# Uma Proposta de Cenário para Ensino de Algoritmos e Programação com Contribuições de Cooperação, Colaboração e Coordenação

**Rodrigo P. Santos[1], Adriana S. Vivacqua[1], Jano M. Souza[1], Heitor A. X. Costa[2]**

[1]COPPE/UFRJ, Universidade Federal do Rio de Janeiro, Brasil
Caixa Postal 68511 – CEP 21945-970 Rio de Janeiro/RJ, Brasil

[2]Grupo PqES/DCC/UFLA, Universidade Federal de Lavras, Brasil
Caixa Postal 3037 – CEP 37200-000 Lavras/MG, Brasil

{rps, avivacqua, jano}@cos.ufrj.br, heitor@ufla.br

***Abstract.*** *Despite the accomplishment of several researches in the area, algorithms and programming teaching still constitutes a challenge for the Computer Science. Starting from a previous analysis of problems detected in this process and possible contributions of cooperation and collaboration, important requirements to compose a more productive learning context were identified. In this sense, the paper objective is to define a proposal of a scenery for algorithms and programming learning, incorporating technologies that include cooperation, collaboration and coordination.*

***Resumo.*** *Apesar da realização de várias pesquisas na área, o ensino de algoritmos e programação ainda se constitui em um desafio para a Computação. A partir de uma análise anterior de problemas detectados nesse processo e possíveis contribuições de cooperação e colaboração, identificou-se requisitos importantes para compor um contexto de aprendizagem mais produtivo. Nesse sentido, este trabalho tem o objetivo de definir uma proposta de cenário para aprendizagem de algoritmos e programação, incorporando tecnologias que englobem cooperação, colaboração e coordenação.*

## 1. Introdução

A fim de produzir melhores resultados em processos de aprendizagem na área de Computação, a atualização de metodologias de ensino deve ser constante. Em se tratando de algoritmos e programação, o raciocínio lógico e o estudo de estruturas de dados e algoritmos em grafos representam pontos críticos devido às dificuldades de entendimento e de representação de suas abstrações. Isso pode conduzir a altas taxas de reprovação e à desistência de cursos na área, devido à diminuição da auto-estima e à geração de apatia [Haden e Mann, 2003]. Por outro lado, estratégias de cooperação e colaboração nem sempre são exploradas o suficiente por questões de tempo e espaço ou dedicação e criatividade no desenvolvimento das aulas [Suthers, 1998], possibilitando a continuidade no uso de métodos tradicionais de ensino, afetando o aprendizado, além de permitir maior competição, isolamento e heterogeneidade nas turmas.

Apesar de não serem tão recentes, esses problemas fazem parte dos grandes desafios de educação em Computação para os próximos dez anos, conforme apresentado

em [McGettrick *et al.*, 2004]. Este documento destaca que estudantes percebem os cursos na área de Computação como "dominados" pela programação e aponta o desafio da *inovação* em reconhecer e acomodar competências e habilidades de estudantes, devido à amplitude de pesquisa e de aplicação da área. Existe um desafio que por si só contempla a questão da programação: *entender o processo de programação e a prática do programador para propiciar a transferência efetiva de conhecimentos e habilidades educacionais*, sugerindo a existência de um "quociente de programação", análogo ao quociente de inteligência. Como alternativa ao ensino presencial, aponta-se *e-learning* como outro desafio relativo ao processo de ensino e aprendizagem, instigando a necessidade de considerar questões de colaboração e cooperação na construção do conhecimento.

Essas considerações impactam os grandes desafios da pesquisa em Computação no Brasil, pois se concentram na busca de soluções melhor sedimentadas para o processo de formação de recursos humanos para a área. Artigos recentes enfatizam os problemas da falta de qualidade em software e consideram suas implicações sobre a indústria brasileira de desenvolvimento para exportação [SBC, 2006]. Para manter um bom mercado interno, deve existir a preocupação com o desenvolvimento de software de forma distribuída, o que ressalta a necessidade de estímulo à cooperação e à colaboração na formação dos discentes, ao longo das disciplinas de algoritmos e programação. Isso atinge diretamente o desafio *desenvolvimento tecnológico de qualidade* [SBC, 2006].

Diante dessa problemática e de suas conseqüências sobre os futuros profissionais da área, novas metodologias e tecnologias precisam ser incorporadas ao processo de ensino de algoritmos e programação, visando facilitar a aprendizagem, o tratamento personalizado dos discentes e o desenvolvimento de suas habilidades individuais. Um cenário para isso precisa congregar requisitos e estratégias que tentem sanar problemas existentes e evitem o surgimento de novos. Em um trabalho anterior [Santos *et al.*, 2008b], uma análise de problemas detectados nesse processo e de possíveis contribuições de cooperação e colaboração foi realizada e requisitos importantes foram identificados para compor um contexto de aprendizagem mais produtivo. Baseado nisso, o objetivo deste trabalho é definir uma proposta de cenário para a aprendizagem de algoritmos e programação aos alunos ingressantes nos cursos da área de Computação, incorporando tecnologias que englobem cooperação, colaboração e coordenação.

