OntoRevPro: Uma Ontologia sobre Revisão de Programas para o Aprendizado Colaborativo de Programação em Java

Maria de Fátima Neves, Juan Manuel Adán Coello

Faculdade de Engenharia de Computação, Pontifícia Universidade Católica de Campinas, BR.

fatimaneves@click21.com.br juan@puc-campinas.edu.br

Abstract. *Learning computer programming is one of the first and most challenging tasks for computing students. The difficulties faced by the students can be perceived by the high degrees of failure and the difficulties presented in the courses directly dependent on the abilities to program. This work aims at contributing to alleviate this problem providing tools to help students to collaborate reviewing programs made by their peers. The approach is to create an ontology that conceptualizes and formalizes the process of reviewing Java programs. This ontology makes the review process explicit permitting that all participants of the process share a common understanding about it.*

Resumo. *Em cursos de graduação na área de Computação e Informática, muitos estudantes encontram dificuldades no aprendizado de programação nas disciplinas introdutórias. Isso pode ser constatado através do alto grau de repetência e baixo índice de assimilação dos estudantes nas disciplinas cujos requisitos exigem tais conhecimentos. Este trabalho procura contribuir para a superação desse problema fornecendo meios para ajudar na colaboração entre os alunos no aprendizado de programação Java através da revisão de programas. A proposta é conceitualizar e formalizar o processo de revisão de programas Java através de uma ontologia tornando explícito este processo e permitindo um entendimento comum sobre o mesmo.*

1. Introdução

As linguagens de programação são o coração da ciência da computação. Elas são as ferramentas que utilizamos para a comunicação não somente com computadores, mas também com pessoas. O desafio de fazer uso de forma apropriada das características de uma linguagem para a expressão clara de algoritmos é parte excitante do estudo de linguagens de programação [Ghezzi e Jazayeri, 1998].

Por sua importância no campo da Ciência da Computação, o estudo de programação de computador é disciplina obrigatória nos cursos de graduação na área de Computação e Informática. No entanto, os estudantes normalmente encontram dificuldades nesse aprendizado. As dificuldades encontradas são diversas e podem ser percebidas através do alto grau de repetência e das dificuldades demonstradas nas disciplinas diretamente dependentes das habilidades de programar, de dominar o raciocínio lógico e de resolver problemas.

Muitos são os motivos que levam os alunos a ter dificuldades em aprender a programar e um desses motivos é o fato de os professores não poderem acompanhar de

forma efetiva as atividades desenvolvidas pelos alunos devido à grande quantidade de alunos que geralmente estão sob a sua supervisão [Tobar et al., 2001].

Ao trabalhar com muitos alunos ao mesmo tempo, o professor geralmente mostra somente uma forma de resolver um determinado problema para todo o grupo, não podendo explorar e discutir outras formas de implementação possíveis. Não existe um trabalho individualizado junto ao aluno e mesmo quando o professor avalia cada programa escrito, essa avaliação muitas vezes é superficial devido à grande quantidade de programas a serem avaliados. Além do mais, nesse contexto, o aluno demora a ter um retorno sobre o trabalho que desenvolveu.

Muitos trabalhos encontrados na literatura apresentam propostas para reduzir as dificuldades no aprendizado de programação, alguns deles utilizando estratégias de colaboração entre os alunos para auxiliar no aprendizado. Em um desses trabalhos, Tobar et al., (2001) propõem um ambiente colaborativo para o aprendizado de programação que permita o envolvimento de estudantes, professores e sistemas inteligentes. Entre as estratégias de colaboração consideradas pelo ambiente está a de revisão de programas entre os alunos, onde cada programa submetido ao ambiente é comentado por um ou mais estudantes e posteriormente pode ser melhorado pelo aluno autor do programa baseado nos comentários recebidos.

Neste artigo é apresentada uma ontologia desenvolvida para auxiliar na revisão de programas escritos na linguagem Java, visando ao aprendizado colaborativo de programação usando essa linguagem. A linguagem Java foia a escolhida porque na crescente utilização do paradigma de orientação a objeto Java é uma das linguagens mais populares, sendo utilizada em um grande número de cursos na área de Computação e Informática.

