Modelo de Desenvolvimento de Objetos de Aprendizagem Baseado em Metodologias Ágeis

Fábio R. Lapolli, Claudia L. R. Motta, Carlo E. T. Oliveira, Cristiane M. Cruz

Programa de Pós-Graduação em Informática - UFRJ
Caixa Postal 68.530 – 21.945-970 – Rio de Janeiro – RJ – Brasil

lapollimaster@gmail.com, {claudiam,cetoli}@nce.ufrj.br,
crismouracruz@hotmail.com

Abstract. *The development of learning objects requires instructional models of flexible development, because it is a software category with a group of specific requisites and singular technological demands. The agile methodology use for learning object development may facilitate this permitting the functionality based on the student's behavior requisites. The present article has the objective to describe the experiment the learning object development based on agile methodologies.*

Resumo. *O desenvolvimento de objetos de aprendizagem exige modelos instrucionais e de desenvolvimento flexíveis, por se tratar de uma categoria de software com um conjunto de requisitos bastante específicos e demandas tecnológicas singulares. O uso de metodologias ágeis para o desenvolvimento de objetos de aprendizagem pode facilitar esse processo permitindo a modelagem das funcionalidades baseados nos requisitos de comportamento do aluno. O presente artigo tem como objetivo discorrer sobre uma experiência no de desenvolvimento de objetos de aprendizagem, baseado em metodologias ágeis.*

1. Introdução

As instituições de ensino investem em seus recursos humanos visando produtividade, eficiência e qualidade na produção de material didático digital diferenciado, para isso, é imprescindível a escolha de uma metodologia mais adequada ao processo de desenvolvimento que atenda às necessidades organizacionais e pedagógicas de forma prática. A abordagem de desenvolvimento tradicional já não satisfaz as necessidades das propostas educacionais atuais, principalmente as de domínio complexo, pois nem sempre contribuem para a aprendizagem final. Portanto, é preciso pesquisar abordagens de desenvolvimento de recursos educacionais mais adequados a prática pedagógica. Nos últimos anos o uso de objetos de aprendizagem (OA) se tornou freqüente na prática pedagógica, entretanto, percebemos que o atual modelo instrucional e de desenvolvimento tem generalizado as características fundamentais desse recurso devido ao alto custo de produção e o vasto conteúdo educacional, o que inviabiliza a adequação das funcionalidades do OA as necessidades específicas de cada curso.

No desenvolvimento de um objeto de aprendizagem para processo de ensino e aprendizagem da disciplina Fundamentos da Meteorologia, pudemos perceber a necessidade de inovação no modelo de projeto instrucional e de desenvolvimento. Motivados também pela expansão e uso das técnicas de desenvolvimento ágil de software, verificamos que a aplicação de suas práticas poderia incorporar melhorias no projeto instrucional e, conseqüentemente, na produção de recursos didáticos.

A proposta deste trabalho aborda a adaptação do Behaviour-Driven Development (BDD) como uma das atividades de um projeto instrucional e de desenvolvimento, possibilitando a identificação dos cenários que permitam ao aluno interiorizar o conceito de tal forma que seja possível a sua abstração e aplicação em diferentes situações, inclusive as não previstas durante o seu treinamento.

O artigo está organizado em cinco seções. Nas seções 2, 3 e 4 apresentamos desenvolvimento orientado ao comportamento e modelo de proposta instrucional baseado em BDD e desenvolvimento ágil em objetos de aprendizagem. Na seção 5 apresentamos a metodologia de avaliação e por fim apresentamos as considerações finais desta proposta.

2. Desenvolvimento Orientado ao Comportamento

Test-Driven Development (TDD) foi uma metodologia ágil de desenvolvimento de software introduzida por Kent Beck (2003) para produzir o que se chama de "código limpo que funciona". Apesar de ser um processo simples, alguns desenvolvedores têm dificuldades de aplicar essa técnica no dia-a-dia e não conseguem guiar um projeto orientado a objetos a partir de testes.