O trabalho está organizado da seguinte forma: a seção 2 discorre sobre a fundamentação teórica; a seção 3 apresenta a proposta de cenário para o processo de ensino e aprendizagem de algoritmos e programação; e a seção 4 apresenta as conclusões.

## 2. Fundamentação Teórica

Visando identificar questões e desafios no processo de ensino e aprendizagem de algoritmos e programação, um trabalho inicial consistiu em analisar problemas e soluções desse processo [Santos *et al.*, 2008b], com o objetivo de reunir requisitos para a definição de um cenário relativo à implantação de um processo apoiado por computador. Dentre os principais resultados encontrados, verificou-se que o ensino das abstrações de programação e a falta de compreensão do raciocínio lógico representam as principais razões do alto índice de reprovação nas disciplinas de algoritmos e da desistência de cursos de graduação da área [Baeza-Yates, 2000] [Rosso e Daniele, 2000].

Foi notado que uma solução adotada consiste na utilização de sistemas inteligentes ou ambientes auxiliados por computador [Pimentel *et al.*, 2003]. Ferramentas computacionais para implementação de animações de algoritmos e de estruturas de dados, aliadas a sistemas de gerenciamento de aprendizagem (*Learning Management System* – LMS), possibilitam um ensino mais colaborativo [Vizcaino *et al.*, 2000]. O uso de LMS auxilia a construção do conhecimento pelos aprendizes, passo a passo e coletivamente. Entretanto, nem sempre isso é alcançado, devido à falta de preparação dos docentes e à competitividade entre os discentes, prejudicando o crescimento intelectual.

Vale ressaltar que o aprendizado baseado na interação entre aprendizes consiste na defesa de um sujeito que construa o conhecimento colaborativamente [Neves e Coello, 2006]. O advento das redes de computadores e dos recursos multimídia permitiu a criação da Aprendizagem Colaborativa Apoiada por Computador (*Computer Supported Collaborative Learning* – CSCL) [Colis, 1993]. Ambientes de aprendizagem que atuam sobre a colaboração se mostram benéficos nos aspectos cognitivos e sociais, pois o foco não está na interação entre professor e aluno, mas em como os alunos podem interagir entre si e ensinar uns aos outros [Chamillard e Braun, 2000] [Nosek, 1998].

De acordo com Vygotsky (1978), em sua teoria sócio-cultural, o sujeito não é apenas ativo, mas interativo, porque forma conhecimentos e se constitui a partir de relações "intra" e "interambientais". Os sujeitos internalizam papéis e funções sociais por meio das trocas realizadas com outros sujeitos e consigo próprios, permitindo a formação da própria consciência. Vygotsky dá grande importância à situação social, ao meio e ao organismo ativo, destacando o papel do contexto histórico e cultural nos processos de desenvolvimento e aprendizagem, chamado sócio-interacionista.

Existem alguns níveis de desenvolvimento identificados por Vygotsky (Figura 1): (i) zona de desenvolvimento atual (ZDA): o aprendiz é capaz de realizar sozinho determinadas tarefas; (ii) zona de desenvolvimento proximal (ZDP): o aprendiz realiza determinadas tarefas com a ajuda de outros personagens mais experientes ou em colaboração com colegas mais capazes; e (iii) nível de desenvolvimento potencial (NDP): o aprendiz não consegue desempenhar as tarefas, mesmo com ajuda de outros aprendizes (as tarefas estão além de seu nível de desenvolvimento cognitivo).

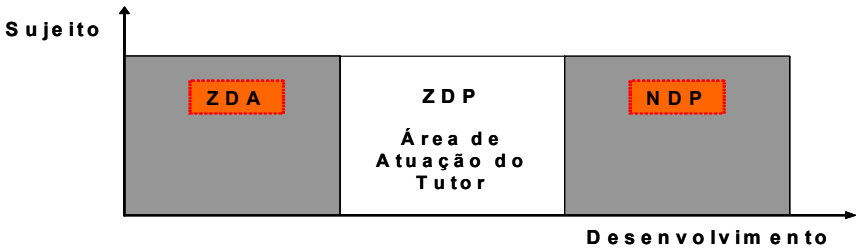


**Figura 1 – Níveis de desenvolvimento**

No cenário proposto, pretende-se concentrar o professor em incitar o desenvolvimento cognitivo dos alunos e interferir na ZDP. As atividades não devem estar na ZDA, pois eles conseguem executá-las, não proporcionando avanços significativos, nem devem estar além da ZDP (i.e., NDP), pois eles não conseguiriam resolvê-las, mesmo com ajuda de um mediador, gerando desestímulo. Portanto, as atividades devem estar além do que o aluno consegue resolver sozinho, mas dentro de sua capacidade e com mediação do professor. Assim, verifica-se que a aprendizagem interativa permite maior agili-

dade no desenvolvimento do aluno. A estratégia do cenário proposto pode ser traçada a partir do cenário de educação tradicional, com a inserção, paulatinamente, de novas técnicas para o ensino e a aprendizagem de algoritmos e programação.

