Um Modelo de Simulação de Processos de Software Baseado em Agentes Cooperativos*

Fábio Augusto das Dores Silva[1]
Rodrigo Quites Reis[1,2]
Carla Alessandra Lima Reis[1,2]
Daltro José Nunes[1]
{fabio, quites, clima, daltro}@inf.ufrgs.br

[1] Universidade Federal do Rio Grande do Sul
Instituto de Informática - PPGC
Porto Alegre, Rio Grande do Sul, Brasil

[2] Universidade Federal do Pará - UFPA
Departamento de Informática
Belém, Pará, Brasil

Resumo

Este artigo apresenta um modelo e uma ferramenta de simulação de processo de software baseada em conhecimento através de agentes inteligentes e cooperativos. O objetivo desta abordagem é validar e refinar um modelo de processo de software antes de sua execução real, prevenindo, desta forma, os custos decorrentes de falhas na modelagem, alocação de recursos e definição de prazos. O modelo AgentProcess simula o comportamento dos desenvolvedores através de agentes cognitivos. Esses agentes possuem habilidades na execução de algumas tarefas e afinidades com outros agentes. Assim, este modelo apresenta um mecanismo mais adequado para a simulação de processos de software pois trata o envolvimento cooperativo dos desenvolvedores nas suas atividades do processo através de uma base de conhecimento distribuída em agentes.
Palavras-chave: Simulação do processo de software; Processo de Software; Sistemas Multiagentes; Sistemas Baseados em Conhecimento.

Abstract

This paper presents a model and a software tool for cooperative multiagent knowledge-based software process simulation. This approach aims to help software process models validation and refinement before real enaction, thus saving the project from modelling, resource allocation and schedule mistakes. The model AgentProcess simulates developers' behaviour by the use of cognitive agents. These agents have some skills and affinities with other agents. Thus, this model presents a more suitable mechanism for software process simulation because it supports the cooperative work between the developers during the enaction of an activity through the use of a distributed knowledge base.
Keywords: *Software Process Simulation; Software Process; Multiagent Systems; Knowledge Based Systems.*

1 Introdução

O processo de desenvolvimento de software, ou processo de software, corresponde ao conjunto de atividades relacionadas que são desempenhadas pelos desenvolvedores desde a concepção até a liberação do produto. Uma forma de analisar e amadurecer tal processo é através da sua descrição, a qual consiste de um modelo de processo de software. A descrição formal de um processo de software é a atividade que permite que o mesmo seja analisado, compreendido e automatizado (executado).

* Trabalho apoiado pelo CNPq e CAPES.

Um Ambiente de Desenvolvimento de Software (ADS) deve proporcionar alto nível de integração entre suas ferramentas e ser capaz de executar um modelo de processo de software através da coordenação dos desenvolvedores na execução de suas tarefas, gerência de alocação de recursos, coleta de métricas, execução automática de algumas atividades e mudança do processo durante sua execução. Os ADS que suportam a modelagem e execução de processos de software são denominados ADS orientados ao processo. EPOS [1], Spade [2] e Adele [1] são alguns dos principais ADSs orientados ao processo encontrados na literatura.

Um modelo de processo é construído através de uma linguagem de modelagem do processo. Antes da execução, o modelo é instanciado através da definição do cronograma, dos desenvolvedores que atuarão no processo e dos recursos a serem alocados. Esta definição pode não ser a mais adequada para o desenvolvimento em questão. Falhas na modelagem, inconsistências nos prazos e na alocação de recursos somente serão detectados quando o processo estiver em andamento. Desta forma, a execução de processos de software pode levar a altos custos em decorrência da falta de uma prévia validação do modelo.

A necessidade de refinar e validar um modelo antes de ser executado em um projeto real tem gerado propostas diversas, dentre as quais destaca-se a simulação de processos de software. Assim, um ADS orientado ao processo deve permitir a verificação e validação dos modelos através de simulação, permitindo a detecção de falhas, inconsistências e comportamento anômalo no processo [4].

A literatura descreve várias iniciativas no sentido de incorporar mecanismos de simulação de processo de software em ADS. Algumas dessas ferramentas enfatizam a simulação evento-discreta, que define um caminho de simulação a partir da geração simbólica de um grande número de eventos alternativos. Segundo [5], uma nova geração de ferramentas para simulação do processo de software vem surgindo com o objetivo de incorporar mecanismos baseados em conhecimento para complementar o trabalho realizado pela simulação evento-discreta.

Este artigo apresenta uma abordagem para simulação de processos de software baseada em conhecimento através da utilização de agentes cooperativos [6]. Nesta abordagem, o comportamento dos desenvolvedores durante o ciclo de vida de software é simulado. Uma das vantagens deste modelo está na capacidade de representar as habilidades dos desenvolvedores e suas afinidades com outros desenvolvedores. Essas informações podem ser obtidas de observações sobre o ambiente de desenvolvimento, levando em consideração resultados obtidos em experiências anteriores. Além do modelo de simulação, será apresentado o funcionamento da ferramenta construída para validar o modelo através da experimentação.

O artigo é organizado como segue. A seção 2 apresenta o contexto da automação de processo de software. A seção 3 apresenta os principais conceitos de simulação de processos de software. A seção 4 apresenta sistemas multiagentes. A seção 5 apresenta o modelo de simulação de processo de software proposto. A seção 6 mostra o protótipo de simulador construído para a experimentação deste modelo. A seção 7 apresenta trabalhos relacionados e comparações com outros ambientes. Na seção 8 são apresentadas as conclusões.

2 Automação do Processo de Software

Esta seção apresenta a terminologia utilizada na construção do modelo de simulação de processos de software proposto neste artigo.

2.1 Processo de Software

O processo de produção de software envolve atividades complexas desempenhadas por pessoas (desenvolvedores) com as mais diversas capacidades. Um modelo de processo de software é uma descrição abstrata do processo de software. Vários tipos de informação devem ser integradas em um modelo de processo de software para indicar quem, quando, onde, como e por que os passos são realizados [7]. Para representar um modelo de processo de software é utilizada uma linguagem de modelagem do processo de software, a qual deve oferecer recursos para descrever e manipular os passos do processo.

Um modelo de processo instanciado ou processo executável é um modelo de processo pronto para execução [8]. Um projeto, segundo [1], é a instância de um processo, com objetivos e restrições específicos e envolve desenvolvedores, prazos, orçamentos, recursos e um processo de desenvolvimento.

2.2 Execução de Modelos de Processo

Com um modelo de processos pronto é possível controlar a execução dos passos de forma automatizada. Na fase de execução de modelos de processo de software devem ser levadas em consideração as questões de coordenação de múltiplos usuários e a interação entre as ferramentas automatizadas e os desenvolvedores.

