma espécie de visão regional. O recurso de troca de planilha é comumente utilizado para simular múltiplas fases em um jogo. Com essa nova faceta, seria possível perceber qual é o fluxo de execução das planilhas e o que faz a transição ocorrer. A Figura 4.15 apresenta uma representação de como poderia ser esta faceta. No *Frogger*, a transição entre as planilhas acontece de 'level 1' para 'level 2' quando o sapo está empilhado em cima da bandeirinha.

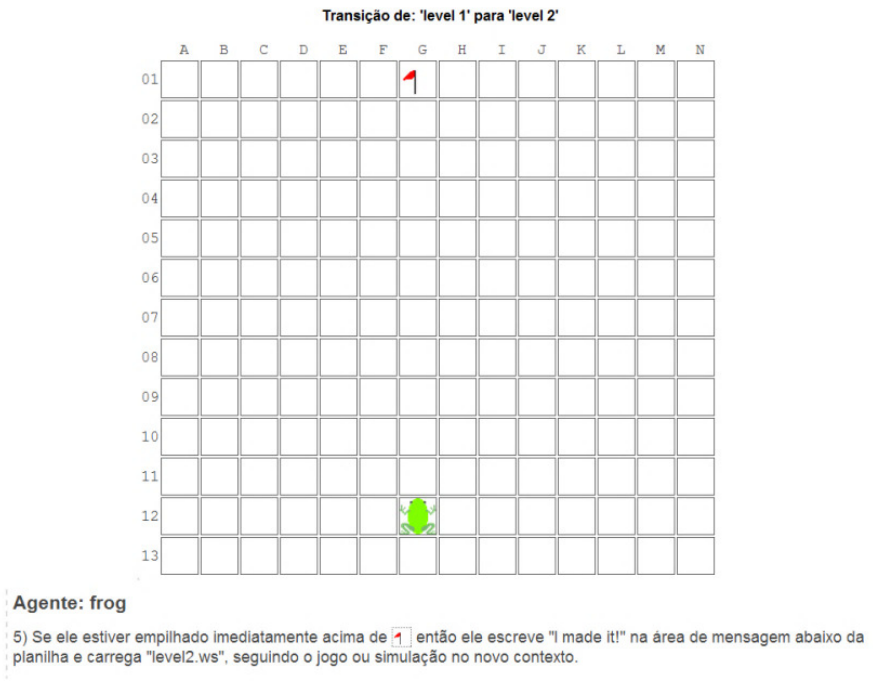

Figura 4.15 - Ilustração da faceta de transição de planilhas

No modelo

Perspectiva da interação

Nesta faceta a metacomunicação acontecerá principalmente através de signos estáticos com o foco em um conjunto de elementos do programa, pois é preciso mostrar como as planilhas estão relacionadas no projeto.

Perspectiva do usuário

A faceta D é uma paráfrase, pois está apresentando informações que estão no AS, mas são difíceis de perceber. Desse modo, ocorrerão mudanças de expressão e intenção.

Perspectiva do sistema

É preciso realizar mudanças semânticas para descobrir como acontecem as relações entre as planilhas, por isso a operação simbólica é de reescrita.

Ligação

Existirá uma ligação idealizada entre a faceta D e a faceta 'regras', pois as regras dos agentes que promovem mudança de planilha devem ser apresentadas.

4.2.5. Faceta E

Em um contexto relacionado à faceta ‘tags’, suponha que fosse criada uma nova faceta com uma nuvem de *tags* para cada agente. Assim, seria possível observar quais comandos são mais ou menos utilizados em cada agente separadamente.

No modelo

Perspectiva da interação

Na faceta E, a metacomunicação ocorreria principalmente através de signos estáticos com o foco em um elemento do programa, considerando uma nuvem de *tags* para cada agente.

Perspectiva do usuário

Assim como a faceta ‘tags’, esta faceta seria uma figura de linguagem. A analogia entre o tamanho da expressão e a quantidade de usos dos comandos foi mantida.

Perspectiva do sistema

Não ocorreriam mudanças semânticas, somente sintáticas e léxicas, o que caracteriza esta faceta como uma reformulação do ponto de vista do sistema.

Ligação

Assim como a faceta ‘tags’, a faceta E estaria ligada a faceta ‘regras’. Desse modo seria possível acessar os agentes as regras que usam os comandos destacados.

4.2.6. Faceta F

A faceta F seria uma representação que explicasse a ordenação de regras de um agente. Neste caso, é preciso explicar como a máquina de execução do AS funciona realmente, o modo como as regras de produção são avaliadas nos determinados instantes no tempo. Por isso, o sistema de significação precisa ser diferente.

No modelo

Perspectiva da interação

Na faceta F, a metacomunicação ocorreria principalmente através de signos estáticos e o foco estaria em apenas um elemento do programa, considerando um agente de cada vez.

Perspectiva do usuário

Devem ocorrer mudanças em todas as dimensões semióticas, caracterizando uma faceta de expansão linguística, especialmente pela necessidade de mudar o sistema de significação.

Perspectiva do sistema

Ao explicar como funciona a máquina de execução do AS de acordo com a ordem das regras no agente, seria preciso realizar alterações semânticas do ponto de vista do computador, a operação simbólica seria de reescrita.

4.2.7. Faceta G

A faceta G deveria apresentar novos comandos que causassem um mesmo efeito de jogo ou simulação já programado de outra forma pelo aprendiz. Como acontece no exemplo ilustrado no capítulo de Introdução (p. 21-25), em que um mesmo efeito pode ser programado de várias formas.

No modelo

Perspectiva da interação

A metacomunicação nesta faceta ocorreria principalmente através de signos estáticos com o foco em um conjunto de elementos do programa.

Perspectiva do usuário

O resultado da execução do projeto seria o mesmo, mas os comandos utilizados seriam diferentes, seria como dizer o mesmo de outra forma, caracterizando uma paráfrase.

Perspectiva do sistema

Não ocorreriam mudanças semânticas porque os comandos apresentados que promovessem o mesmo efeito não seriam executados pelo sistema. Logo, a operação simbólica desta faceta é a reformulação.

4.2.8. Faceta H

A faceta H seria uma demonstração do uso de novos comandos de modo dinâmico com parte da execução do projeto, seria possível visualizar como os efeitos são os mesmos de acordo com cada regra. A ideia é dizer que usar os comandos de um jeito ou de outro funciona de uma maneira similar na prática.

