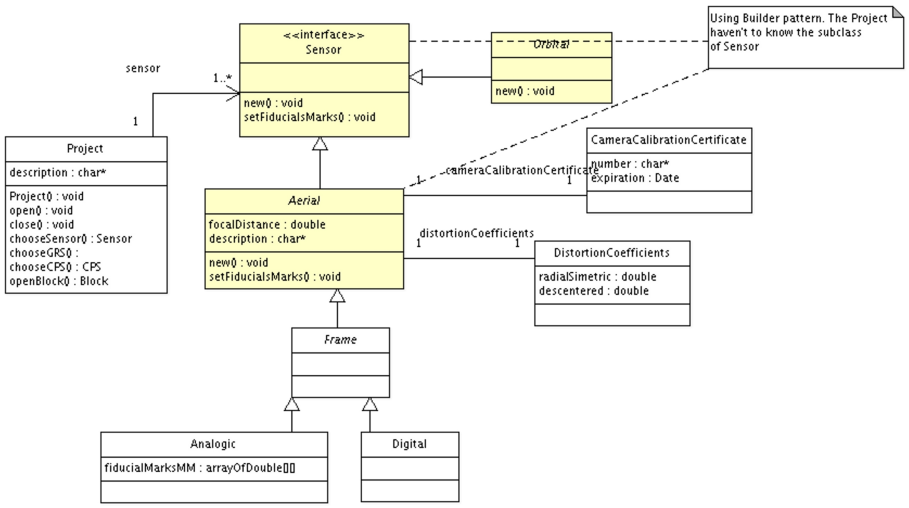
Figura 59 - Classe Sensor, com Padrão Builder



Fonte: O autor, 2008.

armazenamento.

3.3.3.2 A classe Sensor

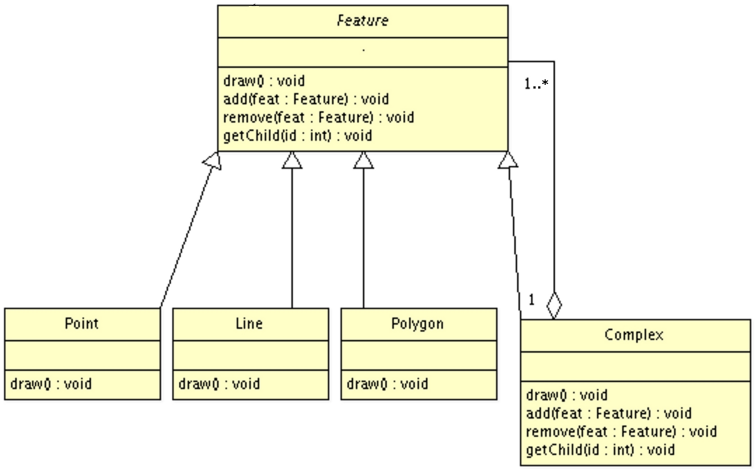
No modelo original a classe *Sensor* é abstrata e não era implementada até os níveis mais inferiores de suas heranças. Consequentemente, a classe *Project* (Projeto) precisa conhecer o tipo de sensor que irá trabalhar para que a classe *Sensor* possa instanciar corretamente a classe respectiva. A Figura 59 apresenta a solução desse problema com o uso de Padrões de Projeto

O padrão **Builder** permite que se tenha a mesma estrutura sem que, contudo, esteja obrigada a classe cliente, que nesse casso é a classe *Project*, conhecer previamente o tipo de sensor com o qual irá trabalhar. Ao invés disso, a classe *Sensor* se encarrega de instanciar a classe correta.

3.3.3.3 A classe Feição

As feições são as representações da geometria dos objetos naturais ou artificiais a serem mapeados. As feições podem ser introduzidas no modelo estereoscópico pela modelagem original, sendo mapeadas pelas classes *Point*, *Line* e *Polygon*. Porém, pode existir uma feição que possua mais de um desses tipos. Por exemplo, um edifício é uma feição composta por polígonos, linhas e até pontos. Esse problema pode, também, se modelado com padrões de projeto.

Figura 60 - Classe Feição com Padrão Composite



Fonte: O autor, 2008.

A Figura 60 presenta a modelagem através das classes de Feição com o padrão Composite. As classes *Point*, *Line* e *Polygon* passam a ser uma herança da classe *Feature* do qual a nova classe *Complex* (Complexo) também é herdada. Entretanto, um relacionamento de composição, simboliza que tal classe é composta de uma ou mais feições (relacionamento representado por 1..*). Isso permitirá compor o edifício como sendo um complexo de feições de diversos tipos (*Point*, *Line* e *Polygon*).

