INSTITUTO DE INFORMÁTICA UFG

UFG

Projeto Pedagógico do Curso

Engenharia de Software

(bacharelado)

Instituto de Informática

Universidade Federal de Goiás (UFG)

Preparado por:

Fábio Nogueira de Lucena, Auri Marcelo Rizzo Vincenzi, Juliano Lopes de Oliveira e Plínio de Sá Leitão Júnior

Goiânia — Goiás — Brasil

17 de maio de 2010

Conteúdo

Resumo

Introdução

Este documento apresenta o projeto pedagógico do curso de graduação em Engenharia de Software, modalidade bacharelado, oferecido pelo Instituto de Informática da Universidade Federal de Goiás (UFG). O referido curso foi criado pelo Conselho Universitário da UFG na reunião plenária de 27 de junho de 2008 conforme a resolução CONSUNI 10/2008. O presente projeto pedagógico foi aprovado em reunião extraordinária do Conselho Diretor do Instituto de Informática no dia 12/02/2009.

Versões anteriores e preliminares do presente projeto pedagógico foram publicadas e receberam sugestões de melhorias. Convém destacar duas versões delas, publicadas nos artigos:

- *Bacharelado em Engenharia de Software na Universidade Federal de Goiás,* Fórum de Educação em Engenharia de Software (Simpósio Brasileiro de Engenharia de Software), Monografias em Ciência da Computação, número 43/08, páginas 16-21, Campinas/SP, 17/10/2008, ISSN 0103-9741.
- *Engenharia de Software: Graduação (bacharelado) em Engenharia de Software,* Engenharia de Software Magazine, Ano I, 10a. edição, páginas 56-60, fevereiro de 2009, ISSN 1983127-7.

Contexto

O Instituto de Informática possui extensa experiência no ensino superior e consistente envolvimento com empresas locais de Tecnologia da Informação (TI). O resultado é um rico conjunto de lições aprendidas, que adicionadas à percepção de que a educação em Engenharia de Software é o fator mais relevante para a promoção da indústria de software regional, conduziram ao presente projeto pedagógico de curso (PPC). Conforme [Garg 2008], a educação em Engenharia de Software é o principal fator para a promoção da indústria de software de vários países.

O que é Engenharia de Software?

Engenharia de Software é “a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software, ou seja, a aplicação de engenharia a software, além do estudo de tais abordagens” [IEEE 1990]. Suas principais bases estão na Ciência da Computação e na Matemática [SEEK 2004] e se dedica aos problemas práticos da produção de software [Sommerville 2006]. O conhecimento pertinente encontra-se devidamente documentado [SWEBOK 2004]. A Engenharia de Software usa a matemática, a ciência da computação e a engenharia para resolver problemas em domínios de aplicação.

Organização do Documento

A Seção Projeto Pedagógico do Curso (PPC) fornece detalhes da organização do curso de Engenharia de Software. As disciplinas são detalhadas (ementas, bibliografia e outros) na Seção Disciplinas. A Seção Considerações Finais seguida da Seção Referências fecham o presente documento.

Projeto Pedagógico do Curso

Identificação

Curso	Engenharia de Software
Modalidade	Bacharelado
Título do egresso	Bacharelado em Engenharia de Software
Área do conhecimento	Ciências Exatas e da Terra
Educação	Presencial (ensino a distância restrito ao percentual imposto por legislação em vigor).
Habilitação	Engenharia de Software
Local de oferta	Instituto de Informática Universidade Federal de Goiás Campus II, Goiânia, Goiás, Brasil
Coordenação	Instituto de Informática
Executores	Instituto de Informática
Número de vagas	60 vagas anuais (entrada única)
Carga horária	3000 horas
Horário de funcionamento	Predominantemente noturno.
Forma de acesso	Processo seletivo (vestibular). Em caso de existência de vagas é possível o ingresso através: (a) de transferência de outras Instituições de Ensino Superior; (b) portadores de diploma ou (c) reingresso, de acordo com processo seletivo específico e resoluções da UFG.

Motivação (e Justificativa)

Software é elemento imprescindível no mundo moderno. Construir software exige a disponibilidade de profissionais devidamente aptos a fazer uso da Engenharia de Software, ou engenheiros de software.

Segundo a Brasscom [BRASSCOM] e o governo brasileiro, para exportar US$ 5 bilhões em software até 2010, o Brasil terá que formar “100 mil novos desenvolvedores de software”. A sociedade para a promoção do software brasileiro, Softex [Softex], e órgãos de fomento à pesquisa como o CNPq [CNPq] e a FINEP [FINEP] têm estimulado, por meio de editais específicos, o desenvolvimento de software pelo caráter estratégico para o país.

Apesar da demanda, os cursos de computação na região não contemplam a formação plena do engenheiro de software. De fato, cursos de graduação em Engenharia de Software são desconhecidos no Brasil, embora numerosos pelo mundo. O primeiro curso do qual se tem notícia é oferecido pelo Imperial College (Londres) desde 1985 [SEEK 2004].

Adicionalmente, “a carência de profissionais adequadamente treinados será o maior empecilho para a manutenção e posterior crescimento da indústria de software indiana” [Garg 2008]. Sem generalizações, o cenário nacional não é diferente. Por último, o uso extensivo de software pela sociedade cria uma dependência de profissionais desta área e um lucrativo mercado, que emprega muita mão-de-obra e não agride o meio ambiente.

Objetivos

Objetivo geral

Formar profissionais éticos e capazes de contribuir efetivamente com a produção e manutenção de software competitivo e de alta qualidade.

Objetivos específicos

- Atender demanda por mão-de-obra qualificada em Engenharia de Software, capaz de produzir software com qualidade bem como de efetuar atividades de manutenção com qualidade.
- Atender demanda por profissionais capazes de intervir positivamente em processos de software executados por empresas produtoras de software.
- Atender demanda por profissionais capazes de conduzir empreendimentos de software complexos.
- Formar egresso com sólida percepção e domínio do contexto de uma empresa produtora de software.

Da Denominação do Curso

O termo Engenharia de Software é adotado pelas respeitadas IEEE Computer Society [IEEE CS] e ACM [ACM] (principais associações mundiais de profissionais da área de computação) para denominar um corpo de conhecimento específico, assim como para designar um curso de graduação específico na área de computação. O CNPq adota esta denominação para a especialidade correspondente, assim como a CAPES [CAPES] para designar área. A Sociedade Brasileira de Computação [SBC] apoia o Simpósio Brasileiro de Engenharia de Software que, em 2008, completou a sua edição XXII [SBBDES 2008]. Em 2009 ocorreu a IX Jornada Goiana em Engenharia de Software (um dos mais sólidos eventos de computação do Estado de Goiás). O artigo *Google,* publicado pela revista INFO, Editora Abril, edição de setembro de 2008, número 271, informa na página 42: “40% das vaga do Google Brasil em 2008 ainda não foram preenchidas. Entre elas estão de engenheiro de software, gerente de contas do YouTube e gerente de desenvolvimento de produto.”

Engenharia de Software é um termo conhecido e amplamente empregado além dos domínios comentados no parágrafo anterior. O Gráfico 1 exibe o total de páginas brasileiras para alguns termos segundo o Google (consulta em 07/05/2008).

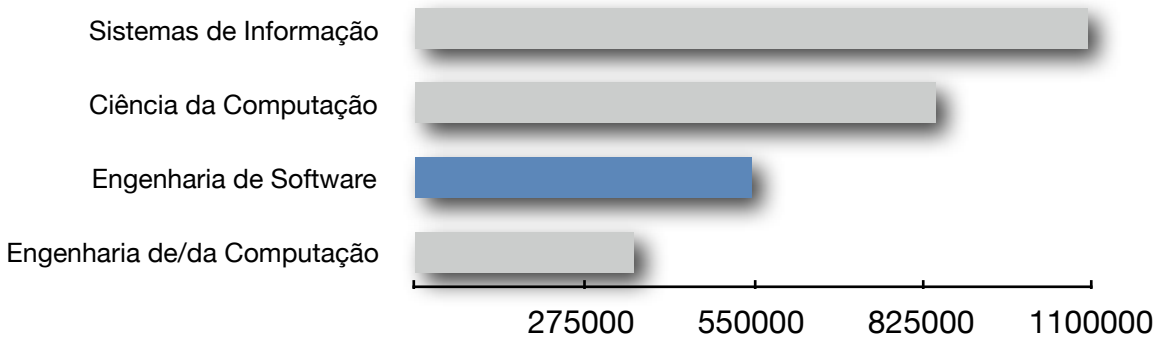

Gráfico 1: Quantidade de páginas retornadas por termos pesquisados no Google em 07/05/2008.

Cursos similares aos termos empregados na Figura 1 existem em todo o mundo. Referências internacionais podem ser consultadas tanto para uma visão geral das diferenças [ACM Careers] quanto para uma abrangente comparação [ACM/IEEE 2005]. Diretrizes para cursos de graduação em Engenharia de Software estão disponíveis [SEEK 2004] e cujo corpo de conhecimento encontra-se devidamente identificado [SWEBOK, 2004]. Diretrizes para os demais cursos da área de computação também estão disponíveis [ACM Cursos].

Cursos de Computação da UFG

A partir de 2009 a UFG passa a oferecer os seguintes cursos pertinentes à computação: (a) Ciências da Computação; (b) Engenharia da Computação; (c) Engenharia de Software (bacharelado) e (d) Sistemas de Informação. Cada um deles possui um conjunto distinto de objetivos, trata a computação de perspectiva peculiar e forma um egresso com perfil específico. O

presente documento esclarece a perspectiva exclusiva do curso de Engenharia de Software. Está além do escopo deste documento elucidar as diferenças entre estes cursos, o que exige consulta aos respectivos projetos pedagógicos.

Fundamentação do Curso

O curso de Engenharia de Software reutiliza informações produzidas por fontes respeitadas nacional e internacionalmente. Por exemplo, o conjunto de disciplinas e as respectivas ementas não introduzem novidades, pois seguem as orientações de tais fontes ao mesmo tempo em que acomoda especificidades do contexto local — Estado de Goiás. Tais fontes e demais componentes de sustentação do curso são identificados e comentados a seguir.

Software Engineering Body of Knowledge (SWEBOK)

Software Engineering Body of Knowledge, ou SWEBOK [SWEBOK 2004], produzido pela IEEE Computer Society, é um guia para o corpo de conhecimento da Engenharia de Software. Trata-se de documento atual, revisado por profissionais de todo o mundo, inclusive o Brasil, que mapeia o conhecimento em Engenharia de Software. Este conhecimento é dividido em áreas. Todas as áreas são cobertas no curso proposto. Naturalmente não é possível contemplar cada um dos itens estabelecidos no SWEBOK para cada uma das áreas. Também não é razoável distribuir de forma homogênea a ênfase entre as áreas. Em consequência, escolhas foram decididas em favor do contexto local.

Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering

Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering [SE 2004] é um documento elaborado pela IEEE Computer Society em cooperação com a ACM. O objetivo é fornecer orientações para instituições de ensino e agências reguladoras acerca do que constitui um curso de graduação em Engenharia de Software. Este documento é um guia para elaboração de propostas de como organizar cursos que cobrem o conhecimento apresentado no SWEBOK. Conforme se lê na primeira página, trata-se do que *todo estudante de graduação em Engenharia de Software deve saber.* Não houve a pretensão de produzir um curso "em conformidade" com este e outros documentos. Tampouco foi feito esforço em sentido contrário. Todas as fontes empregadas foram seriamente avaliadas e as contribuições consideradas oportunas foram empregadas.

Computing Curricula Computer Science

O documento *Computing Curricula Computer Science* [CS 2008] estabelece orientações para cursos de graduação em Ciência da Computação e também motivou conteúdos do presente projeto pedagógico. Convém ressaltar que Ciência da Computação inclui vários conteúdos relevantes e que formam a base da Engenharia de Software.

MPS.BR

Capability Maturity Model® Integration [CMMI] é um modelo de melhoria de processos mundialmente reconhecido e empregado. *Melhoria de Processo do Software Brasileiro* [MPS.BR] é a alternativa nacional de propósito similar e mais apropriada ao contexto brasileiro. Subjacente a tais modelos está a suposição de que a melhoria dos processos de software resulta em softwares melhores. Tais modelos podem ser vistos como referências para se atingir tal melhoria. Estes modelos não são apenas conteúdos abordados no curso. Em particular, a bagagem prática que o egresso do curso deverá obter será exercitada sobre processos de software definidos e gerenciados em conformidade com o MPS.BR.

PMBOK

Project Management Body of Knowledge, ou PMBOK [PMBOK 2008], como é conhecido o guia do corpo de conhecimento em gerência de projetos mantido pelo *Project Management Institute* (http://www.pmi.org), é uma referência amplamente empregada e reconhecida internacionalmente acerca de gerência de projetos. Questões pertinentes à gerência de projetos (assunto relevante em curso de Engenharia de Software) foram observadas da ótica do PMBOK.

Padrões Internacionais

Padrões internacionais fornecem uma identificação concreta de processos e atividades a serem executados, além do formato de documentos, para o desenvolvimento de software. Onde aplicável, padrões deverão ser utilizados por representarem conhecimento amplamente aceito. Por exemplo, os padrões [IEEE 12207-2008] e [IEEE 610.12-1990] estabelecem, respectivamente, uma infraestrutura para o desenvolvimento de software em termos de processos e um glossário de termos empregados na área.

Leis e Regulamentações Pertinentes

O Regulamento Geral de Cursos de Graduação da UFG (RGCG), as diretrizes curriculares dos cursos de graduação na área de computação e informática e normas pertinentes do MEC também forneceram insumos e, em alguns casos, restrições. Em particular, dada a inexistência de diretrizes curriculares específicas para cursos de bacharelado em Engenharia de Software, as seguintes decisões foram tomadas: (a) empregar fontes reconhecidas internacionalmente (conforme esta seção); e (b) fazer uso da Portaria 126 do INEP, 07/08/2008, estabelecida para orientar a avaliação de desempenho dos estudantes em relação aos conteúdos programáticos previstos nas diretrizes curriculares (em particular, foi adotada a recomendação de conteúdos comuns aos perfis de todos os cursos da área, conforme tal portaria).

Código de Ética da Engenharia de Software

O presente curso adota o código de ética da Engenharia de Software [ACM/IEEE-CS] com consequências em termos da formação esperada do egresso. Por exemplo, há ênfase na adoção de padrões que, desta forma, devem ser conhecidos pelos estudantes.

Valores

Um código de ética orienta atitudes. Há um forte interesse em proteger o interesse público, entre outras posturas consideradas apropriadas. Adicionalmente ao código de ética, o curso promove os seguintes valores. Esta lista é inspirada em peculiaridades presentes em superprofissionais [Erdogmus, 2009].

- *Responsabilidade individual*. Claro senso de responsabilidade individual com a entrega de resultados prometidos.
- *Sensibilidade refinada*. Sensibilidade aos interesses e necessidades de clientes.
- *Sinceridade*. Comprometimento com fatos independentemente das repercussões indesejáveis que possam acarretar.
- *Senso de justiça*. Reconhecer o trabalho de outros, compartilhar crédito e resolver interesses conflitantes de forma justa.
- *Tolerância à pressão*. Não permitir que pressões possam comprometer os demais valores.

Fábrica de Software

Instrumento empregado para o desenvolvimento da maturidade do egresso. Possui infraestrutura específica (diferente de laboratórios de ensino empregados no curso) e depende de pessoal técnico, assim como de políticas, processos e regulamentação própria. A fábrica é o contexto onde processos da Engenharia de Software serão exercitados pelos estudantes, assim como métodos e ferramentas. Nesta fábrica serão desenvolvidos produtos de trabalho e executadas atividades de qualidade de software no contexto de projetos.

Contexto Local (Estado de Goiás)

O presente curso não brota no âmbito da UFG mas da percepção da necessidade da sociedade goiana e, em particular, do Arranjo Produtivo Local (APL) de software capitaneado tanto pela Comunidade de Tecnologia de Goiás, COMTEC [COMTEC], quanto pelo SEBRAE-GO [SEBRAE]. Formar profissionais que contribuam com tais empresas e, consequentemente, com o progresso de Goiás, não é um objetivo velado. As centenas de empresas que compõem a COMTEC não é o único indicador da necessidade de tal curso. Recentemente Goiás teve a primeira certificação de uma empresa no nível 2 do CMMI (LG Informática [LG]). Também recentemente tivemos as primeiras empresas certificadas no nível G do MPS.BR. Goiás possui uma das poucas instituições avaliadoras do MPS.BR do Brasil (Estratégia [Estratégia]). Embora isolados, outros fatos se adicionam a estes, inclusive dos governos municipal e estadual. Estes fatos fornecem evidências da necessidade de um novo curso.

Neste sentido, o Instituto de Informática mantém posição privilegiada pois acumula experiência de ensino de graduação ao longo de décadas, dezenas de cursos de especialização e um significativo número de atividades de extensão. A proximidade com o setor produtivo de software ainda é reforçada por acento nos conselhos tanto do SEBRAE-GO, Arranjo Produtivo Local de Software (APL de Software) e COMTEC (Comunidade Tecnológica de Goiás) [COMTEC]. Em conjunto, a posição do Instituto é suficiente para estabelecer uma radiografia das principais dificuldades empregadas pelas empresas locais e a definição de um curso superior capaz de contemplá-las.

