UNIVERSIDADE FEDERAL DO CEARÁ

CURSO DE CIÊNCIA DA COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARI DO AMARAL TORRES FILHO

ANÁLISE DA UTILIZAÇÃO DE MÉTODOS ÁGEIS NO DESENVOLVIMENTO DE AMBIENSTES VIRTUAIS DE APRENDIZAGEM: UM ESTUDO DE CASO DO SOLAR 2.0

Fortaleza – CE

2014

ARI DO AMARAL TORRES FILHO

ANÁLISE DA UTILIZAÇÃO DE MÉTODOS ÁGEIS NO DESENVOLVIMENTO DE AMBIENSTES VIRTUAIS DE APRENDIZAGEM: UM ESTUDO DE CASO DO SOLAR 2.0

Dissertação apresentada ao Curso de Mestrado em Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Mestre em Computação.

Orientador: Prof. Dr. Mauro C. Pequeno.

Fortaleza-CE

2014

ARI DO AMARAL TORRES FILHO

ANÁLISE DA UTILIZAÇÃO DE MÉTODOS ÁGEIS NO DESENVOLVIMENTO DE AMBIENSTES VIRTUAIS DE APRENDIZAGEM: UM ESTUDO DE CASO DO SOLAR 2.0

Esta dissertação foi julgada adequada à obtenção do título de Mestre em Computação e aprovada em sua forma final pelo Curso de Mestrado em Computação da Universidade Federal do Ceará.

Fortaleza, 30 de janeiro de 2014.

_____

Professor e orientador Mauro C. Pequeno, Dr.
Universidade Federal do Ceará

_____

Professor. Miguel Franklin de Castro, Dr.
Universidade Federal do Ceará.

_____

Professor. Rodrigo Penteado Ribeiro de Toledo, Dr.
Universidade Federal do Rio de Janeiro.

Dedico a Deus esta obra que tanto tempo e labor me custou, mas cujos resultados sei que não seriam possíveis sem ele.

AGRADECIMENTOS

Agradeço a todos que me ajudaram para conclusão desse trabalho, em especial a minha esposa Ingrid Silveira Sousa Amaral e a minha filha Isabele , porque sem elas nada sou. Agradecer imensamente a meus pais, Ari do Amaral e Julieta Torres pelo apoio, paciência, dedicação e educação que me foi me dada pois sem eles não teria chegado até aqui.

Um agradecimento mais que especial ao meu orientador Mauro Cavalcante Pequeno por ter acreditado no meu trabalho e por todo esforço para que consigamos chegar ao nosso objetivo final. Não poderia deixar de agradecer ao meu grande amigo Wellington Sarmento, que me ajudou, colaborou e me orientou junto com o professor Mauro para a conclusão desse trabalho.

Agradecer também a todo time de desenvolvimento da UFC Virtual que participaram ativamente de todo o processo de construção desse trabalho, pois sem eles o SOLAR 2.0 não estaria no ar. Um agradecimento especial ao Professor Herbert Lima que também acreditou no meu trabalho e no qual tive o prazer de trabalhar em alguns projetos e acabou se tornando também um amigo.

Um Agradecimento especial a todos da comunidade de métodos ágeis que auxiliaram de forma direta e indireta esse trabalho.

E por último porém não menos importante , agradecer ao Orley, pela paciência e presteza de sempre poder resolver os problemas burocráticos no departamento de computação.

“Seja a mudança que você quer ver no mundo”.
(Mahatma Ghandi)

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original”

(Albert Einstein)

RESUMO

Nas duas últimas décadas houve um considerável avanço no uso de Tecnologias da Informação e Comunicação Digitais no processo de ensino e aprendizagem, tanto na modalidade de Educação presencial quanto a distância. Neste contexto, o uso de Ambientes Virtuais de Aprendizagem - softwares criados para o suporte a cursos através da Internet - vêm crescendo e seu acesso se tornando cada vez mais diversificado, indo de computadores desktop a celulares smartphones. Desta forma, a complexidade de criação destes ambientes se torna cada vez maior e exige técnicas de Engenharia de Software e Gestão de Projetos cada vez melhores para garantir a qualidade do produto gerado e a satisfação do cliente. Tendo em vista tal cenário, o presente trabalho propõe o uso de Metodologias Ágeis tanto no desenvolvimento quanto na gestão de projetos de ambientes virtuais como uma solução mais interessante que a tradicional forma de criação de softwares baseada somente na qualidade do produto, esquecendo, muitas vezes, a satisfação do cliente e a motivação dos desenvolvedores. Este trabalho relata a adoção de metodologias ágeis no desenvolvimento do AVA SOLAR 2.0 , mostrando que é possível ter sucesso tanto no nível técnico quanto no organizacional quanto no pessoal com a adoção de práticas ágeis no desenvolvimento de AVAs que podem ser comprovados com os experimentos feitos através de pesquisas de satisfação com os usuários e desenvolvedores do projeto.

Palavras-chave: Ambientes Virtuais de Aprendizagem; Metodologias Ágeis; Engenharia de Software.

ABSTRACT

In the last two decades there has been considerable progress in the use of Information Technologies and Digital Communication in the process of teaching and learning, both in the form of classroom education as distance. In this context, the use of Virtual Learning Environments - software designed to support the courses via the Internet - is growing and access becoming increasingly diverse, ranging from desktop computers to mobile smartphones. Thus, the complexity of creating these environments becomes increasingly technical and requires Software Engineering and Project Management always better to ensure product quality and customer satisfaction generated. Given such a scenario, this paper proposes the use of Agile methodologies in the development and project management of virtual environments as a more interesting way than traditional software delivery based only on product quality, forgetting often , customer satisfaction and motivation of developers. This paper reports the adoption of agile methodologies in the development of the SOLAR AVA 2.0, showing that it is possible to succeed in both the technical level and at the organizational and personally with the adoption of agile practices in the development of AVAs that can be proven through experiments done with the satisfaction surveys with users and project developers.

Keywords: Virtual Learning Environment; Agile Methods; Software Engineer.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Educação a Distância (EaD), modalidade de ensino e aprendizagem na qual os partícipes estão geograficamente dispersos e há uma intensa utilização de meios tecnológicos para estabelecer a comunicação e a interação entre eles, está crescendo consideravelmente nos últimos anos, tanto nas instituições de ensino, quanto nas empresas e em seus setores de treinamento. Segundo Dias (2003), o conjunto das inovações tecnológicas com ênfase na Internet veio favorecer esta modalidade de ensino possibilitando a interação dos participantes envolvidos no processo. Nesse cenário surgiram os Ambientes Virtuais de Aprendizagem (AVA).

Os Ambientes Virtuais de Aprendizagem (AVA) são *softwares* criados para o ensino e aprendizagem e tem como características a integração de diversas tecnologias com o intuito ser uma alternativa para o ambiente educacional tradicional e agregar recursos e formas eficientes de aprendizagem ao processo tradicional de ensino dando uma abordagem baseada na colaboração (Dias, 2003), atuando assim de forma complementar ao processo de ensino e aprendizagem tradicional. Eles são *softwares* educativos nos quais os sujeitos podem interagir e construir conhecimento.

Os Ambientes Virtuais de Aprendizagem vem sendo uma opção tecnológica muito utilizada no âmbito acadêmico e corporativo para atender a demanda educacional. Diante disso existe a importância de um entendimento mais crítico sobre o conceito que orienta o desenvolvimento e uso desses ambientes (Pereira *et al*., 2007).

O desenvolvimento de AVA vem se tornando cada vez mais necessário para uma boa educação. Seja ela presencial ou a distância, haja vista que estes *softwares* dão suporte ao processo de interação e comunicação necessários em todas as modalidades de educação. Os processos de criação de AVA têm seguido o padrão de metodologia para desenvolvimento de *software* educativo que é baseado no modelo cascata da Engenharia de *Software* (Oliveira *et al*., 2001).

Segundo Pequeno *et al*. (2004 apud Sarmento, 2007), os Ambientes Virtuais de Aprendizagem tem a seguinte divisão , quanto ao modelo de interação, em:

- Ambientes de Apoio a Cursos – Direcionados ao aluno ou ao professor e voltados à publicação de textos e atividades voltadas a cursos presenciais e a distância;
- Ambientes Colaborativos cuja principal característica seriam os trabalhos em grupo e a interação entre os participantes;
- Ambientes Híbridos, que mesclariam as características dos Ambientes Colaborativos com os Ambientes de apoio a Cursos.

Quanto ao acesso à ferramenta, os ambientes podem ser divididos em três categorias: Comerciais de Código Fechado, Gratuitos de Código Fechado e Gratuitos de Código Aberto. Como exemplo dos ambientes Comerciais de Código Fechado, temos o Blackboard, desenvolvido pela empresa Blackboard ; para a segunda classificação temos o AulaNet (2005), desenvolvido pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Já para a terceira classificação, temos como exemplo o TelEduc desenvolvido pela Universidade de Campinas (Unicamp). Além destes, têm-se os ambientes institucionais que não são vendidos nem estão disponíveis para uso amplo, pois foram criados para uso interno às instituições que os desenvolveram. Como exemplo destes ambientes, temos o e-Proinfo que foi desenvolvido pelo Ministério da Educação e o *Web*Aula , que provê várias soluções para treinamentos e cursos que são oferecidas para treinamentos corporativos de empresas (Sarmento, 2007).

Os ambientes virtuais são *softwares* e como tais precisaram passar por um processo de desenvolvimento. Esse processo de desenvolvimento na maioria dos ambientes se evidência seguindo o modelo tradicional de desenvolvimento de *software* o que pode ser dada pela documentação do TelEduc onde se mostra algumas fases de desenvolvimento e exemplos de documentos usados no seu processo de criação, como diagramas de casos de uso e diagramas de classes.

Como *softwares* que são, estes ambientes de aprendizagem virtuais têm passado pelos mesmos problemas da engenharia de *software* tradicional. Os requisitos mudam rapidamente e o sistema torna-se legado em pouco tempo (Cockburn, 2006).

Um dos principais motivos desse problema vem sendo a maneira de se desenvolver programas fundamentados no modelo cascata, onde cada etapa ocorre de maneira sequencial com dependências entre si (Pressman, 2006). Esse modelo é convidativo a falhas e é perigoso, portanto é necessário sua superação (Brooks Junior, 2010).

A empresa norte-americana *Standish Group*, publica desde 1994 um relatório chamado de *CHAOS Report*, que faz um levantamento entre milhares de projetos em Tecnologia da Informação, que é um dos principais meios de quantificar os sucessos, e fracassos destes projetos. Estes dados são importantes para o processo de Engenharia de *Software* porque permitem avaliar o nível de qualidade e sucesso do processo de desenvolvimento de sistemas (Teles, 2005).

O CHAOS Report de 2009 apresenta dados que mostram que nossos projetos de *software* apresentam dados preocupantes (Standish, 2009). Pode-se inferir do relatório do Standish Group que 32% dos projetos analisados tiveram sucesso, ou seja, foram realizados no prazo, dentro do orçamento e com escopo completo. Já 44% foi considerado "Desafiado" que significa que atrasaram, ultrapassaram o orçamento, ou reduziram o escopo. Por fim, 24% fracassaram, tendo sido cancelados ou nunca usados. Veja o gráfico baseado do relatório do *Standish Group* sobre a situação dos projetos de *software* na Figura 1 - Situação dos Projetos de Software:

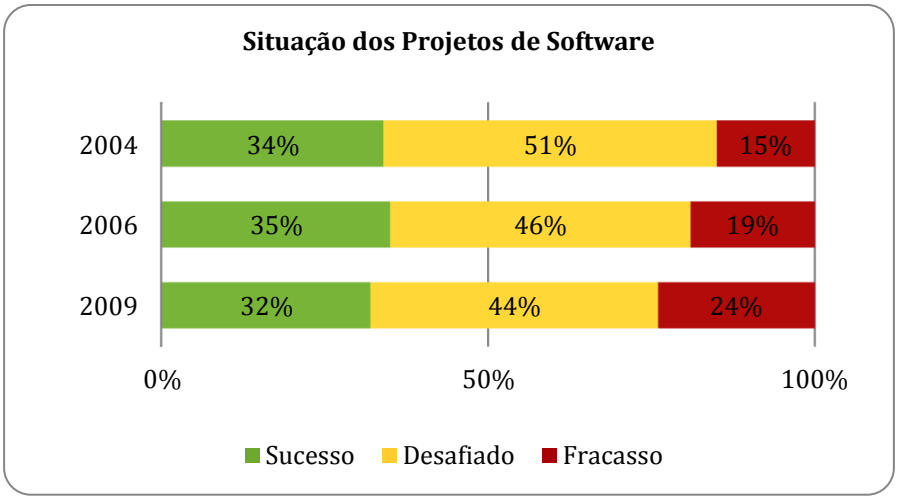


Figura 1 - Situação dos Projetos de Software

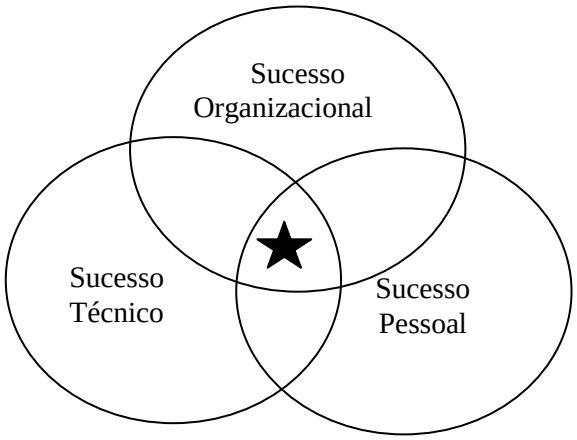
Segundo o relatório supracitado, está-se pior do que se estava em 2004 e apesar da Engenharia de *Software* ter evoluído e incorporado novas técnicas parece que ainda não se aprendeu a fazer *software* corretamente já que os resultados mostram que a maior parte dos projetos de *software* pesquisados não obtiveram sucesso.

A esse relatório cabem alguns questionamentos nas definições de Sucesso, Desafiado e Fracasso que abordam basicamente os critérios tradicionais onde o sucesso está caracterizado por fazer entregas do produto no tempo, orçamento e entregando as especificações determinadas, e sem levar em consideração o valor de negócio do produto. Nesse caso, um projeto pode ter Sucesso mesmo que não gere nem um real de lucro e pode ser Desafiado mesmo que gere milhões em lucros para empresa (Shore e Warden, 2008).

Segundo Shore e Warden (2008), para podermos ter um melhor conceito sobre sucesso dos projetos teríamos que levar em consideração três significados diferentes para este:

- O Sucesso Pessoal onde é olhada a satisfação das pessoas envolvidas no projeto, fazerem aquilo que gostam.
- O Sucesso Técnico onde os projetos são feitos dentro de especificações, prazos e técnicas estabelecidas.
- O Sucesso Organizacional onde o projeto vai gerar Retorno sobre os investimentos (ROI) para empresa, agregar valor a empresa.

Os três tipos de sucesso são importantes, como pode ser visto Figura 2 - Definição de Sucesso(Shore e Warden, 2008). Nessa definição, o Sucesso – representado por uma estrela – é fruto da interseção do sucesso Organizacional, Técnico e Pessoal. Caso não houvesse o Sucesso Pessoal teríamos problemas em causar motivação nos funcionários, sem o Sucesso Técnico seu código fonte pode um dia cair sobre seu próprio peso e ficar de difícil manutenção e entendimento. Sem o Sucesso Organizacional sua equipe pode se sentir deslocada na empresa e a sensação de fracasso do projeto existirá para o cliente mesmo tendo



sido cumpridas todas as especificações técnicas (Shore e Warden , 2008).

Figura 2 - Definição de Sucesso(Shore e Warden, 2008).

Porém dentro do Modelo de Shore e Warden ainda não contempla satisfatoriamente o conceito de sucesso. O Sucesso ainda pode ser complementado colocando o conceito de ser feliz entregando valor(*Use Happiness)*, onde mais que o ROI entregue estamos abrangendo uma nova área derivando a esfera pessoal e organizacional, onde essa intercessão entre essas duas esferas esteja alinhada ao objetivo de gerar valor deixando o cliente e o usuário felizes. A proposta é da esfera pessoal agora crescer tendo uma abrangência ainda maior que as outras duas, fazendo com que o sucesso organizacional e técnico estejam contidos dentro do sucesso pessoal.

Existe ainda uma estatística ainda mais alarmante, segundo Standish (2002), onde descreve que de todas as funcionalidades que são produzidas em nossos softwares, temos somente 20% de valor entregue ao nosso cliente e o restante é desperdício, 64% do que é produzido é raramente ou nunca usado e 16% é usado somente às vezes. Veja o gráfico feito a partir do relatório do *Standish Group* de 2002 sobre o uso de funcionalidades na Figura 3:

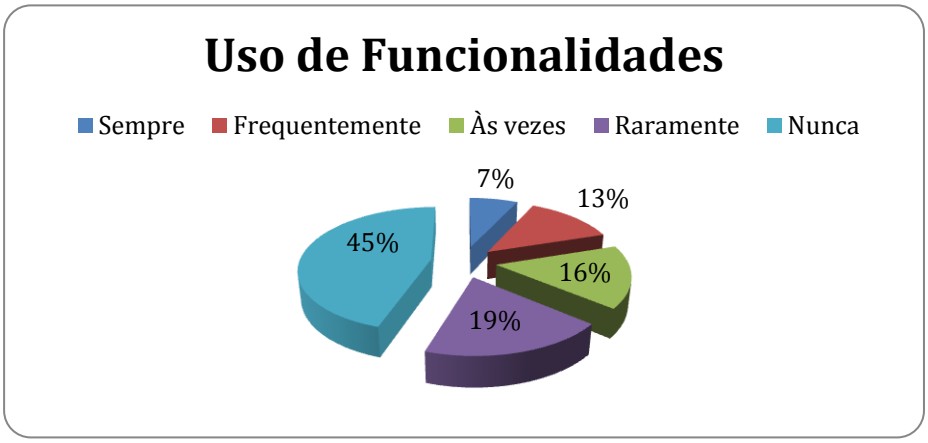


Figura 3 - Uso das Funcionalidades do Software.

Então o grande desafio no desenvolvimento de *software* seria descobrir os 20% que agregam mais valor de negócio ao produto e desenvolvê-los antes. (Teles, 2007)

A técnica de desenvolver eliminando o desperdício e produzindo somente o necessário de acordo com a priorização é chamada de Desenvolvimento Lean (Poppendieck, 2003). Esta metodologia surgiu vindo do modelo Toyota de produção e possui sete princípios: “Elimine os Desperdícios”, “Inclua a qualidade no processo”, “Crie Conhecimento”, “Adie Comprometimentos”, “Entregue Rápido”, “Respeite as Pessoas” e “

Otimize o todo". O Modelo Lean de Desenvolvimento de *software* compõe uma das chamadas metodologias ágeis (Poppendieck, 2003).

Na década de 90 surgiram alguns métodos que usavam uma abordagem leve para o desenvolvimento de *software*, onde os processos a serem adotados estavam baseados na adaptação a mudanças, que cada vez era mais frequente nos projetos de *software*. Esses novos métodos surgiram como uma resposta ao modelo tradicional de desenvolvimento de *software* baseado no processo cascata (Beck, k. *et al*, 2001). Com o objetivo de definir um manifesto para encorajar a melhor maneira de desenvolver *softwar*es, em fevereiro de 2001, um grupo de 17 metodologistas que vinham tendo sucesso em seus projetos se reuniram para criar uma metodologia baseada nos seus métodos de trabalho e perceberam que além de eficazes seus métodos eram parecidos. Essa nova metodologia seria formada por características que contribuíram para o sucesso de seus projetos e da união e intercessão de suas formas de trabalhar surgiria uma nova metodologia de desenvolvimento de *software* para que pudesse substituir o modelo tradicional de desenvolvimento de *software* que era baseado em formalismos e excesso de documentação. As conclusões desta reunião foram que desenvolver *software* era algo complexo demais para ser definido por um único processo, então dessa reunião conseguiu-se criar o chamado, Manifesto Ágil de Desenvolvimento de *Software* que Segundo Beck (2001 apud Keith, 2010), foi baseado nos quatros valores descritos a seguir :

- Indivíduos e interações são mais Importantes do que Processos e Ferramentas.
- *Software* funcionando mais do que documentação abrangente.
- Colaboração com o cliente mais do que negociação de contrato.
- Responder a mudança mais do que seguir um plano.

Através dos valores descritos acima eles aprenderam a fazer *software* melhor e ajudaram as pessoas também a produzir *software* com mais qualidade. Mesmo reconhecendo que existem valores nos itens acima do lado esquerdo eles atribuem um valor maior aos itens do lado direito. Além dos quatro valores básicos, o Manifesto Ágil apresenta 12 princípios que são fundamentais para o seu sucesso. Segundo Ambler (2004) são eles:

- A maior prioridade é satisfazer o cliente mediante entrega de *softwar*es de valor em tempo hábil e de forma contínua.

- Receber bem as mudanças de requisitos ainda que em uma fase mais avançada no desenvolvimento. Os processos ágeis direcionam as mudanças para produzir vantagem competitiva para os clientes.
- Entregar *software* funcional com frequência, de algumas semanas a alguns meses, dando preferência para menor escala de tempo..
- Os grupos de negócios e desenvolvimento devem trabalhar juntas dia-a-dia durante todo o projeto.
- Construa projetos através de indivíduos motivados. Dê-lhes o ambiente e o apoio que eles precisam e confie neles para a realização do trabalho.
- O método mais de transferir informação para e de um time de desenvolvimento é através de conversação face a face.
- Ter o *software* que funcione é a principal medida de progresso.
- Processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter uma, frequência de trabalho constante indefinidamente.
- Ter atenção continuamente em relação à excelência técnica e a um bom projeto, levando-se em consideração o aumento da agilidade.
- Simplicidade é essencial. Entenda-se, neste caso, simplicidade como a arte de elevar ao máximo a quantidade de trabalho que não se realiza-.
- As melhores arquiteturas**,** requisitos e designs emergem de times auto-organizáveis.
- Em intervalos regulares, o time deve refletir de como se tornar mais efetivo e então se ajustar e adaptar seu comportamento.

Neste cenário, uma solução que vem tendo resultados satisfatórios são os Métodos Ágeis, que são baseadas nos valores e princípios do manifesto, aplicadas ao desenvolvimento e gerenciamento de *software*. Esses Métodos surgem como uma solução para melhorar o sucesso dos projetos de *software* e produzir mais valor de negócio (Martins, 2007) evitando o desperdício de funcionalidades utilizando do desenvolvimento iterativo e incremental.

As metodologias ágeis são destinadas ao desenvolvimento de programas complexos e que os requisitos tendem a mudar rapidamente, portanto adequa-se para o desenvolvimento de AVA, podendo contribuir para um ambiente que represente mais fielmente a realidade do seu público e gere resultados mais satisfatórios quanto ao seu uso.

A Proposta é que se tenha um Ambiente Virtual de Aprendizagem em evolução constante, com fácil integração com novas funcionalidades e que seja permitido mudanças através de *feedback* rápido de seus usuários que serão validadas através de práticas ágeis de desenvolvimento de *software*.

1.1 Justificativa

Muitas organizações de desenvolvimento de *software* estão se esforçando para se tornarem mais eficientes. Bem sucedidas equipes ágeis estão produzindo *software* de alta qualidade que melhor atenda as necessidades dos utilizadores mais rapidamente e a um custo menor do que as equipes tradicionais. Equipes Ágeis estão vendo ganhos significativos de produtividade com diminuição dos custos correspondentes e são capazes de entregar produtos ao mercado mais rápido e com um maior grau de satisfação do cliente. Elas estão experimentando uma maior visibilidade no processo de desenvolvimento, levando a uma maior previsibilidade (Cohn, 2010).

Algumas vantagens do desenvolvimento ágil em relação ao modelo tradicional de desenvolvimento de *software* vêm se comprovando através de pesquisas realizadas, que mostram um sucesso maior nos projetos de *software* que adotam essa nova filosofia de desenvolvimento. A pesquisa feita pela VersionOne (Cohn, 2010) olhou alguns fatores adicionais, levando em consideração à satisfação dos interessados. A Tabela 1 - Comparativo entre Métodos Ágeis e Métodos Tradicionais (Conh, 2010), mostra o alto percentual dos entrevistados que relataram que o desenvolvimento ágil leva a uma integração melhor da tecnologia da informação com objetivos do negócio, redução de riscos nos projetos, uma melhor capacidade para gerenciar mudanças, e melhor visibilidade do projeto.

Tabela 1 - Comparativo entre Métodos Ágeis e Métodos Tradicionais (Conh, 2010)

	Melhorou	**Melhorou significativamente**

	Melhorou	Melhorou significativamente
Maior capacidade de gerenciar mudanças de prioridades	41%	51%
A melhoria da visibilidade do projeto	42%	41%
Melhorou o alinhamento entre TI e objetivos de negócio	39%	27%
Redução do risco do projeto	48%	17%

Outra pesquisa que foi feita pela Ambysoft com 642 entrevistados e publicada no Dobb's Journal sobre a eficácia das metodologias ágeis em relação as metodologias tradicionais), revelou os benefícios de quem está adotando Métodos Ágeis em seus projetos de *software* (Ambler, 2008). Podemos concluir que a pesquisa aponta dados que mostram que no custo de desenvolvimento 37% afirmaram que o desenvolvimento ágil teve custo menor e 40% que se permaneceu com os mesmos custos, tendo agora uma melhor satisfação das partes interessadas no negócio em torno de 78%, e qualidade melhorou segundo afirmaram 77% das pessoas gerando segundo eles uma maior produtividade nos projetos ágeis segundo afirmam 82% das pessoas. Na Figura 4 - Comparativo dos Métodos Ágeis com os Métodos Tradicionais podemos ver graficamente a comparação de resultados.

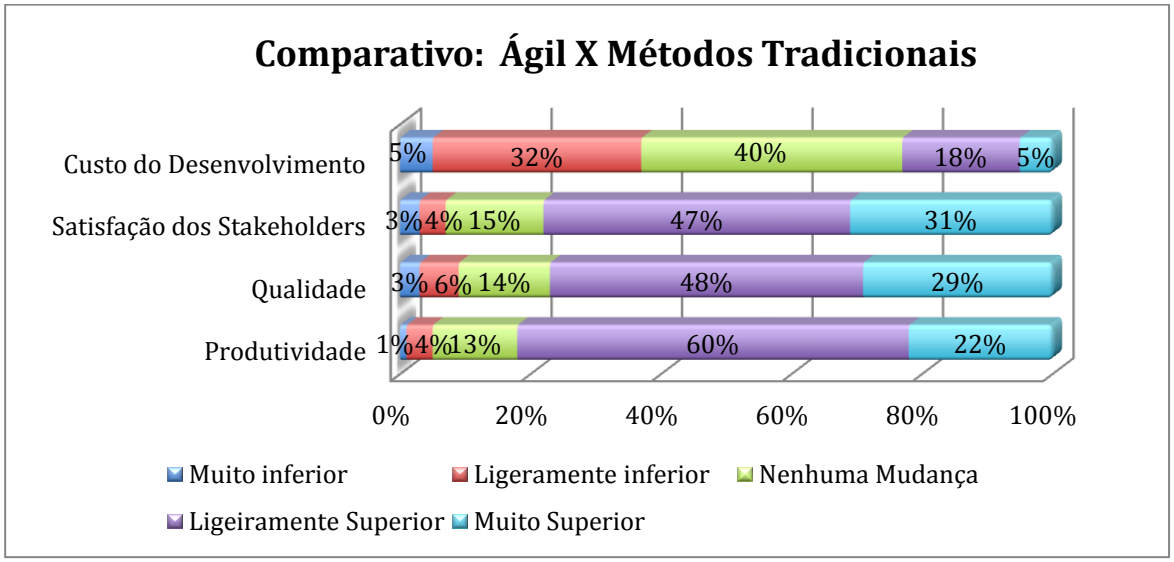


Figura 4 - Comparativo dos Métodos Ágeis com os Métodos Tradicionais

Foi feito um estudo pelo IME–USP em 2012 com 466 pessoas que responderam espontaneamente em listas de discussões uma pesquisa vista na Figura 5 - Principais Benefícios Obtidos com a Adoção de Métodos Ágeis(MELO et al, 2012). Onde se relatam os ganhos obtidos com métodos ágeis. Dentre diversos benefícios que melhoraram ou melhoraram muito, os mais frequentes foram: Habilidade de Gerenciar mudanças de prioridades(76%), Aumento da Produtividade(76%)e da Qualidade(75%), Aumento da Moral do time (75%) e Simplificação do processo de desenvolvimento (75%). Em relação à velocidade dos projetos, 67,1% dos respondentes indicaram que projetos ágeis terminam mais rápido que projetos tradicionais. (MELO et al, 2012).

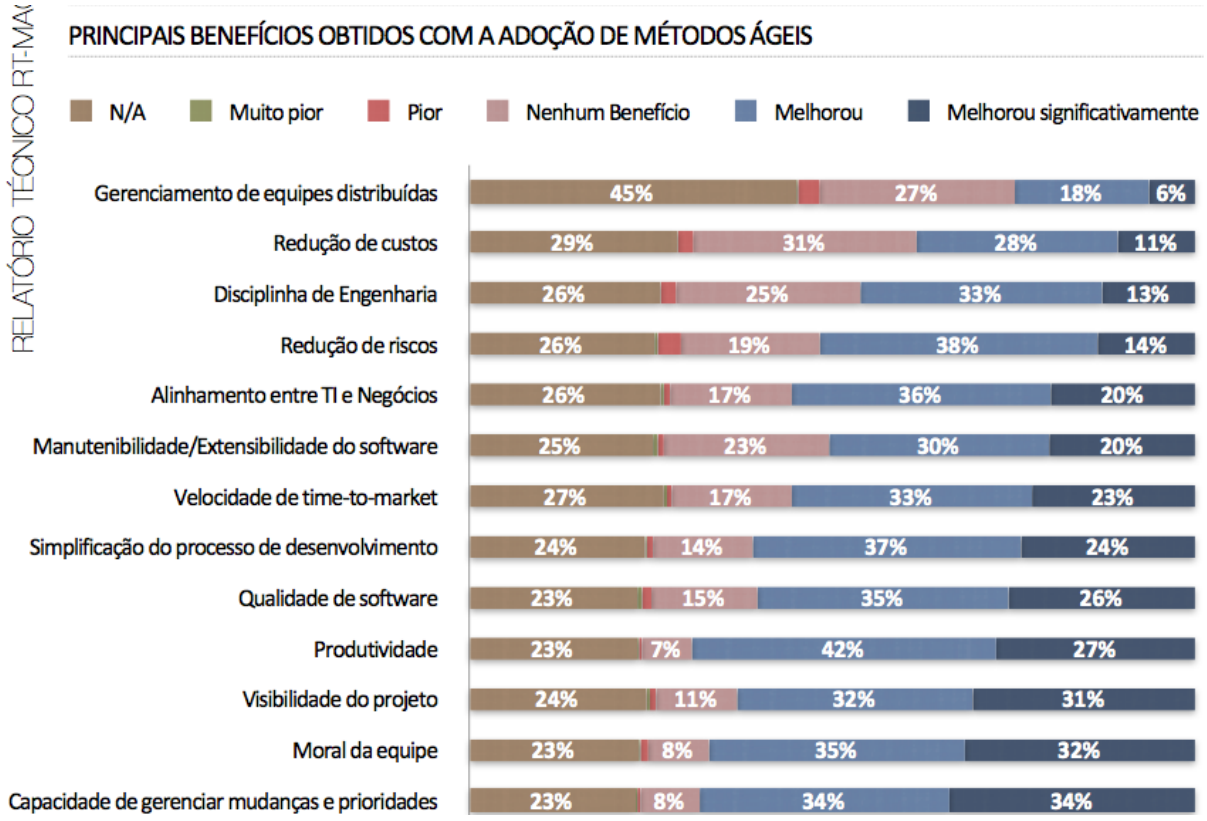


Figura 5 - Principais Benefícios Obtidos com a Adoção de Métodos Ágeis(MELO et al, 2012).

Em virtude dos benefícios apresentados pelo uso de Metodologias Ágeis no desenvolvimento de aplicações complexas é que se propõe o desenvolvimento de Ambientes Virtuais de Aprendizagem utilizando-as, para que se possa tornar o desenvolvimento mais rápido, eficiente e atendendo com mais fidelidade a necessidade dos clientes que a utilizarão.

O Instituto UFC Virtual, da Universidade Federal do Ceará, produziu o AVA chamado SOLAR, na segunda metade da década de 90, que necessita de uma nova versão

para se manter atualizado e satisfazendo as novas formas de promover aprendizagem à distância utilizando tecnologias recentes e conceitos mais atuais de desenvolvimento de *Software*. É neste contexto que se encontra o presente trabalho que propõe um conjunto de práticas e técnicas ágeis para o desenvolvimento de Ambientes Virtuais de Aprendizagem e analisará sua aplicação no processo de desenvolvimento na nova versão do AVA SOLAR.

O SOLAR 2.0 permitirá a criação de um espaço virtual que poderá ser utilizado por cursos presenciais ou semipresenciais – como as graduações da Universidade Aberta do Brasil (UAB) -, servindo como ponto de convergência para a criação do que se está chamando de *Blended Education*, ou seja, a mescla de características de ambas as modalidades de Educação, presenciais e a distância, para formação de um novo modelo educacional que utiliza fortemente as Tecnologias da Informação e Comunicação (TIC).

1.2 Tema

Estudo de um conjunto de práticas e técnicas pautadas nos Métodos Ágeis para Desenvolvimento de *software* no processo de criação de Ambientes Virtuais de Aprendizagem.

1.3 Problematização

- O processo evolutivo dos Ambientes Virtuais de Aprendizagem, bem como seu caráter dinâmico em termos de funcionalidades, fazem destes *softwares* fortes candidatos às chamadas Metodologias Ágeis de Desenvolvimento de *Software*?
- É possível definir um modelo para desenvolvimento de AVA baseado em Metodologias Ágeis?

- Quais práticas e técnicas relacionadas às Metodologias Ágeis seriam importantes para o desenvolvimento de AVA?
- Como os *Métodos Ágeis* podem contribuir para produzir Ambientes Virtuais de Aprendizagem que sejam mais interativos e eficazes no processo de ensino e aprendizagem?
- Como a aplicação de Metodologias Ágeis ajudarão a ter um *software* preparado para mudanças constantes no sistema com pouco impacto no custo, tempo e qualidade do mesmo?
- Como melhorar uma metodologia de desenvolvimento de AVA utilizando práticas ágeis que entregue *software* funcionando, com qualidade e que satisfaça a necessidade dos seus usuários gerando valor de negócio?
- Como preparar a equipe para se tornar comprometida, multidisciplinar e auto-organizável?
- Quais as dificuldades e problemas que serão encontrados ao começar usando Scrum?

1.4 Hipóteses

- O caráter dinâmico das funcionalidades dos AVA, que estão sempre evoluindo conforme a necessidade dos educadores e educandos de acesso a conteúdos didáticos, comunicação e interação, tornam estes ambientes difíceis de serem modelados e documentados pelas técnicas tradicionais de Engenharia de *Software*.
- Práticas e técnicas como o Scrum, Kanban e *Extreme Programing* podem ser utilizadas para melhorar a produtividade e diminuir o tempo de desenvolvimento de Ambientes Virtuais de Aprendizagem.
- Uma modelagem ágil em substituição a modelagem tradicional. Documentando apenas o que for necessário para agregar valor de negócio ao produto desenvolvido.