No TDD os testes são escritos antes do código de produção, compilação e verificação de falhas, provendo o que é preciso para escrever um código de produção que satisfaça esses testes. O ciclo básico do TDD é formado pelas seguintes etapas:

1. Desenhe o que você quer que seu código faça;

2. Escreva o teste;

3. Execute o teste, provocando a falha;

4. Refatore o código para passar no teste;

5. Reinicie o ciclo.

Segundo Dave Astels (2003), a grande vantagem dessa prática não está nos testes gerados, mas sim no fato de se pensar no design antes de escrever a primeira linha de código, no momento em que se descreve o comportamento do sistema. Assim nasceu outra metodologia ágil, sendo o desenvolvimento orientado a comportamento (BDD, do inglês Behaviour-Driven Development), que com o tempo evoluiu para um processo que engloba desde a análise de requisitos, até o desenvolvimento do código.

O desenvolvimento orientado a comportamento é semelhante ao TDD em vários aspectos, mas embora as diferenças pareçam sutis influenciam grandemente no modo de criar um sistema. O BDD transfere o foco dos testes de implementação para os comportamentos que o sistema expõe.

Uma vantagem da abordagem BDD é que os comportamentos se modificam menos vezes do que os testes e, tipicamente, tais modificações refletem a necessidade de novas funcionalidades do sistema.

Os comportamentos do sistema também podem ser descritos em vários níveis de granularidade. Por exemplo, podemos falar sobre os comportamentos que o sistema deve apresentar no conjunto ou caracterizar comportamentos que caracterizam componentes individuais do sistema.

De modo geral, o BDD apresenta um framework baseado em três princípios:

1. A área de negócios e a de tecnologia precisam se referir a mesma parte do sistema da mesma forma;

2. Toda parte do sistema precisa ter um valor identificável e verificável para o negócio;

3. Analisar, projetar e planejar tudo de cima a baixo tem retorno decrescente.

Portanto, podemos definir o BDD como a união de várias práticas consideradas ágeis e úteis no desenvolvimento de software, cuja ênfase está nas funcionalidades de alto valor e na redução dos custos de mudança por meio da identificação do que de fato está sendo testado.

2.1. História do Usuário

Para Boyle et. al (2006), projetos de desenvolvimento de objetos de aprendizagem devem começar, com uma análise das necessidades do aluno. O resultado desta análise serve de base para o design e o processo de desenvolvimento.

Cohn (2004) afirma que a história de um usuário escrita por ele, descreve o que é valioso para os interessados de um sistema ou software e traz em si três características:

1. Descrição escrita da história, usada para planejamento e como lembrete;

2. Conversações sobre a história que servem para aprofundar os detalhes;

3. Testes e documentos que transmitem informações que podem ser utilizados para determinar quando uma história está completa.

Usando o acrônimo em inglês INVEST, Cohn (2004) diz que uma boa história do usuário é: **I**ndependente; **N**egóciável; **V**aliosa ao comprado; **E**stimável; **S**mall (pequena); **T**estável.

Na escrita da história do usuário utilizamos o seguinte modelo:

Como um [pessoa ou papel desempenhado]

Eu quero [funcionalidade]

Para que [benefício ou valor dessa funcionalidade ao negócio]

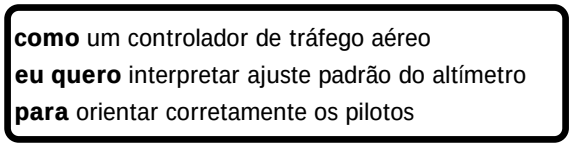

Figura 1 - Exemplo de Story Card usado no desenvolvimento do OA

Uma vantagem significativa de *story cards* é que como são freqüentemente escritas utilizando artefatos muito simples, como por exemplo, fichas, a identificação de novas histórias de usuário ou particionamento de histórias já existentes ou, ainda, a retirada de histórias que não são mais consideradas parte do escopo é uma atividade de processo considerada relativamente fácil. Essas histórias podem ser usadas para descrever uma grande variedade de requerimentos. Embora o nome histórias do usuário seja semelhante a "casos de uso", as histórias do usuário não são simplesmente uma versão reduzida de um caso de uso, elas são de fato um tipo diferente de artefato de desenvolvimento. No cartão definimos também os critérios

para a sua aceitação (Acceptance Criterias). Uma história do usuário só estará pronta quando todos os seus critérios de aceitação forem atendidos.

