Ensino de Desenvolvimento de Jogos Digitais Baseado em Metodologias Ágeis: o Projeto Primeira Habilitação

Leonardo Carlos Comotti Kasperavicius, Luis Naito Mendes Bezerra, Luciano Silva, Ismar Frango Silveira

Universidade Cruzeiro do Sul
R. Galvão Bueno, 868 – 01506-000 – São Paulo, SP – Brasil

{leonardo.kasperavicius,luis.naito,luciano.silva,ismar.silveira}@unicsul.br

Abstract. *Teaching of game development is a non-trivial task, since it is a software category with a set of very specific requirements and singular technological demands. Using agile methodologies for game development can make easier such process, consequently influencing related learning. This paper has as goal present a teaching experience on digital game development, showing how the application of agile methodologies for software development could have a meaningful impact on the learning of students from a Digital Game undergraduate technical course.*

Resumo. *O ensino de desenvolvimento de jogos é uma tarefa não-trivial, 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 jogos pode facilitar esse processo de desenvolvimento e conseqüentemente influenciar na aprendizagem desse processo. O presente artigo tem como objetivo discorrer sobre uma experiência no ensino de desenvolvimento de jogos digitais, mostrando como a aplicação de metodologias ágeis de desenvolvimento de software pode ter um impacto significativo na aprendizagem dos alunos de um curso superior em Tecnologia em Jogos Digitais.*

1. Introdução

A Metodologia Institucional "Aprender na Prática", que prevê "a ação educativa na participação ativa e crítica do aluno em sua aquisição de conhecimentos práticos e teóricos" [UNICSUL, 2004] vem sendo aplicada na condução das disciplinas ligadas a Projetos, nominalmente, Fundamentos de Jogos, Projeto de Jogos I e II e Projeto Final Integrado, responsáveis por conduzir a proposta de cada aluno desenvolver ao menos um jogo a cada semestre. Tal prática didática permite que os alunos tenham construído, ao final de cada uma dessas disciplinas, jogos com ordem crescente de complexidade, os quais passam a fazer parte do primeiro portfólio da vida profissional dos mesmos.

Nesse sentido, o projeto "Primeira Habilitação" nasceu a partir de uma parceria técnica e científica entre a Universidade Cruzeiro do Sul e a empresa J&W Tecnologia em Trânsito. Tal projeto visou o desenvolvimento de jogos e simuladores relacionados aos exames nacionais de habilitação (Figura 1). Neste projeto, foram envolvidos seis alunos do quarto semestre curso de Tecnologia em Jogos Digitais da Universidade Cruzeiro do Sul (UNICSUL), em São Paulo, capital. A participação dos alunos teve caráter extra-curricular, ou seja, não ocorreu no contexto de uma disciplina do curso.

Figura 1. Jogos e simuladores desenvolvidos

O artigo encontra-se organizado como se segue: o item 2 discorre sobre o uso de metodologias ágeis no desenvolvimento do jogos. O terceiro item enfoca especificamente Programação Extrema, sendo que o quarto item estabelece um relacionamento entre esta metodologia e a aprendizagem dos alunos. Por fim, algumas considerações finais e trabalhos futuros são apresentados. Todas as imagens dos jogos utilizadas neste artigo foram gentilmente cedidas pela empresa J&W.

2. Metodologias Ágeis no Desenvolvimento de Jogos

Diversos autores vêm adaptando os processos clássicos de Engenharia de Software para o desenvolvimento de jogos. [Flynt e Salem, 2004], [Rucker, 2002]. Segundo Perucia et al [2005], o processo de desenvolvimento de jogos apresenta classicamente uma série de etapas, exibidas na Figura 2 a seguir:

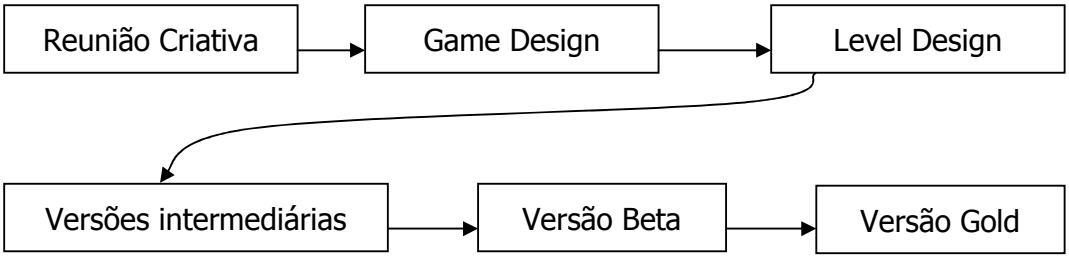

Figura 2. Etapas no processo de desenvolvimento de jogos

Atualmente, o desenvolvimento do software em geral, e o desenvolvimento de jogos, em específico, esbarram com requisitos de tempo cada vez menores, sendo muitas vezes inviável a aplicação de metodologias clássicas de Engenharia de Software. Este desafio de produzir software em prazos exíguos pode ser enfrentado através do emprego de técnicas que priorizem o desenvolvimento ágil [Highsmith et al., 2001] [Cockburn, 2002], como é o caso da Programação Extrema (*eXtreme Programming* – XP) [Beck, 1999] e de outros métodos [Abrahamsson et al., 2002].

Métodos ágeis priorizam a minimização de riscos através do desenvolvimento de software em iterações temporais relativamente curtas, levando a equipe de software à reanálise de prioridades ao final de cada iteração. Tais métodos têm se mostrado eficazes em projetos de software com prazos limitados, equipes reduzidas e complexidade razoavelmente baixa [Sutherland, 2004] [Cardoso,2004] [Ramos e Penteado, 2007]. Entretanto, na visão de autores como Lippert e Roock (2001), metodologias ágeis como XP podem ser adaptadas para modelagem de domínios complexos.

Apesar de haver um forte criticismo a respeito dessas metodologias, especialmente no tocante a uma suposta "indisciplina" inerente em relação à documentação de software, classicamente considerada o cerne dos métodos ditos tradicionais, metodologias ágeis não dispensam a documentação, tampouco a relegam a segundo plano. Ocorre, na realidade, uma priorização das comunicações *peer-to-peer* e uma redução dos artefatos documentais àqueles cuja produção seja essencial para o desenvolvimento do projeto de software. Uma relação recorrente entre métodos ágeis e "tradicionais" é que enquanto estes tentam predizer (e se precaver) em relação às mudanças, aqueles são adaptáveis a elas.

Projetos de jogos digitais demonstram ser adequados ao uso de metodologias ágeis, por se tratar de aplicações de domínio restrito, suscetíveis a mudanças e com requisitos específicos.

3. Princípios da Programação Extrema em Projeto e Desenvolvimento de Jogos Digitais

Programação Extrema(XP) [Beck, 1999] é uma metodologia ágil para pequenas e médias que irão desenvolver software com requisitos vagos, imprecisos ou em constante mudança. Para tanto, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de software.

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, abraçar 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.

A XP incentiva o controle da qualidade como variável do projeto, pois o pequeno ganho de curto prazo na produtividade, ao diminuir qualidade, não é compensado por perdas (ou até impedimentos) a médio e longo prazo. Para alcançar controle de qualidade em XP, existe um conjunto bem definido de boas práticas que será definido a seguir, com indicações de sua aplicação em projeto e desenvolvimento de jogos:

- **Jogo de Planejamento (*Planning Game*):** O desenvolvimento é feito em iterações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. Essa reunião recebe o nome de Jogo do Planejamento. Nela, o cliente identifica prioridades e os desenvolvedores as estimam. O cliente é essencial neste processo e assim ele fica sabendo o que está acontecendo e o que vai acontecer no projeto. Como o escopo é reavaliado semanalmente, o projeto é regido por um contrato de escopo negociável, que difere significativamente das formas tradicionais de contratação de projetos de software. Ao final de cada semana, o cliente recebe novas funcionalidades, completamente testadas e prontas para serem postas em produção. No contexto de cursos de jogos, o cliente é representado pelo professor que reavalia, semanalmente, o escopo do projeto.
- **Pequenas Versões (*Small Releases*):** A liberação de pequenas versões funcionais do projecto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando. As versões chegam a ser ainda menores que as produzidas por outras metodologias incrementais, como o RUP. Estas pequenas versões, no contexto de jogos, são denominadas *game demos* e devem oferecer funcionalidades não só para os processos de *game testing* quanto da aceitação do professor (cliente).
- **Metáfora (*Metaphor*):** Procura facilitar a comunicação com o cliente, entendendo a realidade dele. O conceito de rápido para um cliente de um sistema jurídico é diferente para um programador experiente em controlar comunicação em sistemas em tempo real, como controle de tráfego aéreo. É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto. No contexto de ensino de jogos, estas metáforas não são necessárias, uma vez que o professor conhece bem o domínio de jogos. Porém, recomenda-se que o teste de aceitação seja acompanhado por um outro professor que não seja de área de jogos, de tal forma que os alunos possam praticar o uso de metáforas.
- **Projeto Simples (*Simple Design*):** Simplicidade é um princípio da XP. Um erro comum ao adotar essa prática é a confusão por parte dos programadores de código simples e código fácil. Nem sempre o código mais fácil de ser desenvolvido levará a solução mais simples por parte de projeto. O controle da simplicidade em projeto e codificação não é uma tarefa simples na área de jogos. Uma boa prática nesta linha é utilizar algumas métricas bem conhecidas em Engenharia de Software para quantificar a complexidade do projeto e desenvolvimento.
- **Equipe Coesa *(Whole Team*):** A equipe de desenvolvimento é formada pelo cliente e pela equipe de desenvolvimento. Ambos precisam estar em sintonia e o professor representa um papel mediador bastante efetivo para manter a coesão.

Normalmente, recomenda-se que as equipes para os experimentos iniciais de XP não sejam muito numerosas.

- **Testes de Aceitação (*Customer Tests*):** São testes construídos pelo cliente e conjunto de analistas e testadores, para aceitar um determinado requisito do sistema. Em cursos de jogos, estes testes poderiam ser construídos, por exemplo, por um grupo de professores que ministram aulas para a turma no semestre em que o projeto é desenvolvido.
- **Ritmo Sustentável (*Sustainable Pace*):** Trabalhar com qualidade, buscando ter ritmo de trabalho saudável, sem esforços extremos. Em disciplinas de projetos de jogos, já existe um ritmo semanal de trabalho (horário da disciplina), que pode ser complementado por atividades extra-classe.
- **Reuniões em pé *(Stand-up Meeting*):** Reuniões em pé para não se perder o foco nos assuntos, produzindo reuniões rápidas, apenas abordando tarefas realizadas e tarefas a realizar pela equipe. No contexto de jogos, estas reuniões poderiam ser realizadas no início de cada aula, com a turma toda em pé frente a uma lousa cujo conteúdo deve ser preparado previamente pelo professor.
- **Posse Coletiva (*Collective Ownership*):** O código-fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema. Neste contexto, recomenda-se o uso de algum repositório de versões (CVS, por exemplo), com amplo acesso público aos integrantes da equipe.
- **Programação em Pares (*Pair Programming*):** é a programação em par/dupla num único computador. Desta forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de erros (*bugs*). No contexto de jogos, o processo de programação em pares pode ser extrapolado para outras atividades como modelagem, produção de áudio e animação.
- **Padrões de Codificação (*Coding Standards*):** A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Desta forma parecerá que todo o código fonte foi editado pela mesma pessoa, mesmo quando a equipe possui 10 ou 100 membros. Este padrão de codificação deve ser estabelecido pelo professor no início do semestre e amplamente divulgado através da WEB ou de forma impressa.
- **Desenvolvimento Orientado a Testes (*Test Driven Development*):** Primeiro deve-se criar os testes unitários e depois criar o código para que os testes funcionem. Esta abordagem é complexa no início, pois vai contra o processo de desenvolvimento convencional de software. No início da aplicação da XP, o professor pode criar um conjunto de testes como exemplo e, à medida que os alunos forem ganhando experiência na especificação dos testes, vão sendo transferidas responsabilidades.
- **Refatoração (*Refactoring*):** É um processo que permite a melhoria continua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Refatorar melhora a clareza (leitura) do código, divide-o em módulos mais coesos e de maior reaproveitamento,

evitando a duplicação de código-fonte. Neste item, deve-se estimular o uso de ferramentas automáticas de detecção de pontos de refatoração. No contexto de jogos, o processo de refatoração também pode ser extrapolado para outras atividades como modelagem, produção de áudio e animação.