- Tratando-se de um software complexo com muitas variáveis o modelo de gestão do Management 3.0 (Gestão 3.0) , se adequa para gerenciar o projeto do AVA SOLAR 2.0.
- Seleção de práticas de Engenharia de *Sofware* como integração contínua, testes automatizados incluindo teste de unidade com prática do TDD e BDD como teste de aceitação melhorarão a qualidade do código e o deixará mais legível e de mais fácil manutenção.
- Uma levantamento e priorização constantes de requisitos focados em valor de negócio irão contribuir para um produto com uma maior representatividade da realidade desejada.

1.5 Objetivo Geral

Mostrar que o uso de Metodologias Ágeis no processo de desenvolvimento de Ambientes Virtuais de Aprendizagens pode melhorar tanto a qualidade do *software* como torná-lo um software com uma melhor usabilidade que pode ser validada através dos *feedback* contínuos dados durante o desenvolvimento, além de valorizar uma gestão focada em pessoas, mostrando que pessoas comprometidas e trabalhando felizes são a chave para maximizar os resultados.

Esse trabalho trás como contribuição o uso de métodos ágeis relacionado a um AVA, mostrando através de inpeção e adaptação no processo e nas técnicas seguidas a criação de um padrão de boas práticas e melhorias no processo do *Scrum e Kanban* adaptando-os de forma mais eficiente a se trabalhar no gerenciamento e construção de softwares educacionais. Mostra como trabalhar com ausência de um Product Owner em um projeto,em como otimizar reuniões e de como melhorar o ambiente de trabalho desenvolvendo um estudo valorizando as pessoas .

1.6 Objetivos Específicos

- Analisar o impacto da adoção das Metodologias Ágeis no desenvolvimento do Solar 2.0.
- Sugerir um conjunto de técnicas e práticas que poderiam ser adotadas no processo de desenvolvimento do AVA Solar 2.0.
- Com a aplicação de Métodos Ágeis em um projeto real, analisar e levantar métricas de produtividade e de satisfação dos usuários e do aprendizado do time.
- Mostrar que o desenvolvimento de AVA através de metodologias ágeis facilita o desenvolvimento e maximiza os resultados obtidos.
- Tentar medir o conceito de sucesso abrangendo seus 3 tipos, Organizacional, técnico e pessoal.
- Propor melhorias no processo de algumas metodologias de gerenciamento e desenvolvimento durante do o desenvolvimendo da aplicação.
- Melhorar técnicas de reuniões, tornando-as mais eficientes e produtivas.
- Tratar como aplicar métodos ágeis sem um Product Owner disponível.

1.7 Estrutura do Trabalho

O trabalho dissertativo está dividido em cinco capítulos, dos quais o primeiro é a introdução, os dois seguintes estão relacionados ao referencial teórico e o quarto o estudo de caso feito sobre a versão 2.0 do AVA SOLAR, bem como o levantamento de dados e análise sobre este ambiente, do ponto de vista da satisfação do cliente e dos desenvolvedores. Por fim, são feitas as considerações finais relacionando os dados obtidos na pesquisa com seus objetivos, identificando problemas encontrados, propondo soluções para estes problemas e especificando possíveis pesquisas que estenderiam o trabalho realizado.

2 MÉTODOLOGIAS ÁGEIS

Projetos diferentes precisam de processos ou metodologias diferentes. O caminho é investigar o problema, experimentar, inspecionar e adaptar soluções. O foco em habilidades, comunicação e comunidade permite o projeto ser mais eficaz e mais ágil do que quando focado em processos. (Highsmith, 2012).

As metodologias Ágeis apresentadas nesse capítulo são as utilizadas durante o desenvolvimento do SOLAR 2.0.

2.1 Scrum

O desenvolvimento de software é uma atividade cuja complexidade depende do nível de tecnologia aplicada bem como a quantidade de requisitos que serão utilizadas no projeto, ver Figura 6 - Nível de Complexidade do Software(Schwaber, 2004). Gerenciar projetos complexos tem apresentado um alto grau de dificuldade (Apello, 2011) e o *Scrum* vem para minimizar essa dificuldade, pois ele é para gerenciamento de projetos complexos. *Scrum* trata da complexidade dos projetos de desenvolvimento de software através da implementação dos requisitos de inspeção, adaptação e visibilidade de controle de processo empírico com um conjunto de práticas simples e regras (Schwaber,2004).

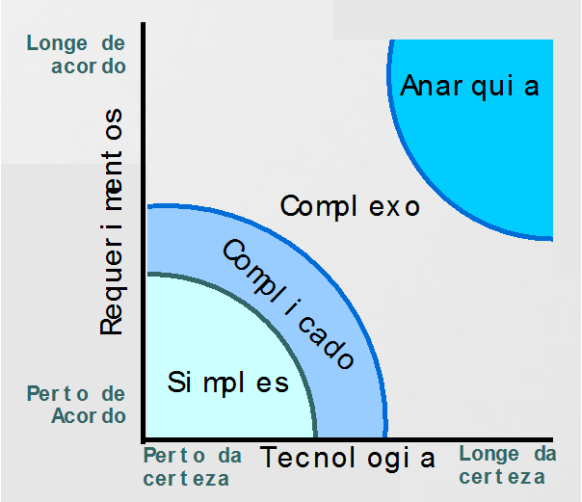


Figura 6 - Nível de Complexidade do Software(Schwaber, 2004)

Historicamente, o termo *Scrum* vem de um artigo publicado por Hirotaka Takeuchi e Ikujiro Nonaka na Harvard Business Review em 1986. Nesse artigo, intitulado "O jogo do Desenvolvimento de Novos Produtos", Takeuchi e Nonaka descreveram uma abordagem holística em que as equipes de projetos são constituídos por pequenas equipes multifuncionais, trabalhando com sucesso em direção a um objetivo comum, que os autores compararam a formação *Scrum* no rugby (Pham, 2011).

Enquanto trabalhava na construção de uma Análise Orientada a Objetos e uma ferramenta de Design, Jeff Sutherland, então vice-presidente de Engenharia da Easel, Inc. percebeu que sua equipe de software precisaria de uma versão melhorada para desenvolver aplicações de forma rápida. O que ele queria era um processo semelhante ao Scrum, onde no final de iterações curtas, o CEO da instituição veria código de trabalho realizado ao invés de gráficos de *Gantt* em papel.

Durante mais ou menos no mesmo período, Ken Schwaber foi ativamente à procura de como ele pode ajudar sua empresa, Métodos de Desenvolvimento Avançadas, Inc. (ADM), afim de melhorar o seu processo de software, aumentando a produtividade de suas equipes.

Após continuar a analisar como outros bem sucedidos fornecedores de software independentes construíam software, Ken veio a perceber que todos os seus processos de desenvolvimento foram semelhantes na medida em que todos usaram processos empíricos, o que exige constante inspeção e adaptação.

A pedido do Object Management Group (OMG), em 1995, Jeff e Ken trabalharam juntos para resumir o que tinham aprendido ao longo dos anos, eles criaram uma nova metodologia, que deram o nome de Scrum, e descrito no artigo de Schwaber, "“Scrum and the Perfect Storm,” em www.controlchaos.com/my-articles. (Pham, 2011).

Desde então, Schwaber e Sutherland, juntos e separados, produziram várias publicações sobre Scrum, incluindo Desenvolvimento Ágil de Software com Scrum (Schwaber e Beedle, 2001), Gerenciamento Ágil de Projetos com Scrum (Schwaber 2004) e O Guia do Scrum (Schwaber e Sutherland 2007, 2011).

Embora Scrum é mais comumente usado para desenvolver produtos de software, os valores e princípios fundamentais do Scrum podem e estão sendo utilizados para desenvolver diferentes tipos de produtos ou organizar o fluxo de vários tipos de trabalho (Sabbagh, 2013).

Scrum é um *framework* para desenvolvimento e manutenção de produtos complexos. Ele tem como características de ser leve de fácil entendimento e de difícil domínio . O principal objetivo do *Scrum* é ajudar a entregar produtos com alto valor agregado o mais constantemente possível(Schwaber, 2011). Ele não é uma ferramenta ou técnica para construir produtos e nem uma metodologia ou conjunto de práticas de engenharia, e sim um *framework* dentre o qual se pode empregar processos e técnicas variadas de acordo com o problema. *Scrum* é uma estrutura leve que foi projetada para o gerenciamento de software e desenvolvimento de produtos. (Lacey, 2012).

O *framework Scrum* consiste nas equipes Scrum, papéis, eventos, artefatos e regras associadas. Cada componente desses serve para um propósito específico e é fundamental para o sucesso do *Scrum.*

O *Scrum* é baseado no empirismo para controlar processos. O Empirismo afirma que o conhecimento vem da capacidade de tomarmos decisões em cima daquilo que conhecemos. O *Scrum* utiliza o modelo iterativo e incremental para ter uma melhor previsibilidade e um melhor controle de riscos no projeto (Schwaber, 2011).

Em seu núcleo *Scrum* é um conjunto de regras, procedimentos e práticas que são todos inter-relacionados e que trabalham juntos para melhorar o ambiente de desenvolvimento, reduzir despesas gerais da organização e garantir que os usuários através de entregas usando o modelo iterativo e incremental tenham produtos com alto valor agregado que correspondam as exigências do cliente(Hunt, 2006).

Segundo Lacey (2012), *Scrum* pode parecer simples de implementar , mas na verdade é bastante desafiador. É desafiador porque requer muito mais do que simplesmente colocar a mecânica no lugar certo e em seguir apertar um botão. Implementar *Scrum* exige equipes que estão dispostas a fazerem as seguintes mudanças:

- Desenvolver uma compreensão aos valores subjacentes do *Scrum.*
- Passar por uma mudança enorme de mentalidade, estar preparado para mudanças e sempre adaptar-se quando as mesmas ocorrerem.
- Lidar com problemas recém- expostos ou emergentes.
- Incorporar práticas de engenharia ágil.

Segundo Schwaber 2011, existem três pilares que são a base para o *Scrum*, Ver Figura 7 - Pilares do Scrum:

- Transparência- Aspectos importantes do processo devem ser vistos por todos os que participam do mesmo. A maneira de trabalhar de cada um e a visibilidade do trabalho estão sempre em evidência, exigindo um grau maior de comprometimento entre os as pessoas que utilizam o framework.
- Inspeção - A aplicação do framework se baseia em sempre investigar o problema , entender como funciona o contexto que se está inserido e as características dos problemas a serem enfrentados pela equipe e de como os artefatos estão sendo usados.
- Adaptação – Depois de inspecionar o ambiente, processos, pessoas e produtos usados , o Scrum prega que devemos sempre promover a melhoria contínua, adaptando as técnicas , artefatos e cerimônias a fim de que sempre se esteja em busca de entregar valor constantemente e para isso o framework pode estar sempre evoluindo dentro do projeto com o objetivo de se ter a solução mais eficiente para o contexto do problema e da organização ao qual está inserida.

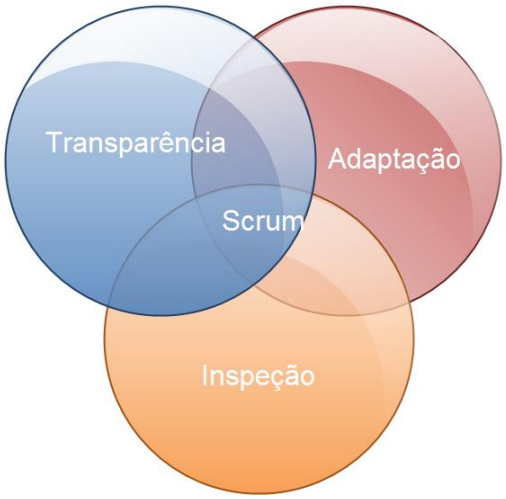


Figura 7 - Pilares do Scrum

O *Scrum* tem 4 cerimônias para promover inspeção e adaptação. São elas o planejamento da Sprint., a reunião diária , a Reunião de revisão da Sprint e a retrospectiva da Sprint.

No *Framework Scrum* existem somente 3 papéis, o de *Scrum Master*, o de *Product Owner* e o do Time Scrum.

Para completar o Framework são utilizados alguns artefatos como o *Backlog* do produto, o *Sprint Backlog* e o *Burn Down Chart*.

2.1.1 Os Papéis do Scrum

O *Scrum* apresenta o conceito de times auto gerenciáveis e auto organizáveis tendo um líder facilitador no processo chamado de *Scrum Master* e um *Product Owner* que é responsável por garantir o ROI(Retorno sobre o Investimento) do produto (Sabbagh, 2013).

2.1.1.1 O Scrum Master

O *Scrum Master* é o responsável por ser um líder servidor que trabalha para maximizar o trabalho da equipe (Leffingwell , 2011).

Segundo Schwaber (2011) e Sabbagh (2013), entre as funções do *Scrum Master* estão:

- Remover barreiras entre o Cliente e os Desenvolvedores.
- Ensinar o cliente a utilizar o Scrum fazendo com que maximifique o retorno sobre o investimento(ROI).
- Ajudar o *Product Owner* a priorizar o *Product Backlog*.
- Combater o Comando-Controle.
- Receber os impedimentos que surgem para equipe e ajuda-la a resolvê-los.
- Facilitar as reuniões.
- Garantir que Scrum esteja sendo usado da maneira correta.
- Entender e praticar a agilidade.

Segundo Cohn, 2011 , entre as características que um *Scrum Master* precisa ter podemos destacar , responsabilidade, boa comunicação , comprometimento , humildade e adquirir novas capacidades e habilidades constantemente, ser influente e sempre bem informado.

Muitas pessoas quando iniciam na função de *Scrum Master* lidam com uma aparente contradição de ser o líder servidor para equipe e ao mesmo tempo alguém sem autoridade sobre os membros da mesma. Mas com o tempo essa essa contradição logo desaparece ao se perceber que o Scrum Master tem autoridade sobre o processo. Ele talvez não possa demitir alguém mas pode dizer mudaremos nosso *sprint* de quatro para duas semanas (Cohn,2011).

O Papel do *Scrum Master* é multifacetado. Primeiro ele deve trabalhar para construir e manter uma equipe de alto desempenho. Olhando superficialmente parece um trabalho simples : gerenciar a reunião diária, coletar o status da equipe e coordenar várias reuniões e tarefas. Porém nada no trabalho do *Scrum Master* é simples. *Ele* não é o chefe da

equipe, pelo contrário, fazendo uma analogia, o *Scrum Master* é o liquido que garante que as engrenagens da equipe estão se voltando para uma eficácia máxima. Esses tem que manter a equipe focada e na trilha. Para fazer isso de forma eficaz, ele precisa ver através das árvores em uma floresta para conseguir ver o todo e identificar onde as equipes podem fazer melhorias (Lacey, 2012).

Em segundo lugar o *Scrum Master* também deve trabalhar junto com o *Product Owner* ,ajudando-o quando necessário, mas também fazendo o equilíbrio entre a vontade do mesmo e a capacidade do time. O Dono do produto vai querer que a equipe tenha uma alta velocidade, é o trabalho do *Scrum Master* fazer que isso aconteça , mas ao mesmo tempo protegendo a saúde da equipe. Bons *Scrum Masters* aumentam a velocidade da equipe através da remoção de impedimentos , monitorando a saúde do time e ajudando a equipe a saber se ela está seguindo em direção ao objetivo do Sprint.(Lacey , 2012 , Schwaber, 2001).

O trabalho final do *Scrum Master* é de ser um agente da mudança na organização .Um bom *Scrum Master* ajuda a evoluir a equipe para melhor, e também ajuda a transformar a organização, procurando oportunidades para criar uma organização de aprendizagem, proporcionando a equipe um bom ambiente de trabalho, incentivando a colaboração, dando treinamentos informais, aconselhando novas equipes *Scrum* e sendo um defensor do *framework* dentro da empresa(Lacey, 2012). O papel do *Scrum Master* é fundamental para o sucesso do Scrum, mas é o papel mais incompreendido. Não é nem uma liderança tradicional e nem um papel de gestão. A liderança servidora do *Scrum Master* pode ter um impacto muito positivo dentro de toda empresa, O Scrum Master melhora o uso do *Scrum* através da facilitação de treinamentos e para eliminação rápida de tudo que distrai a equipe de entregar valor.(Keith, 2010).

Um outro trabalho importante dele é incentivar a equipe a buscar maneiras de melhorar seu desempenho como um time. Isto nunca cessa. Mesmo com as equipes mais produtivas, o *Scrum Master* encoraja-os a procurar até mesmo um ponto percentual de melhoria. Isto promove uma cultura de melhoria contínua. Melhorias podem ser tão simples como aproximar as mesas da equipe para melhorar a comunicação ou tão duro como requerer uma nova tecnologia para melhorar a eficiência da linha de produção. Ele Garante o cumprimento das regras do processo ágil: As regras de agilidade são leves e flexíveis, mas são regras, no entanto, este papel é responsável por reforçar as regras com a equipe (Leffingwell,2010).

O papel *Scrum Master* é principalmente de facilitador podendo reconhecer os problemas perante a equipe e identificar uma solução favorecida, mas nunca deve conduzir através da implementação da solução. Um *Scrum Master* irá ajudar uma equipe a reconhecer os problemas e e se possuem uma solução. Isto ensina-lhes a habilidade de valor inestimável de identificar e resolver problemas por conta própria. Em muitos aspectos, o papel *do Scrum Master* é para treinar a equipe a eliminando gradativamente a sua necessidade junto ao time de *scrum*(Keith, 2010).

O *Scrum Master* confia na equipe, ele orienta a equipe para fazer o seu melhor trabalho através de treinamento e facilitação. O seu papel não é fácil, mas é gratificante, ele tem de ser teimoso e persistente.

Muitos problemas que enfrentam uma equipe necessitam de intervenção a nível pessoal com pessoas com dificuldades de mudar seus comportamentos. Por exemplo, ter um gerente de autoridade considerável e muitos anos de experiência em um ambiente de comando e controle que não acredita em auto-organização. Este gerente interfere em uma equipe de maneira que distraía equipe através da atribuição de um novo trabalho no meio de um *sprint*.

O Scrum Master deve persistentemente lembrar o gerente sobre o propósito de *Scrum* e os compromissos recíprocos entre a equipe e os *stakeholders*. Isso precisa ser feito de uma forma que não crie um clima ruim ou levantem barreiras. É um papel de *coaching*. Nem todos podem fazê-lo.(Krebs, 2008) e (Keith, 2010).

Pode o Scrum Master também ser um membro da equipe?

Um Scrum Master não é geralmente um desenvolvedor na equipe ele pode lidar com duas a quatro equipes antes de seu papel começar a se tornar um trabalho de tempo integral. Depende de quantos impedimentos existem na organização que o ele tenha que resolver. (Keith,2010).

As equipes muitas vezes perguntam: "Se o *Scrum Master* ficar como desenvolvedor na equipe?" Não é aconselhável um Scrum Master ser um desenvolvedor na equipe. O *Scrum Master* como um membro da equipe pode causar alguns problemas:

- Ele se concentra em suas próprias tarefas mais do que sobre o papel *Scrum Master*.
- Ele prioriza seus impedimentos próprios sobre os de outros colegas.

Às vezes não há escolha, mas para ter o *Scrum Master* recrutado entre os desenvolvedores na equipe, quando isso acontecer, todos na equipe precisam tomar cuidado com esses problemas.

Quando for necessário, uma solução é um desenvolvedor na equipe assumir o papel do *Scrum Master*: Eles terem um chapéu com eles. Eles vestirem o chapéu quando eles estão no papel *Scrum Master* e tirá-lo quando estão na função de desenvolvedor. Isso ajuda a equipe sabe quem está falando com eles.(Lacey,2012).

2.1.1.2 O Product Owner (Dono do Produto)

O *Product Owner* é um papel fundamental para que o Scrum funcione corretamente. Ele é o responsável para liderar um esforço para ter um produto vencedor(Picheler, 2011).

Segundo Schwaber,2011, ele é responsável por maximizar o valor do produto e do resultado do trabalhado da equipe .Dentre as atividades *do Product Owner* está a de gerenciar o *backlog* do produto e isso inclui:

- Expressar com clareza os itens que compõe o backlog do produto;
- Ordenar os itens do *backlog* para melhor alcançar os objetivos de negócio.
- Garantir que a equipe de desenvolvimento esteja entregando valor;
- Garantir que o *backlog* de produto seja transparente com boa visibilidade e mostre os próximos requisitos a serem trabalhados.
- Garantir que o time de desenvolvimento entenda os itens de *backlog* no nível necessário.

Esse trabalho descrito acima pode ser realizado pelo *Product owner* ou deixar que a equipe o faça mas o *Product Owner* é que tem a responsabilidade sobre essas atividades.

O Dono do Produto faz parte da equipe Scrum e colabora fortemente com seus membros.

Uma observação importante é que o papel de Product *Owner* é exercido por uma pessoa, não por um grupo. Entretanto, essa pessoa pode ser aconselhada e influenciada por outros patrocinadores do projeto, porém suas decisões devem ser respeitadas, de forma que ninguém além do *Product Owner* possa mudar as prioridades estabelecidas por ele(Armony, 2010).

Segundo Cohn, 2010 Uma grande responsabilidade do Dono do Produto é estabelecer uma visão para o produto. Uma boa visão serve para entusiasmar uma equipe através do compartilhamento desse visão por todos para atingir um objetivo comum que é gerar valor de negócio. Perguntas importantes que ele deve fazer: Para quem podemos vender? O que é exclusivo no nosso produto? O que nossos concorrentes estão criando? Como nosso produto irá evoluir?

Essas perguntas podem variar dependendo do produto ou serviço que estiver sendo desenvolvido, mas ter uma visão compartilhada desse produto é importante para motivar a equipe e criar uma conexão ao longo do tempo entre os desenvolvedores e usuários do produto (Cohn,2010).

Segundo Picheler, 2011 , para um dono do produto ser bem sucedido em suas atividades ele deve possuir algumas características importantes:

1. Visionário e realizador – *O Product Owner* é um visionário que pode vislumbrar o produto final e comunicar essa visão para os demais. Ele também é um realizador capaz de enxergar a visão toda até o fim, que incluí desde descrever requisites, colaborar de perto com a equipe, rejeitar e aceitar resultados dos trabalhos e dirigir o projeto, acompanhando e prevendo seu progresso. Ele é um empreendedor que encoraja a inovação, facilita a criatividade e se sente bem com a mudança, ambiguidade,, debate, conflito, diversão, experimentação e exposição aos riscos.
2. Líder e participante da equipe – Como o responsável pelo sucesso do produto, ele da orientação e direção para cada envolvido no desenvolvimento e garante a tomada de decisões difíceis. Ele deve atuar como um pastor para o processo de inovação, orientando o projeto e buscando um consenso da equipe na hora da tomada de decisões.

3. Ter uma equipe empresarial – Um bom *Product Owner* precisa de uma equipe para que seu produto ganhe vida. Ele precisa de pessoas para avaliar suas idéias testar e até mesmo validar essas idéias.
4. Disponível e qualificado - Ser esse tipo de professional é uma função na maioria das vezes de tempo integral. Ser adequadamente qualificado requer um bom conhecimento do cliente e do Mercado, ser entusiasmado com a experiência do usuário, ter a capacidade de dizer suas necessidades , e descrever os requisitos do projeto, gerenciar um orçamento, orientar a equipe de desenvolvimento e ter confiança em uma equipe multidisciplinar , e auto-organizada.

Segundo Pinchler, 2011, um bom *Product Owner* deve estar sempre colaborando com o *Scrum Master*. Assim como um time esportivo precisa de um técnico para jogar de forma consistente em um nível mais alto, toda equipe Scrum precisa de uma Scrum Master. Esse profissional da suporte ao *Product Owner* e aos membros participantes, protege-os do processo e intervém de modo apropriado quando for necessário para garantir que o ritmo de trabalho seja sustentável , que a equipe permaneça motivada e que nenhum prejuízo técnico seja contraído.

Os papeis do *Product Owner* e do *Scrum Master* são complementares. O primeiro é responsável pelo "que"- Criar o produto correto – e o segundo pelo "como" – usar o Scrum da maneira correta. Esses dois aspectos são representados na figura . Ver Figura 8 - Fazer a Coisa Certa do Jeito Certo(Pinchler, 2011).

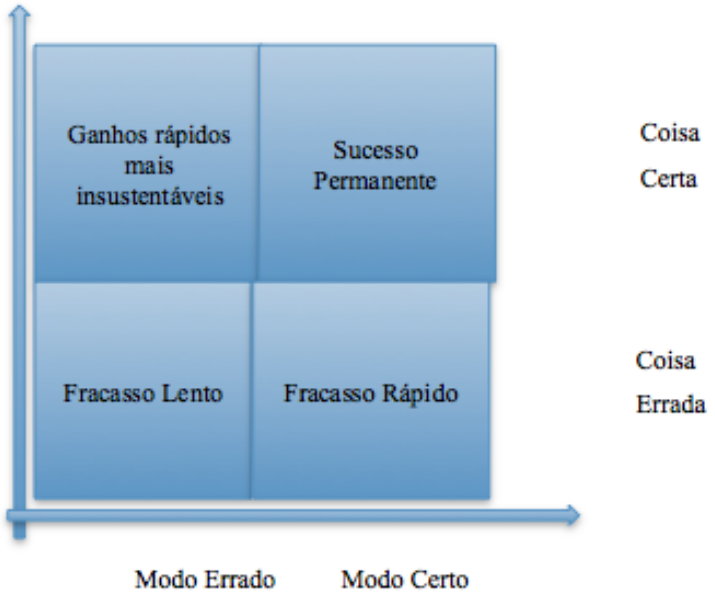


Figura 8 - Fazer a Coisa Certa do Jeito Certo(Pinchler, 2011).

Somente quando o produto certo é criado com processo correto se alcança o sucesso permanente. Podemos concluir também que um Scrum Master ruim e um mau Product owner levarão a um fracasso lento, o que pode não dar tempo de tomar ações corretivas para o projeto.

2.1.1.3 O Time Scrum

O time Scrum ou equipe de desenvolvimento consiste em profissionais que tem o trabalho de entregar uma versão utilizável que incrementa o produto "Pronto" no final de cada Sprint .(Schwaber, 2011).

Essa equipe é auto gerenciável e auto- organizável. O time é o responsável em quebrar os itens priorizados pelo *Product Owner* em incrementos de funcionalidades utilizáveis(Keith, 2010).

Equipes de desenvolvimento são multi funcionais , possuindo todas as habilidades necessárias para criar um incremento do produto(Cohn, 2010).

O Scrum não reconhece perfis da equipe de desenvolvimento além do perfil de desenvolvedor, independente do trabalho a ser realizado por cada um dos membros. A responsabilidade é do time inteiro e não apenas de membros isolados mesmo esses tendo habilidades e especialidades diferentes. Equipes de desenvolvimento não tem subequipes dedicadas a domínios particulares tais como testes, análise ou gerência de configuração. O time é multisciplinar e o conhecimento deve ser compartilhado entre os membros da equipe para que não haja dependência de especialistas.(Schwaber, 2011).

Equipes de desenvolvimento devem ser pequenas , de no máximo 9 pessoas e todo processo de engenharia deve emergir da equipe e ser inspecionado e adaptado constantemente em busca de uma melhoria contínua.(Sabbagh, 2010 apud Schwaber, 2004).

Embora se recomende equipes de até 9 pessoas, isso varia muito, o que é decisivo para o tamanho ideal de uma equipe é principalmente uma boa comunicação. Enquanto a equipe estiver se comunicando e conseguindo produzir e resolver internamente seus problemas , podemos dizer que essa equipe esteja em um bom tamanho.(Cohn, 2010).

Segundo Cohn 2010, existem muitas vantagens em termos equipes pequenas e são elas:

- Menos Ociosidade Social - Quanto menos pessoas estiverem no projeto o risco de pessoas deixarem de fazer suas tarefas pensando que poderá existir outras pessoas para assumirem se torna bem menor. Em equipes maiores a uma tendência das pessoas se esforçarem menos quando sabem que tem outras que poderão assumir suas responsabilidades.
- A interação de forma construtiva tem uma maior probabilidade de ocorrer em equipes menores - Equipes de mais de 10 pessoas tem dificuldades de estabelecer uma relação de confiança entre si, responsabilidades mutuas e coesão.
- Menos tempo é gasto na coordenação dos esforços – Equipes grandes tomam maior tempo em uma planejamento e organização.
- Equipes pequenas dão mais satisfação a seus membros – Em equipes pequenas a visibilidade e o êxito das conquistas são mais visíveis e mais significativas.
- Existe mais coesão – Em equipes pequenas, geralmente todos participam das decisões e essas equipes são mais unidas.

- Menor Especialização individual – Em equipes pequenas é reduzido o risco de pessoas terem papéis específicos e especialistas, favorecendo a multidisciplinaridade.

Uma equipe de desenvolvimento Scrum deve ser auto-organizada: seus membros devem saber que seu trabalho é igual e fundamentalmente importante para o sucesso de todos. Não existe hierarquia, cada um deve possuir consciência de seu compromisso com a meta e pessoas sem comprometimento devem deixar a equipe (Bassi, 2008).

A equipe se fiscaliza e se gerencia, cobrando um dos outros empenho para com suas funções e para o sucesso de cada *Sprint.(*Fonseca, 2010).

2.1.2 Artefatos do Scrum

O Scrum tem alguns artefatos que são utilizados dentro do framework. Dentre eles citamos a visão do Produto , o *Backlog* , o *Sprint Backlog* e o *Burndown Chart.*

2.1.2.1 A Visão do Produto

A visão é uma meta que deve ser atingida com o produto, ela serve para guiar as pessoas em um objetivo de negócio. Segundo Pichler, 2011, uma visão eficaz deve responder as seguintes perguntas :

- Quem vai comprar o produto ? Qual consumidor alvo ?Quem Utilizará o produto ?Quem é seu público alvo ?
- Que necessidades esse produto irá resolver ? Que valor ele irá agregar?
- Quais os atributos do produto são críticos para atender às necessidades selecionadas para ter sucesso ?Como ele vai se comportar e em que áreas se destacará?

- Como o produto pode ser comparado com os seus concorrentes tanto como os da mesma empresa que sejam semelhantes ?
- Como a empresa ganhará dinheiro vendendo o produto ?Quais os modelos de negócio a serem adotados?
- O Produto tem viabilidade? A empresa pode desenvolvê-lo e vendê-lo ?

A visão deve comunicar a essência do futuro produto de maneira clara e descrever uma meta que seja compartilhada oferecendo uma direção e ser ampla o suficiente que facilite a criatividade.

2.1.2.2 O Product Backlog

O Artefato do Scrum mais conhecido é o *product backlog* e tem um motivo, por ser muito simples. Ele na verdade é uma lista priorizada do trabalho pendente necessário para dar vida a um produto. Ele substitui os artefatos de requisitos tradicionais como as especificações de requisitos. O *Product Owner* é o responsável por gerenciar o *Product Backlog* e junto com ele a equipe, *stakeholders* e *Scrum Master* contribuem para isso.(Cohn, 2010).

Segundo Picheler (2011), um bom *Product Backlog* tem como qualidades fundamentais:

- Ser detalhado adequadamente - Quanto maior a prioridade maior o nível de detalhes e quanto menor a prioridade, menor o nível de detalhes.
- Estimado - Estimativas em geral são brutas e em pontos de histórias ou dias ideais.
- Emergente - Ele evoluí o seu conteúdo mudando com frequência. Novos itens são adicionados ou removidos com base na prioridade do cliente ou usuário. O itens existentes são adicionados, modificados e removidos de forma frequente.
- Priorizado – Todos os itens do *Product Backlog* são priorizados e o mais importantes ficam sempre no topo e os menos importantes na base. Depois de

concluídos esses ités vão sendo removidos do Backlog e colocados na condição de pronto.

2.1.2.3 O Sprint Backlog

O *Sprint Backlog* compreende todas as atividades necessárias para atingir o objetivo do *Sprint.* A equipe cria o *Sprint Backlog* na reunião de planejamento e o atualiza pelo menos uma vez por dia. Durante as atualizações o time pode remover as redundantes e acrescentar outras novas e registra os esforço para cada tarefa.(Pincheler, 2011).

2.1.2.4 O Burn Down Chart

É um gráfico para o time visualizar seu progresso e ver a probabilidade de o time cumprir a meta do *Sprint* ou da release podendo então adaptar seu trabalho conforme o comportamento do gráfico., geralmente ele é medido em esforços restantes no *Product Backlog* e tempo(Pincheler, 2011).Ver Figura 9 :

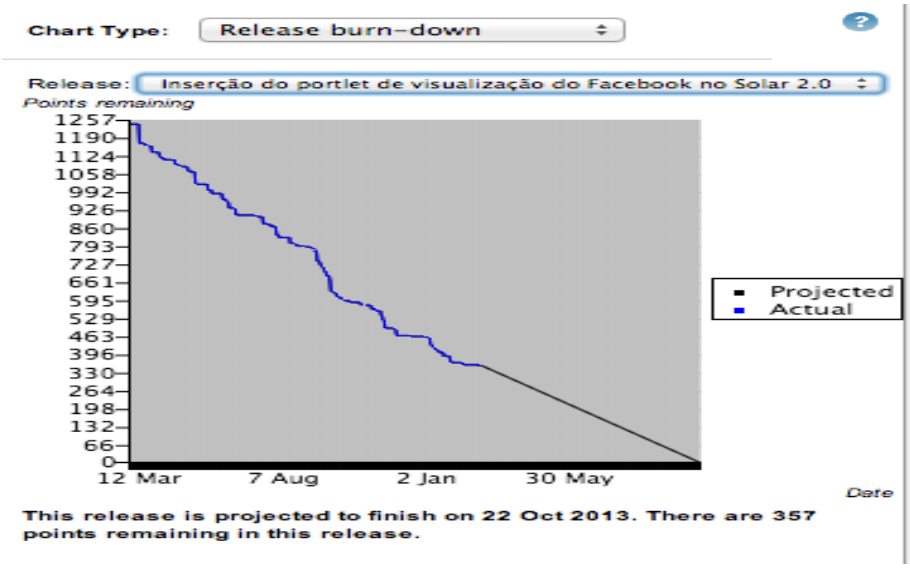


Figura 9 - Burn Down de Um Release do Projeto Solar 2.0

2.1.3 Scrum em Poucas Palavras

Antes de descrevermos o fluxo do Scrum, entendamos que o Scrum é um framework que pode ser resumido de forma simples segundo Kniberg e Skarin(2009) :

- **Divida sua organização** em times pequenos, auto–organizados e multifuncionais. Ver Figura 10:

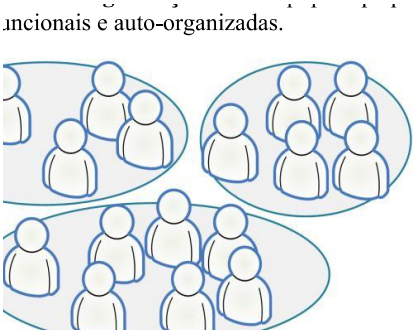


Figura 10 - Divisão em Times Pequenos(Kninberg , Skarin, 2009).

.

