Coleta e análise de métricas do processo de aprendizagem de programação

Daniel dos Santos Pêgas, Edgar Toshiro Yano

Departamento de Computação - Instituto Tecnológico de Aeronáutica
Praça Marechal Eduardo Gomes, 50 - Vila das Acácias
CEP 12228-900- São José dos Campos - SP – Brasil

{dspegas,yano}@ita.br

Abstract. *In a programming course the pupil develops several practical activities to exercise and consolidate the concepts and techniques presented in different lessons. In order to support the learning and evaluation process it is proposed a data collection and analysis system. This system will collect data from pupils development projects. It is expected that these data will be able to supply clues and evidences to better comprehend learning activities and support new strategies to improve the performance of programming language courses.*

Resumo. *Em um curso de programação o aluno desenvolve várias atividades práticas para exercitar e consolidar os conceitos e técnicas vistas em aula. De modo a suportar tanto o aprendizado do aluno como a avaliação do professor, é proposto um sistema de coleta e análise de dados dos projetos desenvolvidos pelos alunos. Esta coleta poderá fornecer evidências e indícios para melhor comprender o processo de aprendizagem e suportar novas estratégias para melhorar o desempenho de cursos de programação.*

Palavras-chave: Educação à distância, programação, observação eletrônica, monitoramento.

1. Introdução

O processo de aprendizagem de programação requer que o aluno desenvolva, implemente e teste um conjunto de projetos para exercitar conceitos e técnicas vistas em aulas. Esse processo apresenta entre outras as seguintes dificuldades:

O desenvolvimento de um único projeto é geralmente demorado requerendo várias horas ou dias de trabalho. O professor não consegue, assim, fazer um acompanhamento mais próximo do trabalho do aluno.

A avaliação do trabalho realizado é complexa e por causa disso bastante imprecisa. O professor muitas vezes não sabe com precisão avaliar se os objetivos do projeto foram atingidos, se as técnicas recomendadas foram utilizadas, quantas horas o aluno realmente gastou para implementar o projeto, etc.

A avaliação comparativa de desempenho dos diferentes alunos é muito subjetiva, pois cada projeto desenvolvido apresenta peculiaridades distintas, ainda que trate de um mesmo tema.

Dadas as dificuldades mencionadas torna-se interessante o desenvolvimento e uso de um ambiente de aprendizado de programação que, de forma transparente, colete e armazene dados do processo de programação. Os dados coletados podem ser, então, posteriormente analisados pelos professores.

Com os dados coletados poderemos obter respostas para várias questões tais como: Quando o aluno realmente iniciou um projeto? Como foi a evolução dos arquivos do projeto? Qual foi o tempo gasto com depuração? Quais trechos de programas foram mais avaliados pelo depurador? Quais foram os erros mais freqüentes de compilação? Quais foram os erros mais comuns da turma? Quais são os alunos que levam mais tempo para terminar um trabalho? Por que os alunos gastam tanto tempo para executar uma tarefa? Qual é o horário mais utilizado para a elaboração de projetos?

As respostas para essas questões poderão direcionar os professores a redefinirem estratégias para melhorar o desempenho de cursos de programação.

O artigo descreve um projeto de pesquisa em andamento que tem como propósito o desenvolvimento de um ambiente de ensino de programação que colete, de forma transparente, dados sobre o desempenho de projetos de programação. Esse ambiente, além de suportar as funcionalidades para coleta de dados, apresenta também os seguintes requisitos:

- Extensão (plug-in) de um ambiente de programação disponível e em uso em cursos de programação.
- Utilização das facilidades da Internet para o acompanhamento remoto de projetos de alunos.
- Arquitetura aberta de modo a ser independente de produtos ou tecnologias específicas.

Descreveremos a seguir os seguintes itens: levantamento de requisitos e descrição de cenários, projeto de arquitetura de sistema, modelagem dimensional de datawarehouse, vantagens e limitações do sistema e conclusões.