- **Integração Contínua (*Continuous Integration*):** Sempre que produzir uma nova funcionalidade, nunca deve-se esperar uma semana para integrar à versão atual do sistema. Isto só aumenta a possibilidade de conflitos e a possibilidade de erros no código fonte. Integrar de forma contínua permite saber o status real da programação. No contexto de projeto e desenvolvimento de jogos, esta integração deve ser acompanhada de forma muito precisa pelo professor. Sugere-se, neste contexto, a divisão das aulas de projeto e desenvolvimento em dois encontros semanais para aumentar a efetividade do controle.

O tópico a seguir relata algumas destas práticas aplicadas no projeto e desenvolvimento do projeto em questão, com avaliação de seus impactos na aprendizagem dos alunos.

4. Uso de XP na Prática e o Impacto na Aprendizagem

Na fase inicial de análise de viabilidade do projeto "Primeira Habilitação", levantaram-se as questões de prazo e custo para o cliente. Após um breve levantamento de requisitos, foi acordado que os jogos seriam desenvolvidos em quatro meses e que a empresa contrataria seis alunos como estagiários por este mesmo período.

A turma do quarto semestre do curso de Jogos Digitais era composta na época de 36 alunos, e os únicos que tinham experiência com algum tipo de desenvolvimento já se encontravam empregados. Desta forma, o processo de seleção foi baseado no interesse que os alunos demonstraram nas entrevistas e na possibilidade de comprometimento com o projeto.

Levando em consideração o tamanho da equipe e o prazo do projeto, no início do projeto foi definida a utilização de Programação Extrema. Nesse contexto, os conceitos inerentes a essa metodologia foram empregados de forma implícita pelo professor-orientador, pois nenhuma disciplina do curso havia abordado o tema até o dado momento. No âmbito do projeto, os seguintes aspectos de Programação Extrema foram empregados:

***Pair Programming* -** Os seis alunos selecionados foram divididos em pares, o que foi muito importante para o projeto em vários aspectos. O primeiro deles diz respeito ao conhecimento específico de cada um: como descrito anteriormente os alunos selecionados constituíram uma equipe bem heterogênea, e a disposição dos pares inicialmente foi feita de forma que os alunos suprissem mutuamente as dificuldades dos demais (dois alunos desenhavam muito bem, outros dois programavam muito bem, um deles já conhecia bem arquiteturas *web-based*, e assim por diante). Isso ajudou muito no início de cada iteração, onde havia um esforço muito grande na parte de modelagem do sistema e na elaboração de esboços para que o cliente pudesse aprovar as artes-finais.

Além disso, uma outra vantagem foi percebida durante os períodos intensivos de desenvolvimento e testes. À medida que um dos alunos atuava como um "co-piloto", ele captava aspectos que passavam despercebidos pelo outro. Um exemplo interessante

pode ser narrado: certa vez, era necessário testar o algoritmo de sorteio das placas de trânsito no "Jogo das Placas" (Figura 3). Para constatar a validade do algoritmo, cada uma das 120 placas de trânsito deveria aparecer uma única vez durante cada jogada. Além disso, o algoritmo deveria garantir que todas as placas fossem sorteadas durante as fases do jogo. A quantidade de placas semelhantes dificultava a tarefa de validação, e após várias tentativas sem sucesso a pessoa incumbida da função pediu auxílio ao seu colega, e desta forma somente conseguiram comprovar a corretude do algoritmo.

