Ambientes de Desenvolvimento de Software e o Projeto ADS



(1) MIAN, P. G.; **(2)** NATALI, A. C. C.; **(3)** FALBO, R.A.

(1) Aluna do Mestrado em Informática da UFES e bolsista da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior). E-mail: **pgmian@inf.ufes.br**

(2) Aluna do Mestrado em Informática da UFES e bolsista da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior). E-mail: **anatali@inf.ufes.br**

(3) Doutor em Informática pela COPPE/UFRJ. Professor do Departamento de Informática da UFES. E-mail: **falbo@inf.ufes.br**

RESUMO:

Desenvolver software de qualidade assegurada, com elevada produtividade, dentro do prazo estabelecido e sem necessitar de mais recursos do que os alocados têm sido o grande desafio da Engenharia de Software. Neste contexto, é crescente a demanda por Ambientes de Desenvolvimento de Software (ADSs). ADSs são ambientes que se propõem a disponibilizar ferramentas de apoio ao longo de todo o processo de desenvolvimento do software. As ferramentas CASE que compõem o ADS, apesar de cada uma possuir sua funcionalidade específica de acordo com a etapa do processo que deve apoiar, devem ser integradas. Este texto apresenta o Projeto ADS, um projeto que visa a construção de um ADS, e discute a integração das ferramentas que o compõem.

Palavras-chave: Ambientes de Desenvolvimento de Software, Ferramentas CASE, Integração de Ferramentas.

ABSTRACT:

Developing software with ensured quality, high productivity, within the schedule and no need of extra resources, is the biggest challenge of Software Engineering. In this context, the need of Software Engineering Environments (SEEs) is increasing. SEEs are environments that aim to support the whole software development process. A CASE tool in a SEE has a specific functionality according to the activity of the process it intends to support. But, all of them must be integrated. This paper describes the ADS Project, which aims to develop a SEE. It also discusses tool integration in the SEE.

Keywords: Software Engineering Environments, CASE Tools, Tool Integration.

1. INTRODUÇÃO

Desenvolver software de qualidade assegurada, com elevada produtividade, dentro do prazo estabelecido e sem necessitar de mais recursos do que os alocados têm sido o grande desafio da Engenharia de Software.

Cada vez mais engenheiros de software têm sido cobrados para realmente fazerem engenharia do produto de software: planejar, acompanhar, executar e controlar. Cresce, então, a necessidade de ferramentas para apoiar estas tarefas. Após se ter verificado que ferramentas isoladas podem oferecer apenas soluções parciais, o que se deseja é utilizar ferramentas de apoio ao longo de todo o processo de desenvolvimento de software (TRAVASSOS, 1994). Neste contexto, é crescente a demanda por Ambientes de Desenvolvimento de Software (ADSs), que buscam combinar técnicas, métodos e ferramentas para apoiar o Engenheiro de Software na construção de produtos de software, abrangendo todas as atividades inerentes ao processo, tais como planejamento, gerência, desenvolvimento e controle da qualidade (FALBO, 1998).

Este trabalho apresenta o Projeto ADS e descreve algumas das ferramentas já construídas, bem como discute aspectos de integração destas no ambiente. Na seção 2, discute-se a importância e as formas de integração de ferramentas em ADSs. A seção 3 apresenta o Projeto ADS, as ferramentas que o compõem e as perspectivas futuras de trabalho. Finalmente, na seção 4 são apresentadas as conclusões.

2. INTEGRAÇÃO DE FERRAMENTAS EM ADS

A Engenharia de Software está em constante evolução e isso se manifesta nos métodos e nas formas de apoiar o desenvolvimento de software. Neste contexto, surgiu a tecnologia CASE (*Computer Aided Software Engeneering*), que permitiu o apoio automatizado a métodos para desenvolvimento de software e, mais recentemente, o conceito de Ambientes de Desenvolvimento de Software (ADSs). Enquanto uma ferramenta CASE se destina a apoiar apenas uma etapa do desenvolvimento de software, um ADS se propõe a disponibilizar ferramentas de apoio ao longo de todo o processo de desenvolvimento. As ferramentas que compõem o ADS, apesar de cada uma possuir sua funcionalidade específica de acordo com a etapa do processo que pretende apoiar, devem ser integradas.