No modelo

Perspectiva da interação

Para mostrar de modo dinâmico o comportamento de um agente usando diferentes comandos, é preciso demonstrar também os agentes que estão relacionados com ele, por isso o escopo dessa faceta seria regional. Por isso, a metacomunicação aconteceria principalmente através de signos dinâmicos com foco em um conjunto de elementos do programa.

Perspectiva do usuário

Assim como a faceta H, esta faceta é uma paráfrase, pois demonstra um mesmo conteúdo com diferentes expressões.

Perspectiva do sistema

Não ocorrem alterações semânticas do ponto de vista do computador, esta é uma faceta de reformulação.

Ligação

Haveria uma ligação entre a faceta G e a faceta H, note que elas são, em certo aspecto, complementares do ponto de vista estático e dinâmico.

4.2.9. Faceta I

A faceta I deveria explicar otimização de código. Seria interessante mostrar quais comportamentos entre os agentes de um programa poderiam economizar recursos através da otimização do código.

No modelo

Perspectiva da interação

A metacomunicação nesta faceta aconteceria principalmente através de signos estáticos com o foco em um conjunto de elementos do programa.

Perspectiva do usuário

Para mostrar conceitos de otimização, o sistema de significação seria diferente do programa, apenas agregando informações à sua ferramenta de construção. A operação semiótica desta faceta é a expansão linguística.

Perspectiva do sistema

É preciso explicar noções mais detalhadas de algoritmos para que os aprendizes tenham consciência sobre que tipo de otimização vale o esforço. Neste caso, a operação simbólica é a reescrita.

4.2.10. Faceta J

Seguindo o mesmo raciocínio da faceta anterior, a faceta J seria a demonstração do efeito de otimização de código com uma representação dinâmica.

No modelo

Perspectiva da interação

O que diferencia esta faceta da faceta I é que a metacomunicação ocorreria principalmente através de signos dinâmicos. O foco ainda seria mantido em um conjunto de elementos do programa.

Perspectiva do usuário

Além de mostrar o que é otimização, seria possível perceber por que, quando e como otimizar um código. Novamente, é preciso expandir o sistema de significação para construir uma faceta desse tipo, caracterizando uma expansão linguística.

Perspectiva do sistema

Assim como na faceta I, este é um caso onde a operação simbólica é a reescrita do ponto de vista do sistema.

Ligação

A faceta J deve conter uma ligação com a faceta I para facilitar o acesso à mesma informação através de signos dinâmicos e estáticos.

4.3. Avaliação

Esta seção apresenta a Fase 5 desta tese, uma avaliação do modelo PoliFacets. Primeiramente, relatamos um estudo sobre a recepção das facetas realizado com alunos participantes do projeto SGD-Br. Em seguida, descrevemos a aplicação do modelo PoliFacets no contexto de outra ferramenta de ensino de programação.

4.3.1. Recepção das facetas

Objetivo

O objetivo inicial deste estudo, realizado por um membro da equipe de pesquisa do projeto, foi entender como os alunos manifestam a autoexpressão quando estão se comunicando através de diálogos e *scripts*, usando sistemas de mediação para navegar no PoliFacets-AS. No contexto desta tese, utilizamos os dados coletados (Monteiro, 2014) para observar resultados que também podem ser lidos como evidências da recepção da mensagem de metacomunicação do PoliFacets-AS por parte dos alunos. O modo como eles explicam o jogo usando as facetas reflete o modo como eles compreenderam os significados das facetas.

Procedimento para coleta de dados e investigação

O estudo foi realizado durante as semanas finais de um curso de programação na escola C, em que os alunos usaram o AS para construir seus jogos. Durante as aulas, o PoliFacets-AS foi utilizado como forma de consulta de material didático. Além disso, os alunos precisavam entregar os projetos finalizados por meio do PoliFacets-AS, fazendo *upload* dos jogos no sistema. A tarefa foi proposta como descrita no Quadro 4.1.

Este estudo também fez uso do SideTalk como ferramenta de coleta de dados. Como nenhum dos participantes conhecia a ferramenta, foi feita uma breve apresentação mostrando como utilizá-la com um exemplo de como um aluno mostraria seu jogo ao professor navegando pelas páginas do PoliFacets-AS.

Quadro 4.1 - Cenário do estudo

O professor não poderá estar com você na última aula. Mas, o time do SGD-Br sugeriu ao professor que usasse o SideTalk para se comunicar com seus alunos. O SideTalk é uma ferramenta que apoia conversas sobre páginas na *Web* através de diálogos especiais. Agora, sua tarefa é contar ao professor o que você tem feito no AgentSheets. Por favor, escolha um dos seus projetos e crie uma conversa usando o SideTalk para mostrar algumas páginas interessantes relacionadas ao seu jogo no PoliFacets-AS. Por favor, inclua na sua conversa as seguintes facetas: 'na prática', 'descrição' e uma terceira faceta da sua escolha. Importante: Você pode escrever em português, inglês ou espanhol.

Em seguida, foi pedido aos participantes que pensassem sobre quais páginas gostariam de navegar e sobre qual projeto gostariam de criar a apresentação para o professor. Foi sugerido que eles anotassem separadamente essas decisões, nessa fase de planejamento. Depois, eles tiveram que gravar o *script* e criar os diálogos no SideTalk. Por último, eles responderam a um questionário com questões abertas sobre a experiência de criar os diálogos e sobre o perfil (Anexo - Questionário do estudo 'Recepção das facetas').

Perfil dos participantes

Os participantes faziam parte de uma turma de 8º ano do ensino fundamental e tinham em média 12 anos. Seis alunos da turma se voluntariaram a participar especificamente deste estudo. Cada participante tinha enviado de 3 a 5 projetos para o PoliFacets-AS.

Principais resultados

É interessante notar como os participantes interpretaram a mensagem que transmitimos através das facetas. Podemos considerar que a ideia geral das facetas foi bem recebida. Mas há muita informação para ser explorada, grande parte dela pode não estar sendo absorvida. Como neste estudo, pedimos que os participantes apenas apresentassem o projeto ao professor, pode ser que eles tenham escolhido não apresentar todos os detalhes. Seria o caso de elaborar cenários diferenciados onde pudéssemos avaliar como são recebidas as mensagens que contêm informações mais complexas, como por exemplo, na grade da faceta 'planilhas'. Ainda na questão da complexidade, notamos que nenhum dos participantes usou a

faceta ‘conexões’, mas seria preciso um estudo mais aprofundado para considerar os possíveis motivos que podem estar ligados à complexidade da faceta ou ainda ao cenário do estudo que solicitava apenas uma faceta adicional além da ‘descrição’ e ‘na prática’, entre outros.