2. Levantamento de requisitos e cenários de uso

O levantamento de requisitos foi executado através das seguintes atividades:

1. Identificação preliminar de métricas.
2. Pesquisa com professores de cursos de programação.
3. Análise dos resultados da pesquisa.

A identificação preliminar de métricas foi executada através da elaboração de uma lista de métricas obtida através de experiências dos autores com ambientes de desenvolvimento de software. A partir dessa lista de métricas foi elaborado um questionário distribuído para professores de diferentes cursos de programação. Foram envolvidos professores de três instituições de ensino; uma das instituições é de ensino médio profissionalizante, outra é uma instituição com curso de graduação e a terceira apresenta cursos de graduação e de pós-graduação.

Na tabela 1 temos as questões apresentadas aos entrevistados, todas foram categorizadas como muito útil, útil, moderado, pouco útil ou inútil.

Tabela 1. Questionário distribuído aos professores

1.Saber quantas vezes o aluno compilou o projeto até chegar à versão final é
2.Saber se o aluno mexe nas configurações do compilador é
3. Saber o histórico de erros, ou seja, ter a lista de todos os erros ocorridos a cada compilação que o aluno executa é
4.Saber quanto tempo a aluno passa depurando (debug) o projeto em relação ao tempo total de desenvolvimento é
5.Saber o momento que o aluno começa a usar o debug, por exemplo, para saber qual erro após uma compilação o aluno esta tentando solucionar é
6.Saber se com o debug o aluno passou por todo o código é
7.Saber quais os breakpoints que o aluno utiliza é
8.Saber quantas vezes o aluno passa com o debug por determinado trecho do código é
9.Saber quantas variáveis e/ou expressões, estado da pilha, threads que o aluno monitora durante o debug é
10.Possuir estatísticas de erros mais comuns de alunos e/ou turmas é
11.Possuir estatísticas de tempos de alunos e/ou turmas é
12.Saber quantas vezes o aluno usa o help durante o desenvolvimento é
13.Saber o que o aluno pesquisou no help é
14.Saber o horário e o local (laboratório, em casa, etc...) que o aluno desenvolveu o projeto é
15.Saber qual parte do programa é reaproveitado de programas anteriores é
16.Saber se durante o desenvolvimento o aluno “cola” trechos de código é
17.Saber se o aluno utiliza “wizards” é
18.Saber se os padrões recomendados foram utilizados é
19.Saber a relação entre linhas de código e comentários é
20.Saber a evolução do arquivo fonte (tamanho, número de classes, etc...) é
21.Você acha que comparar os programas entregue pelos alunos para saber se há partes comuns” é
22.Saber se o aluno customiza o ambiente de desenvolvimento é
23.Possuir relatórios sobre todo o trabalho realizado pelo aluno no ambiente de desenvolvimento de software para futuras comparações é
24.Possuir relatórios de uma turma sobre o desenvolvimento de um projeto para futuras comparações é

Além destas questões havia uma área para sugestões. Na tabela 2 temos as respostas, já totalizadas e em porcentagens.

Tabela 2. Respostas obtidas, totalizadas e em porcentagens.

Pergunta	Muito Útil	Útil	Moderado	Pouco Útil	Inútil
1.	0	30	40	30	0
2.	0	40	30	20	10
3.	40	20	20	20	0
4.	40	50	10	0	0
5.	30	20	30	20	0
6.	20	30	30	20	0
7.	10	30	0	50	10
8.	0	40	30	20	10
9.	30	20	30	20	0
10.	70	20	10	0	0
11.	30	50	20	0	0
12.	10	20	60	10	0
13.	20	40	30	10	0
14.	20	0	40	30	10
15.	20	30	40	10	0
16.	20	20	10	40	10
17.	10	40	20	30	0
18.	33	57	11	0	0
19.	30	30	30	10	0
20.	10	50	30	0	10
21.	40	40	0	20	0
22.	0	40	40	20	0
23.	60	10	20	0	10
24.	60	10	20	10	0