Princípios Norteadores para a Formação do Profissional

As diretrizes curriculares para cursos de graduação em computação não contemplam o curso Engenharia de Software, apesar da sólida orientação internacional, que é seguida no presente Projeto Pedagógico de Curso (PPC).

Prática Profissional, Formação Ética e Função Social do Profissional

Tais elementos são imprescindíveis para a formação da postura profissional do egresso e são abordados em disciplina própria, dedicada especificamente ao assunto. Convém ressaltar que o *Código de Ética e Prática Profissional do Engenheiro de Software* é uma das bases de fundamentação do curso, conforme visto adiante e que, a disciplina supracitada não é o único momento de discussão destes assuntos. Adicionalmente, alguns valores são promovidos entre os estudantes (veja *Valores* na seção *Fundamentação do Curso*).

Formação Técnica

Segue recomendação do SWEBOK [SWEBOK 2004] e SEEK [SEEK]. Adicionalmente, quanto à gerência de projetos, são observadas as recomendações contidas no PMBOK [PMBOK]. Todas estas fontes são internacionalmente reconhecidas. Ainda convém ressaltar, conforme destacado na seção anterior, que foram contemplados os conteúdos comuns aos cursos de graduação na área de computação definidos pela Portaria 126 do INEP (07/08/2008).

Articulação Teoria/Prática

O presente projeto pedagógico faz uso da noção de projeto por meio do qual, no contexto de uma Fábrica de Software, a prática da Engenharia de Software será realizada. Esta articulação, portanto, é parte inerente e indissociável do próprio projeto pedagógico. Será tanto mais intensa e efetiva quanto maior for a habilidade da administração do curso em criar tal contexto.

Interdisciplinaridade

Produzir software significa, necessariamente, o emprego de dois domínios: (a) aquele da computação (Engenharia de Software) e (b) aquele no qual está inserido o problema que motiva a existência do software. A interdisciplinaridade é assegurada pela própria concepção do presente projeto pedagógico, onde projetos em execução na Fábrica de Software servem como bancada para as atividades práticas. Mais especificamente, cada projeto faz emprego de um domínio de aplicação, o que exige contato com outra área do conhecimento. Quanto mais rico o conjunto de projetos definidos para execução na Fábrica de Software, mais rica será a interação com outras áreas.

Perfis

Estudante

Deve se interessar pela computação e, em particular, pela produção de software. Deve possuir entusiasmo para conhecer e dominar novos assuntos, além de disposição para construir sua própria reputação por meio dos produtos do seu esforço próprio ou resultantes de trabalho em equipe do qual participa. Deve possuir atitude e a necessidade de realizar, mesmo sem supervisão. Estes atributos são esperados na conduta do estudante ingressante e utilizados ao longo do curso.

Técnico-Administrativo

O curso depende de técnicos-administrativos engajados com a execução da presente proposta de curso. Por exemplo, para apoiar ou executar atividades pertinentes à Fábrica de Software. Isto significa, desejo de aprender e dominar ferramentas tanto de gerência de projetos quanto de outras tecnologias, geralmente exploradas por meio da Internet. Adicionalmente, extensa atividade de interação com o corpo discente, não exclusivamente para questões administrativas, mas de comunicação pertinente ao andamento de projetos, entre outras, será executada. Isto significa habilidade de interação e resolução de conflitos.

Docente

Deve estar familiarizado e se manter atualizado com conteúdos pertinentes à Engenharia de Software. Deve continuamente desenvolver suas habilidades práticas de desenvolvimento de software. Vínculo com o mercado de trabalho é aspecto considerado positivo. Certificações obtidas da indústria e de instituições sólidas também são desejáveis. O docente do curso deve procurar continuamente zelar pela sua proficiência pessoal e especialização em algum aspecto do desenvolvimento de software, domínio de ferramentas, técnicas e métodos correspondentes. A realização de trabalho em grupo é uma exigência não negociável em ambiente composto por outros docentes e estudantes.

Do Curso

O presente curso foi elaborado a partir de demanda e necessidade da região, contemplando dificuldades identificadas pelo Instituto de Informática ao longo de décadas de interação com o mercado. Baseou-se em fontes sólidas internacionalmente, inclusive padrões adotados mundialmente pela indústria. Uma perspectiva abrangente da Engenharia de Software foi adotada, juntamente com uma ampla possibilidade de aplicação do conhecimento desta área. A pretensão é que o egresso não apenas conheça, mas também compreenda e possa aplicar este conhecimento. O exercício deste conhecimento deve ser consciente e pautado por padrões elevados de conduta profissional, tanto da perspectiva ética quanto de postura. Atitude compatível deve ser adotada pelos vários elementos da organização deste curso. Esta organização, documentada em outra parte deste documento, deve contar com a participação de representantes do corpo discente nos processos de tomada de decisão. O objetivo é aproximar a solução de dificuldades encontradas e tornar o conjunto resultado uma equipe integrada e focada na formação do perfil definido para o egresso.

Do Egresso

Profissional preparado para sólida carreira na indústria de desenvolvimento de software. A bacharela ou bacharel em Engenharia de Software deve ser capaz de efetivamente contribuir com equipes na produção de modelos abstratos correspondentes a software e realizá-los por meio de código de qualidade que satisfaz necessidades de clientes.

Da perspectiva pessoal o egresso deve ser capaz de:

- Trabalhar de forma harmoniosa e efetivamente auxiliar na elaboração de produtos atribuídos a equipes.
- Valorizar e iniciar longo processo de formação de sua própria reputação na área.
- Desenvolver postura ativa e ética.

Da perspectiva cognitiva o egresso deve ser capaz de:

- Elicitar, analisar, modelar, especificar (documentar), validar e gerenciar requisitos de software.
- Projetar (*design*) software (arquitetura e projeto detalhado). Inclui modelagem, análise e avaliação da qualidade, princípios, estilos, métodos, modelos arquiteturais e padrões de projeto.
- Construir (programar) software com qualidade e em equipe. Inclui métodos, técnicas, tecnologias e ferramentas.
- Realizar atividades de manutenção de software.
- Planejar e executar atividades pertinentes à qualidade de software. Inclui verificação, validação, revisões, inspeções e testes.
- Gerenciar projetos de software.
- Personalizar processos de software e contribuir com projetos de melhoria de processos de software.
- Transmitir ideias com clareza (seja na forma verbal ou escrita).

Da perspectiva tecnologia e pragmática o egresso deve ser capaz de:

- Exercitar o conhecimento da Engenharia de Software por meio do emprego de tecnologias, ferramentas e métodos.
- Selecionar tecnologias apropriadas para um dado contexto de desenvolvimento de software.

Organização do Ensino

O curso agrupa orientações acerca da condução (metodologia) e conteúdo das disciplinas, além da expectativa acerca das atitudes dos estudantes em *ambientação*, *fundamentação* e *maturidade*.

Ambientação

Apresenta ao estudante, recém-chegado ao curso, um contexto para a Engenharia de Software, uma visão abrangente da

concepção, da composição e de como as partes da área formam um todo coeso. Esta é a fase inicial do curso. O modelo de ensino “tradicional” prevalece, mas não é o único. O foco reside no conhecimento. Inicia-se o processo de resgate e estímulo à postura ativa do estudante.

Fundamentação

Nesta fase a visão horizontal é substituída por perspectiva pontual e vertical. O que deixou de ser novidade na fase anterior passa por um nível adicional de profundidade e se torna objeto de experimentação. A solidez da formação do egresso é estabelecida nesta fase por meio de conhecimento específico onde a compreensão correspondente é fornecida pela prática. O estudante reconhece com facilidade o objeto de estudo. A importância e a familiarização com principais práticas da área tem início. O modelo de ensino “tradicional” perde espaço. O estudante torna-se o centro do processo de aprendizado e atitude compatível é exigida.

Maturidade

O conhecimento apresentado na ambientação e trabalhado na fundamentação passa a ser exercitado em cenários elaborados (projetos reais). Enquanto práticas da fundamentação possuem motivação didática, aquelas da maturidade não apresentam esta restrição (ao contrário, devem ser motivadas por problemas do mundo real). A execução de um projeto típico deve exigir a cooperação de docentes e, possivelmente, profissionais externos, além dos estudantes. É atribuição do curso criar oportunidades para que o estudante empregue as abordagens mais indicadas para solucionar problemas de um dado projeto. Cabe ao estudante gerar resultados esperados e observar o seu rendimento por meio da qualidade de tais resultados. Exercitar o conhecimento para que a formação do egresso se fortaleça é o principal objetivo. O foco reside na aplicação do conhecimento obtido na ambientação e aprofundado na fundamentação. Convém ressaltar que a experimentação em Engenharia de Software é compatível com o perfil do engenheiro de software [SEEK 2004]: “a educação de todo estudante de Engenharia de Software deve incluir experiências do estudante com a prática profissional da Engenharia de Software”.

A ênfase da ambientação, da fundamentação e da maturidade não é executada sequencialmente, uma após a outra, mas com alguma superposição, conforme o Gráfico 2.

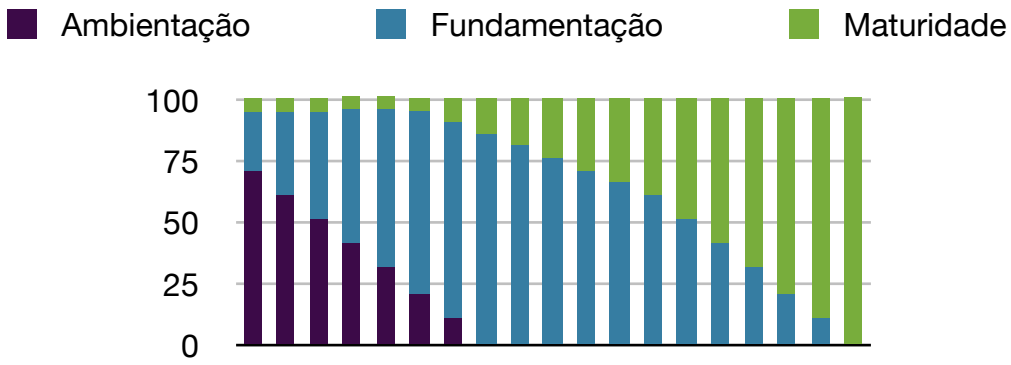

Gráfico 2: distribuição percentual aproximada de ênfases e atitudes das disciplinas ao longo do tempo.

As atenções são inicialmente dedicadas à ambientação e, paulatinamente, dirigidas para a fundamentação e, finalmente, para a maturidade. Isto se faz claro pelo conteúdo, pelo papel desempenhado pelos professores, estudantes e pela complexidade dos problemas abordados. Uma forma menos rigorosa de interpretar tais grupos atribui à fase de ambientação a formação do estudante, à ambientação a um programa de *trainee* em uma Fábrica de Software e, finalmente, à maturidade o verbo aplicar. Estas duas “vertentes”, uma associada à postura do estudante e outra às questões da área de conhecimento do curso podem ser melhor compreendidas por dois dos domínios da Taxonomia de Bloom [Bloom].

A Taxonomia de Bloom [Bloom] é uma classificação dos diferentes objetivos e habilidades estabelecidos para estudantes. Dois dos três domínios estabelecidos pela taxonomia são empregados no curso: o afetivo e o cognitivo. No nível mais inferior do domínio afetivo o estudante apenas recebe, ou simplesmente presta atenção (é passivo). No segundo nível do domínio afetivo o estudante é ativo, ou seja, é capaz de produzir sem necessariamente estar reagindo a eventos externos.

Embora a taxonomia ainda cite três outros níveis para o domínio afetivo, o curso concentra-se no segundo nível. No domínio cognitivo, os níveis considerados no curso incluem: conhecimento, compreensão e aplicação. Existem três outros níveis no domínio cognitivo. Embora a aplicação (da Engenharia de Software) seja o nível cognitivo mais elevado considerado no curso, é desejável e razoável que esta fronteira seja cruzada pelos estudantes em direção a níveis mais elevados.

Instrumentos Pedagógicos

Toda disciplina do curso está associada a um subconjunto do conhecimento da Engenharia de Software, possui carga horária, avaliação e um professor responsável. Ou seja, não se altera o papel administrativo de uma disciplina, em relação ao que é comumente praticado. Contudo, a execução da disciplina, em sua parte prática, ocorre no contexto de um ou mais projetos executados na Fábrica de Software.

A Figura 1 ilustra o modelo tradicional (esquerda) e o modelo proposto (direita). No tradicional o professor "ministra" a disciplina (termo amplamente empregado). A principal acepção deste termo, segundo o dicionário Aurélio da língua portuguesa, é "dar, prestar, fornecer". Também é empregado como "administrar" e "servir". Tais semânticas sugerem uma postura passiva do estudante que, comumente, diz-se que "frequenta" uma disciplina, conforme ilustrado.

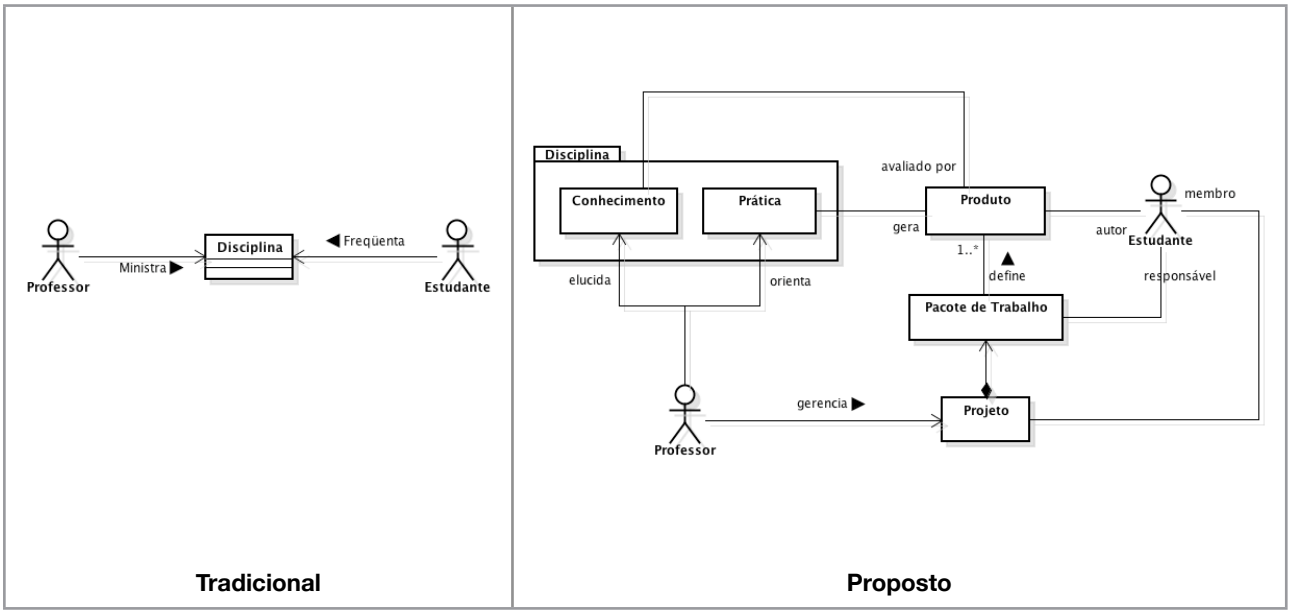

Figura 1: perspectiva do modelo tradicional e proposto.

No modelo proposto persistem as figuras do professor, estudante e disciplina. Contudo, os papéis são alterados da perspectiva de aprendizado. Surgem dois novos elementos: projeto e resultado. O primeiro estabelece o contexto da parte prática das disciplinas, o segundo é que se produz ao executar alguma atividade do primeiro (projeto). O Professor passa a ser gerente ou membro de um projeto e cada Estudante torna-se membro de um projeto e encarregado de atividade que produz resultados que são o motivo da existência do projeto em questão. Resultados são obtidos da aplicação do conhecimento.

Em um projeto não há espaço para elementos passivos. Em vez de "frequentar", estudantes "participam" de um projeto. Todos devem possuir atribuições exequíveis, bem definidas e planejadas. Boa parte destas atribuições serão desempenhadas pelos estudantes.