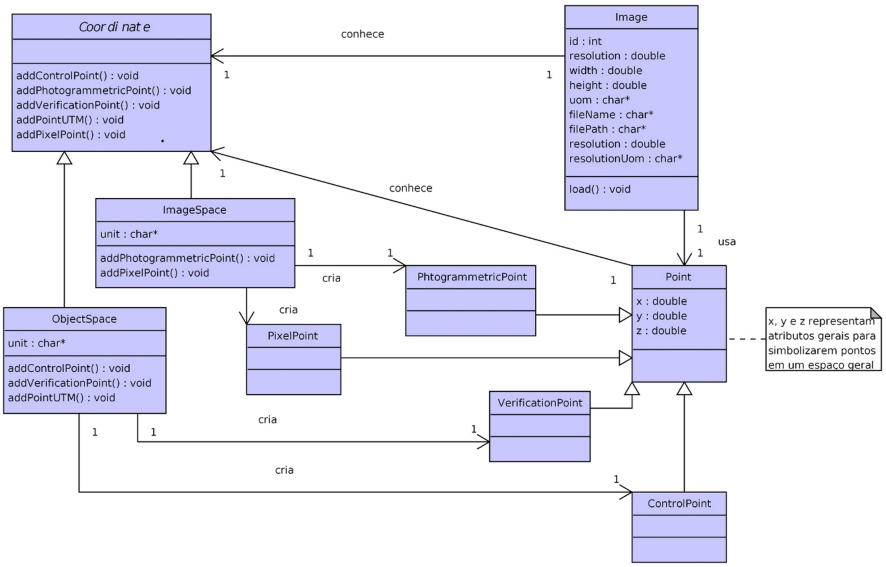
3.3.3.4 A modelagem de Pontos

Conforme descrito no modelo original, a modelagem de pontos da EFDE foi bastante complexa e gerou diversos modelos até se chegar ao modelo apresentado. Contudo, a abordagem com Padrões de Projeto nos forneceu uma solução bastante interessante e que conseguiu atender as demandas da EFDE.

O Padrão **Abstract Factory** nos permite criar uma fábrica abstrata de objetos de famílias distintas porém dependentes. Na Figura 61, a classe *Coordinate* é uma fábrica de pontos que, ora são criados a partir de espaço-imagem, ora a partir de espaço-objeto. A classe *Image*, cliente dessa fábrica, solicita a crição de qualquer tipo de ponto, que é providenciado pela fábrica. No caso da EFDE, a classe *Point* também é cliente da fábrica, pois existem pontos que são cadastrados, antes das imagens serem incluídas no projeto.

No capítulo seguinte será realizada uma análise comparativa entre os três modelos para indicar o que melhor se adequa à solução do E-FOTO.

Figura 61 - Modelagem de Pontos com Padrão Abstract Factory



Fonte: O autor, 2008.

4 ANÁLISE COMPARATIVA ENTRE OS MODELOS UML

Neste capítulo é apresentada uma análise comparativa entre os três modelos apresentados. Tal análise tem por objetivo uma definição de qual modelo se aplica melhor ao Projeto E-FOTO. Percebeu-se que uma análise simples e empírica não seria adequada pelo seu alto grau de subjetividade e, por isso, foi adotada uma técnica para cálculo de tamanho funcional de software. Essa técnica proporcionou quantificar o custo de desenvolvimento do Projeto E-FOTO em cada uma das modelagens apresentadas.

4.1 Análise por ponto de função e caso de uso

A análise por ponto de função (APF) é uma técnica utilizada há muito tempo para calcular o tamanho funcional de um sistema informatizado, oferencendo uma estimativa do tempo total que será utilizado e do custo financeiro (IFPUG, 2005). Através dessa métrica, os gestores podem acompanhar de perto os prazos estimados, comparando-os com os prazos efetivos e fazendo os ajustes necessários.

Em um passado recente, a métrica do desenvolvimento de um sistema era baseada no número de linhas do código-fonte. De acordo com Galvão (1999) a produtividade pode ser calculada pela fórmula abaixo:

$$P = \frac{MPT}{E} \tag{10}$$

onde:

- P é a produtividade;
- MPT é a Medida do Produto do Trabalho; e
- E é o esforço para produzir.

Durante algum tempo a única técnica para a métrica era a contagem de linhas de código-fonte (LOC[24])(FENTON; NEIL, 2000) e, baseado em um histórico, podia-se avaliar o tempo necessário para uma nova tarefa. Contudo, estudos mostraram que haviam problemas com essa técnica, principalmente em razão das linguagens de desenvolvimento rápido. A Tabela **??** mostra que a produtividade das linguagens de alto nível eram inferiores às de baixo nível, como por exemplo o Assembly.

