Universidade Federal de Juiz de Fora – UFJF

Instituto de Ciências Exatas – ICE

Departamento de Ciência da Computação – DCC

SIMULAÇÃO DE PROCESSOS DE SOFTWARE BASEADA EM METAMODELOS DE DINÂMICA DE SISTEMAS: ABORDAGEM E EXEMPLO

Vitor Rodrigues de Oliveira

Juiz de Fora

Dezembro, 2010

SIMULAÇÃO DE PROCESSOS DE SOFTWARE BASEADA EM METAMODELOS DE DINÂMICA DE SISTEMAS: ABORDAGEM E EXEMPLO

Vitor Rodrigues de Oliveira

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Paulo Roberto de Castro Villela
Co-orientador: Prof. Me. Igor de Oliveira Knop

JUIZ DE FORA
DEZEMBRO, 2010

SIMULAÇÃO EM PROCESSO DE SOFTWARE BASEADA EM METAMODELOS DE DINÂMICA DE SISTEMAS: ABORDAGEM E EXEMPLO

Vitor Rodrigues de Oliveira

MONOGRAFIA SUBMETIDADA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

_____
Nome, título
(Presidente)

_____
Nome, título

_____
Nome, título

JUIZ DE FORA, MG – BRASIL
DEZEMBRO, 2010

Sumário

Resumo

O estudo de Engenharia de Software é focado em modelos teóricos que descrevem práticas relacionadas ao gerenciamento de Processos de Softwares. Apesar de esses modelos descreverem práticas bem sucedidas, é necessária uma experiência em desenvolvimento para boas tomadas de decisões em desenvolvimentos futuros. Este trabalho explora uma infra-estrutura computacional de simulação de modelos em Dinâmica de Sistemas e Metamodelos de Dinâmica de Sistemas, como ferramenta de modelagem e simulação de processos de softwares, visando um maior entendimento nas interações entre os diversos aspectos envolvidos em um processo de Software. O JynaCoreSim utiliza diagramas de estoque e fluxo e Metamodelos de Dinâmica de Sistemas para simulações em qualquer domínio, e será usado como ferramenta principal neste trabalho, com o objetivo de demonstrar os efeitos de decisões tomadas em processos de softwares no exemplo abordado.

Palavras-chave: Processos de Software, Dinâmica de Sistemas, Metamodelos de Dinâmica de Sistemas.

Abstract

The study of software engineering is focused on theoretical models that describe practices related to managing software processes. Although these models describe successful practices, experience in software development is needed for good in making decisions on future developments. This work explores a computational infrastructure simulation models in system dynamics and Metamodels of system dynamics as a tool for modeling and process simulation software, to better understand the interactions between the various aspects involved in a process of Software. The JynaCoreSim use stock and flow diagrams and system dynamics Metamodels for simulation in any domain, and will be used as a main tool in this work, aiming to demonstrate the effects of decisions taken in cases of software in the example discussed.

Keywords: Software Process, System Dynamics, Metamodels for system dynamics.

1. Introdução

Tomada de decisão consiste em escolher caminhos entre um conjunto de alternativas para realização de tarefas. Em projetos de desenvolvimento de software, tomadas de decisão são mais eficientes quando apoiadas por experiências anteriores. A teoria de engenharia de software apresenta modelos e práticas necessárias para gerenciar todo o processo de criação de um software, porém essas técnicas são demonstradas de forma abstrata, dificultando a visualização do dinamismo existente entre os aspectos relacionados neste processo.

O conceito de dinâmica de sistemas introduzido por Forrester (FORRESTER, 1961) é utilizado para descrever problemas nos mais diversos domínios. Esta técnica utiliza um reduzido conjunto de elementos, chamado diagramas de estoque e fluxo, que descreve um sistema de forma simplificada. Várias abordagens em processo de software utilizando conceitos de dinâmica de sistemas mostram resultados interessantes sobre gerenciamento de atividades em processo de software, como por exemplo, LEHMAN(2001).

Metamodelos de dinâmica de sistemas definem uma abordagem mais flexível dos diagramas de estoque e fluxo, mostrando uma visão mais abstrata do domínio modelado. Os metamodelos utilizam três conceitos básicos: metamodelos de domínio, de instância e de cenários, que estendem a capacidade de representação de um determinado domínio sem perder a simplicidade oferecida pelos diagramas de dinâmica de sistemas.

Com a utilização das ferramentas PowerSim Constructor e JynaCoreSim, será explorado um exemplo ilustrativo sobre tomada de decisão em alocação de recursos em um processo de software. Entende-se por recurso um programador ou engenheiro de software, em ambos os casos podendo ser tratado também apenas como desenvolvedor. Esse exemplo utiliza-se de duas técnicas para decisão sobre alocação de recursos para determinadas tarefas.

A primeira técnica visa determinar, a partir de um conjunto de dados referentes às habilidades envolvidas na produção de um software, o quanto determinado recurso se adapta ao desenvolvimento de uma dada tarefa.

A segunda demonstra, utilizando o resultado da primeira técnica, como seria a evolução da produtividade de um recurso e a quantidade de trabalho realizado sobre um dado tempo, levando em conta uma fórmula de curva de aprendizado.

A ferramenta PowerSim Constructor provê os recursos para a modelagem e simulação de diagramas de dinâmica de sistemas. Alguns recursos adicionais para a modelagem desses diagramas são introduzidos pelo próprio software.

O JynaCoreSim pode ser utilizado para a simulação de modelos de dinâmica de sistemas, bem como metamodelos de dinâmica de sistemas. Será enfatizado neste trabalho, a modelagem e simulação utilizando apenas o recurso de metamodelos dessa ferramenta.

Em um primeiro momento, o metamodelo representará o exemplo como proposto originariamente. Depois uma pequena modificação no exemplo será apresentada usando o recurso de cenários, oferecido pelo conceito de metamodelos de dinâmica de sistemas.

2. Objetivos

Este trabalho visa explorar ferramentas de simulação para apoio em decisões em projetos de desenvolvimento de software. Será proposto um exemplo, utilizando técnicas encontradas na literatura, com o intuito de meramente ilustrar um cenário fictício para a modelagem e simulação nos dois softwares, objetos de estudo deste trabalho.

O exemplo será explorado primeiramente utilizando-se a ferramenta proprietária PowerSim Constructor, em sua versão Lite. Esse software utiliza como base os conceitos de dinâmica de sistemas, elucidado no capítulo 3.1. Além dos elementos básicos propostos por esse conceito, o PowerSim Constructor se utiliza de uma série de recursos próprios para expansão dos modelos.

Em um segundo momento, o exemplo será construído para simulação na ferramenta de código aberto JynaCoreSim. Essa ferramenta se utiliza não só dos conceitos de dinâmica de sistemas, mas também de uma extensão destes, chamado Metadmodelos de dinâmica de sistemas, apresentado no capítulo 3.2.

Buscaremos analisar as vantagens e desvantagens no uso de cada um dos softwares propostos.

3. Fundamentação

Este capítulo visa elucidar os conceitos que serão aplicados tanto na descrição do exemplo como na modelagem e simulação do mesmo.

3.1 Dinâmica de Sistemas

O conceito de Dinâmica de Sistemas foi proposto na década de 50 pelo engenheiro eletricista Jay W. Forrester, então professor da escola de administração do MIT (Massachusets Institute of Technology). Com a publicação do livro “Industrial Dynamics” (Dinâmica Industrial), em 1961, Forrester apresentou uma ferramenta poderosa para avaliar as complexas interações sociais e econômicas. Em 1968, o conceito mostrou sua real utilidade na publicação de “Urban Dynamics” (Dinâmica Urbana) e “World Dynamics” (Dinâmica Mundial).

Um sistema é um conjunto de elementos que se relacionam entre si, formando um todo organizado e coerente. Cada um desses elementos é chamado de variável e possui características próprias.

Assimilar todas as variáveis de um sistema e visualizar as suas interações pode ser uma tarefa complicada sem a utilização de uma técnica mais elaborada. Diagramas de Dinâmica de Sistemas vêm para facilitar essa tarefa, utilizando uma linguagem gráfica para expressar um sistema e seu comportamento. Essa linguagem foi proposta de forma simples, com poucos, porém suficientes, recursos, tornando a descrição de qualquer sistema uma tarefa mais intuitiva.

A seguir, serão apresentados os dois tipos de diagramas utilizados em Dinâmica de Sistemas, os Diagramas Causais que expressam qualitativamente as variáveis de um sistema e suas interações, e os Diagramas de Estoque e Fluxo, que abordam uma visão quantitativa dos aspectos envolvidos em um sistema.

3.1.1 Diagramas Causais

Os Diagramas Causais demonstram as influências entre variáveis de um sistema. Essas influências podem ser positivas ou negativas, sendo positivas quando o aumento da quantidade de uma variável acarreta no aumento da quantidade de outra, e negativas quando ocorre o oposto.

Setas são usadas para representar uma influência de uma variável em outra, e os sinais de mais ou menos para indicar se é essa influência é positiva ou negativa.

A Figura 1 demonstra a interação entre algumas variáveis envolvidas em um Processo de Desenvolvimento de Software.

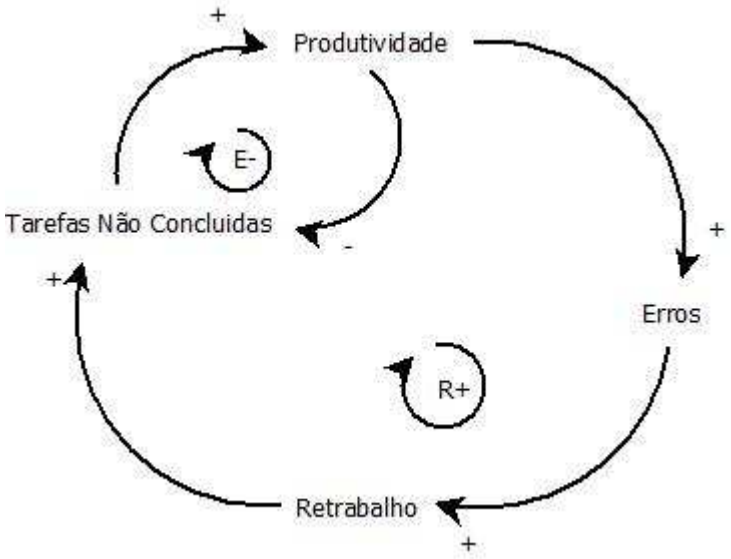

Figura 1. Diagrama causal.

Podem ocorrer sistemas com ciclos fechados de relações, como exemplo a Figura 1. Quando isso ocorre, uma variável possui uma influência indireta para com si mesma. Essa influência pode ser negativa ou positiva. Em um ciclo fechado onde ocorre um número par de relações negativas, essa auto-influência será positiva e esse sistema será chamado de sistema de realimentação. Por outro lado, se o ciclo apresentar um número ímpar de relações negativas, a auto-influência será negativa e esse sistema será chamado de sistema de equilíbrio.

Em alguns casos, a influências podem ter efeitos atrasados. Quando isso ocorre, a alteração em uma variável levará algum tempo até afetar outra variável, não sendo instantânea como nos outros casos. A Figura 2 demonstra um exemplo onde

Figura 2. Diagrama Causal com atraso.

3.1.2 Diagramas de Estoque e Fluxo

Os Diagramas de Estoque e Fluxo são representados por cinco elementos básicos utilizados para descrever o comportamento de um sistema. Esses cinco elementos são descritos a seguir e exemplificados na Figura 3.

Variáveis: São representados por um círculo. São valores ou expressões que representam parâmetros do sistema. Quando esses valores são constantes, são representados por um losango.

Estoques: Expressam o estado do sistema. Possuem um valor inicial que irão variar de acordo com os fluxos, sejam de entrada ou de saída, através do tempo. Os estoques são representados por um quadrado.

Fluxo: Expressam a vazão de quantidades de um estoque para outro. Pode conter um valor fixo ou uma expressão. São representados por uma seta com um círculo no meio. O sentido da seta representa o sentido do fluxo.

Fonte externa: Expressam estoques onde seus valores não são de importância no escopo do sistema avaliado, sendo considerados de capacidade inesgotável. São representados por uma nuvem.

Informação: São as relações entre os componentes do sistema. Se o valor de um componente influi no valor de outro, essa influência deve ser demonstrada com setas, que são as informações.

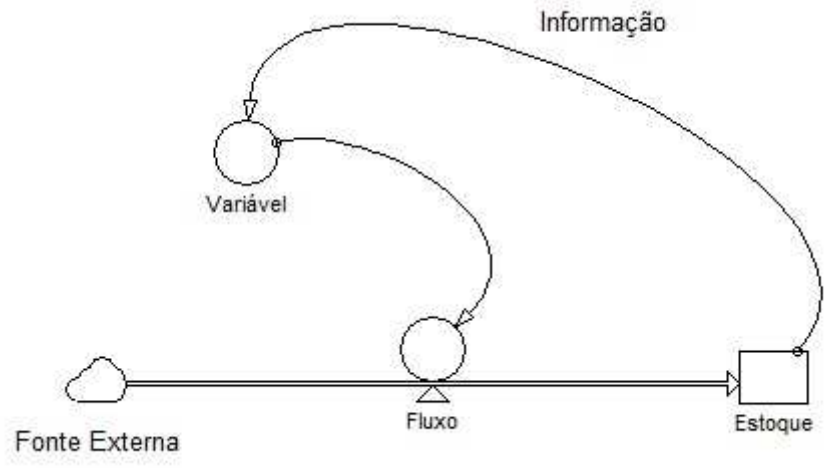

Figura 3. Diagrama de Estoque e Fluxo.

3.2 Metamodelos de Dinâmica de Sistemas

Metamodelos de Dinâmica de Sistemas é uma linguagem proposta por Barros (BARROS, 2001) para estender os Diagramas de Estoque e Fluxo. Essa linguagem tem como objetivo aumentar o controle da complexidade crescente de um modelo. Ela aumenta o nível de abstração dos construtores, deixando-os mais próximos do domínio modelado.