Um ADS orientado a processos contém uma Máquina de Processo que auxilia na coordenação das atividades realizadas por pessoas e por ferramentas automatizadas. Segundo [2], uma máquina de processos pode: ativar automaticamente atividades sem intervenção humana através de uma integração com as ferramentas do ambiente; apoiar o trabalho cooperativo; monitorar o andamento do processo e registrar histórico da sua execução.

A máquina de processo também deve garantir a execução das atividades na seqüência definida no modelo de processo (fluxo de controle); a repetição de atividades; a informação de *feedback* sobre o andamento do processo; a gerência das informações de processo (incluindo gerência de versões); a coleta automática de métricas; a mudança do processo durante sua execução; a interação com as ferramentas do ambiente e a gerência de alocação de recursos.

3 Refinamento de Modelos de Processo de Software através de Simulação

A simulação pode ser utilizada em planejamentos ou antes da implantação de políticas de desenvolvimento de grandes protótipos, testes ou implementações em um ambiente operacional nos diversos domínios de aplicação do conhecimento humano [9]. O trabalho de [11] define simulação como o processo de desenvolver um modelo matemático ou lógico de um sistema real e então conduzir experimentos baseados em computador, usando o modelo para descrever, explicar e predizer o comportamento de um sistema real. Simulação é, portanto, uma ferramenta poderosa para analisar escalonamentos, algoritmos e políticas.

Na área de processos de software, a simulação é útil no refinamento de modelos de processos, através da percepção da execução do modelo. É possível enriquecer o modelo executável através de informações de prazos e gastos obtidos da simulação. A utilização de simulação permite antever resultados da realização do projeto, que sem a simulação só seriam possíveis de prever em tempo de execução. Dessa forma pode-se aumentar o grau de automatização de uma execução de processos [10].

Nas seções a seguir são apresentadas as características do meta-processo de software adaptado para a fase de simulação e uma análise dos tipos principais de simulação de processos de software.

3.1 Meta-Processo de Software

Um processo de software e seu modelo têm uma natureza evolucionária, devido à necessidade de melhoria e correção contínua e devido à instabilidade do ambiente operacional. Ou seja, o modelo é primeiro estabelecido para representar o mundo real inicial, e, durante seu tempo de vida, é exposto a mudanças causadas por eventos planejados e não-planejados de fora e de dentro da organização [12]. Existe um ciclo de vida para processos de software análogo ao ciclo de vida de produtos de software. As atividades do ciclo de vida de processos de software são chamadas de meta-atividades, e o processo de desenvolvimento e evolução de processos de software é denominado de meta-processo.

A simulação ocupa um papel chave na verificação e validação dos processos definidos, tal como mostrado na figura 1, onde é apresentado um meta-processo para desenvolvimento de software adotado para o modelo de simulação aqui apresentado (adaptado de [3]).

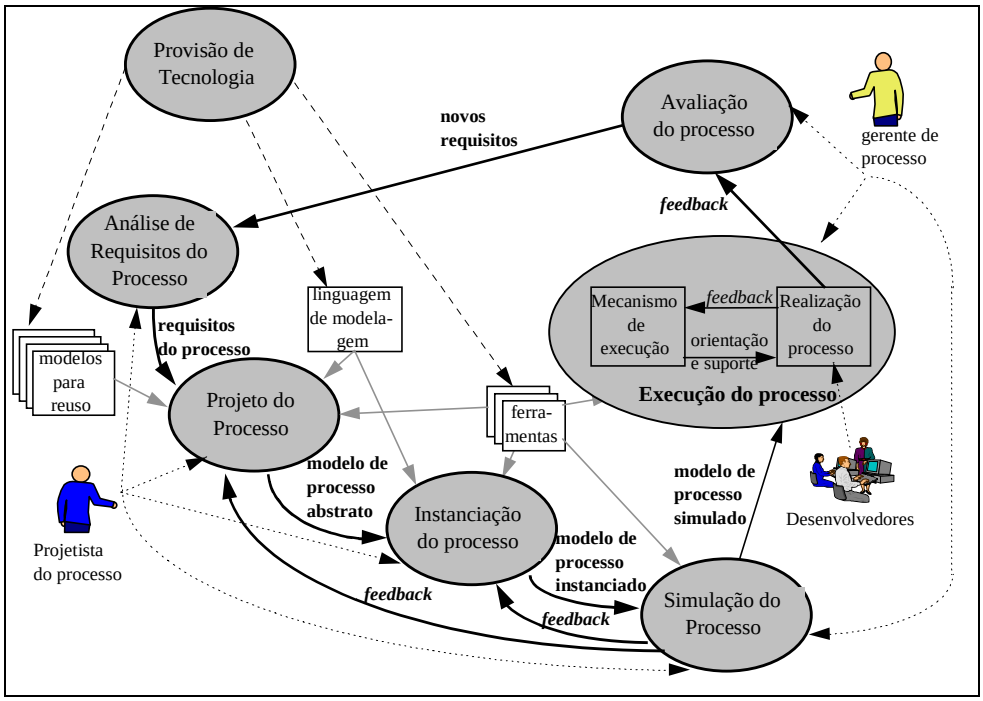

Figura 1. Metaprocesso de software baseado em simulação (adaptado de [3])

No meta-processo é necessária a participação dos agentes humanos que operam na execução do processo. Entretanto, para que o processo seja modelado entra em cena um projetista de processo, que é o responsável por descrever o processo a ser executado e um gerente do processo que deverá acompanhar a execução e a avaliação do processo, analisando seu desempenho. A simulação, como mostra a figura 1, é uma tarefa que geralmente é acompanhada tanto pelo projetista de processo quanto pelo gerente do desenvolvimento.

3.2 Simulação de Processos de Software

A execução simbólica (simulação) de modelos de processos de software permite determinar o caminho e o fluxo de transições de estados intermediários, de maneira que os modelos de processo podem tornar-se persistentes, podendo ser executados novamente, analisados dinamicamente e reconfigurados para múltiplos cenários alternativos [9].

A simulação de processos de software é utilizada no refinamento de um modelo de processo de software. Através da simulação é possível prever prazos e gastos com o

desenvolvimento de sistemas e definir métricas. A simulação é uma eficiente ferramenta de comunicação para mostrar como processos trabalham e como podem ser melhorados. E também pode ser usada em treinamento individual, através da interação dos participantes com modelos executando em tempo real para perceberem os efeitos de suas decisões.

Vários ambientes estão sendo propostos para tratar simulação de processos de software. No entanto, para os nossos propósitos convém destacar a simulação baseada em conhecimento e a simulação discreta, descritas sucintamente nas seções a seguir.

3.2.1 Simulação baseada em conhecimento

O desenvolvimento de sistemas de simulação baseados em conhecimento resulta da combinação de simulação com tecnologias de sistemas baseados em conhecimento. O principio é suprir o software com a maior quantidade possível de conhecimento e experiência sobre o domínio do problema [13].