O texto a seguir apresenta, na seção 2, uma breve revisão de literatura sobre aprendizado colaborativo e revisão pelos pares; na seção 3 é feita uma conceituação de ontologia e são apresentados alguns benefícios decorrentes da sua utilização; na seção 4 é apresentada a OntoRevPro e, finalmente, a seção 5 apresenta algumas considerações finais.

2. Aprendizado Colaborativo

O aprendizado baseado na interação entre os aprendizes vem sendo defendido por diversas teorias educativas como as teorias de Piaget e de Vigotsky no âmbito da psicologia educacional. A idéia principal dessas abordagens é a defesa de um sujeito que construa o conhecimento em colaboração com outros, ou seja, segundo essas teorias, aprender é uma atividade social mediada pelos colegas e pelo professor e não uma tarefa isolada.

Os ambientes de aprendizagem colaborativa têm sido vistos como benéficos, tanto em aspectos cognitivos quanto em aspectos sociais. Assim, o foco não está mais na interação entre professor e estudante, mas em como os estudantes podem interagir entre si e como eles podem ensinar uns aos outros [Ramos et al., 2002].

Em um experimento sobre interação construtiva Myiake (1986) mostrou que aproximadamente 80% da autocrítica (reflexão) ocorre durante o aprendizado colaborativo enquanto que apenas 20% ocorre durante o aprendizado individual.

Também são encontrados na literatura vários trabalhos sobre o aprendizado colaborativo de linguagem de programação. Em um desses trabalhos Chamillard e Braun (2000) afirmam que a habilidade de discutir conceitos e detalhes de implementação com outros estudantes durante o curso ajuda os estudantes a aprender mais efetivamente.

A interação entre pares motiva a resolução de problemas de programação resultando em uma solução que é freqüentemente superior àquelas que são produzidas individualmente, essa afirmação pode ser constatada em [Wilson et al., 1993; Nosek, 1998]. Em experimentos realizados por Faria e Adán Coello (2004) ficou evidente que o trabalho colaborativo resulta em programas mais compreensíveis do que programas feitos individualmente.

Uma das formas de colaboração entre os alunos é a técnica de revisão pelos pares (*peer review*), técnica esta bem conhecida no meio acadêmico e que tem sido muito utilizada visando incrementar o processo de ensino-aprendizagem e como forma de melhorar a interação entre estudantes.

De acordo com Anderson (1997), um processo de revisão pelos pares aplicado a programas tem o potencial de fornecer *feedback* para programadores, de oferecer uma experiência educacional para o revisor e o autor, de melhorar a comunicação entre membros do grupo e de melhorar a qualidade do código produzido. Isso é facilitado pela obrigatoriedade de os programadores lerem os códigos uns dos outros, receberem criticas sobre seus próprios códigos e poderem avaliar o quanto outras pessoas podem compreender seus programas.

A revisão de programas é uma forma efetiva de obter código entendível e mais fácil de manter. Em cursos de programação tradicionais, os estudantes experimentam somente um lado da revisão – o resultado da avaliação do instrutor. Com a revisão pelos pares os estudantes têm seus programas revisados por outros alunos [Zeller, 2000].

3. Ontologias

O uso de ontologias vem sendo explorado em muitos trabalhos na área da Ciência da Computação com o objetivo de padronizar informações de um determinado domínio possibilitando o seu compartilhamento e reuso por diferentes usuários. O termo surgiu na Filosofia e no início da década de 1990 começou a ser utilizado pela área de Inteligência Artificial. Segundo Pernas e Dantas (2004), ontologias foram inicialmente utilizadas pela área de Inteligência Artificial visando criar representações que fossem além da descrição de simples instâncias do domínio considerado. Atualmente são utilizadas em diversas áreas que buscam desenvolver um vocabulário contendo os conceitos relativos ao domínio da aplicação.