- **Divida o Trabalho a ser realizado** em uma lista de entregáveis pequenos e bem definidos. Organize e Priorize essa lista e estime o esforço relativo de cada item. Ver Figura 11 :

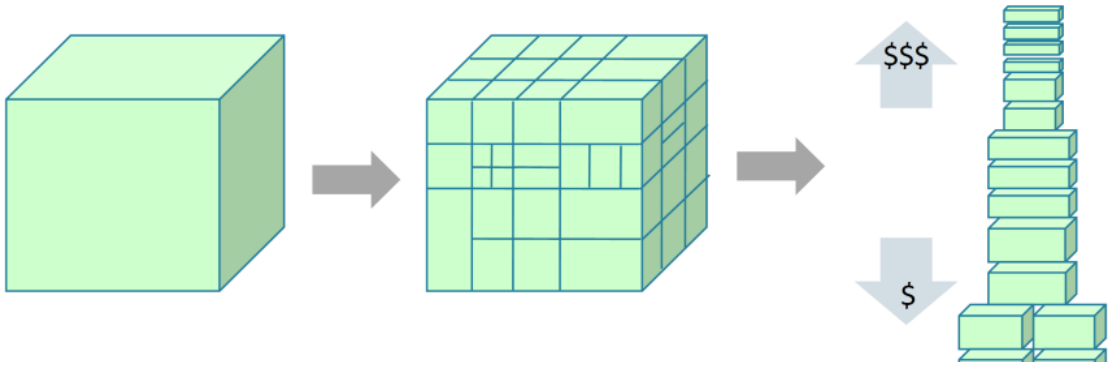


Figura 11 - Divida o Trabalho a Ser Realizado(Kninberg, Skarin,2009).

- **Divida o Tempo** em pequenas iterações de duração fixa(normalmente 1 a 4 semanas), ao final de cada iteração deve ser desenvolvido e entregue código funcionando. Ver Figura 12 :

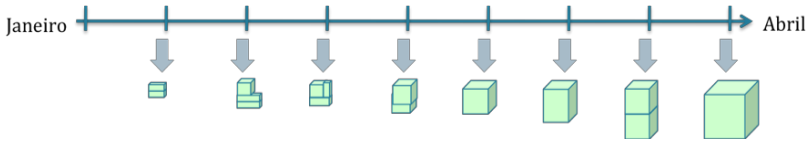


Figura 12 - Divida o Tempo em Interações(Kninberg, Skarin, 2009).

- **Otimize o Plano de entrega** e com a colaboração com os clientes atualize as prioridades frequentemente através da inspeção da entrega ao final de cada iteração.

- **Otimize o Processo** através de uma retrospectiva depois de cada iteração para a busca da melhoria contínua e adaptação as necessidades dos clientes.

Resumidamente, temos uma pequena equipe trabalhando em intervalos de tempos curtos, construindo algo pequeno. Mas integrando regularmente para entregar um todo, e fazendo melhorias a cada iteração a fim de entregar sempre algo de valor ao cliente.

2.1.4 As Cerimônias do Scrum

Agora que entendemos de maneira geral o resumo do framework e vimos os papéis que o compõe vamos entender o que cada cerimônia representa dentro desse fluxo.

2.1.4.1 A Reunião de Planejamento

A reunião de planejamento é a primeira cerimônia do Scrum, ela é dividida em duas partes. A primeira parte a equipe entenderá o que será feito no Sprint e a segunda parte entenderá como irá fazer (Sabbagh, 2013).

Na primeira parte o *Product Owner* irá priorizar o *Product Backlog* (Lista de funcionalidades a serem desenvolvidas.) o *Product Owner* apresenta ao Time o que é mais prioritário no *Product Backlog*. Eles juntos irão definir qual funcionalidade deverá ser desenvolvida durante o Sprint. As entradas para essa reunião são o Product Backlog, o incremento mais recente ao produto, a capacidade da equipe e o histórico de desempenho da equipe de desenvolvimento.

Cabe somente a equipe a decisão de quanto do Backlog ele deve selecionar. Somente o Time pode avaliar o que é capaz de realizar no Sprint. Nessa etapa é definida a meta do Sprint e criar os *backlogs* selecionados conforme as prioridades de negócios estabelecidas pelo Dono do Produto (*Product Owner*) (Bassi, 2008).

É importante destacar que para o estabelecimento de uma meta, o *Time* deve ter um conceito de pronto único para o projeto, o conceito de pronto" para um desenvolvedor é

quando a codificação está encerrada, para um analista de teste é quando a execução dos testes e verificação dos problemas encontrados estejam concluídas, para o Dono do Produto uma funcionalidade pronta se dá quando foi entregue aos usuários finais.(Picheler,2011).

Assim, se não houver um consenso a respeito do que é considerado pronto para todos os envolvidos no projeto, torna-se inviável avaliar se a meta estabelecida foi realmente atingida .(oran,2010).

Na segunda etapa da reunião de planejamento do *sprint* , o time se reúne para quebrar os itens do *backlog* selecionado em tarefas de implementação de menor granularidade e atribuindo pontos de complexidade para as mesmas e definindo o que irá caber no *Sprint Backlog*.(Oran ,2010).

Selecionando o trabalho do *Sprint*, a Equipe de Desenvolvimento decide como irá construir essas funcionalidades durante o *Sprint* e transformá-las em um incremento de produto "Pronto". Os itens de Backlog do Produto selecionados para a Sprint, junto com o plano de entrega destes itens é denominado de Backlog da Sprint.(Schwaber,2011)

No final da reunião de planejamento da Sprint, o Time de Desenvolvimento deve ser capaz de explicar ao *Product Owner* e ao *Scrum Master* como pretende trabalhar como equipe auto- organizada para completar o objetivo da Sprint e criar um produto potencialmente intregável.(Schwaber, 2011).

Para um planejamento de sucesso é preciso de um bom *Product Owner*. Um PO que ajude a equipe a entender o que tem que ser feito, depois ela descobre o quanto pode ser feito e como faze-lo.(Pichler, 2011)

2.1.4.2 A Reunião Diária

O maior objetivo da reunião diária é promover a transparência e comunicação entre a equipe permitindo que a equipe gerencie seu trabalho e descubra impedimentos diariamente, e com ajuda do Scrum Master, podendo remover esses impedimentos para que o fluxo da entrega e objetivo da Sprint sejam atingidos.(Cohn, 2010).

A Reunião Diária é um fluxo para que seja aplicado o micro gerenciamento e é uma das cerimônias mais importantes para o sucesso do Scrum em um projeto. É basicamente um evento de 15 minutos no máximo para que o time de desenvolvimento possa sincronizar atividades e criar um plano paras as próximas 24 horas. Através da reunião diária que se identifica impedimentos e que se promove a colaboração da equipe para ajudar a resolvê-los. Não existem problemas individuais. Problemas e soluções são buscadas por todos juntos da melhor maneira possível.(Sabbagh,2010).

Nessa reunião são feitas 3 perguntas :

- O que você fez desde a última reunião ?
- Que dificuldades você encontrou ?
- O que você vai fazer hoje ?

Deve ser objetiva e serve principalmente pra identificar os problemas e descobrir quem possa resolvê-los. Não é a hora para detalhar as soluções, deve ser objetiva e direta. Por meio dessa reunião, o *Time* e o *Scrum Master* ganham visibilidade de como está o caminho para a meta e planejam o dia seguinte de trabalho.(Schaweber, 2011).

2.1.4.3 A Reunião de Retrospectiva

A Retrospectiva do Sprint é uma oportunidade para o Time Scrum inspecionar a si próprio e criar um plano para melhorias contínuas a serem aplicadas no Sprint seguinte. A Retrospectiva do Sprint ocorre depois da Revisão da Sprint e antes da reunião de planejamento do Sprint seguinte.

Esta é uma reunião com tempo variável dependendo da periodicidade do Sprint. Proporcionalmente um tempo menor é alocado para Sprints menores. A literatura sugere um tempo de 3 horas para sprints de 1 mês, porém tudo é baseado em inspeção e adaptação. Não existe uma receita de bolo bem definida.(Schaweber,2011)

O propósito da Retrospectiva da Sprint é:

- Inspecionar como o último Sprint foi em relação as pessoas, relações, processos e ferramentas;

- Identificar e ordenar os principais itens que foram bem e as potenciais melhorias; e por um plano para implementar melhorias na maneira que o time faz seu trabalho;
- O Scrum Master encoraja o Time a melhorar, dentro do processo do framework do Scrum, o processo de desenvolvimento e as práticas para fazê-lo mais efetivo e agradável para a próxima Sprint. Durante cada Retrospectiva do Sprint, se planeja formas de aumentar a qualidade do produto, adaptando a definição de "Pronto" quando apropriado. Ao encerrar a mesma deverão ter identificado melhorias que serão implementadas no próxima Sprint. A implementação destas melhorias é a forma de inspeção e adaptação que o time faz, A Retrospectiva do Sprint fornece um evento dedicado e focado na inspeção e adaptação, no entanto, as melhorias podem ser adotadas a qualquer momento. (Phan, 2011)

2.1.4.4 A Reunião de Revisão do Sprint

Quando o *sprint* termina, a equipe apresenta o trabalho na reunião de **revisão do *sprint*** ao *Product Owner"* através de uma demonstração. Este faz testes para verificar se cada item atende às suas expectativas e determinar se a meta foi atingida. A meta sempre "é atingida" ou "não é atingida". Não existem opções intermediárias. Desta forma a equipe deve se preocupar com a qualidade de tudo que entrega para não colocar em risco o trabalho de todo o *sprint* .(Bassi, 2008).

Na reunião de entrega do *sprint* participam o time, o *Scrum Master* e o *Product Owner* que deve convidar os stakeholders do projeto para apresentar os resultados do *Sprint.* É importante saber que o *Product Owner* faz parte do time seja este estando do lado do cliente dou não. Ele é o responsável na *review* por estar validando cada item de *backlog* apresentado pelo time. Durante a *review* o *Product Owner* pode também apresentar junto com o time os resultados para os *stakeholders* para validação dos resultados. Após isso o mesmo faz observações em cima dos resultados que servirão para os *sprints* seguintes(Sabbagh, 2010).

2.1.5 O Fluxo do Scrum

A junção de todas as cerimônias, papéis e artefatos utilizados no *Scrum* formam o ciclo de trabalho do *Scrum, Ver* Figura 13 :

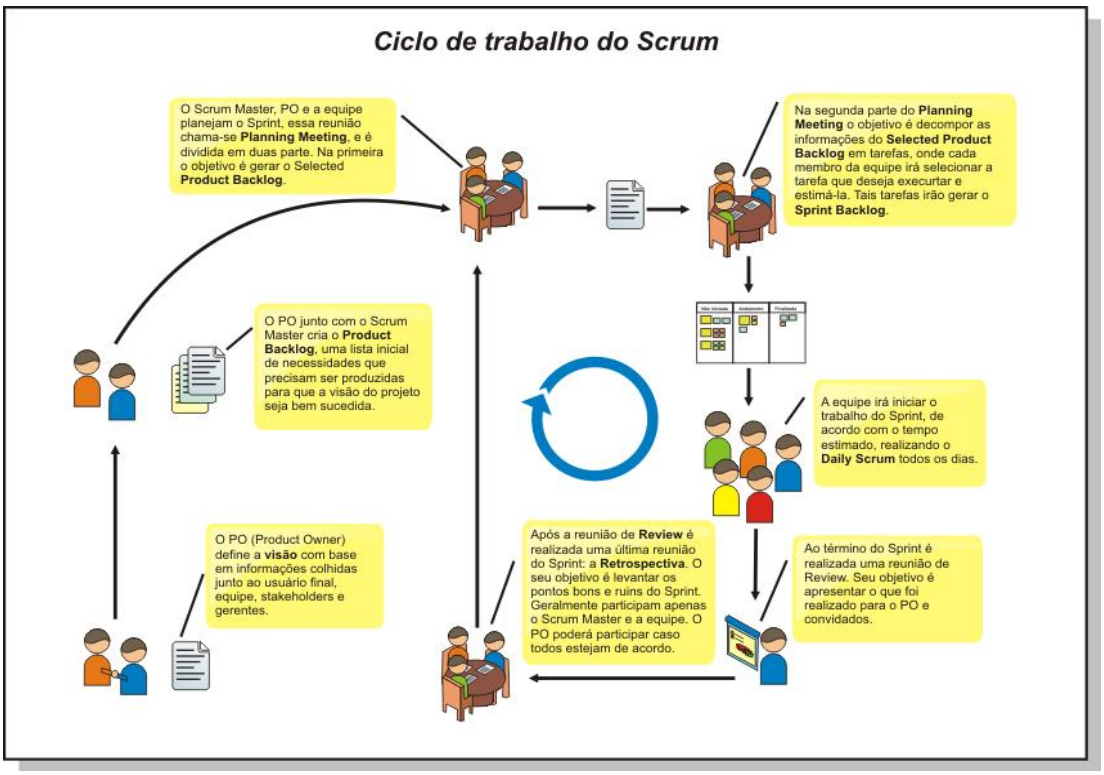


Figura 13 - Ciclo de Trabalho do Scrum.

Esse ciclo ocorre iniciando com o *Product Owner* definindo uma visão para o produto a ser desenvolvido e apresentando essa visão para o time, após isso ele e o *Scrum Master* criam o *Product Backlog* que é uma lista de necessidades que precisam serem produzidas para que a visão do projeto seja bem sucedida, posteriormente após esse passo, eles se juntam ao time para planejar o *Sprint*, dando origem a reunião chamada de *Sprint Planning* que é dividida em duas partes, onde na primeira teremos os itens priorizados do *backlog*, na segunda parte esses itens serão quebrados e estimados e isso entrará *no Sprint backlog.* Após isso a equipe inicia os trabalhos do Sprint, e fará pontos de controle todos os dias chamados de *Daily Scrum (* Reunião diária). No termino do *Sprint* é feita uma reunião para apresentar os resultados onde todos participam , podendo ter convidados externos para a validação do resultado. Por último temos a reunião de retrospectiva onde todos podem participar e será colocado o que funcionou e o que pode melhorar em relação o *Sprint* que terminou.

2.2 Kanban

O Kanban vem da cultura kaizen que prega a eliminação do desperdício e promove a melhoria contínua.(Anderson, 2011).

O Kanban é baseado numa idéia muito simples. As Atividades em andamento devem ser limitadas. Algo novo só deve ser iniciado quando uma peça de trabalho existente é liberada ou quando uma função automática inicia isso.(Anderson, 2011).

O Kanban, ou cartão de sinalização, é um sinal visual produzido indicando que novo trabalho pode ser iniciado e que a atividade atual não coincide com o limite acordado. Isso não soa muito revolucionário nem parece afetar profundamente o desempenho, cultura, capacidade e maturidade de uma equipe e a organização na qual está inserida. Mas o impressionante é que afeta! O Kanban parece uma mudança pequena e, no entanto, muda tudo a respeito de uma empresa(Kninberg, Skarin, 2009).

O que percebemos sobre o Kanban é que ele é uma abordagem para mudança gerencial. Ele não é um processo ou ciclo de vida de gerenciamento de projetos ou de desenvolvimento de software. O Kanban é uma abordagem para introduzir mudanças em um ciclo de desenvolvimento de software ou metodologia de gerenciamento de projetos. O princípio do Kanban é que você inicia com o que estiver fazendo agora. Você entende seu processo atual ao mapear o fluxo de valor e ao concordar, em seguida, em limitar as Atividades em andamento (do inglês WIP) para cada estágio desse processo. A partir daí você começa a rastrear as atividades pelo sistema para iniciá-las quando os sinais do Kanban aparecerem.(Anderson,2011).

O Kanban tem sido útil para equipes ágeis de desenvolvimento de software, mas tem ganhado popularidade, igualmente, em equipes que utilizam uma abordagem mais tradicional. Ele está sendo introduzido como parte de uma iniciativa Lean (enxuta) para moldar a cultura das organizações e encorajar a melhoria contínua.(Hiranabe, 2007).

Porque o WIP é limitado em um sistema Kanban, tudo que fica bloqueado por qualquer motivo tende a parar o sistema. Se certa quantidade de itens de trabalho fica bloqueada, todo o processo pára de funcionar. Isso cria a necessidade de concentrar toda a

equipe e toda a empresa na solução do problema para desbloquear o item e restaurar o fluxo(Anderson,2010).

Segundo , Kninberg, Skarin(2009). O Kanban usa um mecanismo de controle visual para acompanhar o trabalho à medida que ele flui através das várias etapas do fluxo de valor.

Tipicamente usa-se um quadro branco com post-its, ou um sistema de cartões eletrônicos. Fazer os dois é, provavelmente, uma boa prática. A transparência que isso gera também contribui para a mudança cultural. Métodos ágeis têm sido bons provendo transparência sobre as atividades em andamento e concluídas, e reportando métricas como velocidade (a quantidade de trabalho finalizado em uma iteração).

O Kanban, no entanto, vai um passo além e dá transparência ao processo e seu fluxo. O Kanban expõe gargalos, filas, variabilidade e desperdício. Tudo que impacta o desempenho da organização em termos de quantidade de trabalho de valor entregue e o tempo de ciclo necessário para entregá-lo. Proporciona aos membros da equipe e às partes interessadas externas a visibilidade sobre os efeitos de suas ações (ou falta de ações). Sendo assim, os primeiros estudos de caso estão mostrando que o Kanban muda o comportamento e incentiva uma maior colaboração no trabalho.(Hiranabe, 2008).

A visibilidade dos gargalos, desperdício e variabilidade, e seus impactos, também incentivam a discussão sobre melhorias e as equipes rapidamente iniciam as melhorias nos seus processos.

Como resultado, o Kanban encoraja a evolução incremental de processos existentes, evolução que é geralmente alinhada a valores Ágeis e Lean. O Kanban não demanda uma revolução drástica no modo como as pessoas trabalham, encoraja, ao invés disso, uma mudança gradual. É mudança entendida e aceita por consenso entre trabalhadores e seus colaboradores.

O Kanban, através da natureza do sistema puxado, encoraja também comprometimento tardio, tanto em priorização de trabalho novo quanto na entrega de trabalho existente.

Tipicamente, os times vão concordar em uma cadência de priorização para atender partes interessadas que estejam contra a maré e decidir a próxima coisa em que trabalhar. Estas reuniões podem ser feitas regularmente porque são normalmente muito curtas.(POPPENDIECK, 2003).

Uma questão muito simples deve ser respondida, algo como, ”Desde a nossa última reunião, 2 vagas ficaram livres. Nosso tempo de ciclo corrente é de 6 semanas para entrega. Quais 2 coisas vocês mais gostariam de ter entregues daqui a 6 semanas?” Isso tem dois desdobramentos. Perguntar uma simples questão geralmente resulta em uma resposta de boa qualidade, rapidamente desdobrada, e mantém a reunião breve.(Anderson,2011).

A natureza da questão significa que o comprometimento em que se trabalhará é atrasado até o último momento responsável. Isto aumenta a agilidade gerenciando expectativas, diminuindo tempos de ciclo do comprometimento à entrega e eliminando retrabalho, pois a chance de que prioridades mudem é minimizada.

Uma última coisa sobre Kanban é que o efeito de limitar o WIP fornece previsibilidade de tempo em ciclos e faz as entregas mais confiáveis. A abordagem de "parar a linha de produção" para superar os obstáculos e os erros encontrados, também parece encorajar níveis mais elevados de qualidade e uma queda rápida de retrabalho.

2.2.1 Cultura Kaizen

Para entender porque é tão difícil alcançar uma cultura *kaizen*, primeiro devemos entender como tal cultura seria. Só então podemos discutir por que podemos querer alcançar tal cultura e quais seriam os seus benefícios.(Anderson, 2011).

Na cultura *kaizen* a força de trabalho tem poder. Indivíduos sentem-se livres para agir; livres para fazer a coisa certa. Eles espontaneamente se debruçam sobre os problemas, discutem as opções e implementam correções e melhorias. Em uma cultura *kaizen,* a força de trabalho não tem medo. A norma subjacente é para a gestão ser tolerante a falhas se a experimentação e a inovação fizerem parte do processo - assim como a melhoria de desempenho. Em uma cultura *kaizen,* indivíduos são livres (dentro de alguns limites) para se auto-organizarem em torno do trabalho que eles fazem e como eles o fazem. Controles visuais e sinais são evidentes e as pessoas se oferecem para trabalhar em tarefas de trabalho ao invés de serem atribuídas por um superior. Uma cultura *kaizen* envolve um elevado nível de colaboração e uma atmosfera de coleguismo onde todo mundo olha para o desempenho da

equipe e os negócios acima de si. Uma cultura *kaizen* centra-se no pensamento em nível de sistemas ao fazer melhorias locais que melhoram o desempenho geral.(Anderson, 2011).

Uma cultura *kaizen* tem um elevado nível de capital social. É uma cultura de alta confiança em que indivíduos, independentemente de sua posição na hierarquia de tomada de decisões de negócios, respeitam uns aos outros e a contribuição de cada pessoa. Culturas de alta confiança tendem a ter estruturas mais horizontais do que culturas de confiança inferior. É o grau de capacitação que permite uma estrutura mais horizontal trabalhar de forma eficaz. Assim, alcançar uma cultura *kaizen* pode permitir a eliminação de desperdício de camadas de gerenciamento e reduzir os custos de coordenação, como resultado.(POPPENDIECK, 2005).

Muitos aspectos de uma cultura *kaizen* estão em oposição às normas culturais e sociais estabelecidas na cultura ocidental moderna. No ocidente, somos levados a ser competitivos. Nossos sistemas de ensino incentivam a concorrência nos estudos e nos esportes atléticos. Mesmo nossos esportes de equipe tendem a incentivar o desenvolvimento de heróis e equipes construídas em torno de um ou dois jogadores excepcionalmente talentosos. A norma social é enfocar primeiramente o indivíduo e confiar em pessoas extraordinárias para alcançar a vitória ou para nos salvar do perigo. Não é de se admirar que tenhamos dificuldade no local de trabalho para incentivar coleguismo e pensamento no nível de sistemas e cooperação.(Anderson,2011).

2.2.2 Resumo

Kninberg e Skarin(2009) descrevem o Kanban em poucas palavras :

- Visualize o fluxo de trabalho
- Divida o trabalho em partes, escreva cada item em um cartão e coloque na parede.
- Use colunas nomeadas para ilustrar onde cada item está no fluxo de trabalho.
- Limite o trabalho em progresso (WIP - work in progress) – associe limites explícitos para quantos itens podem estar em progresso em cada estado do fluxo de trabalho.

- Acompanhe o tempo de execução da tarefa (tempo médio para completar um item, algumas vezes chamado de “tempo de ciclo”), otimize o processo para tornar o tempo de execução o menor e mais previsível possível. Ver Figura 14 :

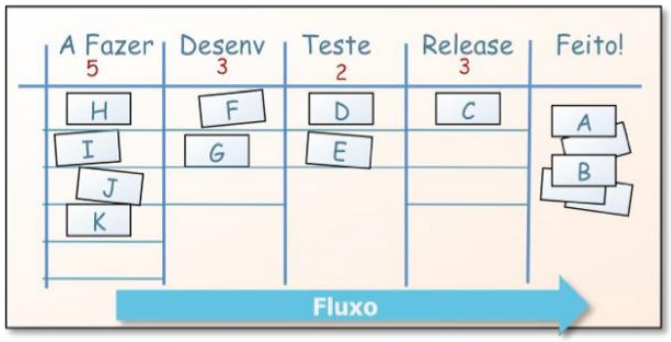


Figura 14 - Fluxo do Kanban (Kninberg, Skarin, 2009).

2.3 Scrum e Kanban

Scrum e *Kanban* são metodologias que podem ser complementares, e vale dar uma analisada nas suas similaridades e diferenças.

2.3.1 Similiaridades

Segundo Kninberg e Skarin,(2009) :

- Ambos são Lean e Agile.
- Ambos usam controle de cronograma.
- Ambos limitam atividades em andamento.
- Ambos usam transparência para direcionar a melhoria do processo.

- Ambos concentram-se na entrega de software que funcione, o mais rápido possível e freqüentemente.
- Ambos são baseados em equipes auto-organizáveis.
- Ambos exigem que o trabalho seja dividido em partes.
- Em ambos, o planejamento de release é continuamente otimizado, baseado em dados (velocidade / tempo de execução) empíricos.

2.3.2 Diferenças

O quadro abaixo, ver Tabela 2, resume as principais diferenças entre Scrum e Kanban mostrando que são ferramentas complementares e que pode coexistir e se serem adaptadas pra funcionarem juntas dependendo da característica do projeto. (Kninberg, Skarin, 2009)

Tabela 2 - Diferenças entre Scrum e Kanban(Kninberg, Skarin, 2009)

Scrum	**Kanban**
Iterações Timeboxed prescritas.	**Iterações Timeboxed opcionais.** Pode ter cadência separada para o planejamento, release e melhoria de processos. Pode ser orientada para eventos, em vez de timeboxed.
Equipe compromete-se a uma quantidade específica de trabalho para esta iteração.	**Compromisso opcional.**
Usa a **velocidade** como padrão métrico para o planejamento e melhoria de processos.	Usa o ***Lead time*** como padrão métrico para o planejamento e melhoria de processos.
Equipes multifuncionais prescritas.	Equipes multifuncionais opcionais. **Equipes de Especialistas autorizada**.
Os itens devem ser divididos para que possam ser concluídos dentro de 1 *sprint*	Nenhum tamanho especial de item é prescrito
Gráfico Burndown	Nenhum tipo específico de diagrama é prescrito
WIP limitado indiretamente (por sprint)	**WIP limitado diretamente** (por situação do fluxo de trabalho)
Estimativa prescrita	**Estimativa opcional**
Não poderá adicionar itens à iteração em uso	**Pode adicionar novos itens sempre que houver capacidade disponível**
O ***sprint backlog*** **é de uma equipe especifica**	**Quadros kanban podem ser compartilhados por várias**

Dessa análise comparativa é percebido que o *Kanban* é menos prescritivo que o Scrum e que se adapta melhor a contextos onde se tenham um fluxo mais frequente de funcionalidades a serem desenvolvidas. Em análises mais atuais , hoje o Kanban indica o uso de Gráficos CFD (Dagramas que mosram o fluxo acumulado) e as equipes multifuncionais são incentivadas

2.4 TDD (Test-Driven Development)

Desenvolvimento Guiado por Testes, tradução do termo em inglês *Test-Driven Development* (TDD), é uma das práticas sugeridas pela Programação Extrema (XP) A prática

é baseada em um ciclo, no qual o desenvolvedor escreve um teste antes de implementar a funcionalidade desejada e, depois, com o código passando no teste criado, refatora para remover possíveis duplicação de dados e de código (Aniche, 2012).

Simplicidade deve ser também algo intrínseco ao processo; o praticante busca escrever o teste mais simples que falhe e escrever a implementação mais simples que faça o teste passar. Esse ciclo é também conhecido como "Vermelho-Verde-Refatora"(ou "*Red-Green-Refactor*"), ver Figura 15, uma vez que lembra as cores que um desenvolvedor normalmente vê quando faz TDD: o vermelho significa que o teste está falhando, e o verde que o teste foi executado com sucesso(Beck,2004).

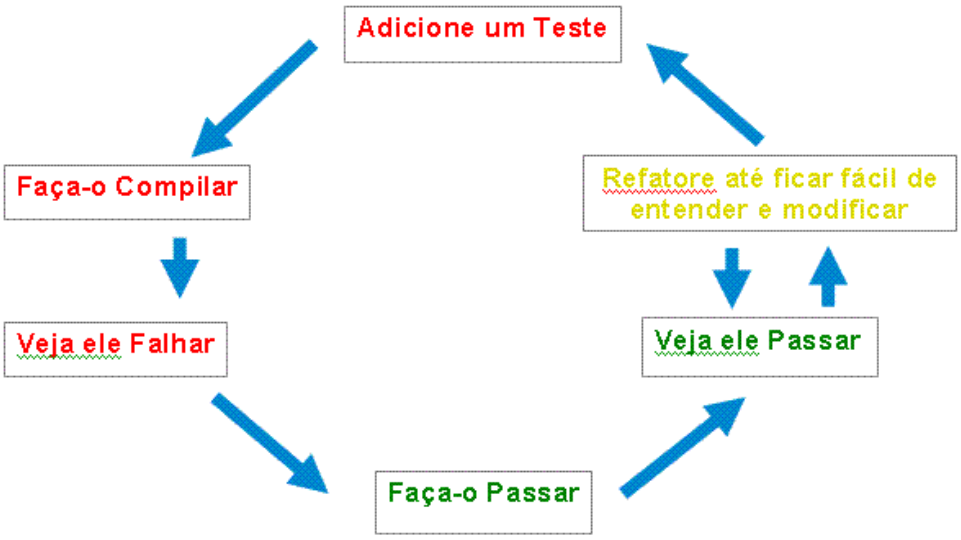


Figura 15 - Ciclo do TDD

Uma consequência da prática de TDD é a bateria de testes de unidade gerada. A prática ajuda o programador a evitar erros de regressão, em que a implementação de uma nova funcionalidade quebra uma outra funcionalidade já existente no sistema. Essa bateria também provê segurança durante as constantes refatorações de código que são feitas durante o processo de desenvolvimento. (Teles,2004). A quantidade de código coberto pelos testes também tende a ser alta, uma vez que o desenvolvedor deve sempre escrever um teste antes de implementar uma nova funcionalidade.(Aniche, 2012).

2.5 BDD (Behavior Driven Development)

Behavior Driven Development (BDD)ou Desenvolvimento Guiado por Comportamento (North, 2006) é uma técnica de especificação que está crescendo em aceitação nas comunidades ágeis. BDD permite com segurança verificar se todos os requisitos funcionais foram tratados adequadamente pelo código-fonte, ligando a descrição textual dos requisitos para os testes.

BDD coloca o foco sobre o comportamento em vez de estrutura, e faz isso em todos os níveis de desenvolvimento. Se estamos falando de um objeto calcular a distância entre duas cidades, ou uma tela de usuário voltada para fornecer feedback quando temos uma entrada inválida, tudo é o comportamento.(North, 2006).

Uma vez que reconhecemos isso, ele muda a forma como pensamos sobre a condução para fora do código. Começamos a pensar mais sobre as interações entre pessoas e sistemas, ou entre objetos, do que sobre a estrutura dos objetos. Obtendo as palavras certas acreditamos que a maioria dos problemas que equipes de desenvolvimento de software enfrentam são problemas de comunicação. BDD visa ajudar a comunicação, simplificando a linguagem que usamos para descrever cenários em que o software será utilizado (Chelimsky, 2010).

Dado algum contexto, quando algum evento ocorre, então eu espero que algum resultado. BDD, são simples palavras que usamos se estamos falando sobre o comportamento de uma aplicação ou comportamento de um objeto. Eles são facilmente compreendidas por analistas de negócios, testadores e desenvolvedores também (Chelimsky, 2010).

Podemos dizer que o BDD está fundamentado em 3 princípios:

- Negócio e Tecnologia deveriam “falar” sobre um sistema da mesma forma;
- Qualquer sistema deveria ter um valor identificável e verificável para o “negócio”;
- Análise, design e planejamento precoce tem, sempre, retorno questionável.

Por apoiar todos os envolvidos e reduzir riscos de desenvolvimento inadequado, a "venda da ideia" para BDD costuma ser mais fácil do que para TDD. Embora, por envolver mais gente, seja mais "trabalhoso" de implantar (Chelimsky, 2010).

BDD associa os benefícios de uma documentação formal, escrita e mantida pelo "negócio", com testes de unidade que "demonstram" que essa documentação é efetivamente válida(North, 2006).

Na prática, isso garante que a documentação deixa de ser um registro estático, que se converte em algo gradualmente ultrapassado, em um artefato "vivo" que reflete constantemente o estado atual de um projeto. (Chelimsky, 2010).

O BDD pode ser pensado como uma evolução do TDD , focando no comportamento do sistema somado a um desenvolvimento visando testes. As especificações criadas quando utilizado BDD usam a mesma linguagem onipresente como visto no domínio real, o qual pode ser benéfico tanto para usuários técnicos como para usuários de negócio.

Veja a Figura 16, onde temos um exemplo de BDD escrito em linguagem natural na língua portuguesa.

```
Funcionalidade: Efetuar login
  Como um usuário do solar
  Eu quero efetuar login
  Para acessar os recursos do sistema

@javascript
Cenário: Usuário já logado com step
  Dado que estou logado com o usuario "user" e com a senha "123456"
  E que estou em "Login"
    Então eu deverei visualizar "Unidade Curricular"

Cenário: Usuário não logado tenta acessar "Meu Solar"
  Dado que eu nao estou logado no sistema com usuario user
  E tento acessar "Meu Solar"
    Então eu deverei ver "Usuário"
    E eu deverei ver "Senha"

Esquema do Cenário: Login com usuários inválidos
  Dado que eu nao estou logado no sistema com usuario user
    E que estou em "Login"
    E preencho o campo "login" com "<login>"
    E preencho o campo "password" com "<password>"
  Quando eu clicar em "Entrar"
    Então eu deverei ver "<action>"
  Exemplos:
    | login        | password       | action                       |
    | unknown_user | any_password   | Usuário ou senha inválidos. |
    | user         | wrong_password | Usuário ou senha inválidos. |
```

Figura 16 - BDD na Feature de Login do Solar 2.0

2.6 A Programação em Par

Programação em par é uma das práticas mais conhecidas e mais polêmicas utilizadas pelos que adotam a programação Extrema(XP). Ela sugere que todo e qualquer código produzido seja sempre implementado por duas pessoas juntas, diante do mesmo computador, revezando-se no teclado.(Beck,2004)

A programação em par parece ser uma prática fadada ao fracasso e ao desperdício. Afinal, embora possa haver benefícios, temos a impressão de que ela irá consumir mais recursos ou irá elevar o tempo do desenvolvimento. Entretanto, não é exatamente isso o que ocorre. (Teles, 2004).

A programação em par dobra a capacidade mental disponível durante a codificação e da oportunidade de sempre os membros pensarem sobre assuntos estratégicos no desenvolvimento a longo prazo. (Shore e Warden, 2008).

A formação de pares reduz a incidência de bugs , melhora a qualidade do código e promove uma melhor interação e troca de conhecimentos entre os membros do time.(Shore e Warden, 2008)

A programaçãoo em par mantém ambos membros do par mais focados e concentrados em suas atividades. O nível de dispersão é bem menor do que quando se estar sozinho.(Cohn, 2010).

Segundo Shore e Warden, 2008 essas são algumas dicas para se ter uma programação em par com sucesso :

- Para tudo que precisar manter, faça pares.
- Permita que os pares se formem de maneira expontânea ao invés de impor a formação dos mesmos.
- Troque de par a medida que precisar de novas perspectivas.
- Evite fazer sempre par com a mesma pessoa.
- Sente-se de forma confortável lado a lado.
- Produza código através de conversa. Colabore mais e critique menos.
- Troque os papéis de piloto e navegador com frequência.

2.7 Modelagem Ágil

Segundo Ambler, 2004 , A modelagem ágil é uma técnica para melhorar a comunicação e compreensão em cima dos artefatos gerados em um projeto de software.

A Modelagem Ágil busca criar modelos simples usando ferramentas simples. A adoção da simplicidade é a chave para maximizar os resultados de um trabalho.