A simulação baseada em conhecimento permite avaliar um modelo de processo de software através de "raciocínio" computacional na forma de inferência utilizando regras de produção. O "raciocínio" é possível, porque as regras monitoram o valor dos objetos, atributos e relações armazenados na base de conhecimento [5].

Diversos trabalhos tratam da simulação baseada em conhecimento, cada um apresentando diferentes conceitos e pontos de vista. Meng [13] indica que um sistema baseado em conhecimento armazena fatos e heurísticas (regras) de um domínio de aplicação particular. A base de conhecimento pode ser atualizada de acordo com as necessidades. Uma máquina de inferência estabelece como e quando os fatos e regras da base de conhecimento devem ser aplicados na resolução de problemas. A máquina de inferência em essência é um programa de controle que decide qual regra trilhar, qual alternativa eliminar e qual atributo encontrar aonde as estratégias de controle são *forward chaining* e *backward chaining*.

3.2.2 Simulação evento-discreta

A natureza da simulação evento-discreta é a de que o estado do modelo se altera somente em conjuntos de pontos discretos, mas possivelmente aleatórios, conhecidos como "tempos de evento". Duas ou mais "unidades de tráfego" (transações), sempre serão manipuladas em um mesmo ponto de tempo. Tais movimentos "simultâneos" de tráfego em um ponto de tempo são tratados como unidades de tráfego serialmente naquele ponto [14].

A simulação evento-discreta fornece a possibilidade de analisar dinamicamente diferentes amostras de parâmetros em instâncias de processos de software. Ao contrário da simulação baseada em conhecimento, a evento-discreta só trabalha com modelos instanciados. Alta densidade de tráfego e gargalos de congestionamentos de processos podem ser avaliados de acordo com taxas alternativas e intervalos de serviços [5].

Estão disponíveis vários pacotes comerciais de simulação evento-discreta, alguns com animação visual de execução de processos ou resultados. Estas ferramentas geram "filmes" de desenvolvimento de software ou processos de negócio, sendo que a animação pode ser modificada, re-executada e vista como outra simulação. Exemplos são o GPSS/H [15] e SIMAN V [16].

3.2.3 Simulação baseada em conhecimento *versus* evento-discreta

A motivação para se usar um simulador baseado em conhecimento vem da necessidade de se trabalhar com simulação contínua, em que informações possam ser obtidas em qualquer momento do processo. Simulação baseada em conhecimento permite a representação dos

componentes do ambiente no simulador de forma fiel ao seu real funcionamento. A simulação evento-discreta é mais útil quando o objetivo é verificar escalonamentos de processos e a ocorrência de filas em gargalos de tráfego.

Levando em consideração a utilização de simulação baseada em conhecimento e simulação evento-discreta, [5, 6] apresentam como principal conclusão o fato de que estes dois paradigmas de simulação são complementares, não estritamente alternativos. A definição rigorosa de meta-modelos de processo é elemento-chave para a integração de simulação no ciclo de vida de um processo, com qualquer abordagem de simulação adotada.

4 Sistemas Multiagentes

A Inteligência Artificial Distribuída (IAD) tornou-se nos últimos anos um domínio de pesquisa bastante promissor [17]. Enquanto a IA clássica usa como modelo de inteligência o comportamento individual humano, o modelo de inteligência em IAD é baseado no comportamento social. Uma abordagem deste tipo torna-se desejável para resolver problemas grandes e complexos, que requeiram conhecimento de vários domínios de conhecimento distintos, e que algumas vezes envolvam coleta de dados fisicamente distribuídos.

Dado um determinado sistema, denomina-se **agente** cada uma de suas entidades ativas. O conjunto de agentes forma uma **sociedade**. As entidades passivas são designadas pelo termo **ambiente**. Um agente raciocina sobre o ambiente, sobre os outros agentes e decide racionalmente quais objetivos deve perseguir, quais ações deve tomar, etc. Um agente é uma entidade real ou virtual; está em um ambiente; percebe o ambiente; é capaz de agir no ambiente; pode se comunicar com outros agentes; e tem um comportamento autônomo.

Os sistemas multiagentes dividem-se em duas classes principais: Agentes **Cognitivos** e Agentes **Reativos**. A abordagem reativa [18] baseia-se na idéia de que agentes com ações elementares podem realizar trabalhos complexos. Um exemplo clássico é a colônia de formigas, cujo trabalho individual não é inteligente mas o resultado é bastante complexo.

Segundo [17], nos sistemas multiagentes reativos não há representação explícita do conhecimento nem do ambiente; não há memória das ações; a organização é etológica, ou seja, similar a dos animais; e existe grande número de membros. Os sistemas multiagentes cognitivos são baseados em modelos organizacionais humanos, como grupos, hierarquias e mercados. Nestes sistemas os agentes mantêm uma representação explícita de seu ambiente e dos outros agentes da sociedade; podem manter um histórico das interações e ações passadas; a comunicação entre os agentes é direta, através de mensagens; seu mecanismo de controle é deliberativo, ou seja, os agentes raciocinam e decidem seus objetivos, planos e ações; seu modelo de organização é sociológico e uma sociedade contém poucos agentes.

Apesar da classificação, sistemas multiagentes podem não ser totalmente cognitivos ou reativos. Um sistema pode ser uma mistura dos dois para atender a solução de um problema. Os agentes inteligentes (como também são chamados na literatura) podem ser construídos e manipulados através das mesmas técnicas de representação do conhecimento, raciocínio e aprendizado da IA. O modelo de simulação de processos de software apresentado na seção 5 trabalha com agentes inteligentes que podem ser classificados de cognitivos.

5 O Modelo de Simulação de Processo de Software *AgentProcess*

As ferramentas tradicionais de simulação de processo de software descritas na literatura não tratam de forma efetiva o caráter cooperativo do processo de desenvolvimento,

isto é, a possibilidade de existirem atividades que são realizadas por vários profissionais interagindo entre si para desempenhar a sua tarefa [10]. A partir disto, pode-se destacar nestas ferramentas algumas deficiências:

- A simulação não leva em consideração as aptidões e preferências dos profissionais existentes na organização de desenvolvimento de software;
- A granularidade dos eventos observados por estas ferramentas é grossa, ou seja, observa eventos que ocorrem somente no nível das atividades (com pouca ou nenhuma observação dos fenômenos que ocorrem no interior das atividades);
- Pouca assistência é fornecida ao gerente do desenvolvimento de software na seleção dos desenvolvedores mais indicados para ocupar cada uma das atividades do processo.

Com o objetivo de contornar estas deficiências e apresentar novas soluções para o problema da simulação de processo de software foi construído o modelo ***AgentProcess*** (Simulação de Processo de Software baseado em Agentes Cooperativos), desenvolvido no contexto do projeto PROSOFT [19]. O modelo *AgentProcess* foi desenvolvido a partir do modelo de processo apresentado em [3] e [20], onde se encontram discussões e comparações com propostas similares. Este modelo é apresentado na figura 2 (um diagrama de classes na notação UML) e suas características são detalhadas nas seções subsequentes.