## 3. Uma Proposta de Cenário para Ensino de Algoritmos e Programação

A partir de uma análise de problemas do processo de ensino e aprendizagem de algoritmos e programação, requisitos importantes foram identificados para formular uma proposta de cenário para esse processo envolvendo fundamentos de programação, estruturas de dados e algoritmos em grafos. A proposta (i) distingue três tipos de personagens presentes no processo; (ii) fundamenta-se na zona de desenvolvimento proximal de Vygotsky [Vygotsky, 1978]; e (iii) caracteriza-se por distribuir conteúdos em três disciplinas do ciclo básico da área de Computação (conforme currículo de referência sugerido pela SBC [SBC, 1999]), que englobam ferramentas computacionais para a visualização de abstrações de algoritmos e uma tecnologia de suporte a CSCL (i.e., LMS), além de considerar questões relativas a ensino não presencial, sistema de avaliação e distribuição de carga horária teórica e prática.

### 3.1. Personagens Identificados no Cenário Proposto

Três tipos de personagens foram identificados no cenário proposto (Tabela 1). Isso constitui um componente fundamental, uma vez que os três tipos de personagens se distinguem entre si por seus papéis, envolvendo as características de CSCL coordenação (tutor), colaboração (monitor) e cooperação (aprendiz), integradas por comunicação.

**Tabela 1 – Personagens do cenário proposto**

| PAPEL | DESCRIÇÃO |
|---|---|
| **Tutor** *professor* | Possui a função de coordenar e mediar o processo de ensino e aprendizagem. Deve agir com criatividade e didática no exercício de seu papel, elaborando uma boa seqüência de conteúdo e alocando recursos para a disciplina relacionada (monitores, livros, textos, exercícios e softwares). Não deve ser visto como detentor do conhecimento, mas como facilitador da aprendizagem. |
| **Monitores** *alunos experientes* | Atuam como colaboradores do processo de ensino e aprendizagem, complementando o tutor. Precisam estar presentes e agir com seriedade, pois acompanham diretamente a evolução dos alunos e da turma. Devem ajudar a prover um ambiente de pesquisa ao motivar discussões (evitando fornecer respostas diretas a questionamentos) e contribuir com materiais que considerem importantes. |
| **Aprendizes** *alunos da disciplina* | Compõem um conjunto de discentes que deve utilizar a cooperação para atingir seu objetivo (i.e., assimilação de conteúdo), uma vez que apresenta certa imaturidade, grande curiosidade e ansiedade para adquirir informações e entender seu futuro papel na sociedade. Os aprendizes devem assumir uma postura ativa na construção do conhecimento e podem atuar como colaboradores no processo de ensino e aprendizagem, assumindo papel estrutural na composição da disciplina. Sua bagagem cultural precisa ser vista como variável de interferência no processo. |

Considerando as conexões entre tutores e aprendizes no cenário proposto, o tutor deve permitir que os aprendizes participem de novas interações entre si e que tenham liberdade de escolha para encontrar estilos próprios de resolver problemas. Isso pode ser atingido mediante um processo contínuo de colaboração, de desenvolvimento de senso crítico e de criatividade, pois os aprendizes não são seres isolados e podem se beneficiar de seu círculo social para adquirir conhecimento, ou seja, a ênfase deve estar mais na interação com o meio do que apenas na ação [Piaget, 1932]. Além disso, o tutor deve mudar a forma de encarar o erro, buscando a aprendizagem significativa, na qual a avaliação é mediadora do processo de ensino e aprendizagem; isso significa encarar o erro como algo valioso, que permita estabelecer trocas entre os personagens, por meio de

discussões envolvendo diferentes pontos de vista, com uma avaliação contínua, cumulativa e sistemática.