David Churchville (2006) apresenta algumas dicas para escrever boas histórias do usuário. Elas devem: i) estar focadas naquele que necessita da solução; ii) ser perfeitamente explicáveis em 30 segundos; iii) "caber" no trabalho a ser executado em uma semana pela equipe de desenvolvimento e iv) ser facilmente testáveis.

2.2. Critérios de Aceitação

O comportamento de um sistema constitui o seu critério de aceitação. Isto é, se o sistema cumpre todos os critérios de aceitação assumimos que ele se comporta corretamente. Caso contrário, não. Assim, Dan North (2007) aponta que um padrão para os critérios de aceitação da história do usuário é bastante livre, a fim de não ser artificial para os analistas, mas suficientemente estruturado de modo a permitir a divisão da história nos fragmentos que a compõem e automatizá-los. Em termos de cenários, os critérios de aceitação tomaram a seguinte forma:

Dado algum contexto inicial (os dados),

Quando ocorre um evento,

então assegure alguns resultados.

Em **dado** definimos tudo o que precisamos antes para **quando** o evento ocorrer **então** verificarmos o resultado. Note que uma funcionalidade pode admitir N-cenários.

Funcionalidades: Controlar o Tráfego Aéreo

Como um controlador de tráfego aéreo
Eu quero obter informações meteorológicas e climáticas
Para utilizar estas informações em outras atividades do Sistema

Cenário 1: Ajustar Altímetro
Dado que estou em vôo nivelado
E vou iniciar o procedimento de sua aproximação para pouso
E defino "altitude"
E defino "pressão atmosférica"
E defino "temperatura"
Quando "reunir" as informações altitude +pressão atmosférica+temperatura
Então preciso ver "o valor a ser ajustado no altímetro"

Cenário 2: Identificar Altitude
Dado que temos uma atmosfera padrão
E defino "pressão padrão"
E defino "nível médio do mar" (QFF)
Quando "reunir" as informações pressão padrão + QFF
Então preciso ver "a distância vertical que separa

Figura 2 - Exemplo de história de usuário

O uso deste padrão fornece uma linguagem ubíqua para a área de negócios e a de tecnologia, possibilitando a escrita de melhores softwares em que o foco está na essência, no que realmente importa.

3. Modelo de Projeto Instrucional Baseado em BDD

O modelo proposto foi idealizado levando em consideração os pressupostos de boas práticas das metodologias ágeis de desenvolvimento, mais especificamente o desenvolvimento dirigido pelo comportamento esperado do sistema, que visa aplicar testes unitários, funcionais e de integração para descrever como uma determinada funcionalidade deveria trabalhar. Sendo assim, o modelo foi baseado no padrão de teste de aceitação definido por Dan North (2007) visando atender algumas expectativas e necessidades apontadas por instrutores e alunos durante as entrevistas semi-estruturadas. Assim definimos o Modelo de Projeto Instrucional Dirigido pelo Comportamento, composto por cinco etapas: (1) Programa Oficial da Disciplina; (2) Proposta de Atividades; (3) Plano de Atividades; (4) Formas de Mediação; (5) Análise do Processo e do Produto.