[24] LOC é um acrônimo para *Lines of Code* que significa Linhas de Código

Tabela 7 - Produtividade por tipo de linguagem

	Tamanho	Esforço	Produtividade
Assembly	5000 linhas	28 semanas	714 linhas/mês
Linguagens de alto nivel	1500 linhas	20 semanas	300 linhas/mês

A técnica de ponto de função foi desenvolvida por Albrecht (1979) e aprimorada por ele mesmo e por Gaffney Albrecht e Gaffney (1983), no intuito de ser independente da linguagem e de se basear no contexto mais genérico de qualquer desenvolvimento de sistemas.

A primeira vantagem do uso dessa técnica consiste no fato dela ser transparente para todos os envolvidos, uma vez que boa parte dela é objetiva e a parte subjetiva é controlada por uma lista de características com forte significado. Além disso, com algumas modificações, pode ser utilizada para medir todo o processo de desenvolvimento de sistema, desde a concepção, até a implantação.

A técnica consiste das seguintes etapas:

1. Identificação das funções do sistema;
2. Classificação de cada função quanto à complexidade funcional relativa como: simples, média ou complexa;
3. Cálculo dos pontos de função brutos através da aplicação dos pesos de acordo com Tabela específica;
4. Avaliação das 14 características gerais de um sistema;
5. Determinação do Fator de Ajuste;
6. Cálculo dos pontos de função ajustados.

A contagem de pontos propostas por IFPUG (2005) é baseada nos requisitos do sistema e por isso é independente do pradigma de programação e análise aborados. Para essa dissertação, foi utilizado o estudo de caso 3 do IFPUG, que demonstra a contagem de pontos de função baseados em um sistema orientado a objetos.

4.1.1 A contagem da APF

O IFPUG (2005) descreve o processo de contagem de pontos de função em sete etapas:

1. Tipo de Contagem - estabelecer se a contagem se dará em um projeto em desenvolvimento, melhorias ou manutenção;
2. Fronteira da aplicação e escopo - estabelecer quais funções são importantes para o sistema. "(...) a fronteira da aplicação separa o projeto a ser contado dos usuários e das aplicações externas ao domínio do projeto.", segundo Edméia (2004);
3. Contar as funções de dados - consistem em Arquivos Lógicos Internos (ALI) e Arquivos de Interface Externa (AIE);
4. Contar as funções transacionais - consistem em Entradas Externas (EE), Saídas Externas (SE) e Consultas Externas (CE);
5. Determinar o ponto de função não ajustado;
6. Determinar o fator de ajuste;
7. Calcular os pontos de função ajustados.

Para realizar a contagem de pontos de função, precisa-se antes delimitar as regras de contagem que serão utilizadas. As técnicas da contagem são apresentadas diretamente no projeto da EFDE. Abaixo estão descritas as regras utilizadas para a contagem.

4.1.1.1 Tipo de Contagem

Apesar do Projeto E-FOTO estar em pleno funcionamento, inclusive com módulos já desenvolvidos, decidiu-se realizar a contagem como se fosse um projeto novo em desenvolvimento. Essa escolha é importante pois o cálculo do ponto de função poderá ter um peso maior ou menor de acordo com o tipo de contagem (IFPUG, 2005)

4.1.1.2 Fronteira da aplicação

Na delimitação da fronteira da aplicação, verifica-se que a EFDE é um único projeto, dividido em sub-projetos ou módulos. Desta forma, decidiu-se que cada módulo seria visto como uma aplicação independente e, assim, delimitar as fronteiras de cada módulo e as interfaces com os demais. Para cada modelo, foi realizada a contagem de pontos de função por módulo, definindo a fronteira através dos diagramas de caso de uso e sequência.

4.1.1.3 As funções

De acordo com Silva (2000), as funções podem ser divididas em dois grandes grupos:

- Funções do Tipo Dados - podem ser Arquivo Lógico Interno (ALI) ou Arquivo de Interface Externa (AIE);
- Funções do Tipo Transações - podem ser Entradas Externas (EE), Saída Externa (SE) ou Consulta Externa (CE).

Os divesos tipo de funções serão vistos sob a ótica de projetos OO (orientado a objetos) e, portanto, serão utilizados os jargões desse paradigma.

Função Tipo Dados

Um grupo de dados com características semelhantes e relacionáveis caracterizam um arquivo lógico de dados e no escopo da OO esses grupos formam as classes do sistema. Portanto, uma classe será sempre candidata a ser um arquivo lógico. Contudo há exceções, como será visto mais adiante.