Os metamodelos podem ser traduzidos para diagramas de estoque e fluxo através de um compilador de metamodelos.

3.2.1 Modelos de Domínio

Os Modelos de Domínio descrevem os conceitos principais, através de classes e relacionamentos, um domínio de conhecimento, sendo uma descrição genérica, porém podendo se especificar para cada problema.

Classes são utilizadas para descrever um conjunto de elementos com comportamentos e propriedades similares. Uma classe é a descrição de como uma estrutura é, ou seja, a classe é apenas a abstração do conceito modelado e ela não pode ser simulada. Para simulação, exigi-se a utilização das técnicas de diagramas de estoque e fluxo.

As classes possuem elementos chamados propriedades, que representam valores relevantes que as instâncias deverão ter. Essas propriedades possuem valores padrões que podem ser alterados de acordo com o comportamento das instâncias.

Classes possuem comportamentos, que emergem da estrutura descrita pelos elementos de dinâmica de sistemas. Um comportamento representa a dinâmica da classe.

Um estoque é análogo ao estoque finito de um diagrama de estoque e fluxo. Representa um valor que alimenta ou é alimentado por um fluxo. Diferentemente dos diagramas de estoque e fluxo, um estoque infinito não é representado, sendo implícito na falta de um estoque em alguma das conexões de um fluxo.

Uma taxa é o fluxo entre dois estoques. Representa a vazão de quantidades em relação ao tempo entre um estoque e outro.

Um auxiliar é utilizado para destacar valores e expressões importantes. Esses valores ou expressões são calculados em relação a propriedades, estoques, fluxos ou outros auxiliares.

Uma tabela é uma lista de valores associados a um intervalo contínuo, representado um auxiliar que retorna valores discretos.

3.2.2 Instâncias de Modelo

Para representar sistemas reais, são criadas instâncias a partir das descrições de um modelo de domínio. Cada instância é única e possui um valor identificador único, assim como valores definidos para cada uma de suas propriedades.

Uma instância serve como uma implementação do modelo do domínio, modelando um problema mais próximo do mundo real, permitindo a manipulação e entendimento do problema estudado.

3.2.3 Modelos de Cenário

Cenários são elementos reutilizáveis que descrevem modificações estruturais no comportamento das instâncias. Barros (BARROS, 2001) descreve modelos de cenários no domínio de desenvolvimento de software como políticas, procedimentos, ações e estratégias gerenciais que não podem ser considerados parte de um projeto, mas práticas impostas ou aplicadas sobre o projeto ou situações excepcionais que o gerente pode encontrar ao longo do projeto.

O repositório de cenários agrega as informações sobre comportamentos e eventos que podem ocorrer no sistema. Essas modificações no modelo são utilizadas para ampliar a visão dinâmica do problema, mostrando como o sistema interage perante diversas situações.

3.3 Método Best-Fitted Resource (BFR)

Desenvolvimento de software comumente enfrenta o problema de atrasos em sua agenda. De acordo com Linberg (Linberg, 1999), apenas 16% dos projetos de software estão dentro do seu limite de tempo e orçamento.

Um importante fator para o aumento de resultados em desenvolvimento de software está nas decisões tomadas pelos gerentes de projeto (Otero, Centeno, Ruiz-Torres, Otero, 2008). Com intuito de aumentar a taxa de produtividade, assim diminuir o tempo de desenvolvimento, é importante uma boa escolha na alocação de recursos em um projeto de software.

A alocação de recursos em processo de softwares torna-se complicada à medida que a demanda de profissionais cada vez mais especializados aumenta. Como resultado dessa demanda, fica cada vez mais difícil obter mão-de-obra com exatamente as qualificações especificas de um projeto.

A falta de métodos para quantificar o conjunto de competências de um recurso com conhecimento exigido para desenvolver uma tarefa, força os gerentes de projetos a alocarem esses recursos utilizando métricas subjetivas.

O método Best-Fitted Resource (Otero, Centeno, Ruiz-Torres, Otero, 2008) define uma metodologia para escolha de recursos para um projeto de software, correlacionando o conjunto de conhecimentos de um recurso com o conjunto de habilidades necessárias para realização de uma determinada tarefa.

As habilidades necessárias para desenvolvimento de uma tarefa podem variar de experiência em linguagens de programação, frameworks, sistemas operacionais, equipamentos, etc.

Este método busca aperfeiçoar a alocação de recursos mesmo quando nenhum recurso com as especialidades desejadas está disponível. Ele considera um conjunto de variáveis, quantitativas, que se relacionam e definem como um conjunto de habilidades e o nível de experiência se qualificam para o desenvolvimento de determinadas tarefas.

Fazendo a correlação de níveis de conhecimento necessários para uma determinada tarefa e níveis de experiência nas habilidades de cada recurso, calcula-se qual desses terá um menor tempo de treinamento, devido às suas afinidades.

As variáveis utilizadas no método BFR estão descritas na Tabela 1.

H	Conjunto de todas as habilidades.
H(t)	Conjunto de habilidades necessárias para tarefa *t.*
e_{jt}	Usado esperado da habilidade *j* na tarefa *t*.
c_{jt}	Complexidade da habilidade *j* na tarefa *t*.
s_{jt}	Significância da habilidade *j* na tarefa *t*.
r_{jk}	Relação entre nível de conhecimento na habilidade *j* e o nível de conhecimento na habilidade *k*.
l_{yj}	Nível de conhecimento do recurso *y* da habilidade *j*.
b_{yj}	Relação entre todas as habilidades do recurso *y* e a habilidade *j*.
f_{yt}	Adaptação do recurso *y* à tarefa *t*.

Tabela 1. Conjunto de variáveis utilizadas no método BFR.

O método BFR é descrito em quatro passos, cada um com o objetivo de desenvolver uma tabela de informações. Essas informações têm o intuito de estabelecer quão melhor é escolha de um recurso *y* para uma dada tarefa *t*. A seguir serão detalhados esses quatro passos.

3.3.1 Tabela de Habilidades Necessárias Para a Tarefa

Este passo tem como objetivo definir o conjunto de habilidades necessárias para se desenvolver uma dada tarefa (H(t)). O nível de conhecimento necessário em dada habilidade é especificado em relação à (e_{jt}) e (c_{jt}).

A significância (s_{jt}) de uma habilidade específica *j* em relação a uma tarefa *t* é igual ao produto do uso esperado (e_{jt}) e da complexidade (c_{jt}).

Os valores de e_{jt} variam entre 0 a 1, sendo 0 quando a habilidade *j* não será utilizada na tarefa *t* e 1 quando terá um uso extensivo. A complexidade varia da mesma forma, sendo próximo de 0 quando seu uso for simples e 1 quando possuir uma complexidade desafiante.

Considere o seguinte exemplo: uma tarefa será desenvolvida utilizando a linguagem de programação Java. Como toda a sua programação será feita nesta linguagem, seu nível de uso esperado (e_{jt}) será igual a 1. Apesar disso, a tarefa não exigirá recursos muito complexos da linguagem, atribuindo assim um valor de 0,2 para sua complexidade (c_{jt}). Logo, a significância (s) da habilidade *j* Java para a tarefa *t* será:

$S_{jt} = 1 \times 0,2 = 0,2$

Isso significa que um desenvolvedor precisa apenas um conhecimento básico em Java para desenvolver a tarefa.

3.3.2 Tabela de Relacionamento entre Habilidades

Certo conhecimento em algumas habilidades pode interferir no tempo de aprendizado de outras. O conhecimento em uma habilidade com muitas características semelhantes à outra diminui o tempo de aprendizagem dessas.

A tabela de relacionamento entre habilidades busca elucidar como a relação entre diferentes habilidades pode reduzir o tempo de aprendizado.

Há situações onde os recursos possuem pouco ou nenhum conhecimento sobre uma habilidade necessária para realização de tarefas. Quando isso ocorre, é importante estabelecer quais recursos terão maior facilidade em absorver o conhecimento de tal habilidade.

O relacionamento (r_{jk}) de uma habilidade *j* para com uma habilidade *k* define a afinidade entre essas duas habilidades. Os valores das relações variam de 0 a 1, sendo atribuído 0 quando não há relação entre as habilidades, e 1 quando elas são extremamente afins.

A linguagem Java teria uma relação de nível 1 com a linguagem C++, uma vez que suas notações são similares e ambas seguem o mesmo paradigma de programação. A relação da linguagem Java com o sistema operacional Linux seria 0, já que são habilidades completamente dissociadas.

3.3.3 Conjunto de Habilidades dos Recursos

O terceiro passo consiste em descrever a tabela de recursos disponíveis e seus determinados níveis de conhecimento em cada habilidade do conjunto (H). Os valores dos níveis de conhecimento (l_{yj}) variam num intervalo de 0, quando o recurso *j* não possui conhecimento na habilidade *j*, e 1 quando ele a domina.

3.3.4 Tabela BFR

O último passo é determinar, pela tabela BFR, qual recurso é mais desejável para desenvolver determinada tarefa. Esse será o que levar menos tempo de aprendizagem das habilidades requeridas para a tarefa, e logo terá maior produtividade em menos tempo.

Para uma determinada habilidade *j*, há dois fatores que devem ser levados em consideração para o recurso *y*. O primeiro é o nível de conhecimento que *y* possui em relação à habilidade *j*, denotado por (l_{yj}). O outro é o nível de conhecimento do recurso *y* em todas as outras possíveis habilidades e suas relações com a habilidade almejada, obtido pelo produto de (l_{yj}) e (r_{jk}), onde *k* pertence à *H*. A capacidade do recurso na habilidade desejada é definida como:

$$B_{yj} = MAX_{\ h\ e\ H}\ [l_{yj} * r_{jk}]$$

Fórmula 1. Cálculo do relacionamento do recurso *y* com a habilidade

A adequação de cada recurso é determinada pelo somatório dos produtos entre a significância da habilidade *j* para a tarefa *t* e a capacidade do recurso *y* com a habilidade *j*:

$$F_{yt} = \Sigma\, s_{ij} * b_{yj}, \text{ para todo y}$$

Fórmula 2. Calculo do Fitness do recurso *y* para a tarefa *t*

3.4 Curva de Aprendizagem

As curvas de aprendizado, ou curvas de experiência, são utilizadas para analisar a progressão da eficiência de um ser humano através de sua experiência.

Será apresentado um método para estudo dessas curvas de aprendizagem, introduzido por Lawrence (Lawrence, 1996). Esse estudo parte do princípio de que o padrão de melhora segue um padrão previsível.

Considere um exemplo em que um carpinteiro leve duas horas para construir sua primeira cadeira. A segunda cadeira ele levará uma hora e quarenta e oito minutos. A quarta cadeira lhe tomará aproximadamente uma hora e trinta e sete minutos, e assim por diante. A cada vez que ele dobra a quantidade de cadeiras produzidas, por aprimoramento da técnica, ele diminui o tempo em dez por cento. Isso demonstra uma taxa de aprendizagem de noventa por cento.

Para calcular o custo do enésimo produto utiliza-se a seguinte fórmula:

$$K_n = K_m\,(n/m)^b$$

Fórmula 3. Cálculo do custo de produção do enésimo produto.

Onde:

K_n é o custo de produção da unidade n

K_m é o custo de produção da unidade m

m é uma quantidade produzida

n é uma quantidade produzida maior que m

b é Log R / Log 2

R é o fator de aprendizagem

4. Definição do Exemplo

Neste trabalho utilizaremos como objeto de estudo, um exemplo de tomada de decisão sobre alocação recursos apoiado pelos dois conceitos anteriormente apresentados, método BFR e curvas de aprendizado.

Utilizaremos os resultados obtidos com a análise do Best-Fitted Resource para determinar o fator de aprendizagem de cada recurso. Seguindo os passos descritos pelo método BFR, será descrito uma determinada tarefa, com seus determinados requisitos.

Consideremos um caso hipotético onde um cliente deseja um software para um determinado Hardware H. Toda a programação do software será feita utilizando-se as linguagens de programação C++ e Perl.

O Hardware H possui um manual bem detalhado, porém é um hardware complexo, logo atribuiremos um fator de complexidade alto, com o valor de 0,8. Apesar de sua complexidade ser alta, ele será muito pouco utilizado pelo software e será assumido um valor de 0,2 para sua expectativa de uso.

O software em sua maior parte será desenvolvido na linguagem de programação C++, logo seu esperado será determinado como 1,0. Como os recursos exigidos terão uma complexidade desafiadora, o fator de complexidade também será determinado como 1,0.

Alguns recursos do produto serão desenvolvidos na linguagem de programação Perl, e necessitarão de um conhecimento pouco mais aprofundado da linguagem. O seu uso esperado e complexidade serão respectivamente 0,3 e 0,8.

Temos então a seguinte tabela que determina as habilidades necessárias para o desenvolvimento da tarefa com suas respectivas taxas de uso esperado, complexidade e significância:

Habilidades Necessárias	e_{jt}	c_{jt}	s_{jt}
Hardware H	0,2	0,8	0,16
C++	1,0	1,0	1,0
Perl	0,3	0,8	0,24

Tabela 2. Tabela de habilidades necessárias para a tarefa (H(t)).

O próximo passo será determinar a tabela de relacionamento das habilidades.

Consideremos as seguintes habilidades como o conjunto de habilidades relevantes possuídas pelos recursos disponíveis: Hardware H, C++, Java, Perl, VB.

A tabela de relacionamentos entre habilidades com seus respectivos valores (r_{jk}) é descrita a seguir:

	Hardware H	C++	Java	Perl	VB
Hardware H	1,0	0	0	0	0
C++	0	1,0	1,0	0,2	0
Java	0	1,0	1,0	0,2	0
Perl	0	0,2	0,2	1,0	0
VB	0	0	0	0	1,0

Tabela 3. Tabela de Relacionamentos entre Habilidades

O terceiro passo é determinar a tabela com os recursos disponíveis e seus respectivos níveis de conhecimento (l_{yj}) em cada uma das habilidades listadas acima.