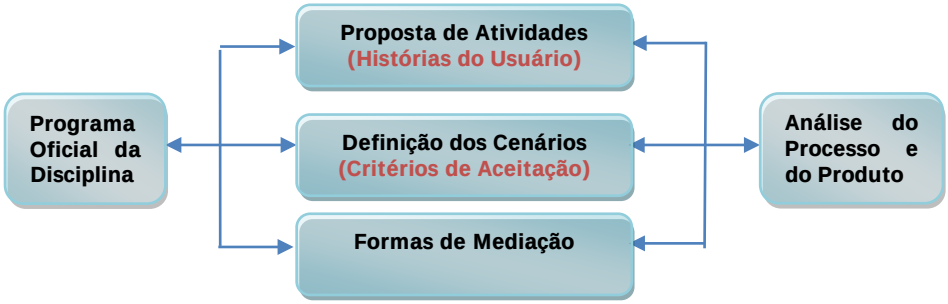

Figura 3 - Modelo de projeto instrucional dirigido pelo comportamento

1ª Etapa: Programa Oficial da Disciplina

Nesta etapa a equipe responsável pela área de Treinamento e Desenvolvimento deve identificar os conteúdos (conhecimentos/habilidades) que permitirão alcançar os objetivos (resultados) esperados. A esta combinação chamamos de **Programa Oficial da Disciplina**.

Cursos que visam preparar profissionais para atuar em Sistemas Complexos contêm no seu programa oficial, diversas disciplinas distribuídas por vários grupos disciplinares. Cada disciplina ocupa um lugar próprio na formação básica destes profissionais, porém o programa oficial deve favorecer a aquisição simultânea de saberes oriundos de diferentes áreas dado sua característica multidisciplinar e interdisciplinar.

O produto desta etapa é, portanto, um Programa não linear (em alguns momentos) que possibilite diferentes percursos de aprendizagem, com vistas a potencializar os resultados a partir do perfil de cada grupo de profissionais.

2ª Etapa: Proposta de Atividades

Da proposta de atividades de treinamento e desenvolvimento dos profissionais que irão atuar em Sistemas Complexos importa salientar que ela precisa atender aos conhecimentos, habilidades e comportamentos desejados para a realização de procedimentos operacionais previstos em sua rotina de trabalho.

Durante esta etapa são definidos os objetivos de aprendizagem específicos para cada assunto; os conteúdos que serão desenvolvidos; e as atividades de aprendizagem. Cabe salientar que as atividades de aprendizagem devem estar fundamentadas nas teorias de aprendizagem adequadas tanto para o ensino do conteúdo quanto para as habilidades e comportamentos a serem desenvolvidos.

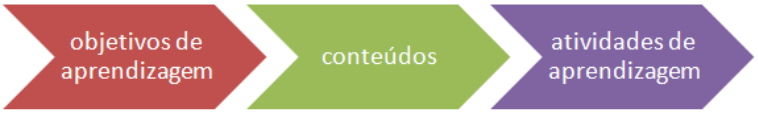

Figura 4 - Produtos da etapa proposta de atividades

Uma vez elaborada a proposta de atividades podemos iniciar a etapa de identificação dos cenários que serão utilizados para contextualizar os conhecimentos necessários para a realização das tarefas, apresentada a seguir.

3ª Etapa: Definição dos Cenários

Segundo definição do dicionário Aurélio, **cenário** é o lugar onde decorre uma ação ou parte da ação de uma peça, romance, filme etc. Sob o ponto de vista educacional, cenários são representações realistas (simulações) com a finalidade de prover a experimentação e exploração de fenômenos (hands-on). Nesta etapa são recomendadas algumas atividades da análise do trabalho cognitivo como, por exemplo, entrevistas com *expertises* e observação *in loco*.

Existem diferentes métodos e ferramentas para realizar a análise do trabalho cognitivo. Para que atenda a expectativa de construção de cenários educacionais, os métodos e ferramentas utilizados na identificação e definição de cenários devem incluir o mapeamento das tarefas, identificando e priorizando os pontos de decisão crítica. A Figura 5 mostra os produtos desta etapa. Eles são bastante úteis para explicitar como um conceito deve ser apresentado, de maneira tal que facilite a compreensão de sua a aplicação prática.