Do ponto de vista do modelo PoliFacets, foi possível observar como os participantes conectaram as facetas em sequência. Este raciocínio reforça a ligação entre as facetas que é descrita no modelo. Além disso, o estudo sinaliza para refletirmos sobre novas ligações que podem ser realizadas na instância para facilitar o acesso às informações no PoliFacets-AS.

Os dados analisados são provenientes dos diálogos elaborados e do questionário final aplicado aos participantes. As categorias que emergiram da análise foram: a sequência de acesso às facetas; indicativos positivos, negativos e surpreendentes da recepção em cada faceta.

Sequência de acesso às facetas

Cada participante precisava usar a faceta ‘descrição’ e ‘na prática’, mas tinha liberdade para escolher outra faceta. E ficaram livres para escolher em que ordem construiriam a navegação durante a conversa. Na Tabela 4.3 está a ordem de visita às facetas de cada participante.

Tabela 4.3 - Facetas e ordem de visualização

	Participantes					
Facetas	P1	P2	P3	P4	P5	P6
Descrição	1	1	1	1	1	1
Na prática	2	3	3	3	2	3
Tags	3		2	2		2
Regras		2			3	
Planilhas			4			

O participante P1 mostrou autocrítica ao dizer que sua sequência não foi muito adequada. P1 disse: “*eu acho que minha sequência não foi a melhor de todas*”. Essa afirmação é válida para confirmar que ele sabe que poderia ter criado uma melhor sequência na exploração do seu jogo.

O participante P2 revelou que escolheu a ordem baseado em um critério de ‘imersão’. P2 explicou: “*eu escolhi começar pela descrição e terminar com a prática porque eu queria que o professor tivesse uma boa imersão no jogo [...] primeira pagina [descrição] = interesse médio, segunda página [regras] = a pior de todas, e a terceira página [na prática] = a mais interessante*”. É relevante

notar como ele classificou a faceta ‘regras’ como a ‘pior’ de todas, pois ela realmente é a mais complexa entre as escolhidas. Ainda assim, esta foi a faceta que ele escolheu incluir, além das duas obrigatórias.

P3 disse que gostou do resultado quando respondeu sobre a sequência em que as facetas foram apresentadas. Ele foi o único participante que visitou quatro facetas e o único a incluir a faceta ‘planilhas’. P3 relatou: “*[...] comecei com a descrição, explicando o jogo, então eu fui para tags onde eu expliquei um pouco sobre os comandos. Depois fui executar o jogo e para a página da planilha*”.

P4 comentou a importância de passar pela descrição antes de executar o jogo. Ele se justificou: “*eu comecei com as páginas desse jeito porque eu penso que a ordem é importante, [a ordem que eu escolhi] diz pra você como jogar, sobre o que é o jogo e deixa você jogar*”. Vale notar que ele se refere à faceta ‘tags’ dizendo sobre o que é o jogo, onde aparecem os comandos do AS usados no projeto. P6 também escolheu a mesma ordem de P4. P6 disse: “*eu escolhi começar pela descrição porque essa é a primeira página que você abre. Eu decidi ir para 'na prática' por último porque eu acho que eles precisam entender o que estão jogando antes de jogar. As tags ficaram no meio”*. Realmente, não é por acaso que a faceta ‘descrição’ foi a primeira página para todos os participantes. Afinal, é a página de abertura das facetas, mas a justificativa adotada por eles é sinal de uma boa recepção, pois eles explicam como é fundamental saber o que é o projeto antes de partir para a execução do jogo ou simulação.

Indicativos positivos, negativos e surpreendentes da recepção

Quanto aos indicativos de sucesso na recepção, a ideia geral da faceta ‘descrição’ foi bem compreendida pelos participantes. P4 explicou nos diálogos relacionados à página de descrição: “*aqui está uma amostra de como o meu jogo se parece (apontando para a miniatura da planilha) / aqui estão as instruções sobre como executar o jogo (apontando para o campo de instruções) aqui está a descrição sobre o que é o jogo, mas eu não coloquei nenhuma porque eu esqueci (apontando para o campo de descrição, que estava vazio)*”. P6 relatou no diálogo: “*essa é a página principal de descrição. Aqui você pode ver a figura do meu jogo e outros detalhes sobre ele*”. Os dois participantes comentam sobre a miniatura do jogo.

Assim como P4, o participante P3 escreveu no diálogo: "*as instruções não estão muito elaboradas, então eu gostaria de explicar isso melhor aqui [...]*. P4 disse que se esqueceu de preencher a descrição e P3 aproveitou a chance para explicar melhor as instruções. Isso pode ser um sinal de um problema de comunicação no PoliFacets-AS, pois eles poderiam editar os campos de instrução e descrição a qualquer momento, mas o acesso a edição dessas informações ainda não está muito claro.

A essência de que a faceta 'descrição' permite expressar a mensagem do criador do jogo foi especialmente compreendida por P6. Ele disse em um dos seus diálogos: "*a descrição é o mais divertido de escrever, em minha opinião, porque você pode ser tão criativo quanto você quiser*".

Quanto à faceta 'tags', o participante P4 teve problemas em entender ou explicar o seu funcionamento corretamente. P4 disse: "*aqui você pode ver os comandos que cada agente tem*". Na verdade, é possível ver os comandos de todo o projeto, mas isso pode estar mal comunicado na interface. Por outro lado, P3 soube explicar detalhadamente a faceta. P3 explicou: "*se você rolar a tela pra baixo estará vendo um diagrama divertido com algumas palavras. Essas palavras são os comandos que eu usei para criar o meu jogo, quanto maior o comando é, mais vezes eu o utilizei. Como você poder ver eu usei 16 comandos apesar de ser um jogo simples (isso está em destaque na cor verde do lado direito)"*. P6, além de explicar o que a faceta representa, destacou detalhes da interação com os comandos. P6 comentou: "*esse link mostra os comandos usados no jogo [...] por exemplo, 'mover' é um dos comandos que eu usei. Ao clicar em 'mover' você pode ver que agentes usaram esse comando*".