Para essa tarefa serão considerados seis desenvolvedores disponíveis:

Recursos	Hardware H	C++	Java	Perl	VB
Desenvolvedor 1	0,0	1,0	0,3	0,3	0,3
Desenvolvedor 2	0,0	0,8	1,0	0,0	0,3
Desenvolvedor 3	1,0	0,3	1,0	0,5	0,2
Desenvolvedor 4	0,0	0,3	0,0	1,0	1,0
Desenvolvedor 5	0,5	0,5	0,8	0,0	0,8
Desenvolvedor 6	0,3	0,8	0,0	0,0	0,5

Tabela 4. Conjunto de Habilidades dos Recursos

O último passo é determinar a tabela BFR e calcular a aptidão de cada desenvolvedor para a tarefa em questão.

A tabela 5 demonstra os valores dos produtos dos níveis de conhecimento (l_{yj}) de cada desenvolvedor com os valores dos relacionamentos (r_{jk}) entre as habilidades, assim como o valor do relacionamento (b_{yj}) do recurso *y* com a habilidade *j*.

	Hardware H	C++	Java	Perl	VB	b
Desenvolvedor 1						
Hardware H	0	0	0	0	0	0
C++	0	1	0.3	0.06	0.06	1
Java	0	1	0.3	0.06	0	1
Perl	0	0.2	0.06	0.3	0	0.3
VB	0	0.2	0	0	0.3	0.3
Desenvolvedor 2						
Hardware H	0	0	0	0	0	0
C++	0	0.8	1	0	0.06	1
Java	0	0.8	1	0	0	1
Perl	0	0.16	0.2	0	0	0.2
VB	0	0.16	0	0	0.3	0.3
Desenvolvedor 3						
Hardware H	1	0	0	0	0	1
C++	0	0.3	1	0.1	0.04	1
Java	0	0.3	1	0.1	0	1
Perl	0	0.06	0.2	0.5	0	0.5
VB	0	0.06	0	0	0.2	0.2
Desenvolvedor 4						
Hardware H	0	0	0	0	0	0
C++	0	0.3	0	0.2	0.2	0.3
Java	0	0.3	0	0.2	0	0.3
Perl	0	0.06	0	1	0	1
VB	0	0.06	0	0	1	1
Desenvolvedor 5						
Hardware H	0.5	0	0	0	0	0.5
C++	0	0.5	0.8	0	0.16	0.8
Java	0	0.5	0.8	0	0	0.8
Perl	0	0.1	0.16	0	0	0.16
VB	0	0.1	0	0	0.8	0.8
Desenvolvedor 6						
Hardware H	0.3	0	0	0	0	0.3
C++	0	0.8	0	0	0.1	0.8
Java	0	0.8	0	0	0	0.8
Perl	0	0.16	0	0	0	0.16
VB	0	0.16	0	0	0.5	0.5

Tabela 5. Tabela BFR

Com todos os valores calculados, resta calcular o Fitness de cada recurso para a dada tarefa, através da fórmula 2.

Desenvolvedor	Fitness
Desenvolvedor 1	1,072
Desenvolvedor 2	1,048
Desenvolvedor 3	1,280
Desenvolvedor 4	0,540
Desenvolvedor 5	0,918
Desenvolvedor 6	0,886

Tabela 6. Fitness de cada desenvolvedor

5. Modelagem e Simulação do Exemplo Proposto

Este capítulo visa apresentar e explicar a construção dos modelos nas ferramentas PowerSim Constructor e JynaCoreSim. Além da modelagem, serão analisados também, os valores obtidos através da simulação dos modelos.

O primeiro tópico deste capítulo trata da modelagem do exemplo utilizando-se dinâmica de sistemas, apoiado pela ferramenta PowerSim Constructor. Logo em seguida, será apresentado o tópico onde o exemplo será simulado com a ferramenta JynaCoreSim, e um sub-tópico usando uma abordagem do exemplo com cenário.

5.1 Modelagem utilizando a ferramenta PowerSim Constructor

Para simular o modelo usando Dinâmica de Sistemas será utilizada a Ferramenta PowerSim Constructor.

Sua modelagem é feita divida em duas etapas. Primeiramente deve-se montar o modelo que representa a abordagem Best-Fitted Resource. Em seguida é construído o modelo da atividade de desenvolvimento do software apoiado pela teoria das curvas de aprendizado.

Esta parte do modelo é feita apenas de variáveis e constantes. O PowerSim Constructor permite que variáveis possuam vetores e matrizes como valores.

Para expressar a tarefa a ser desenvolvida são utilizadas três variáveis: uso esperado, complexidade e significância. Cada uma dessas variáveis contém um vetor com os valores definidos no início deste capítulo. O PowerSim Constructor permite a criação de intervalos de valores enumerados. Para esse exemplo, foi criado um vetor com os seguintes índices: Hardware H, C++, Java, Perl, VB. Todos os vetores deste modelo possuem suas dimensões correspondentes à esse índice.

Os valores de uso esperado (e_{jt}) e complexidade (c_{jt}) são pré-definidos, logo estas variáveis são representadas como constantes. O valor da significância (s_{jt}) é dado pelo produto simples dos valores de uso esperado e complexidade, portanto esses servem de informação para a significância.

Quando valores de variáveis ou constantes são vetores ou matrizes, o PowerSim Constructor representa esses auxiliares com uma borda dupla.

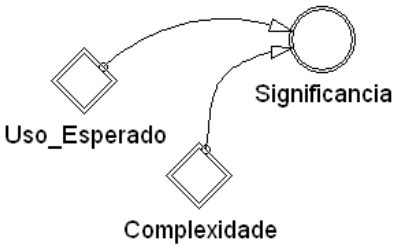

Figura 4. Representação da tarefa a ser executada

O segundo passo é determinar a tabela de relacionamentos entre habilidades. Esta tabela será representada por uma constante que possui como valor uma matriz. Cada valor dessa matriz representa o valor do relacionamento (r_{jk}) entre as habilidades.

Figura 5. Tabela de Relacionamento entre habilidades

Um modelo representará a produtividade de apenas um desenvolvedor. Para expressar um desenvolvedor, será utilizada uma variável com um vetor de valores referentes aos níveis de conhecimento (l_{yj}) deste desenvolvedor. Para cada desenvolvedor, esses valores serão ajustados e o restante do modelo será replicado, ou seja, apenas a linha da tabela de conjunto de habilidades referente este desenvolvedor será expressa.

Figura 6. Tabela de habilidades de um desenvolvedor.

O último passo da construção do modelo BFR consiste em calcular o Fitness do desenvolvedor, que representa a sua capacidade de adaptação para a dada tarefa.

A tabela BFR é divida e expressa por linha. Primeiramente é calculada a linha que representa o produto (b_{yj}) x (l_{yj}) para cada habilidade *j* em relação ao recurso *y*. Uma

variável “Habilidade” é criada para conter o vetor com os valores para cada uma dessas linhas, como mostra a Figura 7.

Para calcularmos o valor máximo de cada relacionamento, que ajudará a definir o Fitness do recurso, é utilizado uma variável auxiliar para cada habilidade contendo esse valor.

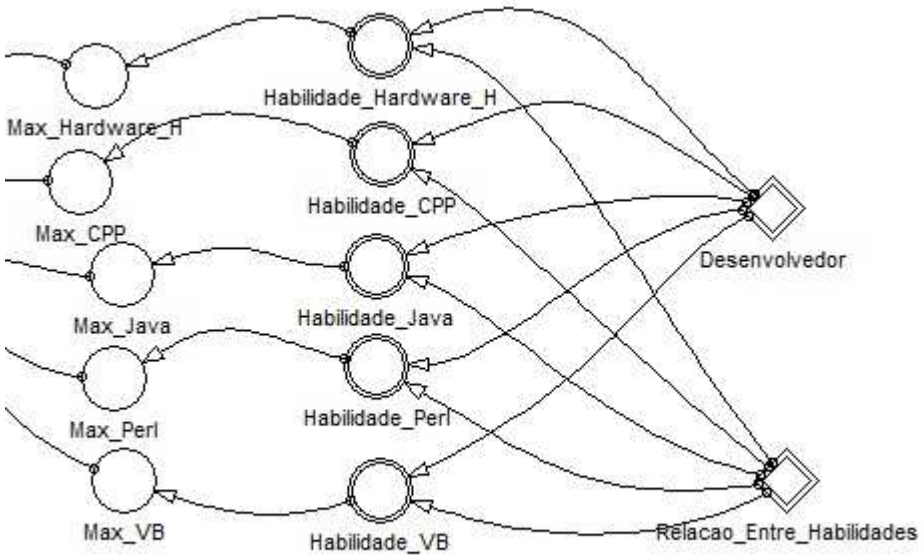

Figura 7. Cada variável representa a linha correspondente da tabela BFR para o desenvolvedor *y*.

É utilizado uma variável auxiliar, chamada “Relacionamento”, para agrupar os valores dos máximos em um vetor.

Finalmente, é calculado o Fitness do recurso. O produto dos elementos do vetor “Relacionamentos” com os elementos do vetor “Significância” resultara em um vetor com os valores para o calculo do Fitness do recurso *y*. Somatório desses valores representa o Fitness real do recurso, mas ele só será calculado no próximo passo da modelagem.

A figura 8 representa o modelo BFR completo, com suas variáveis e informações representadas.

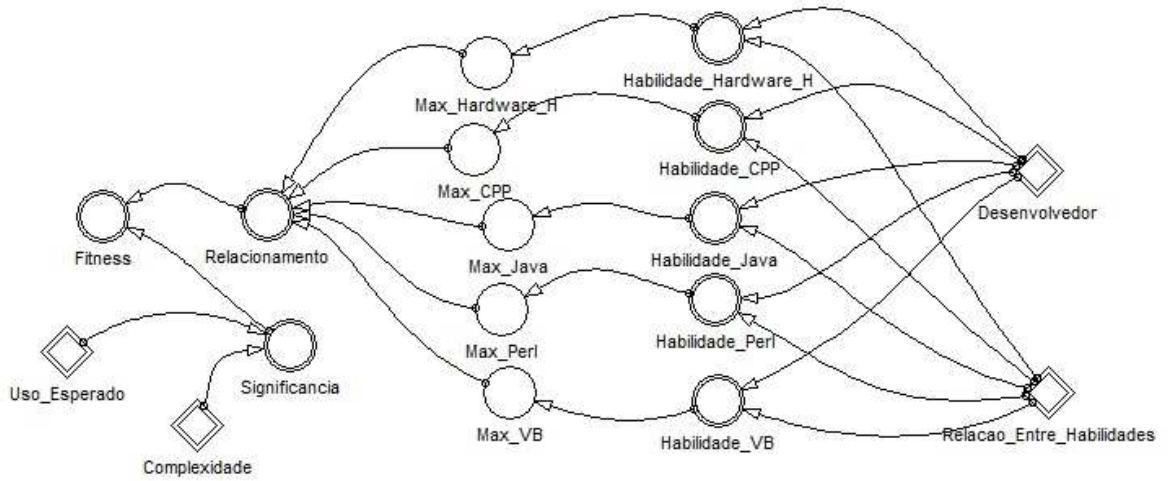

Figura 8. Modelo BFR

Para calcular a produtividade do recurso ao longo do tempo utilizaremos à fórmula de curva de aprendizagem.

Seguindo a Fórmula 3 (Lawrence, 1996), apresentada no capítulo 2.4.

A variável “B”, assim como na fórmula, representa Log(R)/Log(2), sendo “R” a “Taxa_de_Aprendizagem”. Essa por sua vez é calculada usando o valor do somatório da variável Fitness encontrado pelo método BFR. Note que o somatório dos valores dessa variável representa o real valor do Fitness do recurso *y*, como determina o método BFR.

O custo K_n para produzir a primeira unidade é expresso pela constante “Custo_Inicial”. Este é definido arbitrariamente e representa o tempo necessário para se produzir à primeira linha de código, por quaisquer dos recursos.

O custo da m-ésima unidade a se produzir será expresso pela variável “Custo_por_Unidade”. Este valor é definido pelo produto Custo Inicial, vezes Software_Desenvolvido elevado a “B”.

A “Produtividade” será o inverso do valor do “Custo_por_Unidade”e representará quantas linhas de código são produzidas em determinado passo de simulação.

A Figura 9 mostra todas as variáveis envolvidas no cálculo da quantidade de software desenvolvido.