5.1 Características do modelo *AgentProcess*

O modelo de simulação de processos *AgentProcess* propõe uma forma de testar modelos de processos de software usando agentes inteligentes (atuando no papel de desenvolvedores). Dado um modelo de processo instanciado (com atividades definidas, desenvolvedores e recursos alocados para as atividades, cronograma inicial, etc.), o modelo de simulação disponibiliza atividades para os agentes-desenvolvedores (através de uma agenda do agente), e estes utilizam seu conhecimento armazenado para executar as atividades nas quais eles estão envolvidos.

Durante a simulação são levadas em consideração as habilidades dos agentes desenvolvedores, as afinidades entre os desenvolvedores e os recursos disponíveis. As características principais do modelo *AgentProcess* são apresentadas nas seções a seguir.

5.1.1 Modelagem de Processos

Antes da simulação, o modelo de processo deve ser descrito. O modelo *AgentProcess* permite descrição de processo instanciado, ou seja, com desenvolvedores e recursos alocados bem como um cronograma previsto inicial. A descrição do processo é feita através da definição das atividades que compõem o mesmo. A figura 3 mostra a decomposição de atividades em processo de software no modelo *AgentProcess*, onde as setas indicam dependência entre atividades e as linhas pontilhadas indicam a decomposição de uma atividade em um novo processo. Como mencionado anteriormente, as características de um modelo de processo no *AgentProcess* baseiam-se nas apresentadas em [4,23], onde encontram-se mais detalhes e comparações com outras abordagens.

Um processo de software corresponde a um conjunto de atividades. Cada atividade é executada por um ou vários agentes-desenvolvedores (atividade cooperativa), necessita de recursos, possui cronograma previsto e cronograma real, possui um conjunto de atividades das quais ela depende, artefatos que produz e um *script* onde é descrita a tarefa a ser feita. O *script* pode ser um novo processo, uma atividade automática (executada por uma ferramenta sem intervenção de desenvolvedores) ou ainda uma descrição da tarefa a ser realizada.

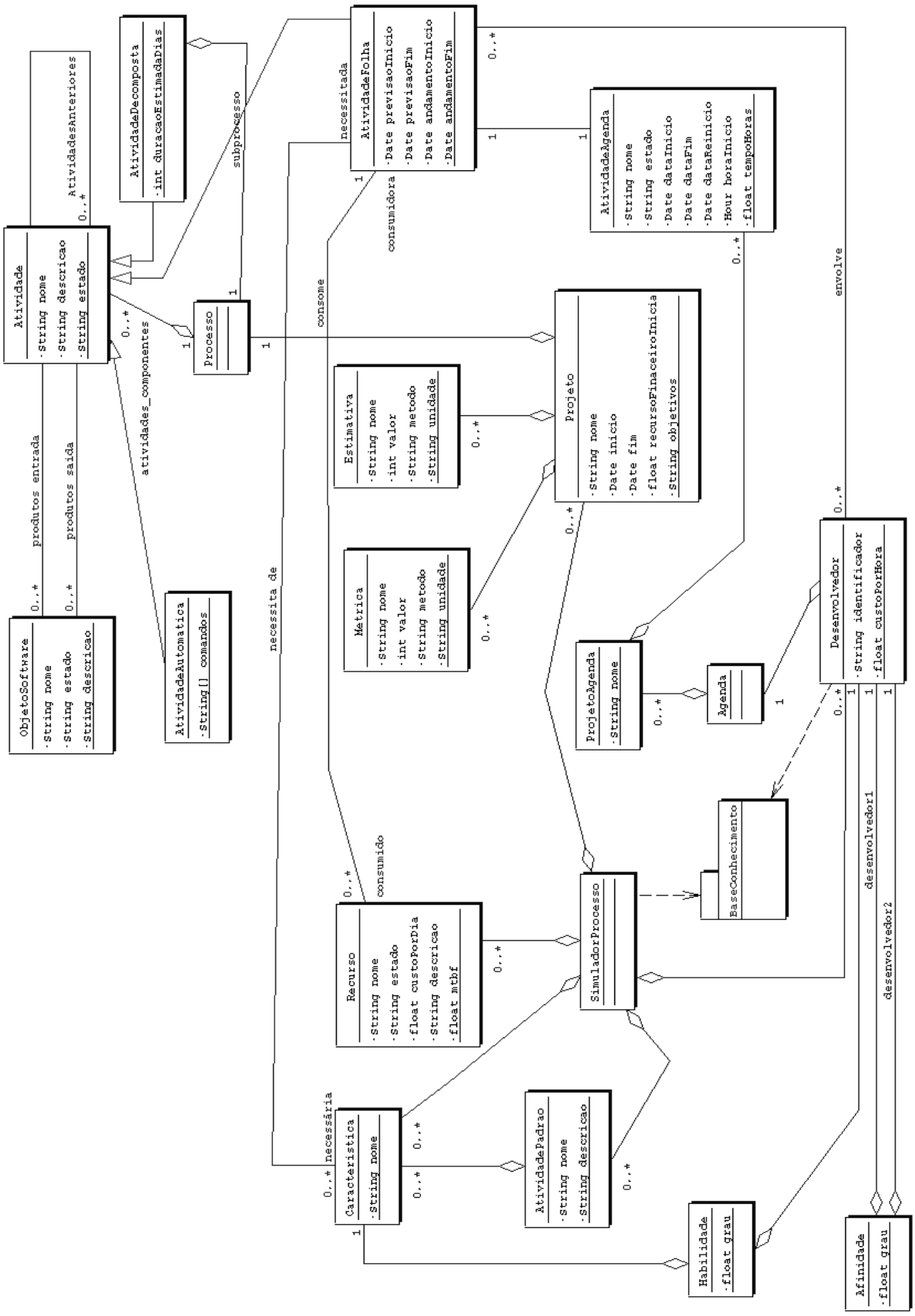

Figura 2. Modelo de classes *AgentProcess*

Toda atividade possui um estado que deve ser manipulado somente pelo simulador do processo. Os estados que uma atividade pode assumir são: **Esperando** (as anteriores ainda não concluíram); **Pronta** (as anteriores já concluíram); **Ativa** (sendo realizada); **Parada** (em pausa) e **Completa** (atividade concluída).

A fim de facilitar a definição de processos, existe um conjunto de atividades-padrão, que são componentes de processo pré-definidos e prontos para reutilização. Uma atividade-padrão possui nome, descrição e um conjunto de características nas quais seus desenvolvedores devem ser hábeis. Isto facilita a reutilização de processos uma vez que não é necessário redefinir todas as atividades de um processo quando ele é criado, pois pode-se partir da lista de atividades-padrão.