Princípios Pedagógicos e Suposições

Em vez de estabelecer com precisão a atuação de docentes e discentes optou-se pela apresentação de princípios a serem observados na condução de qualquer atividade do curso:

- *Conhecimento deve ser indissociável da prática e problemas reais correspondentes.* Se não houver como aplicar, como fazer uso pragmático de um assunto, então provavelmente este deve ser evitado. Conhecimento neste curso é instrumento para resolver problemas do cotidiano de engenheiros de software. Problemas reais devem ser empregados para motivar, introduzir e conduzir assuntos. Teoria e prática devem estar integrados. Segundo David Parnas [Parnas 1999], "Em um curso de Engenharia de Software a ênfase está na seleção de algoritmos conhecidos e na aplicação de ferramentas e tecnologias desenvolvidas por outros".
- *Avaliação deve ser obtida, preferencialmente, do desempenho do estudante ao fazer uso de conhecimento.* Ou seja, deve se concentrar no exame do que o estudante é capaz de produzir. O exercício prático do conhecimento é analisado por meio dos artefatos produzidos pelo estudante.
- *Ambiente do curso deve ser receptivo à criatividade e inovações.* Inovar, reformar e vitalizar são ações a serem abraçadas. Isto deverá repercutir na forma de ensino, na forma de interação dos docentes com o corpo discente e nas atividades a serem realizadas. Em vez de castrar, deve-se potencializar mudanças e inovações que tragam melhorias.
- *Trabalho em equipe deve ser estimulado.* O desempenho de atividades em equipe, a cooperação entre os membros da equipe e um ambiente que estimule tal interação devem ser fomentados no âmbito das disciplinas. Deve-se criar oportunidades para tal, trabalhar a auto-estima, o amor-próprio, o civismo, o zelo pela própria reputação, aspectos éticos e de boa convivência.
- *Atividades do curso devem ser executadas, preferencialmente, no contexto da Fábrica de Software*. Os processos da fábrica incluem modelos de artefatos, diretrizes, atividades e outros. Atividades do curso deverão preferencialmente seguir os modelos de artefatos e outros elementos dos processos da Fábrica de Software do curso. Quando a prática não for executada no contexto de um projeto (da Fábrica de Software), então deve-se procurar por problemas interessantes capazes de despertar curiosidade. Por exemplo, o IBGE (http://www.ibge.gov.br) fornece acesso a vasto conjunto de informações; há dicionários na língua portuguesa como o br-ispell e em outras línguas; bases de genomas humano ou de animais (http://www.bovinegenome.org); tabelas de mapeamentos de IPs em países/cidades e outros. Estes são apenas alguns exemplos de bases de dados que podem ser empregadas de forma prática no contexto de várias disciplinas (também servem de inspiração para outras fontes).
- *Atividades devem fomentar as habilidades de comunicação oral e escrita.* A interação do engenheiro de software com o seu meio inclui, necessariamente, a comunicação escrita e oral. Toda disciplina deve trabalhar estes canais de comunicação onde aplicável.
- *Abstração e modelos devem ser extensivamente trabalhados.* Software é entidade cuja complexidade exige do profissional o domínio da expressão de ideias na forma de modelos. A criação de modelos depende da capacidade de abstração. Disciplinas deverão, onde aplicável, exercitar a habilidade de abstração dos estudantes e a confecção de modelos correspondentes.
- *Domínio da língua inglesa.* O domínio do Inglês é reconhecidamente um fator que dificulta a ampliação do mercado de software no Brasil. Adicionalmente, há poucos títulos em português em Engenharia de Software. O curso faz uso extensivo de referências na língua inglesa com o propósito de diretamente expor o egresso do curso à língua que predomina na computação e cujo domínio é valorizado. Disciplinas deverão estar atentas para as dificuldades decorrentes e facilitar a ambientação do estudante.
- *Estudantes estão dispostos a trabalhar sem supervisão e buscam realizações.* Esta atitude é compatível com a Teoria Y de MacGregor [MacGregor]. Adicionalmente, todas as atividades do curso, também deverão adotar a Teoria de Herzberg [Herzberg]. Segundo esta teoria, o ambiente pode destruir a motivação, embora a melhoria do ambiente, em muitas circunstâncias, não é suficiente para melhorá-la. A primeira delas orienta os docentes do curso quanto a atitude esperada e adotada. Decisões dos docentes deverão refletir tais teorias.
- *Estimular o emprego de meios alternativos de interação do corpo discente com o curso.* A interação assíncrona entre os estudantes e professores é estimulada para permitir o esclarecimento de dúvidas, a disponibilidade de conteúdo adicional, orientações da coordenação, comunicados e outros. O objetivo é ampliar as oportunidades de interação além do período do curso.

Estágio Curricular Não-Obrigatório

O estágio é opção do estudante (não obrigatório) e devidamente regulamentado tanto pela UFG quanto pelo Instituto de Informática. Em tempo, a região não possui um número de empresas produtoras de software com reconhecida competência e em número significativo para admitir a exigência de estágio curricular. Convém ressaltar que estágio relevante para egresso deste curso envolve a produção de software com processos bem definidos e institucionalizados. s

Certificado de *Prática Exemplar em Engenharia de Software*

Estudantes que desempenharem papel relevante, qualitativa e quantitativamente, em projetos conduzidos pela Fábrica de Software serão reconhecidos por meio de certificado específico, no qual é atestado o desempenho de destaque do estudante em questão. O objetivo é reconhecer e estimular esforços em que a Engenharia de Software é exercitada. Indiretamente, o estudante premiado o terá como instrumento para demonstrar experiência em Engenharia de Software. A implantação deste certificado será definido em documento próprio onde deve constar, entre outras, informações pertinentes à carga horária mínima, tipo de atividade desenvolvida e como a atividade será avaliada.

Certificado de *Prática em Engenharia de Software*

Estudantes que desempenharem atividades na Fábrica de Software, além daquelas previstas no escopo de disciplinas, e cujos resultados forem considerados satisfatórios, receberão certificado que comprova a prática executada. À semelhança do certificado descrito na seção anterior, documento próprio deverá fornecer detalhes de implantação. Há duas diferenças fundamentais entre estes certificados. O anterior enfatiza a qualidade como elemento de destaque das atividades realizadas, enquanto o presente comprova esforço satisfatório, além da carga horária do curso, em atividade relevante para a formação do egresso.

Trabalho de Conclusão do Curso (TCC)

As horas práticas do curso e a dinâmica de execução das disciplinas, exercidas em ambiente próximo de uma empresa produtora de software (Fábrica de Software) contribuem com os objetivos do Trabalho de Conclusão de Curso (TCC), realizado por meio da disciplina *Prática em Engenharia de Software*. A aprovação de um estudante nesta disciplina significa aprovação deste estudante no TCC.

Deve ser observado:

- O TCC é realizado por grupos, preferencialmente de 5 a 9 estudantes, e orientado por um professor.
- Cada grupo executa um projeto. Preferencialmente o projeto envolve a produção de software.
- Cabe ao professor orientador a avaliação de cada estudante do grupo que ele orienta (não existe banca).
- Cada estudante recebe sua própria nota, independente das notas dos demais estudantes do mesmo grupo.
- Os produtos (entregáveis) de um grupo são definidos em conformidade com o projeto em questão.
- Todos os projetos devem ser referendados pelo Escritório de Projetos da Fábrica de Software.
- Situações diferentes dos itens anteriores serão tratadas pelo Escritório de Projetos da Fábrica de Software.

Organização do Curso (e Recursos Humanos)

O curso compreende uma coordenação, uma secretaria e uma Fábrica de Software. A Fábrica de Software inclui um Escritório de Projetos. Nestas “unidades organizacionais” estarão lotados docentes, estudantes, técnico-administrativos e colaboradores. Um colaborador é uma pessoa ou empresa que tem interesse e capacidade para contribuir com o curso (pode ser elemento externo ao curso e à Universidade).

Cada recurso humano pode desempenhar um ou mais papéis: coordenador, gerente de programa, gerente de projeto ou membro de projeto. Membro de projeto pode ser refinado em programador júnior, engenheiro de requisitos, arquiteto de software ou outro conforme exigência de um dado projeto. A definição precisa de papéis deve ser estabelecida no processo

de software a ser adotado na Fábrica de Software e, naturalmente, adaptada para o projeto em questão, conforme ressaltado. Da perspectiva de pessoas, convém ressaltar, o curso depende fortemente de monitores (nível de graduação ou superior). Esta dependência decorre da significativa quantidade de atividades práticas do curso que exigem “monitoramento” e “acessoramento” dos estudantes.

Da perspectiva hierárquica, a coordenação, auxiliada pela secretaria, é responsável pelo bom andamento de todas as atividades do curso, o que inclui o Escritório de Projetos e a Fábrica de Software. Convém ressaltar que todas as atividades práticas do curso são, preferencialmente, conduzidas no âmbito de projetos.

Ao escritório de projetos cabe a seleção de projetos, o planejamento e a gerência deles. O registro de propostas de projeto, a manutenção destas propostas e a comunicação entre os interessados dos vários projetos em andamento são algumas das atribuições do escritório de projetos. A gerência centralizada dos projetos pelo escritório de projetos deve viabilizar o compartilhamento de recursos e outros benefícios típicos da gerência de programas.

A Fábrica de Software exige uma organização específica definida em documento próprio.

Os alunos de mestrado deverão compor equipes de projetos da Fábrica de Software, particularmente aqueles projetos com exigências específicas em qualificação ou com características inovadoras. Adicionalmente, tais alunos também deverão ser empregados para auxiliar no ensino do curso de Engenharia de Software, tanto para as atividades de ensino quanto para criar a oportunidade de experiência em docência.

Núcleo Docente Estruturante (NDE)

A coordenação do curso possui um coordenador e outros docentes que formam o Núcleo Docente Estruturante, ou NDE. Ao NDE cabe a manutenção do presente Projeto Pedagógico de Curso (PPC) e a correspondente implementação. O NDE é um órgão consultivo, cujas sugestões e decorrentes ações devem ser avaliadas e aprovadas pelo Conselho Diretor (CD) do Instituto de Informática. A definição precisa das atribuições e da constituição do NDE, dentre outras, deverão ser fornecidas em regimento próprio devidamente aprovado pelo CD.

Integração Ensino, Pesquisa e Extensão

A Fábrica de Software deverá hospedar projetos que, além de contribuir com a formação do egresso, despertem o interesse dos alunos e da comunidade. Neste cenário, é razoável que um dado projeto possa ter como colaborador um profissional em atividade ou uma empresa. O que permite uma clara e benéfica interação dos estudantes do curso com a sociedade (extensão). Um colaborador externo pode, inclusive, ministrar treinamento em tópico relevante para o projeto em questão. Em particular, projetos específicos podem demandar qualificação pontual em determinada tecnologia (dominada por uma empresa colaboradora) e/ou exigir a participação de estudantes de mestrado com a incumbência de desenvolver atividade de pesquisa. Um estudante de mestrado também pode, como colaborador, agregar competência exigida por um projeto ou, em “sentido inverso”, se beneficiar da Fábrica de Software e de projeto que experimente resultados parciais produzidos por determinada pesquisa em andamento.

Tecnologias, Paradigmas, Métodos e Processos Adotados pelo curso

Um “eixo” de tecnologias e ferramentas é compartilhado entre as atividades do curso e inclui linguagens de programação, sistemas gerenciadores de bancos de dados, ferramentas de modelagem, sistemas operacionais e outras. Esta estratégia permite que ferramentas deste eixo sejam reutilizadas em várias disciplinas, o que potencializa a familiaridade e o domínio delas por parte do estudante. Este eixo deve evoluir com o propósito de manter a proximidade com “melhores práticas” vigentes.

Não se tem a pretensão de definir ferramentas e tecnologias que serão empregadas pelo mercado, restringir os possíveis projetos a serem desenvolvidos na Fábrica de Software ou as opções de ensino. O presente documento reconhece a importância e estabelece a necessidade, mas não define um eixo. A definição de tecnologias, métodos, processos da Fábrica de Software e outros elementos relevantes para a prática da Engenharia de Software é responsabilidade do Núcleo Docente Estruturante (NDE).

Um eixo não pode ser confundido como “recomendação do curso”. A escolha de tecnologias apropriadas é consequência da investigação de várias variáveis. Conforme o SWEBOK, “um engenheiro deve estar equipado com o conhecimento essencial que oferece suporte à seleção da tecnologia apropriada, no momento e circunstância apropriados”.

Embora o curso não tenha compromisso com tecnologia específica, é feita a opção explícita, conforme ementa das disciplinas, pelo paradigma orientado a objetos. Segundo [Clements, 2009]: “programação orientada a objetos (OO) é o principal paradigma de desenvolvimento de software do nosso tempo”.

Política de Qualificação Docente e Técnico-Administrativo

O Instituto de Informática tem sido consistente, ao longo de sua história, em relação à qualificação dos seus recursos humanos. A orientação é realizar adaptações necessárias para viabilizar a realização de cursos de pós-graduação e cursos técnicos, dentre outros, com o propósito de melhor qualificar seus recursos humanos. Esta política é relevante para o curso de Engenharia de Software, pois demanda constante atualização em termos de tecnologias e ferramentas. Adicionalmente, concursos têm historicamente valorizado candidatos com o título de doutor. De forma mais específica, ações deverão ser executadas, estimuladas ou viabilizadas com o objetivo de assegurar que tanto os técnico-administrativos quanto o corpo docente possam atender os perfis definidos em outras seção deste documento.

Organização das Disciplinas

As disciplinas estão divididas em três grupos: ambientação, fundamentais e maturidade. Seguem uma orientação geral que se inicia com foco no conhecimento, depois na compreensão e solidificação do conteúdo para, finalmente, concentrar-se no uso (aplicação) deste conhecimento. A distribuição da carga horária por grupos é fornecida na Tabela 1.

GRUPO	TEÓRICAS	PRÁTICAS	TOTAL
Ambientação	416	224	640
Fundamentação	672	608	1280
Maturidade	192	512	704
OPTATIVAS	64	0	64
TOTAIS	**1344**	**1344**	**2688**

Tabela 1: Carga horária por grupo.

Conforme Tabelas 2, 3 e 4, para cada disciplina há uma sigla única, nome, núcleo ao qual pertence, a unidade responsável, a carga teórica, a carga horária prática e totais.

SIGLA	DISCIPLINAS DE AMBIENTAÇÃO	NÚCLEO	UNIDADE	TEO	PRA	TOTAL
A1	Introdução à Programação	NC	INF	32	32	64
A2	Introdução à Engenharia de Software	NE	INF	64	0	64
A3	Lógica	NC	INF	32	32	64
A4	Matemática Discreta	NC	INF	64	0	64
A5	Ética, Normas e Postura Profissional	NC	INF	64	0	64
A6	Construção de Software	NC	INF	16	48	64
A7	Algoritmos: Fundamentos e Estruturas de Dados	NC	INF	32	32	64
A8	Método de Desenvolvimento de Software	NC	INF	16	48	64
A9	Arquitetura de Computadores	NC	INF	64	0	64

SIGLA	DISCIPLINAS DE AMBIENTAÇÃO	NÚCLEO	UNIDADE	TEO	PRA	TOTAL
A10	Banco de Dados	NC	INF	32	32	64
Carga horária total da ambientação				416	224	640

Tabela 2: Disciplinas do grupo Ambientação (total de 640 horas)

SIGLA	DISCIPLINAS DE FUNDAMENTOS	NÚCLEO	UNIDADE	TEO	PRA	TOTAL
F1	Engenharia de Software	NE	INF	64	0	64
F2	Algoritmos: Ordenação e Busca	NC	INF	32	32	64
F3	Redes e Sistemas Distribuídos	NE	INF	32	32	64
F4	Sistema Operacional	NC	INF	32	32	64
F5	Linguagens de Programação	NC	INF	32	32	64
F6	Interação Homem-Computador	NE	INF	32	32	64
F7	Projeto Detalhado de Software	NE	INF	32	32	64
F8	Algoritmos em Grafos	NC	INF	32	32	64
F9	Requisitos de Software	NE	INF	32	32	64
F10	Processo de Software	NE	INF	32	32	64
F11	Segurança	NC	INF	32	32	64
F12	Arquitetura de Software	NE	INF	32	32	64
F13	Gerência de Projeto de Software	NE	INF	32	32	64
F14	Gerência de Configuração de Software	NE	INF	32	32	64
F15	Qualidade de Software	NE	INF	32	32	64
F16	Verificação e Validação	NE	INF	32	32	64
F17	Manutenção de Software	NE	INF	32	32	64
F18	Métodos e Ferramentas da Engenharia de Software	NE	INF	32	32	64
F19	Experimentação em Engenharia de Software	NC	INF	32	32	64
F20	Integração 1	NE	INF	32	32	64
Carga horária total da fundamentação				**672**	**608**	**1280**

Tabela 3: Disciplinas do grupo Fundamentos (total de 1280 horas)

SIGLA	DISCIPLINAS DE MATURAÇÃO	NÚCLEO	UNIDADE	TEO	PRA	TOTAL
M1	Leitura de Software	NE	INF	0	64	64
M2	Desenvolvimento de Software Concorrente	NE	INF	16	48	64
M3	Desenvolvimento de Software para Persistência	NE	INF	16	48	64
M4	Desenvolvimento de Software para Dispositivos Móveis	NE	INF	16	48	64

SIGLA	DISCIPLINAS DE MATURAÇÃO	NÚCLEO	UNIDADE	TEO	PRA	TOTAL
M5	Desenvolvimento de Software para a Web	NE	INF	16	48	64
M6	Prática em Engenharia de Software	NE	INF	0	64	64
M7	Integração de Aplicações	NE	INF	16	48	64
M8	Técnicas Avançadas de Construção de Software	NC	INF	16	48	64
M9	Engenharia Econômica para Software	NC	INF	32	32	64
M10	Integração 2	NE	INF	32	32	64
M11	Tópicos em Engenharia de Software	NE	INF	32	32	64
Carga horária total da maturidade				**192**	**512**	**704**

Tabela 4: Disciplinas do grupo Maturação (total de 704 horas)

O curso ainda inclui duas disciplinas optativas, conforme a Tabela 5. Todo estudante deve perfazer pelo menos 64 horas em pelo menos uma das disciplinas desta tabela.

SIGLA	DISCIPLINAS OPTATIVAS	NÚCLEO	UNIDADE	TEO	PRA	TOTAL
OPT1	Mercado de Software	NE	INF	64	0	64
OPT2	Introdução à Língua Brasileira de Sinais (Libras)	NE	LETRAS	64	0	64

Tabela 5: Disciplinas optativas (pelo menos 64 horas)

Duração do Curso (Mínima e Máxima)

O curso tem duração total de 3000 horas distribuídas em 8 (oito) semestres. São necessários pelo menos 8 (oito) semestres para integralização. O período máximo é definido em resolução própria da UFG. A conclusão em 8 semestres se traduz em carga média de 375 horas por semestre (ou 23,43 horas por semana). A sugestão de fluxo é fornecida na seção seguinte. A consolidação da carga horária por núcleos é fornecida na Tabela 6.