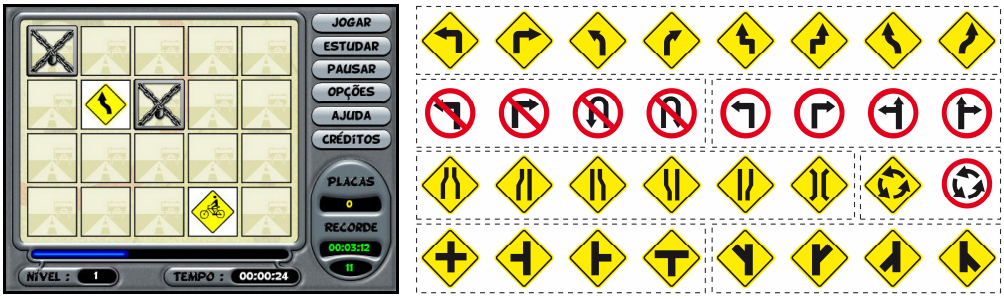

Figura 3. No "Jogo das Placas", a quantidade de símbolos semelhantes tornou praticamente impossível os testes de validação de forma individual

Test-Driven Development - Após o término do desenvolvimento de um componente/módulo, um dos alunos (ou os dois) já perfazia os testes de unidade e integração. Enquanto o componente não estivesse sem erros perceptíveis, ele não era incorporado ao sistema. Todos os erros encontrados eram "cadastrados" em um documento de testes, ilustrado na Figura 4, e eram corrigidos priorizando-se os erros com níveis de gravidade mais altos.

Problema 01	
Descoberto em: 02/04/2007	**Responsável**: Fábio
Categoria: Erro de Funcionamento	
Gravidade: leve	
Descrição: Quando se está estacionado logo após o jogo da baliza e aciona as setas, o reflexo do carro nos retrovisores some. Além disto, toda vez que o carro está parado e o usuário dá seta para começar a ação, as imagens dos retrovisores dão uma leve "piscadinha".	
Sugestões de melhoramento : Sincronizar a imagem dos retrovisores às animações.	
Solução: Método "disparaAnimacao" consertado.	
Solucionado em: 04/04/2007	**Responsável**: Leonardo

Figura 4. Exemplo de documentaçao de testes utilizada no projeto

Coding Standard - No começo do projeto foram definidas quais as padronizações que seriam obedecidas. Isso foi decisivo para garantir que qualquer membro da equipe fosse capaz de compreender todos os códigos desenvolvidos. Com base nos *templates* do RUP (*Rational Unified Process*) [Probasco, 2001] foi definido um documento com essas informações, nomeado "Guia de Padronização de Desenvolvimento".

Continuous Integration – Como uma forma de dar uma oportunidade aos outros alunos que não foram selecionados para participar do projeto, foi criado um concurso para o desenvolvimento de mini-games. Neste concurso, todos os jogos submetidos que fossem aprovados seriam incorporados ao jogo "Cara a Cara Com o Trânsito". A cada jogo recebido, os alunos faziam a validação e os testes, o incorporavam ao sistema em fase

de produção e verificavam se o sistema preservava sua funcionalidade ou se a integração havia afetado algum outro módulo. A figura 5 abaixo mostra alguns dos mini-games que foram incorporados ao sistema:

Figura 5. A cada mini-game incorporado ao sistema em desenvolvimento, os alunos verificavam se o sistema preservava sua funcionalidade

***Collective Code Ownership* –** Cada aluno apresentou possuir uma habilidade especial (um desenhava, possuía um raciocínio lógico apurado, outro conhecia bem arquiteturas *web-based*, etc.). Utilizando as práticas acima mencionadas tornou-se uma tarefa simples modificar ou aprimorar elementos desenvolvidos anteriormente, já que a criação de jogos envolve criatividade e trabalho em grupo. A Figura 6 ilustra a evolução do cenário do jogo "Cara a Cara com o Trânsito", onde cada um dos alunos teve participação, desde a criação do cenário, texturização e criação dos logotipos de cada "casa" do tabuleiro. Para controle de versionamento, foi utilizado o sistema *Subversion* [Pilato, 2004].