Muitas são as definições encontradas na literatura sobre ontologia, uma muito referenciada é dada por Gruber (1996): "*Uma ontologia é uma especificação explícita de uma conceitualização"*. Uma conceitualização é uma visão abstrata e simplificada do mundo que desejamos representar para algum propósito e consiste de um conjunto de objetos, conceitos e outras entidades sobre as quais o conhecimento está sendo expresso, e de relacionamentos entre eles. Todo modelo de conhecimento está confinado a alguma conceitualização, implícita ou explícita. Uma especificação explícita desta conceitualização é chamada ontologia.

De acordo com Guarino (1998), ontologia se refere a um artefato constituído por um vocabulário usado para descrever uma certa realidade, mais um conjunto de fatos explícitos e aceitos que dizem respeito ao sentido pretendido para as palavras do vocabulário.

O vocabulário formado por predicados lógicos forma a rede conceitual que confere o caráter intencional às ontologias. A ontologia define as regras que regulam a combinação entre os termos e as relações. As relações entre os termos são criadas por especialistas e os usuários formulam consultas usando os conceitos especificados. Uma ontologia define assim uma "linguagem" (conjunto de termos) que será utilizada para formular consultas.

Segundo Noy e McGuinness (2001) as razões para o desenvolvimento de uma ontologia são: 1) para compartilhar entendimento comum da estrutura de informação entre pessoas ou entre agentes de software; 2) permitir o reuso de onhecimento de um domínio. Caso exista uma ontologia que modele adequadamente certo conhecimento de um domínio, ela pode ser compartilhada e usada por pessoas que desenvolvam aplicações nesse e em outros domínios; 3) para tornar explicitas pressuposições de um domínio. As ontologias fornecem um vocabulário para representação do conhecimento. Esse vocabulário tem por trás uma conceitualização que o sustenta, evitando assim interpretações ambíguas; 4) para separar conhecimento de domínio de conhecimento operacional; 5) para analisar um conhecimento de um domínio.

Dentre as razões citadas para o desenvolvimento de ontologias, a mais comum é o compartilhamento do entendimento comum de uma estrutura de informação [Noy e McGuinness, 2001].

4. OntoRevPro

Tendo como motivação a proposta de Tobar et al. (2001) citado anteriormente, foi feita uma pesquisa em busca de ferramentas que auxiliassem na revisão de programas entre alunos. Essa pesquisa constatou a carência de ferramentas com esta finalidade e as poucas existentes não orientam o aluno no processo de revisão, o que pode vir a ser um problema para alunos iniciantes. A partir disso foram iniciados estudos para apresentar uma solução mais abrangente para auxiliar na revisão de programas. Esses estudos levaram as seguintes questões: (1) O que se espera de uma boa revisão? (2) O que deve ser revisado? e (3) Como um aluno iniciante em programação pode fazer uma revisão satisfatória em um programa feito por outra pessoa? Talvez somente mostrar o programa para o aluno e esperar que ele aponte possíveis melhoras pode fazer com que a revisão não tenha a qualidade esperada e que o objetivo de aprendizagem não seja alcançado.

A proposta deste trabalho é conceitualizar e formalizar o processo de revisão através de uma ontologia com o objetivo de apresentar respostas para as questões 1 e 2 e contribuir para a solução da questão 3 visto que a ontologia permitirá um entendimento comum sobre o processo de revisão.

Neste contexto, conceitualizar significa fazer uma cuidadosa análise técnica e identificar exatamente o que deve ser revisado em um programa Java, identificar cada conceito e seus relacionamentos. Formalizar significa organizar os conceitos identificados em uma linguagem formal de maneira que possam ser entendidos por pessoas ou agentes de software.

4.1. Metodologia Utilizada no Desenvolvimento da OntoRevPro

Na literatura encontram-se diversas metodologias para a criação de ontologias. O desenvolvimento da OntoRevPro está sendo feito seguindo o método apresentado por Noy e McGuinness (2001). Esse método foi adotado por ser objetivo e de fácil entendimento.