Quanto à faceta 'na prática', P4 explica: "*aqui você pode executar o seu jogo. Divirta-se!*". Essa é a mensagem principal. O projeto é executado como forma de *applet* e acontecem muitos problemas com avisos de segurança e execução de *applet Java* nos navegadores. Apesar de passarmos por esse tipo de situação com certa frequência, não há explicação sobre isso na faceta. P2 disse: "*Aqui, se você clicar, vai abrir em outra página. O Java vai perguntar se você quer executar o programa, clique no [box] que diz que você aceita os riscos, clique no botão executar e espere carregar. Quando terminar, execute o jogo.*" Essa não é uma informação da faceta, nem do projeto, mas é importante porque sem esses passos o *applet* não vai funcionar e o jogo não será executado.

P3 foi o único a passar pela faceta ‘planilhas’. Ele criou uma conversa que acessava os detalhes das duas planilhas do seu projeto falando da quantidade de agentes que usou em cada planilha. P3 disse: “*Agora eu vou mostrar pra você um pouco sobre meus diferentes agentes e como eles são usados na planilha. Primeiramente, eu vou te mostrar a primeira planilha. Olhe! Observe quantos agentes são usados nessa planilha! Veja que na próxima eu usei ainda mais agentes*”. Há muitas outras mensagens comunicadas na faceta ‘planilhas’, inclusive a grade, mas P3 chamou a atenção apenas para o número de agentes.

Dois participantes navegaram e criaram diálogos sobre a faceta ‘regras’. Em poucas palavras, P5 resumiu: “*o próximo link vai simplesmente mostrar que regras eu usei para fazer o meu jogo*”. Com mais detalhes, P2 falou: “*se você quer ver mais informações sobre as minhas regras e como eu fiz os agentes se moverem do jeito que eles se movem [...] esse link leva você ao lugar que você pode encontrar as regras e todos os meus agentes*”. Para completar, P2 destaca como a faceta ‘regras’ explicita problemas ocorridos durante a criação do projeto. P2 justificou: “*aqui você pode ver as regras e como eu editei as representações. Algumas vezes eu acidentalmente pressionei o botão ‘novo agente’ ao invés do botão ‘nova representação’, então existem muitos agentes que eu não usei no meu jogo e que não têm nenhuma regra*”. Esse problema também é visto no AS, mas ao apresentar para o professor no PoliFacets-AS, P2 justificou a causa. Na faceta, o excesso de agentes esquecidos fica mais claro porque é exibida uma lista com os agentes, representações e comportamentos.

Os participantes estavam livres para escolher uma faceta além da ‘descrição’ e ‘na prática’, nós perguntamos o motivo da escolha e as respostas reforçam o modo como eles entendem a mensagem das facetas. Sobre a escolha das regras, P2 explicou: “*eu escolhi incluir as regras porque às vezes os professores querem saber como você fez tudo. Então eles podem ver que você não trapaceou. Como o professor está viajando, essa é uma página que ele deveria ver*”. P6 justificou o uso das *tags* dizendo: “*eu escolhi usar a página tags porque eu acho que fica claro ver quem usou quais comandos*”. O participante P4 revelou que escolheu as três facetas por achar que elas são essenciais para compreender o jogo. P4 revelou: “*eu escolhi ‘na prática’, comandos, descrição e instrução porque elas são o que você precisa para saber como executar o jogo*”. Com outra visão, P5

determinou as páginas mais importantes. P5 disse: "*escolhi as regras, 'na prática', descrição e tags porque eu acho que são as mais importantes*".

Notamos que todos os participantes que passaram pela faceta 'tags', se referiram a ela como 'comandos'. No *design* inicial dessa faceta, havia uma justificativa para que ela tivesse esse nome mais genérico, pois seria possível adicionar outras tags nos projetos, mas atualmente ela é formada somente por comandos e comunica apenas essa mensagem.

4.3.2. Validação dos resultados

Objetivo

O objetivo deste estudo foi realizar uma prova de conceito, aplicando o modelo proposto para o *design* da metacomunicação de documentos ativos para o ensino e aprendizado de programação em uma nova ferramenta.

Procedimento para coleta de dados e investigação

Para esta validação qualitativa da pesquisa, realizamos três etapas: aplicamos o modelo considerando a linguagem NCL (Nested Context Language) e a ferramenta NCL Composer; construímos um protótipo não funcional da interface que representa o documento ativo resultante da aplicação do modelo; e coletamos a opinião de um especialista em NCL.

A NCL é uma linguagem declarativa, uma aplicação XML que tem como base o modelo Nested Context Model (NCM) idealizada por Soares *et al.* (2005). Um documento em NCL apenas define como objetos de mídia são estruturados e relacionados, no tempo e no espaço (Soares e Barbosa, 2012). Para construir um documento hipermídia é preciso responder quatro perguntas: O que? Onde? Como? e Quando?. Deve ser definido o que vai ser executado (que mídia vai "tocar"), em que região da tela e em qual dispositivo, como as propriedades das mídias estarão configuradas, e quando cada mídia será apresentada, seja com determinado tempo ou condição.

NCL Composer

O NCL Composer é uma ferramenta de autoria criada para auxiliar na criação de aplicações para a TV Digital Interativa (TVDI) em NCL. É desenvolvida pelo Laboratório TeleMídia do Departamento de Informática da

PUC-Rio. A ferramenta NCL Composer foi escolhida para ser utilizada nesta prova de conceito porque é consistente com a questão de autoexpressão através de *software*. Além disso, o NCL Composer representa o programa construído através de visões que são diferentes perspectivas do documento hipermídia. Atualmente, ele é composto pelas visões: textual, estrutural e de *layout*.

