Inteligência Artificial Aplicada a Ambientes de Engenharia de Software: Uma Visão Geral

RENATO AFONSO COTA SILVA[1]

Departamento de Informática – Universidade Federal de Viçosa
CEP 36570-000 Viçosa, MG
[1]renatoacs@dpi.ufv.br

Resumo. A Inteligência Artificial sistematiza e automatiza tarefas intelectuais e, portanto, é potencialmente relevante para qualquer esfera da atividade intelectual humana. Softwares são produtos intangíveis e utilizam no seu processo de construção recursos intelectuais humanos, que vão desde sua especificação até sua distribuição e pleno funcionamento. Como meio de auxiliar o processo de Engenharia de Software, foram criados os ambientes de Engenharia de Software centrados no processo, que possuem um conjunto de ferramentas integradas. Baseado neste contexto, este artigo vem mostrar alguns ambientes existentes que utilizam técnicas de Inteligência Artificial e propor o uso de outras técnicas para melhorar os Ambientes de Engenharia de Software, trazendo uma maior facilidade de construção de softwares e uma maior qualidade para os mesmos.

Palavras-Chave: Inteligência Artificial, Ambientes de Engenharia de Software, Processo de desenvolvimento de Software

Artificial Inteligence in Software Engineering Environments: A Roadmap

Abstract. The Artificial Intelligence organizes and automates intellectual task and, therefore, is potentially relevant for any sphere of the human intellectual activity. Softwares are immaterial products and in their construction process use human intelectual resources, that go since its specification to its distribution and full operation. As a mean of helping out the software engineering process, Process-centered Software Engineering Environment were created, which has a set of integrated tools. Presented that, this paper is going to show some existing environments that use techniques of Artificial Intelligence and proposes the use of other techniques in order to improve the Software Engineering Environments, developing a easier technique of software construction and improving its quality.

Keywords: Artificial Inteligence, Software Engineering Environments, Software Process

(Received August 03, 2005 / Accepted November 18, 2005)

1 Introdução

A Engenharia de Software refere-se à aplicação disciplinada de princípios e métodos no projeto e construção de software de qualidade de forma economicamente viável [2]. Alguns métodos procuram apoiar o gerente de desenvolvimento de software na tarefa de observar e controlar o desenvolvimento de software em termos de seus recursos, prazos, alocação de tarefas e orçamento. Entretanto, a natureza multidisciplinar da área demanda cooperação entre profissionais e ferramentas que apoiem o desenvolvimento de software.

Informalmente, o processo de desenvolvimento de software pode ser compreendido como o conjunto de

todas as atividades necessárias para transformar os requisitos do usuário em software [13, 21]. O processo de desenvolvimento de software é formado por um conjunto de passos parcialmente ordenados, relacionados com conjuntos de artefatos, pessoas, recursos, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os software requisitados [17, 7]. A partir de um esboço dos requisitos iniciais para o problema a ser resolvido, através de software, um modelo de processo de desenvolvimento de software será adotado, o qual resultará no software para atender os requisitos dos usuários.

Com a tecnologia de processo de desenvolvimento de software, surgiu nos últimos anos um novo conceito de ambiente de trabalho que dá suporte de forma integrada aos processos de gestão e produção: Ambientes de Engenharia de Software Centrados no Processo, os chamados PSEEs (do inglês, *Process-Centered Software Engineering Environments*) [16], que constituem um tipo especial de ambiente de desenvolvimento de software que apoia a definição rigorosa de processos de software, objetivando a automação da gerência do desenvolvimento. Tais ambientes geralmente provêem serviços para análise, simulação, execução e reutilização das definições de processos, que cooperam no aperfeiçoamento contínuo de processos.

Dentro deste ambiente, existem os agentes, que estão relacionados com as atividades de um processo, podendo ser pessoas ou ferramentas automatizadas. Agentes diferentes terão percepções diferentes acerca do que acontece durante o processo de desenvolvimento de software. Um gerente, por exemplo, perceberá os aspectos de controle e alocação de recursos e cronogramas para atividades, enquanto um desenvolvedor perceberá as suas atividades como atribuições que devem ser feitas para produzir um resultado [7].

Este artigo apresenta, na Seção 2, conceitos sobre ambientes de Engenharia de Software. Na Seção 3 são apresentadas algumas técnicas de Inteligência Artificial que podem ser aplicadas em conjunto com os PSEEs e em seguida, na Seção 4, são mostrados alguns PSEEs que utilizam alguma técnica de IA. Logo após, na Seção 5, é apresentada uma avaliação desses ambientes do ponto de vista do uso de técnicas de IA e algumas propostas são apresentadas, tendo em vista a restrição observada do uso dessas técnicas nos ambientes atuais.

2 Ambientes de Engenharia de Software

Embora o uso de ferramentas para auxiliar os desenvolvedores na produção de softwares venha sendo aplicado há algum tempo, o conceito de ambiente de Engenharia de Software - SEE (do inglês, *Software Engineering Environment*) é bastante recente. Um SEE é definido como uma coleção de ferramentas que fornece apoio automático, parcial ou total, às atividade de Engenharia de Software. Normalmente essas atividades são executadas dentro de uma estrutura de projeto de software, e se referem a aspectos tais como especificação, desenvolvimento, reengenharia ou manutenção de software [25].