4.1.1. Determinando o domínio e escopo da ontologia

Para se definir o escopo de uma ontologia as seguintes questões devem ser respondidas: (1) Qual é o domínio que a ontologia irá cobrir? (2) Para que a ontologia será utilizada? (3) Para quais tipos de questões a informação na ontologia deverá fornecer respostas? e (4) Quem irá usar e manter a ontologia?

Para ajudar a determinar o escopo da ontologia foi criada uma lista de questões que uma base de conhecimento baseada na ontologia deve está apta a responder. Essas são chamadas questões de competências e servem como base para o teste da ontologia posteriormente. A seguir um pequeno exemplo de questões de competência levantadas para a OntoRevPro:
Como as variáveis devem ser inicializadas?
Como deve ser o uso do comando import?
O que devem conter os comentários de um programa?
Que características deve ter um comentário?
Como devem ser tratadas as exceções?
Qual o uso correto para os parênteses?
O que é considerado um bom *layout* de programa?
Quais os tipos de dados corretos para definições de variáveis no contexto do programa?
Qual o padrão adequado para nomes?
Qual o uso adequado para espaços em branco dentro do código do programa?
Qual o tamanho apropriado para o código do programa?
Qual deve ser a ordem de declarações dentro de uma classe?

O domínio da OntoRevPro é a revisão de programas Java. O objetivo é que a ontologia seja mais do que um vocabulário controlado e que seja usada para que o aluno entenda o que deve ser revisado em um programa Java.

4.1.2. Reuso de Ontologias Existentes

Uma das vantagens apresentadas pelo uso de ontologias é o reuso das mesmas. Deve-se considerar se alguém já desenvolveu uma ontologia com o mesmo propósito ou com propósito similar. Ao se reusar ontologias obtém-se ganho no processo de desenvolvimento, pois desenvolver uma ontologia é um trabalho que exige muito esforço. O reuso também permite interagir com ferramentas que usam outras ontologias e utilizar ontologias que já foram validadas através do seu uso em outras aplicações.

Foram encontradas na literatura as ontologias JLOO [Lee et al, 2005] e Java_Ontology [JavaOntology] relacionadas à linguagem Java. JLOO é uma ontologia cujo domínio é definir unidades de conhecimento atômico para cursos introdutórios de programação Java, e Java_Ontology define um ambiente de programação e conceitos básicos e avançados de programação Java e também estratégias de programação que mostram outros paradigmas além da orientação a objetos. Optou-se pela não reutilização

dessas ontologias pelo fato das mesmas focarem domínios diferentes do proposto pela OntoRevPro.

4.1.3. Enumeração de Termos Importantes na Ontologia

É útil escrever uma lista de todos os termos sobre os quais gostaríamos de fazer declarações ou explicar ao usuário. Inicialmente é importante obter uma lista abrangente destes termos sem se preocupar com a sobreposição entre os conceitos que eles representam, relações entre os termos, ou qualquer propriedade que os conceitos possam ter ou se os conceitos são classes ou propriedades. Os seguintes são exemplos de termos enumerados no contexto deste trabalho: classe, método, construtor, objeto, variável privada, comentário, recursão, herança, reutilização, coerção, exceção, constantes, modificador, herança..

4.1.4. Definição das Classes e da Hierarquia de Classes

A definição da hierarquia de classes pode ser feita a partir das seguintes abordagens: de cima para baixo (*top-down*), de baixo para cima (*bottom-up*) ou uma combinação das duas anteriores. Na abordagem top-down o processo de desenvolvimento começa com a definição dos conceitos mais gerais no domínio e a subseqüente especialização desses conceitos. Na bottom-up o processo é inverso, inicia-se com as classes mais específicas com o subseqüente agrupamento dessas classes em conceitos mais gerais. Na abordagem que combina os dois processos anteriores, os termos mais evidentes são definidos primeiro e depois vão sendo feitas as generalizações e especializações apropriadas. Segundo Noy e McGuinness (2001) não existe um método melhor ou pior entre os três citados, a abordagem depende fortemente da visão pessoal de quem vai descrever o domínio. Na OntoRevPro a abordagem utilizada é a top-down.