A identificação dos arquivos lógicos internos (ALI) pode ser facilmente entendida pela seguinte regra, de acordo com Silva (2000): "(...) ALI é um grupo de dados relacionados, identificados pelo usuário, que reside dentro da fronteira do aplicativo e é mantido por entradas externas (...)". Daí pode-se concluir que toda classe que reside dentro da fronteira da aplicação e é mantida por essa aplicação é candidata a ser um ALI.

Por outro lado, arquivos de interface externa (AIE) são as classes que situam-se fora da fronteira do aplicativo mas são referenciadas pelo mesmo (SILVA, 2000). Essas classes são mantidas por outra aplicação.

Para se identificar as ALI e AIE, usa-se as seguintes regras (UEMURA T.; KUSUMOTO; INOUE, 1999):

- classes do esteriótipo "entidade"são candidatas a arquivos lógicos (ALI ou AIE) (FETCKE T.; ABRAN; NGUYEN, 1998).
- Complexidade - as complexidades são identificadas através do número de itens de dados e do número de registros lógicos. O número de itens de dados são os atributos de cada classe. Em caso de classe herdada, os atributos da superclasse também devem ser contados. Os registros lógicos são identificados através do tipo de relacionamento existente entre as classes:
  - relacionamentos do tipo agregação, idependente da cardinalidade, são candidatos a registros lógicos;

Tabela 8 - Complexidade ALI e AIE

Itens de Dados referenciados			
Registros Lógicos	1 a 19	20 a 50	51 ou mais
1	BAIXA	BAIXA	MÉDIA
2 a 5	BAIXA	MÉDIA	ALTA
6 ou mais	MÉDIA	ALTA	ALTA

- em caso de agregação em níveis, todos os objetos agregados serão registros lógicos do maior agregador;
- em um relacionamento do tipo herança, se a superclasse for abstrata essa é candidata a um registro lógico de cada classe que herda suas propriedades;
- se a superclasse não for abstrata, as subclasses são registros lógicos dessa;
- caso uma subclasse, considerada como registro lógico, também herde suas propriedades, toda sua descendência será registro lógico do arquivo lógico superior;
- toda classe de associação é candidata a um arquivo lógico, tendo como atributos, além dos seus próprios, as chaves primárias das classes associadas.

Após a contagem, aplica-se a Tabela 8 para verificar a complexidade de cada função.

Função Transacional

Os métodos das classes são candidatos a funções transacionais. Sua classificação, segue as seguintes regras (FETCKE T.; ABRAN; NGUYEN, 1998):

- caso um objeto se comunique com outro objeto e haja algum processamento no final, pode ser considerado como função transacional do tipo entrada externa (EE), saída externa (SE) ou consulta externa (CE);
- se a mensagem trocada não tiver argumento, não se considera função transacional;
- se a mensagem é enviada por um ator, é candidata a EE;
- se a classe envia uma mensagem e os parâmetros são apenas atributos da classe, é candidata a CE;
- se a classe envia uma mensagem e os parâmetros são cálculos realizados pela classe, é candidata a SE;

De acordo com a contagem, aplicam-se as tabelas abaixo para calcular a complexidade de cada Função transacional:

Tabela 9 - Complexidade Entrada Externa

Itens de Dados referenciados			
Arquivos referenciados	1 a 4	5 a 15	16 ou mais
0 a 1	BAIXA	BAIXA	MÉDIA
2	BAIXA	MÉDIA	ALTA
3 ou mais	MÉDIA	ALTA	ALTA

Tabela 10 - Complexidade Saída Externa

Itens de Dados referenciados			
Arquivos referenciados	1 a 5	6 a 19	20 ou mais
0 a 1	BAIXA	BAIXA	MÉDIA
2 a 3	BAIXA	MÉDIA	ALTA
4 ou mais	MÉDIA	ALTA	ALTA

Tabela 11 - Complexidade Consulta Externa

Itens de Dados referenciados			
Arquivos referenciados	1 a 5	6 a 19	20 ou mais
0 a 1	BAIXA	BAIXA	MÉDIA
2 a 3	BAIXA	MÉDIA	ALTA
4 ou mais	MÉDIA	ALTA	ALTA

Tabela 12 - Identificação das Funções do Tipo Dado

Função	Tipo
Projeto	ALI
Imagem	ALI
Orientação Interior	ALI
Orientação Exterior	ALI
Modelo Digital do Terreno	ALI
Ortorretificação	ALI
Fototriangulação	ALI
Normalização	ALI
Visão Estéreo	ALI

4.1.2 Fator de Ajuste

Para essa dissertação não foi necessário o cálculo do fator de ajuste, visto que tal cálculo é baseado em 14 características gerais de sistemas e que não fariam diferença na análise comparativa dos modelos de classes.