Figura 5 - Produtos da etapa definição dos cenários

Estes produtos são de extrema importância para a escolha das formas de mediação e o ponto de partida para suas respectivas construções.

4ª Etapa: Formas de Mediação

Dentre as ações delineadas no modelo de projeto instrucional dirigido pelo comportamento, conta a escolha e construção das formas de mediação que serão utilizadas para apoiar e/ou facilitar a aprendizagem dos que almejam uma formação profissional com mais eficiência, eficácia e agilidade. Assim, nesta etapa, a equipe de Treinamento e Desenvolvimento especifica e cria os recursos educacionais adequados para facilitar o ensino e aprendizagem de profissionais para atuar em Sistemas Complexos. De modo semelhante à etapa Proposta de Atividades, uma mesma *storyboard* pode dar origem a diferentes recursos

educacionais. Além disso, o modelo possibilita combinar diferentes teorias de aprendizagem em uma mesma forma de mediação.

5ª Etapa: Análise do Processo e do Produto

Embora o termo análise seja freqüentemente utilizado para designar a fase inicial de um processo, no modelo de projeto instrucional dirigido pelo comportamento ele representa nossa constante busca pelos interesses e aspectos relevantes a serem considerados na formulação e construção de um curso de formação profissional. Nesta etapa, quanto mais precisas forem as informações coletadas sobre o funcionamento do Sistema e o comportamento esperado para os profissionais que nele atuam, maiores são as possibilidades de planejar de forma mais plena e consciente a disponibilização da disciplina, o que ela pode oferecer e as estratégias mais eficazes para facilitar a aprendizagem. Uma das vantagens do modelo proposto é permitir a análise recursiva do processo e do produto permitindo incorporar melhorias ao longo do processo, o que viabiliza um processo de educação flexível, através de diferentes recursos que despertem no aluno o interesse pelo seu auto desenvolvimento.

4. Desenvolvimento Ágil de Objetos de Aprendizagem

A fim de identificar os requisitos e funcionalidades que deveriam ser abordados desenvolvidos nesse OA, realizamos entrevistas semi-estruturadas com instrutores e alunos no Instituto de Controle do Espaço Aéreo (ICEA), em setembro de 2007, onde apuramos as dificuldades em se construir cognitivamente um modelo do momento em que os conceitos abstratos de Meteorologia influenciam significativamente nos procedimentos operacionais.

Na elaboração da proposta de atividades contamos com a colaboração de um controlador de tráfego aéreo com uma gama de conhecimentos, habilidades, convicções e conceitos adquiridos ao longo dos 20 anos de atividade profissional. Sua contribuição faz com que o *workflow* de produção de um objeto de aprendizagem dirigido a treinamentos para atuar em Sistemas Complexos pareça diferente de um OA 'simples', ao inserir a participação de *expertises* nos processos de identificação de cenários e produção de recursos educacionais.

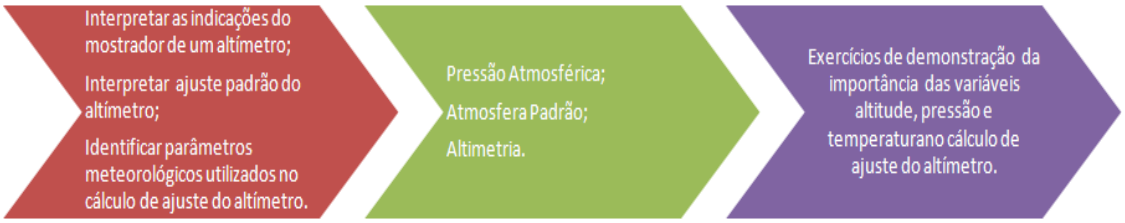

Figura 6 - Proposta de atividade do aplicativo

Cabe ressaltar que um mesmo conteúdo pode admitir mais que uma atividade de aprendizagem assim como uma mesma atividade de aprendizagem ser utilizada para alcançar diferentes objetivos previstos pelos grupos disciplinares.