A integração tem sido considerada o elemento central para prover um suporte efetivo ao desenvolvimento de software, pois é ela que define todas as regras e diretrizes que governarão o uso das ferramentas. A integração é responsável pela combinação de ferramentas de forma que elas trabalhem em harmonia em todas as etapas do processo.

Existem vários níveis de integração de ferramentas, diferindo no suporte fornecido. Dentre eles, pode-se citar (TRAVASSOS, 1994):

- **Integração de Dados:** A chave para integrar ferramentas é a habilidade de compartilhar informação de projeto.
- **Integração de Apresentação:** A integração de apresentação tem como objetivo tornar as interfaces do ADS consistentes, permitindo que o usuário utilize as ferramentas, alternando de uma para a outra, sem sofrer um impacto na interface. Os componentes utilizados são os mesmos, com as mesmas funcionalidades, não havendo a necessidade de aprender como utilizar cada ferramenta. A partir do momento que se aprende uma, consegue-se trabalhar mais facilmente com as outras.
- **Integração de Controle:** Para integrar um conjunto de ferramentas é preciso ter um foco forte no gerenciamento do processo. Engenheiros de software têm se tornado mais orientados a processos, usando métodos, técnicas e ferramentas para controlar e guiar seu trabalho.
- **Integração de Conhecimento:** Com o aumento da complexidade dos processos de software faz-se necessário considerar informações de natureza semântica, sem as quais a integração não é plenamente obtida. O

conhecimento, assim como os dados, deve estar disponível e representado de forma única no ambiente, de forma a poder ser acessado por todas as ferramentas que dele necessitarem, evitando redundância e inconsistências.

A dimensão de controle tem se mostrado tão importante, que deu origem a uma nova classe de ambientes, os Ambientes Centrados em Processo (ACPs). Tais ambientes concentram-se em como computadores podem ser utilizados no desenvolvimento de sistemas, focalizando a integração de controle como mecanismo para permitir a coordenação de equipes, o gerenciamento de configurações e o gerenciamento de contexto, ou seja, o controle do que é visível para cada ferramenta e para cada usuário, simplificando as interações do usuário com as ferramentas e outros usuários (CHEN et al., 1992). Porém, para que todos estes benefícios possam ser alcançados, um ACP precisa ter seus processos modelados para que possam ser automatizados.

Um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software. Um processo eficaz deve considerar as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários e a habilidade, o treinamento e a motivação do pessoal envolvido. Neste contexto, os seguintes conceitos são importantes (FALBO, 1998):

- **Atividade:** É uma tarefa ou trabalho a ser realizado, requerendo um ou mais recursos para executá-la. Uma atividade pode consumir ou produzir artefatos. Num processo real, as atividades são divididas em sub-atividades, que, por sua vez, podem também ser subdivididas. Dependendo da complexidade, uma atividade pode ser subdividida em vários níveis;
- **Artefatos:** São produtos de software. Podem ser produzidos ou consumidos por uma atividade. São exemplos de artefatos: documentos de especificação de requisitos, manuais de qualidade, atas de revisão, diagramas de classes e código fonte;
- **Recursos:** São as pessoas da organização, as ferramentas de software, os equipamentos ou quaisquer outros recursos necessários à execução de uma atividade;
- **Processo:** Um processo de desenvolvimento de software é um conjunto organizado de atividades, métodos, ferramentas, procedimentos e técnicas, com o fim de desenvolver e manter um software, e;
- **Projeto:** Consiste em um desenvolvimento de software, que emprega um processo de software, englobando o conjunto de atividades necessárias, artefatos consumidos e produzidos e recursos utilizados.

3. O PROJETO ADS

Visando suprir a necessidade de ferramentas e ambientes de desenvolvimento de software, foi criado em 1999, o *Projeto ADS*. Desenvolvido no Laboratório de Engenharia de Software (*LabES*) do Departamento de Informática (DI) da Universidade Federal do Espírito Santo (UFES), seu principal objetivo é a construção de um ambiente integrado destinado à experimentação em Engenharia de Software.

O projeto tem sido distribuído por vários trabalhos, cada um deles tratando do desenvolvimento de um módulo ou ferramenta específica do ambiente.