CONSOLIDAÇÃO DA CARGA HORÁRIA DO CURSO	% do TOTAL GERAL	TOTAL
Disciplinas do Núcleo Comum (NC)	38.40%	1152
Disciplinas do Núcleo Específico (NE) (obrigatórias)	49.07%	1472
Disciplinas do Núcleo Específico (NE) (optativas)	2.13%	64
Disciplinas do Núcleo livre (NL)	4.27%	128
Atividades Complementares	6.13%	184
TOTAL	100%	**3000**

Tabela 6: Consolidação da carga horária total do curso por natureza.

Sugestão de Fluxo (ou Fluxo de Referência)

Uma possível distribuição das disciplinas por período é apresentada a seguir por período.

1º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Introdução à Programação	64	OBRIGATÓRIA	NC

DISCIPLINA	CH	NATUREZA	NÚCLEO
Introdução à Engenharia de Software	64	OBRIGATÓRIA	NE
Lógica	64	OBRIGATÓRIA	NC
Matemática Discreta	64	OBRIGATÓRIA	NC
Ética, Normas e Postura Profissional	64	OBRIGATÓRIA	NC
Carga horária do período	320		

2º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Construção de Software	64	OBRIGATÓRIA	NC
Algoritmos: Fundamentos e Estruturas de Dados	64	OBRIGATÓRIA	NC
Método de Desenvolvimento de Software	64	OBRIGATÓRIA	NC
Arquitetura de Computadores	64	OBRIGATÓRIA	NC
Banco de Dados	64	OBRIGATÓRIA	NC
Carga horária do período	320		

3º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Engenharia de Software	64	OBRIGATÓRIA	NE
Algoritmos: Ordenação e Busca	64	OBRIGATÓRIA	NC
Redes e Sistemas Distribuídos	64	OBRIGATÓRIA	NE
Sistema Operacional	64	OBRIGATÓRIA	NE
Linguagens de Programação	64	OBRIGATÓRIA	NE
Carga horária do período	320		

4º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Interação Homem-Computador	64	OBRIGATÓRIA	NE
Projeto Detalhado de Software	64	OBRIGATÓRIA	NE
Algoritmos em Grafos	64	OBRIGATÓRIA	NC
Requisitos de Software	64	OBRIGATÓRIA	NE
Processo de Software	64	OBRIGATÓRIA	NE
Segurança	64	OBRIGATÓRIA	NE

DISCIPLINA	CH	NATUREZA	NÚCLEO
Carga horária do período	384		

5º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Arquitetura de Software	64	OBRIGATÓRIA	NE
Gerência de Projeto de Software	64	OBRIGATÓRIA	NE
Gerência de Configuração de Software	64	OBRIGATÓRIA	NE
Qualidade de Software	64	OBRIGATÓRIA	NE
Verificação e Validação	64	OBRIGATÓRIA	NE
Manutenção de Software	64	OBRIGATÓRIA	NE
Carga horária do período	384		

6º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Métodos e Ferramentas da Engenharia de Software	64	OBRIGATÓRIA	NE
Experimentação em Engenharia de Software	64	OBRIGATÓRIA	NE
Integração 1	64	OBRIGATÓRIA	NE
Leitura de Software	64	OBRIGATÓRIA	NE
Desenvolvimento de Software Concorrente	64	OBRIGATÓRIA	NE
Carga horária do período	320		

7º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Desenvolvimento de Software para Persistência	64	OBRIGATÓRIA	NE
Desenvolvimento de Software para Dispositivos Móveis	64	OBRIGATÓRIA	NE
Desenvolvimento de Software para a Web	64	OBRIGATÓRIA	NE
Integração de Aplicações	64	OBRIGATÓRIA	NE
Engenharia Econômica para Software	64	OBRIGATÓRIA	NE
Carga horária do período	320		

8º PERÍODO

DISCIPLINA	CH	NATUREZA	NÚCLEO
Técnicas Avançadas de Construção de Software	64	OBRIGATÓRIA	NE
Prática em Engenharia de Software	64	OBRIGATÓRIA	NE
Integração 2	64	OBRIGATÓRIA	NE
Tópicos em Engenharia de Software	64	OBRIGATÓRIA	NE
Mercado de Software	64	OPTATIVA	NE
LIBRAS	64	OPTATIVA	
Carga horária do período	320		

O Grafico 3 ilustra a distribuição de disciplinas por período e, adicionalmente, as agrupa por temas com a respectiva participação percentual em toda a carga horária do curso.

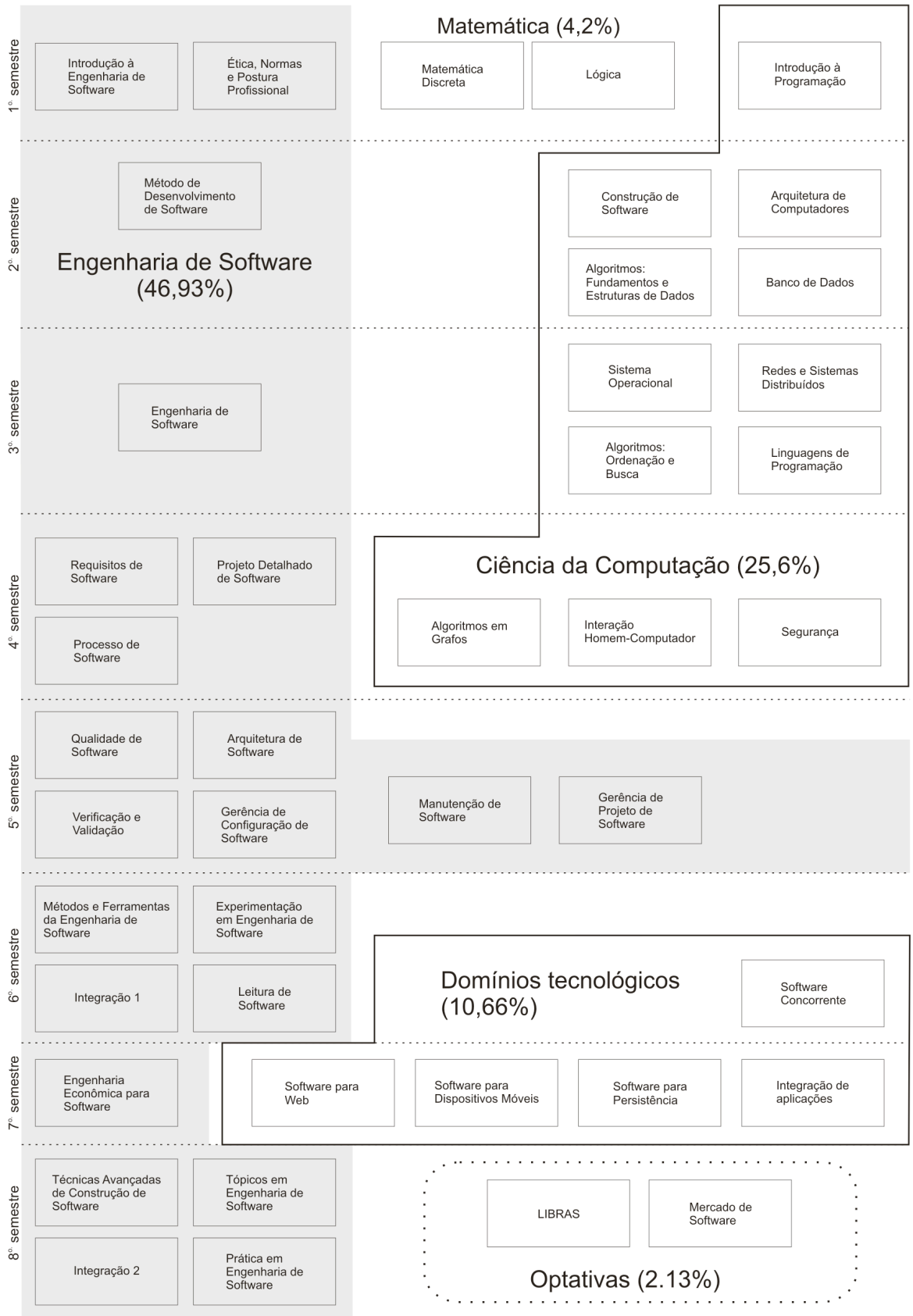

Gráfico 3: Representação gráfica do perfil de formação.

A Tabela 7 compila as disciplinas por semestre e, adicionalmente, acrescenta possíveis horas dedicadas ao Núcleo Livre (NL) e Atividades Complementares (ATC), além da carga horária dedica a disciplinas optativas (OPT).

1		2		3		4		5		6		7		8		
A1	64	**A6**	64	**F1**	64	**F6**	64	**F12**	64	**F18**	64	**M3**	64	**M8**	64	
A2	64	**A7**	64	**F2**	64	**F7**	64	**F13**	64	**F19**	64	**M4**	64	**M9**	64	
A3	64	**A8**	64	**F3**	64	**F8**	64	**F14**	64	**F20**	64	**M5**	64	**M10**	64	
A4	64	**A9**	64	**F4**	64	**F9**	64	**F15**	64	**M1**	64	**M6**	64	**M11**	64	
A5	64	**A10**	64	**F5**	64	**F10**	64	**F16**	64	**M2**	64	**M7**	64	**OPT**	64	
ATC	64	**NL**	64	**NL**	64	**F11**	64	**F17**	64	**ATC**	64	**ATC**	56			
	384		384		384		384		384		384		376		320	**3000**

Tabela 7: Sugestão de disciplinas (inclusive **NL** e **OPT**) e atividades complementares (**ATC**) por período.

Pré-requisitos

Nenhuma disciplina do curso apresenta pré-requisito. Formalmente é possível que o estudante curse qualquer disciplina em qualquer semestre do curso. Naturalmente, isto não significa que todas as ordenações das disciplinas são equivalentes, algumas são mais adequadas do que outras. É da responsabilidade de cada estudante, na existência de opções, identificar as disciplinas mais apropriadas com o apoio do coordenador do curso.

Fábrica de Software (Ambiente de Aprendizado)

O curso exige um escritório de gerência de projetos e a existência de um ambiente de desenvolvimento de software, aqui denominado de Fábrica de Software. Convém ressaltar que tal ambiente não necessariamente segue a noção geralmente associada ao termo conforme aquela em [FS]. Neste documento Fábrica de Software é a designação do principal laboratório do curso. Atividades executadas no âmbito da Fábrica de Software são similares àquelas típicas de um estágio na área, embora realizadas no próprio ambiente do curso.

Nesta fábrica serão executados os projetos nos quais os estudantes (membros de projetos) desempenharão atividades de desenvolvimento de software (parte prática das disciplinas do curso). Projetos poderão ser propostos pelos estudantes e pela sociedade em geral. Independente da origem, cabe ao escritório de projetos gerir a demanda e aqueles em andamento de tal forma a oferecer condições para que os estudantes tenham atividades suficientes, práticas, reais e coerentes com o perfil que se deseja formar.

Além da infraestrutura física, uma Fábrica de Software depende de procedimentos, políticas, processos e possui uma organização própria. Estudantes, professores e colaboradores, ou membros de projetos, deverão seguir as orientações estabelecidas nos procedimentos, políticas e outros. Tais elementos “não físicos” serão motivados pelo nível D do MPS.BR. Adicionalmente, as melhores práticas sugeridas pelo ITIL [ITIL] também deverão ser consideradas ao longo da definição destes procedimentos, processos e outros.

Da perspectiva externa ao curso, a Fábrica de Software pode ser vista como “ponte” entre as empresas do setor e os profissionais em formação apoiados pelo quadro de docentes. Neste sentido, problemas reais vivenciados por estas empresas poderão ser abordados por meio de projetos executados total ou parcialmente na Fábrica de Software. A Fábrica de Software não deve ser interpretada exclusivamente como instrumento didático. A intenção é permitir que o estudante, ainda no curso, possa vivenciar a realidade de um engenheiro de software.

Atividades Complementares

A conclusão satisfatória do curso exige que o estudante perfaça pelo menos 184 horas de atividades complementares. Uma atividade complementar para o curso de Engenharia de Software deve *complementar* ou *suplementar* a formação prevista para o egresso deste curso. Exemplos de atividades complementares incluem, mas não estão restritas àquelas a seguir:

- Participação em congresso e eventos da área;

- Participação em curso pertinente à área;
- Elaboração de artefatos de desenvolvimento de software;
- Elaboração de material pertinente à Engenharia de Software; e
- Outras atividades (conforme julgamento da coordenação do curso).

A definição e julgamento do que é ou não atividade complementar é fornecida em documento próprio do Instituto de Informática. Em geral, ações executadas no âmbito da UFG e julgadas válidas como atividades complementares serão computadas integralmente. Atividades executadas fora do ambiente da UFG poderão ser computadas apenas parcialmente (50%), conforme a relevância para a formação do egresso.

Planejamento Pedagógico

Antes do início de cada período letivo deve ocorrer o planejamento pedagógico das atividades do curso para o semestre em questão. Este planejamento necessariamente inclui a participação dos docentes envolvidos e ocorre com antecedência mínima de uma semana do início das atividades planejadas. A participação de representação do corpo discente é bem-vinda e tem influência nas decisões. O planejamento pedagógico do curso é atribuição do Núcleo Docente Estruturante (NDE).

O planejamento deve considerar as lições aprendidas no passado e como poderão ser empregadas no semestre em pauta. Estratégias didáticas a serem utilizadas são discutidas durante este planejamento. De grande relevância para um curso com significativa carga horária prática está a coordenação destas atividades (escopo, tempo esperado, resultados esperados e outros) a serem executadas no âmbito de cada disciplina. O objetivo é permitir a “interação” entre os conteúdos e compatibilizar recursos (laboratórios e outros).

No planejamento pedagógico deverão ser definidas datas de avaliações (ou trabalhos) ou de outras atividades que não fazem parte das disciplinas mas que interferem na vida acadêmica de estudantes e docentes. A própria identificação de atividades relevantes deve ser abordada, assim como formatos de artefatos e procedimentos a serem empregados nas disciplinas (observar as orientações da Fábrica de Software), dentre outros.

O planejamento pedagógico do curso de Engenharia de Software deve ser interpretado como um momento de “integração” de conteúdos, procedimentos e práticas, além de oportunidade para introduzir melhorias (principalmente após análise dos problemas do passado), discutir estratégias e atitudes que contribuam com a formação do egresso e o bem-estar do corpo docente. Neste sentido, é imprescindível observar a compatibilidade do perfil do docente com as atividades planejadas e a disponibilidade de horas para a devida preparação do docente.

O planejamento pedagógico precede a abertura do semestre, cujo fechamento inclui uma avaliação (seção seguinte).

Sistema de Avaliação do Processo de Ensino e Aprendizagem

O resultado da avaliação do processo de ensino e aprendizagem do curso é um dos insumos para o planejamento pedagógico. Basicamente consiste nas lições aprendidas no semestre anterior, e que serão acrescidas à base de conhecimento do curso. Esta base de conhecimento é um dos principais ativos do curso e será mantida semestralmente ou sempre na ocorrência de evento que justifique alteração. É atribuição do Núcleo Docente Estruturante (NDE) do curso a avaliação do processo de ensino e aprendizagem.

Lições aprendidas serão obtidas da avaliação e análise do semestre anterior (fechamento do semestre) e da avaliação discente das estratégias adotadas pelo curso. A avaliação discente deve ser feita formalmente (inclusive como forma de registro) e por disciplina. Estas avaliações servirão de insumos para a análise do semestre anterior, da qual participam o corpo docente e discente, sob a coordenação do coordenador do curso.

Esta análise é o produto do sistema de avaliação do processo de ensino e aprendizagem, que servirá de insumo para o planejamento pedagógico do semestre posterior. Tal análise deverá ocorrer ao final de cada semestre. Convém ressaltar

que problemas detectados ao longo de um semestre podem ser resolvidos no âmbito da execução de uma disciplina, envolvendo o docente e discentes da disciplina, antes do término da mesma.

Sistema de Avaliação do Projeto de Curso

O presente projeto pedagógico deverá ser avaliado formalmente de quatro em quatro anos ou sempre na ocorrência de evento que justifique tal avaliação. Cabe ao Núcleo Docente Estruturante (NDE) a responsabilidade por tal avaliação, além de registrar informações relevantes para o processo de avaliação do curso, estimular a participação dos docentes, do corpo discente e da sociedade em geral, por meio da indústria de software e de entidades representativas como o SEBRAE, COMTEC e outras.

Disciplinas

Aproveitamento e Equivalência de Disciplinas

Estudantes poderão requisitar aproveitamento em disciplinas conforme normas da UFG. O objetivo do aproveitamento deve evitar que o estudante tenha que cursar disciplina cujo conteúdo já é do seu domínio. Um instrumento empregado para comprovar tal domínio inclui certificações mundialmente conhecidas e respeitadas. Por exemplo, o aluno que é certificado como *Project Management Professional* pelo PMI (http://www.pmi.org) possui domínio suficiente do tema de gerência de projeto para o aproveitamento da disciplina Gerência de Projeto de Software. Esta é apenas uma dentre muitas outras certificações que podem ser empregadas.