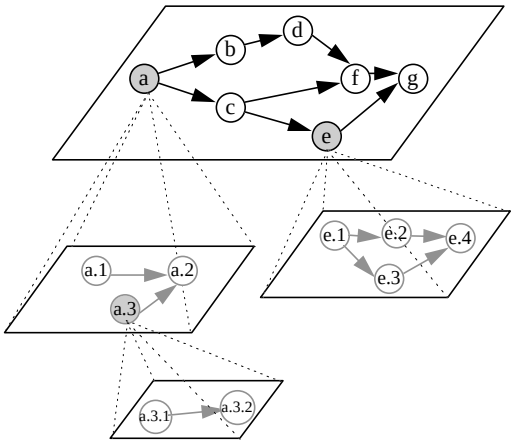

Figura 3. Decomposição de Atividades de um processo de software [3]

Para cada atividade de processo de software é definido um conjunto de características que os desenvolvedores deverão assumir para realizá-la. Por exemplo, uma atividade pode ter necessidade da característica "análise orientada a objetos". Isto significa que os desenvolvedores desta atividade deverão ser bons analistas de sistemas com domínio de metodologias orientadas a objetos, ou a atividade não será bem realizada.

5.1.2 Agentes-Desenvolvedores

Os desenvolvedores envolvidos com as atividades serão representados no simulador como agentes inteligentes que simularão o comportamento de desenvolvedores reais. Devido a essa característica, o modelo de simulação é um sistema multiagente. No modelo aqui apresentado, esses agentes são denominados *Agentes-Desenvolvedores*.

Cada agente-desenvolvedor possui, dentre outros atributos, um valor de hora de trabalho e um conjunto de habilidades com grau variando de 0 a 1 para a capacidade do agente na habilidade. Por exemplo, um agente tem habilidade = 0,6 para a característica "análise orientada a objetos". Isto significa que ele tem algum conhecimento sobre o assunto mas não é um especialista na área.

As habilidades dos agentes influenciam a simulação e podem ser manipuladas a qualquer momento. O gerente de processo pode alterar esses valores para fazer com que um agente do simulador fique mais parecido com um desenvolvedor real. Portanto, a modelagem dos agentes do simulador deve ser feita a partir de dados reais.

Assim como as pessoas envolvidas no desenvolvimento de software possuem afinidades com outras pessoas, os agentes-desenvolvedores do modelo *AgentProcess* também terão. Cada agente possui um valor de 0 a 1 para a afinidade com cada outro agente do modelo. Por exemplo, o agente X tem afinidade de 0,9 com o agente Y. Isto significa que quando eles realizarem uma atividade cooperativa, terão maior probabilidade de terminar a atividade com eficiência. Agentes com pouca afinidade podem atrasar o processo, da mesma forma que em um desenvolvimento real.

Cada agente-desenvolvedor possui uma agenda com as atividades que ele deve desenvolver. O agente *Simulador de Processo* é quem adiciona nas agendas dos desenvolvedores as atividades que já podem ser realizadas. Um agente-desenvolvedor pode estar trabalhando em vários projetos ao mesmo tempo (vários processos podem estar sendo

simulados). Existe uma agenda para cada projeto em que o agente participa. O comportamento do agente-desenvolvedor será apresentado posteriormente.

5.1.3 Simulador do Processo

O agente inteligente encarregado da simulação do processo, coleta de métricas e coordenação de outros agentes é chamado *Simulador do Processo*. Este agente constitui a parte essencial do modelo. Suas características são as mesmas de uma máquina de execução de processos e foi baseado no Gerenciador de Processos de Software descrito em [20].

Durante a simulação de um processo de software, o *Simulador do Processo* coleta métricas sobre o atraso das atividades, quantidade de agentes ociosos, agentes com carga de trabalho alta, utilização dos recursos, dentre outras métricas. Essas métricas são armazenadas no projeto. Cada projeto também possui estimativas sobre o tempo de desenvolvimento previsto, valor do desenvolvimento, custos com desenvolvedores, dentre outras estimativas.

O modelo de processo pode ser modificado durante a simulação. Por isso o Simulador deve atualizar as estimativas e métricas periodicamente. Isto permite que o modelo de simulação seja usado até que o processo esteja dentro do prazo e orçamento permitidos, bem como esteja livre de falhas e inconsistências.

O Simulador de Processo atua em todos os componentes do modelo *AgentProcess*. Através da camada de Interface, o usuário da ferramenta baseada neste modelo pode manipular os agentes-desenvolvedores, os projetos com processos a serem simulados e os recursos disponíveis. Na figura 4 é apresentada a arquitetura de alto nível do modelo *AgentProcess*.

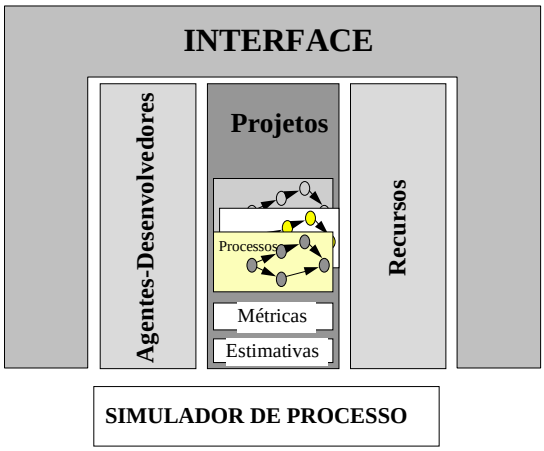

Figura 4. Arquitetura do *AgentProcess* .

5.2 Comportamento dos Agentes

Nesta seção será apresentado o comportamento dos agentes representados no modelo. Apesar dos agentes possuírem um comportamento básico, relacionado com a sua função no modelo, foi disponibilizada uma base de conhecimento que deve ser consultada pelos agentes a fim de definir mais precisamente seu comportamento. Desta forma, é possível acrescentar novas regras de comportamento para melhorar a simulação do processo. Na simulação existem dois tipos de agentes inteligentes atuando, que são os Agentes-Desenvolvedores e o Simulador de Processo. O comportamento desses agentes é apresentado nas seções a seguir.

5.2.1 Comportamento do Agente-Desenvolvedor

O agente-desenvolvedor possui uma agenda com as atividades que ele deve realizar. Este é o principal mecanismo de comunicação entre esse agente e o simulador do processo. Para realizar uma atividade da sua agenda, o agente-desenvolvedor leva em consideração sua habilidade para realizar a tarefa e sua afinidade com os membros da equipe responsável pela atividade. Como a execução da atividade é simbólica, o agente apenas aguarda o período de tempo necessário para realizar a atividade e depois informa que a concluiu. Esse período de tempo nem sempre será o mesmo período previsto para a atividade pois o agente pode atrasar ou mesmo adiantar a execução da sua tarefa. Em uma atividade cooperativa (com vários

agentes) o tempo total da atividade será o tempo do agente mais lento.