### 3.2. Estrutura e Caracterização dos Requisitos do Cenário Proposto

A atual conjuntura da área de Computação apresenta as disciplinas que abordam fundamentos de algoritmos e programação normalmente inseridas nos três ou quatro primeiros semestres dos cursos relacionados [ACM, 2001] [ACM, 2004] [CEECInf, 1999] [SBC, 1999]. A partir desse fato e da realidade apresentada na seção 2, o cenário proposto visa integrar requisitos importantes (boas práticas de metodologia de ensino e CSCL) em busca de um cenário educacional mais produtivo. O primeiro passo consiste em analisar a estrutura das disciplinas relacionadas. Devido à carga de conceitos e abstrações a serem assimilados, é interessante existir três níveis de conteúdo distintos.

O *nível I* consiste em uma disciplina introdutória, com apresentação de conceitos básicos de algoritmos e de uma linguagem natural para sua codificação. Os conteúdos envolvidos são: (i) comandos de atribuição/entrada/saída; (ii) expressões aritméticas/lógicas/literais; (iii) estruturas de seleção/repetição; (iv) variáveis compostas homogêneas (vetores e matrizes) e heterogêneas (registros e arquivos); e (v) modularização (funções e procedimentos). Essa disciplina deve apresentar caráter tradicional, com acompanhamento direto de tutores e monitores, natureza presencial e integração entre aulas teóricas e práticas. O paradigma procedural é importante nesse nível para facilitar o contato dos aprendizes com a construção de algoritmos, evitando desviá-los de seu foco em entender o uso de algoritmos para a resolução de problemas e seus comandos básicos. No que diz respeito às avaliações: (i) provas periódicas para aplicação de conceitos, lógica e resolução de problemas; e (ii) programação em grupos de trabalho dinâmicos, mediados pelo tutor, a fim de estimular conflitos de idéias e maior rotatividade de relações. Para incentivar a iniciação à pesquisa independente e a colaboração ao longo da aprendizagem, o tutor deve utilizar *Webquests* [Dodge, 1995], uma metodologia simples para dimensionar usos educacionais da *Web*, com base em aprendizagem cooperativa e processos investigativos na construção do saber, construídas e mediadas pelo tutor. Isso contribui para romper a competição e o individualismo, preparando aprendizes para as estratégias utilizadas nos próximos níveis.

O *nível II* define uma disciplina mais robusta de algoritmos, estruturas de dados e programação, envolvendo: (i) recursividade; (ii) tipos abstratos de dados; (iii) algoritmos de ordenação e busca em vetor; (iv) uso de alocação estática e dinâmica de memória (listas, filas e pilhas); (v) conceitos e propriedades de árvores, árvores binárias e balanceamento; e (vi) tabela de dispersão. O *nível III* envolve tópicos mais avançados, delineando a composição dos fundamentos de programação: (i) manipulação de memória e tratamento de arquivos; (ii) estruturas de dados avançadas (árvore B e árvore B*); e (iii) conceitos de grafos e seus algoritmos. Nesses níveis, a orientação a objetos seria adotada para que os aprendizes se familiarizem com esse paradigma.

Considerando certa maturidade dos aprendizes, adquirida no nível I, e a complexidade das abstrações dos níveis II e III, o cenário proposto visa intensificar propriedades de cooperação e de colaboração na construção do conhecimento nos dois últimos níveis. Dessa forma, a função da metodologia *Webquest* consiste em ajustar o contexto

social e intelectual dos aprendizes que se encontram no nível I, preparando-os para os demais níveis. Consideram-se três questões importantes nos níveis II e III:

A **primeira questão** consiste na *natureza das disciplinas relacionadas*, que deve incorporar características de ensino à distância à sua modalidade presencial. Isso não corresponde à eliminação das aulas presenciais, uma vez que esse processo, de forma súbita, pode prejudicar a construção do conhecimento pelos aprendizes, devido às dificuldades de adaptação e à sua postura inexperiente. Com esta finalidade, a *Internet* e as tecnologias de comunicação serão inseridas no cenário proposto por meio de um LMS, que permitirá maior interação em processos de colaboração e cooperação entre os aprendizes e sua coordenação pelo tutor, em conformidade com a teoria sócio-construtivista de Piaget [Piaget, 1932]. Considerando a elaboração das disciplinas desses dois níveis do cenário proposto, os personagens definidos terão participação ativa. O tutor deverá adotar um LMS que julga suficiente para atender ao cenário proposto e iniciar a construção da disciplina. Paralelamente, recursos virtuais e impressos de qualidade devem estar disponíveis: um livro-texto, artigos científicos, bibliografia complementar e *webliografia* (bibliografia virtual via URL). Além disso, devido ao fato das abstrações apresentarem difícil visualização, ambientes de animação gráfica passo a passo, previamente definidos pelo tutor, devem ser disponibilizados no LMS a fim de facilitar a compreensão dos conteúdos envolvidos. Considera-se essa estratégia importante, uma vez que a assimilação de informações via animação e interação com computadores é maior em relação a processos tradicionais (via quadro-negro) [Santos e Costa, 2006]. Para a disciplina do nível III, ambientes gráficos que permitam a inserção de novos algoritmos e animações podem estimular os aprendizes a contribuírem para o processo.