As disciplinas do curso são descritas nesta seção. São fornecidas informações para cada um dos seguintes itens:

- Código. Identificador único da disciplina no curso.
- Nome. Nome da disciplina.
- Detalhes. Inclui carga horária total da disciplina, carga horária teórica e prática, além da identificação do núcleo ao qual pertence a disciplina: Núcleo Livre (NL), Núcleo Comum (NC) e Núcleo Específico (NE).
- Pré-requisitos.
- Objetivo. Meta a ser satisfeita após o término da disciplina, inclui resultados esperados.
- Ementa. Detalha o conhecimento a ser abordado na disciplina ou escopo da disciplina.
- Metodologia sugerida. Estratégia sugerida para as atividades acadêmicas pertinentes à disciplina.
- Bibliografia básica. Referências adotadas como livro-texto na disciplina.
- Bibliografia complementar. Referências adicionais que estendem, elucidam e aprofundam questões pertinentes à disciplina.

A1	**Introdução à Programação**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NC
Objetivo	Habilitar o estudante a definir algoritmos para pequenos problemas e implementá-los em uma linguagem de programação.				
Ementa	Lógica de programação. Constantes. Tipos de dados primitivos. Variáveis. Atribuição. Expressões aritméticas e lógicas. Estruturas de decisão. Estruturas de controle. Estruturas de dados homogêneas e heterogêneas. Vetores (*arrays*) e matrizes. Funções. Recursão. Desenvolvimento de algoritmos. Transcrição de algoritmos para uma linguagem de programação. Domínio de uma linguagem de programação: sintaxe e semântica. Estilos de codificação. Ambiente de desenvolvimento. Desenvolvimento de pequenos programas.				
Metodologia sugerida	Cada um dos itens da ementa deverá ser extensivamente praticado. O professor deverá demonstrar, repetidas vezes, um processo elementar onde um problema é analisado, um algoritmo correspondente definido e posteriormente implementado. Um conjunto significativo e coerente de exercícios deverá ser oferecido para prática. O professor deve, preferencialmente, fazer uso de problemas "estimulantes".				
Bibliografia básica	• *Java™ Como Programar*, H. M. Deitel e P. M. Deitel, 6a. edição, Pearson Prentice Hall, 2005.				
Bibliografia complementar	• *The Algorithm Design Manual,* Steven S. Skiena, 2nd edition, Springer, 2008 (aqui). • *Introduction to Programming in Java: An Interdiciplinary Approach,* Robert Sedgewick and Kevin Wayne, Addison-Wesley, 2007 (aqui). • *Head First Java,* Kathy Sierra and Bert Bates, O'Reilly Media, 2005 (aqui). • *Head First Programming,* Vern Ceder, O'Reilly, 2008 (aqui). • *Dictionary of Algorithms and Data Structures.* URL: http://www.nist.gov/dads/.				

Variações são admissíveis e devem ser investigadas durante o Planejamento Pedagógico que antecede cada semestre.

A2	**Introdução à Engenharia de Software**				
Detalhes	64 horas	Teóricas: 64	Práticas: 0	Pré-requisitos: nenhum	NC
Objetivo	Despertar o interesse e adquirir visão abrangente acerca da Engenharia de Software.				
Ementa	Definição de sistema, software e Engenharia de Software. Contexto social e de negócio da Engenharia de Software. Áreas do conhecimento da Engenharia de Software (requisitos, projeto de software e demais). Métodos de desenvolvimento de software. Ferramentas.				
Metodologia	O conteúdo deve ser abordado mais em abrangência do que em profundidade. É imprescindível estabelecer uma clara relação entre as várias partes da ementa e apresentar a área como um todo coeso. Não contempla atividades que exercitam o conhecimento contido na ementa (emprego prático). É desejável que o docente apresente artefatos de projetos reais que ilustrem a teoria. Por artefatos reais entenda planos (por exemplo, plano de gerência de configuração), documentos (por exemplo, definição da arquitetura de software), código e outros oriundos de desenvolvimentos reais (sempre que possível).				
Bibliografia básica	• *Software Engineering*, 8th edition, Ian Sommerville, Pearson Addison-Wesley, 2006 (aqui).				
Bibliografia complementar	• *Software Engineering: A Practitioner's Approach*, Roger S. Pressman, 6th edition, McGraw-Hill, 2004 (aqui).				

A3	**Lógica**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NC
Objetivo	Desenvolver o raciocínio analítico e a habilidade de elaborar sentenças logicamente precisas.				
Ementa	Lógica proposicional. Proposições e conectivos. Operações lógicas sobre proposições. Construção de tabelas-verdade. Tautologias, contradições e contingências. Implicação lógica. Equivalência lógica. Álgebra das proposições. Métodos para determinação da validade de fórmulas da lógica proposicional. Demonstração condicional e demonstração indireta. Lógica de predicados.				
Metodologia sugerida	Todo o conteúdo apresentado deve ser previamente motivado por aplicação em problemas da Engenharia de Software (uma abordagem abstrata do conteúdo deve ser evitada). Listas de sentenças extraídas de documento de especificação de requisitos de software podem ser empregadas para exercitar o conteúdo da ementa. Lógica também pode ser explorada por meio da implementação de tabelas de decisão que expressem regras de negócio ou ações a serem executadas se determinados cenários ocorrerem. Preferencialmente, a parte prática deve ser elaborada e executada sobre informações extraídas de problemas reais.				
Bibliografia básica	• *Iniciação à Lógica Matemática,* Edgard de Alencar Filho, Editora Nobel, 2002.				
Bibliografia complementar	• *Lógica para Ciência da Computação,* João Souza, Editora Campus, 2002. • *Introduction to Mathematical Logic,* E. Mendelson, Academic Press, 2000. • *Lógica e Álgebra de Boole,* Jacob Daghlian, 4a. edição, Atlas, 1995.				

A4	**Matemática Discreta**				
Detalhes	64 horas	Teóricas: 64	Práticas: 0	Pré-requisitos: nenhum	NC
Objetivo	Desenvolvimento do raciocínio abstrato e domínio de técnicas úteis à modelagem e construção de programas.				
Ementa	Técnicas de demonstração. Conjuntos. Combinatória. Relações, relações de equivalência. Ordens parciais e totais. Funções. Indução matemática. Estruturas algébricas (princípios de números naturais, inteiros e racionais).				
Metodologia sugerida	Todo o conteúdo apresentado deve ser motivado, preferencialmente, por aplicação em problemas da Engenharia de Software ou de domínios de aplicação “comuns”. Ou seja, deve-se evitar abordagem abstrata do conteúdo. Exercícios práticos são fortemente recomendados.				
Bibliografia básica	• *Fundamentos Matemáticos para a Ciência da Computação,* Judith L. Gersting, 5a. edição, Editora LTC, 2004 (aqui).				
Bibliografia complementar	• *Matemática Discreta,* E. Scheinerman, Thomson Pioneira, 2003. • *Discrete Mathematics and its Applications,* Kenneth Rose, McGraw-Hill, 6th edition, 2006 (aqui). • *Student’s Solutions Guide to accompany Discrete Mathematics and Its Applications,* Kenneth H. Rosen, McGraw-Hill, 6th edition, 2006 (aqui).				

A5	**Ética, Normas e Postura Profissional**				
Detalhes	64 horas	Teóricas: 64	Práticas: 0	Pré-requisitos: nenhum	NE
Objetivo	Elucidar ética, a importância para a sociedade e auxiliar o estudante a desenvolver um nível de consciência da sua própria postura. Este auto-conhecimento deve ser suficiente para ativar agentes motivadores de mudanças. A disciplina deve estimular a habilidade empreendedora e fornecer noções de desafios de um negócio. Adicionalmente, a disciplina deve apresentar normas relevantes para o engenheiro de software com o propósito de esclarecer a função e o contexto onde podem ser empregadas (a aplicação será assunto de outras disciplinas no curso). Estabelecer a nomenclatura empregada pela área, conforme a norma internacional IEEE Std 12207-2008 é um dos principais objetivos da disciplina. À semelhança das normas, espera-se que o estudante adquira um nível de consciência acerca do que elas tratam.				
Ementa	Noções de ética. Código de ética para engenheiros de software. Visão geral de normas e padrões internacionais, leis e resoluções locais pertinentes à Engenharia de Software. Nomenclatura empregada pela área conforme a norma IEEE Std 12207-2008. Resolução de conflitos. Como se preparar para e se portar em reuniões. Aspectos higiênicos. Aspectos de apresentação pertinentes a trajes. Aspectos de conduta. Atitudes empreendedoras. Instrumentos do empreendedor (plano de negócios e outros). Técnicas de identificação de oportunidades e procedimentos para abertura de um negócio.				
Bibliografia básica	• *Software Engineering Code of Ethics and Professional Practice,* Version 5.2. Disponível em http://www.acm.org/about/se-code.				
Bibliografia complementar	• *O Segredo de Luísa,* Fernando Dolabela, Cultura Editora Associados, 2002. • Padrões internacionais pertinentes à Engenharia de Software (http://standards.ieee.org/software/). • Tecnologia da Informação — Legislação Brasileira, Ministério da Ciência e Tecnologia, Secretaria de Política de Informática, 6a. edição, 2008.				

A6	**Construção de Software**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NC
Objetivo	Dominar “boas” práticas de construção de software.				
Ementa	Fundamentos de construção de software (minimizar complexidade, antecipar mudanças, construção para a verificação e padrões em construção de software). Gerência de construção (modelos de processos de construção, planejamento de construção, medidas de construção). Projeto detalhado e construção de software. Linguagens empregadas na construção de software. Codificação. Testes de unidade e de integração. Reutilização de software. Qualidade de código. Integração.				
Metodologia	A apresentação do conteúdo da ementa deve ser acompanhada de exemplos, preferencialmente reais (software livre pode ser consultado para esta finalidade). O desenvolvimento de um projeto exemplo				
Bibliografia básica	• *Code Complete: Um guia prático para a construção de software*, Steve McConnell, Microsoft Press, 2nd Edition, 2004 (aqui).				
Bibliografia complementar	• *Head First Software Development,* Dan Pilone and Russ Miles, O’Reilly, 2008 (aqui). • *IEEE, Guide to the Software Engineering Body of Knowledge, 2004 Version* (aqui).				

A7	**Algoritmos: Fundamentos e Estruturas de Dados**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NC
Objetivo	O estudante deve ser capaz de selecionar, implementar e fazer uso das estruturas de dados abordadas.				
Ementa	Princípios de análise de algoritmos. Estruturas de dados elementares. Tipos abstratos de dados. Recursão e árvores.				
Metodologia sugerida	O professor deverá identificar cenários de emprego necessário para cada uma das estruturas de dados. Extensa prática de implementação e uso de tais estruturas deverá ser realizada, preferencialmente sobre problemas práticos (ou seja, não abstratos).				
Bibliografia básica	• *Bundle of Algorithms in Java, Parts 1-5: Fundamentals, Data Structures, Sorting, Searching, and Graph Algorithms,* Robert Sedgewick, Addison-Wesley, 3rd edition, 2003 (aqui).				
Bibliografia complementar	• *Dictionary of Algorithms and Data Structures.* URL: http://www.nist.gov/dads/.				

A8	**Método de Desenvolvimento de Software**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NC
Objetivo	Fornecer visão horizontal e ampla de processos técnicos de desenvolvimento de software orientado a objetos de forma prática, com ênfase na construção de software. Esta visão deve ser suficiente para servir de base para disciplinas posteriores acerca de atividades de desenvolvimento de software.				
Ementa	Paradigma orientado a objetos (abstração, encapsulamento, classes, métodos, objetos, herança, polimorfismo, delegação e outros). Modelagem orientada a objetos usando UML. Noções de princípios de projeto orientado a objetos. Implementação de modelos. Método de desenvolvimento de software orientado a objetos. Visão detalhada de método ágil de desenvolvimento de software. Desenvolvimento de pequenas aplicações modeladas e implementadas de forma orientada a objetos seguindo um método ágil e o emprego de orientação a objetos.				
Metodologia	Extenso conjunto de cenários deverão ser exercitados e viabilizar a análise da perspectiva orientada a objetos, tanto conceitos, quanto modelagem e posterior conversão dos modelos em código. Todas as atividades deverão ser executadas no contexto de um método ágil de desenvolvimento de software, que deve ser apresentado, elucidado e seguido. É sugerido o desenvolvimento de um estudo de caso ao longo de toda a disciplina.				
Bibliografia básica	• *Head First Software Development,* Dan Pilone and Russ Miles, O'Reilly, 2008 (aqui).				
Bibliografia complementar	• *Head First Object-Oriented Analysis & Design,* Brett D. McLaughlin et al., O'Reilly, 2006 (aqui). • *Object-Oriented Analysis and Design with Applications,* Grady Booch et al., 3rd edition, Addison-Wesley, 2007 (aqui). • *OOP Demystified,* James Keogh and Mario Giannini, McGraw-Hill, 2004 (aqui). • *Object Thinking,* David West, Microsoft Press, 2004 (aqui). • *Object-Oriented Thought Process,* Matt Weisfeld, Sams, 2nd edition, 2003 (aqui). • *UML Distilled: A Brief Guide to the Standard Object Modeling Language,* Martin Fowler, 3rd edition, Addison-Wesley, 2003 (aqui). • *UML Demystified,* Paul Kimmel, McGraw-Hill, 2005 (aqui).				

A9	**Arquitetura de Computadores**				
Detalhes	64 horas	Teóricas: 64	Práticas: 0	Pré-requisitos: nenhum	NC
Objetivo	Fornecer uma visão geral da organização de computadores.				
Ementa	Visão geral dos computadores modernos. Evolução da arquitetura dos computadores. Sistemas de numeração e aritmética binária. Memória e representação da dados e instruções. Processador, ciclo de instrução, formatos, endereçamento, e programação em linguagem de montagem. Dispositivos de entrada e saída. Sistemas de interconexão (barramentos). Interfaceamento e técnicas de entrada e saída. Hierarquia de memória. Paralelismo ao nível de instrução. Arquiteturas paralelas.				
Bibliografia básica	• *Computer Organization and Design: The Hardware/Software Interface,* David A. Patterson and John L. Hennessy, Morgan Kaufmann, 3rd edition, 2007 (aqui).				
Bibliografia complementar	• *Arquitetura de Computadores Pessoais,* Raul Fernando Weber, Sagra Luzzatto, 2a. edição, 2001.				

A10	**Banco de Dados**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	O estudante deverá ser capaz de modelar e implementar bases de dados a partir de uma especificação de requisitos. A implementação inclui a criação de bases de dados e a execução de consultas SQL em um SGBD real. Adicionalmente, o estudante deverá adquirir uma visão sólida de atomicidade, consistência, isolamento e durabilidade (conhecidas por ACID) para uma transação.				
Ementa	Conceitos básicos. Componentes de sistemas de bancos de dados (*database systems*). Modelagem conceitual (ER e EER). Modelo relacional. Prática de modelagem de dados. Noções de álgebra e cálculo relacional. Mapeamento de esquema conceitual para esquema relacional. Linguagem SQL (extensiva apresentação e prática). Restrições de integridade. Dependências funcionais e formas normais. Transações. Visão geral de mineração de dados e *Data Warehousing*.				
Metodologia	Domínios relativamente restritos poderão ser definidos (requisitos) a partir dos quais a modelagem de dados correspondente deverá ser desenvolvida pelo docente em sala de aula, com interação dos estudantes. O processo deve finalizar com a correspondente implementação (inclusive consultas SQL). Todo este processo idealmente deverá ser exercitado algumas vezes.				
Bibliografia básica	• *Beginning Database Design: From Novice to Professional,* Clare Churcher, Apress, 2007 (aqui).				
Bibliografia complementar	• *Fundamentals of Database Systems,* Ramez Elmasri e Shamant B Navathe, Addison-Wesley, 5th edition, 2007 (aqui) • *Database Systems: An Application-Oriented Approach, Introductory Version,* Michael Kifer et al., Addison-Wesley, 2nd edition, 2004 (aqui). • *Head First SQL: Your Brain on SQL - A Leaner's Guide,* Lynn Beighley, O'Reilly, 2007 (aqui).				

F1	**Engenharia de Software**				
Detalhes	64 horas	Teóricas: 64	Práticas: 0	Pré-requisitos: nenhum	NE
Objetivo	Adquirir uma visão consistente e em profundidade da Engenharia de Software, o que inclui a compreensão das várias áreas do conhecimento pertinentes e as relações entre elas.				
Ementa	Definição de sistema, software e Engenharia de Software. Contexto social e de negócio da Engenharia de Software. Áreas do conhecimento da Engenharia de Software (requisitos, projeto de software e demais). Métodos de desenvolvimento de software. Ferramentas.				
Metodologia sugerida	Esta disciplina aborda o conhecimento da Engenharia de Software com relativa profundidade. É importante que o docente estabeleça associações frequentes entre as áreas. Deve haver constante oferta de "material real" para contato do estudante (por exemplo, um processo de software empregado por determinada empresa ou artefatos produzidos por um projeto real). O docente deve estimular extensa atividade de leitura e o envolvimento com normas e padrões internacionais.				
Bibliografia básica	• *Software Engineering: The Development Process*, Richard H. Thayer and Mark J. Christensen, Wiley-IEEE Computer Society Press, 3rd edition, 2005 (aqui). • *Software Engineering: The Supporting Processes,* Richard H. Thayer and Merlin Dorfman, Wiley-IEEE Computer Society Press, 3rd edition, 2005 (aqui).				
Bibliografia complementar	• *IEEE Computer Society Real-World Software problems: A Self-Study Guide for Today's Software Professional,* Wiley-IEEE Computer Society Press, 2006 (aqui). • *Guide to the Software Engineering Body of Knowledge,* IEEE Computer Society, 2004 (aqui).				