Para definir as classes e relacionamentos na OntoRevPro, foi feita uma pesquisa abrangente sobre elementos de estilo de programação, convenções para programação Java, práticas de programação, livros sobre a linguagem Java , guias para programação Java, dicas para manutenção de código Java e *The Java Tutorial* da Sun (2006).

4.2. Ferramentas Utilizadas para o Desenvolvimento da OntoRevPro

A construção de ontologias é uma tarefa dispendiosa onde se podem obter ganhos significativos quando se utilizam ferramentas adequadas para esse fim.

Para a construção da OntoRevPro foi escolhido o editor Protégé [Protégé, 2006a]. A sua escolha se deveu ao fato de ser um editor interativo, de código aberto, que oferece uma interface gráfica para edição de ontologias e também pelo fato de o mesmo ter sido utilizado com sucesso para o desenvolvimento de ontologias em diversos trabalhos relatados na literatura, conforme pode ser verificado em [Protégé, 2006b]. Com a utilização deste editor, o projetista da ontologia pode concentrar-se diretamente nos conceitos e relacionamentos do domínio sem se preocupar com os detalhes da linguagem que será usada para representar a ontologia, pois o Protégé permite escolher entre diversas linguagens, incluindo XML, RDF ou OWL.

A linguagem OWL - Web Ontology Language foi escolhida para a OntoRevPro pelo fato da mesma ser recomendada pelo W3C – World Wide Web Consortium e ser, atualmente, uma linguagem mais completa que XML, RDF e RDF Schema para o desenvolvimento de ontologias [W3C].

4.3. Estrutura da OntoRevPro

As classes são o coração de uma ontologia. Essencialmente uma ontologia é uma hierarquia de classes. Dessa forma, uma classe indica um conjunto de conceitos que podem existir em um domínio e os *slots*, também chamados de propriedades ou atributos, estabelecem os relacionamentos entre as classes [Lewis et al., 2001]. Segundo Noy e McGuinness (2001), uma ontologia é um modelo da realidade do mundo e os conceitos na ontologia devem refletir essa realidade. No caso da OntoRevPro os conceitos representados pelas classes são os itens que devem ser observados durante a revisão de um programa Java.

A taxonomia é tradicioalmente a parte central da maioria das ontologias, sendo que em algumas essa é a única parte. Os relacionamentos taxonômicos são *relações de ordenação parcial* do tipo *is-a* e *part/whole*. A relação *is-a* é a base da taxonomia e é a relação mais comum para modelagen de conceitos [Pattuelli, 2003].

Um pequeno exemplo da hierarquia de classes da OntoRevPro pode ser observado na Figura 1, ou seja, durante uma revisão de programas deve-se verificar se foi feito uso adequado de comentários. Nesse diagrama encontramos a classe Comentário e suas subclasses.