4.1.3 Divisão do modelo UML para tipificação dos pontos de função

Com base no modelo desenhado e nas regras descritas acima, realizou-se o trabalho de identificação da tipificação dos pontos de função do modelo orientado a objetos. Com os módulos bem definidos, foi possível identificar as entradas, consultas, saídas e interfaces externas, bem como as internas. A Tabela 12 relaciona as funções do tipo dado identificadas na EFDE.

A Tabela 13 mostra as funções do tipo transação que foram identificadas no modelo da EFDE.

Com a divisão das funções e suas respectivas identificações, pode-se realizar o cálculo dos Pontos de Função de cada um dos modelos desenhados e, através das tabelas de contagem definidas por IFPUG (2005), chegar aos Pontos de Função Não Ajustados.

A Tabela 14 mostra uma comparação entre os três modelos para o módulo de Ortorretificação. No **Apêndice D** são exibidas as tabelas completas dos três modelos, com as complexidades atribuídas a cada função.

Tabela 13 - Identificação das Funções do Tipo Transação

Função	Tipo
Abrir Projeto	EE
Criar Projeto	EE
Dados do Terreno	EE
Escolher Sensor	EE
Gravar Projeto	EE
Carregar Dados Projeto	CE
Incluir Imagem no Bloco	EE
Montar Bloco	SE
Salvar Dados Projeto	SE
Medir Marcas Fiduciais	EE
Calcular OI	EE
Medir Qualidade	SE
Medir Pontos	EE
Calcular OE	EE
Achar Ponto Homólogo	EE
Calcular DTM	EE
Escolher Grid	EE
Calcular Ortorretificação	EE
Adicionar Feição	EE

Tabela 14 - Cálculo do Ponto de Função Não Ajustado para Ortorretificação

Processo	PF UML	PF Aspecto	PF Padrão
Calcular Ortorretificação	6	6	6
Adicionar Feição	6	6	6
Visão Estéreo	10	10	10
Carregar dados projeto	6	6	6
Salvar dados projeto	5	7	5
Medir qualidade	4	4	4

Tabela 15 - Cálculo do Ponto de Função Não Ajustado para os três modelos UML

Modelo	Total PFNA
UML	306
Com Aspectos	309,6
Com Padrões de Projeto	310,8

4.2 **Resultados da Análise**

Na Tabela 15 são apresentados os resultados obtidos com a aplicação do método de cálculo de ponto de função para sistemas orientados a objetos. Pode-se observar que o modelo UML obteve o menor índice de Ponto de Função Não Ajustado entre os três modelos apresentados. Os índices dos modelos com Aspectos e com Padrões de Projeto são muito próximos, revelando que as alterações são, em termos de complexidade, similares. O modelo de Padrões de Projeto obteve o maior índice entre todos devido à característica intrínseca da aplicação de Padrões, isto é, a modelagem através de Padrões de Projeto passa, necessariamente, pela criação de diversas classes abstratas que são, em primeira instância, registros lógicos (vide seção 4.1.1.3).

De fato, os valores encontrados na APF são compatíveis com aqueles esperados em virtude das soluções adotadas, isto em função da APF ter sido realizada em um modelo conceitual. Tais modelos apresentam como característica a propriedade de se melhor identificar as classes do modelo, enquanto que os atributos e métodos são mais facilmente identificados em um modelo físico. Com isso, pode-se verificar que o modelo UML apresenta menos classes que os demais e, portanto, um PFNA menor. Já o modelo com Padrões de Projetos e Aspectos apresentam mais classes para permitir uma abstração maior da solução.

4.2.1 Média e Desvio-padrão do resultado

Foi realizado o cálculo da média aritmética simples e desvio padrão para os resultados encontrados na APF. Os valores são apresentados na tabela 16 abaixo:

Pode-se observar que os valores calculados da APF estão a uma distância da média inferior a um desvio padrão, demonstrando que tais valores são muito próximos, impossibilitando uma análise apenas objetiva para a definição de qual modelo melhor se aplica ao projeto E-FOTO. Por isso, a análise também foi realizada à luz das características estruturais de cada modelo-solução, avaliando os benefícios para a implementação e manutenção

Tabela 16 - Medidas Estatísticas do resultado da APF

x-x-	Valor
Média	308,8
Desvio Padrão	2,04

da EFDE.

Cabe salientar que essa análise por Ponto de Função foi realizada a partir de diagramas conceituais e, portanto, apresenta uma contagem que pode ser utilizada para quantificar o esforço necessário para a implementação do modelo físico da EFDE e não para a implementação da estação.