Na tabela 2 foram selecionadas as respostas cuja soma das porcentagens de respostas úteis e muito úteis foram superior ou igual a cinqüenta por cento; com isso foram identificados os seguintes requisitos para entender a aprendizagem do aluno:

- Relacionar todos os erros acusados, cada vez que o aluno compila o projeto.
- Mostrar a relação entre o tempo que o aluno passa depurando o projeto (debug) com o tempo total gasto no desenvolvimento
- Mostrar o momento em que o aluno começa a usar o debug
- Mostrar se o aluno passou por todo o código com o debug
- Mostrar o que o aluno monitora durante o debug (variáveis, expressões e threads)
- Mostrar estatísticas de erros mais comuns de um aluno e/ou turma
- Mostrar estatísticas de tempos de um aluno e/ou turma
- Mostrar o que o aluno pesquisou no help
- Mostrar os trechos de código que foram aproveitados de atividades anteriores
- Mostrar se o aluno utiliza “wizards”
- Mostrar se os padrões recomendados foram utilizados

- Mostrar a relação entre linhas de códigos e comentários
- Mostrar a evolução do arquivo fonte

Muitos entrevistados ressaltaram a importância de saber a qualidade dos comentários, se as estruturas de decisão e de repetição estão sendo usadas corretamente, o uso correto de passagem de parâmetros e especialmente a importância de comparar os resultados das análises obtidas com analises de fluxogramas, algoritmos e diagramas que os alunos venham a desenvolver em conjunto. Esta comparação levará, segundo os entrevistados, a uma visão mais detalhada de como o aluno aprende linguagens de programação.

Com as informações obtidas elaboramos a seguinte lista de requisitos para o sistema proposto:

1. Armazenar horário e data de inicio e fim do uso do debug;
2. Armazenar linha inicial e final por onde passa o debug;
3. Armazenar erros acusados pelo compilador;
4. Armazenar horário e data de inicio e fim de uma compilação;
5. Armazenar horário e data, bem como os próprios eventos gerados pelo compilador;
6. Armazenar variáveis, expressões e threads que são monitoradas pelo debug;
7. Armazenar linha de inicio e fim de cada comentário e classificá-los quanto a sua relevância;
8. Armazenar número de classes, variáveis, funções e referências do projeto de acordo com o tempo;
9. Armazenar data e hora que o aluno abre o projeto;
10. Armazenar número IP da máquina que o aluno esta utilizando;
11. Armazenar data, hora e expressão utilizada pelo help;
12. Armazenar linha de inicio e fim de um padrão recomendado identificado;
13. Cadastrar e identificar o aluno através de login e senha;
14. Armazenar teclas relevantes acionadas pelo aluno;
15. Armazenar data e hora de inicio e fim da atividade;
16. Disponibilizar atividade para o aluno;
17. Analisar informações e
18. Mostrar resultado das análises.

3. Cenários de uso

Com base nesses requisitos temos vários cenários que ilustram melhor como o sistema pode colaborar para melhorar o processo de aprendizado de programação.

3.1. Cenário Professor

O professor configura o módulo de dados para um novo curso da linguagem C#. O professor registra os alunos matriculados, cria repositórios para os projetos e configura os coletores de dados a serem instalados em cada ambiente de desenvolvimento de aluno. Estes coletores são instalados de forma transparente assim que o aluno se conectar ao ambiente de desenvolvimento.

Para cada projeto passado para os alunos o professor obtém dados de desempenho individual ou coletivo. O professor pode acompanhar a evolução do código fonte e o estado corrente do projeto de um determinado aluno ou de toda a classe.

Para avaliar um projeto o professor poderá ter com precisão a data de início e término do trabalho e ter uma boa idéia da qualidade do código fonte em função do tempo que o aluno gastou para depurar um programa e estrutura dos arquivos desenvolvidos. Um analisador de código poderá verificar a presença de comentários, aplicação de princípios de programação modular, etc.

O professor poderá fazer uma avaliação comparativa e verificar se os tempos de início e entrega de trabalhos e atributos de qualidade estão compatíveis com a turma corrente ou turmas passadas.

