SCIENTIA

REVISTA DO CENTRO UNIVERSITÁRIO VILA VELHA

VILA VELHA (ES), v. 4, n. 1/2, JANEIRO/DEZEMBRO DE 2003

REVISTA DO CENTRO UNIVERSITÁRIO VILA VELHA

Revista interdisciplinar semestral

Nota: As opiniões e conceitos emitidos nos artigos publicados nesta revista são de inteira responsabilidade dos seus autores.

Tiragem: 1.000 exemplares

ISSN 1518-2975

Coordenação Executiva:
Angela Maria Monjardim
Danièlle de Oliveira Bresciani

Revisão:
Artelírio Bolsanello
Assessoria em Normalização da Informação/UFES

Capa:
Juan Carlos Piñeiro Cañellas

Impressão:
Artgraf - Gráfica e Editora

SCIENTIA. V. 4, n.1/2, (jan./dez.2003) – Vila Velha (ES): Sociedade Educacional do Espírito Santo, 2003.

ISSN 1518-2975

Semestral

1. Cultura – Periódico. 2. Generalidades – Periódico, Centro Universitário Vila Velha - SEDES/UVV-ES
CDD 002

E-mail: scientia@uvv.br

SUMÁRIO

EDITORIAL

Em um mundo norteado pelas informações e por rápidas transformações tecnológicas que afetam o funcionamento e a própria lógica da sociedade, estimulando o estudo em diferentes áreas do saber, a Revista Scientia, traz os mais variados assuntos, com o propósito principal de permitir uma leitura analítica, capaz de agregar novos conhecimentos e conferir a você, leitor, uma nova percepção e posicionamento nesse contexto.

Os temas abordados, neste volume, contemplam tanto a área tecnológica quanto a área da saúde, trazendo para a discussão, diferentes enfoques que envolvem, diretamente, à questão do controle, através do desenvolvimento de um produto capaz de monitorar o acesso de pessoas a locais específicos, como forma, inclusive, de garantir a segurança, cada vez mais frágil na sociedade atual bem como trata da otimização na utilização de recursos físicos disponíveis, considerada questão essencial, numa era consubstanciada na busca do aumento vertiginoso do acúmulo de capital, proporcionado, em grande medida, pelo contenção de custos e desperdícios.

Associado às variáveis estudadas, cabe destacar que os artigos tratam ainda de uma questão fundamental, que é a saúde, nos seus mais variados aspectos, seja mediante à observância de microorganismos contaminantes presentes em estabelecimentos públicos, notadamente, bares e restaurantes, que podem desencadear distúrbios digestivos e dermatites, seja através da avaliação da composição corporal de triatletas, com vistas a investigar as correlações existentes entre os dados relatados na literatura e o percentual de gordura apresentado.

As temáticas ligadas à segurança, controle, otimização, custos, despedícios, saúde pública e composição corporal formam o leque de opções oferecido neste volume. A Scientia é multidisciplinar, procura tratar de temas emergentes e, acima de tudo, está sempre “preocupada” em garantir, ao seu leitor, qualidade e diversidade. Exatamente, por isso, está passando por um processo de reavaliação, que se traduz, neste momento, na publicação de dois números em um único volume, destinado a você, leitor, como um instrumento do processo de aprendizagem.

Danièlle de Oliveira Bresciani, Ms.

COMPOSIÇÃO CORPORAL DOS TRIATLETAS CAPIXABAS

SCIENTIA

MIGUEL ÂNGELO ALVES DOS SANTOS[1]
GIL PEIXOTO ATAYDE[2]

[1] Mestre em Ciências Fisiológicas pela Universidade Federaldo Espírito Santo. Professor do Centro Universitário Vila Velha. E-mail: miguel@uvv.br.

[2] Aluno de iniciação científica do Núcleo de Biodinâmica das Atividades Corporais do Centro Universitário Vila Velha.

RESUMO

Avalia a composição corporal dos triatletas capixabas que participaram do triathlon do Exército - 2000. Foram medidas as dobras cutâneas do tríceps, subescapular, supra-ilíaca e abdominal, os diâmetros ósseos do rádio e do fêmur, além da massa corporal e estatura, visando à determinação do peso de gordura, ósseo, muscular e residual. Os dados foram tratados na forma de estatística descritiva. Os triatletas apresentaram as seguintes características antropométricas: idade 31,69 anos, massa corporal 70,7kg, 173,8cm de estatura, 44,35mm somatório das dobras cutâneas, 12,57 de % de gordura, 47,75% de peso muscular e 15,68% de peso ósseo. Em relação às médias de idade, peso e estatura, observa-se valor semelhante aos dados relatados na literatura, porém o percentual de gordura apresenta um índice elevado em relação aos padrões médios internacionais e nacionais.

Palavras-chave: Triatletas. Composição Corporal. Triathlon. Fracionamento do peso corporal.

1 INTRODUÇÃO

Em eventos poliesportivos de endurance é comum a execução contínua e seqüenciada de exercícios de diferentes modalidades. Dentro desse ponto de vista, destaca-se o triatlo composto pela natação, ciclismo e corrida. Essa modalidade esportiva sofre uma série de classificações. A primeira classificação é o triatlo olímpico, geralmente disputado em percursos com 1,5km de natação, 40km de ciclismo e 10km de corrida. A segunda classificação é o triatlo de ultra-endurance ou *ironman*, que exige que os competidores nadem 3,9km, pedalem 180,2km e corram 42,2km (O'TOOLE et al., 1987).

Características físicas e fisiológicas de atletas de elite têm sido constantemente pesquisadas. Vários fatores contribuem para uma melhor performance do triatleta, entre esses se destacam o nível de consumo máximo de oxigênio, a fração de utilização do consumo máximo de oxigênio, eficiência do movimento e fatores ligados as variáveis antropométricas: massa corporal, estatura e composição corporal (O'TOOLE; DOUGLAS, 1995; WILMORE; COSTILL, 1974).

Pesquisas anteriores têm mostrado que a constituição antropométrica dos triatletas, isto é, estatura, peso e composição corporal mostrou ser

muita parecida com os atletas de elite do ciclismo (O'TOOLE et al., 1987). O percentual de gordura foi também comparado entre triatletas e atletas de outros esportes, como corrida, natação e ciclismo. De acordo com algumas pesquisas, tanto os nadadores masculinos, quanto os femininos possuíam um percentual de gordura em torno de 1.4 a 8% maior que os triatletas (LAMB, 1984; WILMORE; BROWN; DAVIS, 1977). Além disso, observou-se que os corredores têm um menor percentual de gordura que os triatletas masculino (LAMB, 1984; WILMORE; BROWN; DAVIS, 1977). Já em relação à massa corporal, os triatletas masculinos apresentaram um peso que foi ligeiramente menor que os atletas nadadores e corredores pesquisados (POLLOCK, 1977; WILMORE; BROWN; DAVIS, 1977).

É amplamente aceita por atletas e treinadores a existência de um peso e composição corporal ideal para cada esporte específico. Geralmente esses são baseados na média dos valores de percentual de gordura e massa corporal magra obtidos de várias amostras de atletas de elite (SINNING; WILSON, 1984). Embora, o percentual de gordura dos atletas esteja ligado ao desempenho dentro do esporte, a performance não pode ser predita somente pela observação desse fator. Evidências demonstram uma relação inversa entre o peso de gordura e a performance nas atividades que requerem deslocamento da massa corporal tanto no sentido vertical quanto horizontal (BOILEAU; LOHMAN, 1977; MALINA, 1992; PATE; SLENTZ; KATZ, 1989). Visto que a aceleração é proporcional à força, mas inversamente proporcional à massa, um excesso de gordura para um dado nível de aplicação de força resulta em uma menor mudança na velocidade e direção do movimento (BOILEAU; LOHMAN, 1977; HARMAN; FRYKMAN, 1992).

Excesso de gordura também aumenta o custo metabólico da atividade física que requer movimento corporal (BUSKIRK; TAYLOR, 1957). Assim, nas performances que envolvem movimento do corpo, uma diminuição do percentual de gordura melhoraria a performance tanto mecanicamente, quanto metabolicamente (BOILEAU; LOHMAN, 1977).

Fica evidente, assim, que conhecer as diversas formas e implicações da determinação da composição corporal é, hoje, um importante diferencial na performance de atletas, principalmente de elite. Diante disso, o objetivo desse estudo foi avaliar a composição corporal dos triatletas capixabas que participaram do triathlon do Exército, Vitória – ES.

MATERIAL E MÉTODOS

Para a elaboração deste estudo foram avaliados 13 triatletas do sexo masculino capixabas, que participaram do triatlo do exército do ano de 2000, com 1,5km de natação, 40km de ciclismo e 10km de corrida.

Foram mensuradas as variáveis antropométricas: massa corporal; estatura, dobras cutâneas e medidas dos diâmetros ósseos do rádio e fêmur. Para a mensuração da massa corporal utilizou-se uma balança antropométrica Filizola, da estatura por meio de um estadiômetro da marca Cefise, medidas de dobras cutâneas do tríceps, subescapular, supra-íliaca e abdominal com um plicômetro Cescorf e dos diâmetros ósseos do rádio e fêmur, por meio de um paquímetro modelo Cefise. Os procedimentos foram realizados segundo normas técnicas estabelecidas por Sá (1975).

O protocolo utilizado para a determinação do percentual de gordura foi o estabelecido por Faulkner e, em seguida, realizou-se o fracionamento da massa corporal para determinação do peso de gordura, peso ósseo, peso residual e peso muscular (POLLOCK; WILMORE; FOX III, 1984). Para tanto, foram utilizados os seguintes protocolos (DE ROSE; PIGATTO; DE ROSE, 1984): a) Peso de gordura (PG) = (%G/100) x massa corporal; onde %G= percentual de gordura; b) Peso ósseo (PO) = 3.02 (H^2 x R x F x 400) $^{0.712}$; onde: H = estatura em m; R = diâmetro do rádio em m; F = diâmetro do fêmur em m; c) Peso residual (PR) = massa corporal x 0,24; d) Peso muscular (PM) = PC – (PG+PO+PR); onde PC= massa corporal em kg.

Os dados foram tratados em termos de estatística descritiva e valores percentuais, analisados conforme os objetivos pesquisados.

RESULTADOS

Na Tabela 1 encontram-se os valores médios e desvio-padrão referente à idade, peso corporal, estatura, diâmetros ósseos e das dobras cutâneas medidas. Na Tabela 2 estão os valores da somatória das dobras cutâneas, densidade corporal, massa corporal magra, peso de gordura, peso ósseo e peso muscular.

Tabela 1 – Valores médios e desvio padrão das variáveis antropométricas da amostra

	IDADE	MC	ESTATURA	DCT	DCSUB	DCSUP	DCABD	DR	DF
MÉDIA	31,7	70,7	173,8	7,7	12,1	10,2	14,4	5,6	9,2
DESVIO PADRÃO	12,9	7,9	7,3	1,9	4,3	5,0	6,3	0,3	0,6

MC Massa corporal
DCT Dobra cutânea do tríceps
DCSUB Dobra cutânea subescapular
DCSUP Dobra cutânea supra-ilíaca
DR Diâmetro do rádio
DF Diâmetro do fêmur.

Tabela 2 – Valores médios do SDC, DC, MCM, MG, MO, MM, MR da amostra

	SDC	DC	MCM	MG	MO	MM	MR
MÉDIA	44,4	1,0561	61,8	8,9	11,0	33,8	17,0
DESVIO PADRÃO	16,2	0,01	7,0	2,1	1,3	4,2	1,9

SDC Somatória das dobras cutâneas
DC Densidade corporal
MCM Massa corporal magra
MG Massa gorda
MO Massa óssea
MM Massa muscular
MR Massa residual.

Na Tabela 3 encontram-se os valores percentuais de gordura (%G), do peso ósseo (PO) e do peso muscular (PM) dos triatletas capixabas que participaram do triatlo do exército.

Tabela 3 – Valores médios dos percentuais de gordura, ósseo e muscular

	%G	%PO	%PM
MÉDIA	12,6	15,7	47,8
DESVIO PADRÃO	2,5	1,5	1,5

%G percentual de gordura
%PO percentual ósseo
%PM percentual muscular

Na Figura 1 observam-se os valores médios e os respectivos desvios-padrões, das dobras cutâneas do tríceps (DCT), subescapular (DCSUB), supra-íliaca (DCSUP) e abdominal (DCABD).

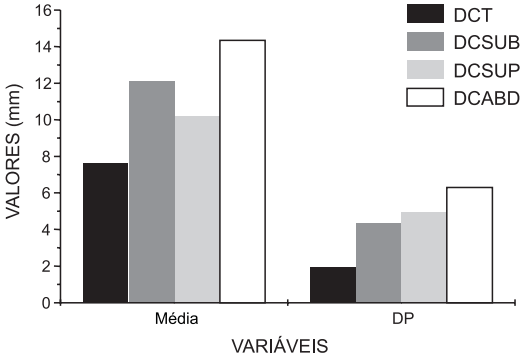


Figura 1 - Valores médios e dos desvios padrões das dobras cutâneas dos triatletas capixabas

DISCUSSÃO E CONCLUSÃO

Analisando os dados da Tabela 1, observa-se que o valor médio da idade é similar aos encontrados na literatura internacional, porém, já em relação aos valores da massa corporal e da estatura dos triatletas pesquisados, existe uma tendência a valores menores do que a média internacional (O'TOOLE et al., 1987). Contudo, quando se compara a idade média dos triatletas com competidores de modalidades isoladas: corrida, natação e ciclismo, observa-se uma tendência a valor mais alto dos triatletas em relação aos outros atletas das modalidades especificadas. O mesmo fato observa-se em relação à estatura e à massa corporal (CONLEY; KRAHENBUHL, 1980; O'TOOLE et al., 1987).

Os valores das dobras cutâneas medidas nos triatletas capixabas estão abaixo (dobra cutânea do tríceps), ou na média (dobras cutâneas subescapular, supra-ilíaca e abdominal), preconizada por Nieman (1986).

Os triatletas de sexo masculino, tanto os de endurance, quanto os de ultra-endurance, geralmente apresentam um percentual de gordura que varia entre 6 e 11% (LAURENSEN; FULCHER; KORKIA, 1993; O'TOOLE et al., 1987; SCHNEIDER; POLLOCK, 1993). O percentual de gordura dos triatletas capixabas, 12,6%, está acima dos valores médios preconizados para uma boa performance física. Pesquisas realizadas com triatletas brasileiros relatam um percentual de gordura, em torno de 9% para os atletas amadores e 7,7% para os atletas profissio-

nais (BASSIT; MALVERDI, 1998). Esse valor acima poderia ser explicado devido a um balanço energético positivo, ou seja, a uma dieta hipercalórica; a um gasto energético menor do que a média necessária para essa modalidade ou uma combinação desses dois fatores. Quanto mais pesado o triatleta, excesso de gordura corporal, menor é o desempenho motor durante a competição (BOILEAU; LOHMAN, 1977; MALINA, 1992; PATE; SLENTZ; KATZ, 1989). Se, por um lado, o excesso de gordura poderia melhorar a flutuabilidade do atleta durante a fase de natação, por outro o atrito hidrodinâmico na natação e a força da gravidade durante a corrida são as maiores forças a serem vencidas pelo atleta durante a competição (GULLSTRAND, 1992; TITTEL; WUTSHERK, 1992). Com isso, o excesso de gordura corporal aumentaria o desgaste físico do atleta, comprometendo a sua performance final.

Muito poucas pesquisas têm investigado a relação entre a composição corporal de atletas de alto nível e os níveis de saúde. Os níveis mínimos de % de gordura, compatível com uma boa saúde, para indivíduos dos sexos masculino e feminino são, 12% e 5%, respectivamente (LOHMAN, 1992). Dados indicam que o excesso de gordura pode influenciar negativamente na performance de muitos esportes, e que a relação entre a massa corporal magra versus percentual de gordura, para uma determinada massa corporal, está diretamente relacionada à performance no esporte e que níveis extremamente baixos de percentual de gordura pode resultar em deteriorização na saúde e performance física (HOUTKOOPER; GOING, 1994).

Em relação aos valores de % de peso ósseo e peso muscular poucas pesquisas têm relatado a influência desses fatores na performance dos triatletas. É necessária uma melhor investigação para estabelecer a relação dessas variáveis no triatlon, visto que essa modalidade exige uma realização alternada de natação, ciclismo e corrida.

As demandas fisiológicas da execução seqüenciada de três modalidades diferentes exigem do triatleta características físicas e fisiológicas específicas. Os triatletas capixabas apresentam em relação às médias de idade, peso e estatura valores semelhantes ou próximos aos dados relatados na literatura nacional e internacional, porém um valor elevado de percentual de gordura. As pesquisas confirmam que valores elevados de percentual de gordura interferem negativamente na performance do triatleta, tanto devido à relação inversa entre o peso de gordura e a performance nas atividades que requerem deslocamento, pela redução da

aceleração, como devido ao aumento do custo energético. Diante disso, é necessária uma investigação mais aprofundada dos fatores que influenciaram os atletas a apresentarem um elevado valor de percentual de gordura, visto que este pode ser causado por diversos fatores tais como: tipo de protocolo utilizado para predição de percentual de gordura, falta de um protocolo específico para predizer a composição corporal de triatletas, gasto energético menor nos treinamentos dos triatletas capixabas, uma alimentação hipercalórica ou associação desses fatores.

BODY COMPOSITION OF THE CAPIXABAS TRIATLHETES

ABSTRACT

This experiment evaluated the corporal composition of the triatlhetes capixabas that participated in the triathlon of the Army - 2000. The skinfolds thickness were measured of the triceps, subscapular, suprailiac and abdomen, the bony diameters of the radius and of the femur, besides the body mass and stature, seeking the determination of the fat mass, bony mass, muscle mass and residual mass The data were treated in the form of descriptive statistics. The triatlhetes presented the following characteristic anthropometric: age 31,69 years, body mass 70,7 Kg, 173,8 cm of stature, 44,35 mm sum of the skinfolds, 12,57 % for relative body fat, 47,75% of muscle mass and 15,68% of bony mass. In relation to the averages of age, weight and stature is observed value similar to the data related in the literature, even so the relative body fat presents an index elevated in relation to the international and national medium patterns.

Keywords: Triatlhetes. Body composition. Triathlon. Fat mass. Bony mass. Muscle mass. Residual mass.

REFERÊNCIAS

BASSIT, R. A.; MALVERDI, M. A. Avaliação nutricional de triatletas. **Rev. Paul. Educ. Física**, São Paulo, n. 12, p. 42-53, 1998.

BOILEAU, R. A.; LOHMAN, T. G. The measurement of human physique and its effect on physical performance. **Orthopedic Clin. N. Am.**, n. 5, p. 15-18, 1977.

BUSKIRK, E.; TAYLOR, H. L. Maximal oxygen intake and its relation to body composition with special reference to chronic physical activity and obesity. **J. Appl. Physiol.**, n. 10, p. 24-28, 1957.

CONLEY, D. L.; KRAHENBUHL, G. S. Running economy and distance running performance of highly trained athletes. **Med. Sci. Sports Exerc.**, n. 12, p. 357-360, 1980.

DE ROSE, E. H.; PIGATTO, E.; DE ROSE, R. C. F. **Cineantropometria, educação física e treinamento desportivo**. Brasília: MEC, SEED, 1984.

GULLSTRAND, L. Swimming an endurance sport. In: SHEPHARD, R. J., ASTRAND, P.-O. (Ed.). **Endurance in sport**. Oxford: Blackwell Scientific, 1992.

HARMAN, E. A.; FRYKMAN, P. N. The relationship of body size and composition to the performance of physically demanding military tasks. In: MARRIOTT, B. M.; GRUMSTRUP-SCOTT, J. (Ed.). **Body composition and physical performance**: applications for the military services. Washington: National Academy Press, 1992.

HOUTKOOPER, L. B.; GOING, S. B. Body composition: how should it be measured? Does it affect sport performance? **Sports Sci. Exch.**, n. 7, p. 45-50, 1994.

LAMB, D. R. **Physiology of exercise, responses and adaptations**. 2nd ed. New York: MacMillan, 1984.

LAURENSEN, N. M.; FULCHER, K. Y.; KORKIA, P. Physiological characteristics of elite and club level female triathletes during running. **Int. J. Sports Med.**, n .14, p. 455-459, 1993.

LOHMAN, T. G. Basic concepts in body composition assessment. In: ADVANCES in body composition assessment. Champaign: Human Kinetcs, 1992.

MALINA, R. M. Physique and body composition: effects on performance and effects on training, semistarvation, and overtraining. In: BROWNELL, K. D.; RODIN, J.; WILMORE, J. H. (Ed.). **Eating, body weight, and performance in athletes**. Philadelphia: Lea & Febiger, 1992.

NIEMMAN, D. C. **The sports medicine fitness course**. Palo Alto: Bul Publishing, 1986.

O'TOOLE, M. L.; DOUGLAS, P. S. Applied physiology of triathlon. **Sports Med.**, n. 19, p. 251-267, 1995.

O'TOOLE, M. L. et al. The ultraendurance triathlete: a physiological profile. **Med. Sci. Sports Exerc.**, n.19, p. 45-50, 1987.

PATE, R. R.; SLENTZ, C. A.; KATZ, D. P. Relationships between skinfold thickness and performance of health related fitness test items. **Res. Quart. Exerc. Sport**, n. 10, p.120-128, 1989.

POLLOCK, M. L. Submaximal and maximal working capacity of elite distance runners. Part I. Cardiorespiratory aspects. **Ann. N. Y. Acad. Sci.**, n. 301, p. 310-322, 1977.

POLLOCK, M. L.; WILMORE, J. H.; FOX III, S. M. **Exercise in health and disease**: evaluation and precripton for prevention and rehabilitation. New York: W. B. Saunders, 1984.

SÁ, S. A. G. de. **Biometria em Educação Física**. São Paulo: McGraw-Hill, 1975.

SCHNEIDER, D. A.; POLLOCK, J. Ventilatory threshold and maximal oxygen during eyeling and running in female triathletes. **Int. J. Sports Med.**, n. 12, p. 379-383, 1993.

SINNING, W. E.; WILSON, J. R. Validity of "generalized" equations for body composition analysis in women athletes. **Res. Quart. Exerc. Sport**, n. 35, p. 90-98, 1984.

TITTEL, K.; WUTSHERK, H. Anatomical and anthropometric fundamentals of endurance. In: SHEPHARD, R. J.; ASTRAND, P.-O. (Ed.). **Endurance in sport**. Oxford: Blackwell Scientific, 1992.

WILMORE, J. H.; COSTILL, D.L. Semi-automated systems approach to the assessment of oxygen uptake during exercise. **J. App. Physiol.**, n. 36, p. 618-620, 1974.

WILMORE, J. H.; BROWN, C. H.; DAVIS, J. A. Body physique and composition of the female distance runner. **Ann. N. Y. Acad. Sci.**, n. 301, p. 764-776, 1977.

UM TRATAMENTO PARA O PROBLEMA DE ALOCAÇÃO DE DISCIPLINAS ÀS SALAS DE AULA UTILIZANDO A META-HEURÍSTICA COLÔNIA DE FORMIGAS

SCIENTIA

ELIZABETH MARIA KLIPPEL AMANCIO PEREIRA[1]
ESTER MARIA KLIPPEL[2]
LUIZ CARLOS FRAGA JUNIOR[3]

[1] Mestre em Informática pela Universidade Federal do Espírito Santo. Professora do Centro Universitário Vila Velha. E-mail: elizabeth@uvv.br.

[2] Mestre em Informática pela Universidade Federal do Espírito Santo. Professora do Centro Universitário Vila Velha. E-mail: ester@uvv.br.

[3] Graduando do Curso de Ciência da Computação do Centro Universitário Vila Velha.

RESUMO

Apresenta a meta-heurística Colônia de Formigas como uma possível solução para o problema de alocação de disciplinas às salas de aula. O problema consiste na distribuição de disciplinas às salas de acordo com o número de matrículas, horário das disciplinas e características das salas. A solução do problema visa evitar a ociosidade no uso das salas enquanto atende as restrições: sobreposição de horários; adequação da capacidade da sala ao número de matrículas; e, adequação dos recursos oferecidos pela sala às particularidades exigidas pela disciplina.

Palavras-chave: Otimização. Meta-heurística. Alocação de disciplinas às salas de aula. Colônia de Formigas.

1 INTRODUÇÃO

O problema Alocação de Disciplinas às Salas de Aula diz respeito à distribuição de disciplinas, com horários previamente estabelecidos às salas de aula, levando em consideração um conjunto de restrições de várias naturezas (SOUZA; MARTINS; ARAÚJO, 2002). Este problema se repete semestralmente ou anualmente, ocasionando a necessidade de adequação das salas à quantidade de alunos matriculados e às particularidades de cada disciplina. Em geral, as instituições realizam este trabalho de forma manual, e cada alteração realizada na alocação de alguma disciplina, ocasiona sérios transtornos, além de prejuízos financeiros com a ocorrência de salas com carga horária não sendo utilizadas na totalidade.

O problema Alocação de Disciplinas às Salas de Aula pode ser considerado como uma parte integrante do problema de programação de cursos, modelado na Figura 1.

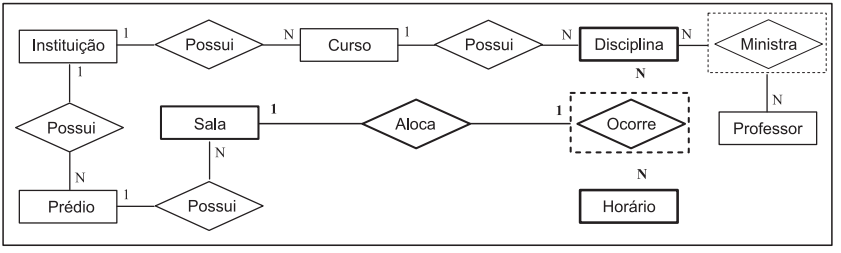


Figura 1 – Problema de Programação de Cursos

A solução manual do problema de Alocação de Disciplinas às Salas de Aula é uma tarefa árdua e normalmente requer vários dias de trabalho. Além do mais a solução pode ser insatisfatória com relação a vários aspectos, por exemplo, em função da alocação feita, pode haver em um dado horário um fluxo acentuado de alunos deslocando-se de salas com conseqüente perturbação no ambiente. Dada a dificuldade de se obter uma solução manual, a característica combinatória e a difícil generalização em virtude da quantidade de variantes que o problema pode assumir, este vem sendo objeto de estudo de vários pesquisadores, que buscam uma solução automatizada. Entretanto, sua automação não é uma tarefa das mais simples, pois o problema é NP-difícil, o que, em casos reais, dificulta ou até mesmo impossibilita sua resolução por técnicas de programação matemática, ditas exatas.

Assim sendo, esse problema é normalmente abordado através de técnicas heurísticas, que apesar de não garantirem a otimalidade, conseguem, em geral, gerar soluções de boa qualidade sem um elevado custo computacional. Dentre as heurísticas, destacam-se as chamadas meta-heurísticas, que têm caráter geral e são dotadas de mecanismos para escapar de ótimos locais. Como possíveis meta-heurísticas a serem aplicadas ao problema podemos destacar, dentre outras: *Simulated Annealing*, *Genetic Algorithm*, *Tabu Search* e *Ant Colony*.

O presente trabalho focaliza o problema de alocação de disciplinas às salas de aula do prédio de Ciências Exatas e Tecnológicas do Centro Universitário Vila Velha, no qual, a cada período letivo, as disciplinas devem ser designadas às salas, procurando-se evitar a ociosidade, adequando o número de alunos à capacidade das salas, evitando sobreposição de horários, e atendendo às particularidades de algumas disciplinas. O prédio de Ciências Exatas e Tecnológicas recebe, em média, 3000 alunos por semestre e oferece cerca de 40 (quarenta) disciplinas nos turnos matutino e noturno. As aulas são realizadas de segunda a sexta. Para realização das aulas estão disponíveis 30 (trinta) salas, 7 (sete) laboratórios de informática e 1 (um) laboratório de Física. Os horários das aulas são confeccionados previamente pelas Coordenações de Curso e encaminhados ao setor de apoio para que se faça a alocação das disciplinas às salas.

Para resolver o problema propomos um ambiente integrado de resolução de problemas, onde utilizaremos métodos meta-heurísticos para a obtenção de uma metodologia alternativa de solução. As principais con-

tribuições esperadas com a conclusão deste trabalho são: melhorar a qualidade dos serviços prestados aos alunos, diminuir a ociosidade do espaço físico disponível e disseminar a utilização de técnicas de Pesquisa Operacional.

Este trabalho está organizado como segue. Na seção 2 descreve-se o problema, modelando-o como um problema de Programação Linear Inteira. Na seção seguinte detalha-se a meta-heurística a ser usada para tratar o problema. A seção 4 traz uma proposta de adequação da meta-heurística ao problema.

2 CARACTERIZAÇÃO DO PROBLEMA DE ALOCAÇÃO DE DISCIPLINAS ÀS SALAS DE AULA

O problema de alocação de disciplinas às salas de aula pode ser considerado um problema de Programação Linear Inteira onde os recursos estão sendo alocados às atividades numa base de um para um.

O problema de alocação de disciplinas às salas de aula pode ser caracterizado por três conjuntos:

– conjunto $H = \{h1, h2, ..., hp\}$ de p horários de aula;

– conjunto $D = \{d1, d2, ..., dm\}$ de m disciplinas; e

– conjunto $S = \{s1, s2, ..., sn\}$ de n salas.

A solução do problema visa evitar a ociosidade no uso das salas enquanto atende a restrição de sobreposição de horários no sentido de que uma sala deve alocar no máximo uma disciplina em um determinado horário e uma disciplina pode estar no máximo em uma sala em um mesmo horário; a restrição de adequação da capacidade da sala ao número de alunos matriculados; e, a restrição que trata da adequação dos recursos oferecidos pela sala às particularidades exigidas pela disciplina.

Um horário de aula h_k pertencente a H, para $k=1,...,p$, pode ser caracterizado pelos seguintes parâmetros:

- curso;
- disciplina;

- turma;
- professor;
- turno (matutino, vespertino ou noturno);
- dia da semana (segunda, terça, quarta, quinta ou sexta-feira); e
- aula (primeira, segunda, terceira, quarta, quinta ou sexta).

Cada horário é caracterizado pela sua duração, pelo assunto e pelos participantes, professores e alunos, que devem reunir-se para a sua realização. Assim, um horário define um período de horas durante o qual os respectivos participantes estão disponíveis para a sua realização, de tal forma que sejam satisfeitos os requisitos fundamentais, de que cada participante esteja vinculado a não mais do que uma aula ao mesmo tempo, ou seja, um professor não pode ministrar aula para mais de uma turma ao mesmo tempo e uma turma não pode ter aula com mais de um professor em um mesmo horário. Assumiremos, no presente trabalho, que horários viáveis das disciplinas já estão devidamente definidos.

Uma disciplina d_i pertencente a D, para $i=1,...,m,$ pode ser caracterizada pelos seguintes parâmetros:

- curso a qual pertence;
- carga horária total;
- carga horária semanal;
- área do conhecimento;
- horário;
- número de alunos matriculados; e
- tipo de sala utilizada (sala de aula, laboratório de informática ou física, etc.).

Assumiremos, no presente trabalho, que as matrículas já foram efetuadas.

Uma sala s_j pertencente a $S,$ para $j=1,...,n,$ pode ser caracterizada pelos seguintes parâmetros:

- prédio;
- destinação (sala de aula, laboratório de informática, laboratório de física, etc.);

- capacidade;
- conjunto de horários semanais nos quais a sala *j* está disponível para ser alocada a alguma disciplina do conjunto *D*. Este conjunto visa definir horários para limpeza e preparação da sala *j*.