Em uma análise subjetiva das soluções utilizadas nos modelos, temos:

- UML - todos os três modelos são baseados na UML e, portanto, possuem características idênticas. Entre elas pode-se citar a reusabilidade como uma das mais importantes, permitindo uma manutenção mais fácil. Entretanto, o modelo não se utiliza de nenhuma técnica adicional que possa agregar valor, ficando a cargo do analista/projetista a preocupação com a qualidade do modelo físico a ser gerado;

- Aspectos - nesse modelo, a principal característica é a separação dos requisitos transversais do sistema. Essa separação garante ao modelo uma flexibilidade em uma posterior alteração ou manutenção. Contudo, a EFDE não apresenta muitos requisitos transversais para serem modelados;

- Padrões de Projeto - o modelo com utilização de Padrões de Projeto apresenta algumas características importantes: permite larga reutilização, apresenta um vocabulário comum entre projetistas e reduzem a complexidade do sistema. Na EFDE foi possível a aplicação de alguns padrões de projeto no modelo conceitual e ainda é possível que no modelo físico sejam identificados outros padrões a serem utilizados.

Tais características observadas indica o modelo UML com utilização de Padrões de Projeto para ser utilizado no modelo físico e implementação da EFDE, principalmente pela necessidade de adoção de soluções já testadas, estáveis e reutilizáveis, facilitando futuras manutenções e permitindo uma documentação mais padronizada.

5 CONCLUSÕES

O projeto E-FOTO possui grande importância para a comunidade acadêmica e científica por disponibilizar, gratuitamente e para qualquer fim, diversos produtos destinados ao ensino e pesquisa de fotogrametria digital, incluindo uma estação fotogramétrica digital com finalidade educacional (EFDE). Muitos dos algoritmos matemáticos utilizados pela fotogrametria digital são de domínio público e largamente utilizados desde a fotogrametria analítica. Apesar disto, diversas empresas espalhadas pelo mundo possuem soluções para fotogrametria digital de domínio privado.

O E-FOTO disponibiliza produtos de forma gratuita e livre para diversos perfis de usuários. Desde o usuário que apenas quer utilizar uma estação fotogramétrica digital, até os usuários mais avançados que pretendem utilizá-la para compreender o processo fotogramétrico. Além dos módulos do software utilizados nas diversas operações e os respectivos códigos-fonte, o projeto disponibiliza um E-book, livro digital sobre fotogrametria e manuais de uso da EFDE.

As atuais versões ou releases da EFDE que foram elaboradas, carecem de documentação técnica e de um trabalho de Engenharia de Software. Este trabalho se propôs, mesmo de forma tardia, à análise e projeto da EFDE através de técnicas e preceitos modernos da Engenharia de Software. Os passos que se seguiram foram feitos através do uso de metodologias consagradas. Todo o processo de engenharia de requisitos foi realizado minunciosamente, com o intuito de produzir um modelo de classes, segundo os conceitos da Orientação a Objetos (OO) e UML, que representasse, conceitualmente, uma estação fotogramétrica digital.

A modelagem conceitual foi realizada em um trabalho comparativo de três modelos distintos: UML, orientação a aspectos e utilizando Padrões de Projeto. A intenção era verificar qual modelo apresenta melhor custo/benefício, tanto para a implementação de melhorias, quanto para a correção de erros. A análise comparativa está respaldada em um estudo objetivo, com a utilização de contagem de Pontos de Função.

A dissertação permitiu a observação de que, apesar dos modelos terem apresentado valores muito próximos de contagem de Ponto de Função, a análise com Padrões de Projeto é a melhor alternativa para o projeto E-FOTO. Tal destaque se dá, principalmente, em virtude da necessidade de adoção de soluções já testadas e estáveis, facilitando futuras manutenções. Cabe salientar, que os modelos apresentados são conceituais, ou seja, representam a base conceitual de uma estação fotogramétrica digital e não sua implementação, modelo físico. Apesar disto, os produtos desta dissertação podem ser utilizados na documentação da EFDE, bem como servir de fonte de consulta para usuários da área de sistemas.

A partir deste trabalho, a EFDE ganhará um amadurecimento e permitirá o aco-

plamento de outros módulos, bem como a incorporação de outros profissionais de forma colaborativa, criando condições para o exercício da "liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades", uma das quatro liberdades do software livre.

Dentre os objetivos específicos desta dissertação, a comparação entre o modelo orientado a objetos e o outro que segue a lógica da programação orientada a aspectos (POA) tem grande destaque. A este respeito, pode-se mencionar que, entre essas duas abordagens, a UML apresenta melhores resultados para projetos similares ao E-FOTO. Isso se deve ao pouco espalhamento de código, característica intrínseca ao problema apresentado. A bibliografia sobre programação orientada a aspectos normalmente mostra exemplos que são do cotidiano humano: sistema bancário, sistema de viagens, sistema de biblioteca etc. De certo que a POA pode atender a estes sistemas de forma bem satisfatória, pois existe muito código semelhante espalhado por todo o sistema - autenticação de usuário, privilégios de uso, trilhas de auditoria. Contudo, projetos de características similares ao E-FOTO serão mais bem modelados com a UML.