3.2. Cenário aluno

O aluno ao receber um projeto de estrutura de dados faz um esboço em papel, usando pseudo-código, dos algoritmos a serem utilizados. Para ter acesso ao ambiente de desenvolvimento ele ativa o módulo de aluno passando o nome de login e senha. Todos os arquivos fonte criados ou utilizados pelo aluno estarão em repositórios controlados pelo módulo de dados. Com repositório com acesso controlado o aluno não poderá fazer uso de arquivos não autorizados.

O módulo de aluno monitora o processo de criação e uso de arquivos fonte e objeto, uso de compilador, uso de depurador, uso do help e uso de wizards (ferramentas de geração de código).

Em cada sessão de trabalho ficam registrados: quais foram os arquivos criados, quantas vezes foi ativado o compilador, quais foram os erros de compilação mais comuns, quantas vezes foi ativado o depurador, quais trechos de programas foram visitados pelo depurador, como os wizards foram utilizados e como o help foi ativado.

3.3. Cenário supervisor de curso

Um supervisor de curso poderá analisar o andamento de um curso verificando quais projetos foram passados pelos professores, qual é o desempenho das diferentes classes ou o desempenho de um aluno ou grupo de alunos. Um supervisor poderá, por exemplo, avaliar o impacto do projeto de uma matéria sobre projetos de outras matérias.

4. Arquitetura do sistema

O sistema está dividido em três módulos conforme figura 1.

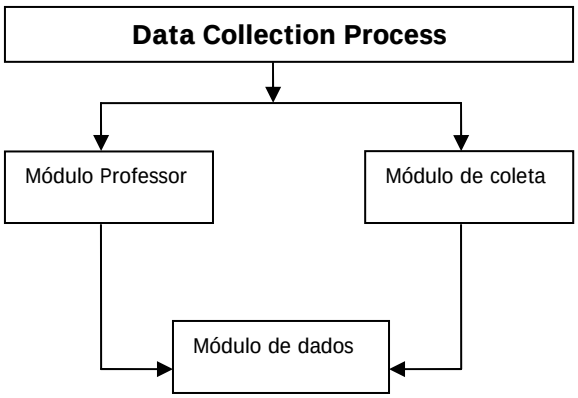

Figura 1. Arquitetura do sistema.

O módulo de aluno consiste em um sistema instalado no ambiente de trabalho do aluno e de forma transparente, coleta e transmite ao módulo de dados, as informações sobre as atividades do aluno no ambiente de desenvolvimento. A única funcionalidade que é apresentada ao aluno é uma tela de identificação (login) para que o sistema identifique o aluno cujas informações estão sendo coletadas. O módulo de aluno é instalado como uma extensão do tipo plug-in em um ambiente de desenvolvimento como o MS-VisualStudio.net.

O módulo do Professor consiste em um sistema de administração, onde o professor irá configurar o processo de coleta, bem como ter acesso a relatórios referentes às informações coletadas de determinada atividade, de um aluno ou de uma turma. O módulo de professor é acessado através um navegador Web tal como IExplorer.

O módulo de dados consiste em um ou mais repositórios de arquivos com acesso controlado, um datawarehouse para armazenamento de dados coletados e um conjunto de ferramentas para consultar, analisar e apresentação das informações.

A comunicação entre os módulos de aluno e o módulo de dados será feita via Web Services, assim teremos a flexibilidade do aluno poder desenvolver a atividade em casa, de qualquer computador conectado à internet, ou de um laboratório da própria instituição de ensino, via rede local.

A implementação do módulo de dados e de alunos está sendo feita através do uso das seguintes ferramentas: VisualStudio.net, MS-Sqlserver e servidor web IIS. Entretanto, a arquitetura foi concebida de modo a permitir a criação de uma implementação com ferramentas de código aberto.

5. Modelo dimensional

O núcleo do sistema proposto é o módulo de armazenamento de dados que utiliza a tecnologia de datawarehouse [Kimball 1998]. Para a elaboração do modelo de datawarehouse foi feito um trabalho de modelagem dimensional. O modelo dimensional do datawarehouse proposto é o representado na figura 2.