O termo SEE pode ser aplicado a vários sistemas de alcances bem diferentes: desde um conjunto de poucas ferramentas executando sobre um mesmo sistema, até um ambiente totalmente integrado capaz de gerenciar e controlar todos os dados, processos e atividades do ciclo de vida de um software. Graças a automatização total ou parcial das atividades, um SEE pode contribuir com importantes benefícios para uma organização: redução de custos, aumento da produtividade, melhora da gestão e maior qualidade do software final. Por exemplo, automatização de atividade repetitivas como execução de casos de teste que não apenas melhoram a produtividade, mas também ajuda a garantir o término e a consistência das atividades testadas [25]. Normalmente, um SEE gerencia informações relacionadas com:

- Desenvolvimento ou manutenção do software (especificações, dados de projeto, códigos fonte, dados de teste, planos de projeto, ...);
- Recursos do projeto (custos, recursos de informática, pessoal, responsabilidades e obrigações, ...);
- Aspectos organizacionais (política da organização, padrões e metodologias empregadas, ...).

Um SEE oferece suporte às atividades humanas mediante uma série de serviços que descrevem as capacidades do ambiente. Os serviços proporcionam uma correspondência entre um conjunto de processos selecionados, relativos ao ciclo de vida do software, e sua automatização mediante o uso de ferramentas. Na maioria dos casos as funcionalidades de uma ferramenta estão relacionadas com um ou mais serviços.

2.1 Ambientes de Engenharia de Software Centrados no Processo

Os ambientes de Engenharia de Software centrados no processo, os chamados PSEEs (do inglês, *Process-centred Software Engineering Environment*) [16] constituem um tipo especial de ambientes de Engenharia de Software que surgiu nos últimos anos para apoiar a definição rigorosa de processos de software, objetivando automatizar a gerência do desenvolvimento. Tais ambientes geralmente provêem serviços para análise, simulação,

execução e reutilização das definições de processos, que cooperam no aperfeiçoamento contínuo de processos.

A modelagem de processos de software não consiste apenas em escrever programas que automatizem completamente o processo de desenvolvimento de software e nem descrevem tudo o que os atores do processo devem fazer [23]. Enquanto os programas de computador são escritos para definir o comportamento de uma máquina determinística, os programas de processo são escritos para definir possíveis padrões de comportamento entre elementos não-determinísticos (atores) e ferramentas automatizadas [26]. Como conseqüência, um PSEE deve ainda permitir que os atores envolvidos no processo recebam orientação automatizada e assistência na realização de suas atividades, sem interferência no processo criativo [20]. Além disso, processos ainda podem ser modificados dinamicamente, em resposta a estímulos organizacionais ou mudança nos requisitos do software em desenvolvimento.

A literatura especializada define três tipos principais de modelos [9, 6, 8]:

- **Modelos Abstratos** (*patterns* ou *templates*), que fornecem moldes de solução para um problema comum, em um nível de detalhe que idealmente não está associado a uma organização específica. Um processo abstrato é um modelo de alto nível que é projetado para regular a funcionalidade e interações entre os papéis de desenvolvedores, gerentes, usuários e ferramentas em um PSEE [27];

- **Modelos Instanciados** (ou executáveis) são modelos prontos para execução, podendo ser submetidos à execução por uma máquina de processo. O modelo instanciado é considerado uma instância de um modelo abstrato, com objetivos e restrições específicos, envolvendo agentes, prazos, orçamentos, recursos e um processo de desenvolvimento;

- **Modelos em Execução** ou **Executados** registram o passado histórico da execução de um processo, incluindo os eventos e modificações realizadas no modelo associado.

A arquitetura de um PSEE usualmente define como componente central a máquina de processo [6] que auxilia na coordenação das atividades realizadas por pessoas e por ferramentas automatizadas, sendo responsável pela interpretação/execução dos modelos de processos descritos com PMLs (*Process Modelling Language*). Uma máquina de processos é responsável por: ativar automaticamente atividades sem intervenção humana através de uma integração com as ferramentas do ambiente; apoiar o envolvimento cooperativo dos desenvolvedores; monitorar o andamento do processo e registrar o histórico da sua execução [16]. A máquina de processo também deve garantir a execução das atividades na seqüência definida no modelo de processo; 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 [16].

O mecanismo de execução de processos de um SEE pode conter diversas instâncias simultâneas de máquinas de execução. Isto é necessário porque o SEE pode estar sendo utilizado para desenvolvimento em diversos projetos em uma organização. Portanto, de forma geral um mecanismo de execução consiste de uma ou várias máquinas de execução. As máquinas de execução possuem componentes que trabalham na execução de processos e na integração com o restante do ambiente pois a execução envolve desde a interface com o usuário até a gerência dos objetos no banco de dados. Dentro de um PSEE, o mecanismo de execução pode ser tratado como mais uma ferramenta ou pode ser um componente básico do ambiente.

3 Técnicas de Inteligência Artificial

Inteligência Artificial é uma das ciências mais recentes, que atualmente abrange uma variedade enorme de subcampos, que vão desde áreas de uso geral, como aprendizado e percepção, até tarefas mais específicas, como jogos de xadrez [24]. A IA sistematiza e automatiza tarefas intelectuais e, portanto, é potencialmente relevante para qualquer esfera da atividade intelectual humana, que neste caso, será abordada técnicas de IA para auxiliar o processo de desenvolvimento de softwares através dos PSEEs.