Se o agente possuir mais de uma atividade na sua agenda, então ele deve optar por uma usando como critério o menor tempo previsto de conclusão da atividade. Para calcular o tempo de execução da atividade o agente baseia-se no tempo previsto da atividade e nas suas habilidades e afinidades para realizá-la. São calculadas a habilidade geral do agente para a atividade (valor de 0 a 1 baseado nas habilidades do agente para cada característica da atividade) mais a afinidade geral do agente (valor de 0 a 1 obtido através das afinidades com cada agente do grupo). Após esse cálculo, o agente possui um valor de afinidade (*afin*) e um valor de habilidade (*hab*) de onde são obtidos os acréscimos e decréscimos de tempo de realização da atividade.

Um algoritmo que determina o comportamento básico do agente-desenvolvedor é mostrado na figura 5. O cálculo da habilidade (*hab*) e da afinidade (*afin*) do agente para uma atividade é mostrado na figura 6. Os acréscimos e decréscimos de tempo (*tempo_hab* e *tempo_afin*) da realização de uma atividade são obtidos das tabelas A e B (os quais devem ser ajustados através de experimentação).

```
Enquanto existe atividade pronta na agenda:
Informa início de atividade para simulador do processo
(no caso de existirem várias: escolhe a mais rápida)
        Obter valor para hab    /* verificar habilidades x características da ativ.
        Se atividade for cooperativa
          Obter valor para afin  /*verificar afinidades com outros agentes
        Obter tempo_hab                 /* tabela A
        Obter tempo_afin                /* tabela B
        Tempo total = tempo previsto + tempo_hab + tempo_afin
        Delay(tempo total)              /* aguarda pelo período de tempo calculado
        Termina atividade
```

Figura 5. Comportamento do Agente-Desenvolvedor.

Cálculo da habilidade do agente para uma atividade / Obter valor para *hab*

Características da atividade: $C_1, C_2,...,C_n$

Habilidades do Agente para a atividade: $H_1, H_2,..., H_m$

onde $H_i = (C_i, Valor_i)$ /* valor de 0 a 1

$$Hab = (valor_1 + valor_2 +, ... , + valor_m) / n$$

Cálculo da Afinidade do agente para uma atividade de grupo/ Obter valor de *afin*

Agentes que trabalham na mesma atividade: $A_1, A_2,..., A_n$

Afinidades do agente com os outros: $afin_1, afin_2,..., afin_n$

Onde $afin_i$ = afinidade entre o agente e o agente A_i

$$Afin = (afin_1 + afin_2 + ... + afin_n) / n$$

Figura 6. Cálculos de Habilidade e Afinidade Gerais do Agente.

Tabela A. Cálculo da variação do tempo em função da habilidade do agente.

Habilidade Geral do Agente (*HAB*)	Tempo_hab =
hab < 0.3	+ 30% do tempo_previsto
hab < 0.4	+ 20% do tempo_previsto
hab < 0.6	= 0
hab < 0.7	- 10% do tempo_previsto
hab < 0.8	- 20% do tempo_previsto
hab < 0.9	- 30% do tempo_previsto
Hab >= 0.9	- 45% do tempo_previsto

Tabela B. Cálculo da variação do tempo em função da afinidade do agente.

Afinidade Geral do Agente (*AFIN*)	Tempo_afin =
Afin < 0.5	+30% do tempo_previsto
Afin < 0.6	+ 10% do tempo_previsto
Afin >= 0.6	= 0

5.2.2 Comportamento do Agente Simulador do Processo

O agente Simulador do Processo é único no modelo e é o responsável por toda a execução do processo instanciado e pela coleta de métricas e estimativas. O comportamento básico deste agente é mostrado na figura 7.

```
Para as atividades no estado "Esperando": Verificar a conclusão das atividades anteriores
Repetir
Coletar métricas
Para as atividades no estado "Pronta":
    Se a atividade é automática então altera estado da atividade para "Completa"
    Para as atividades-folha que estão prontas:
        Colocar as atividades nas agendas dos agentes responsáveis
    Aguardar solicitações dos agentes:
        Quando um agente solicitar "iniciar atividade":
          alocar recursos para a atividade
          alterar estado da atividade para "ativa"
          atualizar agenda do agente (ativar ativ. na agenda e alterar data e hora de início)

        Quando um agente solicitar "termina atividade"
          alterar agenda do agente (alterar ativ. p/ completa e anotar data e hora de fim)
          verificar término da atividade real (se outros agentes da atividade também já concluíram)
          Se atividade do processo terminou:
                Desalocar recursos
                alterar estado da atividade para "completa"
```

Figura 7. Comportamento do Agente *Simulador do Processo*.

6 Ferramenta *SimAgentProcess*

A ferramenta de simulação *SimAgentProcess* foi desenvolvida para validar o modelo *AgentProcess* descrito na seção anterior. A ferramenta objetiva treinar gerentes de processo de software, fornecendo para estes possíveis resultados de comportamento do processo, diante de características de componentes do processo normalmente desprezadas nas ferramentas de simulação convencionais. As características do funcionamento desta ferramenta e sua arquitetura são resumidos nas seções seguintes.

6.1 Funcionamento da ferramenta

O simulador permite que o seu usuário "carregue" um modelo de processo de software com informações que levam à realização de uma simulação automatizada da execução do modelo. A ferramenta de simulação emite relatórios de **verificações estáticas** e **dinâmicas**. As verificações estáticas são realizadas antes do início do processo de simulação propriamente dito e apresentam sugestões de alterações no modelo baseadas na base de conhecimento. O exemplo da figura 8 apresenta um exemplo de sugestão proposta para o usuário, enfatizando o mecanismo de explanação desenvolvido.

O relatório das verificações dinâmicas, por sua vez, corresponde à execução do processo com os dados da base de conhecimento acerca do histórico dos desenvolvedores desempenhando atividades que consomem recursos. O processo de simulação pode ser efetuado no modo "Encenação" (onde os eventos da simulação ocorrem em uma velocidade que pode ser ajustada pelo usuário – utilizando um *clock* de simulação tal como adotado por [21]) ou "Instantâneo" (quando todos os eventos possíveis são gerados e apresentados na forma de um relatório final e consolidado para o usuário).