Para trabalhos futuros, elenca-se as seguintes sugestões:

- Projeto físico da EFDE - a especificação física dos modelos de classe, bem como demais produtos da UML, dos módulos da EFDE;
- Implementação dos módulos - a implementação dos módulos de acordo com os modelos conceituais (e futuramente os físicos); este trabalho pode ser realizado através de desenvolvimento colaborativo[25];
- Alteração da arquitetura - implementação de uma arquitetura de sistema em "n-camadas, como por exemplo, a arquitetura MVC.

[25] Desenvolvimento de sistemas onde o código fica disponível para programadores do mundo todo contribuir.

REFERÊNCIAS

ALBRECHT, A. Measuring application development productivity. In: PROCEEDINGS OF JOINT SHARE, GUIDE AND IBM APPLICATION DEVELOPMENT SYMPOSIUM. Monterey, CA, 1979.

ALBRECHT, A. J.; GAFFNEY, J. E. *Software Functions, Source Lines of Code, and Development Effort Prediction: A Software Science Validation*. [S.l.]: IEEE Transactions on Software Engineering, 1983.

ALEXANDER, C. *Notes on the synthesis of form / Christopher Alexander*. Cambridge: Harvard University Press, 1964.

ALEXANDER, C. *The timeless way of building*. [S.l.]: Oxford University Press, 1979.

ANDRADE, J. *Fotogrametria*. [S.l.]: Ed. UFPr, 2003.

ARAUJO, J. et al. Aspect-oriented requirements with uml. In: *Aspect Modeling with UML workshop at the Fifth International Conference on the Unified Modeling Language and its Applications*. [S.l.: s.n.], 2002.

ASPRS. *Manual of Photogrammetry*. [S.l.]: American Society of Photogrammetry, 1966. 1220 p.

BOOCH G.; RUMBAUGH, J.; JACOBSON, I. *UML guia do usuário*. [S.l.]: Elsevier Campus, 2006.

BRITO, J. L. N. e. S.; COELHO, F. L. C. T. Projeto e-foto: Uma estação fotogramétrica digital educacional. In: *III Encontro de Iniciação Científica do Instituto Militar de Engenharia*. [S.l.: s.n.], 2002. Rio de Janeiro.

BRITO, J. L. N. S.; COELHO, L. C. *Fotogrametria Digital*. [S.l.]: Instituto Militar de Engenharia, 2002.

BRITO, J. L. N. S.; COELHO, L. C. T. F. Projeto e-foto: Uma estação fotogramétrica digital educacional. In: *Simpósio Brasileiro de Sensoriamento Remoto*. Belo Horizonte, Brasil: [s.n.], 2003.

CATAJY, H. *AspectUML aplicada à criação de sistemas de informaçao geográfica*. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro - Faculdade de Engenharia, 2006.

CMSEI. *CMMI Web Site*. 2008. (http://www.sei.cmu.edu/cmmi/index.html). Arquivo consultado em abril de 2008.

COELHO, L. et al. E-foto: An educational digital photogrammetric workstation. In: *The XXI Congress The International Society for Photogrammetry and Remote Sensing*. Pequim, China: [s.n.], 2008.

D'ALGE, J. Qualidade geométrica de imagens tm do landsat-5. In: *Congresso Brasileiro de Cartografia*. Brasil: [s.n.], 1987.

EDMÉIA, L. *Pontos de Caso de Uso e Pontos de Função na gestão de estimativa de tamanho de projetos de software orientados a objetos*. Dissertação (Mestrado) — Universidade Católica de Brasília, 2004.

ELMASRI, R.; NAVATHE, S. *Fundamentals of Database Systems*. [S.l.]: Menlo Park, CA: Addison-Wesley, 1994.

FENTON, N.; NEIL, M. *Software Metrics: Roadmap*. [S.l.]: Future of Software Engineering Limerick Ireland ACM, 2000. 359-370 p.

FETCKE T.; ABRAN, A.; NGUYEN, T.-H. *Mapping the OO-Jacobson Approach into Function Point Analysis. Technology of Object-Oriented Languages and Systems*. Los Alamitos, CA, USA: IEEE Comput. Soc, 1998.

GALVÃO, A. M. Pontos de função como ferramenta para gerenciamento de projetos de sistemas. Nota de aula. 1999.