O foco é entregar software, não modelos. Modelos devem ser usados quando e onde adicionam valor. Se eles não agregam valor nem nos auxiliam no sentido de entregar software funcionando, então não devem ser utilizados.

Modelos devem ser mantidos pelo tempo necessário. Se um modelo serviu ao seu propósito e deixa de ser necessário, jogue fora. Isso permite manter a agilidade sem burocracia. Por outro lado, se seu modelo pode ainda ser útil, guarde ou recicle.

Existem diversos modelos diferentes de modelagem e dependendo do público e do cliente não existe um modelo ideal. A inspeção e adaptação é a melhor maneira de descobrir a melhor forma de modelar.

A Modelagem Ágil combina de modelos formais e informais conforme a situação, público-alvo e objetivos. Por exemplo, um modelo poderia ser composto de formas simples desenhadas a lápis ajudando o essencial de um sistema, ou utilizando diagramas detalhados de classes do UML.

Você diagrama geralmente para compreender. Para facilitar a comunicação entre o time para que se possa gerar resultados melhores.

Para valorizar pessoas e suas interações é preciso fortalecer a comunicação. Em vez de investir em novas ferramentas ou adotar processos prescritos. A utilização de métodos de modelagem facilita a compreensão do problema e da solução, além de melhorar a comunicação entre os *stakeholders* e resposta a mudanças.

2.8 Gestão 3.0

Segundo APELLO 2011, a Gestão 3.0, propõe uma nova abordagem para a gestão, mais alinhada aos métodos ágeis, e coerente com a teoria da complexidade.

"Para todo problema complexo há uma resposta clara, simples e errada".

H.L. Mencken, Jornalista e Escritor (1880–1956)

Segundo GOMES, 2013, a imensa dificuldade que temos em lidar com sistemas complexos se deve às nossas mentes que trabalham de forma linear e que preferem causalidade à complexidade. Nós procuramos propósito e causa em todas as coisas, buscamos por causa e efeito em tudo. Estamos acostumados a estudar e aprender de forma linear, a contar histórias, e é dessa forma que enxergamos o mundo, como um lugar cheio de eventos fáceis de serem explicados através de simples causas e efeitos. Gerald Weinberg chamou isso de Falácia da Causalidade.

Tanto a gestão ágil quanto os AVAs tem suas bases na complexidade, por isso, o foco da gestão ágil e do desenvolvimento de AVAs deve estar na adaptabilidade ao invés de previsibilidade, uma vez que é aceito que muitos fatores em projetos são imprevisíveis. (Gomes, 2013).

A Gestão 3.0 é focada na complexidade, e encara as organizações como redes, e não como hierarquias; e nelas as pessoas e seus relacionamentos devem estar no foco da gestão, mais do que os departamentos e seus lucros. (Apello, 2011).

Esse tipo de gerenciamento sugere que a gestão trabalhe com seis visões, Ver Figura 17. Fazendo uma metáfora de como se as organizações parecessem monstros.(Apello, 2011). E esse mosntro chamado Martie segundo Apello, 2011 tem 6 visões :

1. **Energizar pessoas** : Como as pessoas são as partes mais importante da organização, seus gestores devem fazer o máximo de esforço para mantê-las criativas, ativas e motivadas. Para isso deve- se conhecer a diferença entre motivação intrínseca e extrínseca e desenvolver e conhecer técnicas para motivar as pessoas em sua volta.(Sabbagh, 2013).

   2. Empoderar times : Auto – Organização é um valor de extrema importância para times ágeis mas para isso precisam empoderamento, autorização e confiança da gestão.(Gomes , 2013). Empoderar não é algo binário, tratar o empoderamento assim aliena as outras pessoas que não adquirem as responsabilidades perante os problemas das organizações. Segundo APELLO, 2011,

ao Gestor 3.0 cabe delegar o nível certo de empoderamento para cada pessoa:

A - Contar (Nível A): O gestor toma a decisão e anuncia às pessoas.

B – Vender(Nível B): O Gestor toma a decisão mas se esforça para ganhar o compromisso das pessoas vendendo suas idéias para elas.

C - Consultar (Nível C): O Gestor só toma uma decisão após ouvir a opinião das pessoas.

D - Concordar (Nível D) : Nesse nível a voz do gestor é igual a dos outros. Ele convida o time para uma discussão e chega em um consenso sobre a decisão a ser tomada junto com elas.

E – Aconselhar (Nivel E) : O Gestor tenta influenciar as pessoas mas deixa a elas a responsabilidade de tomar a decisão.

F - Questionar (Nível F) : O Gestor permite que o time decida , mas depois pedem para que vendam a decisão para ele para poder valida-la.

G – Delegar (Nível G) : O Gestor permite que o time tome decisões sem envolvimentos da sua parte.

Além de do nível de delegação é importante determinar também de acordo com cada atividade os níveis de delegação e de quem são as responsabilidades.

3. Alinhar Restrições : Os gestores devem entender as diferenças entre restrições e regras de forma que as pessoas possam trabalhar de forma auto- organizada em direção aos objetivos colocados pela organização representados pelo gestor. Definir metas para que os resultados possam ser obtidos é uma fato importante nesse caso. A auto-organização pode levar a qualquer resultado, positivo ou negativo, faz parte do papel do gestor dar às pessoas uma meta compartilhada para que essas possam se auto- organizarem de uma forma que produzam valor para a organização. O objetivo desta

meta compartilhada é dar às pessoas uma direção a seguir.(Apello, 2013).

4. Desenvolver Competências : os times só conseguem desenvolver suas metas se seus membros forem capazes o suficiente. Um bom modelo de gestão deve contribuir para que os times maximizem suas habilidades. Um time que esteja em constante melhoria contínua, elevando a excelência técnica e tendo sempre desafios a serem superados que os forcem sempre a estarem adquirindo novas competências.(Apello, 2011).

5. Estruturar : Muitos times trabalham dentro de um contexto organizacional complexo, por- tanto é importante considerar estruturas que promovam a comunicação(Sabbagh, 2013). Como dividir as pessoas em equipes? Quais as melhores formas de Unir as pessoas de acordo com suas funções ou formar equipes que entreguem o produto do início ao fim? Como esses times poderão se comunicar, diretamente ou por intermédio de alguém? Qual a quantidade ideal de membros em um time ágil? Todas essas são questões relevantes em se tratando de estruturar a organização.(Gomes, 2013)

6. Melhorar Tudo : Melhorar de forma contínua é um dos maiores desafios das organizações . Pessoas times e organizações precisam melhorar de forma contínua.(Sabbagh, 2013). Agilidade tem a ver com Melhoria Contínua. E nada melhora se permanecer como está. As coisas melhoram quando mudam. Então, para melhorar, pessoas, times, e organizações precisam melhorar continuamente para evitar falhas e evoluir sempre.(Gomes, 2013). A Gestão 3.0 traz uma abordagem sistêmica para Gestão 3.0 que segundo APELLO, 2012, consiste em :

A) Considerar o sistema: Uma rede social complexa e adaptativa.Ela vai se adaptar às suas ações, e você também deve se adaptar à rede social.

B) Considerar os indivíduos: Saiba que as pessoas são a parte crucial do sistema social, e que elas diferem umas das outras. Não há uma abordagem única que sirva para todos. Simplesmente pedir para que as pessoas mudem, raramente, é suficiente. A diversidade é o que faz sistemas complexos funcionarem, e é por isso que trabalhar com uma diversidade de métodos é crucial para lidar com as pessoas.

C) Considerar as interações: Um comportamento se espalha através de um sistema complexo. Em uma rede social, tudo gira em torno de indivíduos e das interações entre eles. Comportamentos se espalham de forma viral, e estimular a rede social pode ajudar a se vencer a resistência às mudanças e a transformar uma organiza- ção por completo.

D) Considerar o ambiente: As pessoas sempre se organizam dentro do contexto de um ambiente. O ambiente determina como o sistema pode se auto-organizar, e você deve ser capaz de ajustar o ambiente. Isso porque o comportamento das pessoas depende do ambiente, e se você mudar o ambiente, você também muda as pessoas.

Saber aplicar as mudanças na hora certa e saber motivar as pessoas a mudarem em busca de melhoria contínua é um fator que pode fazer a diferença entre o sucesso e o fracasso de uma organização.

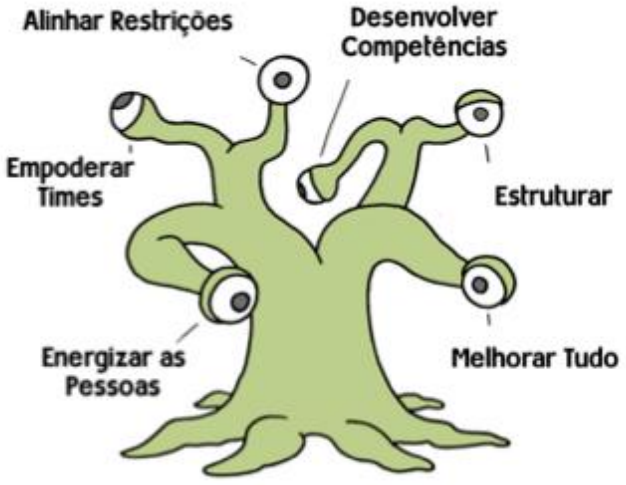


Figura 17 - Martie, o Modelo de Gestão 3.0(Gomes, 2013).

3 AMBIENTES VIRTUAIS DE APRENDIZAGEM

Os avanços tecnológicos vem ao longo dos anos transformando e impulsionando a maneira de ensinar e aprender. Além disso o intenso ritmo do mundo globalizado e a crescente complexidade das tarefas que envolvem informação e tecnologia (Pereira e Scmith , 2007).

Nos últimos anos os Ambientes Virtuais de Aprendizagem (AVA) estão cada vez sendo mais usados no âmbito acadêmico e coorporativo como uma opção tecnológica para atender essa demanda educacional. Diante disso se destaca a importância de se ter um entendimento mais crítico sobre o conceito que orienta o desenvolvimento e o uso desses ambientes, assim como o tipo de estrutura humana e tecnológica que da suporte ao processo de ensino-aprendizagem (Pereira e Scmith , 2007).

Os AVA sendo assim tornaram-se softwares de grande complexidade com muitas interações particularidades tanto nas suas características como na forma de desenvolvê-lo.

A utilização crescente das tecnologias de informação e comunicação tem contribuído para a disseminação de ambientes virtuais de aprendizagem. Ambientes Virtuais de Aprendizagem são softwares desenvolvidos para o gerenciamento da aprendizagem. Segundo Britain,1999 são sistemas que sintetizam a funcionalidade de software para comunicação mediada por computador e métodos de entrega do material de cursos online. Atualmente, inúmeras iniciativas, tanto de instituições educacionais como de empresas comerciais, têm disponibilizado ambientes e ferramentas para suporte a aprendizagem através da web. Muitos destes ambientes possuem um conjunto similar de características e funcionalidades, tais como ferramentas para gerenciamento de cursos, alunos e professores; e ferramentas de comunicação síncrona e assíncrona.

Os Ambientes Virtuais de Aprendizagem (AVA) integram tecnologias de Informação e Comunicação com a finalidade de criar um ambiente baseado na Internet que possibilite o processo de construção de conhecimento e autonomia por parte de seus interagentes (Castro Filho, 2005). Sua principal utilização é na modalidade de Educação chamada Educação a Distância, onde os diversos partícipes do processo de aprendizagem estão distantes geograficamente e/ou temporalmente, e se utilizam de meios de comunicação para a interação entre os mesmos.

3.1 Visão Geral dos Ambientes Virtuais de Aprendizagem

Os Ambientes Virtuais de Aprendizagem remotam da segunda metade da década de 90, quando apareceram softwares como o WebCT e o AulaNet (Sarmento, 2007). Daí para frente houveram uma diversidade de ambientes sendo criados para atender tanto a escopos particulares de determinadas instituições de ensino, quanto de uso geral.

Um ponto em comum a maioria dos atuais AVA é seus aspecto modular de criação, que permite a incorporação de novos recursos através de Application Programming Interface. Desta forma, as instituições que adotam estes ambientes podem personalizá-los e incorporar novas funcionalidades não previstas em seu projeto inicial. outra característica. Outra característica marcante dos atuais ambientes virtuais é a inserção de ferramentas de colaboração e compartilhamento de informações típicas da Web 2.0 - nome dado a World Wide Web que passa a adotar características autorais, colaborativas e personalizadas em suas aplicações.

Nesta seção serão vistos alguns ambientes existentes e que são de uso geral.

3.1.1 aTutor

O Ambiente Virtual de Aprendizagem aTutor (Figura 18) é uma ferramenta de gerência de conteúdo, suporte a redes sociais e tecnologias de Acessibilidade. Este ambiente encontra-se na versão 1.6.3, com suporte multilíngue, contudo, o português ainda se encontra em 86% de desenvolvimento.

O ambiente possui recursos de comunicação síncronos e assíncronos, publicação de conteúdos, avaliação, gerência de usuários, colaboração e personalização de interface com o usuário.

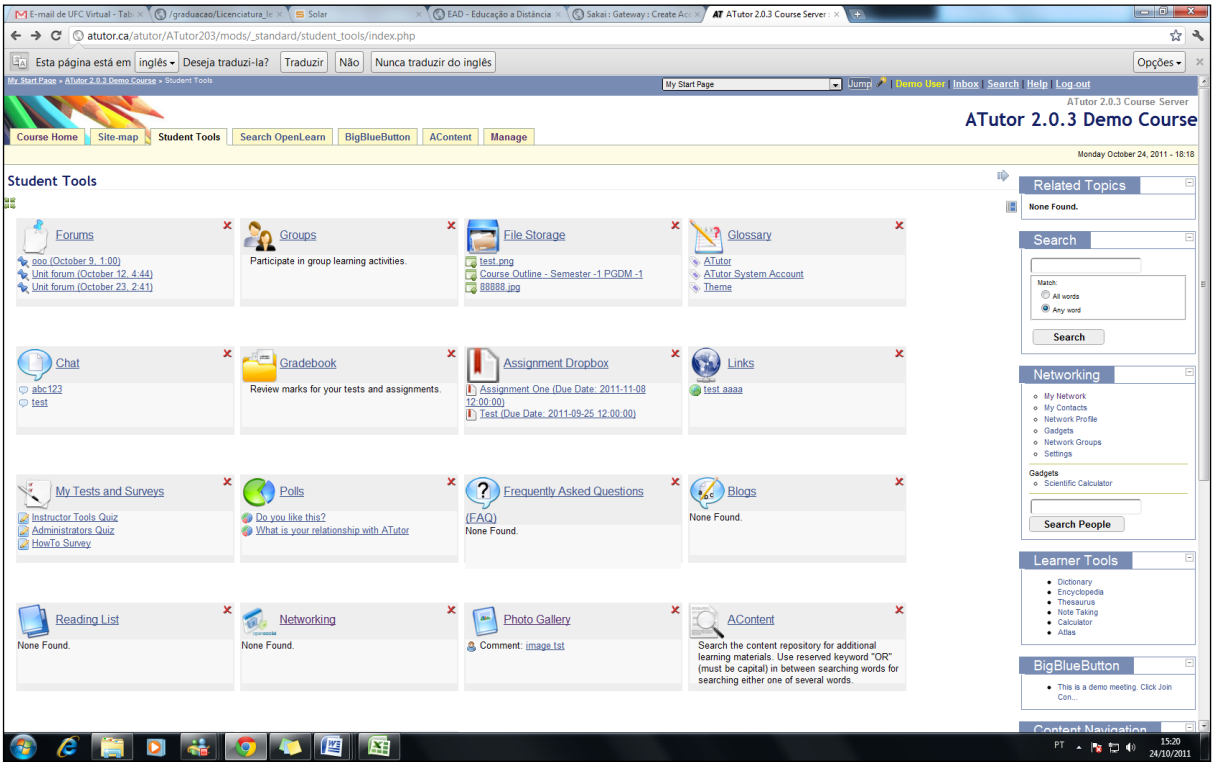


Figura 18 - aTutor

3.1.2 Claroline

O ambiente virtual Claroline (Figura 19) é uma plataforma *e-Learning* e *e-Working* que possibilita a criação de cursos *on-line* e a gerência das atividades de aprendizagem e de colaboração na Web, tendo sido traduzido para 35 idiomas. Ele foi criado pela Universidade Católica de Louvain, na Bélgica, no ano 2000 e atualmente é mantido pelo Claroline Consortium. Este ambiente virtual encontra-se, no momento de escrita deste artigo, na versão 1.9.2.

O Claroline possui recursos de Agenda, Anúncios, Portfólio, Exercícios, Fórum, Grupos, ferramenta para criação de Wiki, e Chat – sendo este a única ferramenta de comunicação síncrona possível. Ele também oferece uma ferramenta chamada *Learning Path*, que possibilita ao aluno verificar o seu progresso em um curso *on-line*.

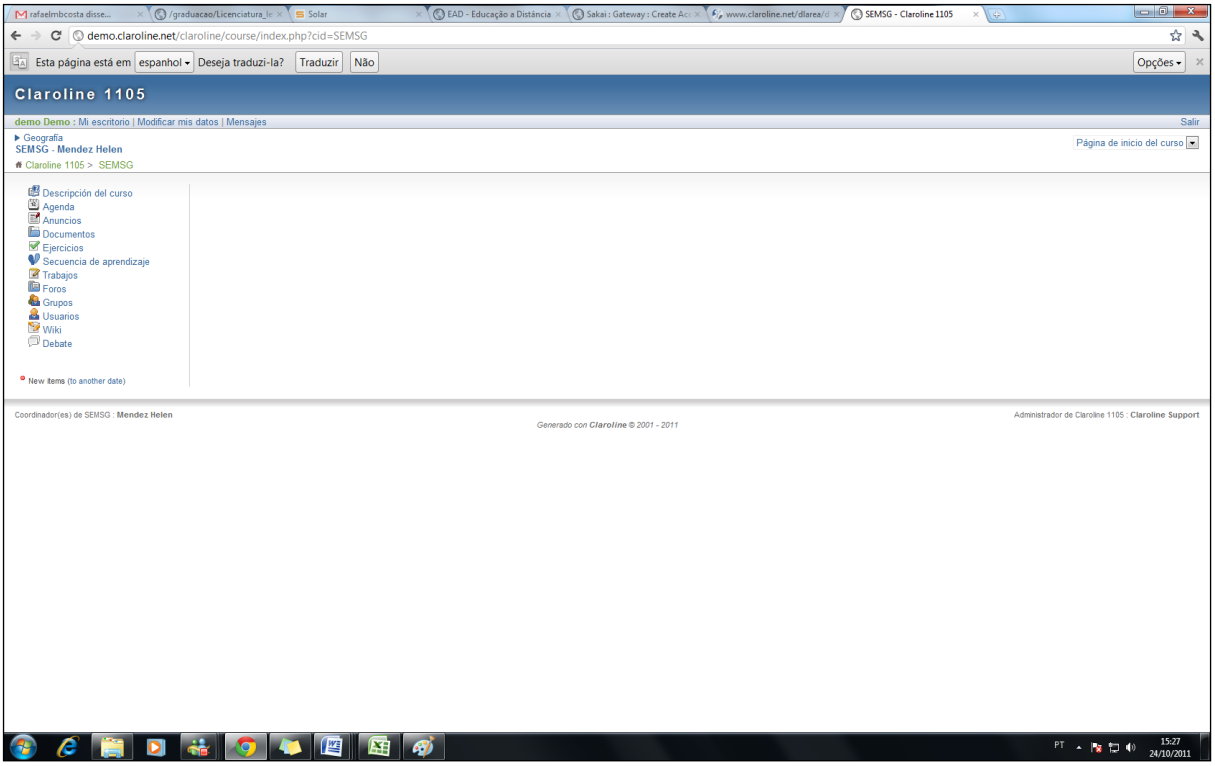


Figura 19 - Ambiente Virtual Claroline

3.1.3 OLAT

O *Online Learning And Training* (OLAT) , ver Figura 20 é um ambiente *Open Source* baseado em Java, criado em 1999 pela Universidade de Zurique, na Suiça, que tem como finalidade a gerência de aprendizagem. O OLAT encontra-se na versão 6.1.1 e está disponível nos idiomas alemão, inglês, francês, italiano, espanhol, português, russo, tcheco, polonês, lituano, grego, chinês, farsi, dentre outros. Este AVA permite que um aluno acesse

Chat, envie mensagens instantâneas e possa criar uma videoconferência utilizando uma ferramenta externa chamada iChat. Além destas ferramentas, têm-se a disposição Web-Fórum, mensagem de correio, notícias, informativos de atividade do Curso (atualizações no chat, fórum etc.), Agenda e informativo de Progresso de curso, além de espaço para notícias. O aluno pode, ainda, publicar aulas, inserir arquivos no Material de Apoio, podendo ter seu Portfólio pessoal, Glossário para eventuais consultas, e acesso a ferramenta de construção de Wiki, que permite também exporta as páginas criadas. Ele também pode criar uma página pessoal e personalizá-la de maneira análoga ao Orkut. O participante pode personalizar a interface do ambiente mudando a disposição das ferramentas e selecionar o idioma que preferir.

O professor pode criar provas e exercícios on-line e pode receber o resultado de suas avaliações e exercícios em uma planilha *on-line* que é armazenada em seu portfólio pessoal. Ele também pode inserir Enquetes utilizando uma ferramenta baseada na tecnologia AJAX.

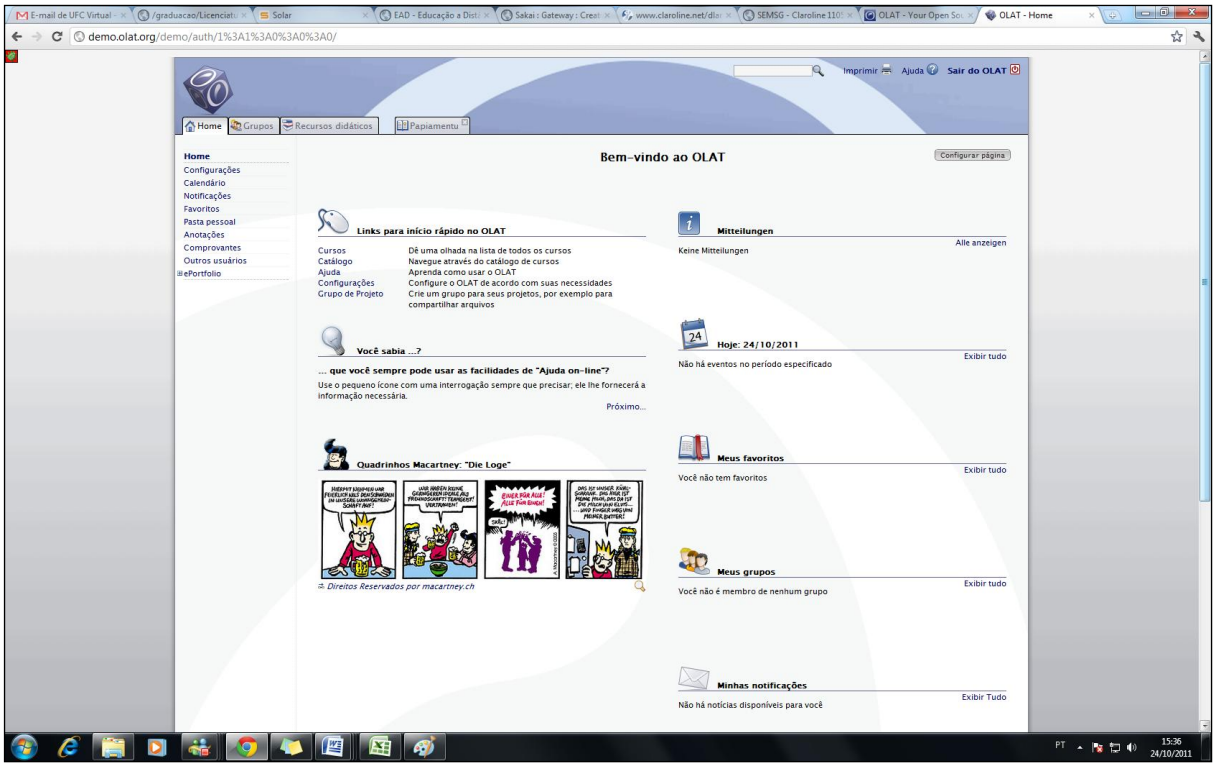


Figura 20 - Ambiente OLAT

3.1.4 eFront

O eFront (Figura 21) é um ambiente feito em PHP pela empresa grega Epignosis com base de dados MySql. Ele é compatível com SCORM, permitindo importar e exportar conteúdo, e é open source desde março de 2007. Sua última versão para download é a 3.5.4.

O ambiente possui uma aplicação básica, permitindo inserir módulos disponibilizados pelo site da própria empresa, como suporte ao Google's translation API e RSS, assim como personalizar módulos diretamente no código fonte. Permite também o uso de tradução, tanto automática quanto manual, disponibilizando pacotes de tradução em seu site.

Entre seus recursos, estão o gerenciamento de usuários, edição de conteúdo, indicadores de progresso e ferramentas de comunicação, como fórum, chat e envio de mensagens.

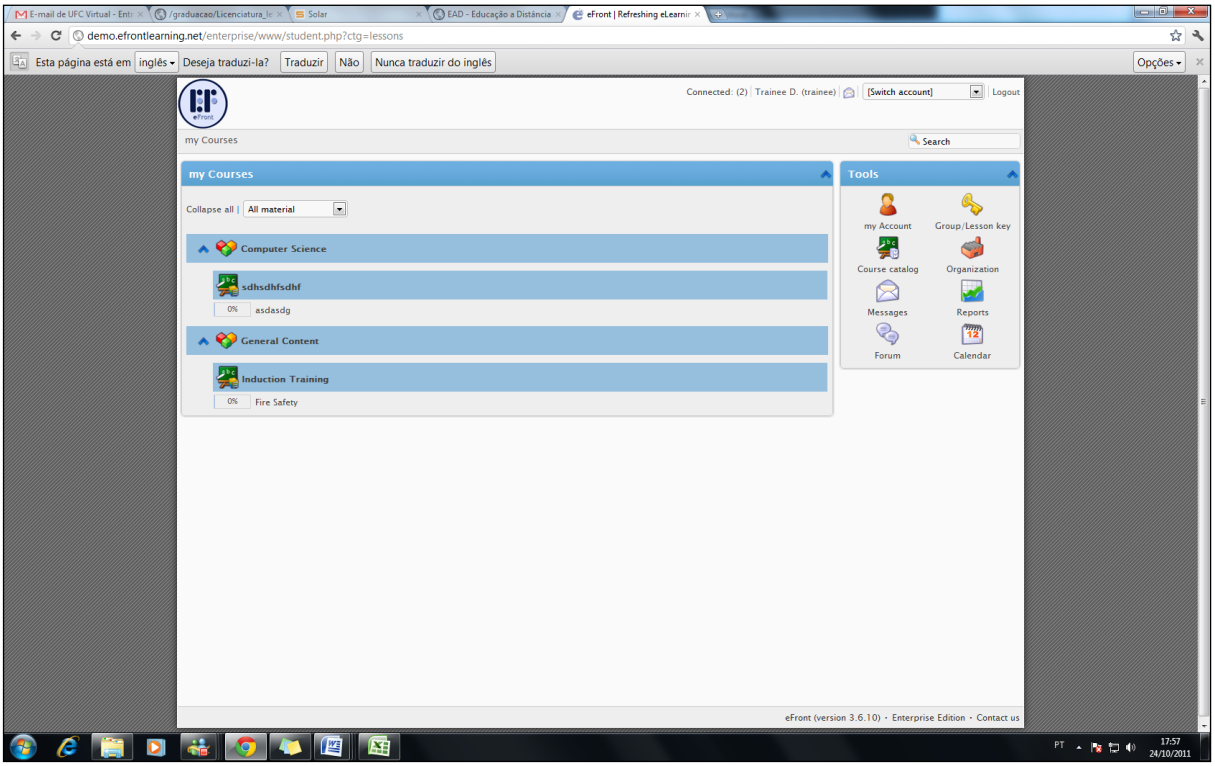


Figura 21 - Ambiente eFront

3.1.5 MOODLE

O AVA *Open Source* Moodle (*Modular Object Oriented Distance Learning* - Objeto Modular Orientado ao Ensino a Distância), ver Figura 22, é um projeto desenvolvido em PHP para dar suporte a uma abordagem sócio-construtivista.

O Moodle possui uma série de recursos para o desenvolvimento das atividades, como ferramentas de avaliação do curso, fórum, chat, pesquisa de opinião, tarefa, trabalho com revisão, wiki, além de dar suporte ao padrão SCORM - Sharable Courseware Object Reference Model.

Figura 22 - Ambiente Moodle

3.1.6 Sakai

Sakai CLE - *Collaboration and Learning Environment* (Figura 23)- é um sistema de gerenciamento de cursos, que suporta tanto o ensino-aprendizagem, quanto a colaboração administrativa. Ele é Open Source e permite ao usuário uma personalização do ambiente de acordo com suas necessidades.

O Sakai CLE apresenta um conjunto de ferramentas divididas em ferramentas de colaboração, como wiki, blog, calendário, chat; ferramentas de ensino-aprendizagem, como construção de material, testes e compartilhamento de material entre os participantes; ferramentas de portfólio dos participantes; e, por último, ferramentas de administração, como gerenciamento de usuário e acompanhamento de uso dos recursos do sistema.

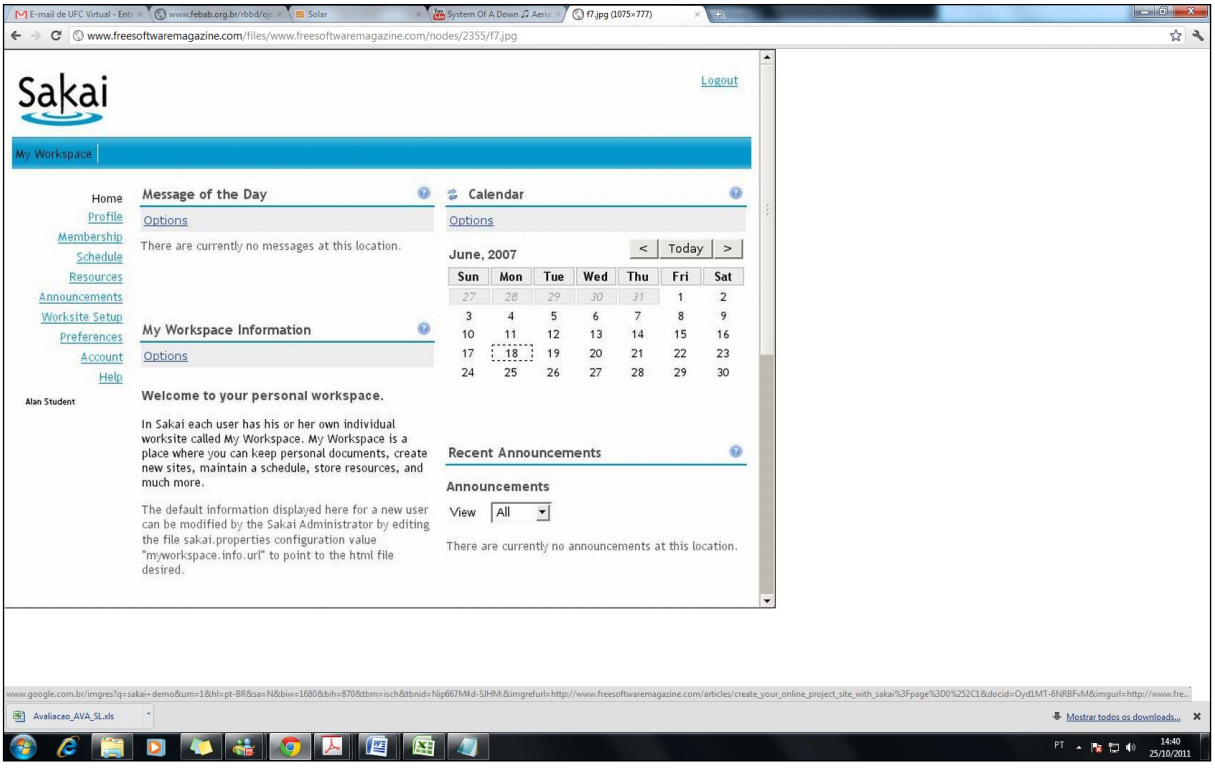


Figura 23 - Ambiente Sakai

3.1.7 TelEduc

O TelEduc (Figura 24) é um ambiente *Open Source* brasileiro para a criação, participação e administração de cursos na Web. Seu uso é de fácil assimilação e possui um conjunto enxuto de funcionalidades.

Possui ferramentas de comunicação, como Correio Eletrônico, Grupos de Discussão, Mural, Portfólio, Diário de Bordo, Bate-Papo etc., além de ferramentas gerenciais que permitem consulta às informações geradas em um curso, como as ferramentas Intermap e Acessos. Seu foco central são as atividades a serem desenvolvidas com os alunos - atividades essas embasadas pelas ferramentas de Material de Apoio, Leituras, Perguntas Frequentes, etc.

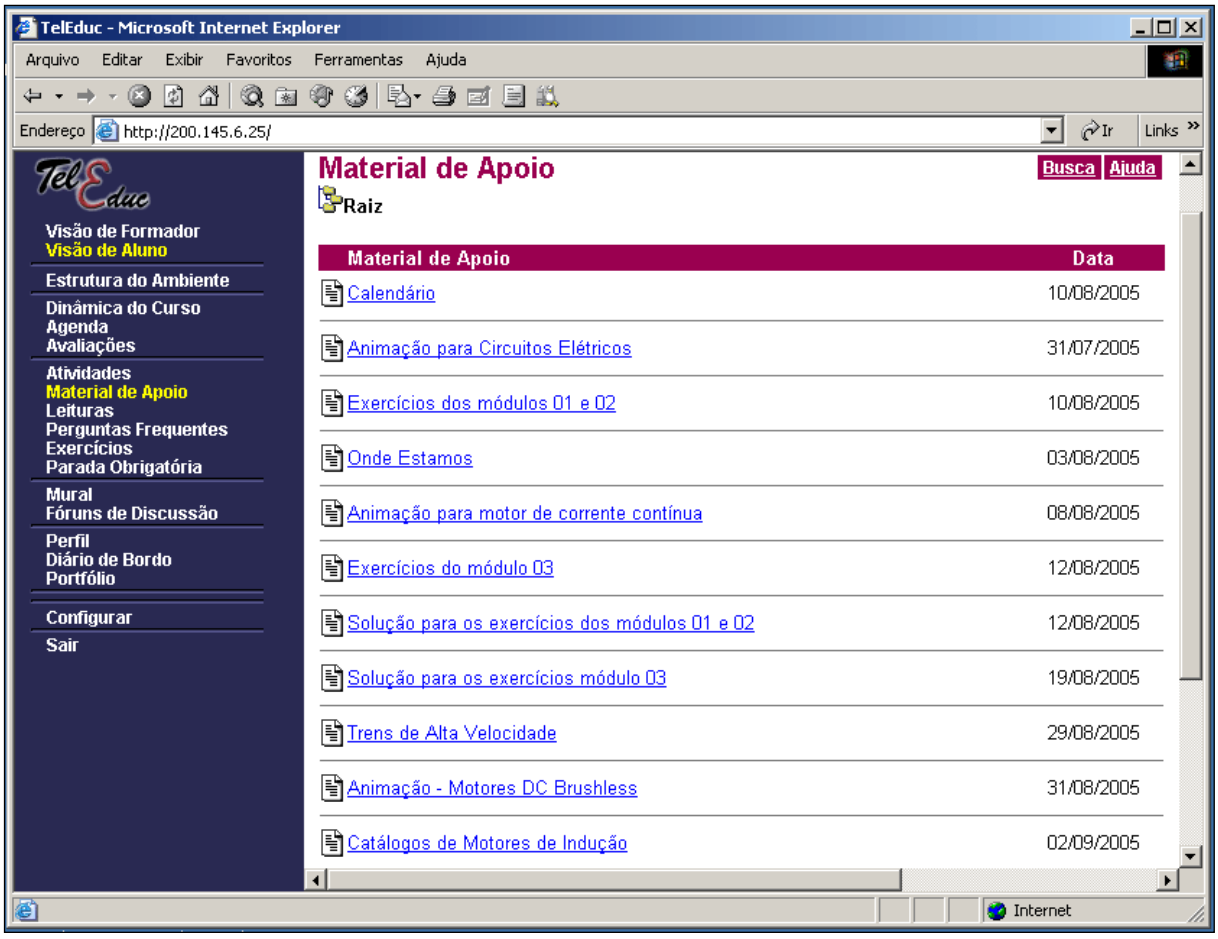


Figura 24 - Ambiente Teleduc

3.1.8 Amadeus

O Amadeus (Figura 25) - Agentes Micromundos e Análise do Desenvolvimento no Uso de Instrumentos - é um sistema de gestão do aprendizado (ou LMS, do inglês *Learning Management System*) de segunda geração Open Source. Desenvolvido pela Universidade Federal de Pernambuco, é baseado no conceito de *blended learning*, no qual os estilos de interação entre os usuários são estendidos.