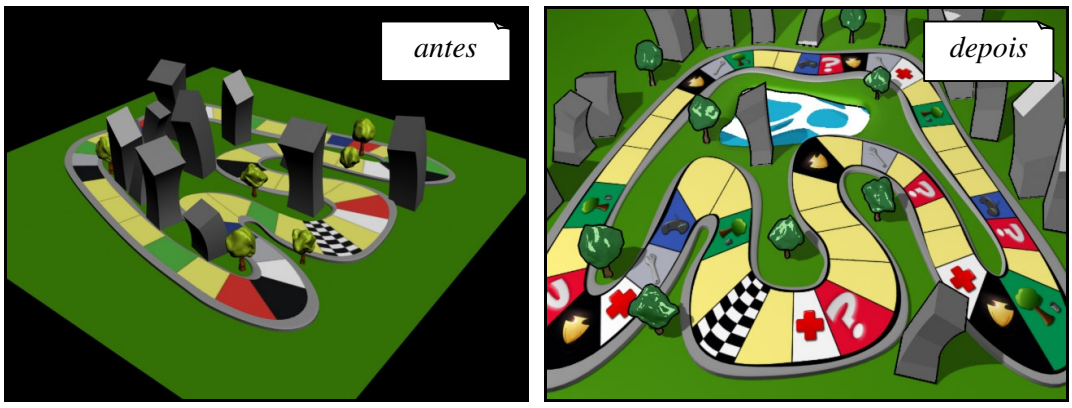

Figura 6. No projeto, não só a parte de programação como também a parte visual eram aprimoradas coletivamente

Small Releases - Esta é uma prática que foi bastante utilizada, especialmente considerando-se o escopo do projeto e as condições de mercado. O requerente do projeto tinha um perfil empreendedor e participava ativamente do processo de desenvolvimento iterativo, o que não ocorre em muitos casos. Com base nas prioridades estabelecidas pela empresa foi especificado um cronograma (Figura 7) que era de conhecimento dos alunos, os quais o cumpriram com rigor propiciando a entrega dos módulos nos prazos previstos.

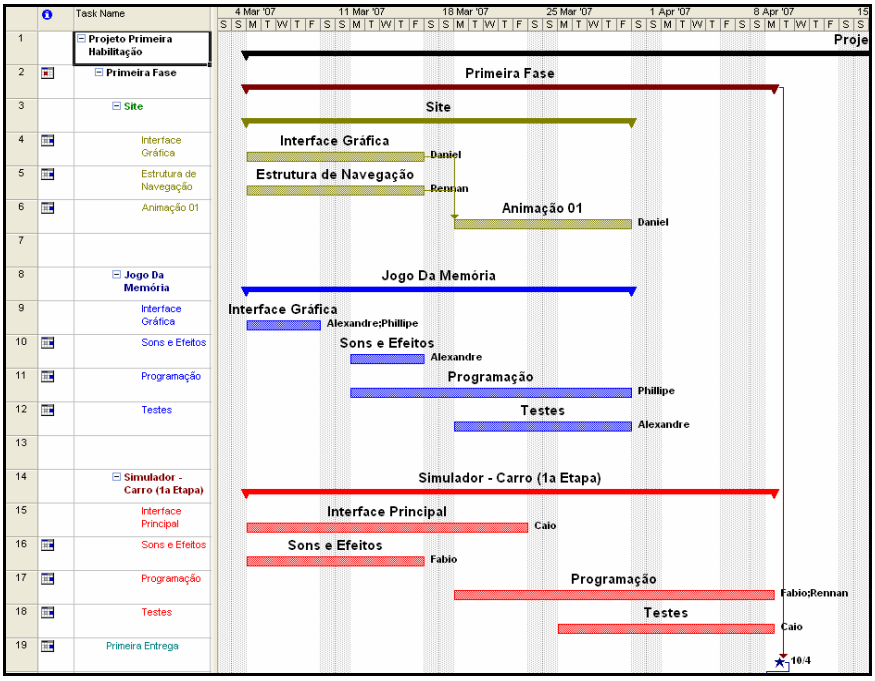

Figura 7. Cronograma do projeto até a data da primeira entrega

5. Considerações Finais e Trabalhos Futuros

O curso de Tecnologia em Design de Jogos Digitais da UNICSUL vem atendendo ao seu objetivo inicial, que é formar profissionais qualificados para atender a um mercado carente de recursos humanos e em franca expansão. Este trabalho apresentou a experiência com o ensino de desenvolvimento de jogos em um projeto real utilizando metodologias ágeis, com alguns exemplos dos aspectos que foram mais benéficos durante este processo.