A **segunda questão** refere-se às *características dos personagens envolvidos*. Os monitores devem não apenas acompanhar os aprendizes presencialmente, mas se envolver na metodologia de ensino e aprendizagem suportada pelas tecnologias incorporadas (como um tutor), e verificar o ritmo e a evolução dos aprendizes. Estes, por sua vez, começarão a lidar com a independência e o cooperativismo durante a assimilação do conhecimento, colaborando com os demais ao prover um crescimento contínuo e balanceado da turma e auxiliando na construção da disciplina no LMS. O tutor deve estimular a inserção de materiais pesquisados por aprendizes e monitores no LMS, incentivando a colaboração. Ele deve permitir que os aprendizes tenham liberdade para responder a dúvidas de outros aprendizes no LMS, restringindo respostas diretas a fim de gerar discussão.

A **terceira questão** corresponde à *forma de avaliação*. No cenário proposto, as tradicionais provas devem ser substituídas por um conjunto de exercícios baseados em processos e desafios que exijam maior raciocínio na construção de soluções robustas, admitindo consulta e cooperação. Quanto aos trabalhos práticos de programação, compõem-se de problemas apresentados em aulas práticas e de continuação extra-classe, em grupos construídos dinamicamente. A correção dos trabalhos sairia do domínio do tutor e dos monitores e incluiria os grupos de aprendizes, os quais verificariam erros de trabalhos de outros grupos, considerando tais erros como tentativas de acerto e não como punição (aprendizagem significativa). Além disso, aulas teóricas não presenciais devem ser realizadas em horário pré-estabelecido, mediante o uso de LMS e de *chats* (áudio/vídeo-conferências), dividindo espaço com aulas presenciais tradicionais e valorizando a participação em fóruns e em grupos de discussão. As disciplinas dos níveis II e III organizam-se conforme a estrutura exibida na Tabela 2. Alguns requisitos, apresenta-

dos na Tabela 3, devem ser atendidos pelo cenário proposto para viabilizar a incorporação da cooperação e da colaboração no processo de ensino e aprendizagem.

**Tabela 2 – Segmentação das disciplinas dos níveis II e III e seus objetivos**

| AULAS TEÓRICAS | | AULAS PRÁTICAS |
|---|---|---|
| *PRESENCIAIS* | *NÃO PRESENCIAIS* | *LABORATÓRIO* |
| 25% da Carga Horária | 25% da Carga Horária | 50% da Carga Horária |
| ▪ **Evitar transição brusca** do ensino presencial tradicional para o ensino à distância;<br>▪ **Acompanhar a percepção e a reação** de cada aprendiz face a face;<br>▪ Realizar **avaliações formais** (provas). | ▪ Iniciar o aluno à **pesquisa independente**;<br>▪ **Estimular a colaboração** entre os aprendizes **na construção do conhecimento** por meio de aulas **mediadas pelo tutor**, via fóruns (ao longo da disciplina) e via grupos de discussão e *chats* (em sincronismo com o tutor);<br>▪ Estimular a cooperação na realização de trabalhos de programação via *Web* (**programação cooperativa**). | ▪ **Aplicar os conceitos aprendidos**, por meio de exercícios, desafios e tarefas;<br>▪ Iniciar trabalhos de programação de forma estruturada (**presença do tutor e dos monitores**), de maneira que certa parcela da tarefa seja cumprida extra-classe e em grupo;<br>▪ Promover a interação aluno-computador, sob **supervisão** do tutor. |