3.1 Sistemas Especialistas

Sistemas especialistas (SE's) são uma classe de software que atuam como colaboradores na tomada de decisão em áreas da ciência dominadas por especialistas humanos. Um sistema especialista condensa o conhecimento de um ou mais especialistas e utiliza este conhecimento armazenado para auxiliar na resolução de problemas do usuário.

Os SE's são Sistemas Baseados em Conhecimento (SBC's) que atuam em áreas e em tarefas bem definidas. Estruturalmente, todo SE é constituído de duas partes principais: a Base de Conhecimento (BC), que contém o conhecimento heurístico e fatual sobre o do-

mínio de aplicação do SE, e a Máquina de Inferência (MI), que usa o conhecimento da BC para construir a linha de raciocínio que leva à solução do problema.

O conhecimento obtido dos especialistas pode ser representado através de formalismos distintos: lógica, regras de produção, redes semânticas, frames e raciocínio baseado em casos. Embora tenham abordagens diferentes ao problema da representação do conhecimento, nada impede que sejam utilizadas combinações entre duas abordagens, como por exemplo, regras e frames para a representação do conhecimento de um determinado sistema.

3.2 Sistemas Baseados em Regras

O paradigma de regras é um dos paradigmas mais populares para representação do conhecimento em sistemas de IA. Conforme [12] isso se deve, principalmente, à natureza modular das regras, as suas facilidades de explanação e por modelarem o conhecimento de uma forma muito próxima ao processo cognitivo humano.

Uma regra consiste de uma parte SE, o lado esquerdo da regra, e de uma parte ENTÃO, o seu lado direito. A parte SE lista um conjunto de condições combinadas de forma lógica e a parte ENTÃO representa a ação a ser executada ou a conclusão a ser deduzida, caso todas as condições da parte SE tenham sido satisfeitas.

Sintaticamente, as regras podem ser representadas como :

SE $\text{COND}_1, \text{COND}_2, \ldots \text{COND}_n$ ENTÃO
$\text{CONCL}_1, \ldots, \text{CONCL}_n$
OU
SE $\text{COND}_1, \text{COND}_2, \ldots \text{COND}_n$ ENTÃO
$\text{AÇÃO}_1, \ldots, \text{AÇÃO}_n$

A coleção de predicados é chamada de Memória de Trabalho (MT). A parte ENTÃO da regra especifica novos predicados a serem colocados na MT. O sistema baseado em regras é dito ser um sistema dedutivo quando a parte ENTÃO somente contém predicados. Algumas vezes o ENTÃO especifica ações. Neste caso o sistema é dito reativo.

3.3 Sistemas *Fuzzy*

O interesse nos sistemas *fuzzy* tem aumentado nos últimos anos. A palavra *fuzzy* tornou-se adjetivo usado para descrever tudo o que não é absolutamente preciso, porém, nem sempre o termo é usado adequadamente. Devido ao uso indiscriminado deste termo, [4] define sistemas *fuzzy* como:

"Qualquer sistema computacional onde os valores não são precisamente predeterminados ou onde a confiança no sistema ou em seus dados possa ser discutida ou onde medidas de certeza estatísticas, probabilísticas ou ad-hoc sejam usadas para desenvolver parâmetros."

Os sistemas *fuzzy* combinam a flexibilidade e representação de conhecimento de alto nível dos sistemas especialistas convencionais com a habilidade de tratar problemas complexos e não lineares com o mínimo de regras. Segundo [4], os sistemas baseados em lógica *fuzzy* tem como principal benefício a redução dos custos de desenvolvimento, execução e manutenção. Entretanto não são adequados para todos os tipos de sistemas em que os sistemas convencionais se aplicam [4].

Diferente da lógica booleana, que possui os estados verdadeiro ou falso, a lógica *fuzzy* trata de valores verdade que variam continuamente de 0 a 1. Dessa forma um fato pode ser meio verdade 0,5, quase verdade 0,9 ou quase falso 0.1. O uso da lógica *fuzzy* em sistemas de raciocínio traz impacto não somente na máquina de inferência, mas também na representação do conhecimento. A lógica *fuzzy* permite expressar conhecimento em um formato de regra que é bastante parecido com linguagem natural.

3.4 Raciocínio Baseado em Casos

Case-Based Reasoning (CBR) é uma técnica que utiliza a experiência passada para resolver problemas. A idéia de CBR é descrever e acumular casos significativos para a área de conhecimento especializado e tentar descobrir, por analogia, quando determinado problema é "similar"a um outro resolvido, aplicando a solução armazenada ao novo problema semelhante que surgiu.

Um caso é um pedaço de conhecimento contextualizado representando uma experiência. Ele contém a solução do caso e o contexto onde essa solução pode ser usada. Um caso pode ser a descrição de um evento, uma história ou algum registro contendo tipicamente o problema no momento em que o caso ocorreu mais a solução para esse problema [28].

Uma maneira de visualizar um sistema de CBR é em termos de espaços de problemas e espaços de solução. A descrição de um novo problema a ser resolvido é posicionada no espaço do problema. O caso com a descrição mais similar é recuperado e sua solução é encontrada. Se necessário, pode ser feita uma adaptação e uma nova solução pode ser criada.