A visão textual (Figura 4.16) apresenta o código NCL que pode ser editado normalmente como em qualquer outro editor de texto. Ao criar o código e mudar de visão, o NCL Composer faz a sincronização necessária para que as mudanças feitas no código sejam refletidas nas outras visões. Um processo similar ocorre quando mudanças são feitas nas outras visões: o código NCL é atualizado na visão textual.

```
<ncl id="prepPassiveDevicesEx" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
    <head>
        <regionBase id="regionBase1">
            <region height="100.00%" id="screenReg" left="0.00%" top="0.00%" width="100.00%" zIndex="1">
                <region height="18.50%" id="frameReg" left="5.00%" top="6.70%" width="18.50%" zIndex="2">
                </region>
            </region>
            <region height="40.00%" id="passiveAdvReg" left="5.00%" top="5.00%" width="40.00%" zIndex="2">
            </region>
        </regionBase>
        <descriptorBase id="descriptorBase1">
            <descriptor focusBorderWidth="0" focusIndex="5" id="passiveAdvDesc" region="passiveAdvReg">
                <descriptorParam name="transparency" value="50%">
                </descriptorParam>
            </descriptor>
            <descriptor id="screenDesc" region="screenReg">
            </descriptor>
            <descriptor explicitDur="3s" id="photoDesc" region="frameReg">
            </descriptor>
            <descriptor id="audioDesc">
            </descriptor>
            <descriptor id="dribleDesc" region="frameReg">
            </descriptor>
        </descriptorBase>
        <connectorBase id="connectorBase1">
            <importBase alias="conEx" documentURI="defaultConnBase.ncl">
            </importBase>
        </connectorBase>
    </head>
```

Figura 4.16 - Visão textual no NCL Composer

A visão estrutural apresenta as mídias, suas conexões e contextos. É possível criar novos itens e editar suas propriedades. A Figura 4.17 mostra um exemplo da visão estrutural com os objetos que estão sendo utilizados e uma noção da lógica de execução sobre a ordem das mídias envolvidas no documento.

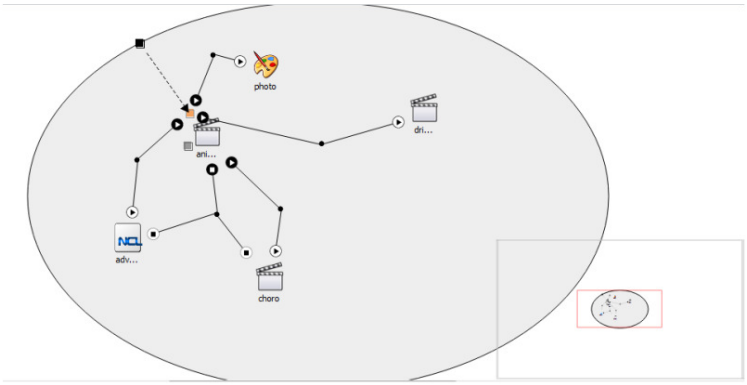

Figura 4.17 - Visão estrutural no NCL Composer

A visão de *layout* responde sobre onde as mídias do documento deverão ser exibidas. Na Figura 4.18 estão definidas as regiões da tela existentes no documento: *screenReg* e *passiveAdvReg*. As propriedades das regiões são definidas separadamente e não há uma visualização direta na visão.

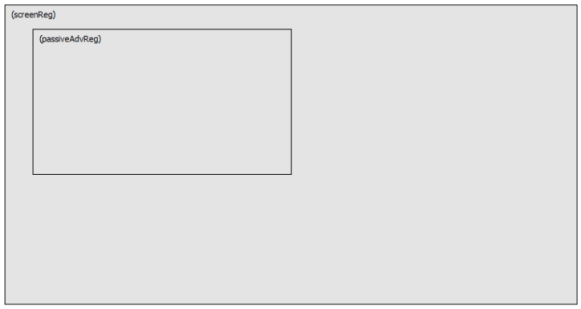

Figura 4.18 - Visão de *layout* no NCL Composer

PoliFacets-NCL

Vamos considerar os conteúdos dessas visões como o conjunto de códigos que forma o sistema de significação do modelo PoliFacets-NCL (Figura 4.19). A partir dessa entrada, foi feita uma análise inspecionando os elementos que resultou na ontologia de domínio referente a questões da linguagem NCL e da ferramenta NCL Composer. Assim, aplicamos a ontologia de metacomunicação para construir as novas facetas e suas ligações que constituem um documento ativo voltado para o processo de ensino e aprendizado de NCL.

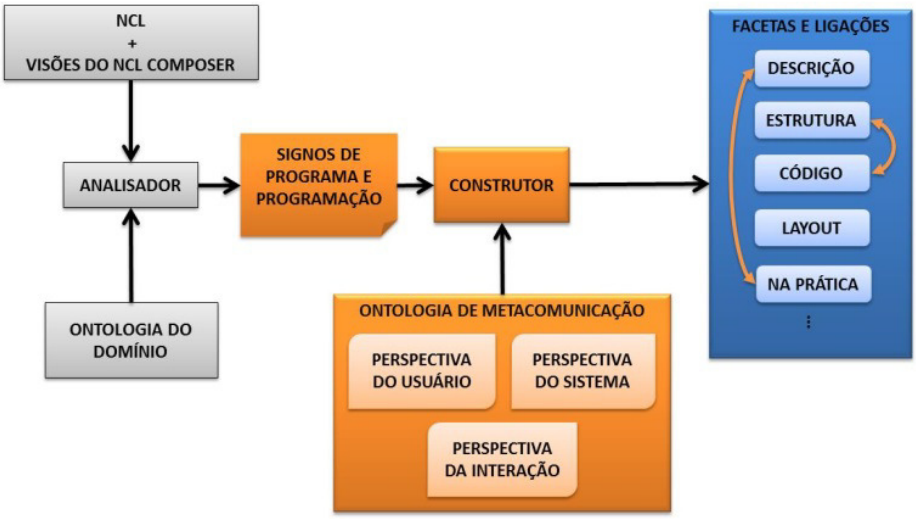

Figura 4.19 - Modelo PoliFacets-NCL

A grade da ontologia de metacomunicação presente na Tabela 4.4 foi utilizada para ajudar a refletir sobre as mudanças que poderiam ser feitas nas representações existentes para apoiar o ensino de NCL. Foram definidas cinco facetas para o PoliFacets-NCL: ‘descrição’, ‘código’, ‘estrutura’, ‘*layout*’ e ‘na prática’. As ligações entre as facetas estão entre: ‘descrição’ e ‘na prática’; e ‘estrutura’ e ‘código’.