**Tabela 3 – Requisitos importantes para o cenário proposto**

| REQUISITOS | DESCRIÇÃO |
|---|---|
| **Comunicação assíncrona** fóruns | Reduz a inibição dos aprendizes e possibilita maior participação em grupos de discussão (mais gerais, para dúvidas e questionamentos) e em grupos de interesse (específicos), para trabalhos de programação e cooperação na construção do conhecimento. Além disso, o crescimento colaborativo tenta diminuir a discrepância entre aprendizes e eliminar o caráter competitivo, imposto pelo formato atual das aulas e avaliações aplicadas. |
| **Comunicação síncrona** *chats* e áudio/vídeo-conferência | Viabiliza as aulas não presenciais, nas quais o tutor assume papel de mediador e os aprendizes, em grupos, executam pesquisas nos materiais disponibilizados no LMS e utilizam os ambientes de animação gráfica de algoritmos, construindo a aula. Os monitores devem atuar como participantes mais experientes e acompanhar o tutor, estimulando a cooperação e atendendo às dúvidas. |
| **Programação cooperativa não presencial** | Permite o desenvolvimento de trabalhos de programação entre os aprendizes, não reunidos em um mesmo ambiente físico. De forma simples, os *chats* em LMS auxiliam nesta tarefa. Entretanto, a incorporação de um editor para programação em um LMS a fim de suportar o desenvolvimento distribuído das implementações dos aprendizes seria interessante. Questões como a estabilidade do repositório de tarefas e o gerenciamento de arquivos (gerência de configuração) precisam de análise cuidadosa, para que sejam transparentes aos aprendizes. |
| **Interdisciplinaridade** | Após a implantação do cenário proposto, o processo de aprendizagem de algoritmos e programação deve buscar relacionamento com outras disciplinas, a fim de integrar o conhecimento verticalmente (disciplinas de diferentes semestres) e horizontalmente (disciplinas de mesmo semestre). Este é um ponto forte, mas depende diretamente da interação e do comprometimento dos tutores envolvidos no processo e do coordenador de curso. |
| **Avaliação do cenário proposto** | As estratégias utilizadas e a aprendizagem efetiva dos aprendizes devem ser acompanhadas constantemente, por meio de formulários e entrevistas. Deve-se registrar conflitos, falhas e necessidades (tecnológicas e educacionais), visando permitir a elaboração de melhorias para as próximas turmas e prover um mecanismo de retro-alimentação para o cenário proposto. Além disso, um conjunto de métricas pode ser estabelecido a fim de verificar a melhoria no desempenho dos aprendizes mediante o uso da proposta. |

### 3.3. Arquitetura de um Ambiente que apóie o Cenário Proposto

Pelo fato da implantação do cenário proposto envolver algumas alterações na estrutura curricular e/ou na ementa de disciplinas relacionadas, além de lidar diretamente com a formação dos discentes e requerer comprometimento dos docentes envolvidos, optou-se por simulá-lo inicialmente com ferramentas existentes. Estas, mesmo não integradas, permitiriam nortear a construção de um ambiente que suporte o cenário proposto ao capturar novos requisitos ou alterar aqueles existentes dinamicamente.

Algumas escolhas foram delineadas, mediante a divisão do conteúdo de algoritmos e programação do ciclo básico em três disciplinas, e são justificadas pelo fato de atenderem à estrutura e aos requisitos do cenário proposto: (i) dividir a carga horária em teórica e prática, conforme a Tabela 2 (para a disciplina do nível I, dividiu-se 50% da carga horária para teoria e 50% para laboratório); (ii) construir *Webquests* para a disciplina do nível I e utilizar os ambientes de animação gráfica de algoritmos TBC-AED [Santos e Costa, 2005], para a disciplina do nível II, e TBC-GRAFOS [Santos e Costa, 2006], para a disciplina do nível III (devido aos resultados positivos obtidos anteriormente [Santos *et al.*, 2008a]); (iii) implantar o LMS Moodle [Graf e List, 2005] para gerenciar conteúdos, fóruns e grupos de discussão, provendo um canal de comunicação entre tutor, monitores e aprendizes; (iv) adotar a tecnologia *Skype* visando a realização de aulas não presenciais (áudio/vídeo-conferências); e (v) prover parcialmente a programação cooperativa não presencial na disciplina do nível III por mecanismos de gerência de configuração, devido ao uso da linguagem Java e do ambiente Eclipse.

A divisão dos conteúdos em três níveis e as escolhas (i) e (ii) e parcialmente a escolha (iii) vêm sendo utilizadas com sucesso no curso de Ciência da Computação da Universidade Federal de Lavras. Essas escolhas foram implantadas durante os dois semestre letivos de 2007 e tiveram como conseqüências: (i) aumento no índice de aprovação dos aprendizes em 30%; (ii) agilidade na construção e na realização das aulas; (iii) avanços na forma de visualização das abstrações envolvidas; e (iv) melhorias no canal de comunicação entre tutores, monitores e aprendizes. A partir disso, pôde-se projetar a arquitetura de um ambiente para apoiar o cenário proposto (Figura 2), na qual cada caixa representa um módulo ou serviço a ser suportado por esse ambiente.