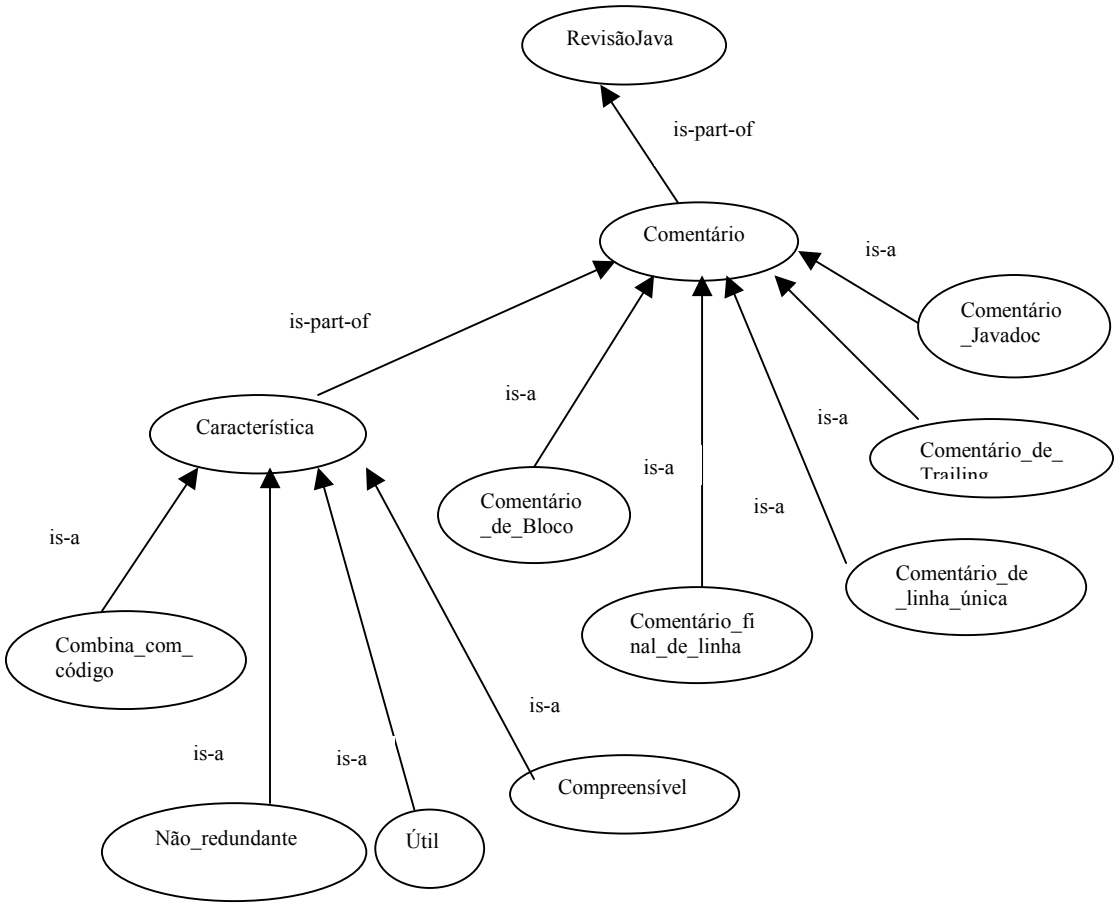

Figura 1 – Diagrama da classe Comentário e suas subclasses.

4.4. Cenários de Uso da OntoRevPro

Uma forma bastante simples para a utilização da OntoRevPro é a disponibilização da ontologia em formato HTML gerado através do próprio Protégé. A própria hierarquia constante na ontologia já mostra o que deve ser revisado, mas o aluno ainda pode fazer uso dos metadados da ontologia para se orientar. Nesse caso a documentação de cada classe e subclasse da ontologia contém uma descrição sobre o item e dicas que orientam o aluno durante a revisão. A Figura 2 mostra parte da ontologia gerada em HTML

Como uma ferramenta de navegação, a ontologia proporcionará ao aluno meios para entender o que deve ser revisado em um programa Java e ainda fornecerá explicações sobre cada termo utilizado na ontologia. Ela pode ser manipulada como uma ferramenta *standalone* oferecendo uma visão do escopo da revisão de programas e também ajudar a aumentar o entendimento do aluno sobre as boas práticas e estilos de programação mais adequados para a programação Java. No entanto, poderá ser necessário uma forma de pesquisa mais rápida e direta a determinado conceito e para tal será desenvolvida uma aplicação para servir como interface no acesso à ontologia. A aplicação será construída em linguagem Java e permitirá uma pesquisa direta a determinado conceito e também aos metadados.

A OntoRevPro poderá ainda ser integrada a um sistema de controle de revisões onde o aluno recebe um programa para revisar e uma lista de itens que devem ser revisados. Caso o aluno tenha dúvida sobre algum desses itens, basta um clique sobre o mesmo para que uma aplicação de pesquisa à ontologia seja executada e permita ao aluno consultar os conceitos da ontologia sobre o item em questão.