Considerando a variável de decisão x_{ijk} abaixo definida:

$$x_{ijk} = \begin{cases} 1, & \text{se a disciplina } i \text{ está alocada na sala } j \text{ no horário } k \\ 0, & \text{caso contrário} \end{cases}$$

Podemos modelar o problema de alocação de disciplinas às salas de aula como segue:

$$\text{Min} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p} w_{ijk}\, x_{ijk} \quad \text{onde } w_{ijk} \text{ mede o nível de ociosidade}$$

$$\text{s.a.} \quad \sum_{k=1}^{p}\sum_{i=1}^{m} x_{ijk} \leq 1 \qquad \forall\, j \in S \qquad (i)$$

$$\sum_{k=1}^{p}\sum_{j=1}^{n} x_{ijk} \leq 1 \qquad \forall\, i \in D \qquad (ii)$$

$$x_{ijk}\, Y_i \leq C_j \qquad \forall\, i \in D, \forall\, j \in S, \forall\, k \in H \qquad (iii)$$

onde $Y_i = n^{o}$ de alunos matriculados na disciplina i

onde C_j = capacidade da sala j

$$x_{ijk}\, T_i = U_j \qquad \forall\, i \in D, \forall\, j \in S, \forall\, k \in H \qquad (iv)$$

onde T_i = tipo de sala usada pela disciplina i

onde U_j = destinação da sala j

A restrição *(i)* garante que uma sala deve alocar, no máximo, uma disciplina no mesmo horário. A restrição *(ii)* garante que uma disciplina pode estar, no máximo, em uma sala no mesmo horário. A restrição *(iii)* garante que o número de alunos matriculados em uma disciplina não deve ultrapassar a capacidade da sala para ela alocada. A restrição *(vi)* garante que uma sala deve ser adequada as exigências da disciplina a ela alocada. Uma solução viável para as restrições *(i)*, *(ii), (iii)* e *(iv)* representa uma alocação de disciplinas às salas. A função objetivo, ao minimizar o nível de ociosidade das salas, combate, em cada disciplina e horário, a ociosidade do espaço físico disponível.

3 A META-HEURÍSTICA COLÔNIA DE FORMIGAS – ANT COLONY

A meta-heurística Colônia de Formigas foi proposta como uma abordagem multi-agente para a resolução de problemas de otimização combinatória. Nesta estratégia, um conjunto de agentes, chamados formigas artificiais, ou simplesmente formigas, buscam em paralelo encontrar boas soluções para o problema proposto. Esta meta-heurística foi proposta a partir de experimentos e observações de uma colônia de formigas real, baseando-se no comportamento das formigas para escolher qual caminho devem tomar para encontrar a comida (DORIGO; MANIEZZO; COLORNI, 1996). Este comportamento permite as formigas encontrar os trajetos mais curtos entre fontes do alimento e seu ninho. Ao andar da fonte de alimento até o ninho e vice-versa, as formigas depositam uma substância chamada feromônio no caminho. Quando se decidem sobre qual sentido ir, os trajetos mais prováveis são marcados por concentrações mais fortes de feromônio. Este comportamento natural é a base para uma interação cooperativa que conduz as formigas ao trajeto mais curto.

É importante destacar também que as formigas são capazes de se adaptar a mudanças no ambiente. Como por exemplo, quando o primeiro caminho escolhido já não é o mais curto entre o ninho e a comida devido a algum obstáculo a reação imediata das formigas é contornar o obstáculo e decidir novamente qual o novo caminho mais curto. Considere, por exemplo, a Figura 2. Na Figura 2A as formigas têm inicialmente um caminho em linha reta entre o ninho e a comida. Na Figura 2B a trilha inicial de feromônio foi interrompida por um obstáculo. Isso obriga as formigas a procurarem um novo caminho até a comida. O problema agora é escolher qual direção tomar para contornar o obstáculo. As formigas então escolhem aleatoriamente a direção, como mostra a Figura 2C. As trilhas de feromônio são então restabelecidas e o próximo passo é decidir qual dos caminhos é o mais curto, ou seja, qual dos caminhos possui mais feromônio por unidade de tempo. Após essa análise, as formigas escolhem o caminho mais curto como mostra a Figura 2D.

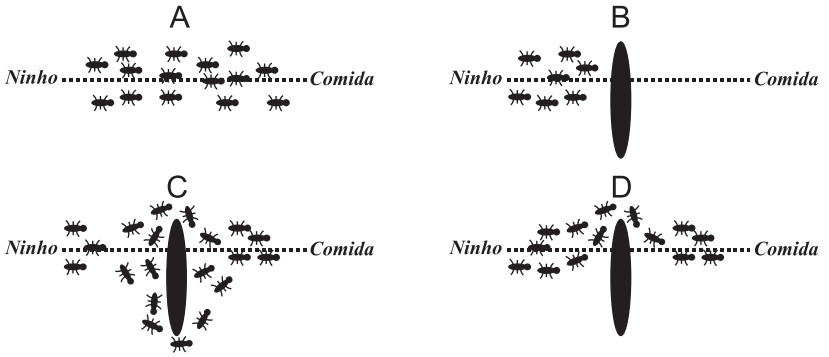


Figura 2 – Formigas reais desviando de obstáulos

Inicialmente, a meta-heurística Colônia de Formigas foi aplicada ao problema do Caixeiro Viajante (DORIGO; GAMBARDELLA, 1997). Desde então, pesquisadores têm desenvolvido atividades com o intuito de aprimorar e aplicar a abordagem aos mais diferentes problemas de otimização. Dentre essas aplicações estão o problema de Escalonamento com Restrição de Recursos (LORENZONI; AHONEN; ALVARENGA, 2001) e Otimização da Alocação de Sondas de Produção Terrestre (ALOISE et al., 2001).

A meta-heurística Colônia de Formigas é baseada em modelos probabilísticos parametrizados usados para modelar o caminho marcado pelo feromônio (DORIGO; MANIEZZO; COLORNI, 1996). Para isso, as formigas artificiais executam caminhadas aleatórias em um grafo completamente conexo $G = (N, A)$, cujos vértices são os componentes N e A representa o conjunto das conexões. Este grafo é comumente chamado de Grafo de Construção. Quando um problema de otimização combinatorial com restrições é considerado, as restrições do problema são embutidas no procedimento de construção das formigas artificiais.

Na maioria das aplicações, as formigas são implementadas para construir soluções viáveis, mas às vezes é desejável também deixar um parâmetro para a trilha de feromônio. Estes valores de feromônio são usados pelas formigas para tomar decisões probabilísticas sobre como se mover no Grafo de Construção.

Cada formiga aplica passo a passo uma política de decisão para construção de uma solução local, para, então, construir uma solução do

problema como um todo. Em cada informação local do nó, mantida no próprio nó ou nos arcos que partem deste nó, é usada uma maneira de decidir o nó seguinte para o qual a formiga irá se mover. A regra de decisão usada por uma formiga *k* que se localiza no nó *i* é a trilha de feromônio t_{ij} , usada para computar a probabilidade p_{ij}^{k} da formiga *k* escolher o nó $j \in N_i$ para o próximo movimento, onde N_i representa o conjunto de nós vizinhos ao nó *i*. No início do processo de procura, uma pequena quantidade de feromônio é associada a todas as arestas e, a cada vez que uma formiga passa por essa aresta, a quantidade de feromônio é atualizada. Desta forma podemos descrever uma fórmula simples para calcular p_{ij}^{k} na meta-heurística de otimização Colônia de Formigas:

$$p_{ij}^{k} = \begin{cases} \dfrac{t_{ij}}{\sum_{j \in N_i} t_{ij}}, & se\ j \in N_i \\ 0, & se\ j \notin N_i \end{cases}$$

Enquanto a solução é construída, as formigas artificiais depositam feromônio nas arestas em que passam. Para esta regra, que simula o feromônio das formigas reais, uma formiga que usa o nó *i* ligado por uma aresta ao nó *j* aumenta a probabilidade de outras formigas usarem o mesmo nó no futuro. A Figura 3 abaixo mostra uma formiga artificial construindo uma solução do ninho (início) até a comida (fim). A cada nó percorrido ela deposita uma quantidade de feromônio. A próxima formiga artificial tenderá a seguir os caminhos que possuírem mais feromônio, no caso da figura, os nós mais escuros.

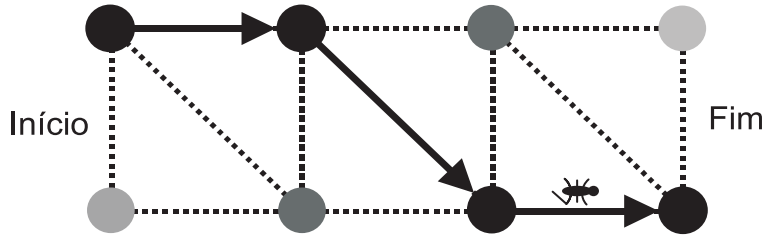


Figura 3 – Atualização de feromônio

Para uma rápida convergência, a trilha de feromônio que não está sendo utilizada “evapora”, assim como em uma colônia de formigas real, decrementando a quantidade de feromônio depositado nos nós que não estão sendo percorridos a cada iteração do algoritmo.

Em geral, as soluções encontradas por cada formiga, trabalhando isoladamente, são de qualidade inferior. Melhores soluções são encontradas quando as formigas trabalham de forma cooperativa.

As soluções são expressas como um caminho, com custo mínimo, que respeite as restrições impostas pelo problema. O estado interno da formiga armazena informações sobre o seu passado histórico, tais como a seqüência de estados que gera a solução, a contribuição de cada movimento executado e o valor da solução gerada, formando assim sua memória.

Cada formiga constrói uma solução movendo-se através de uma seqüência finita de estados sucessivos. Os movimentos são selecionados a partir da regra de transição de estados, que se baseia na memória da formiga e na quantidade de feromônio acumulada nas trilhas. O que se observa é que pontos que não foram muito visitados no início serão menos ainda com o passar do tempo. Uma vez que a formiga realizou sua tarefa, que é a construção de uma solução e o depósito de feromônio, ela morre, ou seja, é eliminada do sistema.

A cada transição é associada uma quantidade de feromônio no ambiente. Decisões sobre quando espalhar e o quanto depositar de feromônio dependem das características do problema e do projeto de implementação. As formigas podem espalhar feromônio durante a construção da solução, assim, cada vez que um novo estado é selecionado, o feromônio associado à transição pode ser atualizado. Em geral, a quantidade de feromônio depositada é proporcional à qualidade da solução que foi construída (ou que está em construção).

Por fim, em pseudocódigo a meta-heurística Colônia de Formigas pode ser descrita da seguinte forma, segundo (LORENZONI; AHONEN; ALVARENGA, 2001):

Meta-heurística colônia de formigas

Repita

Crie as formigas e posicione cada uma num estado inicial;

Repita

Para cada formiga faça

Aplique a regra de transição de estado para incrementalmente construir uma solução;

Aplique a regra de atualização local do feromônio;

Fim para;

Até que todas as formigas tenham construído uma solução completa;

Gere a solução associada a cada uma das formigas;

Aplique a regra de atualização global do feromônio;

Guarde a melhor solução encontrada até o momento;

Até que o critério de parada seja satisfeito.

4 PROPOSTA DE USO DA META-HEURÍSTICA COLÔNIA DE FORMIGAS PARA O PROBLEMA DE ALOCAÇÃO DE DISCIPLINAS ÀS SALAS DE AULA

A chave para aplicações que envolvem a meta-heurística Colônia de Formigas é adaptar e identificar o problema apropriadamente (representando o grafo que será percorrido pelas formigas) e definir o que cada nó e cada aresta representam. Então, a interação probabilística entre as formigas, que vão criando a trilha de feromônio e os nós trarão boas soluções para o problema. Neste artigo, a implementação da meta-heurística Colônia de Formigas é apresentada como uma possível solução para o problema de Alocação de Disciplinas às Salas de Aula. É importante destacar que o próximo passo é definir e adaptar o problema, como foi citado acima, ou seja, definir quais parâmetros do problema serão utilizados para definir os estados, as atualizações de feromônio, os obstáculos pelos quais as formigas artificiais podem ter que enfrentar entre outros parâmetros, para que então a meta-heurística possa ser implementada e executada para comparações de eficiência com outras meta-heurísticas.

A TREATMENT FOR THE CLASSROOM ASSIGNMENT PROBLEM USING THE ANT COLONY METAHEURISTC

ABSTRACT

The Ant Colony metaheuristic is presented as possible solution for the problem of allocation of discipline to the classrooms. The problem consists of the distribution of disciplines to the rooms in accordance with the number of students, schedule of disciplines and classrooms characteristics. The problem solution aims to prevent the idleness in the use of the rooms while it attend the schedules overlapping restriction; the capacity adequacy restriction of the room to the number of registered students; and, the restriction that deals with the adequacy of the offered resources for the room to the discipline demanded particularitities.

Keywords: Optimization. Metaheuristic. Classroom assigment problem. Ant Colony.

REFERÊNCIAS

ALOISE, D. et al. Heurísticas de “Colônia de Formigas” para o problema de otimiza-ção da alocação de sondas de produção terrestre. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 33., 2001, São Paulo. **Anais**… São Paulo: DIMAP, 2001. 13 p.

DORIGO, M.; GAMBARDELLA, L. **Ant colonies for the traveling salesman problem**. Bruxelles: Université Libre de Bruxelles, 1997.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. The ant system: optimization by a colony of cooperating agents. **IEEE Transaction System Man Cybern. B**, New York, v. 26, p. 29-41, 1996.

LORENZONI, L.; AHONEN, H., ALVARENGA, A. Colônia de Formigas para proble-mas de escalonamento com restrição de recursos. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 33., 2001, São Paulo. **Anais**… São Paulo: DIMAP, 2001.

SOUZA, M. J. F.; MARTINS, A. X.; ARAÚJO, C. R. Experiências com Simulated Annealing e Busca Tabu na resolução do problema de alocação de salas. SIMPÓ-SIO BRASILEIRO DE PESQUISA OPERACIONAL, 34., 2002, Rio de Janeiro. **Anais**… Rio de Janeiro: Instituto Militar de Engenharia, 2002. 12 p.

IDENTIFICAÇÃO DE MICRORGANISMOS EM AMBIENTES PÚBLICOS

SCIENTIA

MARCOS SANTOS ZANINI[1]
JOÃO DAMASCENO LOPES MARTINS[2]
DANIELLY FELIX[3]
CARLA GOES BATALHA[3]

[1] Doutorando em Ciência Animal na UFMG. Professor do Curso de Medicina Veterinária do Centro Universitário Vila Velha. E-mail: zaninims@uvv.br.

[2] Biólogo. Professor do Centro Universitário Vila Velha. E-mail: jodam@uvv.br.

[3] Alunas do Curso de Medicina Veterinária do Centro Universitário Vila Velha.

Scientia Vila Velha (ES), v. 4, n. 1/2, **p. 29-37**, jan/dez. 2003

RESUMO

Utiliza a observação de microrganismos contaminantes de mesas de bares e restaurantes da Grande Vitória como atividade prática no aprendizado de conteúdos da disciplina de Microbiologia Veterinária-turma 2000. A amostragem pesquisou 120 bares e/ou restaurantes da Grande Vitória. Selecionou-se aleatoriamente uma mesa ou balcão em cada estabelecimento, restringindo a área de superfície de 10cm^2 para fricção de swab estéril e posterior semeadura em meios de enriquecimento, seletivos e testes bioquímicos. Os dados apontam uma alta contaminação nas superfícies de mesas de alguns estabelecimentos pesquisados e presença significativa de Staphilococcus aureus (30%) o que pode potencialmente desencadear distúrbios digestivos inespecíficos nos consumidores e dermatites diversas, sendo necessária assim uma verificação dos processos de higienização aplicados, assim como valorização da Análise de Perigos e Pontos Críticos de Controle (APPCC). Sob o aspecto de aprendizado dos alunos, a metodologia atingiu seus objetivos, pois eles interagiram com diversas técnicas microbiológicas e perceberam sua inserção profissional na saúde pública.

Palavras-chave: Bares. Distúrbios digestivos. Higienização. Restaurantes. Utensílios

1 INTRODUÇÃO

O objetivo deste trabalho foi utilizar a observação de microrganismos contaminantes de mesas de bares e restaurantes da Grande Vitória como atividade prática no aprendizado de conteúdos da disciplina de Microbiologia Veterinária-turma 2000, a fim de contemplar os aspectos de ensino, pesquisa e extensão. O controle de qualidade de ambientes de lazer como bares e restaurantes consta de vários pontos, que vão desde a verificação de temperatura de alimentos servidos até o monitoramento da higienização dos utensílios e superfícies de contato de uma forma geral. O meio ambiente público, incluindo o ar, a água e as superfícies inanimadas que cercam os freqüentadores de bares e restaurantes, guarda íntima relação com as infecções, podendo proporcionar focos de contato e de transmissão. Embora as principais causas de infecção estejam relacionadas ao doente susceptível à infecção e com os métodos-diagnósticos e terapêuticos utilizados, não se pode deixar de considerar a parcela de responsabilidade relacionada aos padrões de assepsia e de higiene

do ambiente. Dentre as atividades executadas no cotidiano dos bares/ restaurantes há a limpeza freqüente das mesas, reconhecendo-a como uma das formas de manter o ambiente biologicamente seguro. De uma maneira geral, os diferentes tipos de limpeza são realizados para a remoção de sujidade, com a finalidade primordial de impedir a disseminação de microrganismos que colonizam as superfícies horizontais dos mobiliários, como *Staphylococcus aureus, Clostridium difficile, Pseudomonas sp, Proteus sp, Serratia marcescens, Candida sp* e coliformes em geral. No processo de limpeza de mesas recomenda-se a utilização de produtos químicos com ação germicida, eficazes para remoção e destruição de microrganismos existentes na superfície. A utilização de substâncias germicidas tem levado, inclusive, a uma mudança da terminologia de limpeza de ambientes para *desinfecção*, o que já está sendo evidenciado na literatura (AYLIFFE, 1984; BLOOMFIELD; SCOTT, 1997). No Brasil, vários produtos têm sido indicados, devendo eles possuir princípios ativos fenólicos, compostos orgânicos e inorgânicos liberadores de cloro ativo, princípios quaternários de amônia ou de álcoois, ou outros que atendam à legislação atual específica. Dessa forma, decidiu-se realizar avaliação microbiológica como atividade de pesquisa, o envolvimento do aluno na execução de todas as etapas como ensino e os resultados como extensão, pois estas informações podem subsidiar melhoras da saúde pública (SCOTT; BLOOMFIELD, 1985).

2 MÉTODOS

A unidade de estudo definida foi determinada como sendo as mesas sobre as quais são servidos os mais diversos alimentos em 120 bares ou restaurantes. Selecionou-se aleatoriamente uma mesa ou balcão em cada estabelecimento, restringindo a área de superfície de 10cm^2 para a pesquisa. Utilizaram-se para coletar o material das superfícies, swabs estéreis embebidos em solução salina estéril, de onde se partiu para semeadura em ágar-sangue. Seguiu-se a semeadura em meios de seletivos e diferenciais assim como algumas provas bioquímicas. O meio de cultura utilizado para contagem de UFCs (unidades formadoras de cêlonias) foi o ágar-sangue ou AS, meio não seletivo indicado para o crescimento de vários microrganismos, como bacilos gram-positivos, bacilos gram-negativos, cocos gram-positivos, inclusive de fungos. Para o isolamento e identificação de bactérias foram utilizados Ágar Baird-Parker e Ágar EMB (eosin methylene blue Lactose Sucrose Agar), ágar tripticase soy TS (Oxoid); ágar Mc Conkey (Oxoid); ágar

cetrimide (Oxoid); ágar Salmonella-Shigella (Oxoid); ágar triple sugar iron TSI (Difco); e ainda tiras para prova de oxidase (NewProv), provas de coagulase (*Staphylococcus aureus*). As colônias suspeitas emergentes no EMB foram submetidas às provas bioquímicas do indol, vermelho de metila, Voges-Proskauer e citrato (LEITÃO, 1998). Alguns destes testes têm como fim a mesma identificação microbiana, mas foram aplicados com finalidade didática para os alunos.

No laboratório as placas foram incubadas a 37ºC por 24 horas. Decorrido o tempo necessário de incubação, foi observado o aspecto das colônias (cor, forma, cheiro, tamanho) sobre a superfície das placas e a contagem delas. A leitura (observação e contagem das colônias) e identificações foram realizadas pelos alunos com o monitoramento do técnico do laboratório, professor ou monitor da disciplina.

3 RESULTADOS

Foi também evidenciado o crescimento de fungos nas placas de ágar-sangue. Apesar de a investigação de fungos não ter sido objeto do presente estudo, julgou-se relevante, devido à sua presença marcante nas placas. Observou-se o crescimento de fungos em 22 placas. Os resultados estão expressos no Quadro 1.

Microrgamismos/Gênero	120 Bares Analisados
S. aureus	ə (30%) ξ (7%) φ (19%)
S.epidermidis	(6%) ξ (12%) φ (7%)
S. saprophyticus	ə (5%) ξ (1%)
Corinebacterium sp	ə (16%) ξ (6%) φ (2%)
E. coli	ə (3%) ξ (2%) φ (4%)
Klebsiella pneumoniae	ə (8%) ξ (4%) φ (2%)
Proteus mirabilis	ə (2%) ξ (2%) φ (4%)
Klebsiella oxytoca	ə (1%) φ (2%)
Pseudomonas sp	ə (3%) φ (1%)
Enterococo sp	ə (2%)
Citrobacter sp	ξ (2%) φ (1%)
Bacillus	ə (8%) ξ (15%)
Hafnia alvei	ξ (2%)
Alcaligenes sp	ξ (1%)

Microrgamismos/Gênero	120 Bares Analisados
Proteus vulgaris	ξ (1%)
Acinetobacter sp	϶ (2%)
Citrobacter freundii	϶ (2%)
Micrococcus sp	϶ (2%)
Serratia sp	϶ (2%)
Citrobacter diversus	϶ (1%)
Enterobacter cloacae	϶ (1%)
Enterobacter sp	϶ (1%)
Flavobacterium sp	϶ (1%)
Providencia sp	϶ (1%)
P. aeruginosa	϶ (1%)

Quadro 1 – Concentração (UFCs) de aeróbicos isolados em ágar-sangue a partir de 1cm² de mesa de bares e restaurantes da Grande Vitória

϶ (+) até 2 UFC/cm² ξ (++) 3 à 10 UFC/cm² φ (+++) >10 UFC/cm²

4 DISCUSSÃO

As considerações que se seguem procuram evitar a abordagem com caráter de Inspeção Sanitária, aproximando-se de referências com dados hospitalares a fim de permitir uma avaliação mais microbiológica e didática.

Os dados apontam uma alta contaminação nas superfícies de mesas de alguns estabelecimentos pesquisados, o que pode potencialmente desencadear distúrbios digestivos inespecíficos nos consumidores, sendo necessária, assim, uma verificação dos processos de higienização aplicados, validando a prática da Análise de Perigos e Pontos Críticos de Controle (APPCC), conforme Portaria 1.428/93, Ministério da Agricultura (LIMA; MELO; SENA, 1998; SPECK, 1984). Tendo como base todas as considerações em torno de um ambiente público biologicamente seguro, observa-se falta de investimento tecnológico e diversidade de condutas como sendo um inadequado atendimento dos princípios científicos referenciados na literatura. Isto é, faltam indicadores microbiológicos que permitam estabelecer as condições de mesas ideais para sua reutilização. O estudo fornece contribuições relativas à limpeza pela avaliação microbiológica das mesas em que ficou caracterizada a manutenção da carga microbiana mesmo após

sua limpeza. Espera-se que os resultados obtidos fundamentem as decisões no âmbito de gerência, no sentido de introduzir mudanças na atividade de limpeza, em busca de tecnologia que atenda à relação custo/benefício. Os derivados do fenol são indicados para a limpeza e para a desinfecção de paredes, pisos, superfícies fixas em locais de grande risco e possuem efeito reduzido na presença de matéria orgânica. Sua ação é menos ativa em esporos, fungos e vírus. No presente estudo os elevados percentuais de culturas positivas por fungos comprovaram a ineficácia do fenol sobre ele. Embora o tempo utilizado de incubação das placas seja considerado pequeno para a semeadura de fungos, foi suficiente para sinalizar a qualidade do procedimento de limpeza (FRANCO, 1996; RIEDEL, 1992).

Scott e Bloomfield (1985) em estudo sobre a eficiência dos desinfetantes realizaram uma comparação da limpeza, em diversos locais do ambiente doméstico, com água e sabão e com a aplicação de desinfetantes fenólico e de hipoclorito. Os resultados indicaram a ineficiência da mistura água e sabão sem enxágüe em relação aos demais produtos nas mesmas condições. A aplicação dos desinfetantes fenólicos e de hipoclorito produziu um aumento significante de locais limpos para 38% e 76%, respectivamente. Por outro lado, os fenóis são também considerados poluentes ambientais e altamente tóxicos. As questões de alterações ecológicas também devem ser consideradas, o uso inadequado de produtos químicos, utilizados para a desinfecção ambiental e ativos para bactérias gram-positivas, podem favorecer o predomínio de bactérias gram-negativas. O Ministério da Saúde estabelece a legislação que controla o uso e a qualidade dos desinfetantes em termos de toxicidade e de atividade antimicrobiana, devendo possuir ação tuberculocida, fungicida e ser microbicida para *S. aureus, S. choleraesuis e P. aeruginosa.* Também é orientado sobre aspectos importantes relacionados ao preparo (concentração, PH, qualidade da água e outros), à estocagem e à utilização do produto para que se obtenha a máxima eficiência (BRASIL, 1997).

Entretanto, atualmente no mercado, há uma diversidade de produtos químicos germicidas que, apesar de representar melhores recursos para a higienização, traz a insegurança sobre qual produto é o mais indicado. Nesse sentido, há vasta literatura que fornece orientação sobre desinfetantes, deixando transparecer alguns aspectos conflitantes entre uma escolha e outra. Sem dúvida, a escolha do produto químico é de extrema importância, no entanto deve-se analisar quais

seriam os outros fatores que estariam contribuindo para a redução irrelevante de carga microbiana. Todavia, considerando a forma de execução do procedimento de limpeza, é possível antever que vários fatores podem interferir no seu resultado final. Estudos realizados com a finalidade de avaliar a efetividade da limpeza em geral evidenciaram que a limpeza mecânica proporciona diminuição de 80% na quantidade de microrganismos e com a utilização de desinfetante houve a eliminação de 90% a 95%. Nesse contexto, Bloomfield e Scott (1997) citam vários estudos que evidenciam a ineficiência do procedimento de limpeza apenas com água e sabão sem o devido enxágüe em água corrente. Os autores explicam que a descontaminação por meio da limpeza por intermédio da ação mecânica, isso é, apenas pela fricção na água e sabão, será efetiva se for rigorosamente aplicada em conjunto com um processo de enxágüe.

Uma outra atividade utilizada para finalizar o procedimento de limpeza é a secagem. Tal atividade foi analisada por testes de laboratório, cujos resultados indicaram que as bactérias residuais são espalhadas ao redor das superfícies e também no pano utilizado. Bloomfild e Scott (1997) verificaram que após um período limitado (de 90 minutos a 3 horas) muitos locais tornaram-se substancialmente recontaminados, sendo as razões para esse fato atribuídas ao tipo de pano utilizado na limpeza, ao excesso de umidade, que torna o material encharcado, ou à má escolha do produto químico (ação microbiana reduzida, que não destrói algumas espécies de microrganismos comumente encontradas na superfície em questão). Em suma, a secagem com panos, além de retirar o produto químico desinfetante, afetando sua ação residual, também permitiria o deslocamento da carga microbiana residual para outros pontos. Um outro ponto importante de reflexão e de questionamentos está relacionado ao custo, ao consumo de produtos químicos desinfetantes e ao tempo gasto (hora/ homem/ trabalho). Apesar da complexidade de tais questionamentos, acredita-se que medi-los é uma tarefa árdua, mas necessária. Estabelecer a relação custo/benefício ou medir os custos diretos da respectiva limpeza envolve estudos futuros, cuja importância é inquestionável para a saúde pública.

Por fim, sob o aspecto de aprendizado dos alunos, a metodologia atingiu seus objetivos, pois eles interagiram com diversas técnicas microbiológicas e perceberam sua inserção profissional na saúde pública.

ABSTRACT

In this worh used the observation of polluting microorganisms of tables of bars and restaurants of Surround Vitória as practical activity in the learning of contents of the discipline of Microbiology Veterinary-group 2000. The sampling researched 120 bars and/or restaurants of Surround Vitória. Random table or counter was selected a in each establishment, restricting the area of surface of 10cm^2 for friction of sterile swab and subsequent grow in enrichment and selective medium and test biochemical. The data point a high contamination in the surfaces of tables of some researched establishments and significant presence of Staphilococcus aureus (30%) what potentially can caused unspecific disturbances digestive inespecíficos in the consumers and several dermatitis, being like this necessary a verification of the sanitary processes of applied as well as valuing of the Hazard Analysis and Critical Control Point System (HACCP). Under the aspect of the students' learning the methodology reached your objectives, because the same ones interacted with several microbiology techniques and they noticed your professional insert in the public health.

Keywords: Bars. Disturbances digestive. Hygienie. Restaurants. Utensils.

REFERÊNCIAS

AYLIFFE, G. A. J. Princípios para la desinfectión en un hospital. **Laboratorio,** Madrid, v. 78, n. 4 , p. 223-35, 1984.

BLOOMFIELD, S. F.; SCOTT, E. Cross contamination and infection in the domestic environment and the role of chemical disinfectants**. J. Appl. Microbiol.**, v. 83, n. 6, p. 1-9, 1997.

BRASIL. Ministério da Saúde. Portaria n°. 451 de 19 de setembro de 1997. **Diário Oficial [da] República Federativa do Brasil**, Brasília, 22 set. 1997. Seção 1**,** p. 21005-21012.

FRANCO, B. D.; LANDGRA, F. M. Microorganismos patogênicos de importância em alimentos. In: MICROBIOLOGIA de alimentos. São Paulo: Atheneu, 1996. p. 33-81.

LEITÃO, M. F. F. Microganismos patogênicos em alimentos. In: ROITAMM, I.; TRAVASSOS, L. R.; AZEVEDO, J. L. **Tratado de Microbiologia**. São Paulo: Manole,1998. v. 1, p. 30-52.

LIMA, V. L.; MELO, E. A.; SENA , E. N. Condições higiênico-sanitárias de “Fast-food” e restaurantes da região metropolitana da cidade de Recife. **Higiene alimen-tar**, v. 12, n. 57, p. 50-54, set./out., 1998.

RIEDEL, G. Transmissão de doenças pelos alimentos. In: CONTROLE sanitário dos alimentos. 2. ed. São Paulo: Atheneu, 1992. p. 51-129.

SCOTT, E; BLOOMFIELD, S. F. A bacteriological investigation of the effectiveness of cleaning and disinfection procedures for toilet hygiene. **J. Appl. Bacteriol.**, v. 59, p. 291-7, 1985.

SPECK, M. L. **Compendium of methods for the microbiological examination of foods**. 2nd ed. Washington: APHA, 1984.

UM ESTUDO SISTEMATIZADO SOBRE A PROGRAMAÇÃO ORIENTADA A ASPECTOS

SCIENTIA

ELAINE FEITOSA SPERANDIO[1]
ADRIANA DE ANDRADE MOURA[2]

[1] Aluna do Curso de Ciência da Computação do Centro Universitário Vila Velha. E-mail: elainesperandio@yahoo.com.br.

[2] Mestre em Ciência da Computação pela Universidade Federal de Minas Gerias. Professora do Centro Universitário Vila Velha. E-mail: adriana.moura@uvv.br.

RESUMO

Estudos realizados até hoje mostraram muitos problemas relacionados à programação que nem a Programação Orientada a Objetos nem a Programação Estruturada foram suficientes para solucioná-los. Com isso surgiu a Programação Orientada a Aspectos (POA) que veio para solucionar problemas tais como entrelaçamento e espalhamento de código através da definição de aspectos. Com a POA o código se torna mais fácil de ser entendido e mantido, permitindo ganhos consideráveis com relação à qualidade do *software* e produtividade no desenvolvimento. Este trabalho apresenta um estudo sistematizado sobre a POA.

Palavras-chave: Aspectos. Aspectj. Programação Orientada a Aspectos. Preocupações ortogonais.

1 INTRODUÇÃO

O desenvolvimento de *software* já foi um processo bastante demorado e oneroso. Entretanto, as ferramentas de programação visual determinaram uma nova abordagem para a construção de aplicativos, antigas práticas e linguagens de programação cederam espaço para ferramentas cunhadas sob o signo da orientação a objetos e forjadas sob a era das aplicações distribuídas.