A plataforma de desenvolvimento do Projeto ADS está centrada na orientação a objetos, tendo sido *Java* a linguagem de programação escolhida. A escolha da linguagem Java deve-se ao fato dela suportar a Orientação a Objetos e por ser independente de plataforma, característica desejável a um ambiente que se propõe a ser de uso geral. Como mecanismo de persistência, escolheu-se o uso de Bancos de Dados Relacionais.

3.1 ESTABELECENDO A INTEGRAÇÃO

O propósito fundamental do Projeto ADS é obter um ambiente integrado, que forneça serviços de infra-estrutura, permitindo integrar ferramentas individuais ao longo das quatro dimensões anteriormente discutidas: controle, dados, interface com usuário e conhecimento.

Para isso, faz-se necessário prover uma infra-estrutura para a integração, utilizada para integrar as várias ferramentas construídas no escopo deste projeto. O trabalho desenvolvido em (MIAN, 2001) estabelece uma estratégia para a integração de controle a partir da definição e construção de uma infra-estrutura que comporta ferramentas diversas em um protótipo de um Ambiente de Desenvolvimento de Software. Seu foco principal está no acompanhamento e execução de projetos de software, de forma a automatizar o processo de desenvolvimento e fornecer apoio ao Gerente de Projeto no monitoramento do desenvolvimento de software. A meta é apoiar todo o ciclo de vida e oferecer suporte para o gerenciamento de projetos, estabelecendo uma estrutura técnica para a integração de controle.

Desta forma, ferramentas CASE de paradigmas diferentes, que apóiam várias fases do ciclo de vida do processo, podem ser incorporadas ao ADS, mantendo a integridade entre os produtos gerados por cada uma delas. Para tanto, a infra-estrutura estabelecida captura os fundamentos da gerência de projeto e a integração de ferramentas ao ambiente se dá através da especificação dos passos a serem seguidos e de quais das ferramentas disponíveis podem ser utilizadas.

A principal função de um Ambiente de Desenvolvimento de Software é executar e acompanhar um projeto de desenvolvimento. O acompanhamento permite o controle da execução das atividades, ou seja, o seqüenciamento entre as atividades é executado, considerando a hierarquia e o estado destas (ARAÚJO, 1998).

Mas, para que o controle do processo de desenvolvimento seja implementado, é necessário que este seja previamente definido. Somente após a definição do processo de software, é possível iniciar a execução e o acompanhamento de um projeto.

Assim, foi incorporado ao ambiente um modelo de conhecimento de processos, que disponibiliza informações sobre *atividades*, seus *artefatos*, os *procedimentos* que são utilizados para executá-las, os *recursos* utilizados e os *modelos de ciclos de vida* que norteiam o processo de desenvolvimento, descrevendo suas características para

utilização no ADS. Este modelo foi obtido a partir de uma ontologia[1] de processo de software, desenvolvida em (FALBO, 1998). A definição de um modelo de conhecimento sobre processos de software tem por objetivo apoiar a aquisição, organização, reuso e compartilhamento de conhecimento sobre processos de desenvolvimento de software (FALBO, 1998).

O modelo ontológico definido em (FALBO, 1998) foi implementado em (SALOMÃO, 1999), na forma de uma ferramenta para apoiar a definição de processos de software de sistemas de informação. Em (SALOMÃO, 1999), foram parcialmente construídos os modelos necessários para a definição e acompanhamento de processos de software. Esses modelos foram revistos e atualizados para adequá-los aos padrões estabelecidos no Projeto ADS.

Para acompanhar o desenvolvimento de um projeto, o ADS deve prover ao Gerente de Projeto uma forma de visualizar as atividades que compõem o processo de software, seus estados durante a execução do projeto, a ordem de precedência entre elas e se são compostas ou não por outras atividades. A representação gráfica escolhida para modelar um processo de software foi a estrutura de grafo. Na figura 1, é apresentado o significado das representações gráficas no grafo de processos. Essa estrutura foi utilizada do ADS e a representação de um processo de software no ambiente pode ser vista na figura 2.

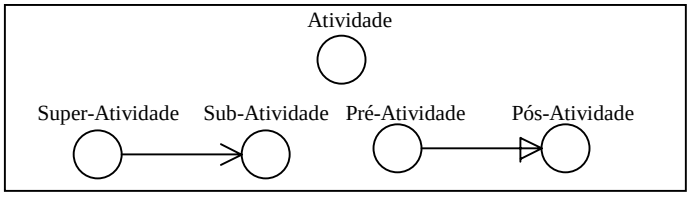

Figura 1 - Representação de Processos de Software.