Tabela 4.4 - Grade da ontologia de metacomunicação no PoliFacets-NCL

			Metacomunicação através de Signos ESTÁTICOS (principalmente)			**Metacomunicação através de Signos DINÂMICOS (principalmente)**		
	Operações Semióticas @ Usuário	**Operações Simbólicas @ Sistema**	LOCAL foca um elemento de programa	REGIONAL foca um conjunto de elementos de programa	GLOBAL foca todos os elementos de programa	LOCAL foca um elemento de programa	REGIONAL foca um conjunto de elementos de programa	GLOBAL foca todos os elementos de programa
Operações sobre o sistema de significação do OBJETO do Documento Ativo	RESSIGNIFICAÇÃO dá novos significados aos signos do programa	-						Na prática
	PARÁFRASE expressa os signos do programa de maneira alternativa	REFORMULAÇÃO sem alteração semântica	Código		Estrutura			
		REESCRITA com alteração semântica						*Layout*
	FIGURAS DE LINGUAGEM faz combinações inéditas com os signos do programa e revela novos significados	REFORMULAÇÃO sem alteração semântica						
		REESCRITA com alteração semântica						
Outro Sistema de Significação	EXPANSÃO LINGUÍSTICA dá novos significados aos signos do programa	REFORMULAÇÃO sem alteração semântica			Descrição			
		REESCRITA com alteração semântica						

Faceta ‘descrição’

A faceta ‘descrição’ (Figura 4.20) contém informações sobre o programa fornecidas pelo seu criador. Novamente, a faceta ‘descrição’ é importante para permitir que o criador do programa possa fornecer em linguagem natural a sua mensagem diretamente aos seus receptores.

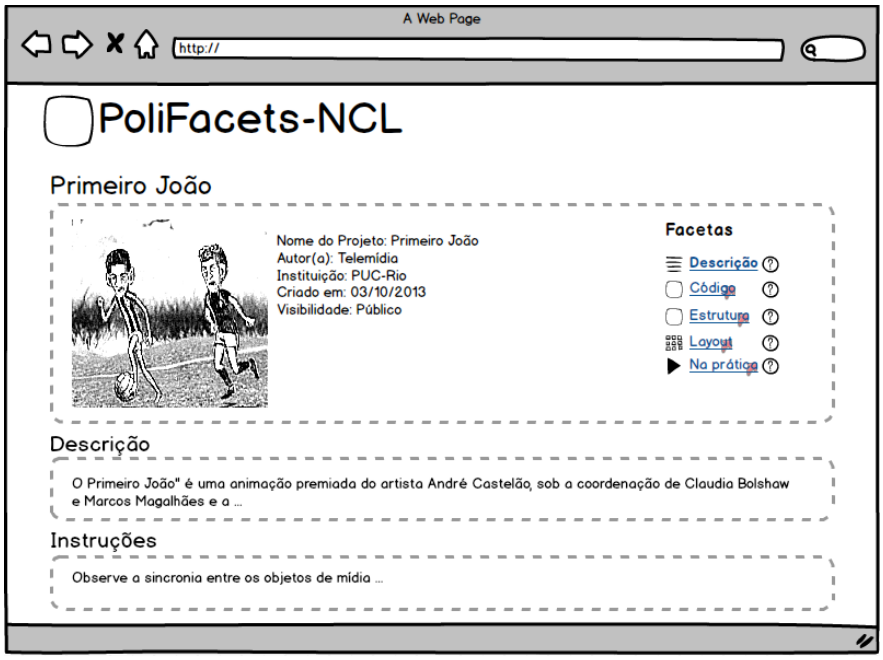

Figura 4.20 - Faceta ‘descrição’ no PoliFacets-NCL

No modelo

Perspectiva da interação

A ideia é usar a faceta para descrever o programa como um todo, logo, a metacomunicação ocorre principalmente através de signos estáticos e o foco está em todos os elementos do programa.

Perspectiva do usuário

A oportunidade de descrever o programa e definir instruções de utilização é uma expansão linguística. Há mudanças na expressão, no conteúdo e na intenção.

Perspectiva do sistema

Esta é uma faceta de reformulação, somente há alteração no vocabulário (léxico) e na gramática (sintático).

Faceta ‘código’

A faceta ‘código’ (Figura 4.21) contém a organização dos componentes da linguagem e o código que faz referência a cada componente. Esse código pode ser apresentado em linguagem pseudonatural e em NCL.

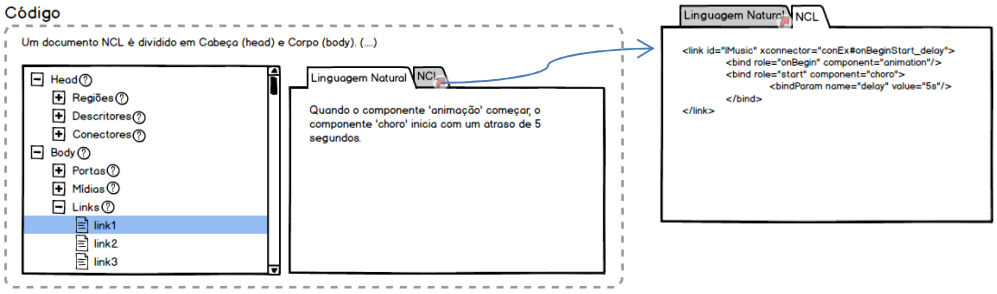

Figura 4.21 - Faceta 'código' no PoliFacets-NCL

No modelo

Perspectiva da interação

A faceta é um signo estático com foco em um elemento do programa porque apresenta o conteúdo textual relativo aos elementos do código NCL de modo particionado, elemento a elemento.

Perspectiva do usuário

O código em NCL é uma das entradas no modelo, mas ele é reapresentado particionado como foco no elemento selecionado. Há modificações na expressão e na intenção, pois a mesma informação é transmitida de outro modo com a intenção de esclarecer o código, caracterizando uma paráfrase.

Perspectiva do sistema

Há modificações léxicas e sintáticas por causa da tradução do conteúdo em NCL para linguagem pseudonatural. Esta é uma faceta de reformulação.

Faceta 'estrutura'

Além de apresentar as mídias e suas ligações, a faceta 'estrutura' deve facilitar o acesso ao conteúdo que define as ligações. Veja na Figura 4.22 como os links que definem as condições para a execução das mídias podem ser acessados. Ao clicar em um dos links, seria possível verificar a tradução do código em linguagem pseudonatural que define a ligação entre as mídias.

Essa faceta é uma paráfrase com reformulação e a metacomunicação ocorre através de signos estáticos com foco em todos os elementos do programa.