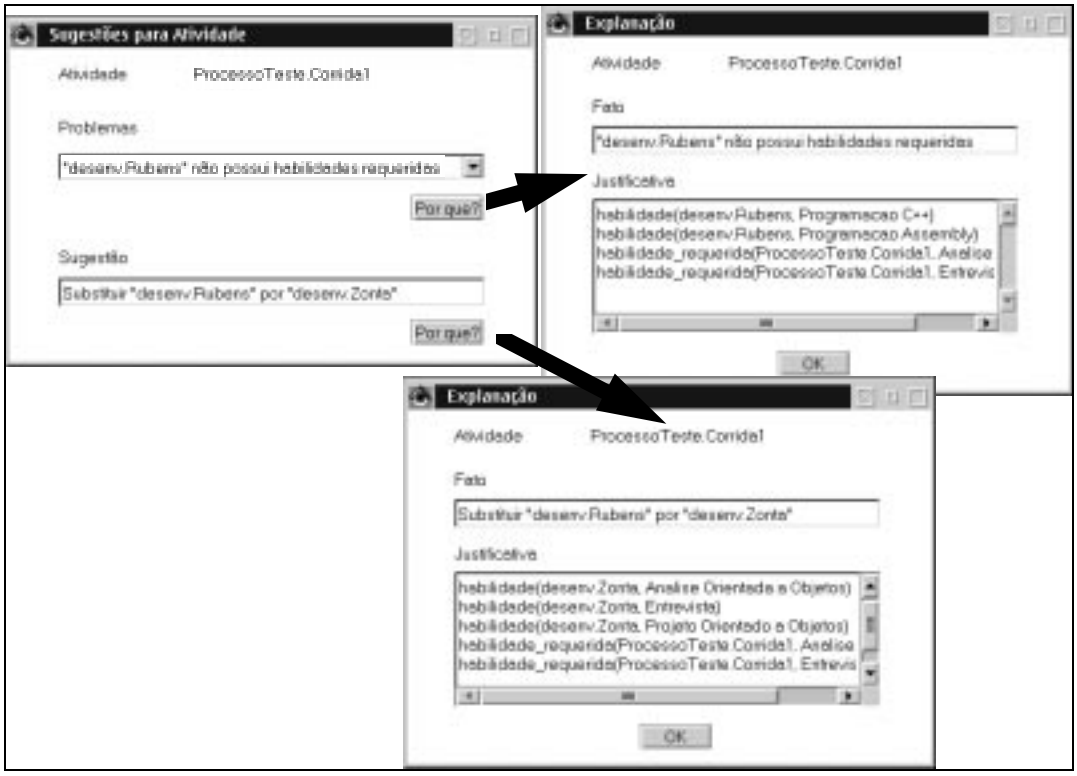

Figura 8. Mecanismo de explanação de sugestões de SimAgentProcess durante a verificação estática do modelo simulado

6.2 Aspectos de implementação da ferramenta

Um primeiro protótipo do simulador *SimAgentProcess* foi desenvolvido em Java 1.1.7 com uso do ambiente IBM VisualAge for Java 2.0 (Professional Edition) na forma de aplicativos que utilizam interface gráfica com usuário através do pacote AWT.

O *framework* CIAgent [6] foi especializado para a construção da base de conhecimento e do mecanismo de inferência dos agentes. O processo de simulação é iniciado através do disparo de todos os agentes de simulação (que atuam concorrentemente no processo de "realização" das atividades). A simulação é encerrada quando todos os agentes possuem suas agendas de atividades a realizar "vazias" e todas as atividades do processo estão com o estado "Completa".

7 Trabalhos Relacionados

Esta seção apresenta uma descrição de ambientes que permitem a simulação de processos de software, bem como comparações com o modelo proposto neste artigo.

7.1 Articulator

O Articulator [27,6] é um ambiente que utiliza simulação baseada em conhecimento através de agentes inteligentes. O componente de simulação apresenta aos usuários as regras de produção utilizadas para a simulação, os eventos gerados a partir destas regras e uma descrição das regras que conflitaram quando disparadas (e qual a política adotada na resolução de conflitos).

Assim como no *SimAgentProcess*, no Articulator o gerente pode executar suas tarefas de gerenciamento de projeto de software. Tarefas são modeladas de acordo com a estrutura de

precedência de sub-tarefas ou passos especificados na instância do processo modelado. Agentes e tarefas podem usar ou consumir tempo simulado, recursos financeiros e outros recursos durante os passos de simulação.

No Articulator os agentes podem ser individuais ou coletivos (uma organização por exemplo). O comportamento dos agentes durante a simulação depende da situação, ou seja, o agente possui alguns atributos (como experiência, especialidade em algumas tarefas, etc.) mas seu comportamento também é derivado da base de conhecimento do simulador.

Além dessas características, o modelo de simulação do Articulator é diferente do *AgentProcess* principalmente em relação à importância dada às habilidades do agente e às afinidades com outros agentes. Além disso, no *AgentProcess* cada agente possui um comportamento diferenciado pois a base de conhecimento utilizada é distribuída, enquanto no Articulator a base de conhecimento é única.

7.2 Modelo DSS

O sistema apresentado por [23], denominado DSS (*Decision Support System*) possui dois subsistemas, um sistema especialista para sugerir alternativas de processos e um subsistema que permite ao usuário avaliar a alternativa selecionada, através da análise das saídas do simulador. O simulador de processos é desenvolvido utilizando duas técnicas, a técnica de paradigma de modelagem dinâmica e a modelagem de eventos-discretos.

DSS possui uma base de conhecimento para ajudar os usuários a definir o processo de desenvolvimento de software de uma forma dirigida a qualidade do produto. O sistema especialista é baseado em lógica *fuzzy*. Quando um modelo é simulado, a dinâmica do modelo causa alteração dos valores de atributos sobre o tempo a cada passo da execução. Os atributos (custo, qualidade, etc.) podem ser traçados graficamente de forma que o gerente possa avaliar a performance do modelo, relativo ao projeto real.

Assim, a ênfase deste ambiente é utilizar a simulação de processo como uma ferramenta para determinar atributos de qualidade no desenvolvimento de software. Pouca atenção é dada às possíveis causas dos problemas encontrados e sobre outros aspectos internos dos processos.

7.3 SESAM - A Software Project Simulator

O ambiente SESAM [24] simula um projeto de software através de um jogo onde o usuário é o gerente do projeto. Neste jogo o objetivo é testar o gerente e fazê-lo ganhar o jogo se o projeto obtiver sucesso. O usuário/gerente deve decidir tudo sobre o projeto, desde a alocação de desenvolvedores para atividades até a solução de problemas que fazem o projeto parar, tais como: falha nos computadores, doenças dos desenvolvedores, etc.

A simulação (baseada em conhecimento) é armazenada em um *log* e o resultado é traduzido para um *score* do jogador. A idéia do jogo é influenciar um comportamento positivo em gerentes de projeto através da experiência.

8 Conclusões e Atividades em Andamento

A simulação de processo de software é uma importante área nova relacionada à automação do processo de software. Além de beneficiar sobremaneira o entendimento do processo de software, permite o aperfeiçoamento contínuo dos modelos construídos. Neste sentido, este artigo apresentou uma contribuição para esta área através da utilização de agentes cognitivos com o objetivo de simular o comportamento dos desenvolvedores durante o ciclo de vida de software.