Para entender melhor as mudanças que ocorreram nesta área é preciso analisar a evolução da computação de rede e da arquitetura cliente/servidor. As empresas passaram a operar com três níveis diferentes de computação em rede: uma estrutura corporativa baseada em *mainframe*, uma rede de microcomputadores e a *Internet* (CYCLADES..., 1996). Os processos de *downsizing* e a constituição de grupos de trabalho levaram as redes de microcomputadores e à rede corporativa a fundirem-se num único sistema. O uso da *Internet* tornou-se um processo integrado a rede corporativa chamada de *Intranet*, ou seja, a fusão da tradicional rede local com a tecnologia da *Internet*.

Sem dúvida, a melhor linguagem é aquela que satisfaz as necessidades da empresa, contudo não se pode fechar os olhos para o fato de que linguagens criadas há três décadas tenham sido concebidas para um modelo de computação e gerência que não existe mais. A Figura 1 mostra a evolução das linguagens de programação ao longo das décadas.

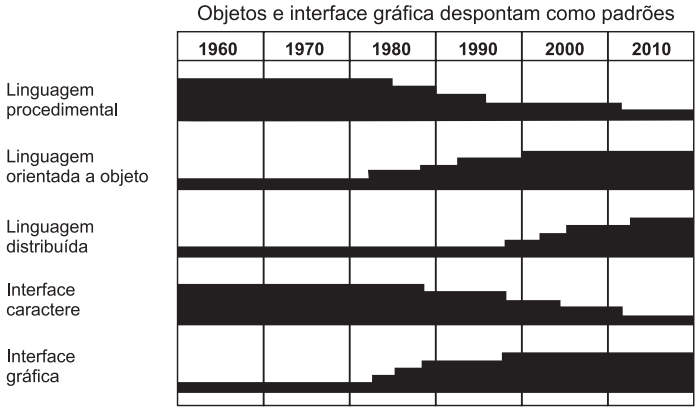


Figura 1 – Evolução das linguagens
Fonte: Ramalho, 1996

Uma Linguagem de Programação é como um meio de comunicação entre o usuário e o computador, ou seja, é através de uma linguagem que expressamos operações e métodos que serão executados pelo computador a fim de resolver um problema.

Existem diferentes formas para representar e relacionar as linguagens de programação. Elas podem ser representadas por níveis, sendo consideradas linguagens de baixo nível e de alto nível. As de baixo nível, podendo também ser chamada de linguagem de máquina, são as linguagens que se comunicam diretamente com o computador através de instruções escritas em binário. Como exemplo pode-se citar a linguagem *Assembly*. Já as de alto nível são linguagens que propiciam um certo nível de afastamento da linguagem de máquina equivalente, por exemplo, à linguagem *Java*.

As Linguagens de Programação podem ser Imperativas, Funcionais, Lógicas e Orientadas a Objetos. A seguir descrevemos algumas características do Paradigma Orientado a Objetos.

2 O PARADIGMA ORIENTADO A OBJETO

É através da Programação Orientada a Objetos (POO) que são desenvolvidos *softwares* mais rápidos e de boa qualidade, tornando-os, assim, mais confiáveis. Um *software* que utiliza Orientação a Objetos é

mais fácil de ser manutenido e de ser entendido porque sua estrutura é desaclopada facilitando futuras modificações (PRESSMAN, 2002).

Para se entender essa programação deve-se saber o significado de Orientação a Objeto. Considere o mundo real onde um objeto seria uma cadeira. Cadeira é uma instância, ou seja, é um elemento de uma classe muito mais ampla chamada móvel. Cada objeto pode estar associado a atributos e métodos de uma classe. Atributos são os dados que descrevem o objeto, ou seja, são as características do objeto. E métodos são os procedimentos ou operações que definem o comportamento do objeto. No exemplo anterior, a classe móvel irá conter atributos como preço, peso, cor e dimensão, entre outros; e métodos como comprar, vender, pintar, reformar. Esses atributos e métodos não só serão aplicados ao objeto cadeira como a qualquer outro objeto instanciado na classe móvel como, por exemplo, mesa, armário, sofá. Portanto, objeto é uma instanciação com atributos e métodos herdados de uma determinada classe. Esse exemplo é representado na Figura 2.

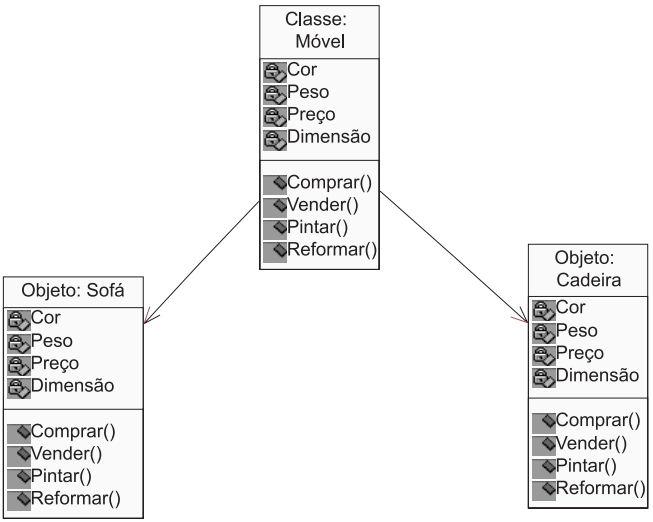


Figura 2 - Herança dos atributos e dos métodos da classe para os objetos

Uma vez definida uma classe, os atributos e os métodos ou operações pertencentes a ela podem ser reusados quando forem surgindo novos

objetos. Essa é uma das características de projeto que leva a um *software* de alta qualidade.

As classes encapsulam os atributos e os métodos necessários para descrever o conteúdo e o comportamento do objeto pertencente a ela. Encapsulamento é o agrupamento de dados e métodos que são aplicados a um objeto escondendo a sua implementação.

Herança é a organização das classes em hierarquia, ou seja, as subclasses herdam o comportamento das superclasses. Uma superclasse é o conjunto de classes e uma subclasse é uma instância de uma classe. Isso implica a existência de uma hierarquia de classes em que os métodos e os atributos da superclasse são herdados pelas subclasses, podendo ser acrescentados novos atributos e métodos privados. Diz-se privados, pois esses atributos e métodos que foram adicionados só fazem parte da subclasse (PRESSMAN, 2002).

Espera-se alcançar com o uso da Orientação a Objetos alguns benefícios como a capacidade de enfrentar novos domínios de aplicação; uso de uma representação básica consistente para a análise e projeto e apoio à reutilização do código, entre outros.

A POO possui suas limitações como qualquer outra linguagem de programação, ou seja, algumas decisões de projetos não são bem tratadas, utilizando o modelo orientado a objeto (PAHLSSON, 2002). Com o interesse de solucionar o que a POO não consegue suportar surgiu a **Programação Orientada a Aspectos** (POA) que será o assunto de maior interesse neste texto.

3 PROGRAMAÇÃO ORIENTADA A ASPECTOS

Hoje em dia a necessidade de desenvolver *software* com qualidade fez com que aumentasse o uso da Orientação a Objetos, buscando, assim, um aumento na produtividade do desenvolvimento e no suporte ao *software*. Isso foi possível através da reusabilidade do código e na facilidade da manutenção do *software*. Como o paradigma Orientado a Objetos possui algumas limitações surgiu, então, a **Programação Orientada a Aspectos** (POA) com o objetivo de solucionar ineficiências da POO.

Com algumas ineficiências têm-se o espalhamento e o entrelaçamento de código que acabam dificultando a manutenção e o desenvolvimento do *software*. A POA trata o problema de espalhamento e entrelaçamento de código aumentando a modulariedade, ou seja, os trechos que se repetem várias vezes ao longo do programa são reduzidos para uma única unidade, no caso os **aspectos**, separando, assim, o código em que são implementadas funções específicas.

Com a modulação diferentes partes do sistema serão afetadas. Essas partes são chamadas *crosscuting concern*, ou preocupações ortogonais. Como exemplos de *crosscuting concern* temos o tratamento de exceções, persistência, distribuição, controle de concorrência e depuração (SOARES, 2002).

Os aspectos são trechos de códigos relacionados a atributos e/ou método do programa que estão repetidos ou distribuídos pelas unidades de códigos, no caso as classes.

A POA tem como função remover esses aspectos do código. Esses aspectos removidos são encapsulados em outras unidades, com isso facilita-se a manutenção do código e o torna muito mais claro ao entendimento. Além disso, a facilidade de modificar os aspectos é muito maior por estarem acoplados ao código, encapsulados em unidades próprias, do que alterar aqueles aspectos que estão distribuídos pelas diversas classes do código (SOARES, 2002).

Uma implementação em POA consiste em uma linguagem de componentes como a linguagem Java, uma linguagem de aspectos e um compilador chamado *weaver*.

O *weaver* é um compilador desenvolvido para ser usado pela POA. É um pouco diferente dos compiladores atuais. Dada a linguagem de componentes e um conjunto de aspectos são gerados pelo *weaver* um código com estes aspectos, por exemplo, o aspecto de distribuição, que permite as alterações necessárias no sistema sem torná-lo distribuído. Para tornar um sistema distribuído utilizando outro protocolo basta implementar outro aspecto com as informações do protocolo requerido e, em seguida, utilizar o *weaver* para gerar o sistema distribuído para o protocolo solicitado (SOARES, 2002).

Existem duas abordagens para a implementação de um sistema orientado a aspectos, o *HyperJ* e o *AspectJ*. O *HyperJ* é uma ferramenta que trabalha com o paradigma de orientação a aspectos, esse é o ponto em comum com o *AspectJ*. Na teoria os objetivos são os mesmos: separar os componentes dos aspectos. Mas o *HyperJ* trabalha de forma diferente, ele separa aspectos e componentes em *Hiper-espaços*. Kiczales (2003) utiliza o termo *Hyperslice* no lugar de *Hiper-espaços*. *Hiper-espaços* seriam fragmentos, ou seja, pequenos blocos de uma hierarquia de classes, enquanto cada classe irá conter subconjunto de atributos e métodos que irão pertencer ao interesse que está sendo modularizado. Já o *AspectJ* é uma extensão da linguagem *Java* em que o principal componente são os aspectos. A seguir descrevemos uma abordagem sobre o *AspectJ*.

4 ASPECTJ

O *AspectJ* é uma extensão da linguagem Java que permite a implementação de aspectos. É um *weaver* que faz a união do programa de componentes com o programa de aspectos, gerando um código fonte. A Figura 3 exibe o esquema do funcionamento de um *weaver*.

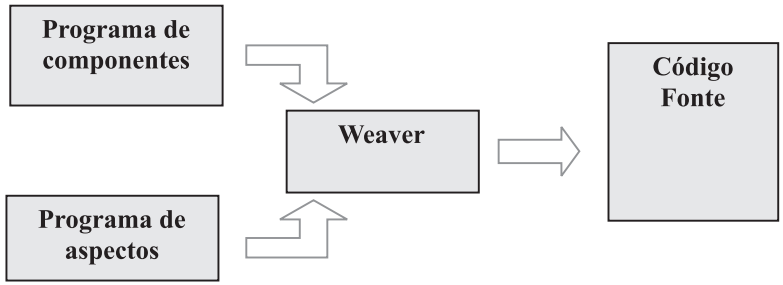


Figura 3 – Funcionamento do weaver

O *AspectJ* possui ferramentas para *debugging* e para fazer a documentação do código (PAHLSSON, 2002). O compilador do *AspectJ*, o *AJC*, produz arquivos-padrão de classes que seguem a especificação do *bytecode* do *Java*. *Bytecode* é o código gerado ao compilar um programa *Java*, ou seja, os arquivos *".class".* Esse é o código universal que pode ser interpretado em várias plataformas por uma máquina Virtual Java

(JMV), o que permite a portabilidade de programas *Java*. O exemplo A faz uma comparação de um programa implementado em *Java* e o exemplo B utilizando aspectos.

```
Package intro;
Import Java.io.*;
Public class HelloWorldA {
    Public static void main (Strings args []) {
        System.out.println ("Hello, World!");
    }
}
```

Exemplo A – Programa implementado em *Java*.

```
Package intro;
Import Java.io.*;
Public aspect HelloWorldA{
    Public static void main (Strings args []) {
        System.out.println ('Hello, World!');
    }
}
```

Exemplo B – Programa utilizando aspectos
Fonte: Kiselev, 2002

Analisando o exemplo acima, percebe-se que a declaração de um aspecto é muito parecida com a declaração de uma classe em *Java*. Os *aspects* são similares a classes por possuírem um tipo, por serem abstratos ou concretos e por possuírem campos, métodos e tipos como membros; e se diferem das classes por possuírem *pointcuts* e *advices* como membros e por poderem acessar membros de outros tipos (PEREIRA, 2002). Em *AspectJ* são os *aspects* que encapsulam os aspectos.

Além da declaração de um *aspect*, podem ser encontrados os *pointcuts*, *joinpoints* e *advices*. Para se entender o que é *pointcut* deve-se saber primeiro o conceito de *joinpoint*. *Joinpoint* é um ponto bem definido em um programa como, por exemplo, uma chamada de métodos, acessos a mem-

bros de uma classe entre outras ações de execução. *Joinpoints* podem conter outros *joinpoints*. Portanto, *pointcuts* são formados pela composição de *joinpoints* através de *wildcards* que são operadores como '&&' (e), '||' (ou) e '!' (não). É através deles que podem ser obtidos os valores dos métodos, dos atributos de um *joinpoint* (SOARES, 2002). O Quadro 1 mostra os designadores que são os que identificam os *joinpoints*.

Call (Assinatura)	*Invocação de método/construtor identificado por* Assinatura.
Execution (Assinatura)	*Execução de método/construtor identificado por* Assinatura.
Get (Assinatura)	*Acesso a atributo identificado por* Assinatura.
Set (Assinatura)	*Atribuição de atributo identificado por* Assinatura.
This (padrão tipo)	*O objeto em execução é instância de* Padrão Tipo.
Target (padrão tipo)	*O objeto de destino é instância de* Padrão Tipo.
Args (padrão tipo,...)	*Os argumentos são instâncias de* Padrão Tipo.
Within (padrão tipo)	*O código em execução está definido em* Padrão Tipo.

Quadro 1 – Designadores de *Pointcut*
Fonte: Soares, 2002

Depois que os pontos a serem logados foram definidos pelos aspectos, os *advices* entram para complementar a implementação. *Advices* são trechos de códigos associados a *pointcuts* que injetam um novo comportamento em todos os *joinpoints* representados pelos *pointcuts* (PEREIRA, 2002). Esses trechos de códigos são executados antes (*before* ();), depois (*after* ();) e simultaneamente (*around* ();) a um *joinpoint*. No Quadro 2 estão representados os tipos de *advices*.

Before	*Executa quando o* joinpoint *é alcançado, mas imediatamente antes da sua computação.*
after returning	*Executa após a computação com sucesso do* joinpoint.
after throwing	*Executa após a computação sem sucesso do* joinpoint.
After	*Executa após a computação do* joinpoint*, em qualquer situação.*
Around	*Executa quando o* joinpoint *é alcançado e tem total controle sobre a sua computação.*

Quadro 2 – Tipos de *Advice*
Fonte: Soares, 2002.

O exemplo C mostra um programa utilizando *aspects*, *pointcuts*, *joinpoints* e *advices*.

```
Public class Hello {.
    Public static void main (String args []) {
        System.out.println (“Hello world!”);
    }
}
Public aspect Greetings {
    pointcut pc() : call(* *main(..));
    before() : pc() {
        System.out.println (“Hi.”);
    }
    after() : pc() {
        System.out.println (“Bye...”);
    }
}
```

Exemplo C – Programa utilizando *aspects*, *pointcuts*, *joinpoints* e *advices*
Fonte: Peeira, 2002

Nota-se que existe uma classe declarada como *Hello* e um *aspect* declarado como *Greetings*. Utiliza-se o *AJC* para compilar o código, verificando, assim, se existem erros ou não. Depois que o código é compilado deve-se executá-lo, chama-se a classe que no caso seria *Hello.java*, o nome da classe principal do programa. O código será executado da seguinte maneira: dentro do *aspect Greetings* existe um *pointcut* chamado *“pc ()”* com um designador *call* que está fazendo uma chamada a um método, ou seja, o *call* está chamando a função *main* declarada dentro da classe *Hello*. Como a função *main* tem como método imprimir uma mensagem, a mensagem impressa na tela será “*Hello world!”*. Logo após a declaração do *pointcut* existe uma implementação de um *advice* que neste caso é o *before*. Portanto a mensagem da função main só será impressa depois que a mensagem do *advice “before”* for impressa que neste caso é *“Hi”*. Por enquanto, tem-se a seguinte execução:

> Hi.

> Hello world!

Seguindo o código, encontra-se uma outra implementação de um *advice*. Esta implementação significa que só será impressa a mensagem

"Bye..." depois que a mensagem da função *main* for impressa. Portanto, o código final executado será:

> Hi.

> Hello world!

> Bye...

Este código final foi gerado através do conceito da Orientação a Aspectos. O exemplo D implementa este mesmo código utilizando o conceito da Orientação a Objetos.

```
Public class Hello
{
    Public static void main (String  args [ ])
    {
        Hi.imprime ();
        System.out.println ("Hello World!");
        Bye.imprime ();
    }
}

Class Hi
{
    Public static void imprime ()
    {
        System.out.println ("Hi.");
    }
}

Class Bye
{
    public static void imprime ()
    {
        System.out.println ("By…e");
    }
}
```

Exemplo D – Programa que utiliza o conceito da Orientação a Objetos

Fazendo uma comparação entre essas duas implementações, pode-se perceber que, com a utilização do conceito relacionado aos aspectos, o código ficou mais modularizado e evita a declaração de várias classes e a repetição de código.

5 ESTUDOS FUTUROS

Como continuidade do estudo apresentado neste artigo, será desenvolvido um trabalho comparativo entre a POO e a POA, ou seja, através de programas implementados em *Java* e em *AspectJ* será feita uma comparação entre códigos-fonte de forma a mensurar as reais vantagens e desvantagens da POA sobre a POO, abordando-se modelos de complexidade variada.

Os programas serão elaborados no *AspectJ* na versão 1.1rc1 e na versão jdk 1.3 do *Java*.

6 CONCLUSÃO

A POA surgiu para tentar solucionar as ineficiências da Programação Orientada a Objetos, assim como a Programação Orientada a Objetos surgiu para solucionar as ineficiências das Programações Procedurais e Funcionais.

Como vantagens da POA tem-se o tratamento do entrelaçamento e do espalhamento de código e dos *crosscuting concerns* através da modularização, facilitando, assim, o entendimento e a manutenção do código. Essas ineficiências dificultam na reusabilidade e na manutenção do código tornando baixa a qualidade do programa. São através dos *aspects* que estas ineficiências são solucionadas, ou seja, os aspectos são os *crosscuting concerns* que afetam o sistema de forma que não podem ser encapsulados com as técnicas de Orientação a Objetos tradicionais. Os *crosscuting concerns* quando são implementados acabam ficando espalhados pelo código, fazendo com que as classes encapsulem mais de um *concern* ao mesmo tempo, resultando, assim, no entrelaçamento de código. Estes *concerns* espalhados são os aspectos que na POA serão encapsulados pelos *aspects* onde se acredita que podem ser incluídos os métodos, os atributos e as regras.

Por se tratar de uma técnica recente de programação, há dificuldade de se encontrarem documentos e uma metodologia sobre a POA. Ainda existem detalhes que precisam ser mais bem estudados, como é o caso da política do tratamento de exceções (SOARES, 2002).

A linguagem *AspectJ* é uma extensão da linguagem *Java* que permite a utilização dos aspectos em *Java*. Como desvantagens têm-se as necessidades de saber utilizar com precisão esta nova ferramenta e a deficiência relacionada ao ambiente de desenvolvimento (SOARES, 2002).

No *AspectJ,* a definição do *pointcut* requer a identificação dos *joinpoints* que são identificados através de nomes e tipos de métodos, entre outros. Com isso os aspectos irão depender do sistema, dificultando, assim, a reusabilidade (SOARES, 2002).

A maior vantagem do *AspectJ* é o fato da implementação das funcionalidades ficarem separadas da parte funcional do sistema, permitindo a inserção e a retirada de tais aspectos. O fato de inserir ou retirar um aspecto não quer dizer que o comportamento do sistema será modificado. Através da separação, ou seja, da modularização, o código funcional será mais fácil de ser entendido, pois não irá conter códigos de propósitos diferentes entrelaçados entre si com o código funcional, obtendo, assim, um melhor desenvolvimento do sistema (SOARES, 2002).

Não se pode considerar a Programação Orientada a Aspectos um novo paradigma, uma vez que estudos sobre essa técnica de programação ainda necessitam ser realizados.

A SYSTEMIZED STUDY ABOUT ASPECT-ORIENTED PROGRAMMING

ABSTRACT

Studies carried through until had today shown the existence of many related problems to the programming that nor the Objects-Oriented Programming and nor the Structuralized Programming had been enough to solve them. With this Aspects-Oriented Programming (AOP) appeared to came to solve problems such as tangling and scattering of code through

the definition of aspects. With the AOP the code if becomes more easy of being understood and being conceded a mandate of maintenance to, allowing to considerable profits with relation to the quality of software and productivity in the development. This work presents a systemized study on the AOP.

Keywords: Aspects. Aspectj. Aspect-Oriented. Programming. Separation of concerns.

REFERÊNCIAS

CYCLADES CORPORATION. **Guia Internet de conectividade***.* São Paulo: Cyclades Informática, 1996.

KICZALES, G. **Programação Orientada a Aspectos**. Disponível em: <http://www. dc.ufscar.br/~rar/aop.htm>. Acesso em: 18 mar. 2003.

KISELEV, I. **Aspect**: Oriented Programming with AspectJ**.** Indianapolis: Sams, 2002.

PAHLSSON, N. **Aspect-Oriented Programming**: an introduction to Aspect-Oriented Programming and AspectJ. 2002. Disponível em: <http:www.bluefish.se/aop/aop_ niklas_ pahlsson.pdf>. Acesso em: 7 maio 2003.

PEREIRA, S. **AspectJ**. 2002. Disponível em: <http://www.ime.uso.br/~kon/MAC5715/ aulas/slides/AspectJ.pdf>. Acesso em: 7 maio 2003.
PRESSMAN, R. **Engenharia de Software**. Rio de Janeiro: McGraw-Hill, 2002.

RAMALHO, J. A. A caminho da linguagem distribuída. **Informática Exame**, n. 123, p. 94-98, jun. 1996.

SOARES, S.; BORBA, P. **AspectJ**: Programação Orientada a Aspectos em Java. 2002. Disponível em: <http://www.cin.ufpe.br/~scbs/artigos/AspectJ_SBLP2002.pdf>. Acesso em: 7 maio 2003.

ESPERMATOGÊNESE EM MAMÍFEROS

SCIENTIA

DEILER SAMPAIO COSTA[1]
TARCÍZIO ANTÔNIO REGO DE PAULA[2]

[1] Doutor em Reprodução Animal pela Universidade Federal de Minas Gerais. Professor do Centro Universitário Vila Velha. E-mail: deiler@uvv.br.

[2] Doutor em Morfologia pela Universidade Federal de Minas Gerais. Professor da Universidade Federal de Viçosa. E-mail: tarcizio@mail.ufv.br.

Scientia Vila Velha (ES), v. 4, n. 1/2, **p. 53-72**, jan/dez. 2003

RESUMO

Aborda a organização da espermatogênese em mamíferos, destacando as fases mitótica, meiótica e espermiogênese. Faz, ainda, um apanhado dos conceitos de vários autores sobre o ciclo do epitélio seminífero, a onda do epitélio seminífero e sobre os estádios do ciclo do epitélio seminífero. Os tipos celulares, tais como espermatogônias, espermatócitos, espermátides, células de Sertoli e espermato-zóides, são caracterizados quanto à sua morfologia e funções. Enfim, aborda a constituição e morfologia do compartimento intertubular, onde as células de Leydig são destacadas, dada sua importância fisiológica.

Palavras-chave: Túbulo seminífero. Espermatogênese. Mamíferos.

1 INTRODUÇÃO

1.1 A ORGANIZAÇÃO DA ESPERMATOGÊNESE

A espermatogênese é um processo sincrônico e regular de diferenciação celular, pelo qual uma espermatogônia tronco é gradativamente diferenciada numa célula haplóide altamente especializada, o espermatozóide. Este processo que ocorre nos túbulos seminíferos dura em torno de 40 a 60 dias na maioria dos mamíferos estudados (FRANÇA; RUSSELL, 1998; JOHNSON, 1991).

Esta diferenciação envolve três classes de células germinativas: as espermatogônias, os espermatócitos e as espermátides. Nos adultos a espermatogênese é um processo contínuo que pode ser dividido em três fases distintas: a mitótica, a meiótica e a espermiogênese, cada uma caracterizada por mudanças morfológicas e bioquímicas dos componentes do citoplasma e núcleo celular (COUROT; HOCHEREAU-DE-REVIERS; ORTAVANT, 1970).

Durante a fase mitótica, também chamada de espermatogonial ou proliferativa, as células tronco (A_0) dão origem a duas outras, uma vai servir para a renovação da população de células tronco e a outra entra no processo espermatogênico. Esta última, por sua vez vai dar origem à espermatogônia intermediária, que se divide para formar as espermatogônias B. No final da divisão, as espermatogônias B, por sua vez, dão origem a outras células que entram numa prolongada fase de meiose como espermatóci-

tos em pré-leptóteno. Mais tarde tais células serão assimiladas dentro do compartimento ad-luminal do epitélio seminífero pela passagem através das junções entre células de Sertoli adjacentes (RUSSELL, 1980). Todos os tipos celulares subseqüentes vão ficar no compartimento ad-luminal. Neste compartimento as células estão isoladas e não têm acesso direto a nutrientes e hormônios, sendo dependentes das células de Sertoli para prover seus requerimentos (RUSSELL; GRISWOLD, 1993).

A fase meiótica envolve síntese de DNA nos espermatócitos em pré-leptóteno, síntese de RNA em espermatócitos em paquíteno e no final da meiose, quando vai ocorrer a divisão reducional, estes últimos geram as espermátides haplóides (PARVINEN et al., 1991).

Na terceira fase da espermatogênese, as espermátides se diferenciam através de uma série de modificações morfológicas progressivas em espermatozóides, processo usualmente conhecido como espermiogênese. Estas modificações incluem desenvolvimento do acrossoma, condensação e alongamento do núcleo e formação da cauda espermática. Durante o desenvolvimento do acrossoma notam-se quatro fases distintas: a Fase de Golgi, de Capuchão, do Acrossoma e de Maturação (RUSSELL et al., 1990). Apesar de as características gerais do desenvolvimento do acrossoma serem semelhantes entre os mamíferos, uma espécie difere das outras nos detalhes de sua formação e na sua forma final sobre a cabeça do espermatozóide.

Durante a condensação nuclear as histonas dentro da cromatina nuclear são substituídas por proteínas de transição que, subseqüentemente, darão lugar às protaminas (HECHT, 1990). Desta forma há uma remodelagem das fibras de cromatina que passam a ser compactas e podem ser colocadas paralelamente (WARD; COFFEY, 1991). Estas alterações ocorrem para que haja a conversão de um núcleo que pode sofrer transcrição em um núcleo inativo para este processo, o que é característico dos espermatozóides (HECHT, 1990).

O alongamento das espermátides é iniciado quase no mesmo tempo em que a condensação nuclear. Este período é caracterizado pela formação do flagelo e acúmulo de mitocôndrias. Associado a estes eventos, ocorre uma redução progressiva do volume citoplasmático, caracterizado pela eliminação de água (SPRANDO; RUSSELL, 1987), formando um corpo residual de citoplasma que é fagocitado pelas células de Sertoli e transportado para sua base, onde será degradado pelos

lisossomos. Finalmente, as espermátides maduras serão liberadas para o lume tubular, passando então a se chamar espermatozóides, marcando, assim, o final da espermatogênese.

O rendimento da espermatogênese não é de 100%, ou seja, durante o processo vão ocorrer várias perdas celulares, de forma que uma espermatogônia A_1 não vai gerar 16 espermatócitos primários e 64 espermátides, como teoricamente seria esperado, em se tratando de animais com quatro gerações de espermatogônias (COUROT; HOCHEREAU-DE-REVIERS; ORTAVANT, 1970; CLERMONT, 1972). Isso se deve ao fato de ocorrerem apoptose e degeneração das células germinativas. Tais alterações são observadas normalmente nos túbulos seminíferos com a mesma regularidade que são encontradas as divisões e diferenciações celulares (ROOSEN-RUNGE, 1955). Entretanto, segundo Amann (1962), parece não ocorrerem perdas apreciáveis de espermátides durante a espermiogênese, ou seja, uma espermátide normalmente vai gerar um espermatozóide.

2 O CICLO DO EPITÉLIO SEMINÍFERO

As células espermatogênicas se encontram arranjadas nos túbulos seminíferos de forma organizada e bem definida, constituindo associações celulares que caracterizam os estádios do ciclo do epitélio seminífero (Figura 1). Na maioria dos mamíferos estudados o arranjo dos estádios do ciclo do epitélio seminífero é segmentado e normalmente existe apenas um estádio por secção transversal de túbulo (RUSSELL et al., 1990). O ciclo do epitélio seminífero em um dado segmento pode ser definido como o período do desaparecimento de um determinado estádio até o seu reaparecimento neste mesmo segmento (LEBLOND; CLERMONT, 1952).

A duração do ciclo do epitélio seminífero é geralmente constante para uma determinada espécie, variando, no entanto, de uma espécie para a outra (AMANN, 1970; RUSSELL et al.. 1990). Este intervalo de tempo dura 13,5 dias no touro (HOCHEREAU-DE-RIVERS; COUROT; ORTAVANT, 1964); 8,6 dias no varrão (SWIERSTRA, 1968); 10,3 dias no carneiro (ORTAVANT, 1959); 11,9 dias na capivara (PAULA, 1999); 12,2 dias no garanhão (SWIERSTRA; GEBAUER; PICKETTI, 1974); 8,9 dias no camundongo (OAKBERG, 1956); 12,9 dias no rato (CLERMONT; LEBLOND; MESSIER, 1959) e 13,6 dias no cão (FOOTE; SWIERSTRA; HUNT, 1972).

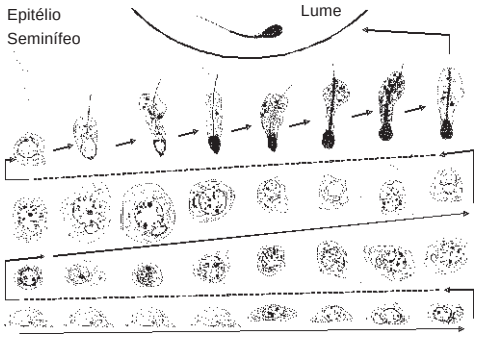


Figura 1 – Ciclo do Epitélio Seminífero

Usualmente, quatro a cinco ciclos são necessários para que o processo espermatogênico se complete, ou seja, para que haja a liberação dos espermatozóides no lume do túbulo seminífero a partir de uma espermatogônia A_1 (AMANN; SCHANBACHER, 1983; FRANÇA; RUSSELL, 1998).

3 A ONDA DO EPITÉLIO SEMINÍFERO

Além da organização seqüencial em um dado segmento, os estádios também apresentam certa seqüência ao longo da extensão do túbulo seminífero, ou seja, usualmente um determinado estádio está em posição contígua a um segmento em estádio subseqüente. Esta disposição seqüencial de estádios ao longo do túbulo denomina-se onda do epitélio seminífero (CURTIS, 1918; PEREY; CLERMONT; LEBLOND, 1961).

A origem deste evento é desconhecida, mas parece ser resultado de uma sincrônica, mas não simultânea divisão de espermatogônias tronco em segmentos tubulares adjacentes. Segundo Johnson (1991) as funções desta onda são: assegurar uma liberação constante de espermatozóides; reduzir a competição por hormônios e metabólicos usados em um dado estádio; reduzir a congestão que poderia ocorrer ao longo do túbulo se a espermiação ocorresse simultaneamente; assegurar o fluxo constante de fluido do túbulo seminífero, mantendo o veículo para o transporte de espermatozóides e hormônios utilizados pelo epitélio do epidídimo; e facilitar a maturação dos espermatozóides no epidídimo por um fluxo constante de espermatozóides e fluidos vindos do testículo.