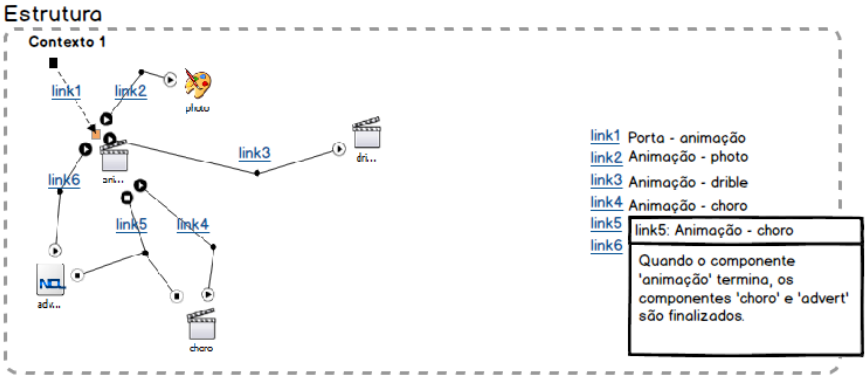

Figura 4.22 - Faceta 'estrutura' no PoliFacets-NCL

No modelo

Perspectiva da interação

Na faceta 'estrutura' a metacomunicação acontece principalmente através de signos estáticos e foco está em todos os elementos do programa.

Perspectiva do usuário

Há mudanças na expressão e na intenção, pois as informações estão sendo reapresentados de uma nova forma com a oportunidade de destacar o conteúdo das ligações. Esta faceta é uma paráfrase.

Perspectiva do sistema

Apenas o vocabulário e a gramática precisaram ser modificados. Não há alterações semânticas, portanto, esta faceta é constituída de uma operação simbólica de reformulação.

Ligação

Há uma ligação entre esta faceta e a faceta 'código' para viabilizar o acesso ao código de modo localizado para visualização do trecho correspondente a cada link apresentado.

Faceta '*layout*'

A visão de *layout* no NCL Composer mostra as regiões onde as mídias são executadas. Na faceta '*layout*', apresentamos as regiões com a visualização das mídias em cada uma delas de acordo um determinado momento. Essa representação é possível quando as mídias são executadas com condições que fazem referência ao tempo na execução das mídias, mas não funciona quando as condições são relacionadas à interação com o usuário final do documento hipermídia. Por exemplo, na Figura 4.23 o vídeo da animação (representado pela cor amarela) é executado na região maior. O vídeo do drible, a foto da queda e a

foto da chuteira são mídias executadas na região menor em verde. Suponha que 10 segundos depois de o vídeo da animação começar, inicie o vídeo do drible, cada um em sua região. Depois que o vídeo do drible terminar, a foto da chuteira será exibida, e por fim, a foto da queda. Nesse caso, é possível determinar algoritmicamente os momentos em que as mídias são exibidas em cada região.

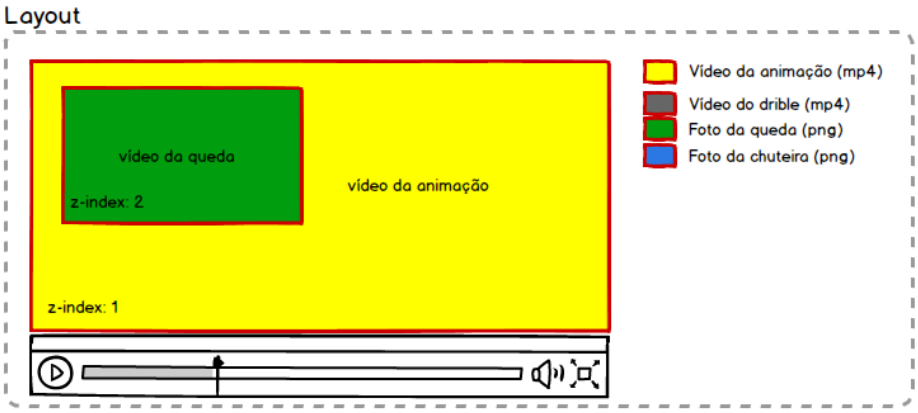

Figura 4.23 - Faceta '*layout*' no PoliFacets-NCL

Imagine que a condição para a execução de uma mídia fosse uma ação interativa, por exemplo, quando o usuário digitar A, então o vídeo X será executado, mas se o usuário digitar B, então o vídeo Z será executado. Nesses casos, a representação não é simples, pois cada caminho é uma espécie de "modelo de interação" do programa que poderia gerar uma representação de *layout* diferente.

No modelo

Perspectiva da interação

A metacomunicação nesta faceta ocorre principalmente através de signos dinâmicos com o foco em todos os elementos do programa.

Perspectiva do usuário

Esta é uma reapresentação das informações em novo formato, portanto, uma paráfrase.

Perspectiva do sistema

Esta faceta é uma paráfrase com reescrita e a metacomunicação ocorre principalmente através de signos dinâmicos com o foco em todos os elementos do programa.

Faceta ‘na prática’

Esta é a faceta onde o programa pode ser executado. É o produto resultante da programação desenvolvida no NCL Composer. Na Figura 4.24, está um exemplo da faceta ‘na prática’ no PoliFacets-NCL.

Figura 4.24 - Faceta ‘na prática’ no PoliFacets-NCL

No modelo

Perspectiva da interação

Por definição, a execução do programa representa um signo dinâmico global. Assim, a metacomunicação acontece principalmente através de signos dinâmicos com foco em todos os elementos do programa.

Perspectiva do usuário

Diferente da representação a partir do NCL Composer, onde ainda estão sendo realizados testes e a visualização funciona como verificação da correção do programa, na faceta ‘na prática’, a intenção é demonstrar o resultado final do programa. Isto caracteriza uma ressignificação.

Perspectiva do sistema

Ressignificações não apresentam operações simbólicas correspondentes.

Ligação

A faceta ‘na prática’ está ligada a faceta ‘descrição’ porque reapresenta o conteúdo da descrição e instruções sobre o programa.

Perfil do participante

Um participante foi escolhido para a avaliação do protótipo. Ele é um especialista, aluno de doutorado em informática, membro do laboratório

TeleMídia, que conhece NCL desde a graduação e esteve envolvido no desenvolvimento do NCL Composer. Ele tinha utilizado o PoliFacets-AS algumas vezes.

Principais resultados

Depois da apresentação do protótipo preliminar PoliFacets-NCL ao especialista, realizamos uma entrevista com questões abertas para obter as impressões dele. Os resultados dessa validação apontam para a viabilidade da aplicação do modelo em uma nova ferramenta de construção de programas no contexto de ensino e aprendizado de programação.

