Centro Universitário FEI

Avai**Lab**le: Informativo quanto as vagas disponíveis nos laboratórios CGI

Leon Ferreira Bellini
22218002-8

Guilherme Ormond Sampaio
22218007-7

Sumário

1 Introdução

Diante de uma época considerada "normal" quando se refere as atividades do **Centro Universitário FEI**, muitos dos alunos os quais dependem dos vários laboratórios disponíveis para livre uso, certamente notaram que estes se encontravam lotados, principalmente durante os períodos de provas **P1** (para os cursos que ainda as possuem) e **P2**.

Para resolver tal problema, imaginando um indivíduo vinculado à empresa, o qual possui o trabalho de resolver problemas genéricos, este pode pensar nas seguintes possibilidades:

1. Adquirir novas unidades de computadores, mesas e cadeiras (cabos e instalação inclusos no cenário)
2. Construir um novo laboratório (cenário 1 implícito, mais caro)
3. Desenvolver um **sistema** para, ao menos, refrear a superlotação dos laboratórios a partir da realização de uma gestão propriamente dita, levando em conta o comportamento dos **usuários finais** (alunos).
4. Limitar a entrada de usuários, impondo condições severas em relação ao uso das máquinas e lugares disponíveis (não mais que **n** pessoas presentes ou uma máquina a cada duas pessoas são possíveis restrições)

Deve-se lembrar, entretanto, que existem finitas soluções possíveis as quais englobam tanto curto, quanto longo prazo. Porém, o grupo, defronte ao potencial da opção quatro (4), decidiu implementar um sistema de software capaz de realizar tal ação.

2 Discussão

Alinhado aos princípios da empresa como entrega de conhecimento de forma ágil, digital e dinâmica está presente um dos métodos de desenvolvimento de *software* ágil, o chamado **XP** ou **Extreme Programming** o qual se baseia num ciclo simples onde relatos (ou histórias) dos usuários, têm extrema relevância, estas entrando em foco para "moldar"o produto final (1). Será mesclado a este processo os métodos já utilizados pelo grupo em experiencias anteriores, estes serão explicitados em seções posteriores.

2.1 A escolha da Programação Extrema

Por ser um método ágil, contendo "apenas" quatro passos para a implementação propriamente dita de um componente e, tendo como parte fundamental da fase de **codificação** a programação em duplas, o **XP** foi escolhido por cobrir a o fato de que a equipe não conta com mais do que, exatamente, dois integrantes. Qualquer outro método não foi enxergado como apropriado com o estilo de trabalho da equipe. Além disso, a proximidade com o usuário ao incluir suas histórias como elemento essencial para a produção dos **cartões CRC** gera a oportunidade de desenvolver o melhor produto para tais indivíduos. Pode-se realizar a comparação com um acionista, por exemplo. Um usuário o qual tenha exigências quanto a um futuro *software* cria sua história (investe), esperando que suas experiências tenham um retorno direto (o *software*) e neste caso, devido à agilidade do **XP**, a entrega é rápida.

2.2 O *workflow* do grupo

Como dito anteriormente, a equipe "adaptou" o método **XP** para melhor trabalhar durante os quatro meses de desenvolvimento. A forma pela qual ocorreu estas mudanças serão informadas nas próximas seções. De forma de se conter no ciclo de desenvolvimento do **XP**, a equipe realiza encontros a cada três dias na plataforma **VOIP** *Discord* (3) para cumprir com cada responsabilidade gerada pelas primeiras **etapas**.

2.2.1 Planejamento

Pode-se traçar uma relação com a fase de **coleta de requisitos**, porém esta de forma "mini", cada usuário entrevistado (estes sendo, em sua totalidade, estudantes da FEI), relata suas experiências quando interagindo com o universo em que o *software* terá influência futuramente. É então, realizada para eles, a pergunta, "O quão relevante é para você que esta característica seja implementada?". Uma resposta, por exemplo, sendo "De extrema relevância", possivelmente garantiria a uma dessas "features" prioridade também extrema. Similarmente, prioridades de pesos diferentes são dadas a relatos de mesmo peso. Logo, esta fase determina o que será trabalhado pelos próximos dias.

Devido ao tempo limitado que a equipe possui, ainda ocorre um processo de filtragem durante as reuniões, nas quais a dupla escolhe dentre as consideradas "mais relevantes"as que são essenciais para, pelo menos, apresentação do projeto.

Além do mais, não são requisitadas dos usuários divisões de histórias em "sub-histórias" e o peso dado para cada história determinada nos cartões é em **dias**.

2.2.2 Projeto

Essenciais para a determinação do que será programado pela equipe na próxima etapa, os cartões **CRC** (Classe-Responsabilidade-Colaborador) são concebidos ao se basear nos componentes definidos à partir das histórias dos usuários da fase anterior. A cada reunião de projeto, é feito um cartão **CRC** em um arquivo **org**, linguagem de *markdown* a qual facilita a formatação de documentos destinados à gestão de tempo por *scheduling*, listas *TODO* e planilhas. O arquivo, então, fica à disposição para futuras consultas. O **org-mode** é parte do editor de texto GNU/Emacs (2), o qual é utilizado pela equipe em todas as fases. A partir do momento que este cartão se apresenta terminado, é adicionado um novo componente no **diagrama de componentes** o qual será apresentado em relatório posterior, bem como uma nova classe relacionada ao novo componente. É correto afirmar que esta fase define como será implementado o componente em foco de forma técnica, uma vez que, acompanhado ao cartão CRC, está a elaboração da classe em si no sistema orientado à objetos que o projeto incluí.

A dupla não cria protótipos.

2.2.3 Codificação

Uma vez que a dupla de programadores também forma a equipe por completo, todo o processo de criação de testes unitários (o conjunto de *stubs* e *drivers*) e o ato de codificação em si ocorre "de uma vez", a implementação, na maioria das vezes, é concluída em uma e, raramente, duas reuniões de codificação. O arquivo **org** de cronograma é, então, atualizado ao fim dos testes.

2.2.4 Testes

Possivelmente a fase mais importante para o desenvolvimento de um sistema robusto, pode-se dizer que, para o grupo, a fase de testes é **constante**, dado que a fase de codificação já incluí o teste unitário como essencial. Nesta fase, a equipe também aplica o que foi determinado no documento de **Especificação de teste**. A fase de testes assegura que o grupo possa "zerar" o ciclo e começar uma nova fase de planejamento, já que, isoladamente, o componente produzido não apresenta mais erros ou *bugs*. O grupo escolheu por apenas realizar o teste *top-down* ao final da programação de todos os componentes de um *package*, este definido no **diagrama de componentes**.

3 Bibliografia

1. PRESSMAN, Roger S. Engenharia de software: Uma abordagem profissional. 7. ed. Porto Alegre: AMGH Editora, 2011.

2. GNU Emacs. Disponível em:< `https://www.gnu.org/software/emacs/` >. Acesso em 20 outubro 2020.

3. DISCORD. Disponível em:< `https://discord.com/` >. Acesso em 20 outubro 2020.