Usualmente a seqüência se inicia com os estádios menos avançados no meio de uma alça de túbulo e progressivamente os mais evoluídos estão mais próximos à rete testis. Percebem-se, ainda, algumas interrupções na seqüência dos estádios, o que alguns autores chamam de modulações *(RUSSELL et al., 1990).*

4 ESTÁDIOS DO CICLO DO EPITÉLIO SEMINÍFERO

Considera-se uma associação celular ou estádio do ciclo do epitélio seminífero como um conjunto definido de gerações de células germinativas encontrado, em determinado momento, num túbulo seminífero seccionado transversalmente (CASTRO; BERNDTSON; CARDOSO, 1997). Uma vez que as gerações de células espermatogênicas desenvolvem-se sincronicamente em estreita relação uma com as outras, ao longo do tempo em uma determinada secção transversal de túbulo seminífero, há uma mudança constante e progressiva de estádios, formando o ciclo do epitélio seminífero.

A identificação dos diferentes estádios do ciclo do epitélio seminífero é essencial para a realização de estudos quantitativos da espermatogênese, sendo importante para a compreensão da espermatogênese normal, bem como para a determinação de fases específicas do processo que possam ser afetadas por um determinado tratamento ou droga (BERNDTSON, 1977).

Duas principais metodologias têm sido empregadas para o estudo dos estádios do ciclo do epitélio seminífero de mamíferos: o primeiro é denominado método da morfologia tubular e se baseia nas alterações de forma do núcleo das células espermatogênicas, na ocorrência de divisões meióticas e no arranjo das espermátides no epitélio seminífero (Figura 2). Este método permite a obtenção de oito estádios do ciclo para todas as espécies (BERNDTSON, 1977; ORTAVANT; COUROT; HOCHEREAU-DE-REVIERS, 1977).

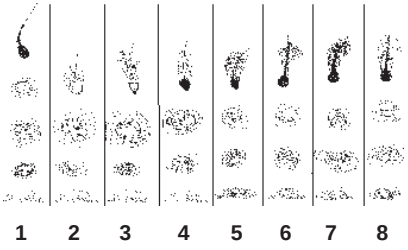


Figura 2 – Estádios do Ciclo do Epitélio Seminífero

O segundo, conhecido como método do sistema acrossômico, baseia-se nas alterações do sistema acrossômico e na morfologia das espermátides em desenvolvimento. Com este método, o número de estádios varia de uma espécie para a outra, girando em torno de 10 a 16 na maioria dos animais (FRANÇA; RUSSELL, 1998; RUSSELL et al., 1990). Isso se deve ao fato de que apesar das características gerais da espermiogênese serem semelhantes entre as espécies, vão existir diferenças nos detalhes do desenvolvimento do acrossoma entre elas.

A freqüência relativa em que os estádios do ciclo do epitélio seminífero são encontrados é um parâmetro de importância fundamental em estudos da espermatogênese e reflete a duração absoluta de cada estádio. O conhecimento da freqüência dos estádios, além de ser essencial para se estimar a duração do ciclo do epitélio seminífero, é muito útil para se monitorar a duração dos efeitos de agentes lesivos ou drogas sobre a espermatogênese (CASTRO; BERNDTSON; CARDOSO, 1997).

Há evidências conclusivas de que a freqüência média dos estádios do ciclo do epitélio seminífero difere entre espécies, mas é um parâmetro relativamente constante entre indivíduos de uma mesma espécie, mesmo quando se utilizam diferentes métodos para sua identificação (CLERMONT, 1972; HESS et al., 1990).

As células de Sertoli

As células de Sertoli localizam-se junto à lâmina basal dos túbulos seminíferos, e seu citoplasma envolve as células germinativas, estendendo-se até o lume tubular. Tais células estão presentes em todos os estádios do ciclo do epitélio seminífero (BILASPURI; GURAYA, 1984) e

dentre as funções que desempenha, podem-se destacar: (1) formação da barreira hematotesticular, (2) suporte estrutural e nutricional das células germinativas, (3) responsável pela progressão das células germinativas, em diferenciação, em direção ao lume, (4) eliminação dos espermatozóides para o lume tubular, processo conhecido como espermiação, (5) fagocitose de células germinativas degeneradas e corpos residuais de citoplasma de espermátides maduras, (6) secreção de fluidos e proteínas para banhar as células germinativas em desenvolvimento e conduzir os espermatozóides através dos túbulos em direção à rete testis *(RUSSELL; GRISWOLD, 1993).*

As junções entre células de Sertoli adjacentes formam a barreira hematotesticular, que divide funcionalmente o epitélio seminífero em dois compartimentos: o basal e o ad-luminal. A renovação e proliferação das espermatogônias até espermatócito primário em pré-leptóteno ocorrem no compartimento basal, ou seja, abaixo das junções entre células de Sertoli adjacentes. No início da fase de leptóteno da primeira divisão meiótica, os espermatócitos primários migram através da barreira hematotesticular para o compartimento ad-luminal, onde a meiose continua e a espermiogênese ocorre (DYM, 1973; RUSSELL, 1977). Obviamente esta passagem através da barreira ocorre sem que haja perda da sua integridade.

As células do compartimento basal recebem suprimento hormonal diretamente da rede de capilares sangüíneos ou fluido intertubular, enquanto aquelas que se encontram no compartimento ad-luminal o fazem através das células de Sertoli que estabelecem junções especiais com estas células (GUARAYA, 1987).

A eficiência da espermatogênese tem sido avaliada a partir da estimativa da população de células germinativas em relação à população de células de Sertoli (BERNDTSON; IGBOELI; PICKETT, 1987; JOHNSON et al., 2000). Isto se torna possível, pois a maioria dos autores considera que as células de sertoli são numericamente estáveis após a puberdade, não sofrendo mais mitoses (COUROT; HOCHEREAU-DE-REVIERS; ORTAVANT, 1970) e por estas células serem menos susceptíveis a agentes deletérios quando comparadas às células germinativas (KUMI-DIAKA et al., 1983). A eficiência da espermatogênese aumenta progressivamente a partir da puberdade até ocorrer a maturidade sexual, quando a razão entre células de sertoli e células espermatogênicas se estabiliza.

Contrastando com a maioria dos autores que afirmam que o número de células de sertoli é estável após a puberdade, França e Russell (1998); Johnson e Nguyen (1983); e Steinberger e Steinberger (1971), trabalhando com 186 garanhões, relataram que durante a estação reprodutiva, estes animais tinham maior número de células de sertoli que fora da estação. Tais autores ressaltam ainda que valores intermediários foram encontrados no período entre as duas estações. Segundo Johnson e Tatum (1989) o fotoperíodo, que é responsável pelas mudanças sazonais na concentração de hormônios sexuais no garanhão (CLAY et al., 1988), provavelmente é responsável pelas mudanças sazonais no número de células testiculares, incluindo as células de Sértoli.

O número de células de Sértoli por testículo é o principal fator para a determinação da produção espermática diária e do tamanho dos testículos (HESS et al., 1993; ORTH; GUNSALUS; LAMPERTI, 1988). Isso se deve ao fato de que as células de sertoli têm uma capacidade de suporte de células germinativas relativamente fixa para cada espécie. Assim, o número de células germinativas suportadas por uma única célula de sertoli (índice de célula de sertoli) é o melhor indicativo da sua eficiência funcional (RUSSELL; PETERSON, 1984).

As espermatogônias

As espermatogônias usualmente sofrem divisões mitóticas incompletas (Figura 3), ou seja, as células originárias de uma divisão permanecem interligadas àquela que deu origem através de pontes citoplasmáticas (DE ROOIJ; GOOTEGOED, 1998). Com base na morfologia; diâmetro e volume do núcleo; número de nucléolos por núcleo; posição topográfica em relação a outras células e à lâmina basal; e disposição dos cromossomos durante a divisão, pode-se identificar basicamente três tipos de espermatogônias nos mamíferos: a espermatogônia tipo A, a espermatogônia intermediária e a espermatogônia tipo B (BERNDTSON; DESJARDINS, 1974; CLERMONT; ANTAR, 1973).

Figura 3 – Espermatogônias durante a divisão mitótica incompleta destacando as pontes citoplasmáticas

O processo de divisões espermatogoniais ainda é um dos aspectos mais complexos e controversos nos estudos da cinética da espermatogênese de mamíferos, já que em boa parte das espécies estudadas, o padrão de multiplicação e renovação de espermatogônias ainda não está inteiramente elucidado (CASTRO; BERNDTSON; CARDOSO, 1997). O número estimado de gerações de espermatogônias varia de quatro a seis na maioria das espécies estudadas (CLERMONT, 1972), entre as quais podem-se citar seis gerações no touro, no carneiro e no cão e cinco gerações no varrão, no coelho e no cavalo (FRANÇA; RUSSELL, 1998).

Em ratos podem-se distinguir espermatogônias A_s ("single"), A_{pr} ("paired") e A_{al} ("aligned") de acordo com o arranjo topográfico delas sobre a membrana basal (HUCKINS, 1971). As espermatogônias A_s são as células tronco da espermatogênese. Por divisão mitótica, normalmente metade destas células dá origem a espermatogônias A_{pr} e a outra metade vai constituir a população de renovação de espermatogônias A_s. (OAKBERG, 1971). As espermatogônias A_{pr} dividem-se outras vezes para formar quatro, oito ou 16 espermatogônias A_{al}. Estas, por sua vez, se diferenciam em espermatogônias A_1, que são a primeira geração de espermatogônias diferenciadas. A espermatogônia A_1 se diferencia em uma série de seis divisões, passando por espermatogônias A_2, A_3, A_4, Intermediária (In) e finalmente espermatogônia B, que vai dar origem aos espermatócitos primários (DE ROOIJ; GOOTEGOED, 1998).

As espermatogônias In e B também podem ser subdivididas em algumas espécies. Nos coelhos, por exemplo, observam-se duas gerações de espermatogônias intermediária (In_1 e In_2), nos ruminantes, com base na coloração do núcleo, distinguem-se dois tipos de espermatogônias B (B_1 e B_2) Já em macacos Cercopithaecus quatro gerações desta espermatogônia, B_1 a B_4, podem ser identificadas *(FRANÇA; RUSSELL, 1998; GURAYA, 1987).*

Os espermatócitos

As células da linhagem germinativa em meiose são chamadas espermatócitos (Figura 4) e são formadas pela última divisão espermatogonial. A meiose compreende duas divisões, as células da primeira são chamadas de espermatócitos primários e as da segunda, espermatócitos secundários.

Logo depois de sua formação, o núcleo dos espermatócitos primários é muito semelhante ao de sua célula progenitora, a espermatogônia B. Da mesma forma que há uma variação na configuração da cromatina da espermatogônia B, entre as espécies, também vai haver uma variação correspondente na morfologia nuclear dos espermatócitos (CLERMONT, 1972).

Próximo ao final da interfase, os grânulos de cromatina se organizam e tornam-se filamentosos, assumindo a configuração morfológica da fase de leptóteno da prófase da primeira divisão meiótica. A partir desta fase, a morfologia do núcleo dos espermatócitos é muito semelhante entre animais de espécies diferentes (CLERMONT, 1972; GURAYA, 1987). Depois da fase de leptóteno os espermatócitos primários entram em zigóteno, período durante o qual há pareamento dos cromossomos homólogos. Então o volume nuclear aumenta progressivamente e os cromossomos tornam-se mais compactos e entram na fase de paquíteno. Isto é seguido por uma curta fase de diplóteno durante a qual os cromossomos se separam parcialmente. Finalmente o núcleo passa por metáfase, anáfase e telófase da primeira divisão, para produzir os espermatócitos secundários. O núcleo em interfase dos espermatócitos secundários é menor que do espermatócito primário em paquíteno. Estas células têm uma vida curta e, sem que haja duplicação do DNA, entram na segunda divisão meiótica, resultando na formação de espermátides haplóides. (CLERMONT, 1972; GURAYA, 1987).

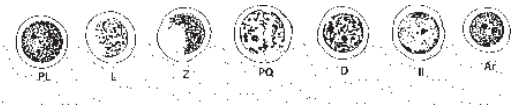


Figura 4 – Espermatócitos primários

PL – pré-leptóteno L – leptóteno Z – zigóteno PQ – paquíteno
D – diplóteno II – espermatócito secundário Ar – espermátide arredondada

As espermátides e a espermiogênese

Depois que as espermátides arredondadas são formadas, uma série de mudanças morfológicas, histoquímicas e bioquímicas vão ocorrer em seu núcleo e componentes do citoplasma (GURAYA, 1987). Tais modificações vão culminar com a formação de células haplóides altamente diferenciadas e especializadas que são os espermatozóides.

As espermátides são caracterizadas por um núcleo pequeno e esférico e um citoplasma que apresenta organelas dispostas em zonas bem definidas. A zona de Gole, que é visível próximo ao núcleo, começa a elaborar pequenos corpúsculos denominados grânulos pró-acrossomicos PAS positivos, que irão coalescer para formar um único grânulo grande, o grânulo acrossômico, que irá se aderir à membrana do núcleo. De sua formação até o período em que aparece o grânulo acrossômico associado à membrana nuclear, as espermátides estão, como se diz, em fase de golgi (CLERMONT; LEBLOND, 1955; LEBLOND; CLERMONT, 1952).

Subseqüentemente, começa a fase de capuchão, que se caracteriza pela expansão do grânulo acrossômico sobre a superfície nuclear. Este processo continua até que cerca de dois terços do núcleo sejam cobertos (BURGOS; FAWCETT, 1955). Enquanto o capuz acrossômico se forma num pólo do núcleo, os centríolos migram para o pólo oposto junto à membrana nuclear onde se dará a formação do axonema e, em seguida, a formação da cauda.

Durante a fase do acrossoma há uma rotação do núcleo e do sistema acrossômico que se orienta em direção à membrana do túbulo seminífero. Essa rotação é acompanhada pelo deslocamento do núcleo para a periferia do citoplasma. Depois desse deslocamento a cromatina nuclear começa a condensar e o núcleo torna-se mais alongado e achatado. O acrossoma que está intimamente aderido ao núcleo, também se condensa e se alonga. Tais modificações são ligeiramente diferentes para cada espécie, de modo que resultam em espermátides alongadas de características específicas (FAWCETT, 1970). Junto com as modificações de morfologia nuclear, vai haver o deslocamento do citoplasma para o pólo oposto da célula, onde irá circundar a porção proximal da cauda em desenvolvimento.

A fase de maturação é aquela em que as diferenciações morfológicas para a formação dos espermatozóides se completam. Durante esta fase o núcleo completa sua condensação e assume sua forma definitiva. Uma bainha com nove fibras forma-se ao redor do axonema. As mitocôndrias, previamente concentradas ao redor do axonema, ordenam-se ao longo da peça intermediária formando a bainha mitocondrial. Grande parte do citoplasma que era visto ao longo do flagelo se desloca em direção ao núcleo e se destaca da célula formando o corpo residual que é assimilado pela célula de sertoli. Ocorrendo em seguida a

espermiação, que é a liberação das espermátides maduras (espermatozóides) para o lume tubular (LEBLOND; CLERMONT, 1952).

Os espermatozóides

Apesar de a liberação dos espermatozóides no lume dos túbulos seminíferos marcar o final da espermatogênese, tal célula ainda não está pronta para fecundar o óvulo, precisando sofrer ainda uma série de modificações que vão desde aquelas que ocorrem durante o trânsito epididimário (maturação espermática) até aquelas que vão se dar no trato reprodutivo da fêmea (capacitação espermática).

Morfologicamente os espermatozóides podem ser divididos em duas estruturas distintas: a cabeça e a cauda. A cabeça, cuja forma, tamanho e estrutura variam enormemente entre as espécies, contém duas maiores regiões: a região acrossomal e a região pós-acrossomal. Segundo Barth e Oko (1989), podem-se distinguir cinco membranas na cabeça do espermatozóide: a plasmalema, a membrana acrossomal interna e externa e a membrana nuclear interna e externa.

O acrossoma contém glicoproteínas secretadas pelo retículo endoplasmático e complexo de Golgi, que são as enzimas responsáveis pela penetração do espermatozóide no óvulo. Tal estrutura também tem sido referida como um lisossomo especializado com função específica de dispersar e dissolver a corona radiata e zona pelúcida do óvulo. Entre outras hidrolases, o acrossoma contém: fosfatase ácida, arilsulfatase, b-N-acetilglicosaminase, fosfolipase, esterase, hialuronidase e acrosina (ALLISON; HARTREE, 1970; STAMBAUGH; BUCKLEY, 1969; STRIVASTAVA; ADAMS; HARTREE, 1965). As duas últimas são as mais importantes na penetração do ovócito. Uma distribuição ordenada destas enzimas dentro do acrossoma é essencial para que a penetração ocorra *(ROGER; YANAGIMACHI, 1975).*

A cauda dos espermatozóides é constituída de componentes funcionalmente dependentes que são formados em diferentes períodos do processo de espermiogênese. Apesar de existirem variações de forma e tamanho dos componentes nas diferentes espécies, em geral, a organização estrutural da cauda é a mesma. A cauda dos espermatozóides pode ser dividida anatomicamente em quatro discretas regiões: o colo ou peça de conexão, a peça intermediária, a peça principal e a peça

terminal, todas envolvidas por uma mesma membrana (BARTH; OKO, 1989).

O colo forma uma lâmina basal que interliga a cabeça à peça intermediária. Esta região é muito frágil e contém uma estrutura complexa de onde partem as nove fibras densas que se projetam por quase toda a extensão da cauda (BEDFORD; HOSKINS, 1990).

A peça intermediária é a região localizada entre o colo e o ânulus. É caracterizada pela presença de um grande número de mitocôndrias arranjadas de forma espiralada (bainha mitocondrial) e pelo axonema, que consiste em nove pares de microtúbulos dispostos radialmente ao redor de dois filamentos centrais (BEDFORD; HOSKINS, 1990). O ânulus é uma estrutura eletrodensa que interliga a peça intermediária à peça principal, onde a membrana plasmática é firmemente fixada.

A peça principal localiza-se posteriormente ao ânulos, estende-se até as proximidades da extremidade da cauda e é composta pelo axonema circundado por uma resistente bainha fibrosa. Finalmente, a peça terminal é constituída apenas pelo axonema recoberto pela membrana plasmática (EDDY, 1988).

5 CONSIDERAÇÕES FINAIS

A compreensão do processo espermatogênico é fundamental para o estabelecimento de técnicas para se otimizar a utilização de um reprodutor. Informações como a freqüência de coletas de sêmen, número de fêmeas por reprodutor e até mesmo a viabilidade de biotécnicas avançadas como transplante de espermatogônias, estão sustentadas no conhecimento básico da espermatogênese em cada espécie. Apesar de já existirem estudos sobre as características morfofuncionais da espermatogênese dos animais domésticos, muitos detalhes ainda precisam ser elucidados. Por outro lado, o estudo da espermatogênese em animais selvagens ainda é incipiente e poucas pesquisas têm sido realizadas com estes animais. O estudo da espermatogênese é um campo aberto para pesquisas e várias perguntas ainda precisam ser respondidas.

ABSTRACT

The aim of this paper is to review the spermatogenesis organization, emphasizing the mitotic, meiotic and spermiogenesis phases. The concepts of many researchers about seminiferous epithelium cycle, seminiferous epithelium wave and the stages of the seminiferous epithelium cycle were reviewed. The spermatogonias, spermatocytes, spermatides, Sertoli cells and spermatozoids were described. Finally, the constitution and morphology of intertubule compartment were correlated, emphasizing the Leydig cell functions due to its importance.

Keywords: Seminiferous tubules. Spermatogenesis. Mammalian.

REFERÊNCIAS

ALLISON, A. C.; HARTREE, E. F. Lysosomal enzymes in the acrosome and their possible role in fertilization. **J. Reprod. Fertil.**, v. 21, p. 501-515, 1970.

AMANN, R. P. Reproductive capacity of dairy bulls. IV. Spermatogenesis ans texticular germ cell degeneration. **Am.J. Anat.**, v. 110, n. 1, p. 69-78, 1962.

AMANN, R. P. Sperm production rates. In: JOHNSON, A. D.; GOMES, W. R.; VANDEMARK, N. L. (Ed.). **The testis**. New York: Academic Press, 1970. v. 1, cap. 7, p. 433-482.

AMANN, R. P.; SCHANBACHER, B. D. Physiology of male reproduction. **J. Anim. Sci.**, v. 57, n. 2, p. 380-403, 1983. suppl.

BARTH, A. D.; OKO, R. J. **Abnormal morphology of bovine spermatozoa**. Ames: Iowa State University Press, 1989.

BEDFORD, J. M.; HOSKINS, D. D. The mammalian spermatozoon: Morphology, biochemistry and physiology In: **Marshall's Physiology of Reproduction**. Male Reproduction Lamming, G.E. (Ed), London: Churchil Livingstone, v. 2, p. 379-568, 1990.

BERNDTSON, W. E. Methods for quantifying mammalian spermatogenesis: a review. **J. Anim. Sci.**, v. 44, n. 5, p. 818-83, 1977.

BERNDTSON, W. E.; DESJARDINS, C. The cycle of the seminiferous epithelium and spermatogenesis in the bovine testis. **Am. J. Anat.**, v. 140, p. 167-180, 1974.

Berndtson, W. E.; Igboeli, G.; Pickett, B. W. Relationship of absolute number of Sertoli cells to testicular size and spermatogenesis in young beef bulls. **J. Anim. Sci.**, v. 64, p. 241-246, 1987.

BILASPURI, G. S.; GURAYA, S. S. The seminiferous epithelial cycle and spermatogenesis in goats (Capra hircus). **J. Agric. Sci.**, v. 103, p. 359-368, 1984.

BURGOS, M. H.; FAWCETT, D. W. Studies on the fine structure of the mammalian testis. I. Differentiation of the spermatids in the cat (Felis domestica). **J. Biophys. Biochem. Cytol.**, v. 1, p. 282-300, 1955.

CASTRO, A. C. S.; BERNDTSON, W. E.; CARDOSO, F. M. Cinética e quantificação da espermatogênese: bases morfológicas e suas aplicações em estudos da reprodução de mamíferos. **Rev. Bras. Reprod. Anim.**, v. 21, n. 1, p. 25-34, 1997.

CLAY, C. M. et al. Influences of season and artificial photoperiod on stallions: luteinizing hormone, follicle stimulating hormone and testosterone. **J. Anim. Sci.**, v. 66, p. 1246-1255, 1988.

Clermont, Y. Kinetics of spermatogenesis in mammals, seminiferous epithelium cycle and spermatogonial renewal. **Physiol. Rev.**, v. 52, n. 1, p. 198-236, 1972.

Clermont, Y.; Antar, M. Duration of the cycle of the seminiferous epithelium and the spermatogonial renewal in the monkey (Macaca arctoides). **Am. J. Anat.**, v. 136, p. 153-166, 1973.

CLERMONT, Y.; LEBLOND, C. P. Spermiogenesis of man, monkey, ram and other mammals as shown by the periodic acid-Schiff technique. **Am. J. Anat.**, v. 96, p. 229-253, 1955.

Clermont, Y.; Leblond, C. P.; Messier, B. Durée du cycle de l'epithélium séminal du rat. **Arch. Anat. Microscop. Morphol. Exp.**, v. 48, p. 37-56, 1959.

COUROT, M.; HOCHEREAU-DE-REVIERS, M. T.; ORTAVANT, R. Spermatogenesis. In: JOHNSON, A. D.; GOMES, W. R.; VANDEMARK, N. L. (Ed.). **The testis**. New York: Academic Press, 1970. v. 1, cap. 6, p. 339-432.

CURTIS, G. M. The morphology of the mammalian seminiferous tubule. **Am. J. Anat.**, v. 24, p. 339-394, 1918.

De Rooij, D. G.; Gootegoed, J. A. Spermatogonial stem cells. **Curr. Opin. Cell Biol.**, v. 10, p. 694-701, 1998.

DYM, M. The fine structure of monkey (Macaca) Sertoli cell and its role in maintaining the blood-testis barrier. **Anat. Rec.**, v. 175, p.639-656, 1973.

EDDY, E. M. The spermatozoon. In: KNOBILL, E.; NEILL, J. **The physiology of reproduction**. New York: Raven Press, 1988. p. 27-68.

FAWCETT, D.W. A comparative view of sperm ultrastructure. **Biol. Reprod.**, v. 2, p. 90-127, 1970. suppl 2.

FOOTE, R. H.; SWIERSTRA, E. E.; HUNT, W. L. Spermatogenesis in the dog. **Anat. Rec.**, v. 173, p. 341-352, 1972.

FRANÇA, L. R.; RUSSELL, L. D. The testis of domestic animals. In: Regadera, J.; Martinez-Garcia (ed.). **Male reproduction:** a multidisciplinary overview. Churchill Livingstone: Madrid, 1998. p. 197-219.

GURAYA. S. S. **Biology of spermatogenesis and spermatozoa in mammals**. Berin: Springer-Verlag, 1987.

HECHT, N. B. Regulation of haploid expressed gene in male germ cells. **J. Reprod. Fertil.**, v. 88, p. 679-693, 1990.

HESS, R. A. et al. Adult testicular enlargement induced by neonatal hypothyroidism is accompanied by increased Sertoli cell and germ cell numbers. **Endocrinology,** v. 132, p. 2607-2613, 1993.

HOCHEREAU-DE-RIVERS, M. T.; COUROT, M.; ORTAVANT, R. Durée de la spermatogenése chez le taureau, étude par autoradiographie testiculaire. In: CONGRESSO INTERNAZIONALE PER LA

RIPRODUZIONE ANIMALE E LA FECONDAZIONE ARTIFICIALE, 5., 1964., Trento, p. 541-546.

HUCKINS, C. The spermatogonial stem cell population in adult rats I. Their morphology, proliferation and maturation. **Anat. Rec**., v. 169, p. 533-558, 1971.

JOHNSON, L., NGUYEN, H. B. Annual cycle of the Sertoli cell population in adult stallions. **J. Reprod. Fertil**., v. 76, p. 311-316, 1983.

Johnson, L.; Tatum, M. E. Temporal appearance of seasonal changes in numbers of Sertoli cells, Leydig cells, and germ cells in stallions **Biol Reprod**, v. 40, p. 994-999, 1989.

JOHNSON, L. Spermatogenesis. In: **Reproduction in Domestic Animals**. Academic Press. 1991.

JOHNSON, L. et al. Efficiency of spermatogenesis: a comparative approach. **Anim. Reprod. Sci.**, v. 60-61, p. 471-480, 2000.

LEBLOND, C. P.; CLERMONT, Y. Definition of the stages of the cycle of the seminiferous epithelium in the rat. **Ann. N.Y. Acad. Sci**., v. 55, p. 548-584, 1952.

OAKBERG, E. F. Duration of spermatogenesis in the mouse and timing of stages of the cycle of the seminiferous epithelium. **Am. J. Anat**., v. 99, p. 507-516, 1956.

Oakberg, E. F. Spermatogonial stem-cell renewal in the mouse. **Anat. Rec**., v. 169, n. 3, p. 515-31, 1971.

ORTAVANT, R. Spermatogenesis and morphology of the spermatozoon. In: **Reproduction in Domestic Animals**, Cole, H.H., Cupps, P.T. (Eds) New York: Academic Press, v. 2, cap 1. 1959.

ORTAVANT, R.; COUROT, M.; HOCHEREAU-DE-REVIERS, M. T. Spermatogenesis in domestic mammals. In: COLE, H.H., CUPPS, P.T. (Ed). **Reproduction in domestic animals**. 3. ed. New York: Academic Press. cap. 8, p.203-227, 1977.

ORTH, J. M.; GUNSALUS, G. L.; LAMPERTI, A. A. Evidence from Sertoli cell-depleted rats indicates that spermatid number in adults depends on numbers of Sertoli cells produced during perinatal development. **Endocrinology**., v. 122, p. 787-794, 1988.

PARVINEN, M. et al. In vitro stimulation of stage-specific deoxyribonucleic acid synthesis in rat seminiferous tubule by interleukin-1alfa. **Endocrinology**, v. 129, p. 1614-1620, 1991.

PAULA, T. A. R. **Avaliação Histológica e Funcional do Testículo de Capivaras Adultas (Hydrochoerus hydrochaeris).** Instituto de Ciências Biológicas. Universidade Federal de Minas Gerais. Belo Horizonte. 1999. 84p. Tese (Doutorado).

PEREY, B.; CELRMONT, Y.; LEBLOND, C. P. The wave of the seminiferous epithelium in the rat. **Am. J. Anat**., v. 108, p. 47-77, 1961.

ROGER, M. H.; YANAGIMACHI, R. Release of hyaluronidase from guinea pig spermatozoa through an acrosome reaction initiated by calcium**. J. Reprod. Fertil**., v. 44, p. 135-138, 1975.

ROOSEN-RUNGE, E. C. Untersuchungen über die degeneration samenbildender zellen in der normalen spermatogenese der ratte**. Z. Zellforsch**., v. 41, p. 221-235, 1955.

RUSSELL, L. Movement of spermatocytes from the basal to the adluminal compartment of the rat testis. **Am. J. Anat**., v. 148, p. 313-328, 1977.

RUSSEL, L.D. Sertoli-germ cell interrelations-a review**. Gamete Res**., v. 3, p. 179-202, 1980.

RUSSELL, L. D.; PETERSON, R. N. Determination of the elongate spermatid-Sertoli cell ratio in various mammals**. J. Reprod. Fert**., v. 70, p. 635-641, 1984.

RUSSELL, L. D. et al. **Histological and histopathological evaluation of the testis**. Cache River Press, Clearwater, Florida, 1990.

RUSSELL, L. D.; GRISWOLD, M. D. (Ed) **The Sertoli cell**. Clearwater, Florida: Cache River Press, 1993.

SPRANDO, R. L.; RUSSELL, L. D. Comparative study of cytoplasmic elimination in spermatids of selected mammalian species. **Am. J. Anat**., v. 178, p. 72-80, 1987.

STAMBAUGH, R. L.; BUCKLEY, J. Identification and subcellular localization of the enzymes affecting penetration of the zona pelucida by rabbit spermatozoa. **J. Reprod. Fertil**., v. 19, p. 423-432, 1969.

STEINBERGER, A.; STEINBERGER, E. Replication pattern of Sertoli cells in maturing rat testis in vivo and in organ culture. **Biol. Reprod**., v. 4, p. 84-87, 1971.

STRIVASTAVA, P. N.; ADAMS, C. E.; HARTREE, E.F. Enzymic action of acrosomal preparation on the rabbit ovum in vitro. **J. Reprod. Fertil**., v. 10, p. 61-67, 1965.

SWIERSTRA, E. E. Cytology and duration of the cycle of the seminiferous epithelium of the boar: duration of spermatozoa transit through the epididymes. **Anat. Rec**., v. 161, p. 171-186, 1968.

SWIERSTRA, E. E.; GEBAUER, M. R.; PICKETTI, B. W. Reproductive physiology of the stallion. I. spermatogenesis and testis composition. **J. Reprod. Fertil**., v. 40, p. 113-123, 1974.

WARD, S. W.; COFFEY, D. S. DNA packaging and organization in mammalian spermatozoa: Comparison with somatic cells. **Biol Reprod**., v. 44, p. 569-574, 1991.

CONTROLE DE ACESSO INTELIGENTE

SCIENTIA

MARCELO OLIVEIRA CAMPONÊZ[1]
MARCELO BRUNORO[2]

[1] Mestre em Engenharia Elétrica pela Universidade Federal do Espírito Santo. Professor do Centro Universitário Vila Velha. E-mail: camponez@uvv.br.

[2] Mestre em Engenharia Elétrica pela Universidade Federal do Espírito Santo. Professor do Centro Universitário Vila Velha. E-mail: brunoro@uvv.br.