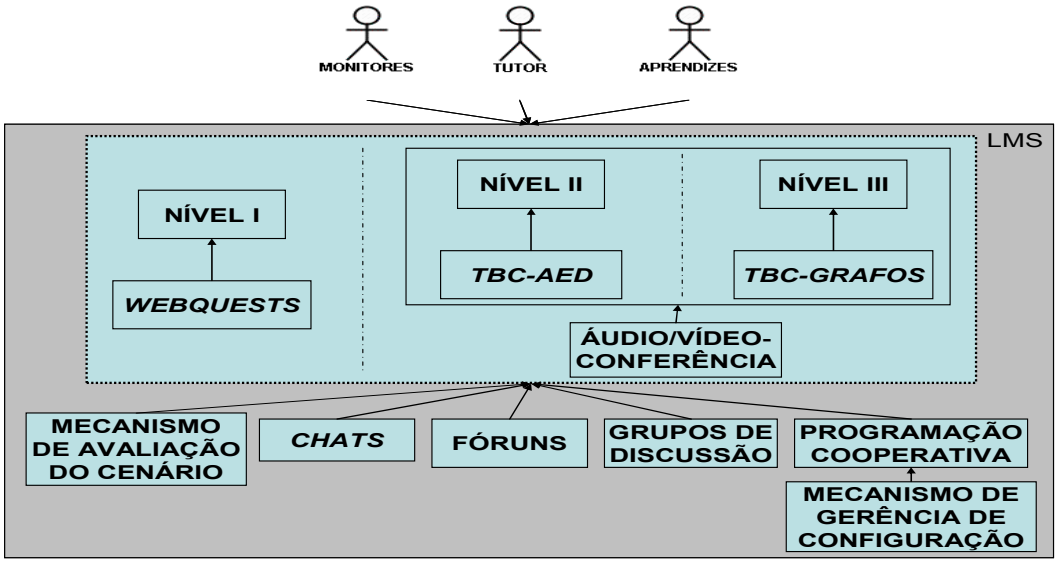


**Figura 2 – Arquitetura de um ambiente para apoiar o cenário proposto**

A arquitetura respeita aos requisitos identificados em [Santos *et al.*, 2008b] e está estruturada de acordo com o cenário proposto no presente trabalho. Deve-se ressaltar a importância de um mecanismo de avaliação do cenário, promovendo sua retro-alimentação e conseqüente evolução, e a definição de métricas que permitam a verificação da melhoria de desempenho dos aprendizes com o estabelecimento do cenário proposto e, mais à frente, com o uso do ambiente. Além disso, o ambiente deve evitar transparecer quaisquer problemas ou detalhes que afetem a usabilidade e o intuito de apoiar um

processo de ensino e aprendizagem, sobretudo com relação a mecanismos de gerência de configuração para o módulo de programação cooperativa.

## 4. Conclusão

Não há dúvida de que a programação constitui o maior desafio no ensino de computação, freqüentemente permeando a experiência do estudante na educação superior da área [McGettrick *et al.*, 2004]. O processo de ensino de algoritmos e programação necessita da incorporação de métodos de aprendizagem diferentes daqueles tradicionalmente empregados. Assim, este trabalho se embasou em uma análise de problemas desse processo e na identificação de requisitos para sua melhoria [Santos *et al.*, 2008b] e reúne estratégias para definir uma proposta de cenário mais produtivo para a aprendizagem como sua contribuição à literatura.

Estrutura-se o cenário a partir da identificação de personagens que interagem no processo de ensino e aprendizagem, buscando inserir tecnologias que agreguem cooperação e colaboração. A forma de avaliação da aprendizagem foi analisada, a fim de suportar um contexto presencial que divida espaço com outro não presencial, visando maior desenvolvimento do aprendiz. A importância das aulas práticas foi ressaltada, por constituírem um processo de aplicação de conhecimentos sob a supervisão do tutor e dos monitores. Além disso, grande importância é atribuída a CSCL por permitir uma aprendizagem conjunta, contínua e dividida em etapas definidas. As atividades de programação, em aulas práticas e trabalhos de implementação, são realizadas em grupos, construídos dinamicamente, vistos como equipes produtoras de conflitos de idéias.

Existem alguns desafios, como o apoio à programação cooperativa em ambientes LMS e a aplicação de uma espécie de gerência de configuração em escala reduzida, para as implementações dos aprendizes em equipes. Um repositório confiável e de fácil gerenciamento para os trabalhos deve estar disponível. Outra questão consiste no apoio à comunicação síncrona de qualidade, por meio de diálogos textuais e áudio/videoconferências. Assim, a partir da arquitetura definida na subseção 3.3, será desenvolvido um LMS que incorpore os requisitos identificados e as estratégias definidas. Um processo para a implantação do cenário proposto será elaborado. Os estudos serão realizados nos cursos de Ciência da Computação, Sistemas de Informação e Engenharia da Computação das duas instituições de ensino superior envolvidas na pesquisa. Estudos de caso serão realizados a fim de verificar o efeito das aulas não-presenciais sobre o índice de aprovação ou sobre o nível de colaboração entre os aprendizes. Além disso, a experimentação será delineada por meio de entrevistas e questionários aplicados constantemente sobre os personagens envolvidos para a verificação das estratégias utilizadas.