A identificação do cenário (contexto) foi obtida por meio de entrevistas semi-estruturadas, em que foram identificados os momentos em que os conceitos relacionados à pressão atmosférica, atmosfera padrão e altimetria influenciam significativamente nos procedimentos operacionais. Desse modo, a idéia principal do objeto de aprendizagem é possibilitar a revisão dos conceitos relacionados à pressão atmosférica, atmosfera padrão e altimetria por meio de exercícios práticos que estimulem sua compreensão e fixação.

Dentro do modelo instrucional proposto de desenvolvimento de AO, a metodologia mais eficaz para atender as demandas variacionais de modelagem das funcionalidades do aplicativo foi utilizada a metodologia ágil de *Extreme Programming* (XP). Essa metodologia ágil permite desenvolver aplicativos com requisitos vagos, imprecisos ou em constante mudança, através da estratégia de constante acompanhamento e realização de vários pequenos ajustes ao longo do desenvolvimento.

Esta abordagem segundo Yeomans (*apud*. Boyle, 2006) salientam:

- Prototipagem rápida e iterativa;
- Uso de pequenas equipes ágeis;
- Usuário na equipe de projeto;
- Ênfase em produtos em vez de seguir processos definidos;
- Prazos apertados, embora controlados por métodos de gestão do tempo.

Os quatro valores fundamentais da metodologia XP são: comunicação, simplicidade, *feedback* e coragem. A partir desses valores, possui como princípios básicos: *feedback* rápido, presumir simplicidade, mudanças incrementais, flexibilidade frente a mudanças e trabalho de qualidade. Dentre as variáveis de controle em projetos (custo, tempo, qualidade e escopo), há um foco explícito em escopo. Para alcançar controle, recomenda-se a priorização de funcionalidades que representem maior valor possível para o negócio. Desta forma, caso seja necessário a diminuição de escopo, as funcionalidades menos valiosas serão adiadas ou canceladas. O desenvolvimento de desenvolvimento do OA foi apoiado, em um conjunto de boas práticas em XP que é definido a seguir de acordo com Beck (2000):

Jogo de Planejamento (*Planning Game*): Nesse modelo, há uma interação muito próxima entre cliente e desenvolvedor, cabendo aos programadores estimarem o esforço necessário para a execução de histórias de clientes e ao cliente a decidir sobre o escopo e o prazo desenvolvimento das versões para acompanhamento do andamento do projeto. No desenvolvimento do objetos de aprendizagem, os clientes foram os instrutores, podendo ser também, especialistas, professores ou o designer instrucional que são pessoas responsáveis por determinarem essas funcionalidades e prioridades a partir do modelo pedagógico e de sua experiência e prática em sala de aula.

Metáfora (*Metaphor*): Todo o sistema é definido por um conjunto de metáforas entre o cliente e os programadores. Esse sistema de códigos de linguagem é compartilhada no descrevendo como o sistema funciona. O estabelecimento desse código visa facilitar a comunicação com o cliente traduzindo esse conjunto de funcionalidades em uma linguagem próxima a realidade dele. Nesse caso, o sistema de códigos foi construído tendo como referencial a realidade do aluno, identificada pelo professor ou pelo designer instrucional.

Equipe Coesa *(Whole Team*): No projeto recomenda-se a composição de equipes não muito numerosas com um representante de cada área de desenvolvimento. Fizeram parte da equipe, o cliente, o programador, o designer e o designer instrucional que estiveram em sintonia com professor onde a coesão permitiu uma melhor integração das partes de desenvolvimento.

Projeto Simples (*Simple Design*): A ênfase está em projetar a solução mais simples possível e exeqüível neste momento. Complexidade desnecessária e códigos extras são removidos

imediatamente, buscando implementar as funcionalidades essenciais para o cumprimento da proposta pedagógica.

Programação em Pares (*Pair Programming*): A programação é realizada em par/dupla num único computador com o aplicativo sendo revisto por duas pessoas, evitando e diminuindo assim a possibilidade de erros. Esse mesmo modelo foi adotado em outras partes de desenvolvimento do aplicativo, como no design.