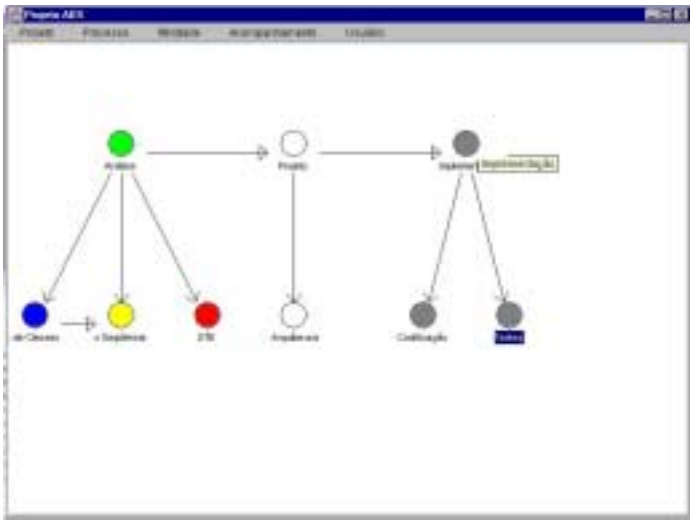

Figura 2 - Um Processo de Software no ADS.

[1] Uma ontologia uma descrição de conceitos e relações que podem existir para um agente ou uma comunidade de agentes. Consiste, basicamente, de conceitos e relações, suas definições, propriedades e restrições, descritos na forma de axiomas.

Para tratar a integração de dados, o ambiente deve manter um repositório com as informações relevantes ao desenvolvimento de software, que possa ser utilizado por todas as ferramentas. O trabalho desenvolvido em (COSTA, 2000) aborda soluções para essa questão através de esquemas de bancos de dados relacionais, possibilitando a integração, portabilidade e interoperabilidade entre ferramentas. Nesse trabalho foi definido, projetado e implementado um *framework* que permite o mapeamento de objetos em esquemas de bancos de dados relacionais. Sua finalidade é implementar um mediador entre o mundo de objetos e o mundo relacional que permita o mapeamento de objetos em esquemas de bancos de dados relacionais. Este *framework* está sendo utilizado pelas várias ferramentas do ADS, criando, como repositório central do ambiente, uma base de dados relacional.

Ao construir um ambiente integrado, com ferramentas diversas, como esse proposto pelo Projeto ADS, é necessário padronizar a interface e construí-la de forma a ser simples e fácil de utilizar. Em (VALADÃO, 2001), foi definida uma proposta para a integração de interface com o usuário, explorando os principais conceitos de interface com o usuário que auxiliam na construção de boas interfaces, apresentando os componentes já fornecidos pela linguagem Java, bem como definindo e implementando novos componentes para a camada de integração de apresentação. Os componentes de interface apresentados (tanto os já fornecidos pelo Java quanto os desenvolvidos) têm sido utilizados no projeto das interfaces das diferentes ferramentas do Projeto ADS, garantindo a integração de apresentação.

Finalmente, a quarta dimensão de integração – integração de conhecimento – foi abordada em (BERGER, 2001). Neste trabalho, foi desenvolvida uma ferramenta para criação de bases de conhecimento a partir dos modelos de conhecimento do ADS. Estes modelos, tal como o modelo de conhecimento de processos, são derivados de ontologias, dando origem a hierarquias de classes descendentes de uma classe Conhecimento, permitindo a construção automática de bases de conhecimento em Prolog. Integrando-se uma máquina de inferência Prolog ao ambiente, é possível incorporar capacidades dedutivas passíveis de uso por qualquer ferramenta do ADS.

3.2 FERRAMENTAS CASE

Estabelecida a infra-estrutura de integração, é preciso povoar o ambiente com ferramentas capazes de apoiar as atividades de engenharia de software. Outros trabalhos do Projeto ADS visam construir produtos que dão suporte às atividades de construção, gerência e controle da qualidade.