## Agradecimentos

Os autores agradecem ao CNPq pelo apoio financeiro para a realização deste trabalho.

## Referências

ACM (2001) “Computing Curricula”. Disponível em <http://www.computer.org/education/cc2001/final/index.htm>. Acesso: 10/03/2008.

ACM (2004) “Curricula Recommendations”. Disponível em <http://www.acm.org/education/curricula.html>. Acesso: 10/03/2008.

Baeza-Yates, R. A. (2000) “Teaching Algorithms”, In: SIGACT News 26, v. 4, p. 51-59.

CEECInf (1999) "Comissão de Especialistas de Ensino de Computação e Informática". Diretrizes Curriculares de Cursos de Computação e Informática, SESu-MEC.

Chamillard, A. T.; Braun, K. A. (2000) "Evaluating Programming Ability in a Introductory Computer Science Course", In: Proc. of the 31st SIGCSE, Austin/TX, USA.

Colis, B. (1993) "Cooperative Learning in CSCW: Research Perspectives for Internetworked Educational Enviroments", In: IFIP Working Group 3.3 Working Conference *Lessons from Learning*, Archamps, France.

Dodge, B. (1995) "WebQuests: A Technique for Internet-Based Learning". In: The Distance Educator, San Diego, v. 1, n. 2, p. 10-13.

Graf, S.; List, B. (2005) "An Evaluation of Open Source E-Learning Platforms Stressing Adaptation Issues", In: Proc. of the 5th ICALT, Kaohsiung, Taiwan, p. 163-165.

Haden, P.; Mann, S. (2003) "The Trouble with Teaching Programming", In: Proc. of the 16th Annual NACCQ, Palmerston North, New Zealand.

McGettrick, A.; Boyle, R.; Ibbett, R.; Lloyd, J.; Lovegrove, G.; Mander, K. (2004) "Grand Challenges in Computing – Education". The British Computer Society, 26p.

Neves, M. F.; Coello, J. M. A. (2006) "OntoRevPro: Uma Ontologia sobre Revisão de Programas para o Aprendizado Colaborativo de Programação Java", In: Anais do XVII SBIE, Brasília/DF, p. 569-578

Nosek, J. T. (1998) "The Case for Collaborative Programming", In: Communications of the ACM, v. 41, n. 3.

Piaget, J. (1932) "The Moral Judgement of the Child". Routledge and Kegan Paul, London.

Pimentel, E. P.; França, V. F.; Omar, N. (2003) "A Caminho de um Ambiente de Avaliação e Acompanhamento Contínuo de Aprendizagem em Programação de Computadores", In: Anais do II WEIMIG, Poços de Caldas/MG.

Rosso, A.; Daniele, M. (2000) "Our Method to Teach Algorithmic Development", In: ACM SIGCSE Bulletin, v. 32, n. 2, ACM Press, USA, p. 49-52.

Santos, R. P.; Costa, H. A. X. (2005) "TBC-AED e TBC-AED/WEB: Um Desafio no Ensino de Algoritmos, Estruturas de Dados e Programação", In: Anais do IV WEIMIG, Varginha/MG.

Santos, R. P.; Costa, H. A. X. (2006) "Um Software Gráfico Educacional para o Ensino de Algoritmos em Grafos", In: Proc. of the IADIS/CIAWI'2006, Murcia, Espanha.

Santos, R. P.; Costa, H. A. X.; Resende, A. M. P.; Souza, J. M. (2008a) "O Uso de Ambientes Gráficos para Ensino e Aprendizagem de Estruturas de Dados e de Algoritmos em Grafos", In: Anais do XVI WEI, CSBC'2008, Belém/PA.

Santos, R. P.; Vivacqua, A. S.; Souza, J. M.; Costa, H. A. X. (2008b) "Questões e Desafios no Ensino de Algoritmos e Programação: Identificando Aspectos Importantes no Processo de Aprendizagem apoiado por Computador", In: Proc. of the X INTERTECH, Peruíbe/SP, p. 568-572.

SBC (1999) "Currículo de Referência".

SBC (2006) "Grandes Desafios da Pesquisa em Computação no Brasil – 2006-2016".

Suthers, D. D. (1998) "Computer Aided Education and Training Initiative", In: Technical Report. Learning Research and Development Center, University of Pittsburgh.

Vizcaino, A.; Contreras, J.; Favela, J.; Prieto, M. (2000) "An Adaptive, Collaborative Environment to Develop Good Habits in Programming", In: Proc. of the 5th ITS, Montreal, Canadá, p. 262-271.

Vygotsky, L. S. (1978) "Mind in Society: The Development of Higher Psychological Processes". Cambridge: Harvard University Press, p. 52-91.