Padrões de Codificação (*Coding Standards*): São estabelecidos códigos padronizados pela equipe de desenvolvimento a fim de integrar e facilitar o processo de construção do aplicativo. No caso do objeto de aprendizagem, a codificação foi pautada na linguagem técnica da área.

Posse Coletiva (*Collective Ownership*): O código-fonte é compartilhado entre os desenvolvedores podendo ser modificado ao mesmo tempo e construído de forma colaborativa. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema podendo ser feita a atualização do desenvolvimento através de repositórios de versões compartilhado pela equipe.

Refatoração (*Refactoring*): Definição e redefinição da arquitetura a todo o momento em um aperfeiçoamento contínuo do projeto, após a consolidação de uma funcionalidade, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. No caso da programação, refatorar melhora a clareza (leitura) do código, dividindo-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código-fonte. Nas áreas de programação e design, a refatoração foi constante, de maneira integrada, com o aperfeiçoamento do código e do designer de forma progressiva, simultânea e complementar.

Reuniões em Pé (*Stand-up Meeting*): Realização de reuniões de curta duração entre os membros da equipe para discutir tarefas realizadas e tarefas a serem realizadas pela equipe. Esse formato é utilizado com o objetivo de conseguir maior concentração dos membros durante a reunião. Esse modelo foi adaptado em alguns momentos, com o uso de ferramentas de comunicação síncrona, no caso o Skype, devido à impossibilidade de deslocamento do cliente.

Integração Contínua (*Continuous Integration*): A integração de novas funcionalidades é realizada de maneira contínua e imediata evitando a possibilidade de conflitos e erros no código fonte, permitindo acompanhar o status real da programação pelo cliente, facilitando a identificação de ajustes a serem realizados nas funcionalidades do aplicativo. No desenvolvimento do AO, houve a integração contínua de novas funcionalidades não só na programação, como também no design, a fim atender a proposta pedagógica.

Pequenas Versões (*Small Releases*): O desenvolvimento modulado do aplicativo permitiu a liberação de pequenas versões com partes essenciais para a utilização e testagem simultaneamente a seqüência de desenvolvimento e aperfeiçoamento do aplicativo. Esse processo permitiu o acompanhamento pelo cliente, orientando os ajustes a serem realizados.

Testes de Aceitação (*Customer Tests*): Testes unitários executados antes do desenvolvimento final do código e são executados de forma contínua ao longo do projeto, sendo os clientes responsáveis por especificar os testes funcionais através da relação de requisitos de funcionamento do objeto de aprendizagem.

Figura 7 - Objeto de aprendizagem para trabalhar conceitos de altimetria

O modelo de desenvolvimento proposto e seguido, resultou no Objeto de Aprendizado chamado **ClearaNCE** (Figura 7) que trabalha os conceitos de altimetria que envolvem variáveis climáticas no ajuste da altitude voada por uma aeronave. Essas variáveis são: temperatura e pressão atmosférica aferidas localmente em cada aeroporto do país. O aplicativo foi desenvolvido em Flash junto com a linguagem de programação Action Script, integrada a busca e captura de dados em linguagem XML diretamente da Rede de Meteorologia do Comando da Aeronáutica (REDMET), dados esses referentes às condições meteorológicas do aeroporto escolhido. Nele, o aluno é estimulado a descobrir qual a relação entre temperatura local e a altitude da aeronave observando a modificação do cenário de acordo com as condições climáticas é usada para representar a temperatura. O conhecimento sobre pressão atmosférica está relacionado à temperatura através da escala barométrica sendo a escala mais baixa referente à temperatura mais baixa (e vice-versa). Através desta escala, o aluno é orientado a fazer o ajuste do altímetro. Esse ajuste também é feito a partir dos dados numéricos capturados. Nesta etapa, ocorrendo à variabilidade da temperatura e da pressão atmosférica de acordo com a localidade escolhida há a necessidade de ajuste do altímetro, desta forma, o aluno é estimulado a descobrir essa correlação entre as variáveis.