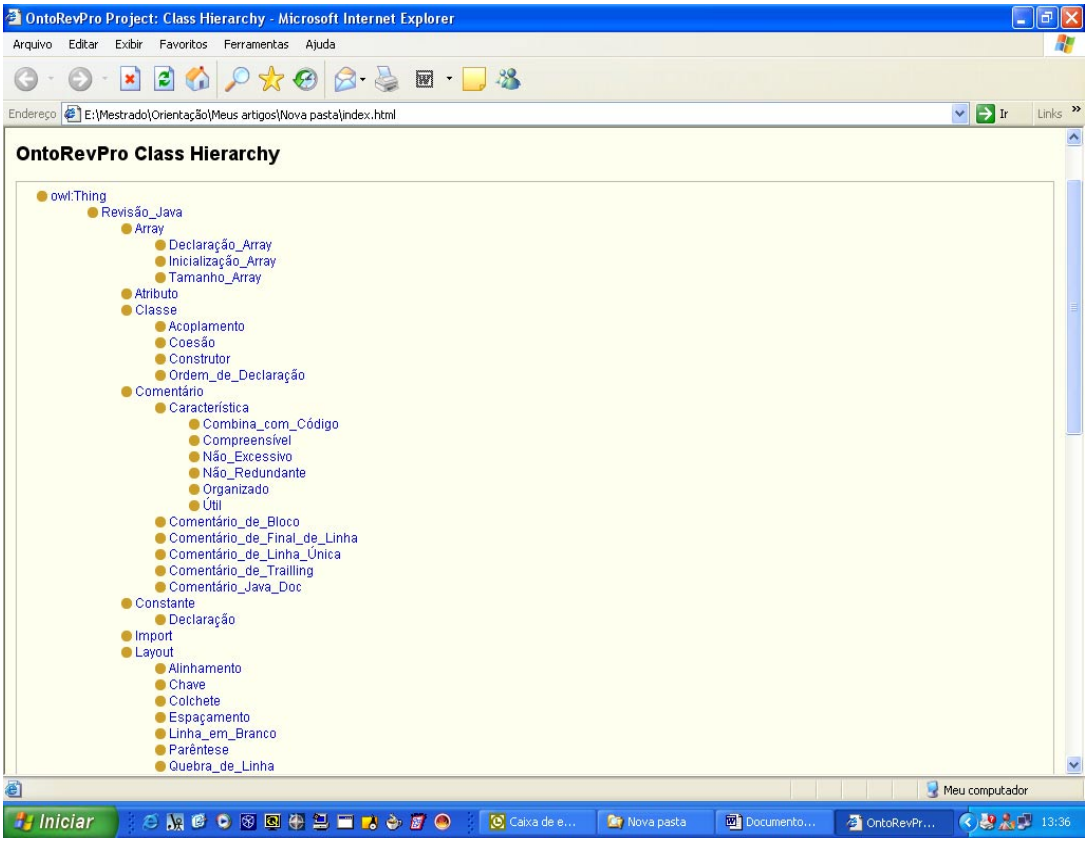

Figura 1 – Hierarquia de classes da OntoRevPro gerada em HTML

4.5. Desenvolvimento e Avaliação da OntoRevPro

O processo de desenvolvimento da OntoRevPro está na fase de documentação da ontologia. Em paralelo com o desenvolvimento está sendo feita uma pré-avaliação tomando como base as questões de competência identificadas no início do desenvolvimento. Esta avaliação inicial visa garantir que todos os conceitos levantados para o domínio de revisão de programas Java, no início do desenvolvimento, estão sendo adequadamente inseridos na ontologia.

Depois de concluída a etapa de documentação, será utilizada a ajuda de especialistas em programação Java para que seja feita uma avaliação completa e possa ser analisado se o propósito esperado na construção da ontologia foi alcançado.

5. Conclusão

A revisão entre pares propicia um aprendizado nos dois sentidos, ou seja, o aluno que revisa aprende ao analisar o programa de um colega e o aluno autor do programa aprende ao ter seu programa revisado por alguém que lhe mostrará pontos que podem ser melhorados.