Possui suporte multi-idioma, suporte ao padrão SCORM, assim como ferramentas de gestão de conteúdo. Também permite que ambientes colaborativos síncronos de interação em tempo real entre alunos e professores (micromundos) sejam usadas como materiais e as atividades realizadas nas mesmas sejam percebidas no LMS. Além disso, viabiliza o recebimento de mensagens e a visão do conteúdo de fóruns e demais atividades por meio de celulares com o objetivo de criar novos mecanismos de engajamento e prover ferramentas para uma melhor gestão de turmas.

Figura 25 - Ambiente Amadeus

3.1.9 SOLAR

A fim de solucionar este problema e criar um ambiente virtual híbrido quanto ao modelo de interação, foi criado o SOLAR (2005) pela Universidade Federal do Ceará, um ambiente que privilegia formas de interação e não de controle.

O SOLAR possui três componentes que interagem entre si, formando a aplicação servidora. A Figura 26 mostra os componentes básicos do sistema – Servidor Web, no caso, Microsoft Internet Information Services; Componentes de Software (na forma de *Dynamic-Link Library* e componentes DCOM[1]) e Sistema Gerente de Base de Dados Microsoft SQL Server –, em uma visão geral. Deve ser ressaltado o fato destes três componentes principais do servidor poder estar sendo executados em diferentes computadores, distribuindo a carga de processamento que incidiria sobre o ambiente SOLAR. A tecnologia de composição das páginas dinâmicas do sistema utiliza, principalmente, a tecnologia Active Server Pages (ASP.Net), porém, módulos como o Chat foram desenvolvidos em ASP.Net. Este ambiente também possui um módulo de suporte a Webconference baseado no sistema Dim-Dim de código aberto, bem como uma ferramenta para criação de questionários e enquetes utilizadas no processo de avaliação dos cursos que foi desenvolvida usando as tecnologias Java para Web (e.g. Java Server Faces, Hibernate, etc.).

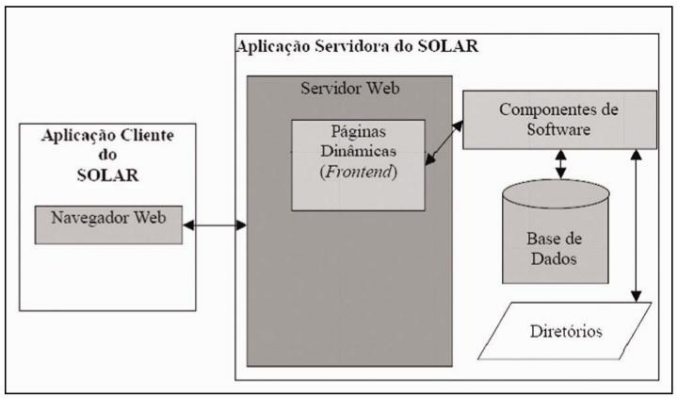


Figura 26 - Arquitetura do Ambiente SOLAR(pequeno et al., 2014 apud Sarmento,2007).

[1] *Distributed Common Object Model*

4 ESTUDO DE CASO: O PROJETO SOLAR 2.0

Esse capítulo detalha como foi feito o estudo de caso do projeto SOLAR 2.0 utilizando métodos ágeis para o desenvolvimento do mesmo e as etapas que seguimos desde o início até a entrega da primeira release do sistema.

4.1 Visão Geral do Sistema SOLAR 2.0

O ambiente Solar 2.0 está sendo projetado para permitir a criação de um espaço virtual que poderá ser usado por cursos presenciais ou semipresenciais, servindo como ponto de convergência para a criação do que se está chamando de *Blended Education*, ou seja, a mescla de características de ambas as modalidades de Educação, anteriormente citadas, para formação de um novo modelo educacional que utiliza fortemente as TIC. Na sua concepção inicial foi feito um levantamento de requisitos antes mesmo do projeto começar onde de forma tradicional foram definidos alguns módulos com suas características que o sistema iria conter. Nesse levantamento inicial o AVA era composto dos seguintes módulos:

- ***Core* de Integração de Módulos**: parte fundamental da arquitetura do sistema que será responsável pela comunicação entre os módulos componentes do AVA e pelo processo de integração destes. A criação de uma plataforma modular permite que novas funcionalidades sejam incorporadas através de novos módulos, ou atualização de existentes;
- **Gerência de Usuários e Controle de Acesso**: módulo responsável pelo cadastro de usuário, controle de senhas, criptografia de informações, controle de acesso do usuário, controle de permissões por perfil, dentre outras funcionalidades;
- **Integrador do AVA com Sistema Integrado de Informação Institucional (SI3)**: módulo responsável pela integração entre o sistema de controle

acadêmico do *software* SI3, utilizado pela Universidade Federal do Ceará, e o AVA SOLAR;

- **Espaço Pessoal do Interagente (meuSOLAR):** módulo de controle do espaço individual de cada usuário, que deverá conter suas atividades, agenda, aplicação de identificação e comunicação com amigos, disciplinas/cursos que está realizando etc.;
- **Avaliação e Acompanhamento:** módulo responsável pelo acompanhamento das atividades e interações dos participantes de uma disciplina ou curso, bem como do processo de Avaliação dos cursistas, tutores e professores que atuam no sistema;
- **Ferramenta de Autoria para Conteúdos:** permite a editoração de aulas, estruturação de conteúdo didático e publicação de aulas desenvolvidas através desta ferramenta;
- **Ferramentas de Comunicação:** criação das ferramentas de Web Fórum, Chat, Mensagens Instantâneas, Avisos, E-mail, Web Conference etc.;
- **Ferramentas de Conteúdo**: ferramentas que possibilitem Publicação de Aulas (desenvolvidos externos ao ambiente), Cadastro de Objetos de Aprendizagem e Materiais Didáticos, Busca de Conteúdos, Importação/Exportação de Material didático e Aulas em Formatos Padronizados Internacionalmente (ex. SCORM) etc.;
- **Ferramentas Administrativas:** módulo responsável pela geração de Relatórios do Sistema e dos interagentes, Criação de Perfis etc. ;
- **Ferramentas de Curso**: módulo responsável pela Criação e Estruturação de Cursos; Alocação de Alunos, Criação de Turmas, etc.;
- **Ferramentas Colaborativas e cooperativas:** permitem a autoria de documentos multimídia, bem como o desenvolvimento de trabalhos e atividades, de forma colaborativa e cooperativa (ex. Wiki, Blog, Grupos de Trabalho).

A fim de que se possa medir a eficiência e a eficácia dos *Métodos Ágeis* no desenvolvimento dos ambientes virtuais de aprendizagem , foi escolhido o AVA Solar 2.0 para estudo de caso, sendo adotado um modelo aqui proposto baseado em Metodologias Ágeis, onde um dos fatores importantes é o comprometimento do grupo de desenvolvimento

com os princípios do Manifesto Ágil. O projeto foi desenvolvido de forma iterativa e incremental e dentro das iterações foram usados, levantamento de requisitos ágeis, modelagem ágil, gerenciamento do projeto com *Scrum* e adoção de práticas de *Extreme Programming* e *Kanban*. O Solar 2.0 foi construído todo iterativo e incremental e possuí uma arquitetura modular.

A necessidade de desenvolver e documentar o processo de desenvolvimento de AVA surgiu pela ausência de documentação e artigos que descrevam o passo a passo de como foram desenvolvidos os AVAs e as dificuldades e boas práticas para desenvolvê-los.

O primeiro passo para o projeto foi a adoção do *Scrum* que já ocorreu no início do projeto(março de 2010) para organizar o ambiente de desenvolvimento e definir cerimônias e regras utilizadas. O *Scrum* é um *framework* para gerenciamento de projetos e se propõe a entregar o máximo de valor no menor tempo possível (Schwaber e Sutherland, 2010).

O *Scrum,* que é fundamentado na teoria de controle de processos empíricos, emprega uma abordagem iterativa e incremental para otimizar a produtividade e controlar riscos, ele não é um método de engenharia de *software*,e não vai te dizer como programar ou fazer testes (Schwaber, 2004). Scrum pode ser usado em conjunto com outras metodologias como a *Extreme Programming* (Keith, 2010).

Para o desenvolvimento e práticas de engenharia de *software* foi usada algumas práticas de Programação Extrema (*Extreme Programming* ou XP).Seu objetivo é a excelência no desenvolvimento de *software*, visando baixo custo, poucos defeitos, alta produtividade e um alto retorno sobre o investimento.(Sato,2007).Segundo Beck e Andress (2004), o XP apresenta um conjunto de práticas, que impactam diretamente na qualidade do código. Dentre as práticas relacionadas temos:

- Programação Orientada a Testes (TDD): Um dos aspectos mais importantes de XP é a ênfase nos testes automatizados. Essa prática sugere que os testes sejam escritos antes do código, trazendo benefícios como: escopo bem definido, menor acoplamento maior coesão, confiança do programador e ritmo de codificação. (Aniche, 2012)
- Integração continua: O novo código é integrado com o sistema atual após algumas horas. Quando integrado, o sistema é re-construído por completo, e

devendo passar em todos os testes, caso contrário, as mudanças são descartadas.(Gomes, 2013).

- Programação Pareada: Os desenvolvedores trabalham em par para realizar suas tarefas. Isso promove o trabalho coletivo e colaborativo, une a equipe e melhora a Comunicação e a qualidade do código.(Gomes, 2013).

Além das práticas do XP no solar se usou Modelagem Ágil e BDD.

A proposta do Solar(Figura 27) é ter uma software mais limpo, rápido incorporando os conceitos de web 2.0 e integrado as redes sociais facilitando assim a interação com o usuário.

Figura 27 - Tela de Login do Solar 2.0

4.2 Descrição dos Procedimentos e Metodologias Utilizados no Processo de Desenvolvimento do SOLAR 2.0

4.2.1 Desafios Encontrados Antes de Começar

Segundo Cohn 2010 , os desafios maiores que as equipes Scrum irão enfrentar são os seguintes:

- Aprender novas habilidades técnicas. É comum para os novos desenvolvedores de Scrum descobrirem que, enquanto eles ainda são bons em seus trabalhos, eles ainda não são bons em ser ágil. Eles terão que desenvolver habilidades que não tinham anteriormente (ou poderiam justificar ignorando essas habilidades). Por exemplo, os programadores terão que aprender a desenvolver o design de um sistema. Testers muitas vezes precisam aprender a testar um sistema sem a confiança tanto na documentação. Ambos geralmente precisam aprender novas formas de automatizar os testes.
- Aprender a pensar e trabalhar como uma equipe. A maioria dos times gosta de trabalhar silenciosamente em um cubículo, com fones de ouvido e com a menor interação possível com o time. "Você desenvolve sua parte, eu vou desenvolver a minha. "Equipes Scrum não são incentivados a pensar em termos de minhas tarefas e as suas tarefas , mas de nossas tarefas.. Trabalhando desta forma também cria-se uma mentalidade de responsabilidade compartilhada, que será nova para muitos membros da equipe.
- Aprender a criar software que trabalham dentro timeboxes curtos. Scrum é muito curto, concentrado, sprints apresentam desafios significativos para a maioria das equipes que são novas para trabalhar dessa forma. Equipes Scrum trabalham para evitar handoffs desnecessários de um membro da equipe de especialistas para o outro. O Desenvolvimento de software, até ao final de

cada sprint vai desafiar os membros da equipe para encontrar maneiras de eliminar desperdício, e trabalhar mais estreitamente com os outros.

Encontrados esses desafios pensou-se em uma maneira de desenvolver capacidades do time para superá-los e foram selecionadas algumas técnicas para isso. A primeira capacidade que deve ser adquirida é a de se tornar ágil e se tornar bom nisso e o maior desafio de adquirir essa capacidade e a de conscientização dos membros do time e o desejo deles para que isso ocorra.

Fornecer treinamento e formação *Scrum* é diferente de desenvolvimento de software tradicional em que o treinamento junto com o treinamento no local ou *mentoring* é normalmente exigido. "Nossa capacidade de ser bem sucedido com agilidade começou com um processo educativo. Na minha opinião, isso foi fundamental. Se não entender alguma coisa, não poderia recebê-lo de braços abertos. Iniciamos nossa transformação ágil, definindo uma meta de realizar um curso de dois dias onde nesses dias foram passados os fundamentos e a razão para começarmos com *Scrum*. Era interessante que o time acreditasse que que aquilo iria funcionar. Isso foi importante para fazer com que todos no time compartilhassem a mesma visão para a construção de algo com qualidade e de vencer um desafio ao longo do projeto. Descobrimos que existia uma desinformação sobre *Scrum*, sobre ágil e antes de começar era preciso fazer as pessoas acreditarem na importância disso tudo.

O desafio inicial foi a criação de uma vontade de tentar *Scrum* e compreender seus princípios fundamentais. Essa formação geral depois foi seguida por um treinamento de práticas específicas e foi introduzido um especialista consultor no time para apresenta técnicas ágeis a serem experimentadas e validadas pelo time.

4.2.2 A Prova de Conceito

Foi definido após o periodo de treinamentos iniciais sobre Scrum, sobre agilidade e de experimentar um pouco viver uma mudança da cultura tradicional para a cultura ágil a criação de um *Sprint Demo* onde iríamos experimentar e validar algumas tecnologias, feramentas e técnicas durante um mês.

O Sprint Demo: Projeto Alexandria

O projeto Alexandria consiste em uma sistema de busca de arquivos em formato pdf, doc, txt , facilitando consultas do seu acervo de documentos como uma biblioteca indexada. (https://github.com/cmilfont/alexandria)

O Impacto da mudança era forte, foi proposto aprender uma nova linguagem de programação para o time e validar sua utilização durante esse mês , desenvolver guiado a comportamento usando BDD , usar um novo tipo de repositório para o time e aprender a trabalhar com ele, utilizar uma ferramenta online para acompanhamento do projeto junto com com o quadro de tarefas e soma-se a isso trabalhar de forma exposta usando Scrum.

As novas tecnologias escolhidas foram :

Linguagem de Programação Ruby com Framework Rails 3.0 (http://rubyonrails.org/). Precisavamos de uma linguagem aberta por recomendação de que na universidade não desenvolvermos com software proprietário. O Time teve uma experiência anterior com JAVA e a curva de aprendizado foi dura, procurou-se pesquisar uma linguagem de código aberto que tivesse sintaxe simples, uma curva de aprendizado baixa e com uma comunidade presente para apoiar quem estava começando. A escolha foi Ruby,

Ferramentas para escrita de testes : Cucumber(http://cukes.info/ e https://github.com/cucumber/cucumber) e Rspec (http://rspec.info/).

Repositório: Git(Foi definido que o projeto seria de código aberto e portanto passamos colocar tudo no github (www.github.com/wwagner33/solar).O time pesquisou e viu que o Git era o que de mais novo e eficiente tinha em repositories e resolvemos experimentar de trocar o SVN pelo Git e junto a isso seguimos um workflow de desenvolvimento novo. Veja na Figura 28 o github do projeto Alexandria onde se descreve a *feature* Indicar autores dos livros.

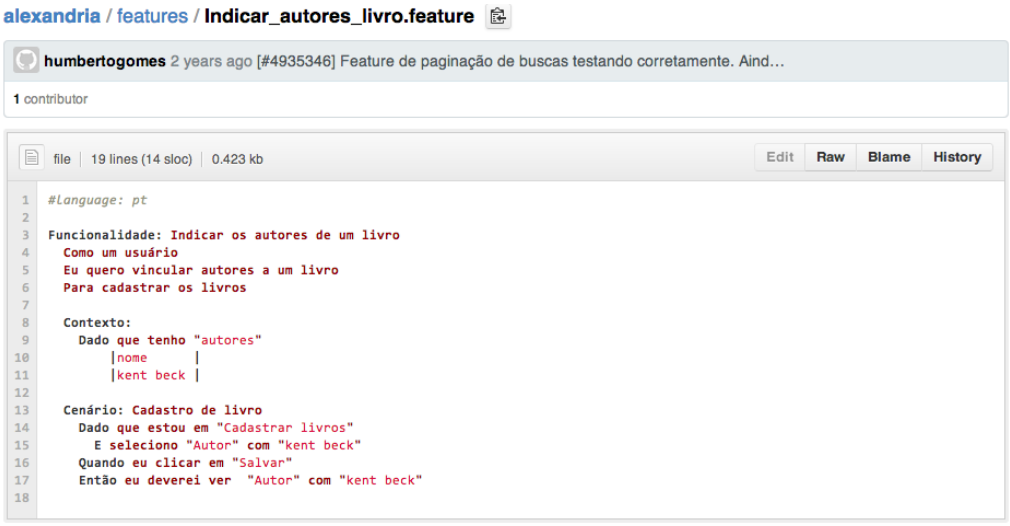


Figura 28 - Github do Projeto Alexandria (Feature de Indicar Autores de Livros).

Pivotal Tracker – Ferramenta para acompanhamento e controle do projeto que se integra as demais ferramentas apresentadas. (www.pivotaltracker.com). Ver Figura 29 :

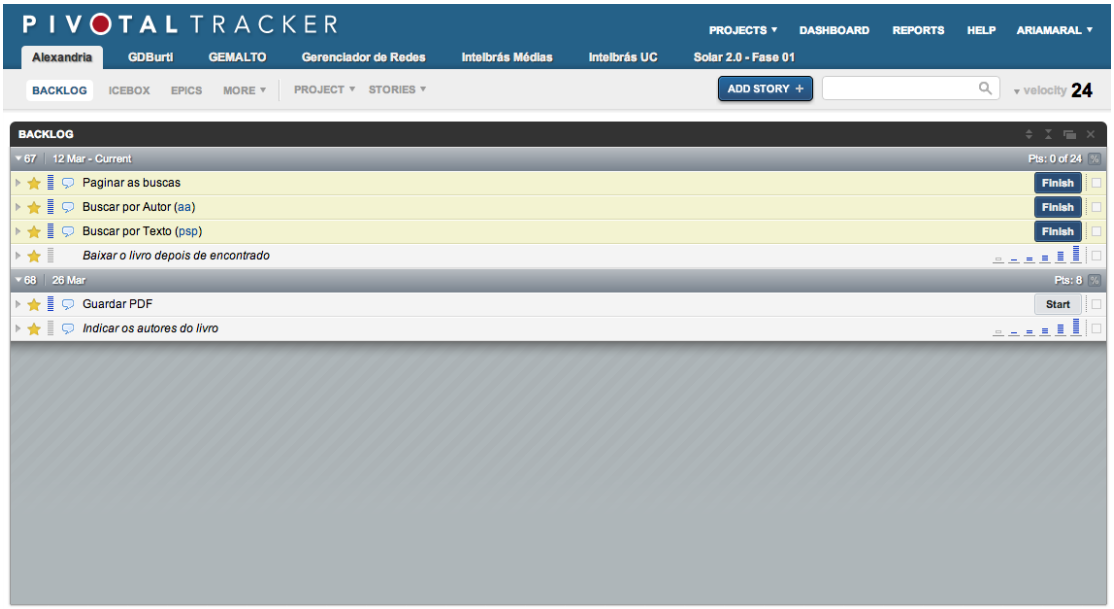


Figura 29 - Pivotal Tracker do Projeto Alexandria.

Técnicas selecionadas para começar:

- Uso de *Scrum*, tentando usar o framework de forma pura seguindo todas suas cerimônias e artefatos e papéis.
- Definição de pares. Inicialmente começamos com 6 desenvolvedores trabalhando sempre em dupla, onde ficou definido que esses pares sempre se revisarem a cada dois dias.
- Uso da modelagem ágil focando em especificar e documentar somente o que for para compreensão e facilitar a comunicação do time.
- Uso do BDD (Desenvolvimento guiado a comportamento) para inserir a cultura de testes no time.Entendemos que começar por testes de comportamento tinha um alto valor agregado para nosso tipo de aplicação. Isso nos traria inclusive o benefício de termos sempre uma documentação atualizada e executável.

Após o time ter certa maturidade com os testes de comportamento ficou decido que o próximo passo seria a incorporação de testes unitários e montagem do servidor de integração continua.

Juntamente com o fornecimento de *coaching* e treinamento, o time tinha que ter o comprometimento que será responsabilizado pela aplicação das novas competências, de compartilhar informações e se tornarem multiplicadores.

Tomamos como decisão nos guiar para o desenvolvimento do projeto solar 2.0 na Declaração de interdependência ,Segundo (pham, 2012), enquanto o Manifesto Ágil lida com desenvolvimento de software, a "Declaração de Interdependência" (*Declaration of Interdependence* ou DOI) da Gestão de Projeto gil, que outro grupo de especialistas elaborou em 2005, focaliza mais o gerenciamento de projetos (http://pmdoi.org):

> "Somos uma comunidade de líderes de projeto que tem sido altamente bem-sucedida em entregar resultados. Para alcançar tais resultados:
> – Aumentamos o retorno do investimento, tornando o fluxo contínuo de valor o nosso foco;
> – Entregamos resultados confiáveis, engajando os clientes em interações frequentes e propriedade compartilhada;

- Esperamos incertezas e gerenciamos levando-as em conta, por meio de iterações, antecipação e adaptação;
- Promovemos criatividade e inovação reconhecendo que os indivíduos são a fonte última de valor e criamos um ambiente em que eles fazem a diferença;
- Impulsionamos o desempenho por meio do compromisso do grupo em obter resultados e da responsabilidade compartilhada pela eficácia do grupo;
- Melhoramos a eficácia e a confiabilidade por meio de estratégias situacionais específicas, processos e práticas."

A criação do ambiente propício para um projeto ágil foi priorizada desde o início do projeto, pois é sabido da grande dificuldade em mudar uma cultura anterior.(Cohn, 2010).

O time por um Mês deve a chance de validar as tecnologias experimentadas, verificar dificuldades, desafios e tomar a decisão de seguir em frente agora no projeto Solar 2.0.

A mudança cultural foi ocorrendo aos poucos, fazer reuniões em tempos pré-estabelecidos eram cada mais difícil e isso foi identificado como um ponto de melhoria.

A cultura de desenvolver guiado a teste foi uma grande dificuldade por exigir conhecimento tanto do framework de testes, como da sintaxe do mesmo bem como conhecimento de domínio do negócio. Os testes foram muito demorados, chegando a consumir mais de duas vezes o tempo de codificação sem os testes.

O pareamento inicial ajudou a pessoas com uma cultura de teste mais forte poderem transferir um pouco de seu conhecimento para os que tinham menos familiaridade com os testes.

No início a razão desse esforço inda não era clara para os desenvolvedores, chegando a gerar certo desânimo no ritmo do time.

Embora com todas essas dificuldades o time se manteve unido e assumiu o compromisso e desafio do desenvolvimento do Solar 2.0 usando o Scrum e as tecnologias escolhidas.

Transparência, inspeção e adaptação começaram a fazer parte do dia a dia do projeto e expor de forma mais forte cada defeito e qualidade de cada membro do time.

4.2.3 O Desenvolvimento do SOLAR 2.0

O sistema é um Ambiente virtual de Ensino e Aprendizagem que tem o propósito de facilitar o processo de educação e de transmissão de conhecimento. Portanto temos como usuários finais, alunos, professores, administradores e editores de conteúdo.

Nossos principais usuários são nossos alunos que foram envolvidos inicialmente através de pesquisas de usabilidade e acessibilidade em cima de uma versão anterior do sistema, buscando suas necessidades e também foi feita reunião com professores para coletar ideias de como melhorar o processo de ensino e aprendizagem usando o sistema.

Isso foi usado para a captação de requisitos do sistema. E foram construídos mapas mentais de como deveria ser o sistema.

Durante a release, sempre eram convidados usuários finais e stakehoders do projeto para participar da Revisão da release , inclusive participando do aceite das mesmas.

Após uma *release* pronta foram marcadas aulas betas, onde turmas presenciais com auxílio do professor testaram o sistema como ferramenta de apoio em uma aula e no final coletamos dados para medir a satisfação do usuário e de como melhorar o sistema.

Posteriormente foram identificados personagens e escolhido representantes dessas para participar junto com o *Product Owner* de reuniões de sugestão de melhoria para o sistema.

O Processo de desenvolvimento inicialmente começou com *Scrum.*

4.2.3.1 Reuniões de Planejamento no SOLAR 2.0

No processo de desenvolvimento do sistema foi realizada técnicas do Scrum, sendo adotada uma reunião de planejamento sempre ao início de nossos *sprints* e procuramos sempre planejar um *sprint* na frente, o que tem facilitado o nosso ganho de velocidade quando o time consegue terminar os itens do backlog da sprint antes do prazo determinado que era de duas semanas. Quando terminamos antes, já puxamos o próximo item priorizado , estimado e

detalhado para o desenvolvimento. Dividimos nosso planejamento e 2 partes, cada uma levando em torno de 2 horas. Durante a primeira fase com o *Product Backlog* já semi priorizado pelo *Product Owner* com ajuda do *Scrum Master* colocamos as funcionalidades para serem discutidas buscando discutir o valor de negócio de cada uma e deixando em formato de *user stories*. Após isso é feito uma *planning poker* para estimar a complexidade de cada funcionalidade, usamos a escala de fibonacci e pontuamos cada funcionalidade.

Para pontuação no planejamento do solar 2.0 foi usado o cálculo em horas e uma formula adaptada e criada observando a realidade do projeto , onde foi considerado que em 8 horas de trabalho se tem de em média 5 horas produtivas. Então foi adotado o seguinte cálculo , 1 turno de 4 horas de trabalho são 3 pontos , 2 turnos são 5 pontos , 3 turnos são 8 pontos. Então uma tarefa de 8 pontos levariam em média um dia e uma manhã de outro dia. Como meta foi colocada para que fosse entregue algo todo dia. Pelo menos uma tarefa por dia.O planejamento foi feito entre os 6 desenvolvedores da equipe de desenvolvimento, o *Product Owner* e o *Scrum Master* e geralmente um convidado do time de usabilidade e acessibilidade que tem uma visão do backlog de produtos e que já começa a trabalhar com antecedência junto com a equipe de usabilidade nos itens que iam ser discutidos na reunião. O time de usabilidade já levava para o sprinto do desenvolvimento os protótipos de telas das tarefas a serem realizadas no Sprint, inclusive já com Cascade Style Sheet (CSS) pré-definido e regras de usabilidade e acessibilidade das telas.

Um dos motivos do *Product Owner* sempre deixar priorizada uma *sprint* na frente e já com uma boa granularidade é da equipe de usabilidade já poder trabalhar nas telas para quando chegar no planejamento do desenvolvimento já se conseguir estimar melhor e ter uma não mais precisa do que vão construir., Isso facilita para o *Product Owner* se realmente é a funcionalidade que ele esperava. Daí em diante são discutidas as funcionalidades e as mesmas são quebradas em tarefas que não ultrapassem os 8 pontos, garantindo assim sempre que alguma coisa seja entregue ao longo do dia. Após a priorização e pontuação o time começa a escolher os itens que vão trabalhar e de acordo com a expertise necessárias os pares vão sendo formados. O padrão do time é trabalhar sempre em pares e somente em casos específicos desfazer as duplas e trabalhar individualmente, isso vai depender da funcionalidade e da necessidade do momento.

Depois de pontuarmos e quebrarmos as tarefas revisamos com o Product Owner a prioridade das tarefas agora pontuadas e se o mesmo deseja alterar algo antes de começar o sprint.

Durante o planejamento o Product Owner junto com o time define um valor de negócio para as funcionalidades.

Na reunião de planejamento também é marcado o horário da reunião diária que normalmente ocorre no período da tarde por volta de quinze horas, horário que praticamente todo time está sempre presente.

4.2.3.2 As Reuniões Diárias no SOLAR 2.0

Durante a reunião diária procurasse manter a comunicação do time em evidência e cada uma fala o que fez desde a última reunião, se teve alguma dificuldade e o que pretende fazer após a reunião diária. Quando a dificuldade pode ser resolvida pelo *Scrum Master* ou algum membro da equipe os mesmos já se manifestam sugerindo ajuda para aquele problema. Quando as pessoas não sabem o que vão fazer, o *Scrum Master* procura olhar para os itens do *backlog* do *sprint* priorizados e ofertar aos membros do time para que escolham dentre os mais prioritários os que se sintam mais a vontade pra fazer naquele momento e nesse mesmo instante caso a tarefa exija um conhecimento a mais que ele se sinta inseguro, alguém da equipe já se oferece para parear com ele pra poder tornar a tarefa mais produtiva, servindo também para a formação de pares durante as tarefas a serem executadas no *sprint* onde pode-se ter ou não , dependendo das funcionalidades uma grande rotação de duplas durante um mesmo *sprint*, contribuindo para um melhoramento da multidisciplinaridade do time.

4.2.3.3 As Reuniões de Retrospectiva no SOLAR 2.0

A Reunião de Retrospectiva foi utilizada para fazer um levantamento das atividades realizadas do ciclo de desenvolvimento. Esta reunião foi realizada ao final do Sprint e dela participam o time, *Scrum Master* e as vezes o *Product Owner* como convidado. Tivemos problemas nas primeiras retrospectivas de ultrapassar o tempo e de discussões improdutivas em virtude de discordância de opiniões. Através de inspeção e adaptação se foi melhorando as retrospectivas até o modelo no qual foi usada a técnica dos 6 chapéus do pensamento de Edward Bono para reduzir os conflitos e fazer todos pensarem naquele momento em uma mesma direção usando o conceito de pensamento paralelo. (Bono, 2008). A ideia consiste em convergir o pensamento do grupo alinhando a visão e o humor das pessoas sob um mesmo aspecto. Com a técnica se conseguiu eliminar o desperdício de energia das discussões retóricas e assuntos polêmicos ou com alto grau de incerteza conseguiram ser analisados em tempo constante e sem o desgaste emocional muito comum em reuniões deste tipo.

Segundo (Bono, 2008), cada Chapéu do Pensamento corresponde a um diferente estilo de pensamento. Estes estilos são explicados de seguida:

- Chapéu Branco: Com este chapéu do pensamento vai focar-se nos dados e informações disponíveis. Olhe para a informação de que dispõe e veja o que pode aprender com ela. Verifique que falhas há no seu conhecimento e tente preenchê-las ou tê-las em conta. É neste momento que analisa as tendências passadas e tenta extrapolar sobre dados históricos.
- Chapéu Vermelho: Colocar o chapéu Vermelho vai fazê-lo olhar para os problemas utilizando a intuição, a reação instintiva e a emoção. Também será importante que tente pensar como é que as outras pessoas irão reagir emocionalmente. Tente compreender as respostas das pessoas que não conhecem as suas razões.
- Chapéu Preto: Com este chapéu tem de olhar para todos os pontos negativos da decisão. Tente ver onde é que pode não funcionar, o que pode correr mal. Isto é importante porque destaca os pontos fracos de um plano, permitindo

eliminá-los, alterá-los ou preparar planos de contingência para combatê-los. Este chapéu ajuda-o a fazer planos mais resistentes e mais resilientes.

- Chapéu Amarelo: O chapéu amarelo ajuda-o a pensar positivamente. É o ponto de vista optimista que o ajuda a ver os benefícios da decisão e o valor da mesma. Este chapéu ajuda-o a continuar quando tudo à sua volta parece sombrio e difícil.
- Chapéu Verde: Este chapéu destina-se à criatividade. É aqui que pode desenvolver soluções criativas para um problema. É uma maneira exuberante de pensar, onde há poucas críticas às ideias apresentadas. Nesse chapéu que se busca o ponto de melhoria contínua.
- Chapéu Azul: Este chapéu representa o controle do processo. Habitualmente este chapéu é utilizado pela pessoa que preside à reunião. Quando se depara com dificuldades devido à falta de ideias, eles podem dirigir a atividade para o Chapéu do Pensamento Verde. Quando são necessários planos de contingência, ele pode pedir opinião ao Chapéu do Pensamento Preto, etc.

Com a aplicação da técnica os resultados melhoraram, e as reuniões se tornaram mais objetivas e produtivas. Ver Apêndice A que mostra os documentos de retrospectiva do SOLAR 2.0 usando a técnica dos 6 chapéus do pensamento.

Para minimizar ainda mais o tempo de reunião foi criado um documento modelo no final da reunião de retrospectiva já da próxima reunião e colocado em um wiki onde os membros da equipe podem alimentar a qualquer instante com os fatos que acontecem no *sprint :* pontos positivos, pontos negativos e ideias para manter o que já funciona e melhorar os pontos negativos. Essas ideias podem ser colocadas a qualquer momento no documento. Quando chega o dia da retrospectiva todos já sabem mais ou menos o que aconteceu, os pontos negativos e positivos e é feita uma leitura no dia da reunião onde se é perguntado se existem mas coisas a serem acrescentadas, depois disso é discutido e votada as possíveis soluções para serem postas em prática no *sprint* seguinte, fazendo assim o processo de melhoria contínua. As reuniões se tornaram bem mais rápidas e eficientes.

4.2.3.4 As Reuniões de Revisão de Sprint no SOLAR 2.0

A Revisão de *Sprint* propriamente dita, só ocorre quando realmente há necessidade de apresentar parte do produto e se tem a necessidade de convidados externos e importantes para validar as funcionalidades do projeto.

O Processo de revisão semanalmente geralmente foi feito da seguinte forma : Quando um par do time durante o Sprint termina uma tarefa, a mesma fica em uma fila de espera pra ser revisada por outra dupla que a pegará assim que terminar a que estar trabalhando. isso fica a critério do time na escolha e geralmente é comunicado na reunião diária ou no momento que o mesmo deseje fazê-lo , isso ajudar a reduzir os erros técnicos e depois disso feito , o *Product Owner* já olha a tarefa e verifica se aceita ou rejeita a mesma, sendo assim procuramos fazer *deplo*y continuo com entregas frequentes.