Scientia Vila Velha (ES), v. 4, n. 1/2, **p. 73-80**, jan/dez. 2003

RESUMO

Descreve o desenvolvimento de um controle de acesso inteligente, microcontrolado, que é um produto utilizado para controlar, restringir e automatizar o acesso de pessoas a locais específicos e pode ser utilizado em empresas ou residências. Neste caso específico, o controle se dá pela autenticação de uma senha pessoal digitada num teclado. A pesquisa e o desenvolvimento foram realizados no laboratório de hardware do Centro Universitário Vila Velha, onde estão disponíveis todos os recursos necessários para o projeto do hardware, do software e do design do controle de acesso inteligente.

Palavras-chaves: Microeletrônica. Controle de acesso. Microcontrolador.

1 INTRODUÇÃO

O Controle de Acesso Inteligente é um produto utilizado para controlar, restringir e automatizar o acesso de pessoas a locais específicos e pode ser utilizado em empresas ou residências. Este controle se dá pela autenticação de uma senha pessoal digitada num teclado do produto, conforme mostra a Figura 1.

Figura 1 – Console

Esta pesquisa, em que foram desenvolvidos o *hardware* (PEREIRA, 2002; SOUZA, 2002), o *software* (GARDNER, 2002; KERNIGHAN; RITCHIE, 1988) e o *design* do produto, foi toda desenvolvida numa parceria entre o **CENTRO UNIVERSITÁRIO VILA VELHA**, instituição de ensino superior mantida pela **SOCIEDADE EDUCACIONAL DO ESPIRITO SANTO - UNIDADE VILA VELHA** – **ENSINO SUPERIOR**, pessoa jurídica de direito privado, sem fins lucrativos, estabelecida no Município de Vila Velha-ES, na rua Comissário José Dantas de Melo, 21 - Boa Vista, inscrita no CNPJ sob o nº. 27.067.651/0001-55, e a **KRL ELETRÔNICA E COMÉRCIO LTDA**., pessoa jurídica de direito privado, estabelecida no município de Vila Velha-ES, na rua Stefana Cribilie, 115 - Nossa Senhora da Penha, inscrita no CNPJ sob o nº 01.119.025/0001-71.

2 MATERIAL E MÉTODOS

Na concepção do projeto do Controle de Acesso Inteligente, com objetivo de produzir-se uma solução com vantagens competitivas em relação aos concorrentes, foi feita uma pesquisa em relação aos produtos similares no mercado, para levantamento dos pontos fortes e fracos de cada um dos concorrentes. O resultado da pesquisa foi o seguinte:

Principais pontos fracos dos produtos concorrentes:

- O relé de acionamento da fechadura de alguns produtos, fica do lado de fora da porta, exposto a fraudes;
- O relé de acionamento da fechadura de outros produtos, fica do lado de dentro da porta, mas o sinal de acionamento não é codificado e pode ser violado;
- Alguns equipamentos têm reset de fábrica e senha padrão de fábrica. Neste caso, basta acessar o manual do produto pela internet, descobrir o procedimento de inicialização para fraudar o sistema;
- Alguns produtos não têm mecanismo de temporização para vários erros consecutivos de senha;
- Todos os produtos similares têm preços acima de R$ 1.000,00 por porta controlada.

Pontos fortes dos produtos concorrentes:

- Uso do state of art da eletrônica e programação;
- Facilidade de operação e programação do produto;
- Designer apropriado para instalações em residências e/ou empresas;
- Opções de gerência.

O Controle de Acesso inteligente implementa todos os pontos fortes dos concorrentes com o uso do *state of art* da eletrônica (PEREIRA, 2002; SOUZA, 2002) e programação (GARDNER, 2002; KERNIGHAN; RITCHIE, 1988), com console de fácil operação e programação, contando ainda com um *display* de cristal líquido para orientar o usuário, além disso, o desenho industrial do produto é apropriado para instala-

ções em residências e/ou empresas e há 8 (oito) opções de serviços implementados que facilitam a gerência do produto.

Todos os pontos fracos encontrados em produtos similares foram resolvidos: o custo foi sensivelmente reduzido em 50% do preço dos concorrentes. Não há senha-padrão nem serviço de reinicialização, o que garante mais segurança; o relé que aciona a fechadura fica do lado de dentro da porta e é inacessível, além disso, o sinal entre o relé e console é codificado, o que torna o sistema inviolável.

3 RESULTADOS E DISCUSSÃO

Depois de finalizadas as fases de projeto, implementação e testes o produto final da pesquisa, o Controle de Acesso Inteligente apresenta as seguintes características:

- visor de cristal líquido (LCD) que facilita a programação das várias funções disponíveis;
- teclado para digitação da senha de acesso e programação de serviços;
- codificação do sinal entre o console (Figura 1) e a fechadura, o que garante mais segurança ao produto;
- a interface entre o console e a fechadura é feita com 3 fios, o que facilita a instalação do produto;
- capacidade para armazenar 30 senhas de usuário com 6 caracteres, para acesso;
- além das senhas de usuário, há uma senha de administrador com 8 caracteres para programação;
- há um serviço que trava, por 5 minutos, o controle de acesso, caso um usuário erre a senha por 5 vezes consecutivas;
- através do teclado, o produto, que é microcontrolado *(PEREIRA, 2002; SOUZA, 2002)*, pode ser programado para várias funções, facilitando a gerência e possibilitando a criação de uma política de acesso a ambientes. A interface de programação é amigável e o feedback para o usuário é feito através de mensagens no LCD e de bips. Os serviços disponíveis são:

1.Acesso Controlado

O principal serviço é o Controle de Acesso Inteligente que, após analisar a senha digitada permite ou não a entrada em um ambiente controlado.

Para entrar, digita-se uma senha de usuário e em seguida a tecla ENTRA. Se a senha estiver correta, após um bip, no LCD aparecerá escrito “Welcome” e a porta se abrirá. Se a senha estiver errada, após um bip, no LCD aparecerá “Try Again ...” e a porta não se abrirá.

2.Mudança de senha de usuário;

Por este serviço é possível alterar a senha de um usuário.

3.Criação de usuários;

Por este serviço o administrador do Controle de Acesso Inteligente pode criar usuários. Este produto suporta até 30 (trinta) usuários e um administrador.

4.Exclusão de usuários;

Por este serviço o administrador do Controle de Acesso Inteligente pode excluir um usuário.

5.Mudança de senha do administrador;

Por este serviço é possível alterar a senha do administrador.

6.Serviço de acesso irrestrito

Neste serviço o administrador pode configurar o controle de acesso para abrir a porta quando qualquer uma das teclas for pressionada. É um serviço que pode ser utilizado, por exemplo, em um dia de reunião, em que várias pessoas diferentes acessarão o ambiente.

7.Serviço de travamento de senha

Para maior segurança, o Controle de Acesso Inteligente travará por 5 minutos toda vez que um usuário errar a senha por 5 (cinco) tentativas seguidas. Cinco bips consecutivos indicam este estado.

8.Serviço de destravamento de senha

O administrador poderá fazer uso do serviço de “destravamento de senha” para destravar o console antes dos 5 minutos de penalidade.

4 CONCLUSÃO

Numa parceria bem-sucedida entre a **UVV/SEDES** e a **KRL,** os professores M.Sc. Marcelo Oliveira Camponêz e M.Sc. Marcelo Brunoro desenvolveram um Controle de Acesso Inteligente, viável economicamente, com uso das mais avançadas práticas de programação *(GARDNER, 2002; KERNIGHAN; RITCHIE, 1988)*; e de eletrônica *(PEREIRA, 2002; SOUZA, 2002)* que, em relação aos potenciais concorrentes de mercado, traz inúmeras inovações tecnológicas.

Dentre essas inovações pode-se destacar:

- Alguns produtos do mercado cometem uma grave falha de segurança ao instalar o relé de acionamento da fechadura eletrônica do lado de fora da porta. Nesta configuração, se o console do produto for aberto, um intruso pode acionar o contato do relé e violar o sistema. Alguns fabricantes instalam o relé do lado de dentro da porta, em lugar inacessível, mas não codificam o sinal que aciona o relé. Nesta configuração um intruso ainda pode, com um sinal de 12 volts, acionar o relé e violar o sistema. No Controle de Acesso Inteligente o relé é colocado do lado de dentro da porta e acionado por um sinal codificado, desta forma o sistema é inviolável.
- Através de uma pesquisa de mercado, constata-se que o produto concorrente de menor custo não implementa todas as funcionalidades incorporadas ao Controle de Acesso Inteligente e, mesmo assim, custa o dobro do preço.
- Compromisso com a segurança. Todos os produtos do mercado foram minuciosamente estudados, os seus pontos críticos de segurança foram mapeados. O projeto do Controle de Acesso Inteligente, desde a sua concepção, teve a preocupação de sanar os problemas detectados no mercado, pelo desenvolvimento de um controle de acesso inviolável.

- Há 8 (oito) opções de serviços implementados no firmware *(PEREIRA, 2002)* do Controle de Acesso Inteligente, que facilitam a gerência e possibilitam a criação de uma política de acesso a ambientes. A interface de programação é amigável e o feedback para o usuário é feito através de mensagens no LCD *(PEREIRA, 2002)* e de bips. Os serviços disponíveis são: Acesso Controlado; Mudança de senha de usuário; Criação de usuários; Exclusão de usuários; Mudança de senha do administrador; Serviço de acesso irrestrito; Serviço de travamento de senha e Serviço de destravamento de senha.

INTELLIGENT CONTROL ACCESS

ABSTRACT

It describes the development of an intelligent control access, microcontrolled, that it is an used product to control, to restrict and to automatize the access of people the specific places and can be used in companies or residences. In this specific case, this control is made for the authentication of a typed personal password through a keyboard. The research and the development had been carried through in the laboratory of the hardware of the University Center Vila Velha, where the necessary resources for the project of the hardware, the software and design of the control of intelligent access are available all.

Keywords: Microelectronics. Control of access. Microcontroller.

REFERÊNCIAS

GARDNER, N. **PIC C**: an introduction to programming the Microchip PIC in C. Brookfield: [s.n.], 2002.

KERNIGHAN, B. W.; RITCHIE, D. M. **The C Programming Language.** 2nd ed. Englewood Cliffs: Prentice Hall, 1988.

PEREIRA, F. **Microcontroladores PIC**: técnicas avançadas. São Paulo: Érica, 2002.

SOUZA, D. J. de. **Desbravando o PIC**. São Paulo: Érica, 2002.

APÊNDICE A – FOTOS DO PRODUTO

COMPARAÇÃO DE API'S NO ACESSO EFICIENTE A BANCOS DE DADOS HETEROGÊNEOS NA *WORLD WIDE WEB*

SCIENTIA

EDMAR EDILTON DA SILVA[1]
CÉLIO CARDOSO GUIMARÃES[2]

[1] Mestre em Ciência da Computação pela UNICAMP. Professor do Centro Universitário Vila Velha. E-mail: edmar@uvv.br.

[2] Doutor pela Case Western Reserve University (EUA). Professor da UNICAMP. E-mail: celio@ic.unicamp.br

RESUMO

A *World Wide Web* atualmente é considerada a plataforma tecnológica ideal para o desenvolvimento de aplicações na Internet e *intranets* corporativas. Este artigo compara algumas API's de domínio público usadas para acessar bancos de dados heterogêneos. Essas API's são muito usadas no desenvolvimento de aplicações de bancos de dados (*stateful*) no contexto da *Web*. Nessas arquiteturas, um *pool* de conexões persistentes é usado para eliminar o custo associado à abertura de uma nova conexão com o SGBD. O ganho de desempenho medido através de uma série de experimentos em que é simulado o tráfego gerado por vários clientes HTTP concorrentes. As API's examinadas foram: Java Servlet (JDBC), Perl (DBI) e Python (*Database* API).

Palavras-chave: *WWW*. SGBDR. Conexões persistentes. Java Servlet. Perl. Python.

1 INTRODUÇÃO

O crescimento e a popularidade da *WWW* têm tornado esta plataforma de *software* essencial no desenvolvimento de aplicações para a Internet e *intranets* corporativas. O desenvolvimento de um sistema de informação de grande escala que utilize os padrões da Internet, pode ser feito através da integração de Sistemas de Gerenciamento de Bancos de Dados (SGBD's) com a tecnologia *Web*. Aplicações de bancos de dados tradicionais são construídas baseadas em técnicas de bancos de dados centralizados ou distribuídos. A maior parte dos bancos de dados distribuídos é interligada por uma *Local Area Network* (LAN) e/ou por uma *Wide Area Network* (WAN). Embora as técnicas desenvolvidas na integração de bancos de dados heterogêneos permitam pessoas acessar sistemas de bancos de dados heterogêneos que podem estar geograficamente distribuídos, o desenvolvimento de tais aplicações ainda é complexo e custoso.

Com a Internet globalmente disponível e *browsers* independentes de *hardware*, torna-se possível construir aplicações de bancos de dados baseadas na *Web*. Uma vez que o banco de dados está conectado à Internet, ele é acessível de qualquer computador interligado à Internet no mundo. Além disso, devido aos *browsers* estarem disponíveis para a maioria das plataformas, eles eliminam a necessidade de se projetar diferentes interfaces através de diferentes plataformas.

O restante deste artigo é estruturado como a seguir. A seção 2 descreve as dificuldades na integração de bancos de dados relacionais e as tecnologias *Web*. A seção 3 mostra algumas das principais técnicas utilizadas na preservação de estado na *Web*. A seção 4 descreve algumas API's de domínio público para acesso a bancos de dados heterogêneos. A seção 5 apresenta as técnicas utilizadas com as API's para o acesso eficiente a bancos de dados heterogêneos através da *Web*. Na seção 6, são realizados os testes para demonstrar o ganho de desempenho que essas técnicas fornecem na utilização das API's. O artigo é concluído na seção 7.

2 INTEGRANDO BANCOS DE DADOS E TECNOLOGIAS WEB

A maioria dos SGBD's não foi projetada para o ambiente da Internet. Embora esses sistemas possuam mecanismos de autorização para controlar o acesso aos dados armazenados no banco de dados, eles normalmente não fornecem um mecanismo de comunicação de rede seguro para a transmissão de dados. A maior parte desses sistemas usa *sockets* padrão ou *Remote Procedure Call* (RPC). Quando os sistemas de bancos de dados são interligados com a Internet, esse protocolo de comunicação nativo do SGBD se torna vulnerável a *hackers* e curiosos que podem interceptar e até mesmo alterar os dados transmitidos. Esse problema não é o objetivo desse trabalho. O projeto da aplicação e as técnicas de bancos de dados tradicionais precisam ser reavaliados no que diz respeito a alcançar uma alta escalabilidade e controle sobre o acesso de usuários aos bancos de dados através da *Web* (LU; LING, 1998; QUAN; LU; LING, 1999).

Alguns dos principais problemas que afetam a integração de sistemas de bancos de dados relacionais e a *Web* são os seguintes:

1. Como a maioria dos programas CGI é executada externamente ao servidor *Web*, um novo processo é criado e finalizado para cada execução do *script*. No caso de um *script* CGI implementado em linguagem interpretada, o interpretador deve ser carregado na memória e executado. Essas são operações caras, pois consomem muito tempo e recursos do servidor;
2. A comunicação entre o *browser* e o servidor Web, através do protocolo HTTP, não mantém informações de estado. Isso significa que toda requisição de um cliente a um servidor é tratada independentemente, não suportando o conceito de transação;

3. A maioria dos *scripts* CGI acessam bancos de dados, devido ao processo CGI ter um pequeno tempo de vida, toda vez que for feita uma requisição ao *script* uma nova conexão com o banco de dados deve ser aberta. O processo de abertura de uma nova conexão é uma operação cara, pois envolve autenticação de usuário e alocação de recursos.

Todos esses problemas podem ser resolvidos com a utilização de algumas técnicas bem conhecidas. O problema 1 pode ser resolvido com a utilização de módulos do servidor *Web* que mantém o interpretador do *script* embutido dentro do próprio servidor, desta forma o *script* não será mais executado externamente ao servidor. Um dos mecanismos de gerenciamento de estado, mostrado na seção 3, resolve o problema 2. Já o problema 3 é resolvido com a utilização de um Gerenciador de Conexões que torna as conexões com o banco de dados persistentes entre requisições de um mesmo usuário.

3 PRESERVAÇÃO DE ESTADO NA *WORLD WIDE WEB*

Aplicações baseadas na *Web*, normalmente precisam manter informações tais como o conteúdo de um carrinho de compras enquanto o cliente está passeando por um supermercado virtual para fazer compras (sessão de usuário) (IYENGAR; DIAS, 1998). Uma sessão é definida como uma série de requisições de um mesmo usuário (*browser*) durante um período de tempo específico.

Todos os métodos que veremos a seguir representam as informações de estado atribuindo valores a variáveis conhecidas como variáveis de estado. Alguns dos principais mecanismos de gerenciamento de estado propostos na literatura são detalhados a seguir.

FORMULÁRIOS HTML

Um método comum para manipular estado na *Web* envolve o uso de formulários *HyperText Markup Language* (HTML). Os servidores respondem a requisições de clientes com formulários HTML gerados dinamicamente que contêm as informações de estado embutidas em variáveis escondidas. Estes formulários HTML são tipicamente criados por programas CGI. As variáveis escondidas são passadas de volta para o servidor depois que o cliente submete o formulário.

Para ilustrar, suponha que um servidor de transação está se comunicando com um cliente. O cliente se identifica junto ao servidor para que o acesso a sua conta seja permitido. Após esta identificação inicial, o servidor manterá alguma informação de estado para que o cliente não tenha que se identificar em todas transações. Quando o cliente submeter o formulário, o servidor identificará o cliente com o identificador de usuário da variável escondida. A informação de estado é, portanto, passada entre o servidor e o cliente, enquanto a sessão de usuário for válida. Essa abordagem limita os tipos de interações que são possíveis entre um cliente e um servidor enquanto estado é preservado. O servidor deve sempre responder ao cliente com um formulário HTML gerado dinamicamente contendo variáveis escondidas, e não existe forma de preservar estado, por exemplo, quando o cliente está navegando entre documentos HTML estáticos.

COOKIES

Cookie é um mecanismo de preservação de estado, proposto pela Netscape e padronizado no RFC 2109 (KRISTOL; LOU, 1997), sem a limitação imposta pelos formulários HTML. Contudo, esse mecanismo requer extensões não padronizadas de servidores e *browsers Web*. Um servidor *Web* pode manter estado através de várias requisições adicionando um *cookie* a sua resposta. O *cookie* contém uma descrição do conjunto de URL's para o qual o estado é válido. O *cookie* é armazenado pelo *browser* do cliente, e quaisquer requisições HTTP futuras feitas pelo cliente para uma das URL's especificadas no *cookie* incluirão na requisição as informações de estado armazenadas no *cookie*.

Servidores *Web* enviam *cookies* para os clientes ao adicionar um cabeçalho, denominado *Set-Cookie*, como parte da resposta à requisição. Normalmente, *cookies* são criados por programas CGI. Por exemplo, um *browser* pode receber o seguinte *cookie* do servidor:

Set-Cookie: USERID=EDMAR; path=/dir1; expires=Wednesday, 28-Feb-2001 23:59:59 GMT

Esse *cookie* contém o valor para a variável de estado *USERID*. Quando o cliente requisitar uma URL do servidor sob o *path* "/dir", o *cookie* será enviado ao servidor:

Cookie: USERID=EDMAR

Nesse exemplo, o *cookie* expira às 23:59:59 horas do dia 28 de fevereiro de 2001. Se a data e hora de expiração não forem fornecidas, um *cookie* expira quando a sessão do *browser* termina.

ARGUMENTOS EMBUTIDOS DINAMICAMENTE

Embutir argumentos dinâmicos é uma técnica geral para manutenção de estado que associa estado com conversação. Uma conversação é uma seqüência de comunicações entre um cliente e um ou mais servidores no qual o cliente seleciona a próxima página seguindo um *link* hipertexto fornecido por um servidor.

Argumentos embutidos dinamicamente modificam *links* hipertexto para adicionar informações de estado (IYENGAR, 1997). *Links* hipertexto são alterados para invocar um programa especial conhecido como *argument embedder*. Esse programa passa as variáveis de estado para todos os programas CGI invocados pela aplicação.

Por exemplo, suponha que *embed* é o nome do *script* no qual é implementado o *argument embedder*. A aplicação deseja preservar as variáveis de estado *x=32* e *y=77*. Portanto, um *link* para um arquivo HTML:

<a href="http://davinci.unicamp.br/main.html">

será modificado para:

<a href="http://davinci.unicamp.br/cgi-bin/embed?url=//http://davinci.unicamp.br/main.html&x=32&y=77">

Quando esse *link* for selecionado, *embed* modificará os *links* no arquivo *main.html* para invocar o *script embed* com a URL original e as variáveis de estado como argumentos. O mesmo processo pode ser usado em um *link* para um programa CGI.

4 API'S DE DOMÍNIO PÚBLICO PARA ACESSO A BANCOS DE DADOS

A composição dinâmica de documentos HTML via *scripts* usando o mecanismo CGI pode ser codificada com linguagens interpretadas ou

compilada. Contudo, as linguagens interpretadas são muito mais usadas devido à sua maior flexibilidade, poder e facilidade de depuração. A seguir são descritas três *Application Programming Interface* (API's) de domínio público que têm sido largamente usadas na integração de SGBD's e tecnologias *Web*.

JAVA SERVLET

Java Servlet é uma ferramenta que resolve muitas das deficiências geradas pelos programas CGI padrão. Servlets são módulos de código em Java que executam como uma aplicação servidora no lado do servidor *Web* (por isso o nome "servlet", similar ao "applet" que é executado no lado do cliente). Servlets são portáveis, ao contrário de CGI's e API's proprietárias (NSAPI da *Netscape* ou ISAPI da *Microsoft*), uma aplicação servlet é escrita uma única vez e pode ser executada em qualquer plataforma disponível para a aplicação.

A aplicação servlet pode conectar-se com outros computadores utilizando *sockets* ou *Remote Method Invocation* (RMI). Também pode conectar-se a bancos de dados relacionais usando o padrão *Java DataBase Connectivity* (JDBC). Servlets têm várias vantagens sobre CGI padrão, tais como:

- Um servlet não é executado em um processo separado. Desta forma, é removido o *overhead* de criação de um novo processo para cada requisição HTTP;
- Um servlet permanece em memória entre requisições HTTP, não precisando ser carregado (inclusive o interpretador Java) e inicializado para toda requisição HTTP;
- Existe uma única instância que responde todas as requisições concorrentemente. Permitindo assim, servlets gerenciarem facilmente dados persistentes.

4.1.1 JDBC

SQL *Call-Level Interface* (SQL CLI) foi originalmente definido pelo SQL *Access Group* (SAG) para fornecer um padrão especial para acesso a dados remotos. CLI requer o uso de *drivers* inteligentes que aceitam uma chamada e a traduzem para a linguagem nativa de acesso ao servidor de banco de dados. CLI requer um *driver* para todo banco de dados ao qual este se conecta.

Um SQL CLI denominado *Java DataBase Connectivity* (JDBC) foi desenvolvido em conjunto pela *Javasoft*, *Sybase*, *Informix*, IBM e outras empresas. JDBC é um CLI orientado a objeto e portável, escrito em Java, mas muito similar ao ODBC. JDBC consiste de um conjunto de classes e interfaces que fornece uma API padrão tornando possível o desenvolvimento de aplicações de bancos de dados independentes de SGBD's.

PERL

Practical Extraction and Report Language (Perl) é uma linguagem de programação interpretada e portável. Perl possui uma grande facilidade na manipulação de texto, esta característica torna Perl uma ferramenta muito popular entre os desenvolvedores de *scripts* CGI. A seguir são apresentados alguns módulos que podem ser usados com a tecnologia Perl com o objetivo de melhorar o desempenho dos *scripts* Perl.

MOD_PERL

O problema 1, citado na seção 2, é o mais crítico na execução de *scripts* CGI. Uma solução para esse problema é a utilização de um módulo Apache denominado mod_perl. Esse módulo embute o interpretador Perl dentro do servidor Apache.

Apache::Registry é um módulo Perl que trabalha em conjunto com mod_perl, esse módulo é usado para emular o ambiente CGI, com isso *scripts* CGI escritos em Perl podem ser usados com mod_perl sem ter que reescrevê-los. Com Apache::Registry, cada programa CGI individual é compilado e cacheado na primeira vez que esse é requisitado ou quando esse é atualizado no disco, e então permanece disponível para todas as próximas requisições desse *script* CGI. O efeito é que os *scripts* CGI são pré-compilados pelo servidor e executados sem a necessidade da criação de novos processos.

DBI

DataBase Interface (DBI) é uma API para a linguagem Perl, que fornece uma interface de conexão a banco de dados simples e padronizada. DBI define um conjunto de funções, variáveis e convenções que fornece uma interface consistente para acesso a bancos de dados. DBI abstrai a necessidade de entender o nível mais baixo de acesso a bancos de dados: é uma camada transparente que fica entre uma aplicação e

um ou mais *DataBase Drivers* (DBD's), isto é, são os *drivers* que realmente fazem o trabalho. Os DBD's implementam suporte para um dado tipo de sistema de banco de dados.

APACHE::DBI

Apache::DBI foi escrito para superar a limitação imposta pelo problema 3. Esse módulo inicializa um *pool* de conexões persistentes ao banco de dados. Todo pedido de conexão ao banco de dados que é feita através do DBI, será transferido ao módulo Apache::DBI. Esse módulo verifica se o *handle* de uma conexão aberta anteriormente já está armazenado e verifica, usando o método *ping*, se esse *handle* ainda é válido. Se essas duas condições forem verdadeiras o Apache::DBI apenas retorna o *handle* da conexão. Se não existir um *handle* apropriado ou se o método *ping* falhar, uma nova conexão será estabelecida e o *handle* armazenado para futuramente ser reutilizado.

Quando o *script* Perl é executado novamente por um processo filho do Apache que ainda está conectado, o Apache::DBI apenas verifica o *cache* de conexões abertas comparando os parâmetros de conexão (*host*, *username* e *password*). Uma conexão com os mesmos parâmetros será retornada se estiver disponível ou uma nova conexão será criada e depois retornada.

PYTHON

Python é uma linguagem de programação interpretada, interativa, orientada a objetos e escrita em ANSI C. Seu projeto combina características de engenharia de *software* de linguagens tradicionais com a grande utilidade de linguagens *script*. A linguagem Python é freqüentemente comparada às linguagens: Java, Perl e Tcl. Python combina um extraordinário poder com uma sintaxe muito clara, podendo ser estendido pela adição de novos módulos implementados em uma linguagem compilada tal como C ou C++. A implementação Python é portável: é executada sobre Unix (Solaris ou Linux), Windows, OS/2 e outros; ou seja, Python é suportado por todos os sistemas operacionais que possuem um compilador C.

PYTHON DATABASE API

Essa API foi definida para padronizar a interface entre módulos Python e diferentes SGBD's. Com isso, é esperado alcançar uma consistência

conduzindo a módulos mais facilmente compreensíveis, códigos que são geralmente mais portáveis através de bancos de dados e um amplo alcance de conectividade de bancos de dados.

5 USANDO API'S DE DOMÍNIO PÚBLICO NA INTEGRAÇÃO DE BANCOS DE DADOS E TECNOLOGIAS *WEB*

Esse artigo apresenta algumas técnicas cujo objetivo é reduzir drasticamente a necessidade de abertura de novas conexões. A idéia central é utilizar um agente de banco de dados, que é permanentemente conectado ao SGBD através de um *pool* de conexões.

JAVA SERVLET

O agente de banco de dados foi implementado como um pacote Java, denominado Gerenciador de Conexões. Esse pacote fornece capacidade à aplicação de realizar a preservação de estado, através de uma tabela de sessões, entre consecutivos acessos de um mesmo usuário. O Gerenciador de Conexões possui um *pool* de conexões compartilhadas. Quando um *script* termina de usar uma conexão, esta conexão é reciclada e não destruída. Com isso, as conexões tornam-se persistentes e podem ser utilizadas entre várias requisições HTTP.

A estrutura com vários *threads* (*multi-threaded*) das aplicações servlets fornece uma vantagem de desempenho sobre as implementações convencionais com um único *thread* (*single-threaded*). Desta forma, várias requisições HTTP podem ser manipuladas simultaneamente diminuindo o tempo de resposta do usuário. A Máquina Servlet utilizada na execução dos servlets foi o Apache Jserv. Esse *software* faz parte do Projeto Java Apache. Os *drivers* para acesso aos bancos de dados foram: *oci8* para acessar o Oracle e *freetds_jdbc* para acessar o MS SQL Server. Todos esses *softwares* com exceção do *oci8* são de domínio público.

PERL

Com o propósito de resolver os principais problemas encontrados na arquitetura CGI padrão, utilizamos os módulos mod_perl e Apache::DBI. Nessa arquitetura, o agente de banco de dados foi implementado com a utilização dos módulos citados acima. Em relação à conectividade,

podemos usar como banco de dados qualquer SGBD suportado pela combinação dos módulos *DataBase Interface* (DBI) e *DataBase Driver* (DBD) da biblioteca Perl, estando incluídos os bancos de dados Oracle, MS SQL Server e outros.

Os *drivers* utilizados no acesso aos bancos de dados foram: DBD::Oracle para acessar o SGBD Oracle e DBD::Sybase para acessar o SGBD MS SQL Server, como o próprio nome diz, esse *driver* também pode ser usado para acessar o SGBD Sybase.

PYTHON

A implementação do agente de banco de dados, em Guimarães (2000), é proposta como a junção do "*Session Manager*" e do "*Database Application*" em Hadjiefthymiades, Martakos e Petrou (1999), em um único processo de longa duração chamado *Database Connection Manager* (DBCM). Essa solução é composta de duas partes: um programa Python (*daemon*) que implementa o DBCM e um pequeno programa CGI escrito em C.

Nessa implementação é utilizada uma nova e eficiente técnica de comunicação de dados entre o servidor *Web* e o DBCM, onde o *script* CGI conecta-se ao DBCM, através de *sockets* Unix, e envia seus descritores (*input* e *output*). Essa implementação requer somente uma conexão socket para comunicação de dados entre o DBCM e o SGBD. Uma outra conexão *socket* entre o DBCM e o *script* CGI é usada somente para receber os descritores. O principal benefício dessa técnica, em relação a de Hadjiefthymiades, Martakos e Petrou (1999), é que além da recepção dos dois descritores de E/S do *script* CGI que são passados ao DBCM, nenhuma comunicação adicional é necessária entre eles: a entrada do usuário é lida diretamente do servidor *Web* pelo DBCM e a saída gerada é escrita diretamente para o servidor *Web* usando o descritor de saída recebido do *script* CGI. Desta forma, é reduzido o número de conexões *sockets* utilizadas na transmissão dos dados entre o servidor Apache e o DBCM.

DBCM foi implementado como um servidor concorrente convencional, que dispara um processo filho para cada requisição de um *script* CGI. DBCM mantém informações de estado e as conexões abertas anteriormente, para cada usuário ativo, em uma tabela residente em memória denominada tabela de conexões. A conexão do DBCM com o SGBD Oracle é feita através do *driver* DCOracle da *Digital Creations Oracle Interface*.

6 MEDIDAS E ANÁLISES DE DESEMPENHO

Nesta seção, é apresentada uma série de experimentos que permite a quantificação dos *over-heads* de tempo impostos pelas arquiteturas de *gateway* convencionais e os benefícios que podem ser obtidos pelas arquiteturas descritas nesse artigo.

Como em Hadjiefthymiades, Varouxis e Martakos (2000) e em Zhao (1999), foram realizados vários testes para demonstrar que as arquiteturas apresentadas são significativamente mais eficientes que os *gateways* tradicionais.

O tempo total (t_{total}) gasto pelos *gateways* tradicionais na recuperação e transmissão da informação solicitada de volta ao *browser* é dado por:

$$t_{total} = t_{processo} + t_{interpretador} + t_{script} + t_{conexao} + t_{execucao} + t_{SQL}$$

Na fórmula 1, os fatores que implicam diretamente no tempo total (t_{total}) de execução de um *script* CGI padrão são representados por:

1. $t_{processo}$ representa o tempo gasto na geração de um novo processo CGI;
2. $t_{interpretador}$ representa o tempo que o sistema gasta para carregar o interpretador do *script* na memória;
3. t_{script} é o tempo gasto pelo sistema para carregar na memória o código do *script*;
4. $t_{conexao}$ é o tempo gasto na reserva de recursos e no estabelecimento de uma conexão com o banco de dados;
5. $t_{execucao}$ representa o tempo de execução do código útil do *script* (código que desempenha a tarefa solicitada pelo cliente);
6. t_{SQL} é o tempo de processamento gasto pelo sistema para executar o comando SQL submetido pelo cliente.