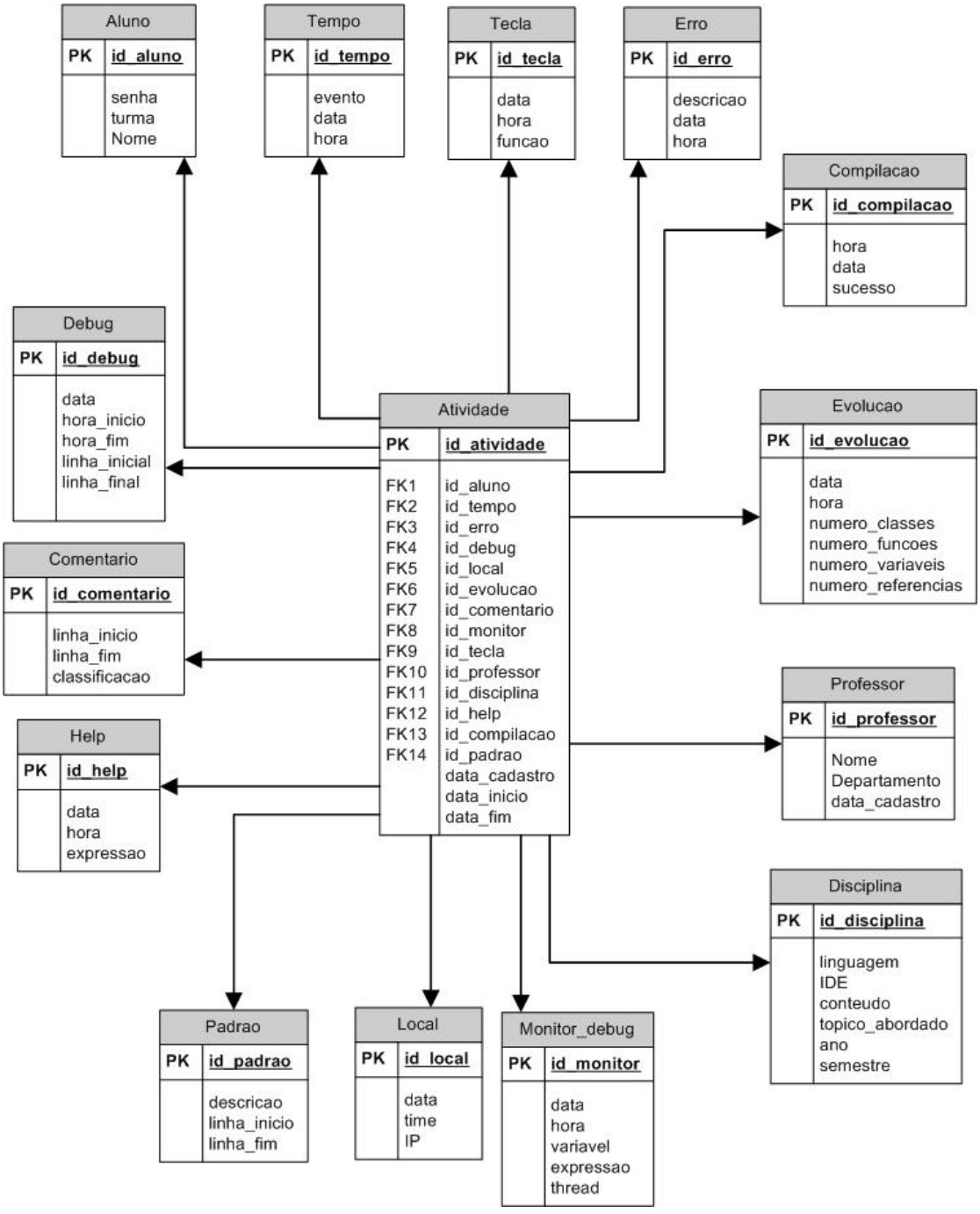

Figura 2. Modelo dimensional do data warehouse proposto

A tabela de fatos foi escolhida como sendo a Atividade, pois é o foco central da coleta de dados. A partir da atividade podemos ter informações sobre o aluno, grupos ou turmas.