Comparando os sistemas CBR com sistemas baseados em regras pode-se observar que para a criação de regras é sempre necessário saber como resolver o problema, sendo esta tarefa complexa e consumidora de muito tempo. Em sistemas CBR não é necessário saber como resolver os problemas, e sim apenas reconhecer se um problema similar foi resolvido no passado. Por isso, para problemas bem compreendidos que não se

modificam com o tempo, um sistema de regras é mais adequado. Porém, quando o problema é pouco conhecido e dinâmico, o CBR deve ser utilizado. Apesar das diferenças, alguns sistemas híbridos (CBR + regras) têm sido desenvolvidos, tais como em [5], onde é apresentado um sistema para a área jurídica que recupera casos para auxiliar a denúncia de homicídios.

Os sistemas CBR e os sistemas de redes neurais possuem como similaridade apenas o fato de se basearem em casos passados. As redes neurais são boas em domínios onde os dados não podem ser representados simbolicamente, tais como reconhecimento de voz e interpretação de sinais. CBR trata de forma adequada dados estruturados simbolicamente e complexos mas não é tão bom para tratar dados puramente numéricos.

3.5 Agentes

Um agente é tudo o que se pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por meio de atuadores [24].

Dado um determinado sistema, denomina-se agente cada uma de suas entidades ativas. O conjunto de agentes forma uma sociedade. As entidades passivas serão 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. O agente pode ser uma entidade real ou virtual que é capaz de agir em seu ambiente, podendo se comunicar com outros agentes, comportando-se de forma autônoma.

Os sistemas multiagentes dividem-se em duas classes principais: Agentes Reativos e Agentes Cognitivos [24]. A abordagem reativa baseia-se na idéia de que agentes com ações elementares podem realizar trabalhos complexos.

Segundo [1], nos sistemas multiagentes reativos não há representação explícita do conhecimento; não há representação 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; uma sociedade contém poucos agentes.

Apesar da classificação, os sistemas multiagentes podem não ser totalmente cognitivos ou reativos. Um sistema pode ser uma mistura dos dois para atender a solução de um determinado problema.

Um tipo de aplicação de agentes que pode também ser usado na Engenharia de Software são os agentes que reduzem o trabalho e o excesso de informação [18].

Uma solução para o excesso de informação está no uso de Assistentes Pessoais, que são agentes autônomos capazes de atuar cooperativamente com o usuário para reduzir trabalho e excesso de informação. Um tipo de assistente pessoal é o agente de interface, o qual é capaz de realizar tarefas pelo usuário, treiná-lo, auxiliar na colaboração entre usuários e monitorar eventos. Os agentes de interface podem ser usados para filtragem de informações, gerência de correio eletrônico, agenda de compromissos, seleção de entretenimento, dentre outros.

Em [18] é apresentada uma abordagem para construção de agentes de interface. As principais questões a serem tratadas são quanto a competência do agente, ou seja, como garantir que ele vai adquirir o conhecimento necessário; e a confiança do usuário, ou seja, como garantir que o usuário vai delegar tarefas para o agente. O agente adquire competência através da observação das ações do usuário, através do *feedback* do usuário quando o agente realiza uma ação, através de exemplos do usuário (treinamento do agente), pedindo conselho a outros agentes e aprendendo quais são os agentes mais confiáveis.

3.6 Aprendizado

O elemento central do comportamento inteligente é a habilidade de se adaptar ou aprender a partir de experiências. Existem várias técnicas de aprendizado [24], tais como:

- por implantação direta do conhecimento: através da inclusão de novas regras diretamente na base de conhecimento;
- por indução: perante um conjunto de exemplos ou dados particulares, o sistema procura inferir conceitos e leis gerais. Este aprendizado pode ser através de exemplos ou por observação e descoberta;
- por analogia: o sistema chega a conclusões sobre uma nova situação a partir de um modelo que foi construído como resultado de experiências anteriores;
- por casos: dados alguns casos de referência, para descobrir uma propriedade de uma situação ou dado particular, são encontrados os casos mais similares de acordo com as propriedades conhecidas;

- por construção de árvores de identificação;

- por redes neurais: Existem vários tipos de redes neurais. Elas podem ser usadas em aprendizado supervisionado (*backpropagation*) ou não supervisionado (mapa de Kohonen). Em uma rede de neurônios artificiais, existem somadores, multiplicadores e limites. Cada neurônio é disparado quando a influência coletiva de suas entradas atinge um limite mínimo. As entradas possuem pesos que são ajustados no processo de aprendizagem.

4 PSEEs que utilizam técnicas de IA

A seguir são descritos brevemente alguns ambientes de Engenharia de Software que utilizam técnicas de Inteligência Artificial.

Marvel - Marvel [11] é um ambiente centrado em processo resultante de um projeto de mesmo nome na *Columbia University* desde 1986. A idéia do projeto era desenvolver um ambiente centrado em processo que orientasse e assistisse os usuários que trabalham em projetos de grande escala. Para isso, juntaram as áreas de Engenharia de Software e Inteligência Artificial.