Durantes o final de uma release é feita uma apresentação formal da mesma ao *Product Owner* e convidados com uma probabilidade menor de rejeição. Esse ponto de melhoria foi identificado em nossas retrospectivas ao identificar uma dificuldade de ter o *Product Owner* junto com *Stakeholders* e todo fim de Sprint. Foi proposto para que o *Product Owner* todos os dias reservasse uma hora para revisar as funcionalidades chamando o *Scrum Master* para quaisquer dúvidas ou mesmo perguntando direto ao time.

4.2.3.5 Dificuldades e Soluções Encontradas no decorrer do Projeto SOLAR 2.0

As maiores dificuldades foram a de mudança de cultura de fazer o time sair de uma zona de conforto e experimentar algo novo e se expor constantemente uns aos outros, defeitos, qualidades, temperamentos. A propriedade coletiva do código inicialmente foi algo complicado. Um ambiente com funcionários públicos , prestadores de serviço, estagiários, um time heterogêneo e sofrendo de multitarefas nocivas pois a maioria atuava em mais de um projeto e conhecer o Framework Scrum e os valores e princípios do manifesto ágil.

O SOLAR 2.0 era um projeto importante que ainda não tinha conseguido sair do papel, um software de grande complexidade e imensa relevância mas com muitos requisitos a serem desenvolvidos e melhorados.

Surgiu a proposta de desenvolver o SOLAR fora dos paradigmas tradicionais de desenvolver de uma forma iterativa e incremental e pautada em cima do manifesto ágil e do framework *Scrum* que veio para ajudar a melhorar a organização dos projeto e facilitar o o seu início projeto que estava parado por acúmulo de requisitos.

Problemas de comunicação foram superados com a reunião diária frequente que se incorporou no dia a dia do time e hoje o time não precisa que o *Scrum Master* os chame pra reunião, eles mesmos se convidam e sentem prazer nesse momento de compartilhamento dos resultados do projeto e de como uns podem ajudar aos outros.

O Mau dimensionamento de tarefas foi resolvido principalmente em se criar protótipos de telas, regras de usabilidade antes de começar o planejamento e de também de serem definidos os itens no *backlog* de todo o time ,discutir a modelagem de banco, e de algumas maneiras de resolver um problema fazendo isso de forma conjunta e ágil, todos juntos buscando a melhor maneira de representar um problema para melhor entendê-lo.

Reuniões demoradas foram resolvidas principalmente com adoção da técnica dos 6 chapéus do pensamento e de criar um WIKI para todos já irem adiantando os tópicos. Também resolvemos isso já fazendo uma priorização na frente do *backlog* e o *Product Owner* procurando deixar as tarefas sempre numa boa granularidade antes da próxima Sprint.

A engenharia de software foi aos poucos sendo melhorada colocando como meta em cada *sprint* seguinte, inserirmos uma boa prática que discutida nas retrospectivas poderiam melhorar o processo.

O objetivo no solar é que o sucesso seja uma interseção entre a satisfação e felicidade do time , O retorno sobre o investimento para o cliente e a boa utilização das práticas do Scrum e Xp e Kanban para o sucesso técnico do projeto.

Depois de uns 6 meses de desenvolvimento o Scrum não mais estava funcionando, os horários do time estavam complicados e não conseguiam mais atender time box então o Kanban surgiu como opção para termos uma implementação baseada em fluxo contínuo e sem *sprints* bem definidas.(Anderson,2010).

Foi aplicado um questionário com 9 pessoas que fizeram parte o time do solar 2.0 e foi extraído dele algumas análises, uma das perguntas feitas foi quais os maiores problemas que atrapalharam o projeto Solar 2.0 ?

A resposta mais citada foi a ausência do *Product owner* para poder maximizar e validar o que era desenvolvido.

De 9 desenvolvedores colheu-se:

- Após aplicação do questionário para colher as informações dos desenvolvedores que participaram do Solar alguns problemas ficaram evidenciados conforme as respostas da pergunta do questionário no Apêndice B sobre quais os maiores problemas que atrapalharam o desenvolvimento do solar 2.0 ?
- Pela análise das respostas concluísse que Ausência do *Product Owner* aparecendo em 3 respostas diretamente e em mais 2 na opção outros , sendo ainda outras 2 pessoas mencionaram de as funcionalidades não estarem bem definidas o que também seria problema com *Product Owner*, totalizando que 7 pessoas de 9 mencionaram algum problema em relação ao *Product Owner* e depois veio a opção e ter que trabalhar em vários projetos simultaneamente, com 3 respostas diretas e mais uma na opção outros. Ver Figura 30.

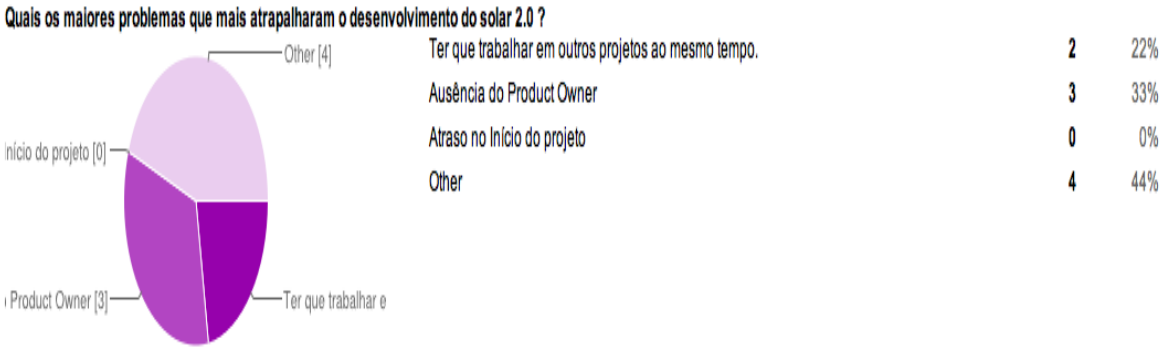


Figura 30 - Problemas que Atrapalharam o Desenvolvimento do Solar 2.0

4.2.3.6 Métricas de Avaliação do Sistema

Foram escolhidas algumas Métricas para medir o sucesso pessoal, técnico e organizacional.

Para medir o sucesso pessoal da equipe a cada *sprint* procurou-se fazer com que alguém relatasse o que aprendeu de novo e no que conseguiu melhorar de uma *sprint* para outra. No final de cada *release* era mensurado o nível de satisfação do time e como poderia melhorar

Usamos gráficos de *burndown* para medir o quanto faltava ser entregue pra cada release e sprints.

Foi medido a velocidade do time, a quantidade de entregas feitas por *sprint* e a quantidade de valor entregue a cada *sprint.*

Foi feita a tentativa de medir a quantidade de iténs rejeitados pelo Product Owner durante cada *Sprint* , o que acabou não funcionando devido a uma ausência forte do Product Owner.

Foi despriorizado a medição de coberturas de testes no projeto devido a cobertura ter sido feita por identificação de itens críticos do sistema.

Procuramos colocar no sistema um mecanismo de log onde medimos a navegabilidade do nosso usuário buscando através de um feedback real melhorar o uso do sistema e a satisfação do nosso usuário.

Também existe um botão no sistema onde o usuário pode dar feedbacks constantes (ver Figura 31) a medida que for utilizando o sistema e queira contribuir dando sugestões ou críticas. fazendo com que sejam vistos o comportamento do usuário e a satisfação do mesmo em relação ao sistema. Essa avaliação é sempre analisada pelo *Product Owner* que depois repassa para o time para que possamos sempre estar fazendo melhorias no sistema. Nosso sistema nasceu com o propósito de ser um software em melhoria contínua e o Scrum foi decisivo para o sucesso desse propósito.

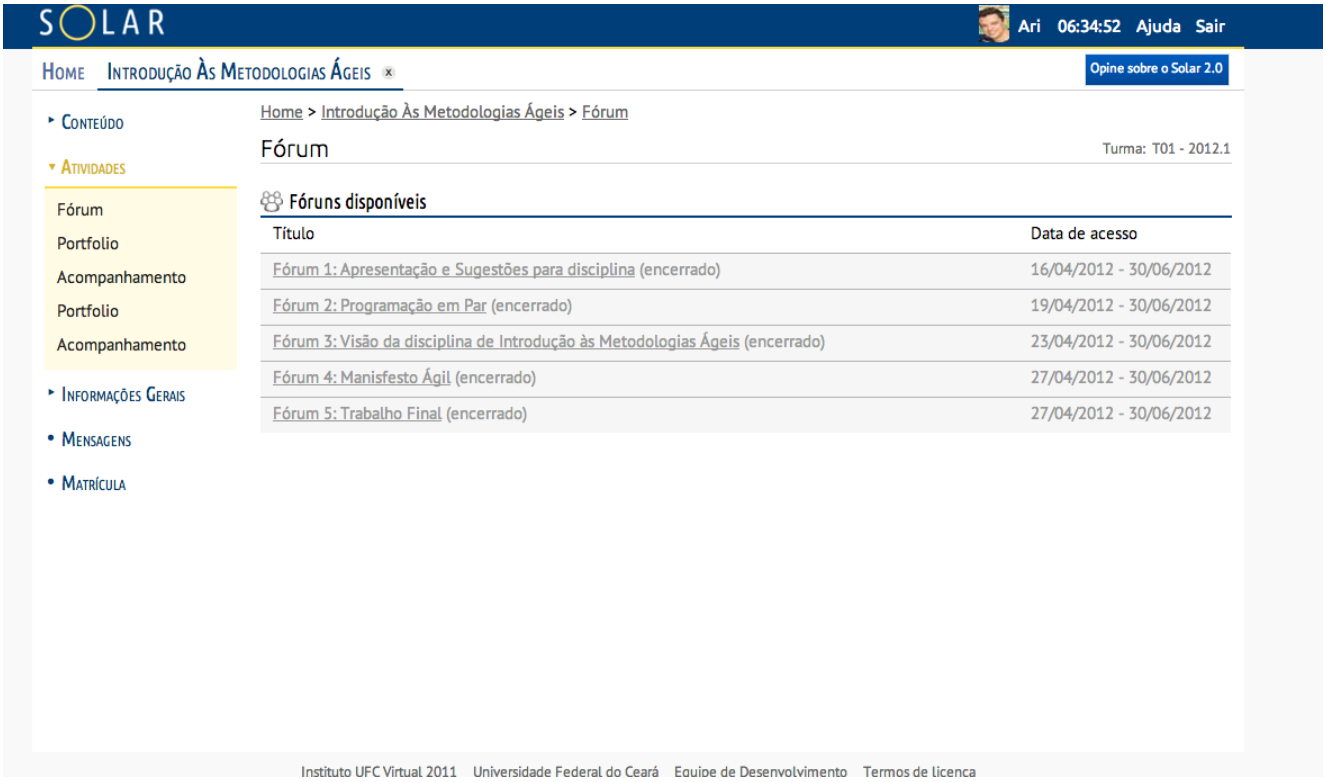


Figura 31 - Fórum do Solar 2.0 (Botão Opine Sobre o SOLAR no Canto Superior Direito).

Para validação do conceito de sucesso mostrado na figura 2 , contemplando o Sucesso técnico, Sucesso organizacional e Sucesso pessoal foram aplicados questionários para os desenvolvedores, e usuários do solar onde foi feita uma análise qualitativa e quantitativa em cima dos dados do questionário para os usuários. E uma análise qualitativa em cima dos questionários dos desenvolvedores.

Também foi feita observação direta dos professores em sala de aula com o uso do solar e discutido pontos de melhorias do sistema com os próprios alunos em sala de aula.

O questionário com usuário final foi aplicado em dois cursos presenciais, um de pós graduação e outro de graduação onde se tinham usuários mais experientes em tecnologia. A amostragem desse questionário foi de 27 pessoas inicialmente.

A quantidade de membros do time do Solar que respondeu o questionário foram de 9 pessoas onde responderam 25 perguntas que tratavam desde sua satisfação pessoal até problemas encontrados a qualidade do código desenvolvido.

4.2.3.7 Avaliação e Teste do Sistema com o Usuário Final

Na avaliação com o usuário final for levado em consideração a usabilidade do sistema e validação de suas funcionalidades. O método foi aplicação de questionário e questionamento direto com os alunos em sala de aula.

Podemos avaliar inclusive o tempo de resposta do sistema para o usuário onde 92% disseram ser bom ou excelente e o restante aceitável, não tendo nenhuma reclamação quanto a performance do sistema. Conforme Figura 32 :

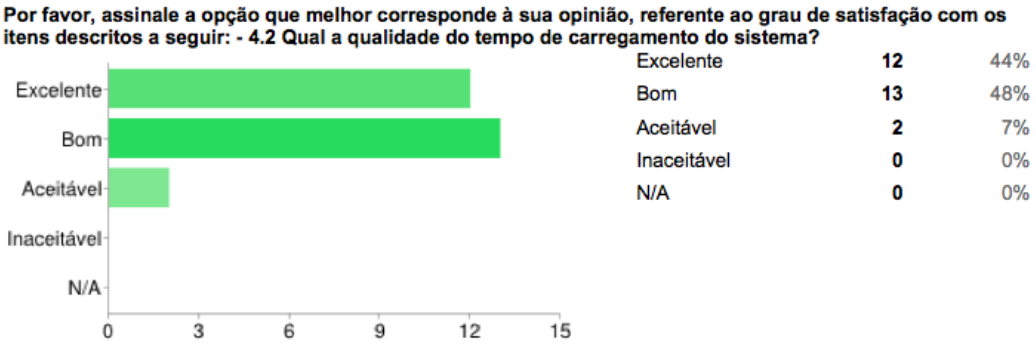


Figura 32 - Qualidade do Tempo de Carregamento do Sistema

.

Quanto a clareza do sistema 67% julgaram o sistema fácil de entender e com uma boa clareza do seu conteúdo, mostrando que a usabilidade encontra-se em uma tendência satisfatória. Ver Figura 33 :

Figura 33 - Clareza e Lógica do Sistema

Um diferencial no Solar 2.0 também é o usuário poder ver o caminho percorrido na sua navegação ficando fácil de voltar ou escolher um caminho anterior. Quanto a esse critério do usuário ver o caminho percorrido 71% acharam bom ou excelente e 26% aceitável , não havendo rejeição a essa funcionalidade, portanto deverá ser mantida no sistema. Ver Figura 34 :

Figura 34 - Possibilidade de Ver o Caminho Percorrido.

Quanto a importância dos elementos gráficos e navegação 59% acharam bom ou excelente e 24% aceitável. Recebemos alguns feedbacks para melhorarmos ainda mais a parte gráfica e de navegação do Solar 2.0. Ver Figura 35 :

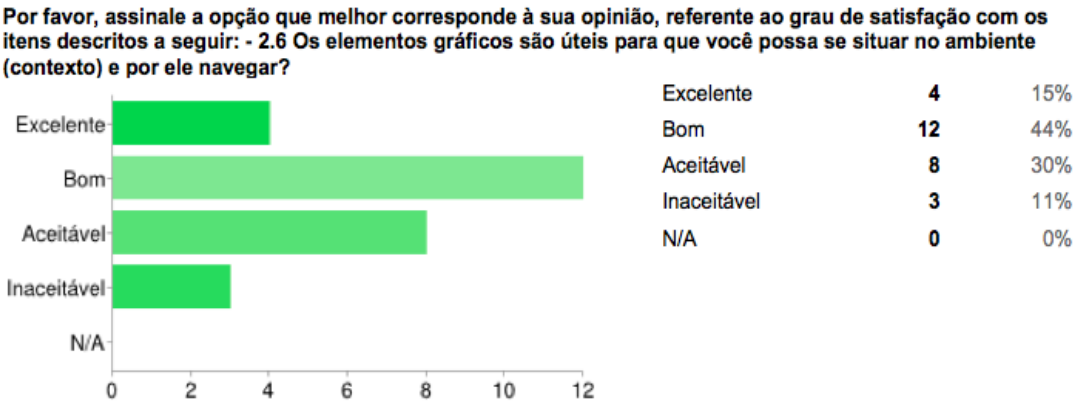


Figura 35 - Elementos Gráficos de Navegação

Sobre a clareza dos menus do sistema o resultado foi mais equilibrado, tivemos 57% entre bom e excelente, 26% aceitável e 15% inaceitável. Vários usuários passaram feedback de sugestões que estão sendo analisadas pela equipe de usabilidade do Solar 2.0 e devem ser implementadas para nova validação, tornando o mecanismo de feedback contínuo uma forma de evoluir a aplicação. Ver Figura 36 :

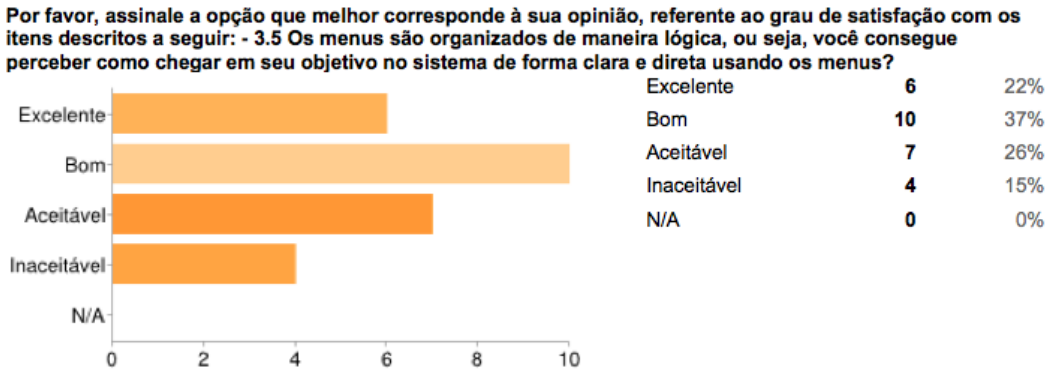


Figura 36 - Menus do Sistema

Sobre as cores escolhidas para o Solar 2.0 foi obtido um alto índice de satisfação onde 74% achou bom ou excelente e 26% aceitável. Ver Figura 37 :

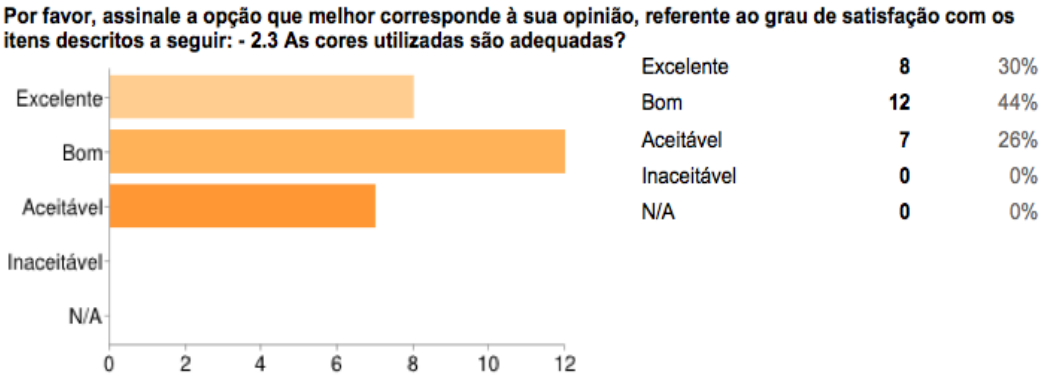


Figura 37 - Cores do Sistema

Os gráficos acima trataram da usabilidade do sistema solar 2.0. As perguntas diretas do questionário trataram também da satisfação e e implementação de novas funcionalidades e recebemos várias sugestões de pessoas pedindo pra ter um botão de dar feedback sobre o sistema que foi logo em seguida implementado, além de sugestões de poder acessar mais de uma disciplina ao mesmo tempo que também está sendo implementada.

O Feedback constante do usuário vai contribuir pra que se construa um software cada vez mais próximo de atender suas necessidades. Ver dados do questionário no Apêndice C, sobre usabilidade e funcionalidades do sistema.

Esse resultado procura preencher a esfera do sucesso organizacional que mede a satisfação do usuário com o produto. O valor que ele agrega ao processo de ensino e aprendizagem.

4.2.3.8 Análise dos Resultados

Em relação a esfera pessoal. O sucesso do projeto estão ligado diretamente a satisfação pessoal das pessoas que nele trabalham. E foi aplicado um questionário de 25 perguntas para tentar extrair do time do Solar um resultado em relação a qualidade de código que se conseguiu ter e quanto a sua satisfação pessoal no Projeto.

Foi perguntado sobre o grau de satisfação em ter trabalhado no projeto do solar e 100% do time respondeu que se sentiu muito satisfeito ou satisfeito. Mostrando que o projeto conseguiu proporcionar com que as pessoas se sentissem bem em fazer parte do mesmo. Ver Figura 38 :

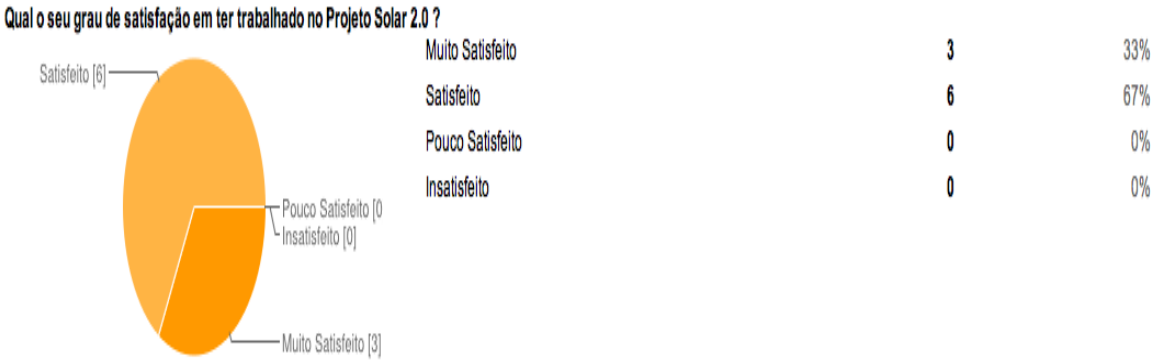


Figura 38 - Grau de Satisfação

Quanto a capacidade de trabalhar em equipe 78% afirmaram que as metodologias ágeis melhoram sua capacidade de trabalhar em equipe, e 22% julgaram indiferente o uso de metodologias ágeis para melhorar sua capacidade de se relacionar. Dentre as respostas positivas, tivemos um comentário na opção outros que afirmou que poderia se trabalhar de forma ainda melhor se algumas pessoas não tivessem resistência. A discussão deve servir pra que busquemos ainda formas melhores de integrar as pessoas mais afastadas do time. Ver Figura 39 :

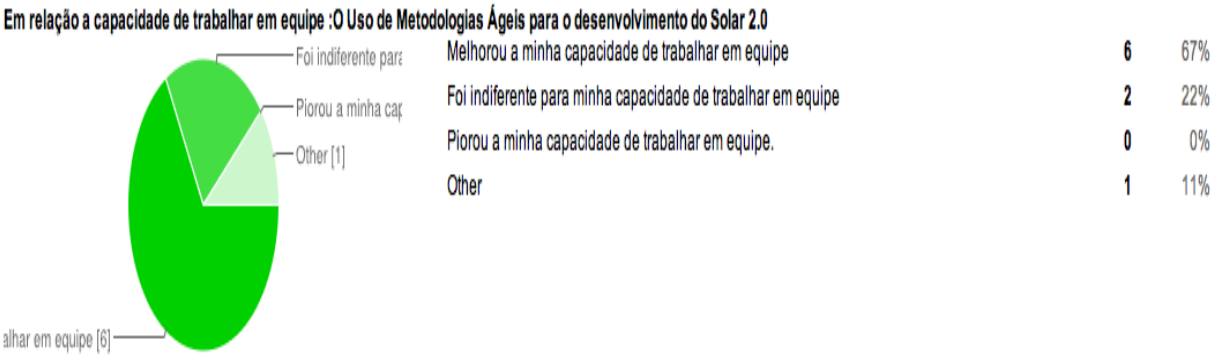


Figura 39 - Capacidade de Trabalhar em Equipe

Em relação a forma de trabalho e satisfação pessoal, 78% se sentiram mais motivadas em trabalhar com metodologias ágeis do que com a forma que trabalhavam anteriormente . Ver Figura 40 :

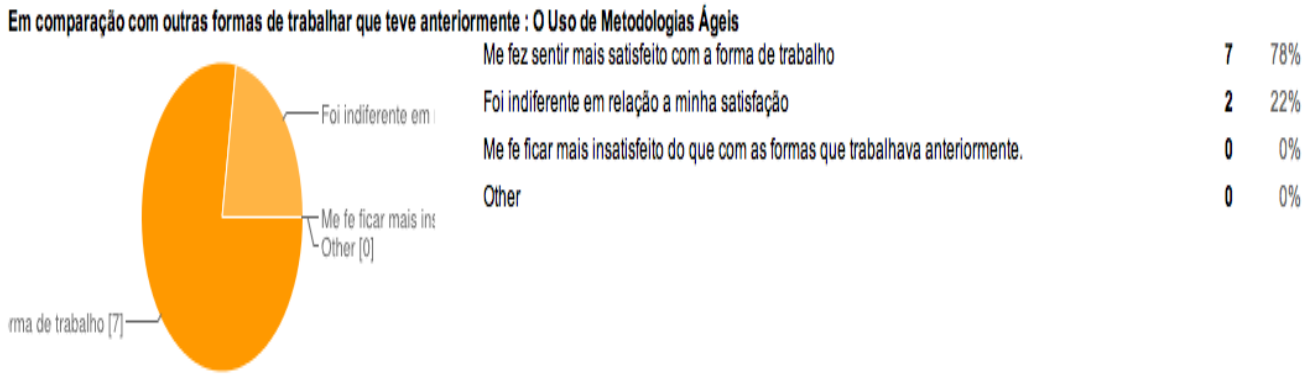


Figura 40 - Comparar Formas de Trabalho.

Em relação aos valores do manifesto ágil tivemos opiniões divididas , mas predominou que Indivíduos e interações são mais importantes que processos e ferramentas com 44%.Nessa pergunta muitos justificaram que a valorização das pessoas no time faz com que os outros valores do manifesto aflorem naturalmente. Ver Figura 41 :

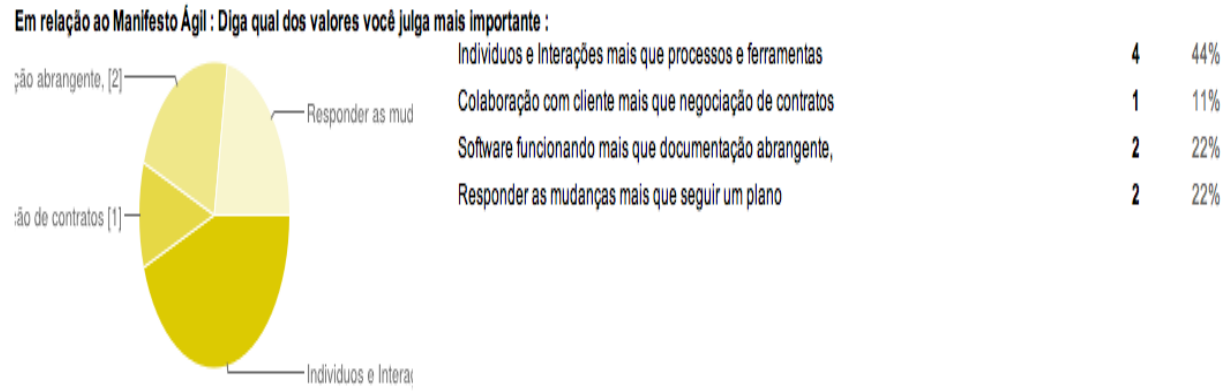


Figura 41 - Valor do Manifesto Ágil Mais Importante

Sobre a Programação em Par , 100% afirmou que facilita a troca de experiências e informações entre os membros do time e 67% ainda disseram que além de facilitar a troca de experiência ainda favorece o entendimento do sistema como um todo. Ver Figura 42 :

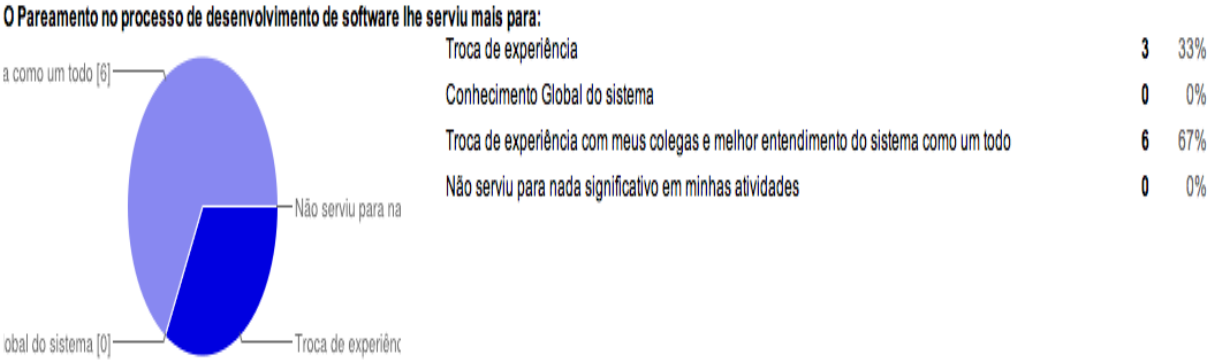


Figura 42 - Importância da Programação em Par

Ainda sobre a programação em par, 100% se mostraram o satisfeitos ou muito satisfeitos em terem trabalhado em par. Ver Figura 43 :

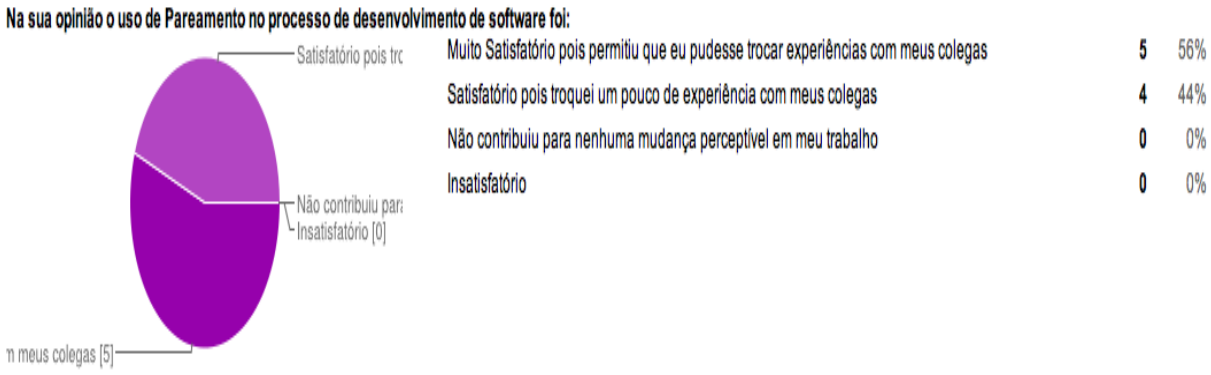


Figura 43 - Satisfação com a Programação em Par

Sobre as técnicas de BDD e cultura de testes nas respostas do questionário do Apêndice B, a maioria do time conseguiu enxergar valor em se ter requisitos executáveis e que esse facilitaram a compreensão e entendimento do sistema para quem viesse conhecer o

sistema. Comentou-se das dificuldades e do tempo investido para se conseguir pensar guiado a comportamento.

Praticamente a totalidade do time também conseguiu enxergar muitos benefícios em ter trabalhado com Metodologias ágeis no Solar e conseguiram ver que se não as tivessem utilizadas teriam mais dificuldade no desenvolvimento do mesmo. OS valores e princípios do manifesto foram usados por quase totalidade dos desenvolvedores e membros do time como fatores que os levaram a trabalharem melhor no Solar 2.0.

Em relação a qualidade do Código desenvolvido , 89% comentou que passaram a desenvolver um código melhor após o uso de práticas ágeis. Ver Figura 44 :

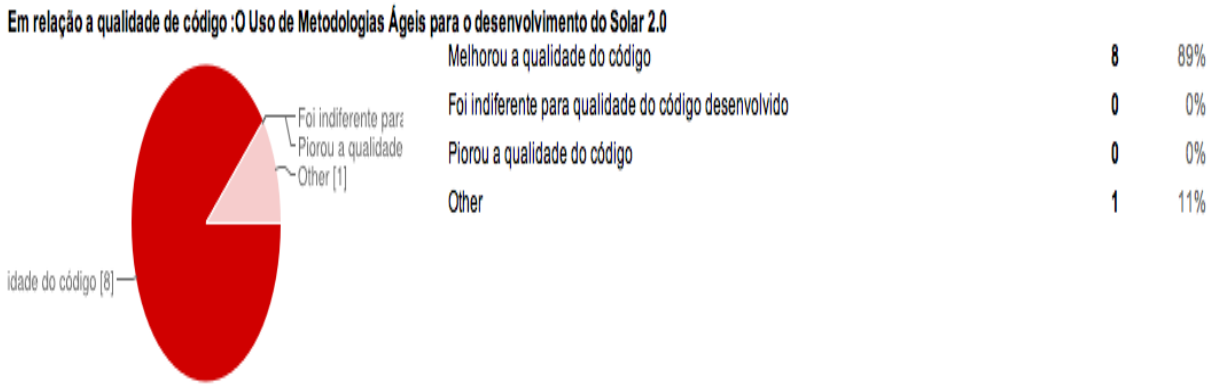


Figura 44 - Qualidade de Código com Métodos Ágeis

Analisando os resultados do questionário do Apêndice B, em sua totalidade concluímos que o os métodos ágeis conseguiram ajudar a maximizar os resultados seja na entrega de valor, seja na melhora da qualidade técnica, seja na satisfação pessoal dos membros do time.

5 CONCLUSÃO

Após dois anos de projeto conseguiu-se ter uma percepção de que sem a utilização de técnicas ágeis o projeto não teria conseguido manter o ritmo de desenvolvimento.

A equipe do Solar adotou o conceito de comprometimento da equipe inteira, reduziu a sua dependência de especialistas, e está fazendo um pouco de tudo o tempo todo, Com certeza ela já fez grandes avanços na forma de trabalhar juntos. E é nesse caso quando a maioria das equipes se tornar complacentes. E em uma análise em cima dos resultados isso de certa forma aconteceu.

Existem ainda oportunidades de melhoria. Para se tornar um time verdadeiramente de alto desempenho e realizar todos os benefícios que o Scrum tem para oferecer, a equipe deve procurar ativamente novas formas de aprender e compartilhar conhecimentos.

Alguns pontos do aprendizado ocorrem naturalmente quando um usuário informa ao *Product Owner* ou por meio do botão de contribuição do solar como quer que um requisito se comporte, ou mesmo quando um desenvolvedor descobre que algumas necessidades não podem ser atendidas com uma tecnologia específica e propõe a melhoria. Outros tipos de conhecimento tem que ser procurados deliberadamente. E é neste tipo de aprendizado que estamos interessados agora . Ao invés de esperar passivamente que a aprendizado ocorra, as equipes mais eficazes e os seus líderes assumem um papel muito ativo na otimização do ritmo e importância do aprendizado.(Cohn, 2010).