A conceitualização e formalização da revisão através da OntoRevPro permitirá um entendimento comum sobre o processo de revisão, pois tornará o processo explícito proporcionando ao aluno meios para entender o que deve ser revisado em um programa Java e ainda fornecerá explicações sobre cada termo utilizado na ontologia.

6. Referências

Anderson, N., (1997), Use of Peer Ratings in Evaluating Computer Program Quality, Proceedings of the Fifteenth Annual SIFCPR Conference, ago. 1977.

Chamillard, A T.; Braun, K. A. (2000), Evaluating Programming Ability in an Introductory Computer Science Course, SIGCSE, Austin, TX, USA, 2000.

Faria, J. S. J. e Adán Coello, J. M. (2004), Detectando diferenças significativas entre programas como auxílio ao aprendizado colaborativo de programação. In: XII WEI - Workshop de Educação em Computação, Salvador. V. 1. p. 973-983, 2004.

Ghezzi, C. e Jazayeri (1998), M. Programming Language Concepts, Ed. John Wiley & Sons, Inc, 1998.

Gruber, T. R. (1996), What is an Ontology? Knowledge Systems Laboratory, Stanford University. Disponível em: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html. Acesso em: 28 abr. 2006.

Guarino N. (1998), Formal Ontology and Information Systems, Proceedings of FOIS'98, Trento, Itália, 6-8 jun. 1998.

JavaOntology http://hoersaal.kbs.uni-hannover.de/rdf/Java_ontology.rdf

Lee, M. C.; Ye, D. Y.; Wang, T. I. (2005), Java Learning Object Ontology, Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies, 2005.

Lewis, W. D.; Farrar, S.; Langendoen, D. T. (2001), Building a Knowledge Base of Morphosyntactic Terminology, Proceedings of the IRCS Workshop on Linguistic Databases, University of Pennsylvania, p. 150–156, 2001. Disponível em:

http://www.ldc.upenn.edu/annotation/database/papers/Langendoenetal/24.2.langendoen.pdf. Acesso 12 jan. 2006.

Miyake, N. (1986), Constructive interaction and the interative process of understanding, Cognitive Science, Institute for Cognitive Science University of California at San Diego, California, USA, vol. 10, n. 2, p. 151-177, 1986.

Nosek, J. T. (1998), The Case for Collaborative Programming, Communications of the ACM, Vol. 41, N. 3, mar. 1998.

Noy, N. F. e McGuinness, D. L. (2001), Ontology Development 101: A Guide to Creating Your First Ontology, Knowledge Systems Laboratory, Stanford University, mar. 2001.

Pattuelli, M. C. (2003), The GovStat Ontology: Technical Report. Disponível em: http://ils.unc.edu/govstat/papers/govstatontology.doc. Acesso em 2005.

Pernas, A. A. M. e Dantas, M. A. R. (2004), Ontologias Aplicadas à Descrição de Recursos em Ambientes Grid, INFOCOMP Journal of Computer Science, Universidade Federal de Lavras, Brasil, nov. 2004.

Potégé (2006a) Protégé Ontology Editor and Knowledge-base Framework http://protege.stanford.edu/ Acesso em: 10 mai. 2006.

Potégé (2006b) http://protege.cim3.net/cgi-bin/wiki.pl?ProjectsThatUseProtege Acesso em: 27 mai. 2006.

Sun (2006) http://java.sun.com/docs/books/tutorial/

Tobar, C. M.; Rosa, J. L. G.; Adán Coello, J. M.; Pannain (2001), R. Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação, Anais do Simpósio Brasileiro de Informática na Educação, SBIE - XII, Vitória, 21-23 nov. 2001.

Wilson, J. D.; Hoskin, N.; Nosek, J. T. (1993), The Benefits of Collaboration for Student Programmers, ACM, 24th CSE, fev. 1993.

W3C (2005) Technical Reports and Publications. Disponível em: http://www.w3.org/TR/owl-features/. Acesso em : 23 nov. 2005.

Zeller, A. (2000), Making Students Read and Review Code, ACM, ITiCE, 2000.