GAMMA, E.; HELM, R. et al. *Padrões de Projeto: soluções reutilizáveis de software orientado a objetos*. [S.l.]: Bookman, 2000.

GEMAEL, C. *Introdução ao ajustamento de observações*. Curitiba: UFPR, 1994.

GRAF, R. *Modern dictionary of electronics*. [S.l.]: Newnes, 1999.

IFPUG. *Function Point Counting Practices: Case Study 3 - Object Oriented Analysis*. 2005. International Function Point Users Group.

INFORMÁTICA, P. *Dicionário OnLine*. 2008. (http://www.priberam.pt/dlpo). Arquivo consultado em julho de 2008.

KICZALES, G. et al. Aspect oriented-programming. In: *ECOOP - European Conference on Object-Oriented Programming, Spring-Verlag*. [S.l.: s.n.], 1997.

LADDAD, R. *AspectJ in Action: Practical Aspect-Oriented Programming*. [S.l.]: Manning,, 2003.

LEITE, J. *Engenharia de Requisitos: Notas de Aula - Parte IV*. 1994.

MATOS, A. V. *UML Prático e Descomplicado*. [S.l.]: Erica, 2002.

MIKHAIL E.M.; BETHEL, J.; MCGLONE, J. *Introduction to Modern Photogrammetry*. [S.l.]: John Wiley & Sons, INC., 2001.

MOTA, G. L. A. et al. Epp - esquema para o arquivo xml de projeto fotogramétrico da estação digital educacional livre e-foto. In: *XIII Simpósio Brasileiro de Sensoriamento Remoto, Florianópolis, Brasil*. [S.l.: s.n.], 2007.

OMG. *UML Infrastructure Specification*. [S.l.], 2007.

PRESSMAN, R. S. *Engenharia de Software*. São Paulo: Pressman, R. S., 1995.

REENSKAUG, T. The original mvc reports. Dept. of Informatics, University of Oslo. 1979.

REIS T.T; TOMMASELLI, A.; RUY, R. Módulo computacional para geração semi-automática de planos de vôos fotogramétricos. In: CONGRESSO BRASILEIRO DE CADASTRO TéCNICO MULTIFINALITáRIO. Florianópolis, Brasil, 2006.

ROYCE, W. *Mananging the Development of large software systems: concepts and techniques*. [S.l.]: Proc. IEEE WESTCON, Los Angeles, CA., 1970.

SCHENK, T.; CHO, W. *Resampling Digital Imagery to Epipolar Geometry*. [S.l.], 1992.

SEINTURIER, L.; RETAILL, J. P. *Foundations of Aop for J2ee Development*. [S.l.]: Apress, 2005.

SILVA, I. J. d. M. *Delphi 5 - Análise de Pontos de Função*. [S.l.]: Express Book, 2000. 277 p.

SILVA, R. P. *Arquivo XML de projeto fotogramétrico e sua auditoria no E-FOTO*. Dissertação (Monografia) — Universidade do Estado do Rio de Janeiro, 2008.

SILVEIRA, F. *Fototriangulação Pelo Método dos Feixes Perspectivos com Auto-calibração*. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro - Faculdade de Engenharia, 2007.

SILVEIRA, M. *Visualização e medição estereoscópicas de imagensfotogramétricas digitais*. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro, 2005.

SOKAL, J. et al. Normalização de imagens fotogramétricas digitais. In: *XXIII Congresso Brasileiro de Cartografia*. [S.l.: s.n.], 2007.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: Pearson Addison Wesley, 2005.

STALLMAN, R. *Projeto GNU*. 1983. Initial Announcement (http://www.gnu.org/gnu/initial-announcement.html). Arquivo consultado em outubro de 2008.

STALLMAN, R. *FSLA*. 2008. (http://www.gnu.org/gnu/initial-announcement.html). Arquivo consultado em outubro de 2008.

SYSTEMS, G. *Inc. Digital Photogrammetric Workstation User's Manual*. [S.l.], 1994.

TEMBA, P. *Fundamentos de Fotogrametria*. [S.l.]: UFMG, 2000.

TOMMASELLI A.M.G.;HASEGAWA, J.; GALO, M. Desenvolvimento de uma biblioteca de funções e classes para fotogrametria. In: XXI CONGRESSO BRASILEIRO DE CARTOGRAFIA. [S.l.], 2005.

UEMURA T.; KUSUMOTO, S.; INOUE, K. Function point measurement tool for uml design specification. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE METRICS. [S.l.], 1999.

WIEGERS, K. *Software Requirements*. [S.l.]: Microsoft Press, 1999.

WIENER, R.; PINSON, L. J. *Fundamentals of OOP and Data Structures in Java*. [S.l.]: Cambridge University Press, 2000.