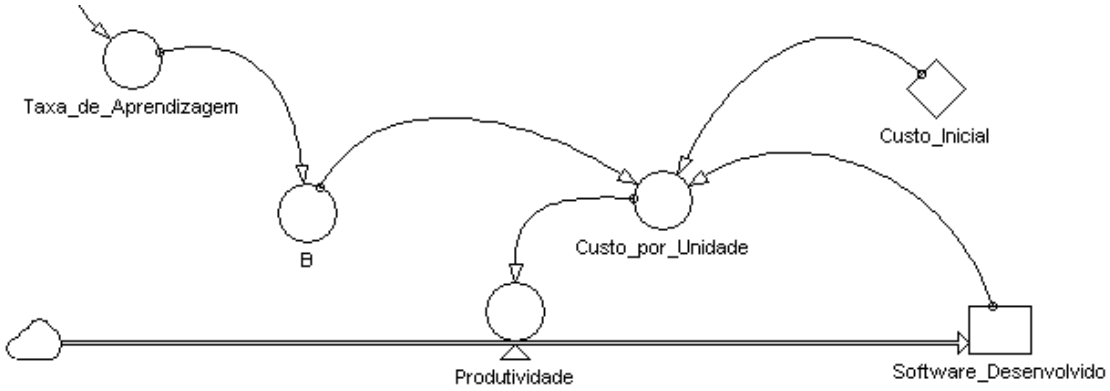

Figura 9. Modelo de curva de aprendizagem

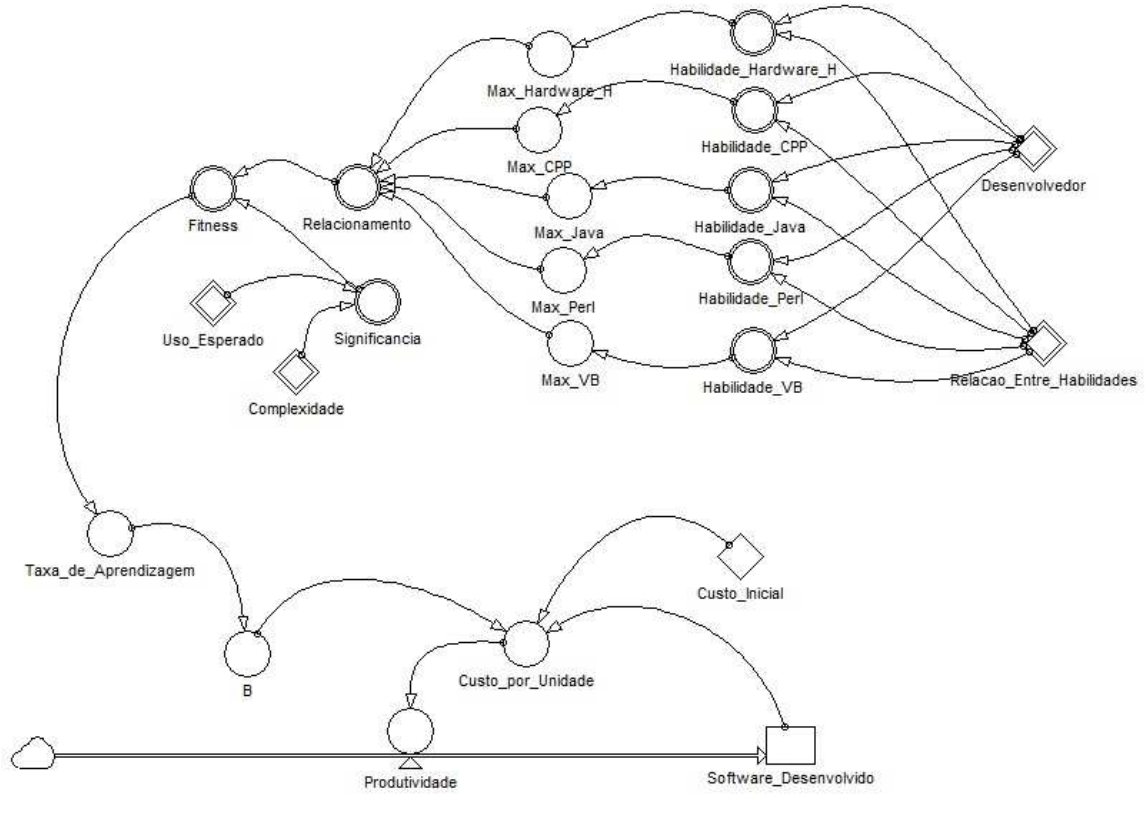

Figura 10. Modelo de Dinâmica de Sistemas para alocação de recursos baseado no método Best-Fitted Resource e curvas de aprendizado

Construído o modelo, é feita uma simulação para cada recurso disponível. As simulações são definidas com um número de 1000 passos, representando 1000 minutos de trabalho de cada recurso.

Para cada simulação, são ajustados os valores da variável “Desenvolvedor”, de acordo com o nível de conhecimento de cada recurso em cada habilidade.

A Tabela 7 demonstra o resultado para cada simulação.

Recurso	Software Desenvolvido
Desenvolvedor 1	201,54
Desenvolvedor 2	197,75
Desenvolvedor 3	239,76
Desenvolvedor 4	137,95
Desenvolvedor 5	179,06
Desenvolvedor 6	174,87

Tabela 7. Quantidade de software desenvolvido por cada recurso na simulação

Note que o valor do custo inicial para se produzir uma linha de código foi definido arbitrariamente, sendo igual para todos os recursos. O intuito desse exemplo é evidenciar as diferenças entre as produtividades dos recursos, e não obter valores absolutos de quantidade de software produzido.

Se baseássemos a escolha do recurso para a realização de uma tarefa apenas nos dados referentes às habilidades requeridas pela tarefa e os níveis de conhecimento de cada desenvolvedor, chegaríamos à conclusão de que o desenvolvedor 1 estaria mais apto a desenvolvê-la.

A simulação demonstra que esta escolha poderia ser equivocada, uma vez que dados como a taxa de aprendizagem seriam desconsiderados, afetando significativamente os resultados.

5.2 Modelo para simulação utilizando a ferramenta JynaCoreSim

Neste tópico será apresentado a construção e simulação do metamodelo do exemplo proposto utilizando a ferramenta JynaCoreSim.

Os metamodelos são representados por modelos de domínio, instâncias de modelo e modelos de cenários. Primeiramente será mostrada a construção do modelo de domínio “Alocação de Recursos” que descreve as classes envolvidas no problema.

A construção do modelo seguirá novamente os passos da construção do modelo BFR, com suas respectivas classes, e em seguida a construção da classe Desenvolvimento. A classe Desenvolvimento representa a produtividade de cada desenvolvedor de acordo com sua curva de aprendizado.

Os metamodelos são descritos na linguagem XML e seus respectivos códigos estão listados no Apêndice A, devidamente comentados.

Para construir as classes, relembraremos o conjunto de itens que servem para descrevê-las. Esses itens podem ser propriedades, auxiliares, taxas, estoques ou

tabelas. Os quatro primeiros itens serão utilizados nesse trabalho e suas respectivas representações estão dispostas na Figura 11.

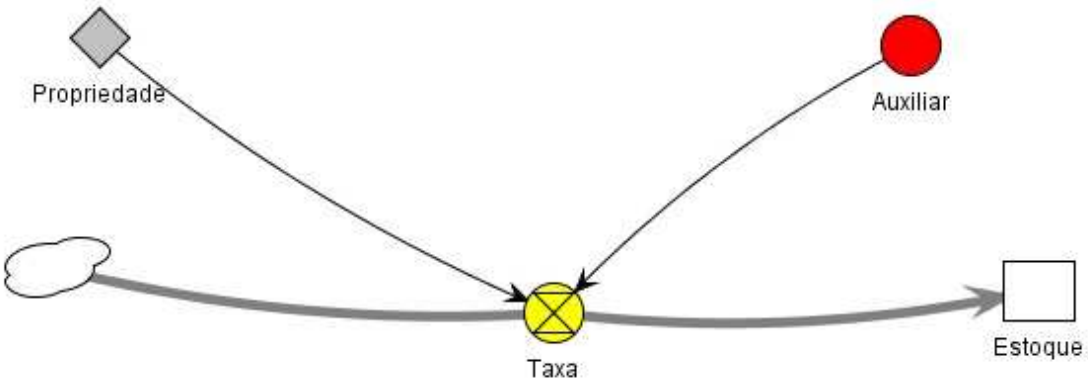

Figura 11. Os itens propriedade, auxiliar, taxa e estoque de uma classe.

Classes podem possuir relações entre si, que representam as possíveis conexões entre as instâncias, e podem ser simples, quando a relação é um para um, ou múltiplas quando a relação é um para muitos.

O primeiro passo da construção é definir a tarefa a ser desenvolvida. Para isso será criada a classe Habilidade. Essa classe representa o conjunto de habilidades H e possui para cada habilidade os valores e_{jt} de uso esperado, c_{jt} de complexidade e s_{jt} de significância.

Os valores de e_{jt} e c_{jt} são representados por propriedades. Cada propriedade é um item da classe do Metamodelo e possui dois campos: nome e valor. O valor de uma propriedade representa um valor padrão que poderá ser ajustado quando a classe for instanciada.

A significância s_{jt} é representada por um auxiliar que contém a expressão e_{jt} vezes c_{jt}.

Identificadores (ou auxiliares) são descritos pelo elemento *ci*. Números são descritos pelo elemento *cn*. O elemento *apply* significa aplique uma operação a uma expressão. O elemento que descreve uma operação, no exemplo *times* e *plus*, precede sempre os dois operandos da expressão, sejam eles auxiliares ou números.

Um elemento *apply* pode representar um operando da expressão, e os *applys* mais internos serão sempre processados antes.

A Listagem 1 do Apêndice A apresenta o trecho do código que define a classe Habilidade e suas propriedades e auxiliares.

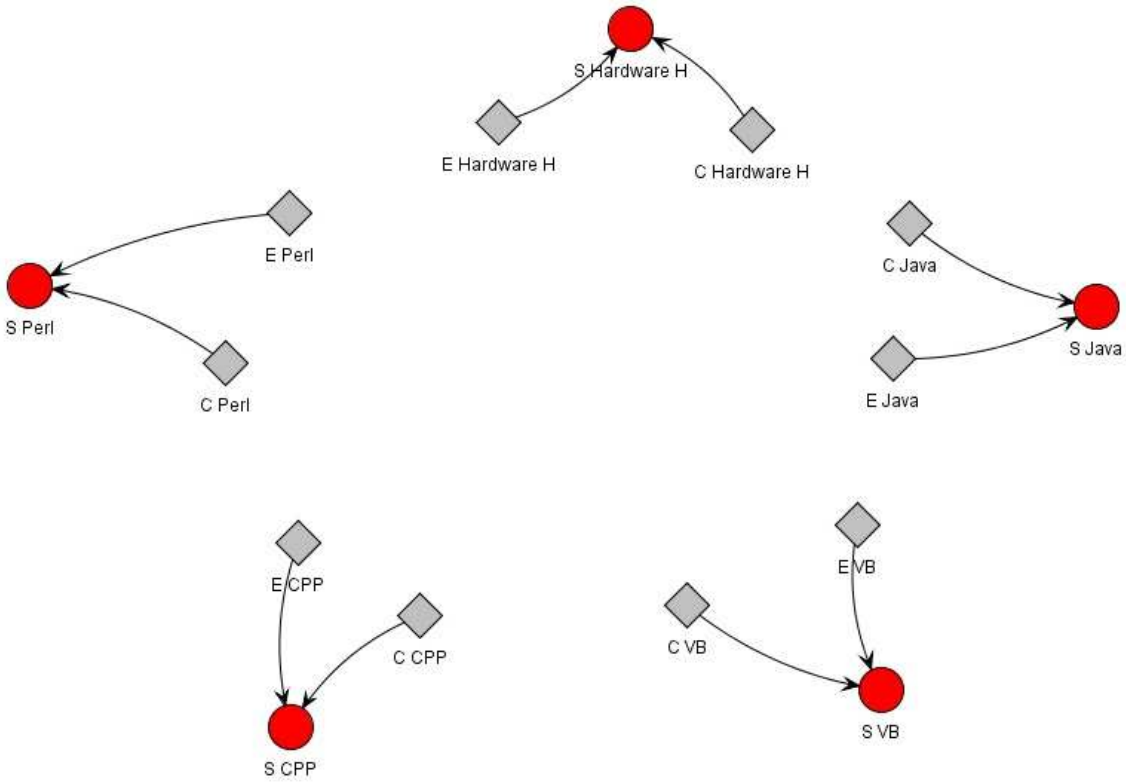

Figura 12. Representação da Classe Habilidade

Logo em seguida deverá ser criada a tabela de relacionamento entre as habilidades. Essa tabela será representada por uma classe que possui uma propriedade para cada relacionamento entre duas habilidades. Como a relação entre uma habilidade e ela mesma é sempre igual a 1, não há necessidade de expressar esse valor como propriedade da classe.

É fácil observar que $r_{jk} = r_{kj}$, logo, a representação das propriedades r_{kj} são dispensáveis no modelo.

A Listagem 2 no Apêndice A demonstra a classe Relacionamentos e seus respectivos itens.

A Figura 13 demonstra a classe Relacionamentos e suas propriedades.

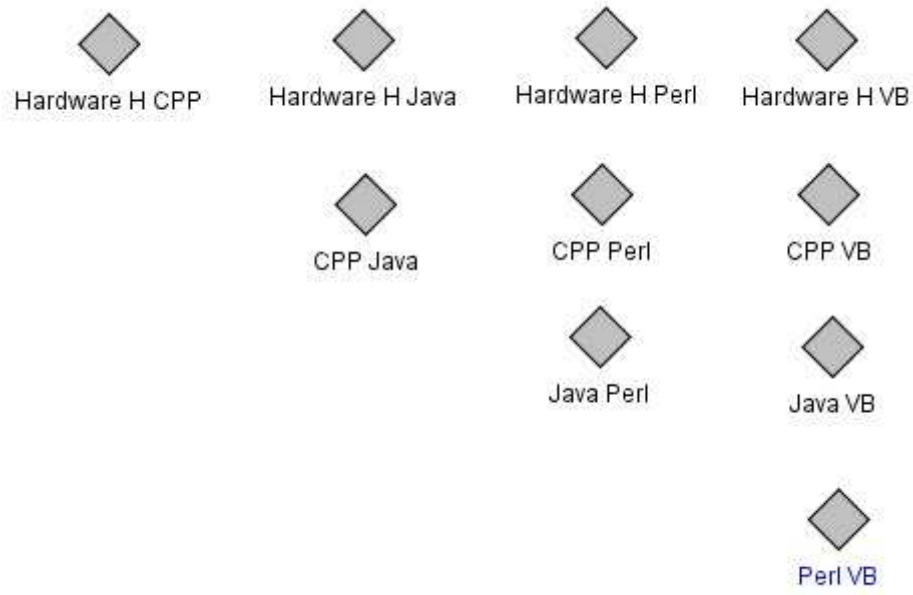

Figura 13. Classe Relacionamentos e suas propriedades

O terceiro passo é definir a tabela de recursos e seus níveis de conhecimento em cada habilidade.

A classe “Recurso” será utilizada para descrever um recurso genérico. Cada habilidade l_{yj} será descrita como uma propriedade da classe “Recurso” e terá seu valor padrão definido com 0, assim como na Listagem 3 do Apêndice A.

Para calcular o BFR, será criada uma série de auxiliares para calcular o relacionamento b_{yj} de cada recurso.