Dentre as ferramentas de construção estão aquelas capazes de automatizar métodos de análise e projeto de sistemas. O trabalho desenvolvido em (CEZANA, 2001), por exemplo, construiu um protótipo de uma ferramenta CASE para auxiliar o engenheiro de software na construção de Diagrama de Entidades e Relacionamentos, apoiando o método de Análise Essencial (POMPILHO, 1995). Existem outras ferramentas CASE em construção para dar suporte ao paradigma estruturado, dentre elas: ferramentas de apoio à

modelagem de Diagramas de Fluxo de Dados, de Listas de Eventos e de Diagramas Hierárquicos de Função.

Outros trabalhos estão sendo desenvolvidos para apoiar o paradigma Orientado a Objetos (OO). As ferramentas para Análise e Projeto Orientados a Objetos visam dar suporte à elaboração de diagramas propostos na UML (*Unified Modeling Language*). A UML é a linguagem padrão para especificar, visualizar, documentar e construir artefatos de um sistema e pode ser utilizada em todos os processos de desenvolvimento orientados a objetos, ao longo de seus ciclos de desenvolvimento, usando diferentes tecnologias de implementação (FURLAN, 1998).

As ferramentas de apoio ao paradigma OO incluem uma ferramenta de apoio à modelagem de Casos de Uso (VELLO, 2000), uma ferramenta de apoio à modelagem de classes (VELLO, 2000) (RATTES, 2001) e uma ferramenta CASE para elaboração de Diagramas de Estados (RATTES, 2001). Encontra-se em desenvolvimento, ainda, uma ferramenta para apoiar a construção de Diagramas de Seqüência. A figura 3 mostra a ferramenta de apoio à modelagem de classes.

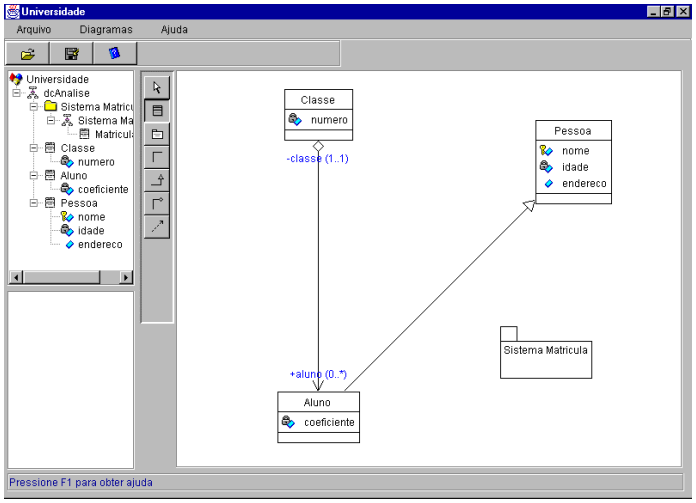

Figura 3 – Modelagem de um Diagrama de Classes .

A integração de controle adotada pelo Projeto ADS é fortemente focada no gerenciamento do processo. Esta abordagem força a definição do processo de forma rigorosa, o que permite melhoria na comunicação e compreensão entre os participantes humanos e consistência em relação a como as coisas são feitas.

Apesar de definir os processos que serão utilizados, um ambiente centrado em processo, como o Projeto ADS, por si só não é capaz de garantir a boa qualidade dos processos definidos. O primeiro passo para solucionar este problema é entender que os processos utilizados no desenvolvimento de software podem ser controlados, medidos e melhorados. Após este primeiro passo, é preciso estabelecer meios de realizar este controle e esta medição de forma a assegurar a qualidade desejada.

Surge, então, a necessidade de ferramentas para apoiar o controle da qualidade no ADS. Estas ferramentas têm como objetivo facilitar a busca por processos de maior qualidade e garantir que os produtos de software produzidos são de qualidade adequada. A qualidade é uma característica que deve ser projetada junto com o sistema e não imposta após o fato. Assim, para que se possa produzir um software de qualidade, é necessário identificar características que determinam a qualidade desejada, estabelecendo seu universo de abrangência.

A ferramenta *ControlQ* (NATALI, 2001) permite este controle da qualidade no Projeto ADS, garantindo meios de planejar, controlar e monitorar resultados do projeto, verificando se estão de acordo com os padrões de qualidade estabelecidos. Como conseqüência, obtém-se a melhoria da qualidade através da identificação das causas de resultados insatisfatórios e tomada de ações para aumentar a eficiência do projeto, provendo benefícios a todas as partes envolvidas.