Marvel segue o paradigma de orientação a objetos e está baseado em uma linguagem de regras. Ele fornece uma ajuda automatizada pela aplicação das estratégias *forward* e *backward chaining* sobre as regras, invocando as atividades que são parte de uma fase do processo de desenvolvimento. A sua base de objetos é persistente. O processo sendo modelado é decomposto em passos de processo. Cada passo é encapsulado em uma ou mais regras.

Merlin - Merlin [14, 10] é um ambiente de desenvolvimento de software centrado em processo construído dentro de um projeto na *University of Dortmund*. O protótipo do ambiente usa o paradigma de regras para descrever e executar processos de software.

Uma definição de processo no Merlin possui atividades, papéis (*roles*), documentos (quaisquer objetos) e recursos (pessoas, ferramentas, etc.). Um documento é ligado a um conjunto de atividades e a um conjunto de ferramentas que suportam as atividades. Nesta abordagem os usuários recebem todas as informações relevantes em um espaço de trabalho chamado *working context* associado ao papel do usuário. Este espaço de trabalho contém os documentos a serem manipulados, suas dependências com outros documentos e as atividades que devem ser realizadas. Quando uma atividade realizada influencia um *working context*, acontece uma atualização dinâmica nos *working contexts* dependentes deste evento.

A máquina de execução do Merlin constrói e atualiza os *working contexts* dos usuários. Cada atividade possui pré-condições que definem, por exemplo, os papéis que podem desempenhar a atividade, a pessoa responsável, os direitos de acesso aos documentos ou a dependência com outras atividades. Quando as pré-condições de uma atividade são verdadeiras, a atividade é inserida no *working context*. O processo de encadeamento é parecido com o do ambiente Marvel apresentado anteriormente. O Merlin suporta mudanças dinâmicas no processo, permitindo que um processo não seja totalmente definido antes de iniciar. A linguagem escolhida para o ambiente é baseada em PROLOG.

Articulator - [19] é um ambiente baseado em conhecimento para processo de desenvolvimento de software. Ele provê um meta-modelo de processo de desenvolvimento de software, uma linguagem baseada em objetos e um mecanismo de simulação automática. Para simular a execução de processos, o Articulator usa uma abordagem multiagentes onde os desenvolvedores são modelados como agentes cognitivos. A arquitetura do ambiente contém cinco subsistemas: base de conhecimento, simulador de comportamento, mecanismo de consulta, gerenciador de instanciação e gerenciador de aquisição de conhecimento.

O meta-modelo do Articulator consiste de recursos, agentes e tarefas. Os recursos são objetos nas tarefas dos agentes. As tarefas consomem e produzem recursos. Um agente representa uma coleção de comportamentos e atributos associados. O comportamento do agente emerge durante a execução das tarefas (incluindo comunicação, acomodação e negociação) levando em consideração a situação do agente. Os agentes são modelos gerais de desenvolvedores, times de desenvolvimento e organizações. Ferramentas de desenvolvimento são modeladas somo subclasse de agentes.

As tarefas são representadas através de uma rede de ações que os agentes realizam. Como não existe execução real de processos neste ambiente, os agentes são agentes cognitivos que simulam execução de tarefas e várias situações para auxiliar a geração da descrição do processo. A simulação é importante para detectar falhas como alocação de recursos, cronograma, dentre outras.

Pandora - Pandora [15] é uma máquina de processos baseada em programação em lógica e conceitos de lógica temporal. Todos os eventos são registrados e o sistema possui um algoritmo de aplicação de regras que otimiza os passos de execução. Além disso, o sistema possui um mecanismo de sincronização que garante que as atividades cooperativas ou que devem ser executadas em alguma ordem serão sincronizadas.

Pandora integra os paradigmas de eventos para mo-

delar interatividade entre mensagens e de regras para representar o conhecimento do processo. O conhecimento pode ser declarativo ou procedural. Conhecimento declarativo descreve o domínio do discurso, no caso, o modelo de processo enquanto conhecimento procedural descreve o comportamento que o processo pode assumir durante o desenvolvimento, ou seja, as regras e eventos que disparam evolução de processo.

O sistema reage aos eventos externos e dispara as ações internas. As regras e eventos são especificados em uma linguagem de lógica de primeira ordem acrescida de operadores de lógica temporal (Linguagem Pandora) onde o tempo é caracterizado por uma linha seqüencial simples de eventos. Isto permite expressar quantitativamente o tamanho dos intervalos temporais, a distância temporal entre os eventos e a viabilidade das atividades modeladas quanto às proposições lógicas estabelecidas. É possível estabelecer não apenas quais atividades são permitidas a cada momento, mas também como as atividades interagem (ou seja, sincronização entre atividades paralelas e disparo de atividades engatilhadas).

Pandora é implementado em PROLOG e consiste de dois módulos principais: compilador de regras e interpretador de regras. O compilador checa as regras quanto a sintaxe e as transforma em uma representação interna. O interpretador permite ativação das ferramentas do sistema operacional, permite rastreamento da aplicação de regras e observação dos estados internos.

5 Avaliação e Propostas

Esta seção apresenta a avaliação do uso das técnicas de IA em PSEEs levando em consideração aspectos de modelagem, reutilização de processo, execução, simulação de processo, diagnóstico e recuperação de falhas na execução, arquitetura do SEE e interface com os usuários do ambiente. Em seguida são apresentadas algumas propostas para utilização de técnicas de IA em PSEE visando facilitar o uso e aumentar a produtividade tanto do desenvolvedor quanto do gerente.