Para cada habilidade haverá uma propriedade chamada Relação *j*, onde *j* é o nome da habilidade. Esse auxiliar conterá uma expressão que calcula o maior entre o produto l_{yj} x r_{jk}. Como os valores de dos relacionamentos r_{jk} estão contidos na classe “Relacionamentos”, a classe “Recurso” terá uma relação simples com essa classe. Os auxiliares que representam esses relacionamentos estão descritos na Listagem 4 do Apêndice A.

Os relacionamentos são demonstrados na parte “relations” do metamodelo. Para uma relação simples entre a classe consumidora “Desenvolvedor” e a classe “Relacionamentos”, será criada uma singleRelation com o nome “Aptidao”, onde a fonte será o “Recurso” e o alvo será a classe “Relacionamentos”. A Listagem 5 do Apêndice A evidência essa relação.

Com essa relação criada, poderemos agora acessar as propriedades da classe “Relacionamentos”.

As operações das expressões de um auxiliar são feitas sempre entre dois operandos. Para expressar grandes fórmulas, devem-se aninhar as operações lembrando sempre que a aplicação de uma operação retornará um valor.

Os valores de cada relação são calculados a partir das instâncias da classe “Recurso” e recebem o valor de b_{yj} referentes a cada desenvolvedor distinto.

Cada desenvolvedor terá também um auxiliar Fitness. O valor desse auxiliar será obtido a partir da soma dos produtos b_{yj} x s_{jt}. Como os valores da significância s_{jt} estão contidos na classe “Habilidade”, a classe “Recurso” deverá possuir uma relação simples com essa classe. Essa relação é chamada de “Conhecimento”.

A Figura 14 representa a classe “Recurso” e seus itens. Note que os auxiliares “Relação” de cada habilidade devem ser calculadas a partir do conhecimento entre todas as habilidades do “Recurso”.

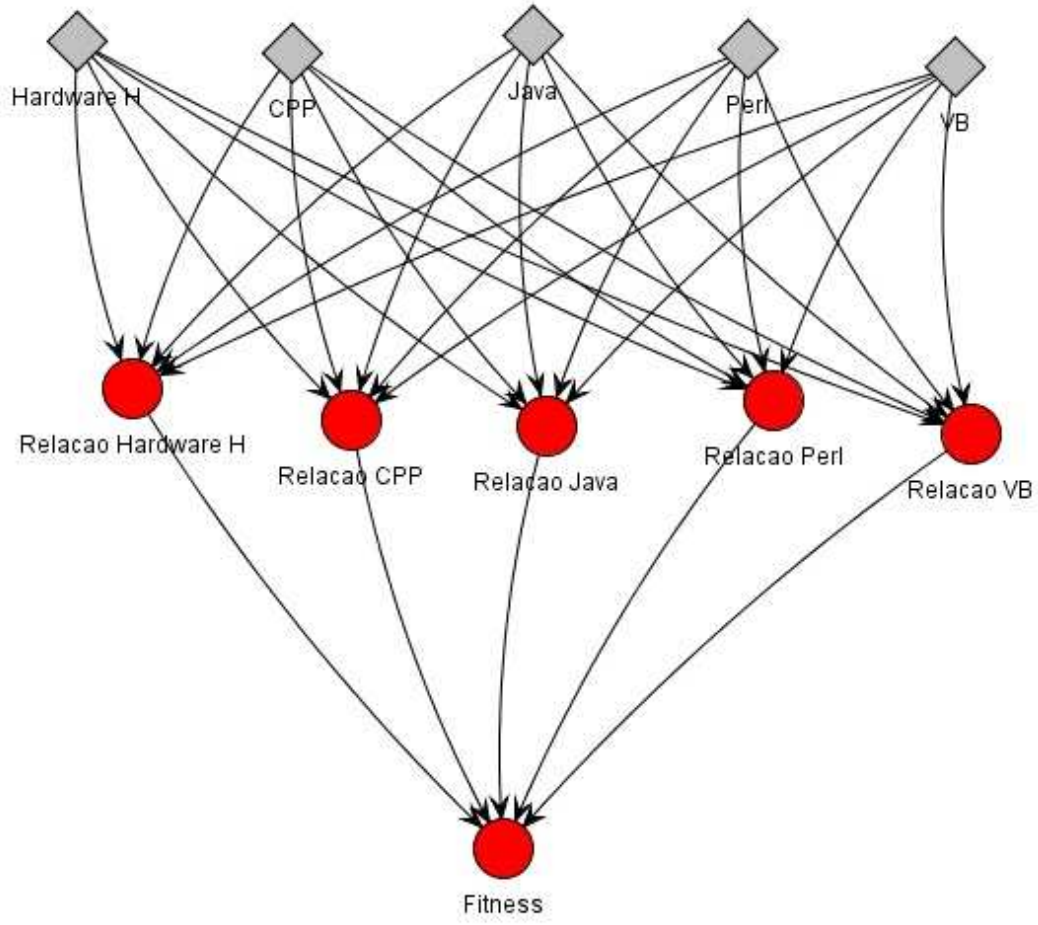

Figura 14. Classe Recurso e seus itens

Criadas as classes “Recurso”, “Habilidade” e “Relacionamentos”, o calculo do Fitness de cada usuário fica completo. Basta agora criar uma classe que denote a evolução da produtividade de cada usuário.

A classe “Desenvolvimento” possuirá um auxiliar para denotar o custo inicial para se produzir uma linha de código, um auxiliar para denotar o custo por unidade a partir da primeira, e os elementos B e taxa de aprendizagem da fórmula de curva de aprendizagem.

O custo inicial terá será definido com um valor arbitrário de 10.

Um estoque finito “Desenvolvido” será criado para conter a quantidade de linhas de código produzidas, que será alimentado pela taxa “Produtividade”.

O calculo da taxa de aprendizagem é feito a partir do Fitness do desenvolvedor, logo haverá uma relação simples entre a classe “Desenvolvimento” e a classe “Desenvolvedor”.

O auxiliar B é obtido usando logaritmos na base 10. Como o JynaCoreAPI utiliza apenas logaritmos neperianos, será necessário uma transformação de base na expressão. O código que descreve a classe “Desenvolvimento” encontra-se na Listagem 6 do Apêndice A.

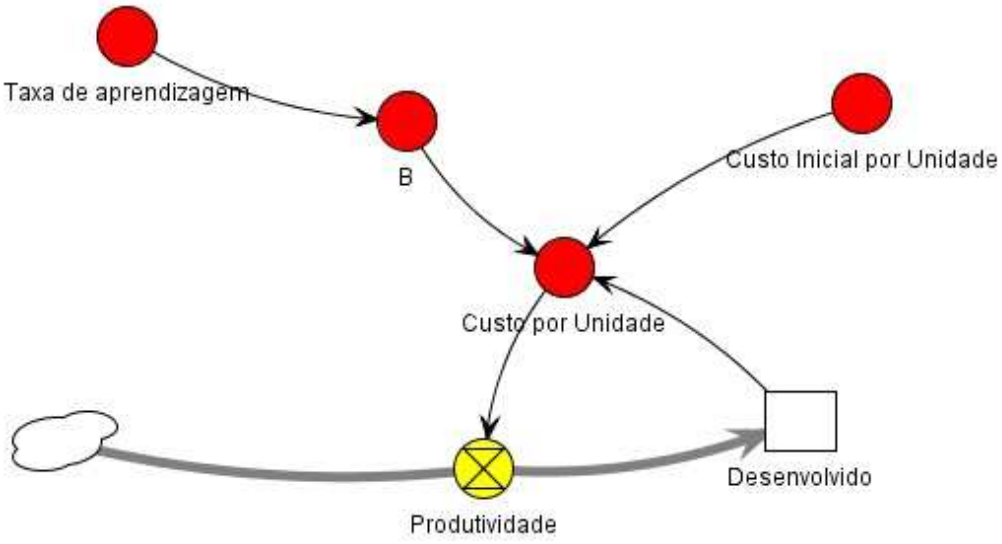

Figura 15. Classe Desenvolvimento e seus itens

O modelo de domínio “Alocação de Recursos” é representado pela Figura 16. Note os relacionamentos entre as classes.

Descritas as classes, o próximo passo será criar as instâncias. Para isso será criado um arquivo “Instancia de Alocação de Recursos.jymmi”. Esse documento deverá conter um nome, um indicador do arquivo de metamodelo que possui as classes, as descrições das instâncias e finalmente, a descrição dos cenários.

O nome dado ao metamodelo de instância foi “Alocacao de Recurso em Projeto de Software”. As classes a serem instanciadas estão descritas no arquivo “Alocacao de Recursos.jymm”.

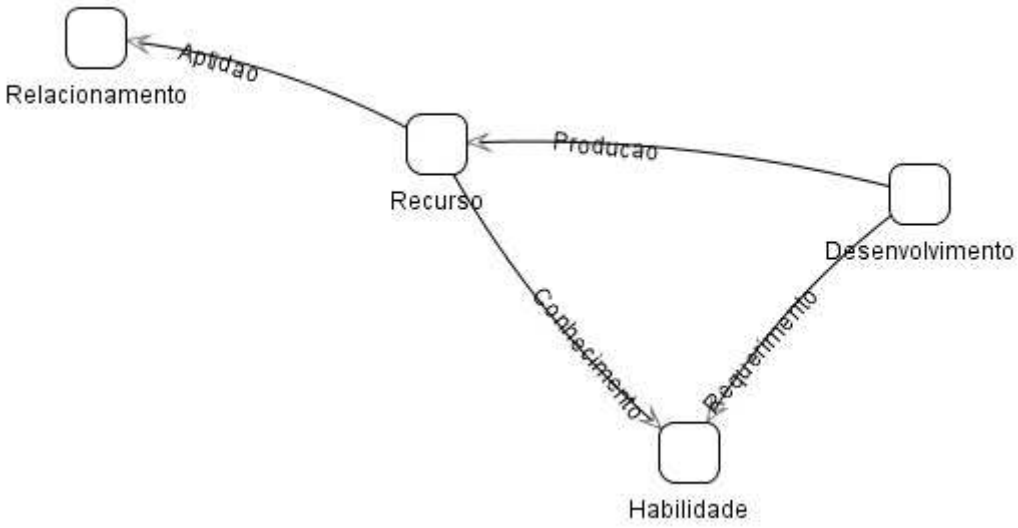

Figura 16. Metamodelo de Domínio “Alocação de Recursos”

Inicialmente será criada a instância da tarefa a ser desenvolvida. É criada uma instância da classe “Habilidade” que recebe o nome “Conjunto de Habilidades”. Os valores das propriedades, referentes às habilidades necessárias para a tarefa, serão iniciadas na instância, refletindo os valores do exemplo proposto, vide Listagem 7 do Apêndice A.

A tabela de relacionamentos entre habilidades deve ser definida com os valores da tabela SR. Para isso, será criada uma instância da classe “Relacionamento” com o nome de “Tabela de Relacionamentos”. Os valores das propriedades são definidos. Note que a definição das propriedades com valor 0 é redundante, pois esse valor é considerado padrão. A criação dessa instância e dada pela Listagem 8 do Apêndice A.

Em seguida, será criada uma instância para cada um dos seis desenvolvedores do exemplo. As instâncias receberão nomes D1, D2, D3, D4, D5 e D6, representando respectivamente os desenvolvedores 1, 2, 3, 4, 5 e 6. As instâncias foram criadas de acordo com a Listagem 9 do Apêndice A.

Seus valores de nível de conhecimento para cada habilidade são definidos aqui. A classe “Recurso” possui uma relação simples “B” para com a classe “Relacionamento”, portanto ela deve ser criada, referenciando como alvo a instância já criada “Tabela de Relacionamentos”. Ela possui também um relacionamento simples “Conhecimento” para

com a classe “Habilidade”, e analogamente, ela deve ser criada, referenciando a instância “Conjunto de Habilidades” como alvo.

Finalmente, serão criadas instâncias da classe Desenvolvimento para cada instância da classe Desenvolvedor. Seus nomes serão “Produção” seguido do nome da respectiva instância ao qual se conectarão, como mostra a Listagem 10 do Apêndice A. O único item dessas instâncias será a relação simples “Aptidão” para seus respectivos Desenvolvedores.

A Figura 17 demonstra a instância de modelo.