Os quatro primeiros fatores são todos controláveis, enquanto que $t_{execucao}$ e t_{SQL} variam dependendo respectivamente do tamanho do código útil do *script* e do tipo de comando SQL submetido pelo cliente. O t_{SQL} depende também se o SGBD utilizado implementa ou não um sistema de *cache* para as consultas SQL.

Ao contrário dos *gateways* tradicionais, onde todos os seis fatores são determinantes do tempo total de execução de um *script* CGI, nas arquiteturas Java, Perl e Python, vários desses fatores podem ser reduzidos ou até mesmo eliminados com a utilização de Servlets (Java), módulos mod_perl e Apache::DBI (Perl) e DBCM (Python).

6.1 CONFIGURAÇÃO DOS EXPERIMENTOS

Foram utilizadas quatro máquinas durante o processo de desenvolvimento dos experimentos. As configurações de *hardware* e *software* utilizadas foram: *host* 1 é um Pentium I 200 MHz com 64 MB de RAM, Linux Red Hat 6.2 e Apache JMeter 1.3; *host* 2 é um AMD-K6 350 MHz com 128 MB de RAM, Linux Red Hat 6.2 e Servidor Apache 1.3.14; *host* 3 é um AMD-K6 300 MHz com 64 MB de RAM, Linux Red Hat 6.2 e Oracle 8.0.5; *host* 4 é um Pentium I 166 MHz com 32 MB de RAM, Microsoft Windows NT Server 4.0 e Microsoft SQL Server 6.5.

Os experimentos consistem de uma série de testes em que um programa gerador de carga (Apache JMeter) direciona uma grande quantidade de requisições HTTP ao servidor Apache. O Apache JMeter é uma ferramenta de domínio público que faz parte do Projeto Java Apache (JAVA..., [2000?]). O Apache JMeter é uma aplicação, escrita em Java, projetada para testar o comportamento e medir o desempenho de URL's sob períodos de alta carga; pode ser usado para testar o desempenho de servidores *Web* na recuperação de documentos HTML estáticos ou gerados dinamicamente (CGI, Servlets, Perl, Python, etc.).

6.2 CENÁRIOS DOS EXPERIMENTOS

O Apache JMeter foi configurado para simular o tráfego causado por até 16 clientes HTTP (*threads*) concorrentes, começando por um *thread* e incrementando de 1 até 16 *threads* concorrentes. Cada *thread* faz 10 repetições de uma mesma requisição, permitindo portanto que o experimento alcance um estado estável. Durante os testes, arquivos de *log* de atividades foram atualizados com as seguintes informações:

- Tempo de Conexão: tempo gasto na alocação de recursos e no estabelecimento de uma conexão com o banco de dados;

- Tempo de Resposta: tempo total gasto para completar a transferência dos dados requisitados pelo cliente (*browser*), incluindo o tempo de conexão com o servidor *Web*.

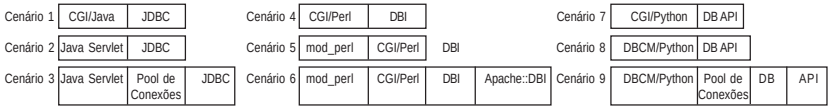


Figura 1 – Cenários dos experimentos avaliados

A Figura 1 mostra os cenários de acesso a banco de dados sujeitos à avaliação através dos experimentos. Nos cenários 1 e 4, o *script* e o intertrepador são carregados a cada requisição. Nos cenários 2, 3, 5, 6, 7 e 8, o *script* e o interpretador são pré-carregados. Nos cenários 3, 6 e 8, existem *pools* de conexões abertas.

Todos os testes foram realizados duas vezes, uma vez para cada um dos SGBD's que estão sendo avaliados (Oracle e MS SQL Server), com exceção dos cenários 7 e 8 onde foram realizados testes somente com o SGBD Oracle. Em todos os casos, o acesso aos bancos de dados envolve a execução de uma consulta SQL simples sobre uma tabela com poucos registros: o tamanho do documento HTML gerado é de 247 *bytes*.

6.3 ANÁLISE DE DESEMPENHO

6.3.1 AVALIAÇÃO DO TEMPO DE CONEXÃO

Ambiente Java

A Figura 2(a), mostra que o tempo de abertura de uma conexão ($t_{conexao}$) foi totalmente eliminado devido à utilização do *pool* de conexões, mostrando claramente uma grande melhora de desempenho no uso da arquitetura do cenário 3. Também pode ser verificado que com um *thread* o tempo médio de abertura de uma conexão no Oracle é aproximadamente 80 % maior que o tempo médio de abertura de uma conexão no MS SQL Server; para os testes com cinco ou mais *threads* os tempos se tornam praticamente iguais.

O tempo de conexão no cenário 3 é sempre zero porque existem tantas conexões abertas quanto requisições HTTP que chegam ao Gerenciador de Conexões. Caso existam mais requisições HTTP concorrentes do que conexões abertas, haverá um *overhead* inicial associado à abertura da conexão, mas nas próximas requisições esse custo não ocorrerá mais.

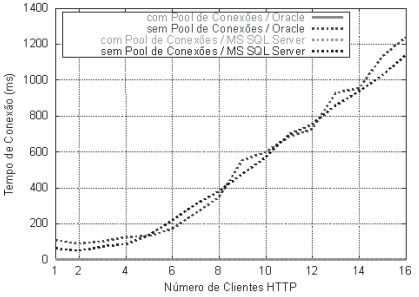
Os tempos de conexões para o cenário 1 não estão presentes no gráfico da Figura 2(a) porque eles são iguais aos tempos de conexão do cenário 2. Isso ocorre porque o fato de um pedido de conexão ser feito por um *script* Java padrão ou por um Servlet não influencia no tempo de abertura dessa conexão já que todos os recursos devem ser alocados da mesma forma.

Ambiente Perl

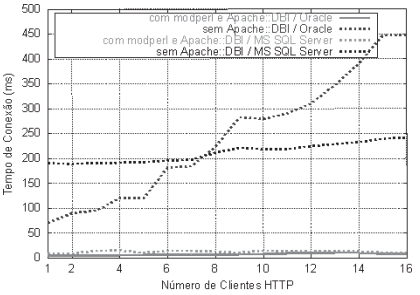
Como mostrado na Figura 2(b), o tempo de conexão foi quase totalmente eliminado devido à utilização do módulo Apache::DBI, reduzindo os tempos de conexão em torno de 10 ms. Os tempos de conexão no Oracle para testes realizados até 7 *threads* são muito menores que com MS SQL Server, a partir desse ponto os tempos para Oracle crescem quase linearmente e os tempos para MS SQL Server permanecem quase constantes. Isto é, com baixa carga os tempos de conexão no Oracle são melhores, já em períodos de alta carga o MS SQL Server se comporta de forma mais eficiente. O tempo de abertura de uma única conexão no MS SQL Server é aproximadamente 150 % maior que o tempo de conexão no Oracle.

Ambiente Python

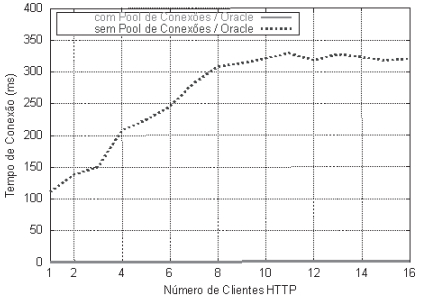
A Figura 2(c), mostra a eliminação do tempo de conexão no cenário 8 com a utilização do *pool* de conexões do DBCM.



(a) Cenários 2 e 3



(b) Cenários 5 e 6



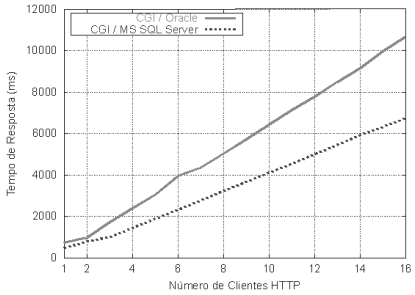
(c) Cenários 7 e 8

Figura 2 – Tempo de conexão x Número de clientes.

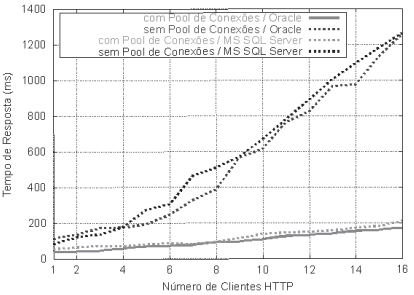
6.3.2 AVALIAÇÃO DO TEMPO DE RESPOSTA

Ambiente Java

Nos cenários 2 e 3, o tempo de resposta (t_{total}) é reduzido drasticamente em relação ao cenário 1 devido a três motivos. Primeiro, $t_{processo}$ é significativamente menor que no cenário 1 (*script* CGI padrão) devido a não ser necessário criar novos processos a cada requisição HTTP que é feita à Máquina Servlet. Segundo, sempre que chegar uma requisição, a Máquina Servlet (Apache JServ) já possui uma Máquina Virtual Java na memória, eliminando com isso o fator $t_{interpretador}$. Por último, após a primeira requisição feita ao servlet o seu código executável sempre estará na memória, eliminando dessa forma o t_{script}.



(a) Cenário 1



(b) Cenários 2 e 3

Figura 3 – Tempo de resposta número de clientes

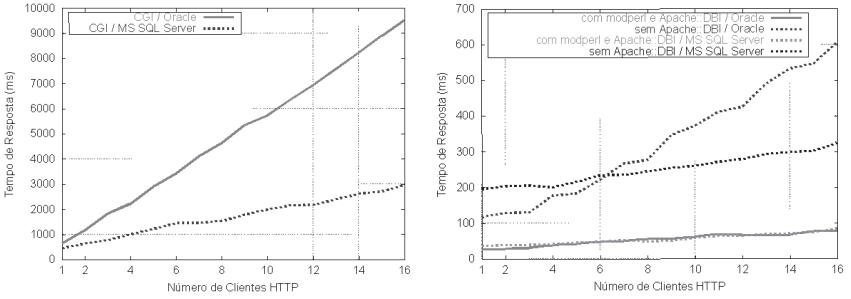
Analisando os gráficos das Figuras 2(a) e 3(b), podemos verificar que a grande redução no tempo de resposta no cenário 3 ocorre exclusivamente devido à eliminação do fator $t_{conexao}$ em relação ao cenário 2.

Devido à eliminação dos fatores: $t_{processo}$, $t_{intepretador}$ e t_{script}; podemos verificar através da Figura 3 que o tempo de resposta de uma requisição HTTP é muito menor nos cenários 2 e 3 do que no cenário 1. Também fica claro que a arquitetura do cenário 3 executa sistematicamente mais rápido que as arquiteturas dos cenários 1 e 2, aproximadamente 6 vezes mais rápido que a arquitetura Servlet padrão (cenário 2) e 55 vezes mais rápido que a arquitetura CGI/Java padrão (cenário 1) com Oracle e 35 vezes mais rápido que a arquitetura CGI/Java padrão com MS SQL Server.

De acordo com a Figura 3(a), o tempo de resposta de um *script* CGI/Java padrão acessando o Oracle é bem maior que o tempo de resposta para o mesmo *script* acessando o MS SQL Server. Isso ocorre porque o tempo de abertura de uma nova conexão no Oracle gasta muito mais recursos (memória) no *host* 2 do que o MS SQL Server e conseqüentemente mais tempo. Mesmo os SGBD's estando instalados em máquinas diferentes de onde o *script* é executado, o consumo de recursos nessa máquina é muito maior devido as chamadas de rotinas que o *driver* faz à biblioteca *Oracle Call Interface* (OCI), essas chamadas são então enviadas sobre o protocolo Net8 para o SGBD Oracle.

Ambiente Perl

A Figura 4, mostra a grande redução no tempo de resposta do cenário 6 em relação aos cenários 4 e 5; a melhora de desempenho é aproximadamente 7,5 mais rápido em relação ao cenário 5 com Oracle e 4 vezes mais rápido com MS SQL Server, em relação ao cenário 4 o tempo de resposta do cenário 6 é aproximadamente 120 vezes mais rápido com Oracle e 37,5 vezes mais rápido com MS SQL Server.



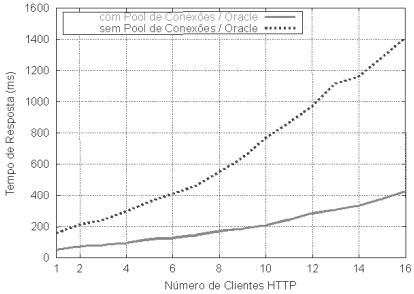
(a) Cenário 4 (b) Cenários 5 e 6

Figura 4 – Tempo de resposta x Número de clientes

Podemos verificar, na Figura 4(b), que o tempo de resposta no cenário 6 com Oracle e MS SQL Server são praticamente iguais, devido aos *pools* de conexões utilizados nos dois casos. No cenário 5 o tempo de resposta com Oracle é menor em relação ao MS SQL Server até os testes realizados com 6 *threads*, apartir deste ponto o tempo de resposta com MS SQL Server se torna bem menor em relação ao Oracle.

Ambiente Python

A Figura 5(a), mostra o efeito da utilização do DBCM em aplicações de banco de dados na *Web*, o tempo de resposta do cenário 8 é bastante reduzido, o cenário 8 executa a mesma requisição aproximadamente 3,5 vezes mais rápido que o cenário 7.



(c) Cenários 7 e 8

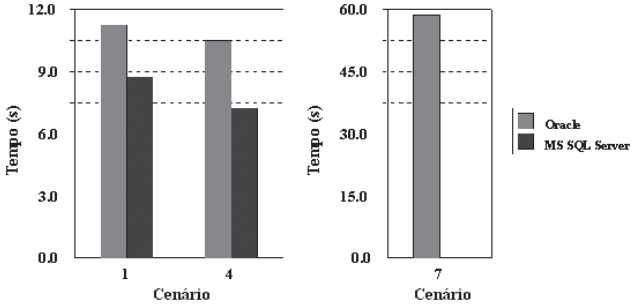
Figura 5 – Tempo de resposta x Número de clientes

6.3.3 AVALIAÇÃO DE THROUGHPUT

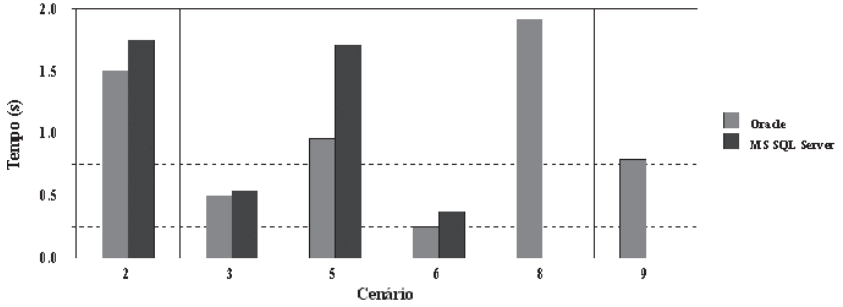
A Figura 6 mostra os tempos de *throughput* para os cenários, onde *throughput* representa o tempo total gasto para a execução de 16 requisições concorrentes de um mesmo *script*. Podemos ver através da Figura 6(a) que o *throughput* do cenário 4 é melhor que no cenário 1, isso ocorre devido à arquitetura Java consumir mais recursos na execução de um CGI padrão que a arquitetura Perl, o tamanho de um interpretador Java que deve ser carregado na memória toda vez que chega uma requisição é muito maior que o interpretador Perl, com isso é gasto mais tempo e o *throughput* aumenta. Em ambos cenários 1 e 4 o *throughput* para o MS SQL Server é melhor pois o processo de abertura de uma nova conexão no Oracle consome mais recursos que no MS SQL

Server, como nesses cenários já existe um grande consumo de recursos, esse fator torna o processo mais lento e em conseqüência o *throughput* com o MS SQL Server é melhor.

Na Figura 6(b) podemos ver que os cenários 3, 6 e 8 possuem *throughputs* melhores que os outros cenários, isto ocorre pois não existem custos associados à abertura de novas conexões. O melhor desempenho é do cenário 6 (Perl), em seguida o cenário 3 (Java) e por último o cenário 8 (Python). Em todos os cenários da Figura 6(b) o *throughput* com Oracle é melhor, isso ocorre porque apesar do Oracle consumir mais recursos para cada conexão ele consegue trabalhar com requisições concorrentes de forma mais eficiente que o MS SQL Server.



(a) Cenário 1 e 4



(b) Cenários 2, 3, 5, 6, 7 e 8

Figura 6 – Comparação do t*hroughput* entre os cenários

7 CONCLUSÃO

Neste artigo são descritas algumas API's de domínio público (Java Servlet, Perl e Python), seus benefícios no desenvolvimento de aplicações de bancos de dados no contexto da *Web* e feita uma comparação de desempenho destas API's. A técnica de conexão persistente foi utilizada no desenvolvimento de um agente de banco de dados. Esse componente reduz a taxa de estabelecimento de novas conexões com os SGBD's eliminando o *overhead* associado à abertura de novas conexões.

Vários testes foram realizados, a fim de quantificar os benefícios na utilização dessa técnica. Verificamos que nos testes realizados a implementação no ambiente Perl teve um melhor tempo de resposta em relação a Java e Python, e Java foi melhor que Python. Verificamos que o Oracle, no ambiente Java, gasta mais tempo que o MS SQL Server no processo de abertura de uma conexão devido à grande quantidade de recursos que devem ser alocados. No ambiente Perl, em períodos de baixa taxa de requisições concorrentes, o Oracle se comporta melhor, pois consegue servir os pedidos de conexões de forma mais eficiente. É possível, no entanto, que essas medidas sejam influenciadas pelos respectivos *drivers*, e não reflitam a eficiência intrínseca dos dois SGBD's.

COMPARISON OF APIS USED IN THE EFFICIENT ACCESS THE HETEROGENEOUS DATABASES ON THE WORLD WIDE WEB

ABSTRACT

The World Wide Web currently considered the ideal technological platform for the development of applications on the Internet and corporate intranets. This paper compares some open source APIs used to access heterogeneous databases. These APIs have often been used for the development of stateful database applications in the context of the Web. In these architectures, a persistent connection pool is used in order to eliminate the cost of establishing new connections to the DBMS's. The performance gain is assessed through a series of measurements in which we emulate the traffic caused by several concurrent HTTP clients. The APIs assessed were: Java Servlet (JDBC), Perl (DBI) and Python (Database API).

Keywords: WWW. RDBMS. Persistent connections. Java Servlet. Perl. Python.

8 REFERÊNCIAS

GUIMARÃES, C. C. A WWW SQL programming tool with persistent database connections. **Technical Report 18-00, IC-Unicamp**. Campinas: UNICAMP, 2000.

HADJIEFTHYMIADES, S.; MARTAKOS, D.; PETROU, C. Stateful relational database gateways for the World Wide Web. **Journal of Systems and Software**, v. 48, n. 3, 1999.

HADJIEFTHYMIADES, S.; VAROUXIS, I.; MARTAKOS, D. Performance of RDBMS-WWW interfaces under heavyWorkload. **Journal of Universal Computer Science**, v. 6, n. 6, 2000.

IYENGAR, A. Dynamic argument embedding: preserving state on the World Wide Web. **IEEE Internet Computing**, v. 1, n. 2, p. 50–56, Mar./Apr. 1997.

IYENGAR, A.; DIAS, D. Distributed virtual malls on the World Wide Web. IEEE International Conference on Distributed Computing Systems (ICDCS'98), 18., 1998, Amsterdam. [**Proceedings**…]. Amsterdam: IEEE Computer Society, 1998.

JAVA Apache Project. **Apache JMeter**. [2000?] Disponível em:<http://java.apache.org/imeter/index.html>.

KRISTOL, D. M.; LOU, M. HTTP state management mechanism. **RFC 2109**. [S.l.: s.n.], 1997.

LU, H.; LING, F. Integrating database and web technologies. **International Journal of World Wide Web**, v. 1, n. 2, p. 73–86, 1998.

QUAN, X.; LU, H.; LING, F. Supporting web-based database application development. INTERNATIONAL CONFERENCE ON DATABASE SYSTEMS FOR ADVANCED APPLICATIONS (DASFAA'99), 6., 1999, Taiwan. [**Proceedings**…]. Hsinchu: IEEE Computer Society, 1999. p. 17–26.

ZHAO, W. A study of web-based application architecture and performance measurements. Australian World Wide Web Conference (AusWeb'99), 5., 1999, Lismore. [**Proceedings**…]. Lismore: UniServe News, 1999.

PARADIGMAS DE CONJUGAÇÃO DO PORTUGUÊS COMPUTACIONAL

SCIENTIA

ERLON PINHEIRO[1]
MILLENE LEÃO BORGES DA SILVA[2]

[1] Mestre em Informática pela UFES. Professor do Centro Universitário Vila Velha. E-mail: erlon@uvv.br.

[2] Aluna do Curso de Ciência da Computação do Centro Universitário Vila Velha. E-mail: mlbsilva@terra.com.br

Scientia Vila Velha (ES), v. 4, n. 1/2, **p. 103-115**, jan/dez. 2003

RESUMO

Descreve um paradigma de conjugação verbal da língua portuguesa, visando a sua implementação computacional. É apresentado em uma breve introdução dos principais conceitos de verbos. É também discutida a dificuldade de uma conjugação automática, que é tratar todos os verbos irregulares da língua portuguesa.

Palavras-chave: Processamento de linguagem natural. Lingüística computacional. Português computacional.

1 INTRODUÇÃO

Como Rich e Knicht (1994, p. 433) afirmam “A linguagem destina-se à comunicação sobre o mundo. E se conseguimos criar um modelo computacional de linguagem, teremos uma ferramenta poderosa para comunicação sobre o mundo”. E a área que fornece aos computadores a capacidade de entender e compor textos é uma subárea da Inteligência Artificial chamada Processamento de Linguagem Natural. E, para se entender um texto, a principal palavra a ser estudada é o verbo.

De acordo com Paschoalin (1989), verbo é a palavra que expressa ação, estado e fenômeno da natureza situados no tempo, o que caracteriza a importância do verbo na formação de frases em Linguagem Natural. Conjugar um verbo, então, é dizê-lo em todos os números, pessoas, modos, tempos e aspectos.

No contexto computacional surge uma série de problemas. Por exemplo, é inviável armazenar todas as palavras geradas nas conjugações dos verbos em base de dados. Uma solução para esse problema é criar mecanismos que gerem a conjugação do verbo em tempo de execução. A criação de tais mecanismos é facilitada pois a maior parte (mais de 99%) são bastante regulares, com regras bastante definidas. Porém, existem verbos que não “respeitam” tais regularidades, dificultando a conjugação automática e forçando um tratamento particular.

2 FUNDAMENTAÇÃO TEÓRICA

Nas formas verbais existem três informações o radical, a vogal temática e as desinências.

O radical é a parte que contém a significação básica do verbo, que indica o significado do verbo. Ele permanece inalterado na conjugação dos verbos regulares, mas pode alterar bastante nos verbos irregulares.
Exemplo:

am- é o radical do verbo **amar.**
beb- é o radical do verbo **beber.**
part- é o radical do verbo **partir.**

A vogal temática indica a que conjugação verbal pertence o verbo. Na língua portuguesa existem três conjugações: 1ª conjugação (verbos terminados em **a**), 2ª conjugação (verbos terminados em **e**) e 3ª conjugação (verbos terminados em **i**).

As desinências verbais indicam as flexões de número, pessoa, tempo e modo.

Por exemplo:

beb- e – **sse- mos**
beb- é o radical
-e- é a vogal temática.
-sse- é a desinência que indica o modo subjuntivo e o tempo imperfeito.
-mos é a desinência que indica primeira pessoa e número plural.

Quando o verbo é conjugado ocorre uma divisão básica em radical e terminações.

Exemplo:

Radical Parte que contém a significação básica do verbo.	**Terminações** Parte variável do verbo.
escrev	o
escrev	es
escrev	e
escrev	emos
escrev	eis
escrev	em

As terminações indicam a pessoa, o número, o tempo e o modo.
São três as **pessoas** na conjugação:

a) primeira pessoa: a que fala, representada no singular pelo pronome pessoal **eu,** e no plural, **nós.**

b) segunda pessoa: a pessoa com quem se fala, representada no singular pelo pronome pessoal **tu,** e no plural, **vós.**

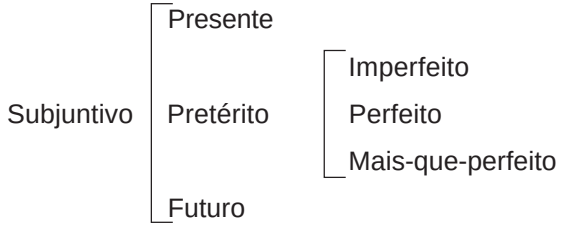
c) terceira pessoa: a pessoa de quem se fala, representada no singular pelo pronome pessoal **ele /elas,** e no plural, **eles/elas.**

O verbo apresenta flexão de **número** e pode estar no singular ou no plural. No singular, quando se refere a um ser único, e no plural, quando se refere a mais de um ser.

Existem três modos na língua portuguesa – indicativo, subjuntivo, imperativo- , além das formas nominais: infinitivo (pessoal e impessoal), gerúndio e particípio.

O fato expresso pelo verbo aparece sempre situado nos tempos: presente, passado e futuro. Mas existem várias localizações do fato no tempo, como demonstrado no Quadro 1.

- Indicativo
  - Presente
  - Pretérito
    - Imperfeito
    - Perfeito
    - Mais-que-perfeito
  - Futuro
    - do presente
    - do pretérito



Quadro 1 – Modos, formas nominais e tempo

Voz é a maneira como se apresenta a ação pelo verbo em relação ao sujeito. São três as vozes do verbo:

a) Ativa: O sujeito é o agente da ação verbal, ou seja, é ele quem pratica a ação.

Exemplo: O garoto **(sujeito agente)** | quebrou **(voz ativa)** o copo.

b) Passiva: O sujeito é paciente, isto é, ele recebe, sofre a ação expressa pelo verbo.

Exemplo: O copo **(sujeito paciente)** | foi quebrado **(voz passiva)** pelo garoto.

c) Reflexiva: O sujeito é, ao mesmo tempo, agente e paciente da ação verbal, isto é, pratica e sofre a ação expressa pelo verbo.

O garoto **(sujeito agente e paciente)** | machucou-se **(voz reflexiva)**.

2.1CLASSIFICAÇÃO DOS VERBOS

2.1.1 VERBOS REGULARES

São conjugados de acordo com as respectivas conjugações. Na conjugação de um verbo regular o radical não sofre alteração alguma e as terminações seguem uma regularidade nos diferentes modos e tempos.

1ª conjugação Cantar radical: cant-		2ª conjugação Beber radical: beb-		3ª conjugação Partir radical: part-	
cant	O	beb	o	part	O
cant	As	beb	es	part	ES
cant	A	beb	e	part	E
cant	amos	beb	emos	part	imos
cant	ais	beb	eis	part	Is
cant	am	beb	em	part	EM

2.1.2 VERBOS IRREGULARES

São irregulares os verbos que sofrem alterações no radical, nas desinências ou em ambas.
Exemplos:

Ferir: radical fer-		Dar: radical d-		Trazer: radical traz-	
Fir	o	d	**ou** **troux**		**e**

Existem alguns verbos que apresentam profundas alterações, podemos destacar os seguintes verbos : dar, estar, ser, haver, ter, ver, vir, ir, rir, pôr, ler e crer, saber e caber, querer, poder, dizer, fazer e trazer .

Por outro lado existem verbos que são irregulares, mas podem “obedecer” a algumas regras, como as citadas por Padilha, Winchler e Nemetz (2003), que veremos a seguir:

a) Alterações em que a letra final do radical é vogal:

- verbos terminados em **oer** (doer,moer,...): no presente do indicativo recebem acento (*d**ô**o*, *d**ó**is*, *d**ó**i*, *d**ó**em*).

- verbos terminados em **oar** (doar, perdoar,..): acento em (eu) *perdôo* por causa do duplo "o";
- verbos terminados em **ear** (basear, passear,..): nos dois presentes ganham um "i" na junção do radical à desinência (*base**i**o*, *base**i**as*,.. *base**i**am*, *base**i**e*,.. *base**i**em*).
- estrear: além do "i" (como o caso acima), recebe acento no "e" adjacente, por causa do ditongo aberto (*estr**éi**o*, *estr**éi**as*,.. *estr**éi**am*, *estr**éi**e*,.. *estr**éi**em*).
- verbos terminados em **air** (cair, sair...): vogal temática "i" aparece inadvertidamente em (eu) *ca**i**o* e no presente do subjuntivo (*ca**i**a*, *ca**i**as*, *ca**i**amos*, *ca**i**ais*, *ca**i**am*).
- verbos terminados em **guar**/**quar** (averiguar,..): recebem acento no "u" nos dois presentes (*averig**u**o*, *averig**u**as*,.. *averig**u**e*,.. *averig**u**em.)*
- verbos terminados em **güir**/**qüir** (arqüir,...): nas formas acentuadas, exatamente como o caso anterior (*arg**u**o*, *arg**u**is*,.. *arg**u**em*, *arg**u**a*,.. *arguam)*

b) Alterações quando o final do radical é consoante:

- verbos terminados em **guir** (distiguir,..): "gu" vira "g" quando adjacente às vogais a/o (*disting**o*, *disting**a*,.. *disting**amos*, *disting**ais*,..).
- verbos terminados em **guer** (erguer,..): mesmo caso, "gu" vira "g" quando adjacente às vogais a/o (*er**g**o*, *er**g**a*,.. *er**g**amos*, *er**g**ais*,..).
- verbos terminados em **gar** (carregar,..): caso oposto aos anteriores, "g" vira "gu" quando junto a "e" (*jul**gu**ei*, *jul**gu**e*,.. *jul**gu**emos*, *jul**gu**eis*,..).
- verbos terminados em **ger** (proteger,...): "g" vira "j" quando unido às vogais a/o (*prote**j**o*, *prote**j**a*, *prote**j**as*, *prote**j**amos*,..).
- verbos terminados em **gir** (agir,..): mesmo caso, "g" vira "j" quando adjacente às vogais a/o (*a**j**o*, *a**j**a*, *a**j**as*,.. *a**j**am*).
- verbos terminados em **car** (ficar,..): "c" vira "qu" quando junto a "e" (*fi**qu**ei*, *fi**qu**e*,.. *fi**qu**emos*, *fi**qu**eis*,..).
- verbos terminados em **çar** (forçar,...): "ç" vira "c" quando junto a "e" (*for**c**ei*, *for**c**e*,... *for**c**emos*, *for**c**eis*,...).