5. Avaliação

A análise do processo foi realizada através de estudo de caso com instrutores de controle de tráfego aéreo com mais de 20 anos de experiência. A partir dessa prática analisamos a relação experimentação – objeto de aprendizagem para verificar se a estrutura aplicada no ClearaNCE proporcionaria uma boa receptividade e percepção da aplicação prática do conteúdo dentro de uma proposta diferenciada, verificando se a metodologia utilizada possibilitaria ou não a ação investigativa e reflexiva do aluno sobre os conceitos abordados.

Os participantes reforçaram que o OA conseguiu representar a aplicação dos conceitos de maneira correta e completa, facilitando o entendimento do procedimento operacional ao dar uma noção concreta da importância destes Fundamentos da Meteorologia. Para eles, OA como este que apresentam os conceitos conjugados permitem generalizar e contextualizar a aplicação concreta do conceito levando-os a conscientização de sua importância, a partir do momento em que o conceito está embasado, as decisões são mais conscientes. De uma maneira geral, eles julgam ser importante a inclusão de recursos como esse em cursos de formação para atuar em

sistemas complexos uma vez que apóiam a compreensão de procedimentos em que há combinação de diferentes variáveis ao possibilitar a visualização dos procedimentos, através de representações eliminado ou reduzindo lacunas que levam a dificuldades durante a participação do aluno no estágio prático.

6. Considerações Finais

O modelo apresentado foi baseado nos pressupostos das metodologias ágeis de desenvolvimento, mais especificamente o desenvolvimento dirigido pelo comportamento esperado do sistema. Ele é aplicável as 3 importantes fases do planejamento: projeção de finalidades (objetivos), formas de mediação (construção de recursos) e metodologia de desenvolvimento. A finalidade da aplicação dos modelos de captura da história do usuário e de é estimular a participação dos *expertises* e futuros usuários do sistema durante o desenvolvimento do software. No âmbito educacional, o que se pretende com tal prática é aumentar o número de cenários que possam contribuir com a formação profissional, possibilitando ao aluno interiorizar conhecimento suficientemente necessário de tal forma que seja possível a sua abstração e aplicação em diferentes situações, inclusive, as não previstas durante o seu treinamento. Assim, cabe a equipe de treinamento e desenvolvimento motivar a participação das partes envolvidas e interessadas na formação de futuros profissionais, mostrando a importância das práticas de desenvolvimento ágil para a definição de objetivos e construção de recursos didáticos, podendo o mesmo ser expandido as outras etapas de modo que a preparação, a implementação, a avaliação, e a revisão da instrução seja visto como um processo integrado.

7. Referências

Astels, D., (2003) Test-Driven Development: A Practical Guide; Prentice-Hall PTR.

Beck, K., (2003) Test-Driven Development: By Example; Addison-Wesley.

_____., (2000) Extreme Programming Explained: Embrace Change; Addison-Wesley.

Churchville, D,. (2006), Disponível em : http://www.extremeplanner.com/blog (acessado em 10/06/2009)

Cohn, M., (2004) User Stories Applied: For Agile Software Development. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

ICEA - Comando da Aeronáutica Instituto de Controle do Espaço Aéreo, *"Apostila de Treinamento dos Controladores de Vôo"*.

North, D., (2007) *Introducing: Behaviour-driven development*, Disponível em: http://dannorth.net/introducing-bdd/ (acessado em 3/06/2009).

_____., (2007) *Whats in a Story*, Disponível em: http://dannorth.net/whats-in-a-story/ (acessado em 3/06/2009).

Boyle, T.; Cook, J.; Windle, R.; Wharrad, H.; Jeeder, D.; Alton, Rob.. *An Agile method for developing learning objects.* Proceedings of the 23rd annual ascilite conference: Who's learning? Whose technology? Sydney, Australia, 2006.