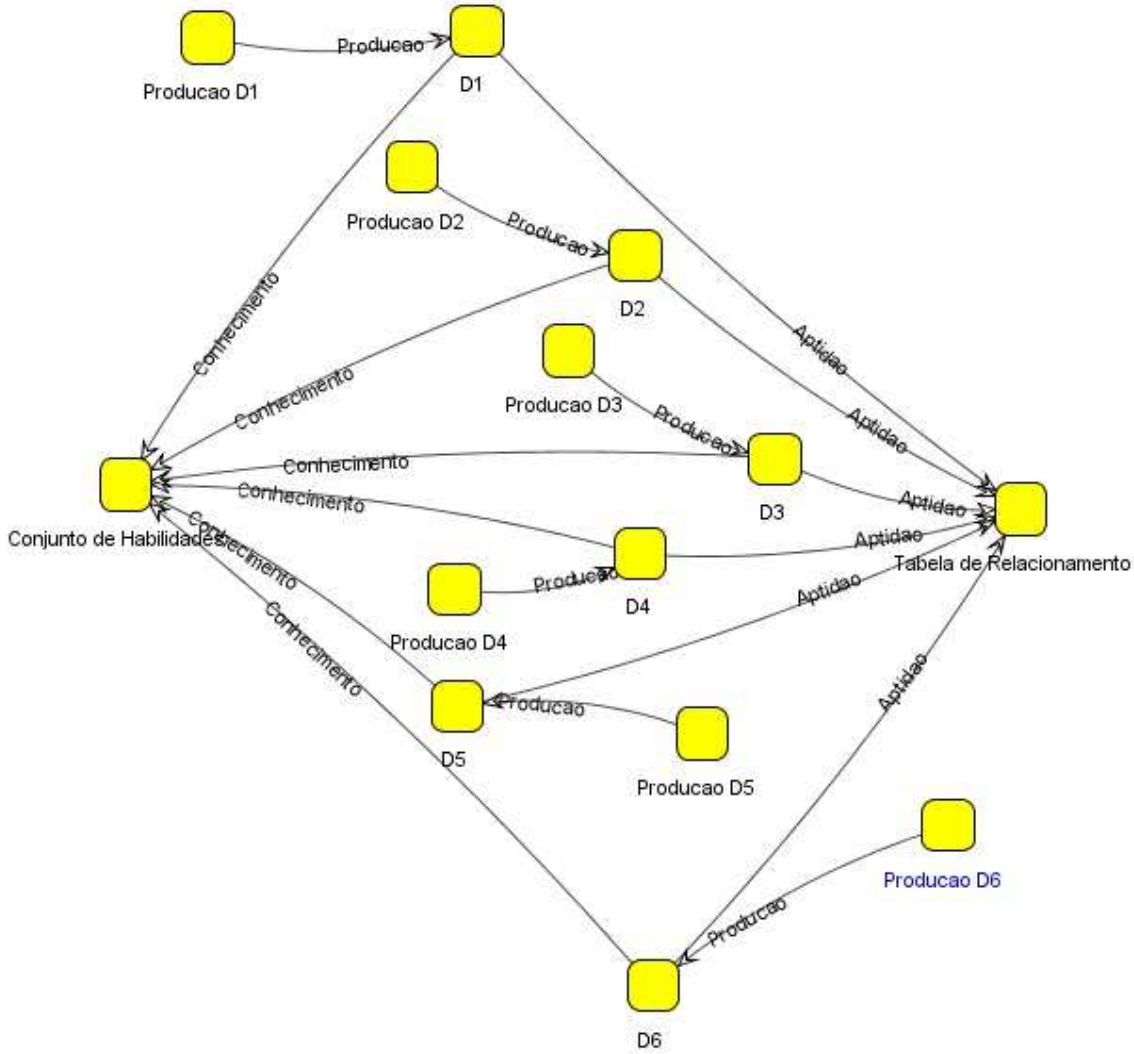

Figura 17. Instância de Modelo

Definidas as classes do modelo de domínio e criada a sua instância, o próximo passo é a simulação do exemplo. A simulação foi definida, assim como na simulação feita na ferramenta PowerSim Constructor, com 1000 passos.

A simulação gera uma tabela e um gráfico com a progressão de todos os valores da instância. A Figura 17 demonstra um exemplo de um gráfico onde foram filtrados apenas os valores de produtividade dos recursos.

Todos os valores contidos no exemplo podem ser filtrados para exibição em tabela ou gráfico.

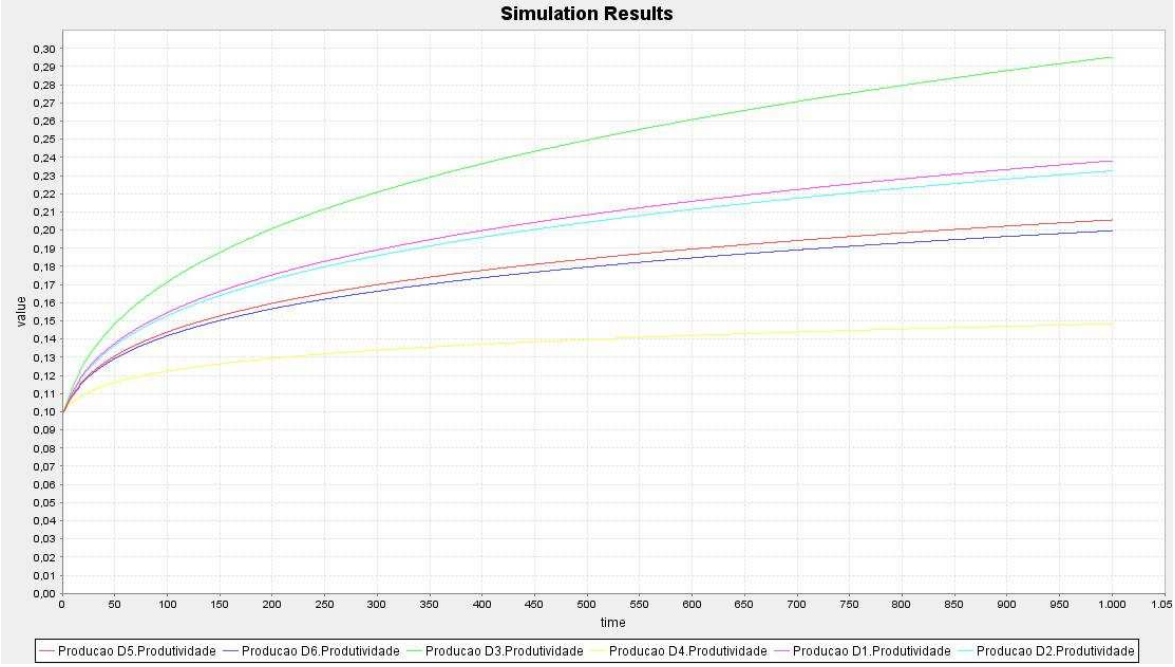

Figura 18. Gráfico da produtividade de cada desenvolvedor

5.2.1 Abordagem do Exemplo Usando Cenários

Como vimos cenários são elementos que modificam a estrutura dos modelos de instâncias.

Nos modelos apresentados, o custo inicial para a produção de uma linha de código foi definida com um valor arbitrário igual para todos os desenvolvedores. E se o custo inicial levasse em conta a experiência do desenvolvedor?

Neste tópico abordaremos o efeito que a experiência prévia de um desenvolvedor tem sobre o custo de se produzir a primeira linha de código. Essa hipótese será testada com ajuda do recurso de modelos de cenários oferecido pelos metamodelos de dinâmica de sistemas.

Para alterar o comportamento de uma classe, é criado um arquivo de modelo de cenário que deverá se conectar ao modelo de instância. Chamaremos esse modelo de cenário de “Custo Inicial Baseado nas Habilidades”.

O campo “connections” contém a lista de conexões com as classes e instâncias que ela afeta.

Para modificar o valor do da expressão que define o custo inicial por unidade, deve-se criar uma conexão com a classe “Desenvolvimento”. Essa conexão recebe o nome de “Experiência”.

Como este cenário altera o modelo de domínio como um tudo, não há necessidade de especificar quais instâncias ele será conectado.

Essa conexão deve especificar qual item da classe “Desenvolvimento” ela afetará. Dentro do conjunto “affects” é listado todos os itens que serão ajustados, e seus respectivos ajustes.

O cenário modifica a expressão que define o valor do item “Custo Inicial por Unidade”. Note que no modelo original, esse item foi definido como um auxiliar e não uma propriedade, apesar de seu valor ser um número e não uma expressão. A utilidade dessa definição será evidenciada com a utilização do cenário.

Basta agora, apenas definir a expressão que definirá o custo inicial. Levaremos em conta os níveis de conhecimento do recurso, bem como os níveis de significância das habilidades utilizadas para realização da tarefa. Esse cálculo será análogo ao calculo usado para o calculo do Fitness de cada recurso, porém ao invés de se utilizar o valor da relação “b” do recurso *y* com a habilidade *j*, será utilizado o nível de conhecimento “l” do recurso *y* sobre a habilidade *j*. Essa modificação é feita, pois o nível de conhecimento atual do recurso sobre a habilidade, e não sua afinidade para com ela define quanto tempo ele levará para produzir uma linha de código de uma determinada tarefa.

A Listagem 11 do Apêndice A mostra a nova expressão que define o cálculo do custo inicial por unidade.

Observe que a classe “Desenvolvimento” possui uma nova relação chamada “Requerimento”, com a classe “Habilidade”. Essa relação deverá ser criada no modelo domínio e definida para cada instância da classe “Desenvolvimento”.

Criado o cenário basta agora conectá-lo às instâncias. Para isso será criado um novo arquivo de instância de modelo chamado “Instância Alocação de Recursos com Cenário”. O conteúdo desse modelo de instância é inicialmente o mesmo do modelo “Instância de Alocação de Recursos”. Como a classe “Desenvolvimento” possui uma relação com a classe “Habilidade”, uma singleRelation será criada para cada instância “Produção” tendo como alvo a instância “Conjunto de Habilidades”.

Para conectar o cenário às instâncias, é indicado na seção “scenarios” o arquivo que contém as modificações. Uma vez especificado o cenário cria-se na seção “connects” uma conexão “Experiencia” para cada instância “Produção”, como na Listagem 12 do Apêndice A.

A nova representação do modelo de domínio, com o cenário que se conecta a ele é exposto pela Figura 19.

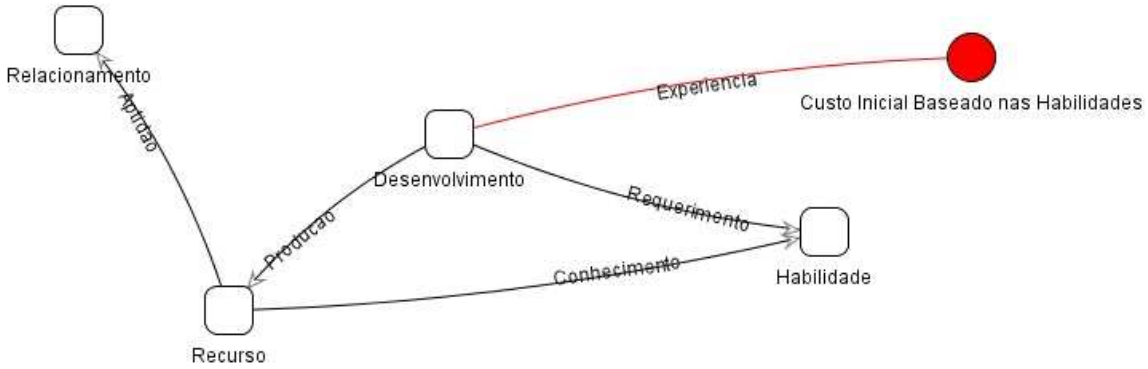

Figura 19. Modelo de domínio “Alocação de Recursos” modificado pelo cenário “Custo Inicial Baseado nas Habilidades”

A aplicação deste cenário implica em diferentes resultados na simulação. Note que a produtividade do recurso 1 supera as demais neste caso. Além disso, é possível observar que as progressões na produtividade não são tão uniformes entre si, com relação à simulação sem cenário, uma vez que a base para o cálculo do custo por unidade varia.

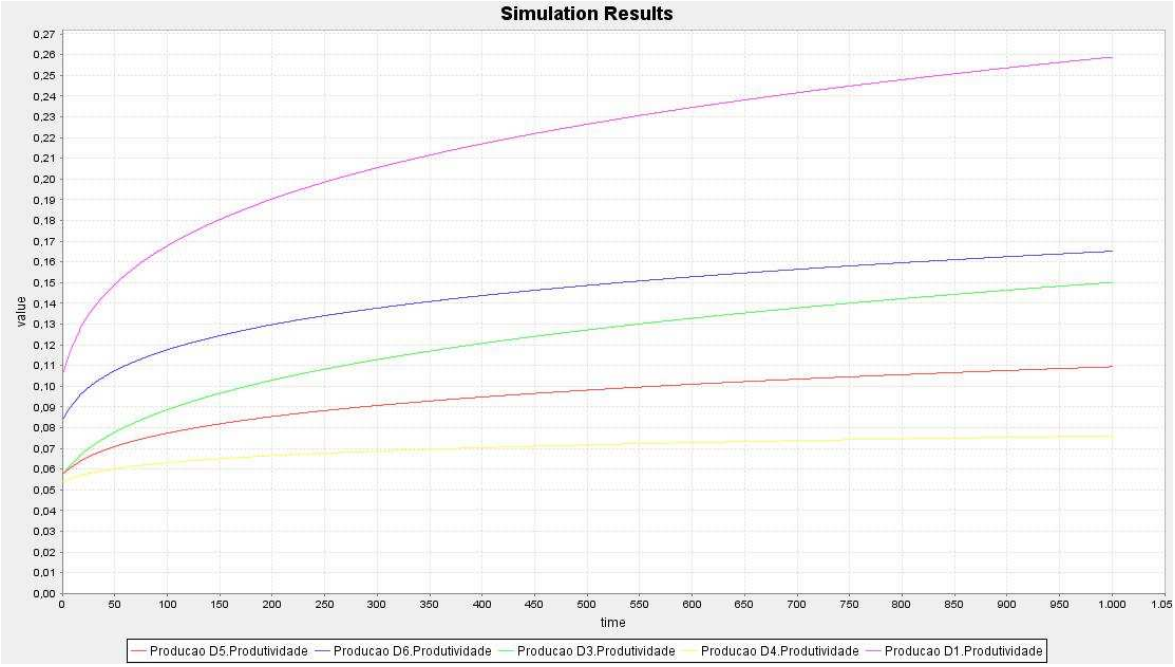

Figura 20. Progressão da produtividade afetada pelo cenário

A utilização de cenários provê uma flexibilização dos modelos. No âmbito de desenvolvimento de software, os cenários podem ser usados para descrever comportamentos, eventos e políticas não consideradas na definição de um projeto.

6. Conclusão

O objetivo principal deste trabalho foi demonstrar o uso de ferramentas de modelagem e simulação para o estudo de uma abordagem em processo de desenvolvimento de software.

Decisões sobre alocação de recursos em ambientes de desenvolvimento de software podem ser vitais para manutenção dos prazos. Ambientes de simulação possibilitam uma maior compreensão da dinâmica envolvida nos processos de software, proporcionando melhores escolhas. A utilização de cenários permite observar os vários comportamentos possíveis de um modelo. A partir da modelagem e simulação é possível observar como os elementos do sistema se interagem ao longo do tempo.

O PowerSim Constructor se baseia na simplicidade dos elementos de dinâmica de sistemas, porém inclui recursos próprios para lidar com a crescente complexidade de modelos. Esses recursos podem providenciar algumas saídas para a construção de alguns modelos, mas por outro lado eles tornam o entendimento dos modelos mais complicado, fugindo a idéia inicial de dinâmica de sistemas.