5.1 Avaliação

Os ambientes pesquisados permitem modelagem e execução de processos de software, sendo que somente o Articulator permite simulação. A adoção de notações gráficas para modelagem de processos foi observada no ambiente Merlin.

O paradigma de regras como forma de representação do processo ou como modelo de execução foi totalmente adotado pela tecnologia de processos face à grande quantidade de ambientes existentes que utilizam o paradigma. As causas para essa adoção podem estar no requisito de modificação dinâmica durante execução do processo, sendo o paradigma de regras apontado como flexível a mudanças, ou ainda a necessidade de raciocínio baseado em conhecimento. O conhecimento sobre processos pode ser armazenado em uma base de conhecimento e regras podem ser utilizadas para obter novas informações ou auxiliar no acesso a esse conhecimento.

Os SEEs encontrados não se comportam como sistemas especialistas. O conhecimento sobre modelos de processo fica embutido no ambiente e os modelos construídos são desenvolvidos por projetistas de processo, que o fazem como um roteiro do desenvolvimento de software. Este roteiro pode ser modificado a qualquer momento. Portanto o conhecimento adquirido não provém de um especialista em processo de desenvolvimento de software visando resolver ou diagnosticar uma situação e sim de experiências em modelagem de processos.

O PSEE estudado que mais utiliza técnicas de IA é o Articulator. Além de ser baseado em regras para modelagem e execução do processo, ele utiliza o paradigma de multiagentes para modelar o comportamento dos agentes do processo. Como o ambiente permite somente simulação do processo, os agentes são modelados para agirem da mesma forma que agiriam agentes humanos. Neste caso, o conhecimento a ser modelado é complexo e baseia-se em perícia, experiência, estilo de trabalho, dentre outras características dos agentes. A este ambiente foi integrada uma abordagem para diagnóstico, replanejamento e re-escalonamento chamada Articulation. Esta abordagem é muito útil, pois o conhecimento sobre as falhas da execução dos processos vai sendo armazenado e gera heurísticas para a solução de novas falhas. Outra ferramenta incorporada ao Articulator é a biblioteca de processos SPLib. Ela permite reutilização de processos e possui uma base de conhecimento de processos que organiza os modelos e suas dependências. A forma de acesso a um processo desta biblioteca é similar à técnica case-based reasoning uma vez que o usuário deve fornecer algumas características desejáveis do processo e a biblioteca faz o *matching* dos processos mais adequados.

Pode-se observar que nem todas as técnicas de Inteligência Artificial são utilizadas pelos PSEEs, como redes neurais por exemplo. Isto se justifica porque o modelo de processo e sua execução necessitam de dados simbólicos e bem estruturados e o uso de redes neurais é mais adequado quando os dados não podem ser representados simbolicamente, como reconhecimento de voz por exemplo. Porém existe um caso de utilização de re-

des neurais em [3] para geração automática de modelos formais do comportamento de processos de software a partir da execução utilizando-se uma técnica chamada *process discovery*, que foi testada usando redes neurais, cadeias de Markov e um método algorítmico, mas o teste com redes neurais não foi satisfatório.

A seguir é apresentada uma tabela comparativa dos ambientes estudados.

Técnicas de IA / Ambientes	Marvel	Merlin	Articulator	Pandora
regras	**Sim**	**Sim**	**Sim**	**Sim**
case-based reasoning	Não	Não	**Sim**	Não
lógica *fuzzy*	Não	Não	Não	Não
sistemas multiagentes	Não	Não	**Sim**	Não
redes neurais	Não	Não	Não	Não
técnicas de aprendizado	Não	Não	**Sim**	Não

Figura 1: Comparação do uso de técnicas de IA pelos ambientes centrados em processo

5.2 Propostas

As técnicas de IA ainda são muito pouco exploradas na Engenharia de Software. As principais utilizações se restringem ao paradigma de regras para modelar sistemas baseados em conhecimento que auxiliam o desenvolvimento de software. Este artigo procurou características de utilização de Inteligência Artificial em PSEEs. Foram apresentadas algumas técnicas de IA e alguns ambientes, com o objetivo de avaliar a utilização das técnicas nos ambientes. Entretanto o uso ainda é muito restrito, o que nos leva a propor algumas utilizações.

Nas seções a seguir são apresentadas algumas idéias a respeito das técnicas de IA que podem ser utilizadas em modelagem, reutilização, execução, simulação e interface com o usuário de um PSEE [22].

5.2.1 Modelagem e Reutilização

Uma técnica de IA que aumentaria o poder de expressão de um modelo de processo seria a lógica *fuzzy*. Através de regras *fuzzy*, situações do tipo “se nível de qualidade da tarefa anterior é alto, então faça operação 1". Neste caso, o nível de qualidade depende de uma avaliação que não é precisa, mas que é importante para o desenvolvimento do processo.

A reutilização de processos de software é altamente necessário e tem sido proposto em vários trabalhos. Para a recuperação de processos de software adequados à situação que se quer modelar, podem ser usados sistemas de *case-based reasoning* como na biblioteca SPLib do sistema Articulator.