Cada dimensão corresponde a requisitos do sistema, como descrito abaixo:

- Aluno: Cadastrar e identificar o aluno através de login e senha;
- Erro: Armazenar erros acusados pelo compilador;
- Compilação: Armazenar horário e data de inicio e fim de uma compilação;
- Debug: Armazenar horário e data de inicio e fim do uso do debug e armazenar linha inicial e final por onde passa o debug;
- Comentário: Armazenar linha de inicio e fim de cada comentário e classificá-los quanto a sua relevância;
- Help: Armazenar data, hora e expressão utilizada pelo help;
- Padrão: Armazenar linha de inicio e fim de um padrão recomendado identificado;
- Local: Armazenar data e hora que o aluno abre o projeto e Armazenar número IP da máquina que o aluno esta utilizando;
- Monitor_debug: Armazenar variáveis, expressões e threads que são monitoradas pelo debug;
- Evolução: Armazenar número de classes, variáveis, funções e referências do projeto de acordo com o tempo;
- Tempo: Armazenar horário e data, bem como os próprios eventos gerados pelo compilador;
- Tecla: Armazenar teclas relevantes acionadas pelo aluno;

Algumas dimensões foram adicionadas, como descritas abaixo:

- Disciplina: armazena informações sobre a disciplina da qual a atividade faz parte, com isso podemos obter comparações relacionadas a diferentes linguagens, IDEs de desenvolvimento e tópicos específicos ao longo do tempo (ano e semestre);
- Professor: armazena as informações sobre o professor que ministra a disciplina, a fim do supervisor obter as informações descritas no cenário Supervisor de curso.

6. Projetos similares

O departamento de computação da Open University, possui em andamento um projeto de pesquisa denominado ASEOP, [Paine 1999], que consiste em coletar dados de um ambiente de aprendizagem (Smalltalk) de orientação a objeto para:

- Informar o educador qual foi o melhor efeito da aula;
- Prover um aparato para identificar problemas que estudantes podem experimentar enquanto aprendem;
- Prover evidências que melhorem o projeto do ambiente de programação.

Inicialmente o projeto contou com a coleta de informação de 30 alunos, em 2000 este número estava em torno de 300 estudantes.

A diferença entre o ASEOP e o sistema proposto é que o primeiro funciona em um ambiente desenvolvido pela Open University, com características próprias e não em um ambiente de desenvolvimento de software comercial como o segundo.

Também na Open University há um outro enfoque que trabalha em conjunto com o projeto ASEOP que trata da gravação e análise do trabalho feito para entender as atividades práticas [Thomas 2002]. Como é um projeto em paralelo contou com os mesmos alunos do projeto ASEOP divididos em 8 grupos de atividades. Puderam ser constatadas diferenças significantes entre a percepção do autor do curso e como os alunos se comportam realmente, principalmente do que diz respeito ao tempo de resolução das atividades. Este trabalho se assemelha ao módulo de dados do sistema proposto neste artigo.

A University of Glasgow possui um projeto de pesquisa (GRUMPS - Generic Remote Usage Measurement Production System) para desenvolver técnicas e software para automaticamente coletar, gerenciar e analisar ações do usuário, [Atkinson 2001], visando atingir as seguintes metas:

- Facilitar a organização de experiências que eficientemente coletam informações de usuários remotos;
- Facilitar a análise de repositórios que contenham estas informações a fim de testar hipóteses sobre as atividades dos usuários e facilitar o suporte a eles;
- Descobrir se esta abordagem pode melhorar a qualidade de sistemas de informação distribuídos e
- Descobrir se esta abordagem é válida para áreas de atuação específicas como educação, avaliação da usabilidade e ciência.