Alguns modelos apresentam repetição de algumas estruturas, como no exemplo proposto os desenvolvedores. No PowerSim Constructor, essas repetições são tratadas utilizando-se replicas dos elementos que descrevem essas estruturas, o que diminui a praticidade dos modelos.

A ferramenta JynaCoreSim torna acessível a construção e estudo de modelos em um ambiente aberto, entretanto é necessário a construção de um editor de modelos. A construção dos modelos utilizando diretamente a linguagem XML pode tornar-se um grave empecilho em modelos mais rebuscados.

É necessário ainda à ferramenta, um tratamento de erros e exceções para guiar os modeladores nas particularidades da linguagem.

A divisão da linguagem em diferentes níveis de abstração possibilita a fácil compreensão dos modelos por mais complexos que eles possam se tornar. A descrição das classes é simples, uma vez que utiliza elementos análogos aos de dinâmica de sistemas. Por manter a linguagem simples, os modelos para o JynaCoreSim são intuitivos de serem construídos.

Ao introduzir instâncias de modelos, a replicação de elementos passa a trabalhar a favor da compreensão dos modelos, ao contrário do que acontece com o PowerSim Constructor.

O repositório de cenários permite que os modelos assumam propriedades menos ideais aproximando-os do mundo real, e cada vez mais de uma simulação mais apurada.

Um cenário é uma forma de guardar uma modificação em um modelo de forma reutilizável.

7. Referências Bibliográficas

BARROS, M. d. O. Gerenciamento de Projetos Baseado em Cenários: Uma Abordagem de Modelagem Dinâmica e Simulação. Tese (Doutorado) COPPE/UFRJ, Rio de Janeiro, 2001.

FORRESTER, J. W. Industrial dynamics. [S.l.]: The M.I.T. Press, 1961. ISBN 0-262-56001-1.

KNOP, I. d. O. Infraestrutura para simulação de processos de Software baseada em metamodelos de dinâmica de Sistemas. Tese (Mestrado) – UFJF, Juiz de Fora, 2009.

LAWRENCE, STEPHEN R Learning curve theory, 1996, http://pages.swcp.com/raccoon/learncurve.html

LEHMAN, M. M ; KAHEN, G. ; RAMIL, J. F. AND P. WERNICK. System dynamics modelling of software evolution processes for policy investigation: Approach and example. Elsiever 2001

LINBERG, KURT R. (1999). Software developer perceptions about software project failure: A case study. The Journal of Systems and Software, 49, 177–192.

MADACHY, R. Software Process Dynamics. [S.l.]: IEEE Press Wiley-InterScience, 2008. ISBN 978-0471274551.

MathML, W3C, 2002, http://www.w3.org/1998/Math/MathML

OTERO, L. D.; CENTENO, G.; RUIZ-TORRES, A. J.; OTERO, C. E. A systematic approach for resource allocation in software projects. Elsevier 2008

VILLELA, P.R.C., 2005, *Introdução a Dinâmica de Sistemas.* Apostila., UFJF, Juiz de Fora, MG, Brasil.

Apêndice A – Listagem dos códigos do metamodelo

Esse apêndice traz os códigos XML que descrevem o metamodelo apresentado no capítulo 5.2.

As notações de expressões seguem as recomendações MathML do W3C (MathML, 2002). Segue um exemplo que introduz a idéia básica da recomendação, essencial para o entendimento das expressões dos códigos.

```
<m:apply>
        <m:times/>
                <m:ci>A</m:ci>
                <m:cn>3</m:cn>
</m:apply>
```

Esse código denota: A x 3

As expressões são sempre resolvidas entre dois operandos. Quando um operando é um identificador utiliza-se *ci*. Quando é um número utiliza-se *cn*.

```
<classes>
    <class>
      <name> Habilidade </name>
      <items>
        <property>
          <name>E Hardware H</name>
          <value>0.0</value>
        </property>
        <property>
          <name>C Hardware H</name>
          <value>0.0</value>
        </property>
        <auxiliary>
          <name>S Hardware H</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:times />
                <m:ci>E Hardware H</m:ci>
                <m:ci>C Hardware H</m:ci>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <property>
          <name>E CPP</name>
          <value>0.0</value>
        </property>
        <property>
          <name>C CPP</name>
          <value>0.0</value>
```

```
        </property>
        <auxiliary>
          <name>S CPP</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:times />
                <m:ci>E CPP</m:ci>
                <m:ci>C CPP</m:ci>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <property>
          <name>E Java</name>
          <value>0.0</value>
        </property>
        <property>
          <name>C Java</name>
          <value>0.0</value>
        </property>
        <auxiliary>
          <name>S Java</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:times />
                <m:ci>E Java</m:ci>
                <m:ci>C Java</m:ci>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <property>
          <name>E Perl</name>
          <value>0.0</value>
        </property>
        <property>
          <name>C Perl</name>
          <value>0.0</value>
        </property>
        <auxiliary>
          <name>S Perl</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:times />
                <m:ci>E Perl</m:ci>
                <m:ci>C Perl</m:ci>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <property>
          <name>E VB</name>
          <value>0.0</value>
        </property>
        <property>
          <name>C VB</name>
```

```
          <value>0.0</value>
        </property>
        <auxiliary>
          <name>S VB</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:times />
                <m:ci>E VB</m:ci>
                <m:ci>C VB</m:ci>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
      </items>
    </class>
```

Listagem 1. Definição da classe Habilidade

```
<class>
      <name> Relacionamentos </name>
      <items>
        <property>
          <name>Hardware H CPP</name>
          <value>0</value>
        </property>
        <property>
          <name>Hardware H Java</name>
          <value>0</value>
        </property>
        <property>
          <name>Hardware H Perl</name>
          <value>0</value>
        </property>
        <property>
          <name>Hardware H VB</name>
          <value>0</value>
        </property>
        <property>
          <name>CPP Java</name>
          <value>0</value>
        </property>
        <property>
          <name>CPP Perl</name>
          <value>0</value>
        </property>
        <property>
          <name>CPP VB</name>
          <value>0</value>
        </property>
        <property>
          <name>Java Perl</name>
          <value>0</value>
        </property>
        <property>
          <name>Java VB</name>
          <value>0</value>
        </property>
        <property>
```

```
          <name>Perl VB</name>
          <value>0</value>
        </property>
      </items>
    </class>
```

Listagem 2. Relacionamento entre as habilidades.

```
<class>
      <name>Recurso</name>
      <items>
        <property>
          <name>Hardware H</name>
          <value>0</value>
        </property>
        <property>
          <name>CPP</name>
          <value>0</value>
        </property>
        <property>
          <name>Java</name>
          <value>0</value>
        </property>
        <property>
          <name>Perl</name>
          <value>0</value>
        </property>
        <property>
          <name>VB</name>
          <value>0</value>
        </property>
(…)
```

Listagem 3. Definição das propriedades relativas ao conhecimento de cada habilidade.

```
<auxiliary>
          <name>Relacao Hardware H</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
              <m:max />
                <m:apply>
                  <m:times />
                  <m:ci>Hardware H</m:ci>
                  <m:cn>1</m:cn>
                </m:apply>
                <m:apply>
                <m:max />
                  <m:apply>
                    <m:times />
                    <m:ci>CPP</m:ci>
```

```
                <m:ci>Aptidao.Hardware H CPP</m:ci>
              </m:apply>
              <m:apply>
              <m:max/>
                <m:apply>
                  <m:times />
                  <m:ci>Java</m:ci>
                  <m:ci>Aptidao.Hardware H Java</m:ci>
                </m:apply>
                <m:apply>
                <m:max/>
                  <m:apply>
                    <m:times />
                    <m:ci>Perl</m:ci>
                    <m:ci>Aptidao.Hardware H Perl</m:ci>
                  </m:apply>
                  <m:apply>
                    <m:times />
                    <m:ci>VB</m:ci>
                    <m:ci>Aptidao.Hardware H VB</m:ci>
                  </m:apply>
                </m:apply>
              </m:apply>
            </m:apply>
          </m:apply>
        </m:math>
      </expression>
    </auxiliary>
    <auxiliary>
      <name>Relacao CPP</name>
      <expression>
        <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
          <m:apply>
          <m:max />
            <m:apply>
              <m:times />
              <m:ci>Hardware H</m:ci>
              <m:ci>Aptidao.Hardware H CPP</m:ci>
            </m:apply>
            <m:apply>
            <m:max />
              <m:apply>
                <m:times />
                <m:ci>CPP</m:ci>
                <m:cn>1</m:cn>
              </m:apply>
              <m:apply>
              <m:max/>
                <m:apply>
                  <m:times />
                  <m:ci>Java</m:ci>
                  <m:ci>Aptidao.CPP Java</m:ci>
                </m:apply>
                <m:apply>
                <m:max/>
                  <m:apply>
                    <m:times />
                    <m:ci>Perl</m:ci>
                    <m:ci>Aptidao.CPP Perl</m:ci>
```

```
                    </m:apply>
                    <m:apply>
                      <m:times />
                      <m:ci>VB</m:ci>
                      <m:ci>Aptidao.CPP VB</m:ci>
                    </m:apply>
                  </m:apply>
                </m:apply>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <auxiliary>
          <name>Relacao Java</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
              <m:max />
                <m:apply>
                  <m:times />
                  <m:ci>Hardware H</m:ci>
                  <m:ci>Aptidao.Hardware H Java</m:ci>
                </m:apply>
                <m:apply>
                <m:max />
                  <m:apply>
                    <m:times />
                    <m:ci>CPP</m:ci>
                    <m:ci>Aptidao.CPP Java</m:ci>
                  </m:apply>
                  <m:apply>
                  <m:max/>
                    <m:apply>
                      <m:times />
                      <m:ci>Java</m:ci>
                      <m:cn>1</m:cn>
                    </m:apply>
                    <m:apply>
                    <m:max/>
                      <m:apply>
                        <m:times />
                        <m:ci>Perl</m:ci>
                        <m:ci>Aptidao.Java Perl</m:ci>
                      </m:apply>
                      <m:apply>
                        <m:times />
                        <m:ci>VB</m:ci>
                        <m:ci>Aptidao.Java VB</m:ci>
                      </m:apply>
                    </m:apply>
                  </m:apply>
                </m:apply>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <auxiliary>
          <name>Relacao Perl</name>
```

```
        <expression>
          <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
            <m:apply>
            <m:max />
              <m:apply>
                <m:times />
                <m:ci>Hardware H</m:ci>
                <m:ci>Aptidao.Hardware H Perl</m:ci>
              </m:apply>
              <m:apply>
              <m:max />
                <m:apply>
                  <m:times />
                  <m:ci>CPP</m:ci>
                  <m:ci>Aptidao.CPP Perl</m:ci>
                </m:apply>
                <m:apply>
                <m:max/>
                  <m:apply>
                    <m:times />
                    <m:ci>Java</m:ci>
                    <m:ci>Aptidao.Java Perl</m:ci>
                  </m:apply>
                  <m:apply>
                  <m:max/>
                    <m:apply>
                      <m:times />
                      <m:ci>Perl</m:ci>
                      <m:cn>1</m:cn>
                    </m:apply>
                    <m:apply>
                      <m:times />
                      <m:ci>VB</m:ci>
                      <m:ci>Aptidao.Perl VB</m:ci>
                    </m:apply>
                  </m:apply>
                </m:apply>
              </m:apply>
            </m:apply>
          </m:math>
        </expression>
      </auxiliary>
      <auxiliary>
        <name>Relacao VB</name>
        <expression>
          <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
            <m:apply>
            <m:max />
              <m:apply>
                <m:times />
                <m:ci>Hardware H</m:ci>
                <m:ci>Aptidao.Hardware H VB</m:ci>
              </m:apply>
              <m:apply>
              <m:max />
                <m:apply>
                  <m:times />
                  <m:ci>CPP</m:ci>
                  <m:ci>Aptidao.CPP VB</m:ci>
```

```
                </m:apply>
                <m:apply>
                <m:max/>
                  <m:apply>
                    <m:times />
                    <m:ci>Java</m:ci>
                    <m:ci>Aptidao.Java VB</m:ci>
                  </m:apply>
                  <m:apply>
                  <m:max/>
                    <m:apply>
                      <m:times />
                      <m:ci>Perl</m:ci>
                      <m:ci>Aptidao.Perl VB</m:ci>
                    </m:apply>
                    <m:apply>
                      <m:times />
                      <m:ci>VB</m:ci>
                      <m:cn>1</m:cn>
                    </m:apply>
                  </m:apply>
                </m:apply>
              </m:apply>
            </m:apply>
          </m:math>
        </expression>
      </auxiliary>
      <auxiliary>
        <name>Fitness</name>
        <expression>
          <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
            <m:apply>
              <m:plus />
              <m:apply>
                <m:times />
                <m:ci>Relacao Hardware H</m:ci>
                <m:ci>Conhecimento.S Hardware H</m:ci>
              </m:apply>
              <m:apply>
                <m:plus />
                <m:apply>
                  <m:times />
                  <m:ci>Relacao CPP</m:ci>
                  <m:ci>Conhecimento.S CPP</m:ci>
                </m:apply>
                <m:apply>
                  <m:plus />
                  <m:apply>
                    <m:times />
                    <m:ci>Relacao Java</m:ci>
                    <m:ci>Conhecimento.S Java</m:ci>
                  </m:apply>
                  <m:apply>
                    <m:plus />
                    <m:apply>
                      <m:times />
                      <m:ci>Relacao Perl</m:ci>
                      <m:ci>Conhecimento.S Perl</m:ci>
                    </m:apply>
```

```
                    <m:apply>
                      <m:times />
                      <m:ci>Relacao VB</m:ci>
                      <m:ci>Conhecimento.S VB</m:ci>
                    </m:apply>
                  </m:apply>
                </m:apply>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
      </items>
    </class>
```

Listagem 4. Relacionamentos entre cada habilidade e o recurso

```
    <singleRelation>
      <name>Aptidao</name>
      <source>Recurso</source>
      <target>Relacao</target>
    </singleRelation>
```