Em relação aos resultados atingidos podemos citar que, mesmo tendo um prazo extremamente curto, os alunos conseguiram desenvolver os jogos no prazo previsto e os jogos entregues não apresentaram erros aparentes até o momento. A nosso ver, as técnicas empregadas foram fundamentais para atingir os objetivos previstos. É importante ressaltar também a evolução surpreendente dos alunos envolvidos diretamente no projeto, não só nas disciplinas do curso, mas também profissionalmente. Para dar continuidade a este trabalho, as metodologias ágeis poderiam ser empregadas em projetos maiores, envolvendo mais alunos. Desta forma seria possível fazer uma análise quantitativa do impacto do uso dessas metodologias no ensino de desenvolvimento de jogos.

Referências

Abrahamsson, P., Salo, O., Ronkainen, J. E Warsta, J. (2002) *Agile Software Development Methods – Review and Analysis*. Vuorimiehentie, Finlândia: VTT Pub.

Beck, K. (1999) *Extreme Programming Explained: Embrace Change*. New Jersey, EUA: Addison-Wesley.

Cardoso, C. H. R. (2004) Aplicando práticas de eXtreme Programming (XP) em equipes SW-CMM nível 2. VI Simpósio Internacional de Melhoria de Processos de Software – SIMPROS 2004 São Paulo, Brasil.

Cockburn, A. (2002) *Agile Software Development*. 3rd ed. Addison WesleyProfessional.

Flynt, J. P e Salem, O. (2004) *Software Engineering for Game Developers*. Sebastopol, EUA: PTR Prentice-Hall.

Highsmith, J. et al. (2001) *The Agile Manifesto.* Disponível na Internet em: http://agilemanifesto.org/ . Acesso em 22/10/2007.

Lippert, M. e Roock, S. (2001) *Adapting XP to complex application domains*. Proceedings of the ACM 8th European software engineering conference, Viena, Áustria.

Perucia, A. S. et al. (2005) *Desenvolvimento de Jogos Eletrônicos – Teoria e Prática*. São Paulo, Brasil: Ed. Novatec.

Pilato, C. M.; Collins-Sussman B.; Fitzpatrick, B. W. (2004) *Version Control with Subversion*. New York, EUA: O'Reilly Media.

Probasco, L. *The Ten Essentials of RUP*. Canada: Rational Software, 2001.

Ramos, M. A. E Penteado, R. A. D. (2007) Princípios Ágeis Aplicados à Reengenharia de Software. Proceedingds do WDRA 2007 - Workshop em Desenvolvimento Rápido de Aplicações. Porto de Galinhas, Brasil.

Rucker, R. (2002) *Software Engineering and Computer Games*. New Jersey, EUA: Addison-Wesley.

Schofield, B. (2007) Embracing Fun: Why Extreme Programming is Great for Game Development. *Gamasutra, March 1st*. Disponível na Internet em: http://www.gamasutra.com/features/20070301/schofield_02.shtml . Acesso em 22/10/2007.

Silva, L.; Bezerra, L. N. M. ; Silveira, I. F. ; Araujo Jr, C. F. (2007) . A aplicação da Metodologia Institucional Aprender na Prática em disciplinas de projeto do curso superior de Tecnologia em Jogos Digitais. In: Jarmendia, A. M; Silveira, I. F.; Farias, L. A; Sparano, M.; Di Iório, P. L.; Domingues, S. F. S.. (Org.). Aprender na Prática - Experiências de Ensino e Aprendizagem. 1 ed. São Paulo: Edições Inteligentes, v. 1, p. 205-213.

Sutherland, J. (2004) Agile Development: Lessons Learned From the First Scrum. *Cutter Agile Project Management Advisory Service – Executive Update*, v. 5, n. 20.

UNICSUL (2004) Aprender na Prática. São Paulo: Unicsul.