Para apoiar a especificação desta ferramenta, faz-se uso de uma ontologia de qualidade de software, desenvolvida em (DUARTE, 2000). Os principais conceitos da ontologia, tais como características de qualidade de processo e de produto, característica de qualidade direta e indiretamente mensurável, sub-característica, métrica e medição, foram mapeados na ferramenta ControlQ.

Vale ressaltar que esta ontologia foi definida em conformidade com a ontologia de processo, utilizada na definição da infra-estrutura de controle do ADS, o que favoreceu substancialmente a integração da ferramenta. Desta forma, a tarefa de medir a qualidade de um determinado projeto é modelada como uma *atividade* do processo e a avaliação de um produto é a avaliação de um *artefato* de um determinado projeto.

Em ControlQ, a primeira etapa do controle da qualidade consiste na criação de um *plano de controle da qualidade* para um projeto. Este plano engloba a definição de todas as *atividades avaliadoras* que farão parte de um projeto. Ou seja, define-se uma atividade para avaliar um artefato, por exemplo, produzido em uma determinada fase do ciclo de vida de desenvolvimento do software. São estabelecidas quais características de qualidade este artefato deve possuir, suas sub-características e a partir de quais métricas estas serão medidas.

A partir destas atividades, é possível realizar a *avaliação*, ou seja, a *medição*, mostrada na figura 4. Na medição, as métricas selecionadas no plano são aplicadas e são informados os valores para as métricas. A medição é seguida da *apresentação de resultados*, mostrando o relato dos resultados obtidos. A partir da análise dos resultados obtidos espera-se que medidas corretivas sejam tomadas o mais cedo possível para garantir a qualidade desejada.

A atividade de gerência de processos é estabelecida pelo ADS através do controle e acompanhamento do processo. Mas, sendo o propósito fundamental do Projeto ADS obter um ambiente integrado, faz-se necessário apoiar também a Gerência de Configuração de Software (GCS), a fim de que a integridade dos produtos gerados em um desenvolvimento usando o ADS seja mantida. A GCS também pode ser vista como uma atividade de garantia de qualidade de software que é aplicada durante todas as fases do processo de software (PRESSMAN, 2000).