F2	**Algoritmos: Ordenação e Busca**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NC
Objetivo	O estudante deve ser capaz de selecionar, implementar e fazer uso das estruturas de dados abordadas.				
Ementa	Métodos elementares e avançados de ordenação. Métodos elementares de pesquisa. Árvores de pesquisa. *Hashing*. Pesquisa externa.				
Metodologia sugerida	O professor deverá identificar cenários de emprego necessário para cada uma das estruturas de dados. Extensa prática de implementação e uso de tais estruturas deverá ser realizada, preferencialmente sobre problemas práticos (ou seja, não abstratos e reais).				
Bibliografia básica	• *Bundle of Algorithms in Java, Parts 1-5: Fundamentals, Data Structures, Sorting, Searching, and Graph Algorithms,* Robert Sedgewick, Addison-Wesley, 3rd edition, 2003 (aqui).				
Bibliografia complementar	• *Dictionary of Algorithms and Data Structures.* URL: http://www.nist.gov/dads/.				

F3	**Redes e Sistemas Distribuídos**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Ambientar o estudante com questões pertinentes ao de desenvolvimento de sistemas distribuídos e com a prática correspondente.				

F3	**Redes e Sistemas Distribuídos**
Ementa	Modelos de referência em redes: camadas, protocolos e serviços. Camada de rede. Camada de transporte. Camada de aplicação. Programação com sockets. Caracterização de sistemas distribuídos. Arquitetura de Aplicações distribuídas. Sistemas de objetos distribuídos. Serviços de nomes. Comunicação assíncrona. Arquiteturas orientadas a serviços.
Metodologia sugerida	A motivação de cada elemento da ementa deve, preferencialmente, ser realizada por meio de exemplos práticos e reais. Trabalhos práticos devem fazer uso de contextos pragmáticos (evitar exercícios exclusivamente teóricos ou didáticos).
Bibliografia básica	• Kurose, James F. and Ross, Keith W., Computer Networking: A Top-Down Approach, 5th edition, Addison-Wesley, 2009 (aqui). • Coulouris, G. F. et al., *Distributed Systems: Concepts and Design,* 4th edition, Addison-Wesley, 2005 (aqui).
Bibliografia complementar	• Tanenbaum, Andrew S., *Computer Networks,* 4th edition, Prentice-Hall, 2002 (aqui). • Stallings, William, *Data and Computer Communications,* 8th edition, Prentice-Hall, 2006 (aqui). • Tanenbaum, A. S. and M. van Steen, *Distributed Systems: Principles and Paradigms,* 2nd edition, Prentice-Hall, 2006 (aqui). • Dantas, Mário, *Redes de Comunicação e Computadores: Abordagem Quantitativa*, Visual Books, 2009 (aqui).

F4	**Sistema Operacional**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Oferecer sólida noção de funções, serviços e compromissos de um sistema operacional, bem como familiaridade com opções adotadas por sistemas operacionais mais comuns.				
Ementa	Conceitos de Hardware e Software. Tipos de sistemas operacionais. Sistemas multiprogramáveis. Estrutura/organização de sistemas operacionais. Processo. Comunicação entre processos. Gerência de processos. Gerência de memória. Gerência de dispositivos. Sistemas de arquivos. Segurança. Estudos de casos de sistemas operacionais atuais.				
Bibliografia básica	• Tanembaum, Andrew S., *Sistemas Operacionais Modernos,* Prentice-Hall Brasil, 3a. edição, 2010 (aqui).				
Bibliografia complementar	• Silberschatz, A. et al., *Operating Systems Concepts with Java*, 8th edition, Wiley, 2009 (aqui).				

F5	**Linguagens de Programação**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE

F5	**Linguagens de Programação**
Objetivo	A principal interface de um computador empregada por quem desenvolve software é a linguagem de programação. O estudante deve reconhecer a relação entre linguagem e problema (nem toda linguagem se aplica a todo e qualquer problema). O estudante deve compreender as construções de linguagens de programação, os estilos empregados e mecanismos de tradução e execução adotados por linguagens de programação de tal forma que o habilite a "aprender" novas linguagens de programação.
Ementa	Visão geral de linguagens de programação e de paradigmas de programação. Tradução de linguagens. Máquinas virtuais. Tipos. Estruturas de decisão e controle. Funções. Programação orientada a objetos.
Metodologia sugerida	A realização de trabalho prático ao longo do curso onde cada conceito é exercitado pelos estudantes e elucidado pelo docente. O trabalho prático deve ser preferencialmente bem definido e real.
Bibliografia básica	• *Concepts of Programming Languages,* Robert W. Sebesta, Addison-Wesley, 9th edition, 2009 (aqui).
Bibliografia complementar	• *Design Concetps in Programming Languages,* Franklyn A. Turbak and David K. Gifford, MIT Press, 2008. • *Essentials of Programming Languages,* Daniel P. Friedman and Mitchell Wand, The MIT Press, 3rd edition, 2008 (aqui). • *The Definitive ANTLR Reference: Building Domain-Specific Languages,* Terence Parr, Pragmatic Bookshelf, 2007 (aqui).

F6	**Interação Homem-Computador**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver a percepção da importância de um projeto de interação adequado e a compreensão necessária acerca de processo, projeto e avaliação de usabilidade de software.				
Ementa	Princípios de projeto de interfaces homem-computador. Modos de uso e navegação. Projeto visual (cores, ícones, fontes e outros). Tempo de resposta e retro-alimentação. Elementos de interação (menus, formulários, manipulação direta e outros). Localização e internacionalização. Métodos de projeto de interação. Modelos conceituais e metáforas. Voz, linguagem natural, sons, páginas web. Dispositivos de interação. Heurísticas de avaliação de interfaces. Abordagens para testes realizados com apoio de usuários. Técnicas de testes para páginas web. Visão geral de ferramentas de desenvolvimento de interfaces homem-computador.				
Metodologia	É recomendado o emprego de um método de projeto de interação para que o conteúdo teórico possa ser exercitado de forma prática no desenvolvimento do projeto de interação de um projeto relevante. O método deve estar em conformidade com processos da Fábrica de Software.				
Bibliografia básica	• *Designing Interfaces: Patterns for Effective Interaction Design,* Jenifer Tidwell, O'Reilly, 2005 (aqui).				
Bibliografia complementar					

F7	**Projeto Detalhado de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver sólida percepção acerca de implicações de decisões de projeto e de estratégias para obtê-las, bem como habilidade na confecção de modelos de projeto para problemas reais.				
Ementa	Definição de projeto. Questões fundamentais (persistência dos dados, exceções e outras). Contexto de projeto em vários modelos de desenvolvimento de software (ciclos de vida). Princípios de projeto (encapsulamento de informações, coesão e acoplamento). Interação entre projeto e requisitos. Atributos qualitativos em um projeto (confiabilidade, usabilidade, manutenibilidade, testabilidade, desempenho, segurança, tolerância a falhas e outros). Compromissos (custo-benefício). Relação entre arquitetura de software e projeto de software. Projeto orientado a objetos. Projeto funcional. Noção de projeto baseado em estrutura de dados e projeto orientado a aspectos. Projeto orientado por responsabilidade. Projeto por contratos. Métodos de projeto de software. Padrões de projeto. Reutilização. Projeto de componentes. Projeto de interfaces entre componentes e sistemas. Notações de projeto. Ferramentas de suporte a projeto (análise estática, avaliação dinâmica e outras). Medidas de atributos de projeto (acoplamento, coesão e outras). Métricas de projeto (principais métricas, interpretação).				
Metodologia sugerida	Exposição ao conhecimento deve ser seguida de exemplos didáticos onde o docente discute decisões de projeto por meio de modelos previamente elaborados. Em um segundo momento o docente desenvolve pequenos projetos (elaboração dos modelos correspondentes). Em um terceiro momento cenários reais devem ser apresentados, seguidos da elaboração e posterior discussão de propostas de projeto que os soluciona.				
Bibliografia básica	• *Software Design,* David Budgen, 2nd ediciton, Addison-Wesley, 2003 (aqui).				
Bibliografia complementar	• *Head First Design Patterns,* Elisabeth Freeman et al., O'Reilly Media, 2004 (aqui). • *Release It!: Design and Deploy Production-Ready Software,* Michael Nygard, Pragmatic Bookshelf, 2007 (aqui). • *Pattern Oriented Software Architecture Volume 1: A System of Patterns,* Frank Buschmann et al., Wiley, 1996 (aqui). • *Pattern Languages of Program Design 5,* Dragos Manolescu et al., Addison-Wesley, 2006 (aqui). • *Domain-Driven Design: Tackling Complexity in the Heart of Software,* Eric Evans, Addison-Wesley, 2003 (aqui).				

F8	**Algoritmos em Grafos**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NC
Objetivo	O estudante deve ser capaz de selecionar, implementar e fazer uso das estruturas de dados e algoritmos abordados.				
Ementa	Conceitos básicos. Representações de grafos. Grafos dirigidos. Pesquisa em grafos. Árvores geradoras mínimas. Caminhos mínimos.				
Metodologia sugerida	O professor deverá identificar cenários de emprego necessário para cada uma das estruturas de dados. Extensa prática de implementação e uso de tais estruturas deverá ser realizada, preferencialmente sobre problemas práticos (ou seja, não abstratos).				
Bibliografia básica	• *Bundle of Algorithms in Java, Parts 1-5: Fundamentals, Data Structures, Sorting, Searching, and Graph Algorithms,* Robert Sedgewick, Addison-Wesley, 3rd edition, 2003 (aqui).				
Bibliografia complementar	• *Dictionary of Algorithms and Data Structures.* URL: http://www.nist.gov/dads/.				

F9	**Requisitos de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Habilitar o estudante a elaborar e manter especificações de requisitos de software em conformidade com necessidades de diferentes tipos de projetos e restrições.				
Ementa	Definição de requisitos (produto, projeto, processo). Processo de requisitos. Níveis de requisitos (necessidades, objetivos, requisitos dos usuários, requisitos de sistema, requisitos de software. Características de requisitos (testáveis, verificáveis e outras). Princípios de modelagem como decomposição e abstração. Pré e pós condições. Invariantes. Visão geral de modelos matemáticos e linguagens formais de especificação. Interpretação de modelos (sintaxe e semântica). Modelagem de: informações; fluxo de dados; comportamento; estrutura (arquitetura); domínio; processos de negócios e funcional. Padrões de análise. Fundamentos (completitude, consistência, robustez, análise estática, simulação, verificação de modelos, segurança, *safety,* usabilidade, desempenho, análise de causa/efeito, priorização, análise de impacto e rastreabilidade). Gerência de requisitos. Interação entre requisitos e arquitetura. Fontes e técnicas de elicitação. Documentação de requisitos (normas, tipos, audiência, estrutura, qualidade). Especificação de requisitos. Revisões e inspeções.				
Metodologia sugerida	Desenvolver especificações de requisitos, ou parte destas, a partir de termos de abertura de projetos reais, possivelmente envolvendo interessados reais destes projetos e que exijam abordagens distintas.				
Bibliografia básica	• *Software Requirements,* Karl E. Wiegers, Microsoft Press, 2nd edition, 2003 (aqui).				
Bibliografia complementar	• *More About Software Requirements: Thorny Issues and Practical Advice,* Karl E. Wiegers, Microsoft Press, 2006 (aqui). • *Software Requirements Patterns,* Stephen Withall, Microsoft Press, 2007 (aqui). • *Writing Effective Use Cases,* Alistair Cockburn, Addison-Wesley, 2000 (aqui). • *User Stories Applied: For Agile Software Development,* Mike Cohn, Addison-Wesley, 2004 (aqui).				

F9	**Requisitos de Software**
Disciplinas relacionadas:	• Arquitetura de Software • Projeto Detalhado de Software • Verificação e Validação • Manutenção de Software • Gerência de Configuração • Gerência de Projeto de Software • Processo de Software • Qualidade de Software

F10	**Processos de Software**				
Carga horária	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver sólida percepção da importância, impacto, constituição, definição e melhoria de processos.				
Ementa	Conceitos e terminologia. Infraestrutura de processos (pessoas, ferramentas, treinamentos e outros). Modelagem e especificação de processos de software. Medição e análise de processos de software. Melhoria de processos de software (individual e equipe). Análise e controle de qualidade (prevenção de defeitos, revisão de processos, métricas de qualidade, análise de causa e outros). Níveis de definição de processos. Modelos de ciclo de vida (ágil, processos "pesados", cascata, espiral, modelo V e outros). Modelos de processos e padrões (IEEE, ISO e outros). Modelo, definição, medida, análise e melhoria tanto de processo de software individual quanto de equipe. Personalização de processo. Requisitos para processos de software (ISO/IEEE 12207). Detalhada apresentação do MSP.BR (guias). Implementação do MPS.BR. Noções de governância de TI.				
Metodologia sugerida	Estudos de casos reais são indicados para ilustrar com precisão as questões abordadas na ementa.				
Bibliografia básica	• *Software Engineering: The Supporting Processes,* Richard H. Thayer and Merlin Dorfman, Wiley-IEEE Computer Society Press, 3rd edition, 2005 (aqui). • *Software Engineering: The Development process*, Richard H. Thayer and Mark J. Christensen, Wiley-IEEE Computer Society Press, 3rd edition, 2005 (aqui).				
Bibliografia complementar	• *Agile Software Development,* Alistair Cockburn, Addison-Wesley, 2001 (aqui). • *Agile Software Development with SCRUM,* Ken Schwaber and Mike Beedle, Prentice-Hall, 2001 (aqui). • *Guia de Aquisição de Software e Serviços Correlatos,* Softex, 2007 (aqui). • *Guia Geral do MPS.BR,* Softex, 2007 (aqui). • *Guias de Implementação do MPS.BR,* Softex, 2007 (aqui).				

F11	**Segurança**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver percepção clara de política de segurança e de problemas de projeto e implementação pertinentes a software seguro, além de estratégias para evitá-los, suficiente para o desenvolvimento de software no qual se pode confiar.				

F11	**Segurança**
Ementa	Ameaças. Segurança como atributo qualitativo de projeto de software. Autenticação. Autorização. Integridade. Confidencialidade. Criptografia (chaves simétricas e assimétricas). Infraestrutura de chaves públicas brasileiras (ICP-Brasil). Certificados digitais. Assinaturas digitais. Fraquezas decorrentes de problemas na implementação e/ou arquitetura de um software. Desenvolvimento de software seguro. Noções de auditoria de sistemas. Norma NBR 27002.
Metodologia sugeridas	A definição de cenários (problemas) envolvendo questões de segurança que exijam uma solução multidisciplinar (não apenas desenvolvimento de software) deverão ser apresentadas e discutidas.
Bibliografia básica	• *Foundations of Security: What Every Programmer Needs to Know,* Neil Daswani et al., Apress, 2007 (aqui).
Bibliografia complementar	• *Software Security: Building Security In,* Gary McGraw, Addison-Wesley, 2006 (aqui). • *19 Deadly Sins of Software Security,* Michael Howard et al., McGraw-Hill, 2005 (aqui). • *Beginning Cryptography with Java,* David Hook, Wrox, 2005 (aqui).