A Monash University está realizando um estudo para entender melhor o esforço do estudante na tarefa de desenvolver programas. Para isso estão desenvolvendo um plug-in para o Visual Studio .NET 2003 Academic para capturar conteúdo dos projetos, progresso de compilação e estatísticas de execução, [Redmond 2004]. Este projeto ainda está em fase de definição dos requisitos necessários.

A University of Hull também está realizando um estudo para entender os hábitos dos estudantes de programação. Para isto estão desenvolvendo um plug-in para o Visual Studio.Net 2003 Academic para capturar quantas vezes o aluno compila um projeto para

entender como o aluno trabalha nele e por quanto tempo, [Grey 2004]. O sistema proposto neste artigo coleta mais métricas a fim de entender o esforço do aluno no aprendizado de linguagens de programação.

7. Vantagens e limitações do sistema proposto.

O sistema em desenvolvimento apresenta as seguintes vantagens:

1. Transparente;
2. Extensível;
3. Arquitetura aberta;
4. Permite o acompanhamento de projetos sendo executados em um laboratório ou em casa em qualquer hora.

   E limitações:

1. A primeira versão não permite a análise de documentos que não sejam o código fonte.

8. Conclusões e comentários finais

O aprendizado de linguagens de programação é uma tarefa ainda não muito compreendida e de difícil acompanhamento. O sistema em desenvolvimento utiliza, de forma integrada, as tecnologias da Internet, datawarehouse e ambientes de desenvolvimento visuais para coletar e analisar dados obtidos do processo de aprendizagem de programação.

A arquitetura utilizada para o sistema pode ser facilmente adaptada para outros processos de aprendizagem. Como extensão futura desse projeto está a construção de um sistema de monitoramento de uso de objetos de aprendizagem (learning objects). A criação e uso de objetos de aprendizagem apresentam características similares ao processo de aprendizagem de linguagens de programação. Assim como um professor de programação deseja ter métricas sobre a evolução de arquivos de programas, um autor de um objeto de aprendizagem deseja também ter métricas para a avaliação da efetividade do uso de um determinado objeto.

A coleta e análise de métricas de aprendizagem é sem dúvida um campo a ser explorado. O uso adequado das tecnologias providas pela Internet e banco de dados relacionais pode oferecer respostas para muitas questões ainda não resolvidas sobre processos de aprendizagem.

9. Referências Bibliográficas

KIMBALL, Ralph. *Data Warehouse Toolkit* .São Paulo:Makron Books, 1998

KIMBAL, Ralph et al. *The Data Warehouse Lifecycle Toolkit: experts methods for designing, developing and deploying data warehouses*. New York:Wiley, 1998.

KIMBALl, Ralph; MERZ, Richard. *Data Webhouse: construindo o Data Warehouse para a WEB*. Rio de Janeiro: Campus, 2000.

THOMAS, Pete; PAINE, Carina. *Monitoring distance education students' practical programming activities. 2002*. Disponível em: <http://ifets.ieee.org/periodical/vol_3_2002/ thomas.html>. Acesso em: 17 abril 2004

PAINE, Carina; THOMAS, Pete. *AESOP-An Electronic Student Observatory Project.* 1999. Disponível em: <http://mcs.open.ac.uk/aesop/>. Acesso em: 17 abril 2004.

ATKINSON, Malcolm et al. GRUMPS - *Generic Remote Usage Measurement Production System.* 2001. Disponível em : <http://grumps.dcs.gla.ac.uk/>. Acesso em: 18 abril 2004

GREY, David J. *Student Profiling & Course Management in Assignment Manager*. 2004. Disponível em : <http://www2.dcs.hull.ac.uk/people/cssdjg/Projects/AM/>. Acesso em: 05 abril 2004.

REDMOND, Wash. *Microsoft Selects Five Universities To Enhance Visual Studio .NET 2003 Academic. 2003*. Disponível em : <http://www.microsoft.com/presspass/press/2003/nov03/11-04UniversitiesVSTOPR.asp>. Acesso em: 05 abril 2004.