5.2.2 Execução

Na fase de execução do meta-processo de desenvolvimento de software, o componente principal é a máquina de processo. Este componente centraliza o controle do desenvolvimento de software e é responsável por inúmeras tarefas, como verificar consistência, direitos de acesso, próximas atividades a serem executadas, agendas dos usuários, estados das atividades, dentre outras. Para distribuir essas tarefas poderiam ser utilizados agentes da IA. Um agente poderia cuidar das agendas dos usuários, outro das atividades e seus estados e assim por diante. Uma vantagem dessa abordagem é a capacidade de aprendizado dos agentes, os quais poderiam agir de forma mais apropriada às peculiaridades do ambiente e seus usuários. Neste caso utilizaria sistemas multiagentes cognitivos. Os agentes reativos também poderiam ser usados para coletar métricas, por exemplo. Eles observariam o que acontece no ambiente e aumentariam a base de métricas com informações importantes sobre a execução do processo.

Há também a possibilidade de se utilizar lógica *fuzzy* na execução de processos. A máquina de execução, ao observar o atraso em uma atividade, pode detectar que o desenvolvedor que causou o atraso é “preguiçoso". Assim, pode existir uma regra *fuzzy* que verifica se a atividade está atrasada e o desenvolvedor é preguiçoso, então delegar automaticamente a tarefa para um agente mais responsável. A variável preguiçoso seria um conjunto *fuzzy* derivado de várias características [22].

Case-based reasoning também seria aplicável na fase de execução pois em casos de parada na execução, o sistema poderia buscar um caso parecido e resolver o problema com uma solução testada.

5.2.3 Simulação

O SEE deve suportar validação e verificação de modelos de processo. Através da simulação de processos de software o ambiente e os usuários podem detectar falhas, inconsistências e comportamentos anômalos na descrição do processo. Conflitos no cronograma e alocação de recursos são comuns. Além disso, o usuário pode prever o progresso do desenvolvimento e comparar modelos [22].

Na simulação de processos não existe o desenvolvedor humano. Portanto é preciso simular o comportamento dele. A abordagem de agentes tem sido utilizada

para essa finalidade como apresentado pelo ambiente Articulator [19].

A simulação de processos pode usar também lógica *fuzzy* da mesma forma que foi proposta na execução (seção 5.2.2).

Para que a simulação atinja seus objetivos de apontar os problemas da modelagem e da execução é necessário que ela seja baseada em situações reais. Portanto um sistema de *case-based reasoning* poderia ser usado para prover conhecimento real sobre situações de execução de processo e como são resolvidas essas situações. Na simulação também poderia se utilizar características reais dos agentes humanos para modelar os agentes cognitivos. Dessa forma, a simulação poderia auxiliar a resolver conflitos entre os desenvolvedores e a selecionar atividades para os mesmos.

5.2.4 Interface

A interface com o usuário de um PSEE depende do papel que ele executa no desenvolvimento. Um gerente precisa ver resultados quantitativos e qualitativos do trabalho dos desenvolvedores. As suas ações visam ajustar o processo para que não atrase e atinja os objetivos, sendo que para isso ele necessita interagir com os desenvolvedores. O desenvolvedor precisa ver as tarefas a serem desenvolvidas e os objetos a serem manipulados. Precisa também priorizar e delegar tarefas quando possível.

Os agentes de interface poderiam auxiliar esses usuários. Da mesma forma que um agente pode adquirir competência no tratamento das mensagens de seus usuários, ele pode adquirir competência no tratamento das tarefas que o desenvolvedor deve executar e tornar-se seu assistente pessoal de processo. O agente pode auxiliar um usuário novato indicando as tarefas mais importantes e fornecendo todas as informações necessárias para o desenvolvimento das mesmas. Por outro lado, com um usuário mais experiente, o agente pode deduzir que o mesmo não necessita de tantas informações para realizar seu trabalho e apresentar somente as informações necessárias. Adquirindo competência, o agente pode automaticamente delegar tarefas para outros desenvolvedores, marcar reuniões e trocar informações com outros agentes para solucionar os problemas que surgirem [22].

Acredita-se que esta abordagem pode aumentar a utilização de PSEEs, pois um dos principais problemas da execução automatizada é a obrigação do seguimento de alguns passos do processo. Um desenvolvedor experiente não fica satisfeito com um sistema que o obriga a seguir passos programados e informa a todo instante como fazer seu trabalho. Um agente de interface pode identificar isso e filtrar as informações de acordo com o nível do usuário facilitando assim o aumento da produtividade no desenvolvimento de software.

6 Conclusão

As técnicas de Inteligência Artificial tem sido cada vez mais utilizadas em várias áreas da ciência da computação. Seu uso se justifica pelo auxílio fornecido ao raciocínio humano em tarefas que se beneficiam da experiência e do conhecimento de especialistas.

Neste artigo foi apresentada a tecnologia de processos de software e suas características com o objetivo de encontrar utilização ou necessidade de utilização de técnicas de IA. Os PSEEs Marvel, Merlin, Articulator e Pandora foram estudados e avaliados. Com isso, podemos dizer que o ambiente que mais utiliza técnicas de IA dentre os apresentados é o Articulator [19], pois, permite simulação de processos e usa um modelo multiagentes para representar o comportamento dos desenvolvedores.