Segundo [5], os sistemas multiagentes podem revolucionar o desenvolvimento e utilização da simulação de processos de software baseada em conhecimento. Verifica-se que a simulação baseada em agentes incorpora os benefícios da utilização das ferramentas de simulação baseada em conhecimento e daquelas baseadas em evento-discreta permitindo, por exemplo, observar os padrões de interação entre os desenvolvedores (recurso de sistemas baseados em conhecimento), e a identificação de gargalos e a encenação de processos (recursos típicos de sistemas baseados em simulação evento-discreta). Assim, apesar do objetivo principal no desenvolvimento da ferramenta *SimAgentProcess* ter sido o de complementar o trabalho de ferramentas de simulação "tradicionais", algumas observações iniciais mostram que é possível combinar a análise qualitativa (fornecida por sistemas baseados em conhecimento) com alguns mecanismos de análise quantitativa.

Um módulo está sendo construído para permitir a extração de informações de um ADS orientado ao processo, acerca do histórico real dos desenvolvedores em projetos que foram executados anteriormente (estabelecendo a base de conhecimento do simulador). Atualmente, todas as informações relativas as habilidades e afinidades dos desenvolvedores devem ser fornecidas pelo usuário.

Estudos adicionais estão sendo realizados com o intuito de obter (através de experimentação) índices realísticos, ou seja, mais precisos em relação a performance do desenvolvedor nas atividades de acordo com os cálculos das habilidades e afinidades dos mesmos. Para obter esses índices, está sendo estudada a possibilidade de utilização de lógica *fuzzy* no modelo, assim como ocorre no modelo DSS apresentado na seção 7.2.

Além disso, a ferramenta de simulação pode ser adaptada para funcionar como um jogo educacional de engenharia de software - objetivando atuar como um mecanismo auxiliar ao professor no ensino de métodos e técnicas de engenharia de software e gerenciamento de projetos.

Um aspecto a ser ressaltado sobre o modelo de simulação proposto é a ênfase nos fatores tempo e custo do desenvolvimento. Investigação mais profunda precisa ser realizada para determinar mais fatores para o cálculo de atividade além de habilidade e afinidade. Um fator muito importante para validar um processo de software é relacionado com a qualidade dos artefatos de software produzidos. Entretanto, este fator ainda não está sendo explorado satisfatoriamente nos ambientes de simulação de processo atuais, pois envolve a análise de como os diversos aspectos tecnológicos e gerenciais afetam o processo e os produtos. Assim, investigações adicionais são necessárias para determinar a real implicação destes aspectos no processo de simulação do desenvolvimento de software.

9 Referências

[1] FINKELSTEIN, A. (Ed.). **Software Process Modelling and Technology**. Taunton: Research Studies Press, 1994.

[2] BANDINELLI, S. et al. Process Enactment in Spade. In: European Workshop on Software Process Technology, 2., 1992, Norway. **Proceedings...** (http://www.elet.polimi.it/section/compeng/se/swproc/public.html)

[3] LIMA, Carla. **Um Gerenciador de Processos de Software para o Ambiente PROSOFT**. CPGCC-UFRGS. Março, 1998. Dissertação de mestrado. (http://www.inf.ufrgs.br/~prosoft)

[4] MI, P. e SCACCHI, W. **A Knowledge-Based Environment for Modeling and Simulating Software Engineering Process.** IEEE Trans. Knowledge and Data Eng., v.

2, n. 3, p. 283-294, Sept 1990.

[5] SCACCHI, W. **Experience with software process simulation and modeling.** To appear in Journal of Systems and Software, 1999. (http://www.usc.edu/dept/ATRIUM/Papers/SPS.html)

[6] BIGUS,J.; BIGUS,J.**Constructing Intelligent Agents with Java.** New York:Wiley,1998.

[7] LONCHAMP, J. A Structured Conceptual and Terminological Framework for Software Process Engineering. In: International Conference on the Software Process, 2. **Proceedings...** Berlin: IEEE Press, 1993.

[8] FEILER,P.;HUMPHREY, W. Software Process Development and Enactment:Concepts and Definitions. In:International Conference on the Software Process, 2. **Proceedings...** Berlin:IEEE Press,1993.

[9] RUSSEL, E. **Building Simulation Models with SIMSCRIPT II.5.** CACI Products Company, 1990.

[10] SILVA, F.A.D. **Estudo sobre execução, validação e simulação de processos de software.** Trabalho Individual. PPGC-UFRGS. 1999.

[11] HOOVER, S.V. , PERRY, R. F. **Simulation a problem-solving approach.** Reading, Massachusetts: Addison-Wesley. 1989.

[12] NGUYEN, M.;WANG, A. **Total Software Process Model in Epos**. (http://www.idt.unit.no/~epos/Papers)

[13] MENG, T. **Application of AI Techniques in Simulation.** Submitted to NUS School of Computing. 1998. (http://www.comp.nus.edu.sg/~teoym/ic52z1/ai.html)

[14] SCHRIBER, T.; BRUNNER, D. Inside simulation software: how it works and why it matters. 1995 Winter Simulation Conference. **Proceedings...** 1995.

[15] BANKS, J.; CARSON, J.S.; SY, J.N. **Getting Started with GPSS/H**, second edition. Annadale, Virginia: Wolverine Software Corporation. 1995.

[16] BANKS,J. et.al.**Introduction to SIMAN V and CINEMA V**. John Wiley & Sons. 1995.

[17] ALVARES, L.; SICHMAN, J. **Introdução aos Sistemas Multiagentes**. Brasília: SBC, JAI, 16., 1997.

[18] BROOKS, R. A Robust Layered Control System for a Mobile Robot. **IEEE Journal of Robotics and Automation**, v. 2, n. 1, p. 14-23, march 1986.

[19] NUNES, D. J. Estratégia Data-Driven no Desenvolvimento de Software. In: Simpósio Brasileiro de Engenharia de Software, 6., 1992, Gramado. **Anais...** SBC, 1992. v. 1.

[20] REIS, C.; REIS, R.; NUNES, D. Gerenciamento do Processo de Desenvolvimento Cooperativo de Software no Ambiente PROSOFT. Simpósio Brasileiro de Engenharia de Software, 12, 1998, Maringá. **Anais...** SBC, 1998, p. 221-236.

[21] BOLTE, J.;FISHER, J.;ERNST, D. An object-oriented, message-based environment for integration continuous, event-driven and knowledge-based simulation. Application of Advanced Information Technologies: Effective Management of Natural Resources. **Proceedings...** ASAE. June 18-19. 1993.

[22] MI, P.;SCACCHI, W.**Process integration in CASE Environments.** IEEE Software,9(2), March 1992.

[23] RUS, I.; COLLOFELLO, J.; LAKEY, P. Software Process Simulation for Reliability Strategy Assessment. Intl. Workshop on Software Process Simulation Modeling, **Proceedings...** 1998 (http://www.eas.asu.edu/~sdm/publications.html)

[24] LUDEWIG, J. **SESAM: A Software Project Simulator.** (http://www.informatik.uni-stuttgart.de/ifi/se/research/sesam/)