- verbos terminados em **cer** (esquecer,..): caso oposto ao de cima, "c" vira "ç" quando adjacente às vogais a/o (*esque**ç**o*, *esque**ç**a*,.. *esque**ç**amos*, *esque**ç**ais*,..).
- verbos terminados em **zir** (conduzir,..): 3a. pessoa do singular do presente perde a desinência "e" normal dessa flexão (*conduz*, em vez de *conduz**e***).

c) Verbos que exigem indicação especial:

- ansiar, incendiar, mediar e odiar (e derivados): conjugam-se como se fossem terminados em **ear**, (*ans**e**io*, *ans**e**ia*,.. *ans**e**iam*, *ans**e**ie*,..), nas formas rizotônicas.
- construir e destruir (outros verbos com raiz semelhante, instruir e obstruir são regulares): "u" vira "ó" em (tu) *constr**ó**is*, (ele) *constr**ó**i*. (obs. Mas podem também manter o se – construis, destruis...)
- medir, pedir (e derivados: despedir, expedir, impedir e desimpedir): "d" vira "ç" em (eu) *pe**ç**o* e no presente do subjuntivo (*pe**ç**a*, *pe**ç**as*, *pe**ç**amos*, *pe**ç**ais*, *pe**ç**am*).
- ouvir: "v" vira "ç" em (eu) *ou**ç**o* e no presente do subjuntivo (*ou**ç**a*, *ou**ç**as*, *ou**ç**amos*, *ou**ç**ais*, *ou**ç**am*).
- perder: "d" vira "c" em (eu) *per**c**o* e no presente do subjuntivo (*per**c**a*, *per**c**as*, *per**c**amos*, *per**c**ais*, *per**c**am*).
- valer (e derivados): "l" vira "lh" em (eu) *va**lh**o* e no presente do subjuntivo (*va**lh**a*, *va**lh**as*, *va**lh**amos*, *va**lh**ais*, *va**lh**am*).

d) Verbos com acento no radical:

- aquar (derivados: desaguar, enxaguar,..): algumas formas são proparoxítonas e, portanto, acentuadas (no "a" do radical).
- minguar, proibir,..: exatamente como o caso anterior, só que a vogal acentuada agora é "i":
- apoiar, boiar,.. (provavelmente todos verbos terminados em **oiar**): "o" é acentuado.
- reunir: "u" é acentuado.

e) Verbos com particularidades na acentuação:

- parar: forma (ele) *pára* é acentuada para evitar confusão com a preposição **para**.
- pelar: formas (eu) *pélo*, (tu) *pélas*, (ele) *péla* são acentuadas para evitar confusão com as contrações **pelo**, **pelas** e **pela**.
- coar: formas (eu) *côo*, (tu) *côas*, (ele) *côa* são acentuadas para distingui-las das contrações, agora em completo desuso, **coo** (com+o), **coas** (com+as) e **coa** (com+a).

2.1.3 VERBOS DEFECTIVOS

Os verbos defectivos são aqueles que não possuem conjugação completa. Esses verbos são divididos em:

a) Impessoais: Surgem apenas na terceira pessoa do singular. Nesse caso os verbos indicam fenômenos da natureza (chover, ventar,...), o verbo haver, quando indica existência, e o verbo fazer, na indicação de tempo transcorrido.

b) Unipessoais: São conjugados apenas na terceira pessoa do singular e do plural. São os verbos que exprimem vozes de animais (miar, latir,...) e que exprimem ocorrência (ocorrer, acontecer, suceder,...).

c) Pessoais: não possuem algumas flexões por motivos formais. Serão apresentados três grupos para esses verbos.

1º grupo: são os verbos que não possuem a primeira pessoa do indicativo, não possuem presente do subjuntivo e imperativo negativo, e no afirmativo, possuem algumas formas.

Estão nesse grupo os seguintes verbos: abolir, banir, carpir, colorir, delinqüir, exaurir, fremir, fulgir, haurir, retorquir.

2º grupo: Esses verbos são conjugados apenas quando o radical é seguido por i. Não possuem presente do subjuntivo e imperativo negativo, e no imperativo afirmativo apresentam somente a primeira pessoa do plural.

Estão nesse grupo os verbos: aguerrir, combalir, embair, forargir-se, remir, renhir.

3º grupo: Estão nesse grupo adequar, precaver, reaver.

2.1.4 VERBOS ABUNDANTES

Os verbos abundantes são aqueles que apresentam mais de uma forma para uma mesma flexão. O verbo haver, por exemplo, apresenta duas formas para a primeira pessoa do plural do presente do indicativo: **havemos** ou **hemos.** Entretanto, a abundância ocorre com freqüência no particípio, como por exemplo, **aceito/aceitado, entregue/entregado.**

2.1.5 VERBOS AUXILIAR

Quando se junta a uma forma nominal de um verbo principal, perdendo com isso sua significação. Os auxiliares mais comuns são: **ser, estar e haver**.

Exemplo:

Tenho comprado frutas.

3 PORTUGUÊS COMPUTACIONAL

Existem diferenças importantes entre radical lingüístico e o radical computacional (ROCHA; CAMELO; LINS, 1999). O Primeiro, como vimos na seção 2, é a parte principal do verbo, que contém o sentido da palavra. E o radical computacionalmente interpretado tem a seguinte definição: “O radical é a parte do lexema [conjunto de palavras que se distinguem através da flexão] que se repete em todas as suas variações.”

Com esta definição podemos agrupar os verbos em modelos, por exemplo, os verbos regulares terminados em **ar** formam um modelo, mas os verbos com uma “particularidade”, como, por exemplo, o verbo **parar**, que na terceira pessoa do singular do presente de indicativo e na segunda pessoa do singular do imperativo afirmativo recebem acento no **a** (ele p**á**ra, p**á**ra (tu), respectivamente), são agrupados em outro modelo.

A representação computacional se dá pela construção de tais modelos de conjugação, por uma lista de radicais computacionais e de uma relação entre o radical e o modelo. Apresentamos no Quadro 2 um exemplo de modelo computacional dos verbos medir, pedir (e derivados: despedir, expedir, impedir e desimpedir), em cujo radical **d** se transforma em **ç** na primeira pessoa do singular do presente do indicativo e nas

formas daí derivadas. Cada forma verbal é obtida pela junção do radical computacional e a respectiva flexão.

Observe a diferença dos radicais (lingüísticos e computacional) nos verbos do modelo, por exemplo:

Radical lingüístico do verbo medir: **med**.

Radical computacional do verbo medir: **me** (a única parte do verbo que se repete durante toda a conjugação).

Indicativo					
Presente	**Pret. Imperfeito**	**Pret. Mais-que-perfeito**	**Pret. Perfeito**	**Fut. do Presente**	**Fut. do Preterito**
-ço	-dia	-dira	-di	-direi	-diria
-des	-dias	-diras	-diste	-dirás	-dirias
-de	-dia	-dira	-diu	-dirá	-diria
-dimos	-díamos	-díramos	-dimos	-díramos	-díriamos
-dis	-díeis	-díreis	-distes	-díreis	-diríeis
-dem	-diam	-diram	-diram	-diram	-diriam

Subjuntivo		
Presente	**Imperfeito**	**Futuro**
-ça	-disse	-dir
-ças	-disses	-dires
-ça	-disse	-dir
-çamos	-díssemos	-dirmos
-çais	-dísseis	-dirdes
-çam	-dissem	-direm

Imperativo	
Afirmativo	**Negativo**
-de	-ças
-ça	-ça
-çamos	-çamos
-di	-çais
-çam	-çam

Gerúndio
-dindo

Particípio
-dido

Infinitivo	
Pessoal	**Impessoal**
-dir	-dir
-dires	
-dir	
-dirmos	
-dirdes	
-direm	

Quadro 2 – Flexões do verbo e seus derivados

Esse modelo é baseado na conjugação computacional de Rocha, Camelo e Lins (1999) que não traz a conjugação dos verbos na voz passiva, ao contrário da de Guedes e Guedes (1994) que apresenta os modelos em ambas vozes, o que pode ser mais complicado organizar computacionalmente, pois os verbos auxiliares também precisariam ser tratados em toda a conjugação da voz passiva.

Alguns autores pesquisados apresentam uma lista de modelos de conjugação de verbos em português bastante sucinta. Guedes e Guedes (1994) trazem 115 modelos de conjugação. Por sua vez, Terra e Nicola (1999) trazem 80 modelos.

4 CONSIDERAÇÕES FINAIS

A área de Processamento de Linguagem Natural é fundamental no aperfeiçoamento da interação humano-computador. Nesse contexto, a conjugação automática de verbos assume papel fundamental. Discutimos nesse artigo alguns dos problemas computacionais envolvidos na conjugação automática de verbos da Língua Portuguesa.

Um sistema de conjugação de verbos é fundamental para sistemas de processamento de linguagem como o ambiente de apoio à tradução descrito em Pinheiro (2002).

Utilizando o modelo teórico de conjugação do Português computacional apresentado (e que será ampliado) é possível implementar um conjugador verbal eficiente para a Língua Portuguesa. Tal sistema será implementado como projeto final de graduação no curso de Ciência da Computação a ser defendido pela co-autora.

CONJUGATION PARADIGMS OF THE COMPUTATIONAL PORTUGUESE

ABSTRACT

This article describes a paradigm of verbal conjugation of the Portuguese language, aiming at its computational implementation. It is presented in one brief introduction of the main concepts of verbs. The also argued the difficulty of an automatic conjugation, that is to deal with all the irregular verbs the Portuguese language.

Keywords: Natural language processing. Computational linguistics. Computational portuguese.

5 REFERÊNCIAS

GUEDES, A. M.; GUEDES, R. **Dicionário prático de conjugação dos verbos da Língua Portuguesa**. Venda Nova: Bertrand, 1994.

PADILHA, E. G.; WINCKLER, M. A.; NEMETZ, F. **Conver**. Disponível em: <http://www.inf.ufrgs.br/~emiliano/conver/>http://www.inf.ufrgs.br/~emiliano/conver/. Acesso em: 4 maio 2003.

PASCHOALIN, M. A. **Gramática**: teoria e exercícios. São Paulo: FTD, 1989.

PINHEIRO, E. **AAT**: um ambiente de apoio à tradução de textos. 2002. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, 2002.

RICH, E.; KNICHT, K. **Inteligência artificial**. São Paulo: Makron Books, 1994.

ROCHA, C. B.; CAMELO, H. A. L.; LINS, R. **Conjugações verbais no Português Computacional**. Recife: UFPE, Departamento de Informática,1999.

TERRA, E.; NICOLA, J. de. **Verbos**: guia prático de emprego e conjugação. São Paulo: Scipione, 1999.

UM MÉTODO PARA ANÁLISE E PROJETO DE SISTEMAS DE *WORKFLOW* ADMINISTRATIVO COM INTERFACE PARA A WEB

SCIENTIA

LUDIMILA MONJARDIM CASAGRANDE[1]

[1] Mestre em Ciência da Computação pela USP. Analista de sistemas do Instituto de Pesquisas Eldorado. E-mail: ludimila.casagrande@eldorado.org.br.

RESUMO

Apresenta um método para análise e projeto de sistemas de workflow administrativo com interface para a web. O processo de desenvolvimento proposto baseia-se na integração de métodos e técnicas preexistentes, específicas para modelar sistemas de workflow e aplicações hipermídia em geral. As técnicas integradas são: a UML, que é a notação padrão para a modelagem de sistemas orientados a objetos, amplamente aceita pela comunidade de engenharia de software, o BPI, que consiste em um processo direcionado especificamente para o desenvolvimento de sistemas de workflow; o RMM, que consiste na primeira metodologia estendida com o intuito de modelar aplicações hipermídia para a web, e o W2000, que consiste em um framework para o projeto de aplicações para a web baseado no HDM — o antecessor de uma família formada por vários métodos que apóiam o projeto de aplicações hipermídia. Para testar o método proposto, foi desenvolvido um sistema para administração de cursos pela web. Esse sistema, denominado Atena, tem características de administração de workflows e foi usado inicialmente como base para o desenvolvimento do método e, em um momento posterior, como base para testar a abrangência e adequação do método ao domínio proposto.

Palavras-chave: Sistemas de informação. Web. Workflow. UML. W2000.

1 INTRODUÇÃO

Uma tendência que pode ser observada, sob o ponto de vista tecnológico, é o fato de que cada vez mais desenvolver um software é sinônimo de desenvolver uma solução baseada na plataforma Web (*Web-based solution*). No entanto, a própria WWW e, conseqüentemente, os sistemas de informação para a Web ou *Web Information Systems* (WISs) ainda são muito recentes e, por essa razão, as técnicas específicas para o projeto dessa categoria de sistemas ainda são escassas ou imaturas e muitas vezes apresentam limitações. Por outro lado, os métodos e as notações utilizados para o projeto dos sistemas de informação convencionais não são adequados para a completa especificação dos requisitos e de características particulares dos sistemas de informação para a Web. Esses são, portanto, os principais fatores que justificam e motivam o desenvolvimento de pesquisas, sobretudo

nas áreas de Engenharia de Software e de Sistemas Hipermídia, direcionadas para a elaboração de modelos, métodos, notações e ferramentas que apóiem a análise, a especificação e o projeto de softwares baseados na Web.

Outra tendência a ser destacada é a implementação de fluxos de trabalho ou *workflows* na Web. A plataforma Web favorece a automatização de *workflows* por uma série de razões. Em primeiro lugar está o potencial de integração de pessoas, repositórios de dados, dispositivos e sistemas proporcionado pela WWW. Os recursos oferecidos pela Web também favorecem a comunicação entre os participantes do *workflow*, a construção de ambientes de trabalho colaborativo e de comunidades de conhecimento, além de serem excelentes meios de difusão de informações e documentos. A construção de hiperbases contribui para uma melhor organização e estruturação do conteúdo armazenado pelos sistemas. Dessa forma, a informação permanece contextualizada e de fácil acesso.

Sob tais perspectivas, existe um consenso de que os sistemas baseados em fluxos de trabalho com interface para a Web podem oferecer vantagens significativas aos seus usuários. No entanto, o seu desenvolvimento ainda é realizado sem nenhum ou com o mínimo de planejamento, o que afeta a qualidade e a confiabilidade dos softwares produzidos.

Dessa forma, o principal objetivo deste artigo é apresentar um método para a análise e o projeto de sistemas de *workflow* administrativo com interface para a Web, que visa a minimizar o problema da falta de recursos para o desenvolvimento controlado e planejado dessa categoria de sistemas. Para dar suporte a esse trabalho foi desenvolvido o Sistema Atena, que permitiu a identificação das características específicas e das relações existentes entre os sistemas de informação para a Web e os sistemas de *workflow*. Em um momento posterior, esse sistema também foi utilizado como base para testar e validar o método proposto.

Na Seção 2 deste artigo são descritas as principais atividades do fluxo de trabalho implementado pelo Sistema Atena. Na Seção 3 é apresentada uma visão geral das fases que compõem o método aqui proposto. Na Seção 4 essas fases são descritas mais detalhadamente e exemplos baseados no Sistema Atena são usados para elucidar as técnicas

de análise e projeto aplicadas. Por fim, na Seção 5 são apresentadas as conclusões e as contribuições deste artigo.

2 ESTUDO DE CASO: SISTEMA ATENA

O sistema escolhido para dar suporte à elaboração e exemplificar a aplicação do método implementa um processo de administração de cursos de extensão. Essa escolha justifica-se pela possibilidade que o sistema oferece de integrar a aplicação de um sistema de informação para Web à aplicação de um sistema que implementa um fluxo de trabalho administrativo bem definido, permitindo a exploração simultânea dessas duas técnicas. Nesta seção somente são apresentadas as partes principais do Sistema Atena que, de fato, são relevantes para a especificação do *workflow*.

As primeiras atividades que devem ser apoiadas pelo sistema proposto são a elaboração da proposta do curso, a criação de um processo referente a essa proposta e a sua aprovação. A proposta do curso deve ser elaborada pelo docente responsável por sua realização, que deve informar os dados gerais do curso e quem irá ministrá-lo. Essa proposta é encaminhada ao Assistente Acadêmico, que abre um processo para registrar a sua existência e para dar início ao processo de aprovação. A nova proposta é, então, submetida à aprovação do Conselho Departamental e da Comissão de Cultura e Extensão Universitária (CCEx) da Unidade. Caso a proposta tenha sido aprovada nas duas instâncias, o curso é divulgado e são abertas as inscrições na Web para os interessados em cursá-lo. Pessoas interessadas em participar do curso podem preencher o formulário de inscrição e encaminhá-lo para que seus pedidos de inscrição sejam aprovados. O docente responsável pelo curso recebe esses pedidos e aprova ou não a sua solicitação. Quando a inscrição de uma pessoa é aprovada, ela deixa de assumir o papel de candidato e passa a assumir o papel de aluno e usuário do sistema. Para isso, recebe uma identificação e uma senha de acesso que são atribuídas pelo docente. Se o docente não aprovar os candidatos até uma data-limite, os pedidos de inscrição são aceitos, de acordo com a ordem de chegada, até que todas as vagas tenham sido preenchidas. Durante a realização do curso, a freqüência do aluno deve ser controlada pelo docente e armazenada pelo sistema. Os resultados obtidos pelos alunos também devem ser registrados para que, concluído o curso, possam ser emitidos os certificados daqueles que cumpriram as

exigências requeridas. Ao término do curso, o docente deve solicitar o cálculo das médias e das freqüências finais dos seus alunos e enviar essas informações para a Diretoria da Unidade. Em seguida, o Diretor encaminha à Pró-Reitoria de Cultura e Extensão Universitária o demonstrativo do aproveitamento obtido pelos alunos do curso em referência e solicita a autorização para a emissão dos certificados. O Pró-Reitor recebe e avalia essa solicitação e encaminha o processo à Assistência Acadêmica com a autorização para a emissão dos certificados. Esses são, então, emitidos pelo próprio sistema e entregues aos alunos aprovados. Após a emissão de certificados, o Assistente Acadêmico arquiva o processo.

3 O PROCESSO DE DESENVOLVIMENTO

Nesta seção é apresentada uma breve descrição das fases que compõem o processo de desenvolvimento de sistemas de *workflow* com interface para a Web, das interdependências entre elas e dos resultados obtidos com a sua execução. Cada etapa produz um modelo, isto é, um conjunto de um ou mais diagramas, que descreve formalmente os aspectos da aplicação focalizados naquela etapa e documenta as atividades realizadas.

Análise e Modelagem de Requisitos: Durante esta primeira etapa é realizada a análise e a modelagem dos requisitos que definem as funcionalidades e metas do sistema. No caso de sistemas com interface para a Web, a análise de requisitos deve ser estendida para especificar também os requisitos navegacionais e de visibilidade. Sendo assim, esta fase consiste na definição de atores ou papéis, na análise de requisitos funcionais e na análise de requisitos navegacionais da aplicação.

Modelagem Conceitual: A modelagem conceitual é representada por um modelo de classes e define a estrutura estática das informações ou conceitos envolvidos pelo sistema, em termos de classes de objetos e dos relacionamentos ou associações existentes entre elas. Esse modelo também define as operações e os atributos que caracterizam cada objeto.

Projeto de Fatias: O projeto de fatias determina como as informações devem ser apresentadas aos usuários. Para isso, os atributos de diferentes entidades são agrupados em fatias significativas de informação

que são organizadas na forma de um hipertexto. Esse é o primeiro passo para a especificação da interface dos sistemas para a Web.

Modelagem de *Workflows*: Esta fase tem como base os requisitos funcionais especificados na fase de análise e modelagem de requisitos. Durante esta etapa é definido o comportamento do sistema em termos das seqüências nas quais as atividades devem ser realizadas e das pré e pós-condições para a execução de cada tarefa, ou seja, é nesta etapa que são especificados os *workflows* abrangidos pelo sistema.

Projeto Navegacional: O propósito do projeto navegacional é definir como os usuários podem navegar pela informação e quais estruturas são utilizadas para o acesso às páginas que compõem o sistema. Ainda nesta fase, são especificadas as diferentes visões do sistema.

Implementação: A implementação é a fase na qual é realizada a conversão dos modelos elaborados durante todas as fases do processo de desenvolvimento apresentadas para os programas codificados em uma linguagem legível por computador.

4 FASES DO PROCESSO DE DESENVOLVIMENTO

4.1 ANÁLISE E MODELAGEM DE REQUISITOS

A análise e modelagem de requisitos deve ser baseada em um documento de requisitos elaborado preliminarmente.

O propósito desta fase é compreender as metas, o comportamento e o domínio da aplicação, por meio do detalhamento dos requisitos identificados. Como resultado desta etapa, deve-se obter uma abstração funcional do sistema expressa por diagramas de casos de uso (*use cases*) da UML. Apesar de definir claramente o conceito de caso de uso, a UML não especifica um formato para a sua descrição detalhada, que deve ser realizada para o melhor entendimento do sistema e dos processos por ele envolvidos. Dessa forma, o formato padrão proposto por Larman (1997) foi adotado para a descrição formal dos casos de uso. Tal formato é composto por três partes. A primeira contém uma descrição sumária do caso de uso, na segunda são especificados os progressos típicos do caso de uso e na terceira são apresentados os progressos alternativos.

A fase de análise e modelagem de requisitos deve ter início com a definição dos atores que participam do *workflow*. Essa atividade facilita a identificação das funcionalidades do sistema e ajuda a determinar e organizar os caminhos de navegação. No caso do Sistema Atena, os atores são: o Assistente Acadêmico, os Docentes, o Chefe Departamental, o Presidente da CCEx, o Diretor da Unidade, o Pró-Reitor de Cultura e Extensão, os Candidatos aos cursos, os Alunos e os Usuários em geral.

Identificados os atores ou papéis do sistema, é realizada a análise de requisitos funcionais, atividade na qual são identificadas as operações executadas pelos diferentes atores, as transformações de dados ocorridas e as reações do sistema às ações dos usuários. Desse modo, o propósito desta etapa é determinar as funções que cruzam os limites do sistema e atribuir essas funções aos atores responsáveis por sua ativação. Vale ressaltar que delegar tarefas e atribuir responsabilidades a classes de usuários com perfis bem definidos são ações exigidas pelos sistemas de *workflow*.

Na Figura 1 é apresentada a modelagem dos casos de uso funcionais do Sistema Atena. Note que nesse modelo o aluno não é incluído. Isso ocorre porque essa classe de usuário não executa nenhuma operação do sistema, apenas acessa informações particulares ou públicas por meio de consultas, o que não deve ser representado nesse modelo e sim no modelo navegacional.

No caso dos sistemas de *workflow* para a Web, a análise de requisitos deve ser estendida para abordar também os requisitos navegacionais e de visibilidade do sistema, conforme o proposto por Baresi, Garzotto e Paolini (2001) no *framework* W2000. Os requisitos navegacionais definem os conteúdos específicos de interesse e que podem ser acessados por cada perfil de usuário em particular. Nos sistemas de informação para a Web, é muito comum a existência de visões ou perspectivas da aplicação. A modelagem de visões agrupa informações, estruturas navegacionais e funcionalidades, gerando diferentes contextos de navegação.

A modelagem de casos de uso navegacionais é capaz de expressar todas as capacidades navegacionais associadas a cada tipo de ator. As estruturas de navegação, porém, somente são definidas no projeto de fatias e no projeto navegacional.

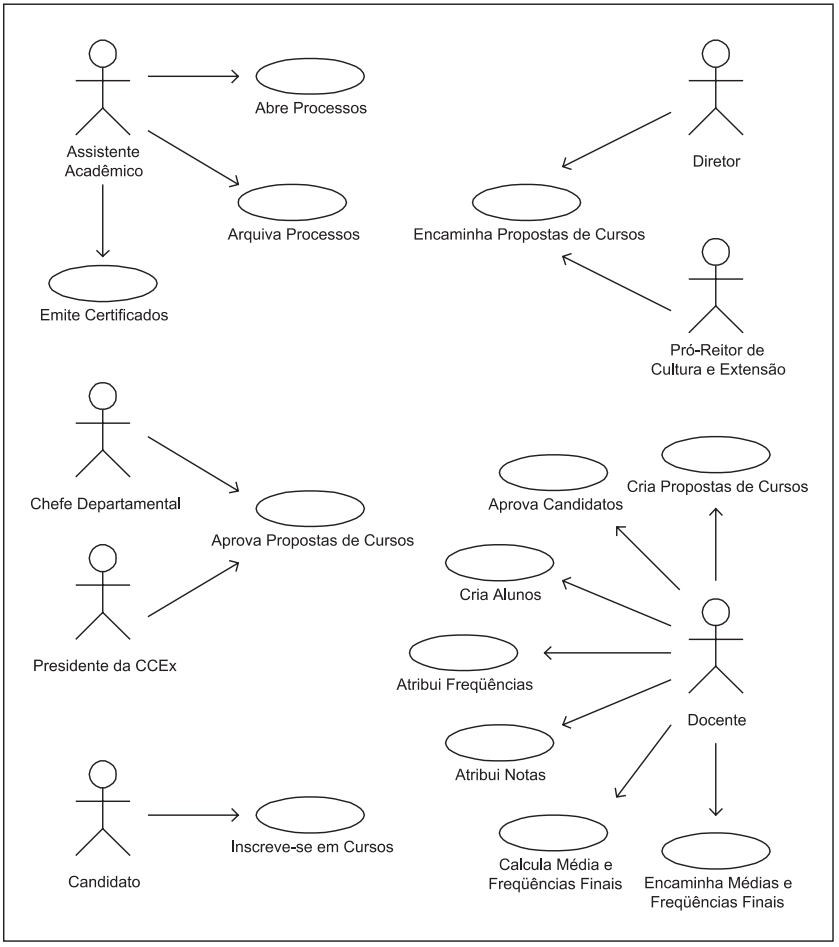


Figura 1 - Modelo parcial de caso de uso funcionais

Na Figura 2 é ilustrado o modelo de casos de uso navegacionais do Sistema Atena. Nesse modelo, o usuário genérico já está presente. Ele é capaz de acessar e navegar sobre todas as informações públicas do sistema. Nesse ponto, o aluno também é incluído e tem acesso às informações sobre notas e freqüências particulares e aos calendários de aulas e de avaliações dos cursos dos quais participa.

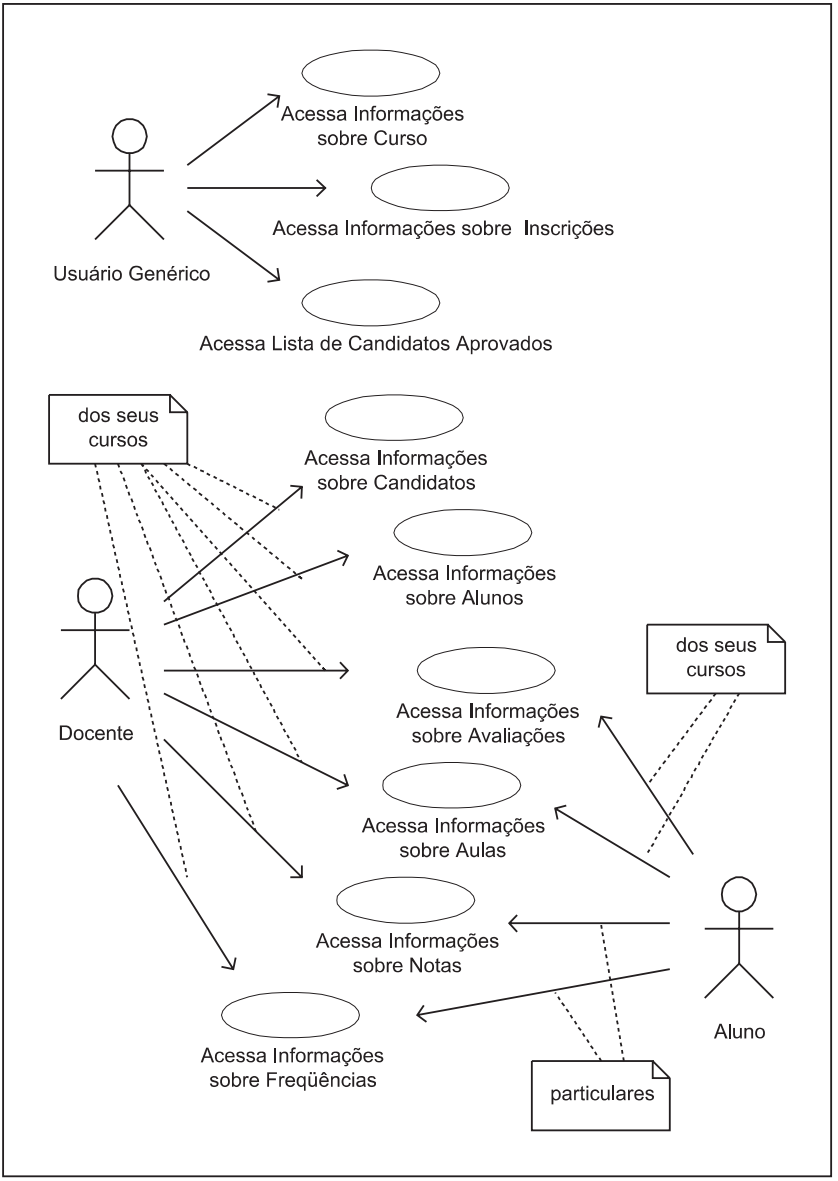


Figura 2 - Modelo parcial de caso de uso navegacionais

Esse é apenas o primeiro passo para a definição das visões ou contextos navegacionais. O aperfeiçoamento dessa atividade ocorre durante o projeto navegacional. As visões são importantes não só para os sistemas para a Web mas também para os sistemas de *workflow*, pois personalizam o sistema de acordo com as necessidades de cada perfil de usuário e restringem o acesso e a execução de serviços específicos aos atores competentes, em conformidade com as exigências do domínio da aplicação.

4.2 MODELAGEM CONCEITUAL

A modelagem conceitual representa a estrutura estática do sistema, isto é, nesta etapa são identificadas as classes de objetos que o compõem, os relacionamentos existentes entre essas classes e os atributos e as operações que as caracterizam. Essa modelagem está presente tanto nos ciclos de desenvolvimento clássicos dos sistemas tradicionais como nos métodos para especificação de aplicações hipermídia, seja na forma de modelos de entidades e relacionamentos ou na forma de modelos de classes ou modelos de objetos. No método proposto por este trabalho, sugere-se o uso da modelagem orientada a objetos.

A modelagem orientada a objetos é vantajosa porque é capaz de expressar importantes características do domínio da aplicação, como a abstração, o encapsulamento e o compartilhamento de operações e atributos por meio da herança. Tais características permitem especificar os componentes do sistema de maneira abstrata, ocultando detalhes de implementação.

Para a especificação da estrutura estática é utilizado o modelo de classes da UML. Esse modelo incorpora, além de classes, atributos e operações, alguns outros conceitos importantes tais como: associação, generalização/especialização, agregação e dependência, que definem diferentes tipos de relacionamentos entre as classes.

A identificação dos componentes do modelo de classes deve ser realizada a partir da definição dos casos de uso. As classes de objetos podem representar, por exemplo, conceitos, porções de informações ou atores envolvidos pelo sistema, devendo as descrições dos casos de uso ser analisadas com o intuito de identificar elementos como esses e de determinar quais são os objetos que compõem o domínio da aplicação.

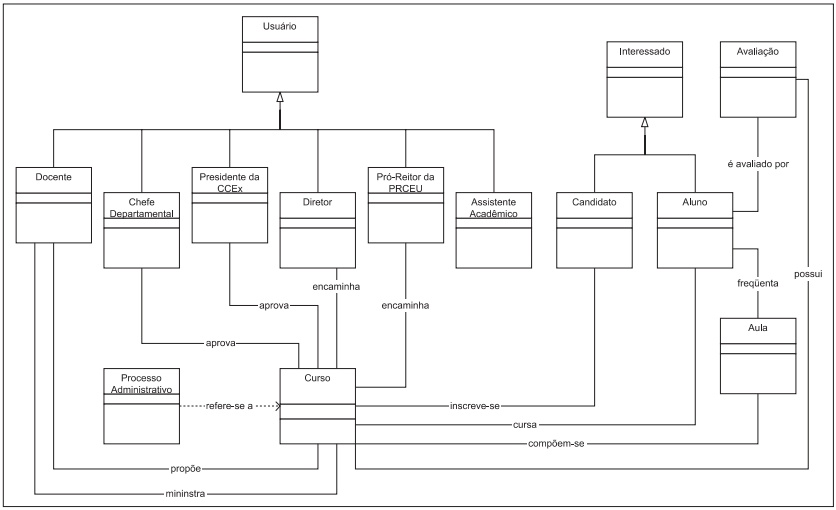


Figura 3 - Modelo parcial de classes do Sistema Atena

Na Figura 3 é ilustrado um modelo parcial de classes para o Sistema Atena. Observa-se nesse modelo a existência de uma relação de dependência entre a classe que representa os processos administrativos e a classe que representa os cursos. Isso ocorre porque um processo administrativo só existe quando e se existir o curso associado. Nesse modelo existem ainda exemplos de generalizações/especializações entre a superclasse "Usuário" e suas subclasses, cada qual representando uma categoria de atores do sistema.

4.3 O PROJETO DE FATIAS

O propósito do projeto de fatias é estruturar o conteúdo do sistema. Sendo assim, durante essa atividade, especificam-se como as informações devem ser apresentadas aos usuários. Para isso, os atributos de diferentes classes são agrupados em fatias ou porções significativas de informação, organizadas na forma de um hipertexto.