Foram construídos protótipos de facetas que são extensões e novas representações das visões que estão presentes no NCL Composer com base no modelo PoliFacets. Nós recebemos uma boa avaliação do especialista sobre os protótipos das facetas resultantes. Ele não só reconheceu a importância que as facetas tinham apresentando características da linguagem NCL que não estavam explícitas no NCL Composer, como sugeriu novas funcionalidades capturando a ideia de extensões propostas pelas facetas e propondo novas aplicações para as facetas imaginadas. Tecnicamente, o participante destacou que uma grande vantagem de aplicar o modelo PoliFacets ao NCL Composer é a sua característica de extensibilidade quanto a criação de *plug-ins*, onde cada nova faceta pode ser desenvolvida por voluntários.

As categorias que emergiram da análise da entrevista com o especialista foram duas: a compreensão da ideia das facetas para o NCL Composer; e possibilidades de aplicações do PoliFacets-NCL.

Compreensão da ideia das facetas

O especialista demonstrou a compreensão sobre a criação das facetas desde o primeiro contato com o protótipo. Ele fez algumas sugestões que foram rapidamente incorporadas ao *mockup* do PoliFacets-NCL usado na validação. Por exemplo, na faceta '*layout*' já aparecem soluções propostas pelo ele. Primeiramente, estávamos apenas definindo que mídias apareceriam em que regiões. Ele reagiu da seguinte forma: "*[...] uma coisa que fica legal pro [PoliFacets-NCL] [...]: você vai dizer 'aqui está a região onde será apresentada a mídia'. Por que ao invés de dizer o nome da mídia, eu não desenho a mídia ali? Isso é uma coisa que a gente já sabe como fazer [...] é bem tranquilo de ir pra*

ferramenta [PoliFacets-NCL]. Acho bem legal; você só vai ter que tratar um caso, porque não é uma região por objeto de mídia. Uma região tem vários objetos de mídia [...]". Depois desse comentário, aperfeiçoamos a faceta que passou a ter o componente visual da mídia além da referência ao nome (Figura 4.23). Neste mesmo contexto, a adição de elementos na interface que possibilitassem visualizar e esconder cada região e objeto de mídia seria interessante. Fazendo uma analogia, seria similar ao que acontece com os agentes empilhados na grade da faceta 'planilhas' no PoliFacets-AS.

O problema dos vários objetos de mídia na mesma região ressalta uma questão importante: o tempo. Versões antigas do NCL Composer continham uma visão temporal. Mas o especialista explicou porque a visão foi removida: "*A gente sente necessidade da visão temporal, mas a gente prefere estabilizar antes [...]. É complicado porque a visão temporal não foi feita pra representar documentos hipermídia, a visão temporal é uma coisa linear, um documento hipermídia por definição não é linear. Então a gente achou melhor estabilizar as visões que tínhamos antes de pensar nesse problema*".

Em sua opinião geral, o especialista comentou: "*eu gostei, é bem a cara das visões que a gente tem no NCL Composer. [...] na de layout, como eu comentei, a gente pode colocar as mídias direto. Na estrutural, tem como a gente gerar os links. Do código NCL, eu gostei dessa coisa de dividir, não entrar numa de tentar mostrar o texto todo ainda, a menos que tivesse a visão temporal [...]*". Ele já imagina como as funcionalidades que a equipe de desenvolvimento do NCL composer está implementando poderiam ajudar na criação das facetas. Ele explicou: *"uma coisa que eu estou querendo com o futuro da estrutural é implementar um algoritmo de desenho de grafo [...], além disso, para gerar [arquivos] png direto, assim como os hiperlinks que podiam te ajudar para o html no [PoliFacets-NCL]*". Ele verbalizou como seria a análise necessária para gerar elementos que integram a ontologia de domínio no modelo PoliFacets.

Aplicações do PoliFacets-NCL

O especialista revela um dos problemas comuns quando os alunos estão começando a estudar NCL. Ele disse: "*[...] eles aninham as regiões sem querer, NCL permite isso, às vezes você quer que tenha um vídeo no fundo e outra região aqui, só que os valores que você usa estão como se essa região daqui estivesse*

relativa à tela inteira, mas na verdade sem querer você colocou [essa região] embaixo dessa outra região. Às vezes você coloca o vídeo aqui completo; se no meio da aplicação você quiser redimensionar o vídeo, você vai redimensionar essa região pequena. Então com o [PoliFacets-NCL] dá pra observar isso, tipo o nível de 'aninhamento'". Note como o discurso dele mostra a apropriação da ideia de explorar mudanças nas representações para criar novas facetas.

Sobre a faceta 'código', o especialista descreveu como os textos em linguagem pseudonatural poderiam ser úteis para compor uma suíte de testes para aplicações de TV digital. Ele disse: "*esse texto podia ajudar também no roteiro de testes. Na recepção quando testamos, a pessoa tem que saber 'ah, em tantos segundos, vai aparecer tal coisa'. O que a gente fez em alguns casos de testes: gravamos o áudio no programa. Por exemplo, 'essa aplicação vai começar mostrando isso e daqui a dois segundos vai começar aquilo', você ia conferindo na tela, então se a aplicação fosse boa, a sincronia tinha que estar correta, é como uma metadescrição da aplicação como um todo. No final das contas a visão temporal e a linguagem natural seria mais ou menos esse roteiro para a suíte de testes*".

Segundo o especialista, a primeira versão do NCL Composer foi criada para ajudar os aprendizes que cometiam muitos erros sintáticos durante a programação. Esse problema foi minimizado porque ao usar as visões com elementos gráficos para criar os documentos, o código é automaticamente gerado, evitando esses erros mais comuns. Quanto à segunda versão, ele destacou: "*[...] tem o espirito da anterior que é evitar esses erros e acelerar a parte do desenvolvimento, mas também, permitir a participação da comunidade por meio da adição dos plug-ins. A característica mais forte dele é a extensibilidade. Nada impede que uma faceta seja um plug-in*". Diante da característica de extensibilidade da ferramenta com a participação ativa da comunidade, o modelo proposto nessa tese deve ajudar os desenvolvedores e *designers* a pensar que tipos de facetas ainda podem ser criadas. Esta é uma oportunidade de criar as facetas para explorar os significados mesmo durante a construção das aplicações.