F12	**Arquitetura de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver clara percepção de arquitetura de software, sua importância e implicações no sucesso ou não de um empreendimento de software. O estudante ainda deve adquirir habilidade para identificar modelos arquiteturais apropriados para problemas e habilidade para definir uma arquitetura de software para um dado cenário.				
Ementa	Definição de arquitetura de software. Importância e impacto em um software. Estilos arquiteturais (*pipe-and-filter*, camadas, transações, *publish-subscribe*, baseado em eventos, cliente-servidor, MVC e outros). Relação custo/benefício entre atributos e opções arquiteturais. Questões de hardware em projeto de software. Rastreabilidade de requisitos e arquitetura de software. Arquiteturas específicas de um domínio e linhas de produto. Notações arquiteturais (visões, representações, diagramas de componentes e outros). Reutilização.				
Metodologia sugerida	Ampla discussão e apresentação do conhecimento disponível seguida da apresentação de casos reais. O professor deverá previamente identificar cenários para estudo e apresentar/discutir arquiteturas possíveis para os mesmos. Trabalhos práticos deverão ser requisitados onde o estudante deve empregar o conhecimento obtido na definição de arquiteturas de software, preferencialmente para problemas reais.				
Bibliografia básica	• *Essential Software Architecture,* Ian Gordon, Springer, 2006 (aqui).				
Bibliografia complementar	• *A Software Architecture Primer,* John Reekie and Rohan McAdam, Angophora Press, 2006 (aqui). • *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives,* Nick Rozanski and Eóin Woods, Addison-Wesley, 2005 (aqui). • *Software Architecture in Practice,* Len Bass et al., Addison-Wesley, 2003 (aqui). • *Patterns of Enterprise Application Architecture,* Martin Fowler, Addison-Wesley, 2002 (aqui). • *Design and Use of Software Architecture: Adopting and Evolving a Product-Line Approach,* Jan Bosch, Addison-Wesley, 2000 (aqui). • *Architecting Enterprise Solutions: Patterns for High-Capability Internet-based Systems,* Paul Dyson and Andrew Longshaw, Wiley, 2004 (aqui). • *Pattern-Oriented Software Architecture Volume 1: A System of Patterns,* Frank Buschmann et al., Wiley, 1996 (aqui). • *Pattern-Oriented Software Architecture Volume 5: On Patterns and Pattern Languages,* Frank Buschmann et al., Wiley, 2007 (aqui). • *Handbook of Software Architecture,* Grady Booch, (disponível na web) (aqui). • *SOA in Practice: The Art of Distributed System Design,* Nicolai M. Josuttis, O'Reilly, 2007 (aqui).				
Disciplinas relacionadas:	• Requisitos de Software • Construção de Software • Projeto Detalhado de Software • Verificação e Validação • Manutenção de Software • Qualidade de Software				

F13	**Gerência de Projeto de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver no estudante uma clara percepção das diferenças entre trabalhar em um projeto e gerenciar um projeto. Particularmente em relação à gerência, o estudante deve ser capaz de planejar, iniciar, planejar, executar, monitorar e concluir a gerência de pequenos projetos de software.				
Ementa	Conceitos, terminologia e contexto de gerência de projetos. Ciclo de vida de produto e projeto. Interessados (*stakeholders*). Organização de empresas (funcionais, matriciais e baseadas em projetos). Estratégias para seleção de projetos. Processos de gerência de projetos. Gerência de escopo. Gerência de tempo (definição de atividades, sequenciamento de atividades, estimativa de recursos, estimativa de duração, desenvolvimento de cronograma e controle de cronograma). Gerência de custos (estimativas, orçamento e controle). Gerência de qualidade. Gerência de recursos humanos. Gerência de comunicação. Gerência de riscos. Gerência de aquisições. Gerência de integração (desenvolver carta de projeto, desenvolver escopo preliminar, desenvolver plano de gerência de projeto, dirigir e gerenciar a execução de projetos, monitorar e controlar atividades de projeto, controle de mudanças e fechamento do projeto). Estabelecer relações com o MPS.BR. Gerência de aquisições deve ser observada da perspectiva do Guia de Aquisições de Software e Serviços Correlatos (MPS.BR).				
Metodologia sugerida	Tópicos deverão ser esclarecidos e abordados da perspectiva de projetos de software. Por exemplo, empregar ponto de função para estimativa de esforço. O emprego de estudos de casos reais é recomendado.				
Bibliografia básica	• *Project Management: A Systems Approach to Planning, Scheduling, and Controlling,* Harold Kerzner, 10th edition, Wiley, 2009 (aqui).				
Bibliografia complementar	• *A Guide to the Project Management Body of Knowledge*, 4th edition, 2008, ISBN 978-1933890517 (aqui). • *Manage It!: Your Guide to Modern, Pragmatic Project Management,* Johanna Rothman, Pragmatic Bookshelf, 2007 (aqui). • *PM Crash Course: A Crash Course in Real-World Project Management,* Rita Mulcahy, RMC Publications, 2006 (aqui). • *Agile Project Management with Scrum,* Ken Schwaber, Microsoft Press, 2004 (aqui). • Agile and Iterative Development: A Manager's Guide, Craig Larman, Addison-Wesley, 2003 (aqui). • *Agile Estimating and Planning,* Mike Cohn, Prentice-Hall, 2005 (aqui). • *Agile Project Management: Creating Innovative Products,* Jim Highsmith, Addison-Wesley, 2009 (aqui). • *Software Estimation: Demystifying the Black Art,* Steve McConnell, Microsoft Press, 2006 (aqui). • *The Mythical Man-Month: Essays on Software Engineering,* Frederick P. Brooks, 2nd edition, Addison-Wesley, 1995 (aqui).				

F14	**Gerência de Configuração de Software**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver habilidade na elaboração, implementação e prática de planos de gerência de configuração de software.				
Ementa	Conceitos e terminologia. Processos de gerência de configuração. Identificação de itens de configuração. Atributos a serem registrados para cada item de configuração. Armazenamento. Controle de mudanças. Relatórios de status. Controle de versões e linhas base ou de referência (*baselines*). Gerência de configuração segundo o MPS.BR. Papéis em gerência de configuração. Normas (IEEE 828). Princípios de gerência de configuração e relação com atividades de desenvolvimento de software. Gerência de configuração segundo desenvolvimento ágil, técnica de *builds* frequentes e desenvolvimento iterativo. Gerência de configuração para diferentes tipos de produtos (compostos, multiplataforma, múltiplas variantes, críticos, pequenos, médios e grandes). Gerência de configuração para desenvolvimento de software distribuído geograficamente, múltiplos interessados e desenvolvimento paralelo. Melhoria de gerência de configuração. Considerações práticas acerca de gerência de configuração de software. Ferramentas.				
Metodologia sugerida	A partir da definição de cenários previamente elaborados pelo docente, os estudantes deverão elaborar planos de gerência de configuração adequados para os cenários apresentados. A discussão dos planos deve incluir clara percepção de estratégias para lidar com os problemas dos cenários e prática correspondente.				
Bibliografia básica	• *Configuration Management Principles and Practice*, The Agile Software Development Series, Anne Mette Jonassen Hass, Pearson Education, 2003 (aqui).				
Bibliografia complementar	• *Software Configuration Management Patterns: Effective Teamwork, Practical Integration,* Stephen P. Berczuk et al., Addison-Wesley, 2003 (aqui). • *The Build Master: Microsoft's Software Configuration Management Best Practices,* Vincent Maraya, Addison-Wesley, 2005 (aqui). • *Software Configuration Management FAQ* (aqui).				

F15	**Qualidade de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver percepção clara de qualidade aplicada a produto, projeto ou processo de software. Quanto a produto, não apenas o produto final, mas também artefatos intermediários (entregáveis ou não). Cabe a esta disciplina apresentar uma visão integral de qualidade, visto que disciplinas abordam o tema "isoladamente". O estudante deve compreender conceitos de qualidade e reconhecer que requisitos definem características de qualidade de um software e influenciam critérios para a validação destas características.				
Ementa	Definições e terminologia de qualidade de software. Custos e impactos de baixa qualidade. Custo de um modelo de qualidade. Terminologia para características de qualidade de software (ISO 9126-1). Papel de pessoas, processos, métodos, ferramentas e tecnologias em qualidade. Padrões de qualidade (ISO 9001, ISO 9003-04, IEEE Std 1028-2008, IEEE Std 1465-2004, IEEE Std 12207-2008, ITIL). Revisões, auditoria e inspeções. Modelos e métricas de qualidade de software. Aspectos relacionados a qualidade de modelos de processos de software. Visão geral do CMMI. MPS.BR. Planejamento de qualidade. Garantia da qualidade. Análise de causa e prevenção de defeitos. Avaliação de atributos de qualidade. Métricas e medidas de qualidade de software. Desenvolver planos de qualidade de software em conformidade com o padrão IEEE Std 730-2002.				
Bibliografia básica	• Software Quality Assurance: From Theory to Implementation, Daniel Galin, Addison-Wesley, 2004 (aqui).				
Bibliografia complementar	• *Metrics and Models in Software Quality Engineering,* Stephen H. Kan, 2nd Edition, Addison-Wesley, 2002 (aqui).				

F16	**Verificação e Validação**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Ementa	Objetivos e restrições de V&V (Verificação e Validação). Planejamento de V&V. Documentação de estratégias de V&V, testes e outros artefatos. Medidas e Métricas. Análise estática de código. Atividades de V&V ao longo do ciclo de vida de um produto. Revisão de software. Testes de unidade. Análise de cobertura. Técnicas de teste funcional (caixa preta). Testes de integração. Desenvolvimento de casos de teste baseados em casos de uso e estórias de usuários. Testes de sistema. Testes de aceitação. Testes de atributos de qualidade. Testes de regressão. Ferramentas de teste (combinação com ferramentas de integração contínua). Análise de relatórios de falha. Técnicas para isolamento e falhas (depuração). Análise de defeitos. Acompanhamento de problemas (*tracking*). IEEE Std 1012-2004.				
Bibliografia básica	• *Software Verification and Validation for Practitioners and Managers,* Steven R. Rakitin, Artech House, 2nd edition, 2001 (aqui).				
Bibliografia complementar	• *Teste e Análise de Software: Processos, princípios e técnicas,* Mauro Pezzè e Michal Young, Bookman, 2008 (aqui). • *JUnit Recipes: Practical Methods for Programmer Testing,* J. B. Rainsberger, Manning, 2004 (aqui). • *xUnit Test Patterns: Refactoring Test Code,* Gerard Meszaros, Addison-Wesley, 2007 (aqui).				

F17	**Manutenção de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Oferecer uma abrangente visão de manutenção (evolução) de software e de questões correlatas, bem como estas questões são relacionadas a outras do ciclo de vida de um software.				
Ementa	Conceitos e terminologia. Categorias (tipos) de manutenção. Questões técnicas e gerenciais de manutenção. Estimativa de custo de manutenção. Métricas/medidas para manutenção. Processos e atividades de manutenção. Compreensão de programas. Reengenharia. Engenharia reversa. Norma IEEE Std 14764-2006. Refatoração. Transformação de programas.				
Metodologia sugerida	Da perspectiva prática, a disciplina deve apresentar cenário de manutenção a ser executado pelos estudantes onde são privilegiadas as questões pertinentes a atividades, processos e artefatos intermediários, em detrimento de mudanças técnicas elaboradas. A manutenção deverá ser executada, acompanhada e comentada pelo docente.				
Bibliografia básica	• *Software Maintenance Management: Evaluation and Continuous Improvement,* Alain April e Alain Abran, Wiley, 2008 (aqui).				
Bibliografia complementar	• *Refactoring: Improving the Design of Existing Code,* Martin Fowler et al, Addison-Wesley, 1999 (aqui). • *Refactoring to Patterns,* Joshua Kerievsky, Addison-Wesley, 2004 (aqui).				

F18	**Métodos e Ferramentas da Engenharia de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Habilitar o estudante a reconhecer métodos e ferramentas relevantes para a produção de software.				
Ementa	Métodos heurísticos, formais e de construção de protótipos. Ferramentas para auxiliar na produção de requisitos, projeto, construção, testes e manutenção. Ferramentas de gerência de configuração, gerência de projeto, processo de software, qualidade e outras.				
Metodologia sugerida	Estudos de caso de projetos relevantes são aconselhados para permitir a identificação de emprego real de ferramentas e métodos, bem como a combinação empregada por estes projetos de ferramentas e métodos.				
Bibliografia básica	• *Software Engineering Methods and Technologies,* Alfonso Fuggetta e Laura Stardini, publicado em *Software Engineering Volume 2: The Supporting Process,* R. H. Thayer e M. Dorfman editors, 3rd edition, 2005. Disponível em http://alfonsofuggetta.org.				
Bibliografia complementar	• *Guide to the Software Engineering Body of Knowledge,* IEEE Computer Society, 2004. Disponível em http://swebok.org.				

F19	**Experimentação em Engenharia de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver a noção de experimento em Engenharia de Software e adquirir percepção de como pesquisas são conduzidas na área. Ambientar o estudante com as ferramentas necessárias (inclui processos e estatística) para o planejamento, condução e apresentação de relatórios de investigações empíricas em Engenharia de Software.				
Ementa	Conceituação e esclarecimento acerca de experimento controlado, estudos de caso e *surveys*. Processo de desenvolvimento de um projeto de pesquisa (inclui atividades, formulação de questões, construção de teoria e análise qualitativa/quantitativa de dados). Investigação de experimentos científicos em Engenharia de Software. Prática acompanhada de pequeno experimento em Engenharia de Software.				
Metodologia	Assuntos devem ser abordados com o complemento de leituras de artigos científicos na área. Toda a parte estatística necessária deve ser apresentada como parte da disciplina e conforme a demanda da própria disciplina. Um pequeno experimento deve ser realizado pelos estudantes, possivelmente com sugestões de tema dos próprios alunos. Neste experimento o professor deve continuamente acompanhar as atividades realizadas (parte prática). As horas teóricas serão dedicadas ao desenvolvimento do conteúdo da ementa.				
Bibliografia básica	• *Basics of Software Engineering Experimentation,* Natalia Juristo e Ana M. Moreno, Springer, 2001 (aqui).				
Bibliografia complementar	• *Experimentation in Software Engineering: An Introduction,* Claes Wohlin et al., Kluwer Academic Publishers, 2000 (aqui).				

F20	**Integração 1**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Fornecer e elucidar uma visão coesa e integrada da Engenharia de Software de tal forma que o estudante desenvolva uma sólida percepção de como fazer uso do conhecimento desta área em toda a sua extensão.				
Ementa	Rever, exemplificar o emprego da Engenharia de Software em abrangência e profundidade. Integrar todo o conhecimento das disciplinas do curso de tal forma a permitir a compreensão, a relação entre elas, a importância, os produtos e atividades pertinentes a cada uma delas.				
Metodologia sugerida	Estudos de caso podem ser empregados para explanação das várias áreas que compõem a Engenharia de Software e como elas foram utilizadas. Apresentações realizadas pelos estudantes ajudam a solidificar o conteúdo apresentado pelo estudante e, adicionalmente, criam oportunidades para esclarecer dúvidas.				

M1	**Leitura de Software**				
Detalhes	64 horas	Teóricas: 0	Práticas: 64	Pré-requisitos: nenhum	NE
Objetivo	Familiarizar o estudante com o que é “bom” por meio do estudo de projetos de desenvolvimento de software (ou seja, com o correspondente código, documentação, dados, ferramentas, processos utilizados e outros elementos empregados no desenvolvimento). A ambientação deve auxiliar a reconhecer o que é “bom” e, adicionalmente, a exercitar o que possivelmente mais fará em sua vida profissional: ler código (talvez mais que escrever).				
Ementa	Estudar, investigar, analisar e discutir projetos de software existentes e “vencedores”.				
Metodologia	O termo leitura deve ser interpretado no sentido de “arte ou modo de interpretar e fixar um texto de autor, segundo determinado critério”. O professor pode selecionar produtos *open source,* de reputação reconhecida, sobre os quais a análise do código disponível é realizada, por exemplo, por meio do acesso ao código fonte e de outros artefatos publicamente disponíveis. Nestes casos, a estrutura, a estratégia empregada, as ferramentas, o processo e outros deverão ser identificados e analisados. Esta análise deverá ser coletiva e de tal forma que o estudante possa se envolver com as opiniões dos colegas (todas moderadas pelo professor). O estudante deverá ser estimulado a confrontar o produto investigado com o que ele próprio produz, faria ou conhece.				
Bibliografia básica	• *Code Reading: The Open Source Perspective*, Diomidis Spinellis, Addison-Wesley, 2003 (aqui).				
Bibliografia complementar	Não se aplica.				

M2	**Desenvolvimento de Software Concorrente**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NE
Objetivo	Expor o estudante a conceitos, desafios e ferramentas disponíveis para o desenvolvimento de software concorrente e à prática de tal atividade.				
Ementa	Liveness. Safety. Semáforos. Locks. Threads. Deadlocks. Implementações de algoritmos concorrentes.				
Metodologia sugerida	Apresentação de problemas intrinsecamente concorrentes ou que se beneficiam de concorrência com a posterior implementação de soluções correspondentes. As soluções implementadas deverão exercitar e lidar com os conceitos da ementa, bem como as dificuldades inerentes.				
Bibliografia básica	• *Concurrent Programming in Java: Design Principles and Patterns,* Douglas Lea, Addison-Wesley, 3rd edition, 2006 (aqui).				
Bibliografia complementar	• *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects,* Douglas Schmidt et al., Wiley, 2000 (aqui). • *Pattern-Oriented Software Architecture Volume 3: Patterns for Resource Management,* Michael Kircher, Wiley, 2004 (aqui). • *Java Concurrency in Practice,* Brian Goetz et al., Addison-Wesley, 2006 (aqui).				

M3	**Desenvolvimento de Software para Persistência**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NE
Objetivo	Solidificar conceitos e habilidade de programação envolvendo persistência de informações.				
Ementa	Definição de persistência. Persistência empregando arquivos binários, documentos XML, objetos serializáveis, SGBDs. Tecnologias para persistência de informações. Persistência de objetos usando base relacional.				
Metodologia sugerida	Desenvolver um estudo de caso de complexidade não trivial (a ser implementado pelos estudantes) onde o conhecimento da ementa possa ser exercitado de forma pragmática.				
Bibliografia básica	• *Persistence in the Enterprise: A Guide to Persistence Technologies,* Geoffrey Hambrick et al., IBM Press, 2008 (aqui).				
Bibliografia complementar	• *Beginning EJB 3 Application Development: From Novice to Professional,* Raghu R. Kodali et al., Apress, 2006 (aqui). • *Pro EJB3: Java Persistence API,* Mike Keith and Merrick Schincariol, Apress, 2006 (aqui). • Spring Persistence - A Running Start, Paul Fisher e Solomon Duskis, APress, 2009 (aqui).				