O projeto de fatias é o primeiro passo para o projeto de interfaces dos sistemas para a Web. Nessa etapa, é realizada a modelagem da parte estática da interface, isto é, daquilo que permanece inalterado, independentemente do estado do sistema.

Para o projeto de fatias, são utilizados os diagramas de *m-slices* do RMM estendido (ISAKOWITZ; KAMIS; KOUFARIS, 1997, 1998). Um *m-slice* define uma fatia de informação que agrupa atributos, possivelmente pertencentes a diferentes classes, e que estão associados a um mesmo contexto. Informações contextualizadas são melhor compreendidas e, portanto, reduzem a desorientação do usuário durante a navegação no sistema.

Para a realização do projeto de fatias, cada caso de uso deve ser analisado em termos dos elementos que uma interface deve conter para que ele possa ser implementado. Um *m-slice* pode aninhar fatias pertencentes à sua classe proprietária ou a outras classes, neste caso, o relacionamento existente entre essas classes e a classe proprietária deve ser explicitado. Os *m-slices* também podem incluir estruturas de acesso como *links* e índices, que normalmente estão associados a relacionamentos.

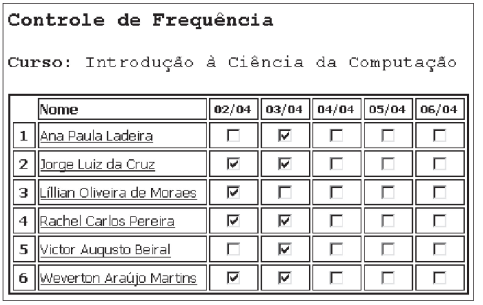


Figura 4 - Interface que agrupa atributos de diferentes classes

Na Figura 4 é apresentado um exemplo de interface do Sistema Atena, associada ao caso de uso funcional “Atribui Freqüência” (Figura 1), que agrupa atributos provenientes da classe “curso”, que é a classe proprietária, da classe associativa “freqüenta” e das classes “aluno” e “aula”. Esse agrupamento de atributos forma uma porção significativa de informação utilizada para realizar o controle de freqüência dos alunos, que é denominada “pauta”. Os nomes dos alunos são utilizados como âncoras para *links* que possibilitam o acesso às informações específicas sobre cada aluno. Todo o conjunto de *links* forma um índice. Essa interface é modelada pelo *m-slice*, ilustrado na Figura 5.

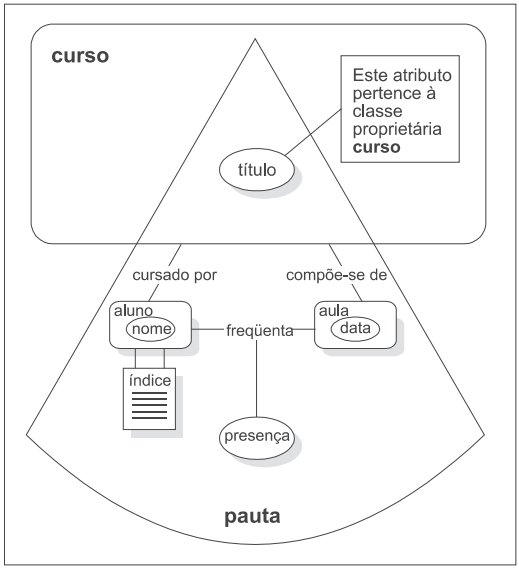


Figura 5 - Exemplo de m-slice que agrupa atributos de diferentes classes

4.4 MODELAGEM DE *WORKFLOWS*

A modelagem de *workflows* concentra-se na especificação dos processos ou fluxos de trabalho automatizados pelo sistema. Um *workflow* é definido como um conjunto coordenado de atividades que devem ser executadas em uma ordem específica e obedecendo a condições e regras predefinidas, com o propósito de atingir um determinado objetivo (AMBERG, 1997).

Os conceitos básicos que compõem um *workflow* e que são necessários para a sua modelagem são: as atividades, as tarefas, os atores ou papéis, os fluxos de trabalho, as pré-condições, as pós-condições e as condições de transição.

Para a modelagem dos elementos citados, são utilizados o diagrama de atividades e o conceito de linhas divisórias (*swimlanes*) da UML. Os diagramas de atividades são úteis para a modelagem de *workflows* por serem capazes de representar processamentos paralelos, por facilitarem a compreensão do comportamento do sistema e por representarem com precisão o progresso dos *workflows* e as interações existentes entre os casos de uso.

Sugere-se também o uso de linhas divisórias, integradas aos diagramas de atividades, para definir explicitamente qual classe de objetos ou qual ator é responsável pela execução de cada atividade (RUMBAUGH; JACOBSON; BOOCH, 1999).

O uso de diagramas de atividade aninhados também é interessante para a modelagem dos *workflows* em diversos níveis. Por essa razão, esse mesmo recurso é utilizado tanto para a definição de atividades, o que corresponde a uma modelagem em um nível mais abstrato, quanto para a definição das tarefas que compõem cada atividade, o que equivale a uma especificação de mais baixo nível.

Desse modo, a modelagem de *workflows* divide-se em duas etapas: a definição de atividades e a definição de tarefas. Na primeira, os casos de uso explicitados na primeira fase do ciclo de desenvolvimento devem ser analisados e compreendidos. Nesse estágio, deve-se identificar as atividades do sistema e suas restrições de execução. De um modo geral, cada atividade do diagrama corresponde a um caso de uso, isto é, a uma macro função do sistema. A próxima etapa baseia-se no detalhamento de cada caso de uso. Nesse momento identificam-se as tarefas que compõem cada atividade (caso de uso). Tais tarefas correspondem a operações ou métodos do sistema e, portanto, devem estar associadas a objetos específicos. As interações ou troca de mensagens entre os objetos e o sistema e alguns detalhes de implementação já passam a ser relevantes.

Durante a segunda etapa, as tarefas identificadas devem ser detalhadas para a sua maior compreensão e para a geração de um documento que servirá de base para a fase de implementação. Dessa forma, as tarefas são classificadas de acordo com: o modo como são iniciadas, o modo como são executadas e o seu conteúdo, como sugerido por Jackson e Twaddle (1997).

Na Figura 6 é ilustrado o uso de um diagrama de atividades para a representação de tarefas. Esse exemplo ilustra simultaneamente o progresso típico e os progressos alternativos do caso de uso “Aprova Candidatos”.

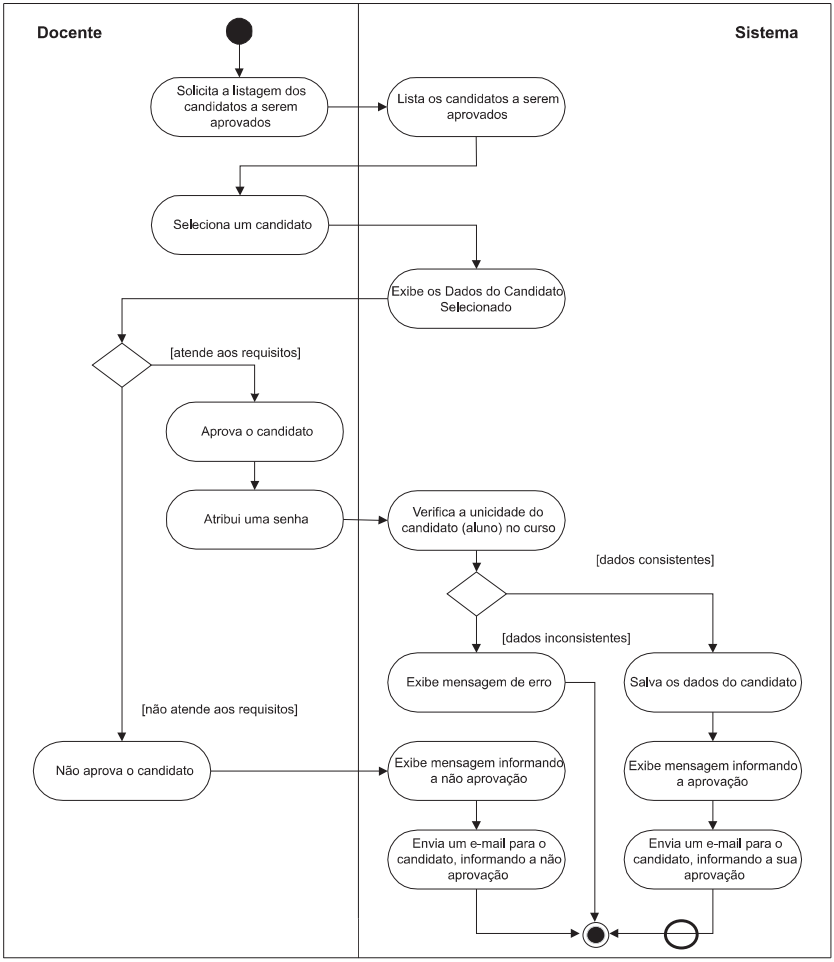


Figura 6 - Diagrama de atividades representando tarefas

4.5 O PROJETO NAVEGACIONAL

O projeto navegacional complementa o projeto de fatias quanto à definição da interface do sistema, acrescentando à especificação os aspectos dinâmicos. A meta desta etapa é definir os caminhos navegacionais que possibilitam o acesso às estruturas de informação definidas anteriormente.

Durante essa etapa do desenvolvimento são definidos os nodos que compõem a hiperbase e as estruturas de navegação ou de acesso a esses nodos. Os nodos formam as unidades de navegação, podem ser compostos por uma ou mais fatias de informação e são os elementos utilizados para a apresentação do conteúdo do sistema e para a interação entre o usuário e a aplicação. As estruturas ou primitivas de acesso incorporadas ao método aqui proposto são as mesmas utilizadas pelo RMM estendido, isto é: as ligações uni e bidirecionais, os agrupamentos, os índices, os roteiros guiados e os roteiros guiados indexados (ISAKOWITZ; KAMIS; KOUFARIS, 1998).

O RMDM e a notação gráfica utilizada no RMM original e estendido, acrescidos de alguns conceitos, tais como as coleções propostas por Garzotto, Paolini e Schwabe (1994) e os centros propostos por Baresi, Garzotto e Paolini (2001), e os estereótipos das extensões à UML propostos por Conallen (1999) constituem a base para a construção do modelo navegacional.

No método proposto neste trabalho, os nodos de informação ou de conteúdo são representados por classes de interface, cada qual associada aos *m-slices* que a compõem e a uma classe do modelo conceitual. Para a melhor especificação dos nodos de informação, um estereótipo é atribuído a cada classe de interface, diferenciando-as entre: páginas do cliente, páginas do servidor e formulários, entre outros, de acordo com as extensões específicas para as aplicações Web feitas à UML, propostas por Conallen (1999). Os demais nodos, embora também contenham informação, são considerados nodos de navegação pois representam sobretudo estruturas de acesso. Esses nodos são modelados por símbolos do RMM que representam os agrupamentos, os índices, os roteiros guiados e os roteiros guiados indexados. Os *links*, ou ligações uni e bidirecionais, formam um caminho que interliga dois nodos.

Aos índices e às demais estruturas de acesso podem estar associadas coleções. Uma coleção é um agrupamento de instâncias de uma mesma classe que satisfazem a um determinado critério de seleção. As coleções também podem especificar critérios de ordenação e/ou agrupamento das informações recuperadas a partir da base de dados. Os diferentes estados assumidos pelos objetos são critérios de seleção comumente utilizados pelas coleções. Também é comum a existência de operações que só podem ser executadas para objetos que satisfazem a determinadas condições. Nesses casos as coleções são os seus

critérios de seleção e os atributos utilizados por esses critérios. Esses atributos, no entanto, normalmente não identificam os objetos selecionados. Desse modo, para a identificação dos objetos que compõem a coleção, são utilizados centros (BARESI; GARZOTTO; PAOLINI, 2001). A Figura 7 ilustra a notação utilizada para representar as coleções e os centros.

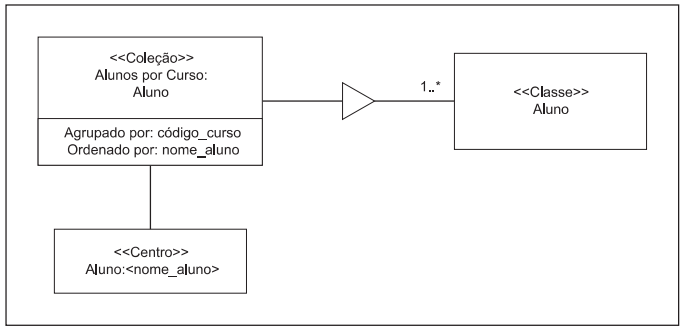


Figura 7 - Exemplo de uma coleção e de um centro

Os centros auxiliam o usuário a compreender o propósito da coleção, quais são os seus elementos e ainda definem pontos de início para a navegação. São estruturas de informação relacionadas a associações semânticas entre duas classes e, por essa razão, são utilizados para complementar a especificação das ligações e das demais estruturas de navegação. Um centro define os atributos que são utilizados pelos usuários para navegar sobre uma associação. Os atributos que compõem um centro identificam quais são os objetos conectados por uma estrutura de navegação, o que permite ao usuário selecionar o objeto de seu interesse antes de realizar a navegação.

Parte da modelagem navegacional do Sistema Atena, a começar pelo nodo raiz ou tela principal da aplicação, é apresentada na Figura 8. Os agrupamentos que são atingidos a partir da tela de entrada do sistema correspondem aos diferentes ambientes que contêm os menus de serviços específicos para cada perfil de usuário.

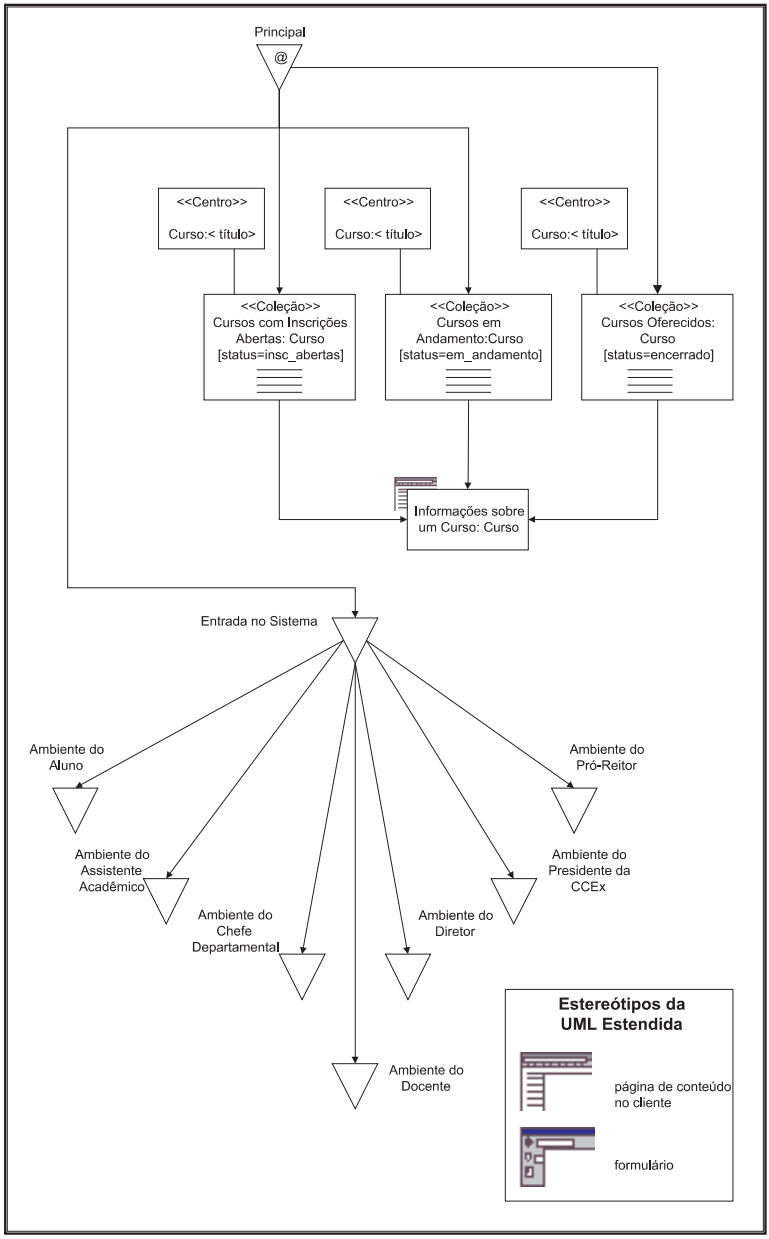


Figura 8 - Modelo navegacional parcial do Sistema Atena

5 IMPLEMENTAÇÃO

Todas as fases do processo de desenvolvimento apresentadas até este ponto são direcionadas a um objetivo final: traduzir as representações do sistema para uma forma que possa ser "entendida" pelo computador. Durante esta última fase do ciclo de desenvolvimento também devem ser realizadas a escolha das mídias apropriadas para a apresentação do conteúdo do sistema e a documentação do código-fonte. A arquitetura mais comumente usada para a execução de aplicações para a Web é a arquitetura de três camadas (3-*tier*), na qual existe um *browser* na máquina cliente, que corresponde à camada de apresentação, um servidor de Web ou servidor HTTP, que corresponde à camada de aplicação, e um servidor de banco de dados, que corresponde à camada persistente.

O Sistema Atena foi desenvolvido na linguagem PHP, sobre a plataforma Linux e, em conformidade com a arquitetura de três camadas (*3-tier*), utiliza o Apache como servidor de Web, o MySQL como servidor de banco de dados e a linguagem JavaScript para a verificação da consistência de alguns dados de entrada, capturados por meio de um *browser* nas máquinas clientes. Como a maior parte do processamento da aplicação concentra-se no servidor e uma parte mínima, direcionada para a consistência local de dados, ocorre no cliente, a arquitetura na qual está baseado o Sistema Atena caracteriza-se como "*thin client*".

Uma das principais vantagens proporcionadas pela implementação de sistemas de *workflow* na Web é a dinamicidade. As interfaces criadas dinamicamente refletem com exatidão o estado corrente do sistema, dos seus componentes e do fluxo de trabalho no momento em que são ativadas. A dinamicidade da interface permite que serviços sejam desativados ou excluídos do menu de tarefas de um determinado ator, caso as condições internas para a sua execução não sejam satisfeitas. Um exemplo de interface gerada dinamicamente é apresentado na Figura 9. Nesse caso, o serviço "Aprovar inscrições de alunos" fica desativado se não existirem inscrições com aprovação pendente.

6 CONCLUSÕES E CONTRIBUIÇÕES

A principal contribuição deste trabalho é a proposta de um método para a análise e o projeto de sistemas de informação baseados na Web que auto-

matizam fluxos de trabalho administrativos ou processos de negócio. O método proposto é uma solução híbrida que enfatiza a orientação a objetos e que agrega técnicas para a especificação de *workflows* a recursos para a modelagem dos aspectos que caracterizam os sistemas para a Web. Devido à atualidade do tema abordado neste projeto, ainda há poucas contribuições direcionadas para o mesmo objetivo. Dentre as soluções pesquisadas, o *framework* W2000 (BARESI; GARZOTTO; PAOLINI, 2001) é a que mais se aproxima da solução aqui apresentada, no entanto, algumas limitações foram identificadas quanto à modelagem de *workflows*.

Os recursos utilizados e integrados para a composição do método foram o diagrama de casos de uso, o modelo de classes, o diagrama de atividades e o diagrama de estados da UML, o padrão para a descrição de casos de uso proposto por Larman (1997), a classificação de tarefas do BPI proposta por Jackson e Twaddle (1997), os *m-slices* e as primitivas de navegação do RMM, as coleções do HDM, os centros e as extensões à UML propostos por Baresi, Garzotto e Paolini (2001) e os novos estereótipos da UML propostos por Conallen (1999).

Uma outra contribuição deste projeto é o próprio Sistema Atena, que tanto pode ser utilizado para a administração de cursos de extensão, que é o seu objetivo original, quanto pode ser integrado, como um módulo administrativo, a um sistema de ensino a distância. O desenvolvimento do sistema possibilitou o teste e a validação do método proposto e serviu como base para uma avaliação preliminar da adequação deste a outros domínios de aplicação.

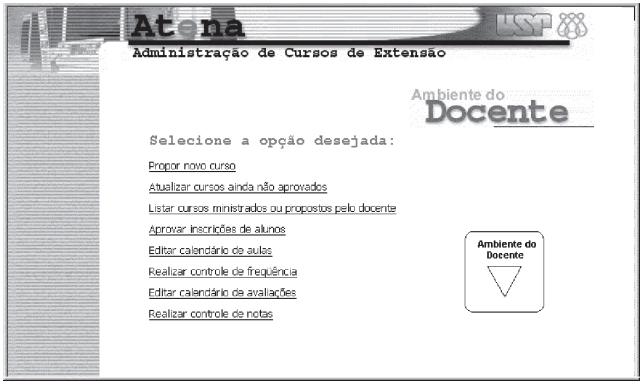


Figura 9 - Exemplo de agrupamento

A METHOD FOR ANALYSIS AND DESIGN OF WEB-BASED BUSINESS WORKFLOW SYSTEMS

ABSTRACT

This article proposes a method for analysis and design of Web-based business workflow systems. The suggested development process is based on the integration of preexisting methods and specific techniques for modeling workflow systems and hypermedia applications in general. The integrated techniques are: UML, the standard notation for modeling object-oriented systems, widely accepted by the Software Engineering community; BPI, a specific process for developing workflow systems; RMM, the first extended methodology for designing Web-based hypermedia applications; and W2000, a framework for designing Web applications based on HDM, the ancestor of a family of several methods that support the design of hypermedia applications. In order to test the proposed method, a system for administration of Web courses was developed. This system, called Atena, has caracteristics of workflow administration, and it was inicially used as a basis for the development of the method and, in a second moment, as a basis to test its scope and adequacy to the proposed domain.

Keywords: Web-based information systems. Workflow systems. UML. W2000.

7 REFERÊNCIAS

AMBERG, M. Understanding workflow**.** In: LAWRENCE, P. **Workflow handbook**. New York: J. Wiley, 1997.

BARESI, L.; GARZOTTO, F.; PAOLINI, P. Extending UML for modeling web applications. HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 34., 2001, Maui. **Proceedings**… Maui: [s.n.], 2001.

CONALLEN, J. **Building web applications with UML**. Boston: Addison-Wesley, 1999.

GARZOTTO, F.; MAINETTI, L.; PAOLINI, P. Adding multimedia collections to the Dexter Model. EUROPEAN CONFERENCE ON

HYPERMEDIA TECHNOLOGIES (ACM ECHT'94), 1994, Edinburgh. **Proceedings**… Edinburgh: [s.n.], 1994. p. 70-80.

GARZOTTO, F.; PAOLINI, P.; SCHWABE, D. HDM: a model-based approach to hypertext application design. **ACM Transactions on Information Systems**, v. 11, n. 1, p. 1-26, 1993.

ISAKOWITZ, T.; KAMIS, A.; KOUFARIS, M. Extending the capabilities of RMM: russian dolls and hypertext. ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 30., 1997, Maui. **Proceedings**… Maui: [s.n.], 1997.

ISAKOWITZ, T.; KAMIS, A.; KOUFARIS, M. The extended RMM methodology for web publishing, **Working Paper IS-98-18**. New York: Center for Research on Information Systems, 1998.

JACKSON, M.; TWADDLE, G. **Business process implementation**: building workflow systems. Boston: Addison-Wesley, 1997.

LARMAN, C. **Applying UML and patterns**: an introduction to object-oriented analysis and design, Upper Saddle River: Prentice Hall, 1997.

RUMBAUGH, J.; JACOBSON, L.; BOOCH, G. **The unified modeling language reference manual**. Massachusetts: Addison -Wesley, 1999.

NORMAS PARA APRESENTAÇÃO DE TRABALHOS

SCIENTIA

NORMAS PARA PUBLICAÇÃO

DAS FINALIDADES

A Revista Scientia é uma publicação da Diretoria de Pós-Graduação do Centro Universitário Vila Velha com vistas à divulgação semestral de produções científicas e acadêmicas nos formatos: editorial, artigo de pesquisa, artigo de revisão, relato de experiência, resenha e resumos de tese, dissertação e monografia de pós-graduação, cujo conteúdo é o que se segue:

- **Editorial:** comentário crítico e aprofundado dos editores.
- **Artigos de pesquisa:** relatos de pesquisas com introdução, metodologia, resultados e discussão, considerações finais e referências.
- **Artigos de revisão:** comentários analíticos e reflexivos sobre temas, a partir do levantamento de bibliografia disponível.
- **Relatos de experiência:** descrições criteriosas de práticas de intervenções e vivências acadêmicas que possam interessar para a atuação de outros profissionais.
- **Resenhas:** revisões críticas de livros, artigos, teses e dissertações.
- **Resumos:** descrições sucintas do conteúdo de tese, dissertação ou monografia de pós-graduação, de caráter informativo, não deve ultrapassar o limite de 500 palavras.

DAS ORIENTAÇÕES GERAIS

Dois terços da revista estão destinados à publicação de **artigos de pesquisa**. No caso do **relato de experiência**, será publicado apenas um por revista.

A critério do Conselho Editorial, poderão ser publicados fascículos ou números especiais, que atendam à demanda das linhas de pesquisa da Pós-graduação.

Cada número da revista buscará enfocar uma área de conhecimento, definida pelo Conselho Editorial, em consonância com as políticas de pesquisa e pós-graduação adotadas pela UVV.

DO CONSELHO EDITORIAL

O Conselho Editorial da revista é constituído pelos seguintes membros:

- Diretor de Pós-graduação, membro nato e seu presidente;
- Coordenador de Pós-graduação Lato Sensu, membro nato;
- Coordenador de Pesquisa, membro nato;
- Coordenador Executivo da Revista, membro nato; e
- Cinco membros da Comunidade Acadêmica, representantes de diferentes áreas do saber com, no mínimo, o título de mestre. Tais representantes, com mandato de dois anos devem ser indicados pelo Diretor de Pós-Graduação.

O Conselho observará:

- A adequação do manuscrito ao escopo da revista;
- A temática proposta para cada volume;
- A qualidade científica, que além de ser atestada por esse mesmo Conselho, deve ser comprovada por um processo anônimo de avaliação realizado por pareceristas *ad hoc,* indicados para esse fim;
- O cumprimento das presentes normas para publicação.

DA ACEITAÇÃO E PUBLICAÇÃO DOS TRABALHOS

A publicação do trabalho estará condicionada ao parecer favorável do Conselho Editorial.

Do resultado da avaliação descrita no item anterior podem derivar três situações, a saber:

- manuscrito aceito, sem restrições;
- manuscrito aceito, com restrições passíveis de revisão;
- manuscrito recusado;

Uma vez aprovado e aceito o manuscrito, cabe à revista a exclusividade de sua publicação.

Uma vez recusado o manuscrito, este poderá ser novamente apresentado à revista. No caso do manuscrito ser recusado duas vezes, a revista não aceitará reapresentação.

O(s) autor(es) de cada manuscrito recebe(m) gratuitamente dois exemplares da revista.

DO ENCAMINHAMENTO

Os manuscritos devem ser encaminhados à Coordenação Executiva da revista, acompanhados de ofício, em que constem:

- Concessão dos direitos autorais para publicação na revista;
- Concordância com as presentes normatizações;
- Procedência do artigo com entidade financiadora;
- Dados sobre o autor: titulação acadêmica, vínculo institucional, endereço para correspondência, telefone e e-mail.

DA APRESENTAÇÃO E ESTRUTURA DOS TRABALHOS

O manuscrito deve ser redigido em português, entregue em duas vias, digitadas em software compatível com o ambiente Windows (Word) e acompanhadas de **um disquete** (3 ½ HD) ou **CD-ROM** contendo o trabalho completo. Na etiqueta do disquete (ou CD-ROM) deverá constar: o título do manuscrito, a autoria e a versão do software.

O manuscrito deve ser organizado da seguinte forma:

- **Página de rosto:** Título (em português e inglês, conciso); Autor(es) (Vínculo Instituicional, titulação, área acadêmica que atua e e-mail), financiamento e endereço para contato.
- **Página de resumo:** título e resumo, em português e inglês, ambos com o máximo de 100 palavras cada, e palavras-chave. Não incluir o nome dos autores.
- **Manuscrito:** Título, Introdução, Desenvolvimento do texto (número de seções, figuras, tabelas e similares, se for o caso); Considerações Finais e Referências. Não incluir nome dos autores.

O texto deve ter no máximo 30 páginas para artigos de pesquisa e artigos de revisão; 10 páginas para relatos de experiência; quatro para resenhas e uma para resumo. Todas obedecendo ao seguinte formato:

- **Fonte:** Arial **12** no corpo do trabalho para o texto; **12** para tabelas, quadros e similares, e **10** para notas de rodapé;
- **Espaçamento:** 1,5 entre linhas de cada parágrafo; e duplo, para figuras e/ou fórmulas;
- **Alinhamento:** justificado, para parágrafos comuns, e adentrado (12 toques a partir da margem esquerda, em bloco e em espaço simples), para citações literais com mais de três linhas;
- **Paginação:** canto superior direito;
- **Configuração:** 3 cm, nas margens superior, inferior e esquerda; e 2 cm na margem direita; 1,25 cm para cabeçalho e o rodapé;
- **Tamanho do papel:** A4;
- **Título:** centralizado, em negrito, e maiúsculas.
- **Nome do(s) autor(es):** dois espaços abaixo do título, em maiúsculas/minúsculas, alinhado à direita, com indicação da titulação e do(s) vínculo(s) institucional(is).
- As citações no interior do texto devem ser digitadas em itálico e separadas por aspas. No final da citação deve aparecer entre parênte-

ses o sobrenome do autor, ano e página da publicação. Exemplo: (Lakatos, 1995, p.18).

- Citações mais longas do que três linhas devem ser destacadas do parágrafo (iniciando a doze toques a partir da margem esquerda) e digitadas em espaço simples, sem aspas. Todas as citações no corpo do texto devem ser listadas na seção de Referências no final do texto. As indicações bibliográficas completas não devem ser citadas no corpo do texto. Entre parênteses devem ser indicados apenas o sobrenome do autor, data e páginas. Ex. (Severino, 2000, p.23).
- As notas de rodapé no final da página deverão se restringir a comentários estritamente necessários ao desenvolvimento da exposição e não para citações bibliográficas.
- As resenhas (de livros, teses, CD sonoro, CD-ROM, produtos de hipermídia etc) devem ter um título próprio que seja diferente do título do trabalho resenhado. O título deve ser seguido das referências completas do trabalho que está sendo resenhado.
- As referências bibliográficas devem ser colocadas em seguida ao artigo, de acordo com o padrão científico da NBR 6023 (ABNT).

a) Livros: sobrenome do autor em maiúsculas, nome em minúsculas, seguido de ponto final, título em negrito seguido por ponto final, cidade seguida por dois pontos, editora e ano de publicação. Exemplo:

b) QUINET, Antonio. **Um olhar a mais, ver e ser visto na psicanálise**. Rio de Janeiro: Jorge Zahar, 2002.

c) Capítulos de livro ou artigos de coletânea:

WEFFORT, Francisco. Nordestinos em São Paulo: notas para um estudo sobre cultura nacional e classes populares. In: VALLE, Edênio; QUEIROZ, José J. (Orgs.). **A cultura do povo**. 3. ed. São Paulo: Cortez, 1984. p. 12-23. (institutos de estudos especiais).

d) Artigos em periódicos:

CHAUÍ, Marilena. Ética e universidade. **Universidade e Sociedade**, São Paulo, ano v. n.8, p.82-87, fev. 1995.

e) Textos da Internet:

CHANDLER, Daniel. An introduction to genre theory. Disponível em: http://www.aber.ac.uk/~dgc/intgenre.html. Acesso em: 23 ago. 2000.

Os trabalhos destinados à apreciação do Conselho Editorial da Revista SCIENTIA devem estar rigorosamente de acordo com as normas da ABNT e ser encaminhados à: Coordenação Executiva da Revista SCIENTIA, Centro Universitário Vila Velha – SEDES/ UVV - ES, Campus Boa Vista, Rua Comissário José Dantas de Melo, 21, Vila Velha, ES, Brasil, Cep 29.102.770. Telefone: (27) 3314 2525. E-mail: scientia@uvv.br