Listagem 5. Relação simples “Aptidão” da classe “Recurso” com a classe “Relação”

```
    <class>
      <name>Desenvolvimento</name>
      <items>
        <auxiliary>
          <name>Taxa de aprendizagem</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:divide />
                <m:apply>
                  <m:minus />
                  <m:cn>10</m:cn>
                  <m:ci>Producao.Fitness</m:ci>
                </m:apply>
                <m:cn>10</m:cn>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <auxiliary>
          <name>Custo Inicial por Unidade</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:cn>10</m:cn>
            </m:math>
          </expression>
        </auxiliary>
```

```
        <auxiliary>
          <name>Custo por Unidade</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:times />
                <m:ci>Custo Inicial por Unidade</m:ci>
                <m:apply>
                  <m:power />
                  <m:ci>Desenvolvido</m:ci>
                  <m:ci>B</m:ci>
                </m:apply>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <auxiliary>
          <name>B</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:divide />
                <m:apply>
                  <m:divide />
                  <m:apply>
                    <m:log />
                    <m:ci>Taxa de aprendizagem</m:ci>
                  </m:apply>
                  <m:apply>
                    <m:log />
                    <m:cn>10.0</m:cn>
                  </m:apply>
                </m:apply>
                <m:apply>
                  <m:divide />
                  <m:apply>
                    <m:log />
                    <m:cn>2.0</m:cn>
                  </m:apply>
                  <m:apply>
                    <m:log />
                    <m:cn>10.0</m:cn>
                  </m:apply>
                </m:apply>
              </m:apply>
            </m:math>
          </expression>
        </auxiliary>
        <rate>
          <name>Produtividade</name>
          <expression>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:apply>
                <m:divide />
                <m:cn>1.0</m:cn>
                <m:ci>Custo por Unidade</m:ci>
              </m:apply>
            </m:math>
          </expression>
```

```
          <target>Desenvolvido</target>
        </rate>
        <finiteStock>
          <name>Desenvolvido</name>
          <initialValue>
            <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
              <m:cn>1.0</m:cn>
            </m:math>
          </initialValue>
        </finiteStock>
      </items>
    </class>
  </classes>
   (…)
    <singleRelation>
      <name>Producao</name>
      <source>Desenvolvimento</source>
      <target>Recurso</target>
    </singleRelation>
```

Listagem 6. Classe “Desenvolvimento” e sua respectiva relação “Produção” com a classe “Recurso”

```
<classInstance>
                  <name>Conjunto de Habilidades</name>
                  <class>Habilidade</class>
                  <items>
                  <property>
                              <name>E Hardware H</name>
                              <value>0.2</value>
                        </property>
                        <property>
                              <name>C Hardware H</name>
                              <value>0.8</value>
                        </property>
                        <property>
                              <name>E CPP</name>
                              <value>1.0</value>
                        </property>
                        <property>
                              <name>C CPP</name>
                              <value>1.0</value>
                        </property>
                        <property>
                              <name>E Java</name>
                              <value>0.0</value>
                        </property>
                        <property>
                              <name>C Java</name>
                              <value>0.0</value>
                        </property>
                        <property>
                              <name>E Perl</name>
                              <value>0.3</value>
                        </property>
                        <property>
                              <name>C Perl</name>
                              <value>0.8</value>
```

```
                        </property>
                        <property>
                              <name>E VB</name>
                              <value>0.0</value>
                        </property>
                        <property>
                              <name>C VB</name>
                              <value>0.0</value>
                        </property>
                  </items>
            </classInstance>
```

Listagem 7. Instância da Classe “Habilidade”

```
<classInstance>
                  <name>Tabela de Relacionamento</name>
                  <class>Relacionamento</class>
                  <items>
                        <property>
                              <name>Hardware H CPP</name>
                              <value>0</value>
                        </property>
                        <property>
                              <name>Hardware H Java</name>
                              <value>0</value>
                        </property>
                        <property>
                              <name>Hardware H Perl</name>
                              <value>0</value>
                        </property>
                        <property>
                              <name>Hardware H VB</name>
                              <value>0</value>
                        </property>
                        <property>
                              <name>CPP Java</name>
                              <value>1</value>
                        </property>
                        <property>
                              <name>CPP Perl</name>
                              <value>0.2</value>
                        </property>
                        <property>
                              <name>CPP VB</name>
                              <value>0</value>
                        </property>
                        <property>
                              <name>Java Perl</name>
                              <value>0.2</value>
                        </property>
                        <property>
                              <name>Java VB</name>
                              <value>0</value>
                        </property>
                        <property>
                              <name>Perl VB</name>
                              <value>0</value>
                        </property>
```

```
                </items>
            </classInstance>
```

Listagem 8. Instância da Classe “Relacionamento”

```
<classInstance>
                <name>D1</name>
                <class>Recurso</class>
                <items>
                      <property>
                            <name>Hardware H</name>
                            <value>0</value>
                      </property>
                      <property>
                            <name>CPP</name>
                            <value>1</value>
                      </property>
                      <property>
                            <name>Java</name>
                            <value>0.3</value>
                      </property>
                      <property>
                            <name>Perl</name>
                            <value>0.3</value>
                      </property>
                      <property>
                            <name>VB</name>
                            <value>0.3</value>
                      </property>
                      <singleRelation>
                            <name>Aptidao</name>
                            <target>Tabela de Relacionamento</target>
                      </singleRelation>
                      <singleRelation>
                            <name>Conhecimento</name>
                            <target>Conjunto de Habilidades</target>
                      </singleRelation>
                </items>
            </classInstance>
            <classInstance>
                <name>D2</name>
                <class>Recurso</class>
                <items>
                      <property>
                            <name>Hardware H</name>
                            <value>0</value>
                      </property>
                      <property>
                            <name>CPP</name>
                            <value>0.8</value>
                      </property>
                      <property>
                            <name>Java</name>
                            <value>1.0</value>
                      </property>
                      <property>
                            <name>Perl</name>
```

```
                        <value>0.0</value>
                    </property>
                    <property>
                        <name>VB</name>
                        <value>0.3</value>
                    </property>
                    <singleRelation>
                        <name>Aptidao</name>
                        <target>Tabela de Relacionamento</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Conhecimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
                </items>
            </classInstance>
            <classInstance>
                <name>D3</name>
                <class>Recurso</class>
                <items>
                    <property>
                        <name>Hardware H</name>
                        <value>1.0</value>
                    </property>
                    <property>
                        <name>CPP</name>
                        <value>0.3</value>
                    </property>
                    <property>
                        <name>Java</name>
                        <value>1.0</value>
                    </property>
                    <property>
                        <name>Perl</name>
                        <value>0.5</value>
                    </property>
                    <property>
                        <name>VB</name>
                        <value>0.2</value>
                    </property>
                    <singleRelation>
                        <name>Aptidao</name>
                        <target>Tabela de Relacionamento</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Conhecimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
                </items>
            </classInstance>
            <classInstance>
                <name>D4</name>
                <class>Recurso</class>
                <items>
                    <property>
                        <name>Hardware H</name>
                        <value>0</value>
                    </property>
                    <property>
```

```
                        <name>CPP</name>
                        <value>0.3</value>
                    </property>
                    <property>
                        <name>Java</name>
                        <value>0.0</value>
                    </property>
                    <property>
                        <name>Perl</name>
                        <value>1.0</value>
                    </property>
                    <property>
                        <name>VB</name>
                        <value>1.0</value>
                    </property>
                    <singleRelation>
                        <name>Aptidao</name>
                        <target>Tabela de Relacionamento</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Conhecimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
            </items>
        </classInstance>
        <classInstance>
            <name>D5</name>
            <class>Recurso</class>
            <items>
                    <property>
                        <name>Hardware H</name>
                        <value>0.5</value>
                    </property>
                    <property>
                        <name>CPP</name>
                        <value>0.5</value>
                    </property>
                    <property>
                        <name>Java</name>
                        <value>0.8</value>
                    </property>
                    <property>
                        <name>Perl</name>
                        <value>0.0</value>
                    </property>
                    <property>
                        <name>VB</name>
                        <value>0.8</value>
                    </property>
                    <singleRelation>
                        <name>Aptidao</name>
                        <target>Tabela de Relacionamento</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Conhecimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
            </items>
        </classInstance>
```

```
            <classInstance>
                  <name>D6</name>
                  <class>Recurso</class>
                  <items>
                        <property>
                              <name>Hardware H</name>
                              <value>0.3</value>
                        </property>
                        <property>
                              <name>CPP</name>
                              <value>0.8</value>
                        </property>
                        <property>
                              <name>Java</name>
                              <value>0.0</value>
                        </property>
                        <property>
                              <name>Perl</name>
                              <value>0.0</value>
                        </property>
                        <property>
                              <name>VB</name>
                              <value>0.5</value>
                        </property>
                        <singleRelation>
                              <name>Aptidao</name>
                              <target>Tabela de Relacionamento</target>
                        </singleRelation>
                        <singleRelation>
                              <name>Conhecimento</name>
                              <target>Conjunto de Habilidades</target>
                        </singleRelation>
                  </items>
            </classInstance>
```

Listagem 9. Instâncias da classe “Recurso”

```
<classInstance>
                  <name>Producao D1</name>
                  <class>Desenvolvimento</class>
                  <items>
                        <singleRelation>
                              <name>Producao</name>
                              <target>D1</target>
                        </singleRelation>
                        <singleRelation>
                              <name>Requerimento</name>
                              <target>Conjunto de Habilidades</target>
                        </singleRelation>
                  </items>
            </classInstance>
            <classInstance>
                  <name>Producao D2</name>
                  <class>Desenvolvimento</class>
                  <items>
                        <singleRelation>
                              <name>Producao</name>
```

```
                        <target>D2</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Requerimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
                </items>
            </classInstance>
            <classInstance>
                <name>Producao D3</name>
                <class>Desenvolvimento</class>
                <items>
                    <singleRelation>
                        <name>Producao</name>
                        <target>D3</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Requerimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
                </items>
            </classInstance>
            <classInstance>
                <name>Producao D4</name>
                <class>Desenvolvimento</class>
                <items>
                    <singleRelation>
                        <name>Producao</name>
                        <target>D4</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Requerimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
                </items>
            </classInstance>
            <classInstance>
                <name>Producao D5</name>
                <class>Desenvolvimento</class>
                <items>
                    <singleRelation>
                        <name>Producao</name>
                        <target>D5</target>
                    </singleRelation>
                    <singleRelation>
                        <name>Requerimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
                </items>
            </classInstance>
            <classInstance>
                <name>Producao D6</name>
                <class>Desenvolvimento</class>
                <items>
                    <singleRelation>
                        <name>Producao</name>
                        <target>D6</target>
                    </singleRelation>
                    <singleRelation>
```

```
                        <name>Requerimento</name>
                        <target>Conjunto de Habilidades</target>
                    </singleRelation>
                </items>
            </classInstance>
```

Listagem 10. Instâncias da classe “Desenvolvimento”

```
<connections>
    <connection>
      <name>Experiencia</name>
      <className>Desenvolvimento</className>
      <classInstanceItems>
      </classInstanceItems>
      <affects>
        <affect>
          <name>Custo Inicial por Unidade</name>
          <expression>
            <m:math xmlns:m = "http://www.w3.org/1998/Math/MathML">
              <m:apply>
              <m:divide/>
              <m:cn>10</m:cn>
              <m:apply>
                <m:plus />
                <m:apply>
                  <m:times />
                  <m:ci>Producao.Hardware H</m:ci>
                  <m:ci>Requerimento.S Hardware H</m:ci>
                </m:apply>
                <m:apply>
                  <m:plus />
                  <m:apply>
                    <m:times />
                    <m:ci>Producao.CPP</m:ci>
                    <m:ci>Requerimento.S CPP</m:ci>
                  </m:apply>
                  <m:apply>
                    <m:plus />
                    <m:apply>
                      <m:times />
                      <m:ci>Producao.Java</m:ci>
                      <m:ci>Requerimento.S Java</m:ci>
                    </m:apply>
                    <m:apply>
                      <m:plus />
                      <m:apply>
                        <m:times />
                        <m:ci>Producao.Perl</m:ci>
                        <m:ci>Requerimento.S Perl</m:ci>
                      </m:apply>
                      <m:apply>
                        <m:times />
                        <m:ci>Producao.VB</m:ci>
                        <m:ci>Requerimento.S VB</m:ci>
                      </m:apply>
                    </m:apply>
                  </m:apply>
```

```
                </m:apply>
              </m:apply>
              </m:apply>
            </m:math>
          </expression>
        </affect>
      </affects>
    </connection>
  </connections>
```

Listagem 11. Nova fórmula que define “Custo Inicial por Unidade”

```
  <scenarios>
    <files>
      <file>Custo Inicial Baseado nas Habilidades.jymms</file>
    </files>
    <connects>
      <connect>
          <scenario>Custo Inicial Baseado nas Habilidades</scenario>
          <name>Experiencia</name>
          <instance>Producao D1</instance>
      </connect>
      <connect>
          <scenario>Custo Inicial Baseado nas Habilidades</scenario>
          <name>Experiencia</name>
          <instance>Producao D2</instance>
      </connect>
      <connect>
          <scenario>Custo Inicial Baseado nas Habilidades</scenario>
          <name>Experiencia</name>
          <instance>Producao D3</instance>
      </connect>
      <connect>
          <scenario>Custo Inicial Baseado nas Habilidades</scenario>
          <name>Experiencia</name>
          <instance>Producao D4</instance>
      </connect>
      <connect>
          <scenario>Custo Inicial Baseado nas Habilidades</scenario>
          <name>Experiencia</name>
          <instance>Producao D5</instance>
      </connect>
      <connect>
          <scenario>Custo Inicial Baseado nas Habilidades</scenario>
          <name>Experiencia</name>
          <instance>Producao D6</instance>
      </connect>
    </connects>
  </scenarios>
```

Listagem 12. Conexão do cenário “Custo Inicial Baseado nas Habilidades” com as instâncias “Produção”