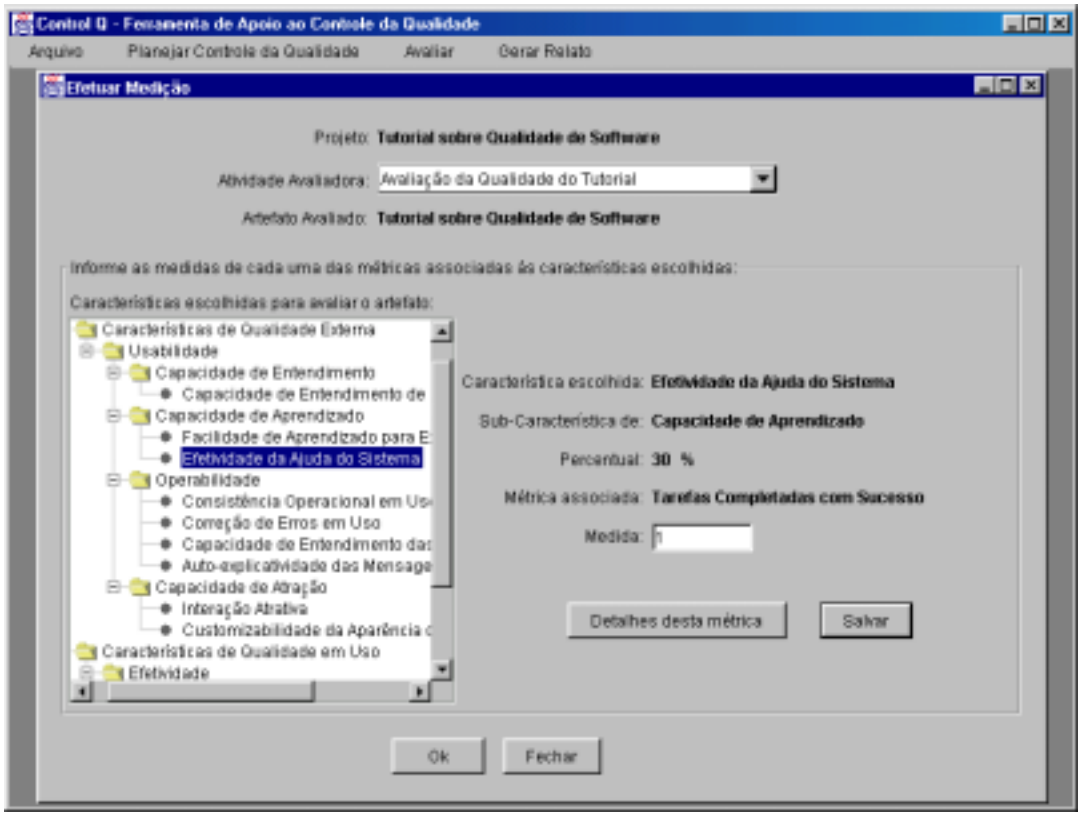

Figura 4 – Medição da Qualidade de um Projeto.

Em (REZENDE, 2001), foi desenvolvida uma ferramenta para Gerência de Configuração integrada ao ADS, que permite identificar os artefatos a serem gerenciados, prover controle de versão e controle de mudanças, apoiar auditorias e fornecer relatos de configuração dos artefatos gerenciados.

3.3 PERSPECTIVAS FUTURAS

Para oferecer suporte automatizado ao processo de software é necessário um conjunto amplo de ferramentas para apoiar suas várias atividades. No estágio atual do Projeto ADS, há poucas ferramentas desenvolvidas para apoiar atividades, sobretudo, de gerência, tal como planejamento. Assim, outras ferramentas estão sendo desenvolvidas, como uma ferramenta para apoiar a realização de estimativas usando pontos de função e uma ferramenta para apoiar a análise de riscos. Contudo, outras necessidades surgirão e, portanto, o desenvolvimento de novas ferramentas deve ser contínuo. Além disso, espera-se incorporar novas tecnologias que têm potencial promissor no apoio ao processo de software, entre elas gerência de conhecimento, ontologias e agentes.

Sempre que o conhecimento requerido não está disponível, o engenheiro de software tem que partir do zero para a solução de um problema, experimentar e construir novamente o conhecimento, o que pode sacrificar a qualidade da solução ou o tempo gasto, além abrir espaço para que erros cometidos anteriormente se repitam.

Utilizando uma abordagem de gerência de conhecimento (LIMA et al., 2000), espera-se disponibilizar o conhecimento adquirido ao longo de vários projetos para o engenheiro de software, que também poderia contribuir para a base de experiências do ADS.

Contudo, para que a gerência de conhecimento seja possível, é fundamental que o conhecimento existente no ambiente esteja mapeado, disponível e que seja entendido por todos. Uma abordagem útil para implementar a infra-estrutura necessária à gerência de conhecimento é a utilização de ontologias.

O Projeto ADS já utiliza duas ontologias: uma de processo de software e outra de qualidade de software. Novas ontologias podem ser desenvolvidas, tal como uma ontologia de artefatos de software, que poderia ser utilizada na construção de meta-ferramentas.

Porém, a construção de ontologias não é uma tarefa simples. Envolve a especificação de conceitos e relações que existem em um domínio de interesse, além de suas definições, propriedades e restrições, descritas na forma de axiomas. Para apoiar esta atividade, está-se desenvolvendo um editor de ontologias, com funcionalidades para apoiar a definição de conceitos, relações e axiomas, permitindo, inclusive, a geração automática de alguns axiomas e *frameworks* de objetos a partir das ontologias.

A modelagem do conhecimento existente no ambiente abre espaço para a utilização de agentes inteligentes na automatização do processo. Um agente inteligente é aquele capaz de tomar ações autônomas e flexíveis, onde flexibilidade significa: ser reativo (ser capaz de perceber o ambiente e responder a mudanças nele), ser pró-ativo (ter

comportamento dirigido a metas, tomando a iniciativa de modo a satisfazer seus objetivos de projeto) e ter habilidade social (ser capaz de interagir com outros agentes de modo a satisfazer seus objetivos de projeto – cooperação) (WOOLDRIDGE, 1999). Sistemas multi-agentes podem trazer muitos benefícios, desde que seus agentes consigam se comunicar adequadamente. Esta comunicação é mais facilmente conseguida quando eles compartilham um vocabulário definido por meio de ontologias.