Isso acabou dando um impacto forte no desenvolvimento do Solar, pois com um time sem concentradores de informação tornou-se capaz de tanto conhecer do domínio do negocio , ter entendimento da aplicação e conseguir deixar sempre o projeto em um ritmo de nunca parar mesmo tendo atravessado momentos conturbados.

Tanto o papel do Scrum Master quanto o do *Product Owner* acabaram sendo absolvidos pelo time em determinado momento fazendo com que as decências de papéis fossem amenizadas.

As técnicas adotadas para desenvolvimento do Solar conseguiram ter seu propósito senão em todos, pelo menos em alguns momentos dentro do projeto. Destacando-se o uso do *Scrum* que serviu pra desenvolver o senso de auto gerenciamento , auto organização

e fazer sempre o time buscar pontos de melhorias respondendo a mudanças rapidamente e fazendo inspeções e adaptações contínuas.

Destacasse também a importância da Programação em Par como técnica que serviu para aumentar a multidisciplinaridade do time, ensinado a todos a trabalhar em equipe, entender e respeitar melhor o trabalho do outro, expondo defeitos e expertises de cada um, mas fazendo disso um ponto positivo para que sempre estivesse alguém disposto a ajudar outra pessoa da equipe em busca de um objetivo comum.

O desenvolvimento guiado a comportamento foi um tema polêmico pois a técnica gerou uma documentação executável e sempre atualizada, facilitou o entendimento do negócio para os desenvolvedores do Solar 2.O e ainda gerou uma qualidade de código na aplicação. Porém a dificuldade de desenvolver guiado a certo comportamento não é fácil e o esforço despendido para se ter uma alta cobertura de testes de comportamento também é alto, chegando em alguns momentos a desmotivar alguns membros do time. A vantagem do teste em si raramente é percebida logo em um projeto, demorasse certo tempo pra ir percebendo que o esforço a mais no início acaba sendo um ganho no final. Como uma análise em cima da utilização da mesma durante o desenvolvimento do solar ela foi de grande valia pois inclusive possibilitou os desenvolvedores terem um conhecimento maior do negócio e do comportamento da aplicação e facilitando pra quem viesse se integrar ao time aprender mais rapidamente sobre o Solar. A criação de Workshops sobre BDD e uma maior troca de experiências com pessoas de projetos externo maximizariam um uso mais eficiente da técnica.

A maneira de modelar do time não focado em apenas produzir documentos, mas de modelar para entender e comunicar maximizou bastante os resultados do desenvolvimento. Quando se tem um time que se conhece e que se comunica, raramente se usa documentação para consulta, visto que com o BDD (Documentação executável no próprio código), programação em par entre novos membros e quem chega e ainda o uso do *framework Scrum* ou Kanban, já promovem uma contextualização e um entendimento do negócio e do código em pouco tempo.

O Kanban veio pra dar uma liberdade maior ao time em um momento que ter mesmo pequenos papéis como *Product Owner* e *Scrum Master* estava difícil, então a ideia de medir o fluxo contínuo sem precisar mais estimar veio dar uma maior realidade do ritmo de trabalho da equipe do Solar. A análise da velocidade do time sempre foi complicada por a equipe não manter um número fixo de desenvolvedores trabalhando direto na aplicação. Ver

Figura 45 onde mostra a variação da velocidade do time no decorrer do projeto, e consegue-se perceber o quanto oscila a velocidade da equipe. Fatores como mudança de pessoas na equipe. Realocação de pessoas do time em outro projeto, ausência do *Product Owner* em alguns momentos, mudanças de requisitos de forma constante, uma não cobrança específica por prazos gerou uma certa zona de conforto para que houvesse uma despriorização do projeto em alguns momentos por parte da organização. A medição da velocidade mesmo assim tem grande valia, pois se consegue olhar pra um ponto específico no gráfico e identificar o problema e o momento que a equipe estava passando. Ver Figura 45 :

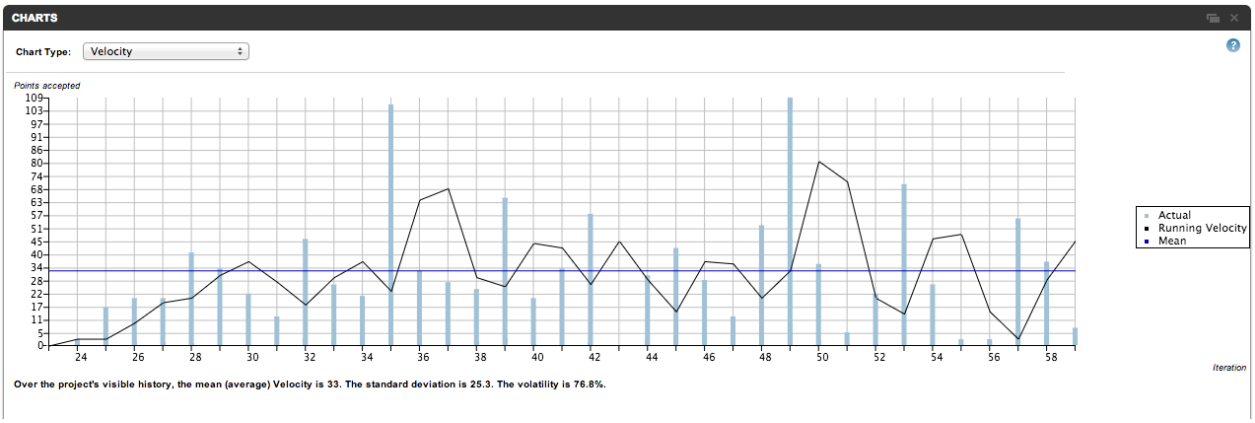


Figura 45 - Velocidade do Time no Projeto Solar.

As retrospectivas sempre que existiram foram a chave para mudanças e pontos de melhoria no projeto. Em certo tempo a ausência de retrospectivas levou o time pra um estado de não saber dos seus problemas, pois perderam o ponto de análise para buscar a melhoria contínua. Se você não sabe onde você está agora, não saberá onde poderá chegar. Uma das chaves para o sucesso em um projeto ágil é sempre buscar a melhoria contínua e sempre fazer retrospectivas. Após um tempo sem fazer, e ver a produtividade e satisfação da equipe diminuir, o próprio time sentiu falta de uma retrospectiva e pediu para que a mesma fosse feita, passando a partir desse ponto gerar novamente inspeção e adaptação para promoção de melhoria contínua. Os documentos gerados nessas reuniões serviram para colocar sempre novas metas a serem seguidas nos *sprints* e a se fazer análises frequentes do que se pode melhorar no projeto, servindo de métrica para o grau de satisfação do time em cada fase do projeto. Ver no Apêndice A, o documento de retrospectiva onde os membros do time expressam como estão se sentindo depois de cada *Sprint* em relação a satisfação pessoal, profissional, organizacional e de qualidade de código desenvolvido.

Os problemas com a despriorização em relação ao desenvolvimento do Solar 2.0 acabaram comprometendo os marcos determinados inicialmente. Como trata-se de um software pra desenvolvimento interno, embora com grande poder de inovação e um grande valor agregado a instituição isso acabou facilitando para um descompromisso em priorizar que se tivesse uma equipe priorizando em período integral o desenvolvimento do Solar. Isso não implica que o time era descompromissado, pelo contrário, o time quando estava focado no solar sempre se mostrou extremamente compromissado mas o problema da multitarefa durante o projeto foi frequente. Raramente se tinha desenvolvedores trabalhando de forma integral somente no Solar.

O que leva a uma reflexão que não só o time precisa fazer ágil e Scrum, mas que a implementação tem que ser em duas vias de direção, de cima pra baixo e de baixo pra cima. Quando não se sabe suas prioridades e se fica gerando muitas interrupções e se promove a multitarefa acaba-se tendo que pagar um preço.

Será que se quando se pedia para alguém parar o que estava fazendo e assumisse outro projeto, aquilo era realmente tão urgente? Será que não se podia esperar um pouco?

Quando se não tem prioridades bem definidas você acaba entrando em um círculo que não tem fim onde tudo vai parecer urgente e nada vai parecer tão prioritário que não mereça deixar de ser interrompido.

Uma gestão de valor e uma priorização, dar pesos as coisas, medir consequências de esperar uma coisa terminar pra outra começar pode ser o caminho. Nem tudo que se pede pra ontem as vezes é realmente tão urgente. Se isso chega a acontecer é porque não se sabe o que quer.(Brooks Junior, 2010) E parece que voltamos pra quando não sabíamos o que eram esses tais métodos ágeis. Uma implantação também na administração teria possibilidades de resolver muitos problemas encontrados internos no projeto do Solar 2.0.

Os questionários aplicados junto as turmas de alunos de graduação e pos graduação presenciais e dos cursos a distância serviram para medir o quanto o software estava sendo usual com as pessoas e no que precisaríamos melhorar na sua identidade visual de forma que facilitasse o mecanismo de ensino e aprendizado.

Em um primeiro momento tivemos um questionário aplicado pra um público de 27 pessoas sendo desses alunos do curso de Sistemas e Mídias Digitais e de uma pós graduação em metodologias ágeis, sendo que na turma de pós graduação fui professor da disciplina e fiz acompanhamento direto em relação a satisfação dos alunos com o Solar. Nesse

caso o Solar foi testado apenas como ferramenta de apoio em cursos presenciais em turmas com nível avançado de conhecimento em tecnologia.

Os feedbacks passados inicialmente pelas duas turmas de alunos mais avançados foram passados para o desenvolvimento e já criadas novas funcionalidades atendendo parte dos desejos desse público..

Através das respostas dos mesmos conseguimos ter uma métrica de quão o software estava satisfazendo em relação a usabilidade e capacidade de facilitação do processo de ensino e aprendizado. Respostas foram coletadas, analisadas, levadas ao time de desenvolvimento e muitas ideias foram colocadas pra serem desenvolvidas por *feedback* dos usuários do sistema. Uma delas foi a integração da plataforma com redes sócias que foi pedido por usuários e analisado através de observação direta em sala de aula que os alunos acabam ficando com redes sociais abertas e desfocando ao ficar mudando de ambiente. A integração acabou servindo de ponto de motivação pra continuar no Solar 2.0. A ideia de se ter um AVA onde quase tudo que você queira fazer na Internet estiver dentro dele parece agradar bastante. Isso será um ponto a ser validado mais pra frente.

Sobre o medir a satisfação pessoal do time foi aplicado um questionário também que levantou vário pontos sobre o lado pessoal, técnico e de gerar valor com o software desenvolvido e se conseguiu perceber pontos ainda a serem discutidos e melhorados no trabalho em equipe do Solar que será discutido em uma próxima reunião buscando sempre , satisfação pessoal, ter um bom nível de desafio, se manter motivado agregando valor a instituição e gerando satisfação dos usuários com os softwares desenvolvidos por esses programadores.

Há indícios fortes de que um time motivado, trabalhando dentro de um ambiente que promova uma cultura de aprendizado, estando desafiado e sempre aprendendo novas habilidades por consequência produzirá software que agregue valor aos usuários, e por consequência estejam aumentando sempre a excelência técnica de seus produtos. O lado humano satisfeito funciona como uma alavanca para os outros dois pontos do sucesso. Sendo assim O sucesso pessoal afeta diretamente o sucesso técnico e organizacional do projeto.

Métodos Ágeis são para softwares complexos. AVA são Softwares bem complexo. Após todas as análises esse casamento realmente parece fazer sentido. Mas precisa sempre está sendo analisado o estado atual para se projetar o estado futuro. Não vai existir uma metodologia ou um processo bem definido que garanta o sucesso do desenvolvimento.

As metodologias ágeis são focadas em pessoas e se conseguirmos através dos valores e princípios do manifesto ágil contaminar essas pessoas a se sentirem mais felizes, conseguirão adaptar melhor o processo conforme a necessidade para poder maximizar cada vez mais seus resultados.

A Mensuração do sucesso do projeto está em ver que se conseguiu ter um time que tem orgulho de participar do projeto de desenvolvimento do SOLAR 2.0. De saber que está fazendo algo diferente. Que está participando de alguma forma para maximizar o nível de conhecimento e aprendizado das pessoas, pois estão construindo Ambientes Virtuais de Aprendizagem e não um simples ambiente, mas um que procura entender e aprender com seus usuários tentando sempre melhorar a usabilidade, forma de interagir com usuário e descobrindo novas funcionalidades que facilitem o processo de ensino e aprendizado tornando as pessoas mais sábias.

O SOLAR foi pensado para todos e o maior desafio do time é estudar *personas* diferentes e entender a importância de seu trabalho. No segundo questionário de usabilidade que ainda será disparado deverá contemplar portadores de necessidades especiais que usam o SOLAR tendo assim o sistema a preocupação de ser um software acessível para todos e a mensagem de que um time de desenvolvimento está promovendo inclusão social e educacional deve ser enfatizada, pois esse tipo de coisa gera motivação intrínseca no time.

Pessoas são a solução. Metodologias Ágeis priorizam pessoas. *Scrum* é um *Framework* adaptado conforme a realidade das pessoas do cliente, o processo deve ser evolutivo e adaptativo sempre buscando a melhoria contínua, seja para organização, seja na qualidade técnica e principalmente nas pessoas.

5.1 - Trabalhos Futuros

Como trabalhos futuros pretendemos lançar questionários de usabilidade e validação das funcionalidades do novo SOLAR para todos os cursos da UFC presenciais em

semipresenciais e nessa coleta de dados fazer um estudo de *personas*, para que possamos desenvolver funcionalidades mais voltadas pra perfis específicos que usem o Solar, fazendo com que os mesmos possam ter um software modularizado de acordo com suas necessidades.

Pretende-se colocar no sistema um mecanismo de *log* onde possamos medir a navegabilidade do nosso usuário buscando através de um *feedback* real melhorar o uso do sistema e a satisfação do mesmo. Sendo assim teremos sempre um Software com *feedback* contínuo dos que usam e em estado de melhoria contínua, pois o SOLAR 2.0 foi criado para ser um *software* fácil de manter, fácil de usar e em constante melhoria contínua.

Em um próximo módulo deverão ser construídas ferramentas para edição de material digital, onde possibilitará ao professor poder criar seu próprio material sem precisar de auxílio de um especialista em EAD. Todo processo de desenvolvimento deve ser feito com uma evolução das técnicas ágeis aplicadas no SOLAR 2.0.

Como trabalhos futuros também se pretende mensurar o aprendizado utilizando técnicas ágeis para auxiliar no processo de ensino e aprendizagem. Trabalho esse que já se iniciou em algumas turmas de graduação e pós graduação onde se foge da maneira tradicional de ensino e se baseia em uma aprendizado coletivo, participativo com feedbacks contínuos, fazendo com que o aprendizado seja facilitado por algumas técnicas usadas, como reuniões de retrospectivas , planejamentos frequentes junto com os alunos, e exercícios que forcem uma mudança cultural .(Torres Filho, 2013).

REFERÊNCIAS

Appelo. Jurgen. **Management 3.0: Leading Agile Developers, Developing Agile Leaders.** Addison-Wesley Professional, 2011.

Appelo Jurgen . **Como Mudar o Mundo: Gestão de Mudanças 3.0**. 2012.

Appelo. Jurgen. **Delegation boards: Management 3.0 workout.** http://www. management30.com/workout/delegation-boards, 2013. Acessado em 05/07/2013

AMADEUS, Universidade Federal de Pernambuco, 2010, Disponível em http://amadeus.cin.ufpe.br/*wiki*/index.php/Documentos. Acessado em 8 de julho de 2010.

AMBLER, Scott W. 2008. **Agile adoption rate survey**, February disponível em http://www.ambysoft.com/surveys/agileFebruary2008.html. . Acessado em 02 de junho de 2008

AMBLER, Scott W.. **Modelagem Ágil:** Práticas eficazes para Programação eXtrema e o Processo Unificado. São Paulo: Bookman, 354 p, ISBN: 85-363-0298-4, 2004.

ANDERSON, David J.. **Kanban: Mudança Evolucionária de Sucesso para seu negócio em Tecnologia.** Wasshington Dc: Blue Hole Press, 2011. 288 p. (9780984521463). Tradução Andrea Pinto.

ANICHE, Mauricio Finavaro. **Como a prática de TDD influencia o projeto de classes em sistemas orientados a objetos.** 2012. 75 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Departamento de Instituto de Matemática e Estatística, IME-USP, Universidade de São Paulo, São Paulo, 2012.

ARMONY, Rafael Sabbagh. **Fatores críticos para a prática de valores ágeis em equipes de tecnologia da informação. 2010. 196 p.** Dissertação (Mestrado em Departamento de Administração) - Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro.

BARBOSA, Rommel M. (org.). **Ambientes Virtuais de Aprendizagem**. Editora Artmed, ISBN: 85-363-0515-0, 2005.

BECK, K. et al. **Manifesto for Agile *Software* Development**. Dezembro 2001. Disponível em http://www. agilemanifesto.org/. Acessado em Junho de 2010.

BECK, Kent; .ANDRES, Cyntia. **Extreme Programming Explained** : Embrace Change.Addison-Wesley Professional, 2nd edition, 2004

BONO, Edward de. **Os seis chapéus do pensamento.** SEXTANTE, ISBN13: 9788575423578, 2008.

BROOKS JUNIOR, Frederick P.. **The Design of Design:** Essays from a Computer Scientist. Boston: Addison-Wesley Professional, 448 p, ISBN: 0-201-36298-8, 2010.

BROOKS, F. P. **The Mythical Man-Month: Essays on *Software* Engineering**. Addison-Wesley Professional, 1995.

COCKBURN, Alistair. **Agile *Software* Development:** The Cooperative Game. 2. ed. Boston: Addison-wesley Professional, 506 p, ISBN: 0-201-36298-8, 2006.

COHN, Mike. Agile Estimating and Planning. Prentice Hall, 2005.

COHN, Mike. **Succeeding with Agile:** *Software* Development Using Scrum. Boston: Addison-wesley Professional, 504 p, ISBN: 978-0-321-57936-2, 2010.

DIAS, Rosana de Fátima. **Ambientes Virtuais de Aprendizagem – Uma metodologia para avaliação de *software*.** Florianópolis, 2003. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-graduação em Engenharia de Produção, UFSC, 2003.

FRANÇA, A. C. **Um Estudo sobre motivação em equipes de desenvolvimento de software.** Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, 2009.

GOMES, André Faria. **AGILE: Desenvolvimento de Software com entregas frequentes e foco no valor de negócio**. São Paulo: Casa do Código, 2013. 149 p.

HIRANABE, Kenji, "Kanban Applied to Software Development: From Agile to Lean," *InfoQ*, January 14, 2008. http://www.infoq.com/articles/ hiranabe-lean-agile-kanban.

HIRANABE, Kenji. **"Visualizing Agile Projects Using Kanban Boards."** *InfoQ*, August 27, 2007; www.infoq.com/articles/ agile-kanban-boards.

HUNT, John. ***Agile software construction.*** **London: Springer, 2006.**

KEITH, Clinton. **Agile Game Development With Scrum.** Boston: Addison-Wesley Professional, 367 p.ISBN: 978-0-321-61852-8, 2010.

KNIBERG, Henrik; SKARIN, Mattias. **Kanban e Scrum: Obtendo o Melhor de Ambos.** Washington: C4media, 2009. 139 p. (978-0-557-13832-6). Coordenação de Tradução por Renato Willi e Vinicius Assef.

LACEY, M. **The Scrum Field Guide: Practical Advice for Your First Year.** Addison-Wesley Professional, ISBN-13: 978-0-321-55415-4, 2012.

LEFFINGWELL, Dean. **Agile Software Requirements**. Addison-wesley Professional, 2011. 560 p. (0321635841).

MARIZ, Leila Maria Rodrigues de Souza. **Uma análise comparativa entre o discurso e a prática na gestão ágil de projetos de software com Scrum,** 2009. Dissertação (Mestrado

em Ciência da Computação)- Centro de Informática da Universidade Federal de Pernambuco, Recife, 2009.

MELO, Claudia de O. et al. **Métodos ágeis no Brasil: estado da prática em times e organizações.**. São Paulo: Ime - Usp, 2012. 9 p.

NORTH, D**. Introducing Behaviour-Driven Development**, 2011. http://dannorth.net/introducing- bdd.

OLIVEIRA, Celina Couto de et al. **AMBIENTES INFORMATIZADOS DE APRENDIZAGEM:** PRODUÇÃO E AVALIAÇÃO DE *SOFTWARE* EDUCATIVO. Campinas, Sp: Papirus, 2001. 146 p. (85-308-0634-4).

PEREIRA, Alice Cybis et al. (Org.). **AVA - Ambientes Virtuais de Aprendizagem:** Em diferentes contextos. Rio de Janeiro: Ciência Moderna, 218 p.ISBN: 978-85-7393-607-0, 2007.

PHAM, A. **Agile Software Project Management and Development.** Course Technology PTR, ISBN-13: 978-1-4354-5913-7, 2011.

POPPENDIECK, Mary e Tom. **Lean *Software* Development: An Agile Toolkit for *Software* Development Managers.** Addison-Wesley Professional, 2003.

POPPENDIECK, Mary e Tom. **A History of Lean: From Manufacturing to *Software* Development**, JAOO Conference, Aarhus, Denmark, 2005.

PRESSMAN, S Roger. Engenharia de Software, 6ªedição,São Paulo, Mc Graw Hill,ISBN: 85-86804-57-6, 2006

SABBAGH, Rafael. **Scrum: Gestão Ágil para projetos de Sucesso**. São Paulo: Casa do Código, 2013. 280 p.

SARMENTO, Wellington W. F. **Integração de um Ambiente Virtual de Aprendizagem com Aplicações Móveis de Suporte a Educação a Distância.** Dissertação de Mestrado aprovada pelo Programa de Pós-graduação do Departamento de Teleinformática da Universidade Federal do Ceará, 2007.

SARMENTO, Wellington W. F. **Relatório Técnico 01/2010**: Análise de Ambientes Virtuais de Aprendizagem Open Source, Texto publicado pelo Instituto UFC Virtual, 2010-a.

SARMENTO, Wellington W. F e TORRES FILHO, Ari do Amaral **Relatório Técnico 02/2010**: Proposta de um novo Ambiente Virtual de Aprendizagem para a Universidade Federal do Ceará, Texto publicado pelo Instituto UFC Virtual, 2010-b.

SATO, Danilo Toshiaki. **Uso Eficaz de Métricas em Métodos Ágeis de Desenvolvimento de *Software*.** 2007. 139 f. Dissertação de Mestrado (Mestre) - Curso de Ciência da Computação, Departamento de Informática, Universidade de São Paulo, São Paulo, 2007.

SCHAWABER, K e BEEDLE, M. **Agile *Software* Development with *Scrum***, Nj: Pretence Hall, 2001.

SCHWABER, K. **Agile Project Management With *Scrum***, Microsoft, 2004.

SCHWABER, Ken and Jeff Sutherland., 2011. "The Scrum Guide"**http://www.Scrum.org/storage/Scrumguides/Scrum_Guide%202011.pdf**

SCHWABER, Ken., *Agile Software Development with Scrum*. Redmond: Microsoft Press, 2004.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide**.2010.Disponível em: http://www.Scrum.org/Scrumguides/.acesso em 05/07/2010

SHORE, James; WARDEN, Shane. **A Arte do Desenvolvimento Ágil:**Rio de Janeiro: Alta Books,422 p. ISBN: 97885-7608-203-3 , 2008.

SOARES, Michel dos Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de *Software*.** 2004. Disponível em: <http://www.dcc.ufla.br/infocomp/artigos/v3.2/art02.pdf> . Acessado em 10 junho 2010.

SOMMERVILLE, Ian. **Engenharia de Software**. ed 8. Tradução: Selma Shin Shimizu Melkinoff, Reginaldo Arakaki, Edilson de Andrade Barbosa. São Paulo: Pearson Addison-Wesley, 2007.

SOUZA, Diego da Silva. Scrum and Scrum: **Uma análise sobre dependências ao escalar times ágeis.** 2013. 107. Tese (Mestrado em Informática) – Instituto de Matemática, Instituto Tércio Pacciti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

TORRES FILHO, Ari do Amaral. **Ensinando e Aprendendo com Agilidade**. Brasília: Amaral, 2013. 29 slides, color. Disponível em: <http://www.slideshare.net/ariamaral/ensinando>. Acesso em: 30 jun. 2013.

STANDISH, Standish Group,2009."The CHAOS Report."The Standish Group International,Inc. http://www.standishgroup.com/newsroom/chaos_2009.php . Acessado em 28 Junho de 2011.

STANDISH, The Standish Group International. **The CHAOS Report**. 1994. Disponível em: http://www.standishgroup.com/sample_research/chaos_1994_1.php.Acessado em 05 de Junho de 2011.

STANDISH, The Standish Group International. **The CHAOS Report**. Standish Group.2004.Disponível em: http://secure.standishgroup.com/reports/reports.php?rid=500. Acessado em 10 de Junho de 2010.

STERLING, Chris., Getting Agile website., **http://www.gettingagile.com/2009/07/15/the-forgotten-Scrum-elements/** (accessed on 3 July 2012).

TELEDUC, Universidade de Campinas, 2005. Disponível em http://teleduc.nied.unicamp.br/teleduc. Acessado em 11 de julho de 2012.

TELES, Vinicius Manhães. **Um Estudo de Caso da Adoção das Práticas e Valores do Extreme Programming.**2005. 180 f. Dissertação (Mestrado em Informática) , Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

THERESINHA, A; SHMITH, V.; ALVARES, M. R. Ambientes Virtuais de Aprendizagem, 2007.

TIDIA-AE, Universidade de São Paulo. Disponível em **http://new.tidia-ae.usp.br/tidia-ae/Login.jsp** e acessado em 8 de junho de 2012.

TSCHWABER, K. "Agile **Project Management with Scrum",** Microsoft Press, 2004.

VERSIONONE, **Pesquisa do Estado do Desenvolvimento Ágil,** ***The State of Agile Development***. Disponível em http://www.versionone.net/whitepapers.asp#survey. Acessado em 29 de junho de 2012.

WOODWARD, Elizabeth; GANIS, Matthew; SURDEK, Steffan. **A Practical Guide to Distributed Scrum.** Boston: Ibm Press. 240 p. ,ISBN: 978-0-13-704113-8. , 2010.

APÊNDICES

Apêndice A – Documentos das Reuniões de Retrospectivas do projeto SOLAR 2.0

Segunda Reunião de Retrospectiva do projeto Solar 2.0

Nessa reunião que aconteceu no dia 14/02/2011 passamos a usar a técnica dos 6 chápeus do **Pensamento** de Edward Bono para **evitar conflitos, evitar discussões** e **sermos mais objetivos. Cada chapéu representa 1 foco, todos discutem ou apresentam o mesmo problema** sobre o mesmo ponto de vista. A técnica incentiva a participação coletiva. A reunião vai ter um tempo pré definido. Máximo 1 hora.

O Scrum master usará o chapéu azul no início e fim da reunião. E fixamos um tempo de 5 minutos para o inicio e para o fim da reunião.

Resolvemos experimentalmente seguir a sequência :

- Chapéu Azul. -- Organizador,Contexto, objetivos gerais. 5 Minutos
- Chapéu Branco. -- Fatos, números, informações, sem emoções, sem julgamentos.
- Chapéu Preto. -- Críticas, o que pode dar errado? Quais os riscos ?Cautela, Pior cenário ,mas não improvável. 10 minutos.
- Chapéu Amarelo. -- Pontos positivos, oportunidades, possibilidades, beneficios, lógica,melhor cenário mas não improvável. 10 minutos
- Chapéu Vermelho. --Sentimentos, emoções, sinceridade,palpites, intuição, sem julgamento, sem explicações, transparência, clima. 1 minuto pra cada membro.

● Chapéu Verde. -- Novas idéias ,criatividade,pensar fora da caixa,evolução, opções, brainstorm. 15 minutos.

● Chapéu Azul. Organizador, resultados obtidos, condução do processo. 5 minutos. Começaremos a retrospectiva usando o Chapéu Azul -- O Scrum Maste rapresentando a técnica dos 6 chapéus ao time, definindo a caracteristica de cada chapéu e o modelo que iríamos seguir. Também foi apresentado o propósito de nossa reunião que consistia em fazermos uma avaliação da última sprint, vendo pontos fortes, pontos fracos, colhendo sentimentos de todos em relação ao projeto, dicas, sugestões de como podemos melhorar, o que devemos manter, modificar pra que sempre o time seja mais produtivo e mais realizado com o trabalho desenvolvido. A técnica foi proposta e aceita pelo time. A escolha da técnica foi por problemas de tempo e não alinhamento de ideias ocorridos na reunião passada. Chapéu Branco -- ● Foi feito muito CSS na Sprint . ● ● Houve uma cobertura de testes de todos os caminhos felizes em todas as features e em algumas foram feitos os caminhos alternativos também. ● O time esteve mais presente. Menos falta.

● Houve criações de Features no meio da Sprint. ● Houve atraso nas telas das features que estávamos desenvolvendo. ● Faltou o time definir um padrão de codificação. ● Laerte esteve ausente por motivo de doença. ● Grande Rotação nas duplas na programação em par. ● Time se auto organizou bem. ● Reuniões diárias aconteceram. ● Não utilizamos o Task board manual e nem criamos o Burndown Chart Visual. ● Meta da sprint foi cumprida. ● Conseguimos entregar muitas features. ● Algumas features rejeitadas pelo PO. ● Features rejeitadas corrigidas. ● Entrada do Wedson no time. ● Palácio e Patricia ausentes por alguns dias da sprint. ● Ausencia do Palácio e da Patricia na reunião de retrospectiva.

Chapéu Preto --

● Seguir os passos errado do workflow do Milfont para o GIT pode fazer o time perder muito tempo.

● Falta de sincronização dos commits. ● Falta de treinamento. ● Mal dimensionamento da complexidade das histórias. ● Dificuldade com o Selenium. ● Quebra de testes constantes. ● Não padronização do arquivo de internacionalização. ● Falta de café. ● Testes feitos ao final. ● Falta de um local pra guardar telas e ícones. ● Product Owner as vezes indisponível. ● Falta do servidor de testes. ● Falta de uma visita do Milfont. ● Pouca participação do time nas listas de discussões.

Chapéu Amarelo -- ● Maior presença da equipe. ● Grande parte das features internacionalizadas. ● Conhecimento sobre tecnologias por parte dos membros vem aumentando. ● Boa rotação de duplas favorecendo o aprendizado. ● Melhor aprendizado com os testes. ● Consciência da importância de realmente se fazer testes. ● Uso do Selenium. É muito bom ver as telas sendo preenchidas, e os testes acontecendo

de forma automatizada. ● A produção do time nessa sprint superou o previsto.

[redacted]

● Novas pessoas no time e melhor integração das que já estavam. ● Treinamento do Rodrigo ajudou em dúvidas sobre os testes e trouxe dicas que ajudaram

a otimizar o processo..

Chapéu Vermelho

- Rafael -- Feliz , gostando da melhora, sensação de ser útil, aprendendo coisas novas, gostando do desafio, otimista.

- Humberto -- Feliz, aprendeu mais nessa sprint, gostou da incorporação do selenium.Gostaria de treinamentos para produzir melhor ainda.

- Patricia-- Não Participou.
- Palácio--Não Participou.
- Laerte -- Gostou que as coisas funcionaram, gostaria de ter estado mais presente, motivado.
- Wedson -- Feliz em ter entrado pro time, gostou do projeto, mostrou interesse em aprender mais e colaborar com a equipe. Bruno -- Mais satisfeito por os tumultos terem diminuidos, mais confiante, sensação de o time mais maduro e entendo melhor o objetivo a ser conquistado, precisando de motivação no estilo de aprendizado(treinamentos). ● ● Alexciei -- Aprendendo bastante, participativo. ● Ari -- Mais otimista em relação ao projeto, preocupado com a fata de treinamentos para o time. **Chapéu Verde**
- Imprimir a sequência do Milfont e colocar na sala visível para todos.(Ari)
- Aprender e fixar os comandos principais do GIT. A Patrícia tem um resumo que pode ser colado na parede.(Ari)
- Tentar Viabilizar parcerias para que existam treinamentos internos que colaborem para melhoria das tecnologias e processos utilizados no projeto.(UFC Virtual)
- Propor ao Rafael que depois de fazer o curso da Caelum sobre Rails avançado semana que vem possa ministrar um treinamento para o restante do time e com o que aprendeu lá passar a usar algumas novas técnicas e tecnologias que possam contribuir para o sucesso do projeto.(Rafael)
- Assistir o Video sobre GIT do Akita que o Scrum Master disponibilizou.Pode entrar como tarefa no backlog. (time)
- Para resolver o problema do Laerte e Alexciei que estarão trabalhando somente pela manhã, o Humberto ficará

disponível para o solar nas terça e quintas e o Wedson todos os dias. Nas reuniões diárias as duplas a cada dia fazem um micro gerenciamento pra ver a melhor produtividade da rotação das duplas.(time)

- Criar um disco virtual nosso para armazenarmos e organizarmos arquivos, imagens, gifs e documentos do solar.(Decidir um responsável do time para viabilizar isso)

- Tentar viabilizar a aquisição de uma conta Plus no Github para podermos ter mais espaço e podermos fechar áreas do código que não quisermos disponibilizar.

(Wellington)

- Utilizar o Dropbox para facilitar o envio e recebimento de arquivos entre o time. (Time)

- Tentar Viabilizar uma nova visita do Milfont para a próxima Sprint. (Ari)

- Mandar e--mails para lista do solar e lista em geral para resolver problemas e tirar dúvidas em relação ao Selenium. (Time)

- Padronizar todas as versões de gems do projeto para que novas instalações mantenham o versionamento evitando quebras no projeto. (Time)

- Convidar o Regis para um dojo com o time, parear durante um dia com a equipe contribuindo para difundir maior conhecimento de git, selenium, rspec e boas praticas de desenvolvimento. (Ari)

- Fazer testes antes. (Time)

- Liberação dos recursos para o projeto TIC na educação, afim de auxiliar na contratação de treinamentos e consultoria para o time. (UFC)

- Colar o mapa mental sobre a técnica dos 6 chapéus na parede da sala. (Ari) **Chapéu Azul**

- A reunião mostrou--se produtiva ocorrendo dentro do tempo previsto de uma hora. Conseguimos fazer uma boa avaliação do que aconteceu na sprint e a equipe olhando para cada problema e solução apresentada durante a mesma conseguiu buscar soluções coletivas para resolver pelo menos parcialmente alguns problemas e temos encontrado durante o projeto.

- Houve uma boa aceitação da técnica dos 6 chapéus e não foi mostrado resistência por parte do time.

Reunião de Retrospectiva do projeto Solar 2.0 05/07/2011

Nessa reunião que aconteceu no dia 05/07/2011 pcontinuamos a usar a técnica dos 6 chápeus **do Pensamento** de Edward Bono para **evitar conflitos, evitar discussões** e **sermos mais objetivos. Cada chapéu representa 1 foco, todos discutem ou apresentam o mesmo problema** sobre o mesmo ponto de vista. A técnica incentiva a participação coletiva. A reunião vai ter um tempo pré definido. Máximo 1 hora.

O Scrum master usará o chapéu azul no início e fim da reunião. E fixamos um tempo de 5 minutos para o inicio e para o fim da reunião.