Também foram apresentadas algumas propostas de utilização de técnicas de IA em modelagem, execução, simulação, reutilização e interface do ambiente. As propostas foram derivadas do estudo das técnicas de Inteligência Artificial e do estudo da arquitetura e funcionamento dos PSEEs, sendo que essas propostas podem vir a ser implementadas em um futuro próximo, de forma a contribuir para o aumento da qualidade do software.

A utilização de técnicas de IA na Engenharia de Software ainda necessita de estudos e experiências, porém estão surgindo ambientes que se preocupam com os fatores psicológicos do desenvolvimento de software e sua facilidade de uso. As pesquisas na área apontam para o objetivo de tornar o PSEE um gerente de auxílio que não influencie a capacidade e a vontade dos desenvolvedores, mas ao mesmo tempo controle os recursos e os direitos de acesso, colete métricas e disponibilize análises sobre o desempenho do processo de desenvolvimento de software. Acredita-se que as técnicas de IA ainda tem muito a contribuir para o desenvolvimento dessa área.

Referências

[1] ALVARES, L. and SICHMAN, J. Introducao aos sistemas multiagentes. *JORNADA DE ATUALIZACAO EM INFORMATICA*, 1997.

[2] BOEHM, B. Software engineering. *IEEE Transactions on Computers*, December 1976.

[3] COOK, J. and WOLF, A. Automating process discovery through event-data analysis. In *INTER-*

NATIONAL CONFERENCE ON SOFTWARE ENGINEERING, Seattle, Washington, United States, 1995. Proceedings of the 17th International Conference on Software Engineering - ACM Press.

[4] COX, E. D. *Fuzzy Logic for Business and Industry*. Charles River Media, 1995.

[5] DAHMER, A. Um modelo híbrido de sistema especialista para a Área jurídica. In *CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN*, San Luis, Argentina, 1996. Universidad Nacional de San Luis.

[6] DERNIAME, J., W., KABA, B. Software process: Principles, mehodology and technology. *Springer*, 1999.

[7] DOWSON, M., R., NEJMEH, B. Fundamental software process concepts. *AICA Press*, May 1991.

[8] FEILER, P. H., H. Software process development and enactment: Concepts and definitions. *IEEE Computer Society Press*, May 1993.

[9] FRANCH, X., R. Supporting process reuse in promenade. February 2002.

[10] G. JUNKERMANN, W. S., B. PEUSCHEL and WOLF, S. Merlin: Supporting cooperation in software development through a knowledge-based environment. In *Software process modelling and technology archive*, pages 103–129, London, UK, 1994. Research Studies Press Ltd.

[11] GAIL E. KAISER, P. H. F. and POPOVICH, S. S. Intelligent assistance for software development and maintenance. *JORNADA DE ATUALIZACAO EM INFORMATICA*, pages 40–49, May 1988.

[12] GIARRATANO, J. and RILEY, G. *Expert Systems: Principles and Programming*. PWS-KENT Publishing Company, 1989.

[13] HUMPHREY, W. S. The software engineering process: Definitions and scope. *IEEE Computer Society Press*, 1989.

[14] JUNKERMANN, G. and SCHÄFER, W. A design methodology for process-programming. In *Proceedings of the Third European Workshop on Software Process Technology*, pages 69–73, London, UK, February 1994. Springer-Verlag.

[15] LAGO, P. and G.MALNATI. Pandora: A temporal logic based process engine. *WORKSHOP IN LOGIC PROGRAMMING APPLIED TO SOFTWARE ENGINEERING*, 1994.

[16] LIMA, C. Um gerenciador de processos de software para ambiente prosoft. Master's thesis, CPGCC-UFRGS, 1998. Dissertação de Mestrado.

[17] LONCHAMP, J. A structured conceptual and terminological framework for the software process engineering. *IEEE Computer Society Press*, May 1993.

[18] MAES, P. and KOZIEROK, R. Agents that reduce work and information overload. *Communications of the ACM*, 37(7), July 1994.

[19] MI, P. and SCACCHI, W. A knowledge-based environment for modeling and simulating software engineering process. In *IEEE Transactions on Knowledge and Data Engineering archive*, pages 283–294. IEEE Educational Activities Department, September 1990.

[20] NGUYEN, M., W. Total software process model in epos. http://www.idt.unit.no/ epos/Papers, 1997.

[21] OSTERWEIL, L. Software process are software too. *International Software Process Workshop*, 1987.

[22] REIS, C. A. L. Estudo da utilização de técnicas de inteligência artificial na tecnologia de processos de software. Trabalho Individual II - PPGC/UFRGS, Março 1999.

[23] REIS, R. Q., N. Uma avaliação dos paradigmas de linguagens de processo de software. Trabalho Individual II - PPGC/UFRGS, Março 1999.

[24] RUSSELL, S. and NORVIG, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 2003.

[25] SATLER, M. F. Utilidad de la tecnología de workflow en los psees. Dep. de Informática, Universidad de Castilla-La Mancha, 2004.

[26] TULLY, C. Representing and enacting the software process. June 1989.

[27] WANG, Y., K. Software engineering processes: Principles and applications. *Boca Raton: CRC Press*, 2000.

[28] WATSON, I. Applying cade-based reasoning: Techniques for enterprise systems. *CA: Morgan Kaufmann Publishers*, 1997.