M4	**Desenvolvimento de Software para Dispositivos Móveis**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NE
Objetivo	Ambientar o estudante com recursos, restrições, ambientes, ferramentas e tecnologias pertinentes a dispositivos móveis, bem como o desenvolvimento de software para tais dispositivos.				
Ementa	Noções de hardware de dispositivos móveis (celulares, PDAs e sensores). Ambientes, tecnologias e ferramentas para desenvolvimento de software para dispositivos móveis. Prática de desenvolvimento de uma aplicação para dispositivos móveis.				
Metodologia sugerida	O docente identifica um domínio e aplicação correspondente, a ser desenvolvida pelos estudantes. A aplicação deve estar devidamente especificada de tal forma a permitir o início imediato do desenvolvimento correspondente.				
Bibliografia básica	• *Mobile Phone Programming and its Application to Wireless Networking,* Frank H. P. Fitzek e Frank Reichert, Springer, 2007 (aqui).				
Bibliografia complementar	• *Beginning J2ME: From Novice to Professional,* Sing Li and Jonathan Knudsen, APress, 3rd edition, 2005 (aqui). • Kicking Butt with MIDP and MSA: Creating Great Mobile Applications, Jonathan Knudsen, Prentice-Hall, 2008 (aqui). • Microsoft® Mobile Development Handbook, Andy Wigley et al., Microsoft Press, 2007 (aqui).				

M5	**Desenvolvimento de Software para a Web**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NE
Objetivo	Exercitar o conhecimento da Engenharia de Software em aplicação que faz uso da web e, em consequência, exige o domínio de tecnologias pertinentes.				
Ementa	Tecnologias, técnicas, ferramentas e abordagens para o desenvolvimento de aplicações web.				
Metodologia sugerida	Caracterização de um domínio e de aplicação web consistente com o domínio, a ser desenvolvida pelos estudantes. A aplicação deverá ser cuidadosamente selecionada com o propósito de enfatizar aspectos relevantes para a disciplina.				
Bibliografia básica	• *Web Application Architecture: Principles, Protocols and Practice,* Leon Shklar e Richard Rosen, Wiley, 2003 (aqui).				
Bibliografia complementar	• *Beginning Web Programming with HTML, XHTML, and CSS,* Jon Duckett, Wrox, 2nd edition, 2008 (aqui). • *Professional Apache Tomcat 6,* Viveck Chopra et al., Wrox, 2007 (aqui). • *Spring in Action,* Craig Walls and Ryan Breidenbach, Manning, 2007 (aqui). • *Building Spring 2 Enterprise Applications,* Interface21 et al., Apress, 2007 (aqui). • *Murach's Java Servlets and JSP,* Joel Murcah, Mike Murach & Associates, 2008 (aqui). • *Beginning JSP, JSF and Tomcat Web Development: From Novice to Professional,* Giulio Zambon and Michael Sekler, Apress, 2007 (aqui).				

M6	**Integração de aplicações**				
Detalhes	64 horas	Teóricas: 16	Práticas: 48	Pré-requisitos: nenhum	NE
Objetivo	Fornecer uma clara noção de integração de aplicações e das abordagens sugeridas para tratar os desafios bem como habilitar o estudante a desenvolver software para integrar aplicações.				
Ementa	Definição de integração de aplicações. Desafios de integração. Abordagens de integração (transferência de arquivos, bases de dados compartilhadas, chamada de procedimento remoto e troca de mensagens). Padrões para integração de aplicações.				
Metodologia sugerida	Apresentação e discussão de casos didáticos de integração a serem solucionados pelos estudantes pelo emprego de uma ou mais abordagens. Um cenário mais sofisticado pode envolver a implementação e experimentação de padrões de integração e pode compor a maior parte da avaliação da disciplina.				
Bibliografia básica	• *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions,* Gregor Hohpe and Bobby Woolf, Addison-Wesley, 2003 (aqui).				

M6	**Integração de aplicações**
Bibliografia complementar	• *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects,* Douglas Schmidt et al., Wiley, 2000 (aqui). • *Pattern-Oriented Software Architecture Volume 4: Pattern Language for Distributed Computing,* Frank Buschmann et al., Wiley, 2007 (aqui). • *Remoting Patterns: Foundations for Enterprise, Internet and Realtime Distributed Object Middleware,* Markus Voelter et al., Wiley, 2004 (aqui).

M7	**Engenharia Econômica para Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Fornecer noções de engenharia econômica e outras questões econômicas aplicadas a software.				
Ementa	Escopo da Engenharia Econômica. Fornecimento, demanda e produção. Lucro produzido por capital (*interest*). Análise custo-benefício. Análise *breakeven*. Retorno de investimento. Avaliação de alternativas. Economia aplicada ao desenvolvimento de software.				
Metodologia sugerida	Análise de estudos de casos para a parte prática da disciplina. O docente também pode envolver empresas reais onde investigações econômicas sobre os produtos por elas desenvolvidos são realizadas.				
Bibliografia básica	• *Engineering Economy,* Willian G. Sullivan et al., 14th edition, Prentice-Hall, 2008 (aqui).				
Bibliografia complementar					

M8	**Técnicas Avançadas de Construção de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Solidificar a habilidade de programação do estudante por meio de técnicas avançadas de programação.				
Ementa	Observação, análise, investigação e prática de técnicas e soluções avançadas de desenvolvimento de software.				
Bibliografia básica	• *Beautiful Code: Leading Programmers Explain How They Think,* Andy Oram e Greg Wilson, O'Reilly Media, 2007 (aqui).				
Bibliografia complementar	• *Why Programs Fail: A Guide to Systematic Debugging,* Andreas Zeller, Morgan Kauffman, 2005 (aqui). Este livro é acompanhado de portal de apoio com slides, ferramentas e outros (aqui). • *Programming Challenges,* Steve S. Skiena e Miguel Revilla, Springer, 2003 (aqui). • *Programming Pearls,* John Bentley, Addison-Wesley, 2nd edition, 1999 (aqui).				

M9	**Prática em Engenharia de Software**				
Detalhes	64 horas	Teóricas: 0	Práticas: 64	Pré-requisitos: nenhum	NE
Objetivo	Desenvolver habilidades do estudante em desenvolvimento de software em ambiente tão próximo do real quanto possível, no contexto de uma Fábrica de Software.				
Ementa	Não definida.				
Metodologia sugerida	Atribuir ao estudante a responsabilidade pela consecução de um ou mais pacotes de trabalhos de um ou mais projetos em andamento na Fábrica de Software.				
Bibliografia básica					
Bibliografia complementar					

M10	**Integração 2**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Fornecer e elucidar uma visão coesa e integrada da Engenharia de Software de tal forma que o estudante desenvolva uma sólida percepção de como fazer uso do conhecimento desta área em toda a sua extensão.				
Ementa	Rever, exemplificar o emprego da Engenharia de Software em abrangência e profundidade. Integrar todo o conhecimento das disciplinas do curso de tal forma a permitir a compreensão, a relação entre elas, a importância, os produtos e atividades pertinentes a cada uma delas.				
Metodologia sugerida	Estudos de caso podem ser empregados para explanação das várias áreas que compõem a Engenharia de Software e como elas foram utilizadas. Apresentações realizadas pelos estudantes ajudam a solidificar o conteúdo apresentado pelo estudante e, adicionalmente, criam oportunidades para esclarecer dúvidas.				

M11	**Tópicos em Engenharia de Software**				
Detalhes	64 horas	Teóricas: 32	Práticas: 32	Pré-requisitos: nenhum	NE
Objetivo	Manter o estudante ciente de assuntos relevantes em Engenharia de Software não cobertos no curso ou fortalecer a formação do perfil do egresso em assunto de uma ou mais disciplinas. Por exemplo, sistemas críticos, tolerância a falhas, especificação formal, sistemas de tempo real e programação funcional, dentre outros, são assuntos candidatos (não cobertos por disciplinas). Adicionalmente, pode-se trabalhar conteúdo coberto por uma ou mais disciplinas. A decisão cabe à coordenação do curso, assistida tanto pelos estudantes quanto professores.				
Ementa	Não definida.				
Bibliografia básica					
Bibliografia complementar					

OPT1	**Mercado de Software**				
Detalhes	64 horas	Teóricas: 64	Práticas: 0	Pré-requisitos: nenhum	NE
Objetivo	Capacitar o estudante a "ler" o contexto corrente de software do mercado interno e global, suas tendências, características, exigências e noção do esforço necessário para satisfazê-lo.				
Ementa	Modelos de negócio para software (aluguel, serviço, *open* source). Leis, normas, impostos e legislação brasileira para o mercado local e para a exportação de software. Programas de incentivo à exportação de software. Fontes de recursos nacionais e internacionais para a produção de software. Características e exigências do mercado interno e externo. Identificação de oportunidades de inovação em software. Planos de negócio de software para o mercado nacional e global.				
Metodologia sugerida	Envolver os estudantes em atividade de pesquisa e identificação de oportunidade de exportação de software por meio da extensão de produtos e/ou serviços já oferecidos por empresas locais.				
Bibliografia básica					
Bibliografia complementar					

OPT2	**Introdução à Língua Brasileira de Sinais (LIBRAS)**				
Detalhes	64 horas	Teóricas: 64	Práticas: 0	Pré-requisitos: nenhum	NE
Objetivo					
Ementa	Introdução às práticas de compreensão e produção em LIBRAS através do uso de estrutura e funções comunicativas elementares. Concepções sobre a Língua de Sinais. O surdo e a sociedade.				
Metodologia sugerida	Envolver os estudantes em atividade de pesquisa e identificação de oportunidade de exportação de software por meio da extensão de produtos e/ou serviços já oferecidos por empresas locais.				
Bibliografia básica	• Felipe, T. e Monteiro, M. S., *LIBRAS em contexto.* Curso Básico. Brasília: Ministério da Educação e do Desporto/Secretaria de Educação Especial, 2001.				
Bibliografia complementar	• Nelson Pimenta, Livro + DVD Curso LIBRAS 1, terceira edição, LSB Vídeo, 2008. • Guiseppe Rinal, BRASIL, Secretaria de Educação Especial. Deficiência Auditiva. Série Atualidades Pedagógicas, no. 4, Brasília, SEESP, 1997. • Oliver Sacks, Vendo Vozes: uma viagem ao mundo dos surdos. Tradução Laura Motta. São Paulo, Editora Cia. das Letras, 1999. • Vítor da Fonseca, Inclusão: um guia para educadores. Porto Alegre, Artes Médicas, 1999. • Maria Cecília Rafael de Góes. Linguagem, surdez e educação. Campinas, SP, Editora Autores Associados, 1999.				

Considerações finais

Acreditamos que o contexto local foi contemplado da perspectiva de instituição que possui décadas de ensino na região. Houve um esforço significativo para ser abrangente e, ao mesmo tempo, projetar um curso onde prioridades encontram-se na “utilidade e aplicabilidade”, conforme sugerido por David Parnas para cursos de Engenharia de Software [Parnas 1999]. Para exercer esta utilidade e aplicabilidade, metade da carga horária do curso é dedicada a atividades práticas, a serem exercidas no ambiente de uma Fábrica de Software, que emprega o modelo de Melhoria de Processo do Software Brasileiro [MPS.BR]. Tudo isto devidamente balizado por pilares como o corpo de conhecimento em Engenharia de Software [SWEBOK 2004] e diretrizes para cursos de graduação em Engenharia de Software [SEEK].

Naturalmente, oportunidades de melhoria serão identificadas. O presente projeto pedagógico deverá ser revisado sempre na ocorrência de evento que justifique alteração ou por período não superior a quatro anos contados a partir de 2009. Isto é compatível com frequentes mudanças na área e reconhece que todo o esforço realizado é teórico e não conta com “referências” nacionais (este é o primeiro curso de graduação em Engenharia de Software conhecido no Brasil). Melhorias permitirão corrigir a rota do curso em direção ao perfil definido para o egresso. Tal perfil é ambicioso, mas reflexo do mercado e da esperança na produção de soluções de Tecnologia da Informação no Estado de Goiás.

Agradecimentos

Várias pessoas contribuíram com o conteúdo aqui descrito e, a todas, nossos agradecimentos.

Direitos

Portal do Curso
http://engenhariadesoftware.inf.br

Fábio Nogueira de Lucena
Instituto de Informática (UFG)
fabio@engenhariadesoftware.inf.br
Telefone: (62) 3521-1510

Auri Marcelo Rizzo Vincenzi
Instituto de Informática (UFG)
auri@inf.ufg.br
Telefone: (62) 3521-1510

Juliano Lopes de Oliveira
Instituto de Informática (UFG)
juliano@inf.ufg.br
Telefone: (62) 3521-1181

Plínio de Sá Leitão Júnior
Instituto de Informática (UFG)
plinio@inf.ufg.br
Telefone: (62) 3521-1181

Referências

ACM	*Association for Computing Machinery (ACM).* URL: http://www.acm.org.
ACM Cursos	URL: http://www.acm.org/education/curricula-recommendations
ACM Careers	*Computing: Degrees & Careers.* Disponível em http://computingcareers.acm.org.
ACM/IEEE-CS	*Software Engineering Code of Ethics and Professional Practice,* Version 5.2. Disponível em http://www.acm.org/about/se-code.
ACM/IEEE 2005	*Computing Curricula 2005: The Overview Report — covering undergraduate degree programs in Computer Science, Computer Engineering, Information Systems, Information Technology and Software Engineering,* 2005.
Bloom	*Taxonomia Bloom de Objetivos Educacionais.* URL: http://en.wikipedia.org/wiki/Bloom's_Taxonomy.
BRASSCOM	*Brasscom.* URL: http://brasscom.com.br.
CAPES	*Coordenação de Aperfeiçoamento de Pessoal e Nível Superior (CAPES).* URL: http://www.capes.gov.br.
Clements, 2009	Paul Clements e Mary Shaw, *"The Golden Age of Software Architecture" Revisited,* IEEE Software, july/august 2009, vol. 6, número 4, págs. 70-72, ISSN 0740-7459.
CMMI	*Capability Maturity Model® Integration (CMMI).* URL: http://www.sei.cmu.edu/cmmi.
CNPq	*Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).* URL: http://www.cnpq.br.
COMTEC	*Comunidade Tecnológica de Goiás (COMTEC).* URL: http://www.comtecgo.com.br.
CS 2008	*Computing Curricula Computer Science,* ACM/IEEE, 2008. URL: http://www.acm.org/education/curricula-recomendations.
Erdogmus, 2009	Hakan Erdogmus, *The Seven Traits of Superprofessionals*, IEEE Software, july 2009, págs. 4-6.
Estratégia	*Estratégia Ltda.* URL: http://www.estrategia.eti.br.
FINEP	*Financiadora de Estudos e Projetos (FINEP).* URL: http://www.finep.gov.br.
FS	*Fábrica de Software.* URL: http://en.wikipedia.org/wiki/Software_factory.
Garg 2008	*Software Engineering Education in India: Issues and Challenges,* Kirti Garg and Vasudeva Varma, 21st Conference on Software Engineering Education and Training, pp. 110-117, 2008
Herzberg	*Teoria dos Dois Fatores.* URL: http://en.wikipedia.org/wiki/Frederick_Herzberg.
IEEE 1058-1998	IEEE Standard for Software Project Management Plans
IEEE 1074-2006	IEEE Std 1074-2006, *IEEE Standard for Developing a Software Project Life Cycle Process,* 2006.
IEEE 12207-2008	IEEE Std 12207-2008*, Systems and Software Engineering — Software Life Cycle Processes, 2008.*
IEEE 14764-2006	IEEE Std 14764-2006*, Software Engineering—Software Life Cycle Processes—Maintenance*, 2006.
IEEE 610.12-1990	IEEE STD 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology,* IEEE Computer Society, 1990.
IEEE 730-2002	IEEE Std 730-2002, *IEEE Standard for Software Quality Assurance Plans*, 2002.
IEEE 828-2005	IEEE Std 828-2005, *IEEE Standard for Software Configuration Management Plans,* 2005.

IEEE CS	*IEEE Computer Society*. URL: http://www.computer.org.
IEEE Cursos	URL: http://computer.org/curriculum
ITIL	*Information Technology Infrastructure Library (ITIL).* URL: http://www.itil-officialsite.com.
LG	*LG Informática.* URL: http://www.lg.com.br.
MacGregor	*Teoria X e Teoria Y de MacGregor.* URL: http://en.wikipedia.org/wiki/Theory_X_and_theory_Y.
MPS.BR	*Melhoria de Processo do Software Brasileiro (MPS.BR).* URL: http://www.softex.br/mpsbr.
Parnas 1999	*Software Engineering Programs are Not Computer Science Programs,* David Lorges Parnas, IEEE Software, 1999.
PMBOK 2008	*A Guide to the Project Management Body of Knowledge,* PMBOK® Guide, Fourth Edition, ANSI/PMI 99-001-2008, 2008.
SBBDES 2008	*Simpósio Brasileiro de Banco de Dados e Engenharia de Software (SBBDES),* 2008. URL: http://sbbdes.ic.unicamp.br.
SBC	*Sociedade Brasileira de Computação (SBC).* URL: http://www.sbc.org.br.
SEBRAE	*Serviço de Apoio às Micro e Pequenas Empresas de Goiás (SEBRAE-GO).* URL: http://www.sebraego.com.br.
SE 2004	*Software Engineering 2004 — Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering,* 2004. Disponível em http://sites.computer.org/ccse/.
Softex	*Softex: Excelência em Software.* URL: http://www.softex.br.
Sommerville	*Software Engineering*, 8th edition, Ian Sommerville, Pearson Addison-Wesley, 2006 (aqui).
SWEBOK 2004	*Guide to the Software Engineering Body of Knowledge,* IEEE Computer Society, 2004. Disponível em http://swebok.org.