No contexto do Projeto ADS, sistemas multi-agentes poderiam ser utilizados para apoiar diversas atividades. Na ferramenta de avaliação de qualidade ControlQ (NATALI, 2001), por exemplo, agentes poderiam auxiliar na obtenção de métricas, agora coletadas automaticamente e auxiliar na interpretação dos resultados obtidos, consultando uma base de conhecimentos específica.

4. CONCLUSÃO

Neste artigo foi apresentado o Projeto ADS do Laboratório de Engenharia de Software da UFES. O principal objetivo do projeto é a construção de um ambiente integrado para experimentação em Engenharia de Software. Foram apresentadas as ferramentas e módulos construídos para estabelecer as quatro dimensões de integração no ambiente: dados, apresentação, controle e conhecimento. Também foram descritas algumas ferramentas CASE, que automatizam atividades do processo de desenvolvimento de software, já construídas no escopo desse projeto.

Os trabalhos desenvolvidos até então dão suporte à automatização de processos de software, à gerência de projetos e à avaliação da qualidade. Mas, para que o ambiente proposto seja capaz de apoiar cada uma das atividades do processo de software, é preciso que novas ferramentas sejam desenvolvidas e incorporadas ao ADS.

AGRADECIMENTOS

Os autores agradecem aos demais participantes do Projeto ADS por sua colaboração no desenvolvimento deste artigo. P.G. MIAN e A.C.C. NATALI são bolsistas da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior).

REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, M.A.P., *Automatização do Processo de Desenvolvimento de Sofware nos Ambientes Instanciados pela Estação Taba*, Tese de Mestrado, Rio de Janeiro, 1998.

BERGER, P.M., *Uma Ferramenta de Criação e Edição de Bases de Conhecimento em um Ambiente de Desenvolvimento de Software*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2001.

CEZANA, C.G., *Ferramenta CASE-DER: Apoio à Modelagem de Entidades e Relacionamentos*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2001.

CHEN, M., NORMAN, R. J, *A Framework for Integrate CASE*, IEEE Software, March, 1992.

COSTA, P.D., *Um Framework para Persistência de Objetos em Bancos de Dados Relacionais*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2000.

DUARTE, K. C., FALBO, R.A., *Uma Ontologia de Qualidade de Software*, Anais do WQS'2000, Workshop de Qualidade de Software, XIV Simpósio Brasileiro de Engenharia de Software, João Pessoa - PB, Outubro 2000.

FALBO, R. A., *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*, Tese de Doutorado, COPPE/UFRJ, 1998.

FURLAN, J.D., *Modelagem de Objetos Através da UML*, Makron Books. 1998.

LIMA, K.V.C., et al.; *Ambientes de Desenvolvimento de Software Orientados a Organização*, Relatório Técnico do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 2000.

MIAN, P.G., *Integração de Controle em um Ambiente de Desenvolvimento de Software*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2001.

NATALI, A.C.C., *Control Q - Uma Ferramenta de Apoio ao Controle da Qualidade*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2001.

POMPILHO, S., *Análise Essencial: Guia Prático de Análise de Sistemas*. IBPI Press, Livraria e Editora Infobook, Rio de Janeiro, 1995.

PRESSMAN, R.S., *Software Engineering: A Practitioner's Approach*, 5th Edition, Mc Graw Hill, 2000.

RATTES, P.D., *CASE para Tecnologia de Objetos: Suporte à Modelagem de Classes e Estados*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2001.

REZENDE, C.F., *Uma Ferramenta de Apoio à Gerência de Configuração de Software*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2001.

SALOMÃO, R.C., *Assistente Inteligente para Apoiar a Definição de Processos de Software de Sistemas de Informação*, Projeto de Graduação, Curso de Ciência da Computação da UFES, 1999.

TRAVASSOS, G.H., *Ambientes de Desenvolvimento de Software*, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, 1994.

VALADÃO, J.T., *Integração de Apresentação em um Ambiente de Desenvolvimento de Software*, Projeto de Graduação, Curso de Engenharia de Computação, UFES, 2001.

VELLO, R.M.A., *CASE para Tecnologia de Objetos: Suporte a Casos de Uso e Modelagem Estática de Classes*, Projeto de Graduação, Curso de Ciência da Computação, UFES, 2000.

WOOLDRIDGE, M., "Intelligent Agents", In: G. Weiss, editor: Multiagent Systems, The MIT Press, 1999.