Resolvemos Continuar com a sequência :

- Chapéu Azul. -- Organizador,Contexto, objetivos gerais. 5 Minutos

- Chapéu Branco. -- Fatos, números, informações, sem emoções, sem julgamentos.

- Chapéu Preto. -- Críticas, o que pode dar errado? Quais os riscos ?Cautela, Pior cenário ,mas não improvável.

10 minutos.

- Chapéu Amarelo. -- Pontos positivos, oportunidades, possibilidades, beneficios, lógica,melhor cenário mas não improvável. 10 minutos

- Chapéu Vermelho. --Sentimentos, emoções, sinceridade,palpites, intuição, sem julgamento, sem explicações, transparência, clima. 1 minuto pra cada membro.

- Chapéu Verde. -- Novas idéias ,criatividade,pensar fora da caixa,evolução, opções, brainstorm. 15 minutos.

- Chapéu Azul. Organizador, resultados obtidos, condução do processo. 5 minutos. Começamos a retrospectiva usando o Chapéu Azul -- ficou decidido nessa reunião que as próximas reuniões serão conduzidas por membros da equipe alternadamente.O propósito de nossa reunião será vermos que práticas nós estamos adotando vendo pontos fortes, pontos fracos, colhendo sentimentos de todos em relação ao projeto, dicas, sugestões de como podemos melhorar, o que devemos manter, modificar pra que sempre o time seja mais produtivo e mais realizado com o trabalho desenvolvido. Nessa reunião devemos avaliar onde estamos e quando devemos terminar nossa Release e com o que devemos nos comprometer para que a mesma seja finalizada com sucesso.
- Chapéu Branco -- Está havendo uma rotatividade de membros da equipe saindo pra outros projetos com frequência.

Tivemos o Agile Brasil 2011 e duas pessoas estiveram ausente da equipe durante a semana do evento. Humberto afastado do projeto temporariamente. Ausência de programação em par.

- Houve uma cobertura de testes de todos os caminhos felizes em todas as features e

em algumas foram feitos os caminhos alternativos também. ● O time esteve mais presente. Menos falta. ● Houve criações de Features no meio da Sprint. ● Houve atraso nas telas das features que estávamos desenvolvendo. ● Faltou o time definir um padrão de codificação. ● Laerte esteve ausente por motivo de doença. ● Grande Rotação nas duplas na programação em par. ● Time se auto organizou bem. ● Reuniões diárias aconteceram. ● Não utilizamos o Task board manual e nem criamos o Burndown Chart Visual. ● Meta da sprint foi cumprida. ● Conseguimos entregar muitas features. ● Algumas features rejeitadas pelo PO. ● Features rejeitadas corrigidas. ● Entrada do Wedson no time. ● Palácio e Patricia ausentes por alguns dias da sprint. ● Ausencia do Palácio e da Patricia na reunião de retrospectiva.

Chapéu Preto --

● Seguir os passos errado do workflow do Milfont para o GIT pode fazer o time perder muito tempo.

● Falta de sincronização dos commits. ● Falta de treinamento. ● Mal dimensionamento da complexidade das histórias. ● Dificuldade com o Selenium. ● Quebra de testes constantes. ● Não padronização do arquivo de internacionalização. ● Falta de café. ● Testes feitos ao final. ● Falta de um local pra guardar telas e ícones. ● Product Owner as vezes indisponível. ● Falta do servidor de testes. ● Falta de uma visita do Milfont. ● Pouca participação do time nas listas de discussões.

Chapéu Amarelo --

- Maior presença da equipe.
- Grande parte das features internacionalizadas.
- Conhecimento sobre tecnologias por parte dos membros vem aumentando.
- Boa rotação de duplas favorecendo o aprendizado.

- Melhor aprendizado com os testes.
- Consciência da importância de realmente se fazer testes.
- Uso do Selenium. É muito bom ver as telas sendo preenchidas, e os testes acontecendo de forma automatizada.
- A produção do time nessa sprint superou o previsto.
- Novas pessoas no time e melhor integração das que já estavam.
- Treinamento do Rodrigo ajudou em dúvidas sobre os testes e trouxe dicas que ajudaram a otimizar o processo..

Chapéu Vermelho

- Rafael -- Feliz , gostando da melhora, sensação de ser útil, aprendendo coisas novas, gostando do desafio, otimista.
- Humberto -- Feliz, aprendeu mais nessa sprint, gostou da incorporação do selenium.Gostaria de treinamentos para produzir melhor ainda.
- Patricia-- Não Participou.
- Palácio--Não Participou.
- Laerte -- Gostou que as coisas funcionaram, gostaria de ter estado mais presente, motivado.
- Wedson -- Feliz em ter entrado pro time, gostou do projeto, mostrou interesse em aprender mais e colaborar com a equipe. Bruno -- Mais satisfeito por os tumultos terem diminuidos, mais confiante, sensação de o time mais maduro e entendo melhor o objetivo a ser conquistado, precisando de motivação no estilo de aprendizado(treinamentos). ● ● Alexciei -- Aprendendo bastante, participativo. ● Ari -- Mais otimista em relação ao projeto, preocupado com a fata de treinamentos para o time.

Chapéu Verde

- Imprimir a sequência do Milfont e colocar na sala

visível para todos.(Ari)

- Aprender e fixar os comandos principais do GIT. A Patrícia tem um resumo que pode ser colado na parede.(Ari)

- Tentar Viabilizar parcerias para que existam treinamentos internos que colaborem para melhoria das tecnologias e processos utilizados no projeto.(UFC Virtual)

- Propor ao Rafael que depois de fazer o curso da Caelum sobre Rails avançado semana que vem possa ministrar um treinamento para o restante do time e com o que aprendeu lá passar a usar algumas novas técnicas e tecnologias que possam contribuir para o sucesso do projeto.(Rafael)

- Assistir o Video sobre GIT do Akita que o Scrum Master disponibilizou.Pode entrar como

tarefa no backlog. (time)

- Para resolver o problema do Laerte e Alexciei que estarão trabalhando somente pela manhã, o Humberto ficará disponível para o solar nas terça e quintas e o Wedson todos os dias. Nas reuniões diárias as duplas a cada dia fazem um micro gerenciamento pra ver a melhor produtividade da rotação das duplas.(time)

- Criar um disco virtual nosso para armazenarmos e organizarmos arquivos, imagens, gifs e documentos do solar.(Decidir um responsável do time para viabilizar isso)

- Tentar viabilizar a aquisição de uma conta Plus no Github para podermos ter mais espaço e podermos fechar áreas do código que não quisermos disponibilizar. *(Wellington)*

- Utilizar o Dropbox para facilitar o envio e recebimento de arquivos entre o time. (Time)

- Tentar Viabilizar uma nova visita do Milfont para a próxima Sprint. (Ari)

- Mandar e--mails para lista do solar e lista em geral para resolver problemas e tirar dúvidas em relação ao Selenium. (Time)

- Padronizar todas as versões de gems do projeto para que novas instalações mantenham o versionamento evitando quebras no projeto. (Time)

- Convidar o Regis para um dojo com o time, parear durante um dia com a equipe contribuindo para difundir maior conhecimento de git, selenium, rspec e boas praticas de desenvolvimento. (Ari)

- Fazer testes antes. (Time)

- Liberação dos recursos para o projeto TIC na educação, afim de auxiliar na contratação de treinamentos e consultoria para o time. (UFC)

- Colar o mapa mental sobre a técnica dos 6 chapéus na parede da sala. (Ari) **Chapéu Azul**

- A reunião mostrou--se produtiva ocorrendo dentro do tempo previsto de uma hora. Conseguimos fazer uma boa avaliação do que aconteceu na sprint e a equipe olhando para cada problema e solução apresentada durante a mesma conseguiu buscar soluções coletivas para resolver pelo menos parcialmente alguns problemas e temos encontrado durante o projeto.

- Houve uma boa aceitação da técnica dos 6 chapéus e não foi mostrado resistência por parte do time.

Terceira Reunião de Retrospectiva do projeto Solar 2.0 Pontos positivos do projeto até o momento

- Implementação dos testes no processo de

desenvolvimento <MANTER e MELHORAR **A EFICIÊNCIA E ABRANGÊNCIA DOS TESTES>;;**

● Não houve muitas mudanças na equipe. Houve somente a saída do Rafael e a entrada do Wedson **<MANTER O QUE TEMOS DESENVOLVIMENTO E USABILIDADE E ENTRAR COM PESSOA NOVA COM BOA EXPERIÊNCIA. Política de carreira, especialização profissional e curso de formação>;;**

● Houve um processo de desenvolvimento em algumas tarefas na forma de pareamento (XP). Outras tarefas não tiveram pareamento, mas houve uma boa interação entre os membros da equipe;;

● Houve uma estabilização da velocidade de desenvolvimento;;

● Finalmente está se habituando com as tecnologias usadas;;

● O código está ficando mais elegante e correto;;

● Estamos conseguindo fluir mais no desenvolvimento;;

● Estamos melhorando o uso do GIT para gerenciar as versões de código. Percebemos que o merge dele é melhor do que o SVN;;

● O SOLAR está melhor definido do que estava antes. Estamos conseguindo concretizá--lo;;

● Uso do Scrum propriamente dito;;

● A união da equipe e facilidade de comunicação entre os membros;;

● Os novatos trouxeram boas experiências para a equipe;;

● <Katryne> O fluxo de trabalho rápido entre Katryne e Andrei;; ○ Entrosamento e comunicação com a equipe de desenvolvimento é boa;;

● <Ticianne> Bom avanço da interface visual do sistema e usabilidade;;

● <Cátia> Equipe de Usabilidade e Design cresceu;; ○ Estamos tentando agora se reintegrar à equipe de desenvolvimento;; ○ Algumas ações do ano passado estão tendo efeito agora, tais como: algunmas questões da acessibilidade estão virando cultura (ex. descrição de imagens). Cresceu uma consciência quanto a acessibilidade;;

● Ter acesso direto a uma equipe de usabilidade e design, para tirar dúvidas imediatas.

Pontos negativos do projeto até o momento

● Não houve um uso correto dos Problemas Inesperados, de forma tal que pudesse abarcar as dificuldades encontradas nas features;;

● Houve uma perda de métricas (ex. velocidade) nas últimas tarefas e dimensionamento destas;;

● Interrupções da equipe para fazer outras tarefas, causando um atraso maior do que o esperado e uma situação que não pode ser suportada pela equipe;;

● Demoramos mais tempo do que se esperava para aprender as novas tecnologias (ex. testes, rails, cucumber, scrum);;

● Geramos MUITOS códigos ruins que deverão ser corrigidos e melhorados assim que o primeiro marco passar (primeiro release);;

● Abrimos mão no processo de algumas coisas que

não deveríamos ter aberto, tais como: não temos sprint bem definidas, não temos sprint review, não temos reuniões de retrospectiva, não temos mais reunião de planejamento periódica;;

● Nossa equipe passou por oscilações grandes de motivação refletindo diretamente na produtividade;;

● Não temos máquina de café;;

● Não temos gelágua;;

● Perdemos tempo, algumas vezes, trabalhando em par. Demoramos a saber quando realmente deve ser utilizado esta modalidade. Por exemplo, em tarefas de pesquisa, tarefas mecânicas. O uso de par na revisão de código, correção de interfaces é importante;;

● Não ter par na revisão de código;;

● Mesmo tarefas bem dimensionadas estão demorando mais do que deveriam (ex. implementação das telas). Talvez por falta de fluência na tecnologia;;

● Falta da regularidade no trabalho em par. Não há uma troca de par com regularidade. Isto ocasiona um não conhecimento coletivo do código gerado;;

● Dificuldade na montagem de telas (ex. HTML, JQuery, CSS);;

● Falta de testes nas últimas sprint;;

● <Katryne> Equipe de usabilidade e design pequena para o volume de trabalho;; ○ Entra e sai constante de bolsistas;; ○ Dificuldade de contato com os colaboradores (ex. Melo);; ○ Dificuldade de horários para acompanhamento do pessoal de desenvolvimento. A equipe de Usabilidade está em peso pela manhã e a de desenvolvimento tem mais problemas a tarde;;

- <Ticianne> Desagregação da equipe;;

○ Centralização de todo o processo no P .O. que gera limitação nas equipes e sobrecarga no P .O.;;

- <Marcus> Falta de dsitribuição correta do trabalho de Usabilidade e Design, deixando um membro da equipe mais sobrecarregado do que outros;;

○ Falta de comunicação interna na equipe de Usabilidade e Design;; ○ Falta saber o que está sendo feito, para quando e como se está, causando uma

deficiência na comunicação para os demais parceiros da equipe;;

○ Falta um conhecimento da parte de Acessibilidade do local onde está se salvando as telas e acesso ao produto (ambiente teste);;

○ Ter mais gente trabalhando com acessibilidade na equipe de Usabilidade e Design;;

○ Saída inesperada de um membro da equipe de Usabilidade e Design que deixou uma lacuna na equipe;;

- <Catia> Perda de contato com a equipe de Acessibilidade da FACED, causando impacto nas nossas atividades;;

○ Doenças pessoais, doutorados e saídas de membros causaram impactos grandes na equipe;;

○ Outras atividades externas ao SOLAR 2.0 ou não que se tornam mais prioritárias e que acabam por impactar negativamente na produtividade da equipe;;

○ Falta de articulação maior com a equipe de

desenvolvimento;;

- Falta de cultura de checar o que está acontecendo através do Pivotal;;

- Falta de espaço físico para programação em par;;

- Falta de espaço físico para trabalhar, incluindo barulho demasiado no espaço devido as reuniões em paralelo entre as diversas equipes;;

- Falta de conhecimento para implementação rápida de algumas funcionalidades, fazendo com que uma dada tarefa tenha que ser refeita várias vezes;;

- Pouco acompanhamento do Product Owner quanto as tarefas, deixando o time esperando para concluir uma tarefa;;

- Falta de detalhamento maior nas tarefas para que pudessem ser melhor compreendidas por quem está executando;;

- Falta de planejamento da estrutura de dados antes do sprint ou da Release;;

- Concentração na sala devido ao barulho e lotação;;

- Paradas inesperadas para ajudar terceiros causam constantes interrupções na execução da tarefa de uma pessoa;;

- Falta um documento que detalhando melhor as funcionalidades a serem desenvolvidas. **Sugestões** Pareamento na Usabilidade;; Política de carreira para os membros das equipes;; Especialização e formação dos membros;; Remanejar o Marcus para mais próximo do resto da equipe;; Colocar em cada funcionalidade o que ela vai fazer, quem vai utilizar e para quem ela vai **servir;;** Melhorar o nosso CSS;; Receber um protótipo

da tela interativo;; Contratar mais uma pessoa para a parte de design e desenvolvimento;; **Motivação: treinamentos, formações e certificações;;** Manter a retrospectiva;;

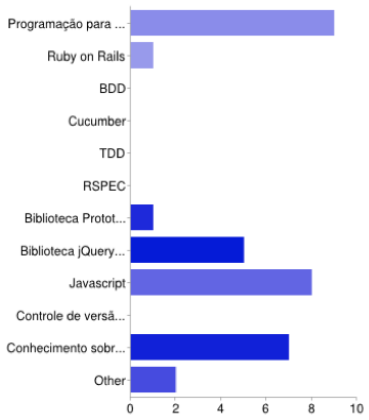
Apêndice B - Pesquisa com os desenvolvedores do SOLAR 2.0

9 respostas

Resumo

Ver as respostas completas

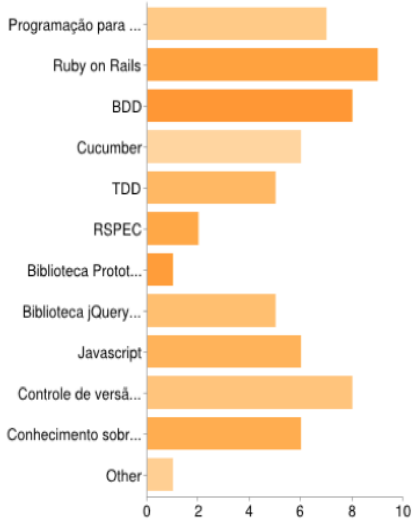
Das tecnologias, sistemas e técnicas utilizadas no SOLAR 2.0 e abaixo listadas, quais as que você já possuía antes de entrar no projeto?



Programação para Web (em qualquer linguagem de programação)	9	100%
Ruby on Rails	1	11%
BDD	0	0%
Cucumber	0	0%
TDD	0	0%
RSPEC	0	0%
Biblioteca Prototype de Javascript	1	11%
Biblioteca jQuery de Java Script	5	56%
Javascript	8	89%
Controle de versão com GIT	0	0%
Conhecimento sobre Ambientes Virtuais de Aprendizagem	7	78%
Other	2	22%

As pessoas podem marcar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

E após o primeiro ano do projeto, quais tecnologias, sistemas e técnicas você agora possui?



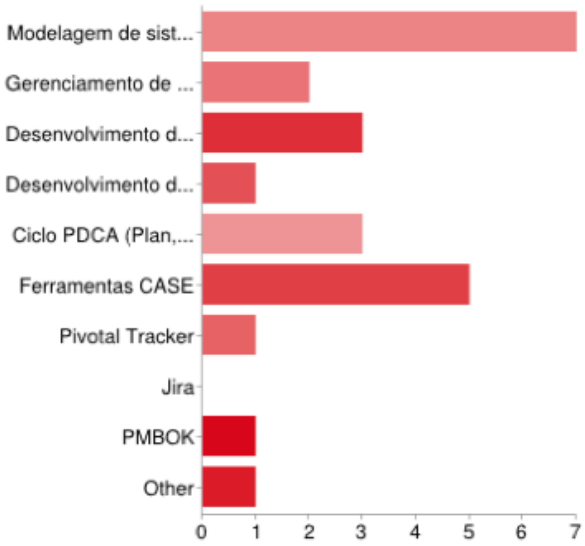
Programação para Web (em qualquer linguagem de programação)	7	78%
Ruby on Rails	9	100%
BDD	8	89%
Cucumber	6	67%
TDD	5	56%
RSPEC	2	22%
Biblioteca Prototype de Javascript	1	11%
Biblioteca jQuery de Java Script	5	56%
Javascript	6	67%
Controle de versão com GIT	8	89%
Conhecimento sobre Ambientes Virtuais de Aprendizagem	6	67%
Other	1	11%

As pessoas podem marcar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

Que metodologias e instrumentos para gerenciamento de projetos de software você conhecia antes de entrar no projeto SOLAR 2.0?

Modelagem de sistemas utilizando UML	7	78%
Gerenciamento de Projetos de Software baseados em SCRUM	2	22%
Desenvolvimento de Software baseado em XP	3	33%
Desenvolvimento de Software baseado em Kan-Ban	1	11%
Ciclo PDCA (Plan, Do, Check and Act)	3	33%
Ferramentas CASE	5	56%
Pivotal Tracker	1	11%
Jira	0	0%
PMBOK	1	11%
Other	1	11%

As pessoas podem marcar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

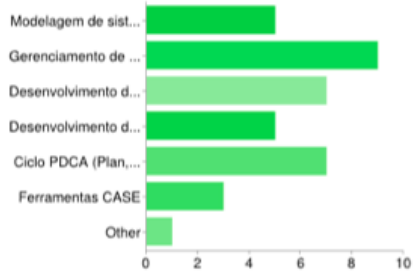


E agora, após este primeiro ano do desenvolvimento do SOLAR 2, das metodologias e instrumentos para gerenciamento de projetos de software você conhece?

Modelagem de sistemas utilizando UML	**5**	56%
Gerenciamento de Projetos de Software baseados em SCRUM	**9**	100%
Desenvolvimento de Software baseado em XP	**8**	89%
Desenvolvimento de Software baseado em Kanban	**5**	56%
Ciclo PDCA (Plan, Do, Check and Act)	**7**	78%
Ferramentas CASE	**4**	44%
Pivotal Tracker	**9**	100%
PMBOK	**1**	11%
Other	**0**	0%

As pessoas podem marcar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

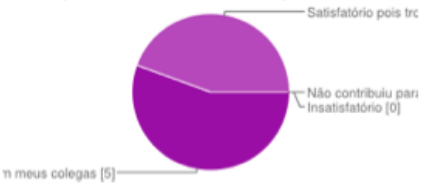
E agora, após este primeiro ano do desenvolvimento do SOLAR 2, das metodologias e instrumentos para gerenciamento de projetos de software você conhece?



Modelagem de sistemas utilizando UML	**5**	56%
Gerenciamento de Projetos de Software baseados em SCRUM	**9**	100%
Desenvolvimento de Software baseado em XP	**7**	78%
Desenvolvimento de Software baseado em Kan-Ban	**5**	56%
Ciclo PDCA (Plan, Do, Check and Act)	**7**	78%
Ferramentas CASE	**3**	33%
Other	**1**	11%

As pessoas podem marcar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

Na sua opinião o uso de Pareamento no processo de desenvolvimento de software foi:

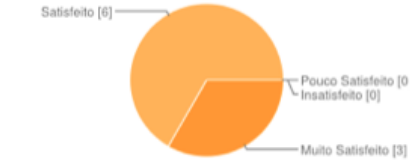


Muito Satisfatório pois permitiu que eu pudesse trocar experiências com meus colegas	**5**	56%
Satisfatório pois troquei um pouco de experiência com meus colegas	**4**	44%
Não contribuiu para nenhuma mudança perceptível em meu trabalho	**0**	0%
Insatisfatório	**0**	0%

O Pareamento no processo de desenvolvimento de software lhe serviu mais para:

Troca de experiência	**3**	33%
Conhecimento Global do sistema	**0**	0%
Troca de experiência com meus colegas e melhor entendimento do sistema como um todo	**6**	67%
Não serviu para nada significativo em minhas atividades	**0**	0%

Qual o seu grau de satisfação em ter trabalhado no Projeto Solar 2.0 ?

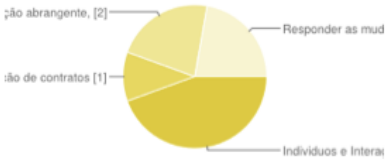


Muito Satisfeito	**3**	33%
Satisfeito	**6**	67%
Pouco Satisfeito	**0**	0%
Insatisfeito	**0**	0%

Em relação a qualidade de código :O Uso de Metodologias Ágeis para o desenvolvimento do Solar 2.0

Melhorou a qualidade do código	**8**	89%
Foi indiferente para qualidade do código desenvolvido	**0**	0%
Piorou a qualidade do código	**0**	0%
Other	**1**	11%

Em relação ao Manifesto Ágil : Diga qual dos valores você julga mais importante :



Indivíduos e Interações mais que processos e ferramentas	4	44%
Colaboração com cliente mais que negociação de contratos	1	11%
Software funcionando mais que documentação abrangente,	2	22%
Responder as mudanças mais que seguir um plano	2	22%

Justifique a escolha do Valor do Manifesto Ágil escolhido na pergunta anterior

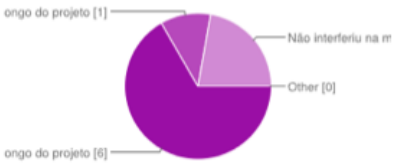
Acredito que com uma boa interação entre os indivíduos envolvidos no projeto e tendo todos o objetivo comum de fazer com que as coisas deem certo, as soluções para as dificuldades e problemas podem surgir independente das ferramentas utilizadas. O relacionamento entre os membros da equipe são um diferencial na escolha ágil devido a constante troca de experiências, é necessário um bom relacionamento, sem membros egocêntricos. Os requisitos mudam muito. Não tem como ter um plano que englobe todas as possibilidades. Então, o mais importante é a equipe responder bem às mudanças. O tempo para desen ...

Em relação a capacidade de trabalhar em equipe :O Uso de Metodologias Ágeis para o desenvolvimento do Solar 2.0

Melhorou a minha capacidade de trabalhar em equipe	6	67%
Foi indiferente para minha capacidade de trabalhar em equipe	2	22%
Piorou a minha capacidade de trabalhar em equipe.	0	0%
Other	1	11%

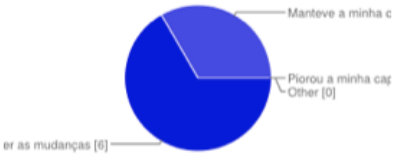
Em relação ao nível de desafio durante o projeto : O Uso de Metodologias Ágeis para o desenvolvimento do Solar 2.0

Conseguiu me manter desafiado e motivado ao longo do projeto	6	67%



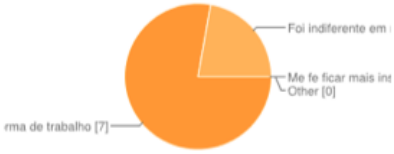
Não conseguiu me manter desafiado e motivado ao longo do projeto	1	11%
Não interferiu na minha motivação e nem no nível de desafio .	2	22%
Other	0	0%

Em relação a minha capacidade de responder a mudanças : O Uso de Metodologias Ágeis para o desenvolvimento do Solar 2.0



Melhorou minha capacidade de responder as mudanças	6	67%
Manteve a minha capacidade de responder as mudanças	3	33%
Piorou a minha capacidade de responder as mudanças	0	0%
Other	0	0%

Em comparação com outras formas de trabalhar que teve anteriormente : O Uso de Metodologias Ágeis



Me fez sentir mais satisfeito com a forma de trabalho	7	78%
Foi indiferente em relação a minha satisfação	2	22%
Me fe ficar mais insatisfeito do que com as formas que trabalhava anteriormente.	0	0%
Other	0	0%

A sua Motivação após trabalhar com Metodologias ágeis

Melhorou	**7**	78%
Piorou	**0**	0%
Não se alterou	**2**	22%
Other	**0**	0%

A sua satisfação pessoal em trabalhar no projeto:

Não afeta a qualidade do código desenvolvido.	**0**	0%
Melhora a qualidade do código desenvolvido.	**9**	100%
É indiferente a qualidade do código desenvolvido.	**0**	0%
Other	**0**	0%

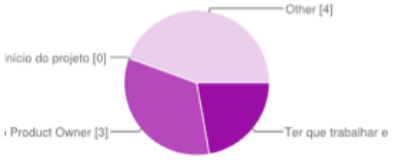
A sua satisfação pessoal em trabalhar no projeto:

Facilita a entrega de valor. Você estando satisfeito e motivado facilita entregar software mais próximo do que o cliente de

Não facilita a entrega de valor. Você estando satisfeito e motivado não facilita entregar software mais próximo do que o cl

Other

Quais os maiores problemas que mais atrapalharam o desenvolvimento do solar 2.0 ?



Ter que trabalhar em outros projetos ao mesmo tempo.	2	22%
Ausência do Product Owner	3	33%
Atraso no Início do projeto	0	0%
Other	4	44%

Como o desenvolvimento guiado a Comportamento (BDD) ajudou a melhorar o desenvolvimento do Solar 2.0 ?

Acredito que o BDD ajudou na visualização de como deve ser o comportamento de uma feature antes mesmo de ser implementada. Também ajuda a prevenir que erros criados a partir da implementação de outra feature deixem de ser percebidos. Outro fator positivo do BDD utilizado no Solar é o auto nível do código BDD escrito. A dificuldade do uso BDD no Solar é a complexidade da escrita do código dos steps necessários para verificação dos comportamentos nas páginas web. Na verdade ele estava atrapalhando. Não tinhamos conhecimento o bastante para desenvolver testes de compartamento e isso estava cons ...

Como a programação em par ajudou a melhorar o desenvolvimento do Solar 2.0 ?

A programação em par ajuda/ajudou em vários aspectos: - Concentração - Troca de experiências - Conhecimento do código - Dispensa a necessidade de revisão do código após o termino da feature É uma pena que nem todos os membros da equipe estejam dispostos a adotar essa prática ou até mesmo experimentá-la por um período seguindo as orientações dos livros de de palestras vistas. Realmente, uma pena. Através dos esclarecimentos de dúvidas, seja na regra de negócio, seja na linguagem, no modelo. O fato de uma discussão existente sobre o código feito contribui para a elucidação dessas dúvidas e to ...

Depois de ter participado do projeto Solar 2.0, diga quais expertises você adquiriu ou foram melhoradas durante o projeto

Fora muitas experiências que podem contribuir para trabalho em outros projetos. Muitas delas são práticas provenientes do SCRUM que ajudam a identificar problemas e procurar soluções. Não estou sabendo listas as expertizes, mas elas existem. - Trabalho em equipe - Gerenciamento de atividades - Nova linguagem e conceitos WEB - Testes Trabalho em equipe e organização. Metodologia Scrum e BDD Relacionamento interpessoal, melhoria no código produzido, testes, planejamento e proatividade. Trabalhar em par, em equipe, compreensão sobre Ruby on Rails Conhecimento sobre o git, experiência com um projeto ...

Que fatores lhe motivam a desenvolver software com mais qualidade e gerando satisfação do cliente ?

Acredito que a própria satisfação do cliente/usuários é uma grande motivação. Verificar que o software está tendo sucesso em cumprir o papel proposto. Reconhecimento. - Vontade de fazer um bom trabalho - Utilidade do sistema - Reconhecimento Um projeto desafiador, uma equipe competente e tempo para estudar novas tecnologias. - Menos erros no projeto, um design de código mais elegante e a satisfação de fazer um trabalho bem feito e que agrade ambas as partes. A satisfação pessoal de enfrentar um desafio interessante. Alcançar os objetivos do Solar, participar da produção de um sistema com a melho ...

Que benefício você que teve com o uso de metodologias ágeis para o desenvolvimento do Solar 2.0 ?

A priorização do que deve ser desenvolvido. A identificação rápida de impedimentos. O conhecimento por todos do trabalho de todos. Transparência. - Trabalho em equipe - Gerenciamento de atividades - Nova linguagem e conceitos WEB - Testes Melhoria na qualidade do código escrito e difusão de conhecimento entre a equipe. Melhor adaptação às mudanças de projeto; Melhor avaliação de prioridades para documentação: Saber o que e quando documentar é muito importante. Melhor relacionamento entre a equipe, mais agilidade na execução de funcionalidades, um melhor planejamento das atividades a serem real ...

Que sugestões você pode dar pra melhorarmos a forma de trabalhar na continuidade do projeto ?

Já que há uma certa resistência na adoção de certas práticas, seria interessante que todos estivessem dispostos a experimentá-las por pelo menos uma Sprint. Ao final verificar se a prática melhora a forma de trabalho ou não. Scrum Master e PO dedicados as suas funções também ajudaria bastante. - Rapidez nos aceites. - Maior foco no SOLAR Mais tempo dos integrantes no projeto e mais participação do PO. - Envolvimento 100% da equipe com o projeto e presença 100% de um PO. Incentivar mais o trabalho em par e fortalecer a presença do P.O. para direcionar o desenvolvimento. Melhor definir as funciona ...

Número de respostas diárias

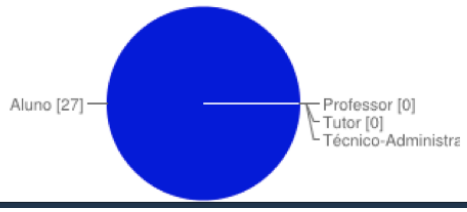
Apêndice C - Questionário de pesquisa com alunos de graduação e pós graduação

27 respostas

Resumo Ver as respostas completas

1. Dados personográficos

1.1 Você é:

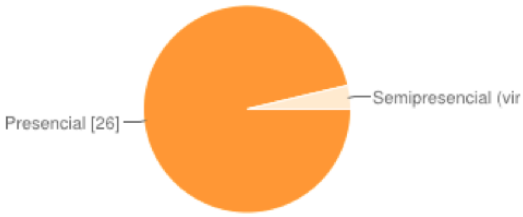


Aluno	**27**	100%
Professor	**0**	0%
Tutor	**0**	0%
Técnico-Administrativo	**0**	0%

1.2 Curso:

SMD Desenvolvimento Ágil SMD Especialização em Desenvolvimento Ágil de Software Desenvolvimento Ágil de Software Especialização em Desenvolvimento Ágil de Software Pós-Graduação em Desenvolvimento Ágil Esp ...

1.3 Qual a modalidade de seu curso?



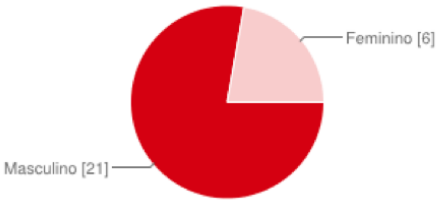
Presencial	**26**	96%
Semipresencial (vinculados à UAB/UFC)	**1**	4%

1.4 Idade (em anos completos):

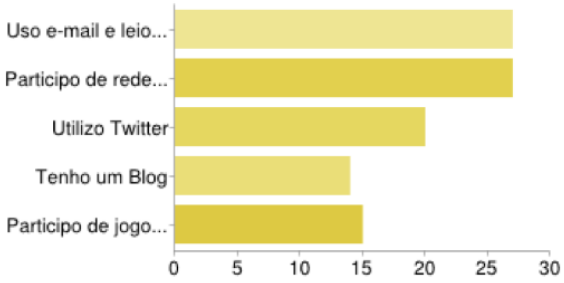
23 22 22 23 24 26 24 26 29 35 18 24 26 21 N/A 28 23 23 39 23 25 20 20 19 20 19 19

1.5 Sexo:

Masculino	**21**	78%
Feminino	**6**	22%



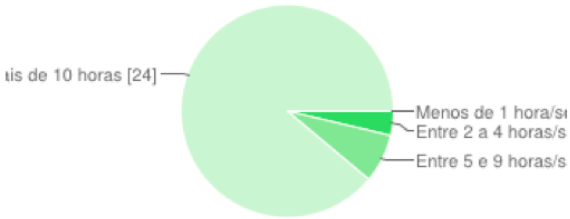
1.6 Qual é sua experiência com a Internet?



Uso e-mail e leio páginas da Web	**27**	100%
Participo de redes sociais (Ex.: Orkut, Facebook)	**27**	100%
Utilizo Twitter	**20**	74%
Tenho um Blog	**14**	52%
Participo de jogos on-line	**15**	56%

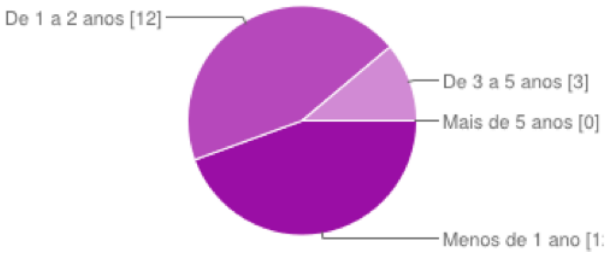
As pessoas podem marcar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

1.7 Quanto tempo por semana você acessa a Web?



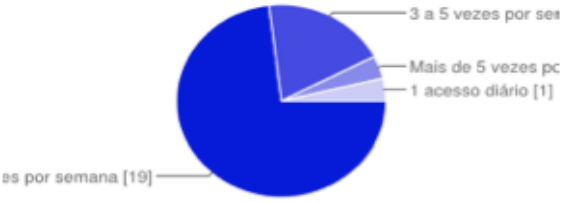
Menos de 1 hora/semana	**0**	0%
Entre 2 a 4 horas/semana	**1**	4%
Entre 5 e 9 horas/semana	**2**	7%
Mais de 10 horas	**24**	89%

1.8 Há quanto tempo você usa o SOLAR?



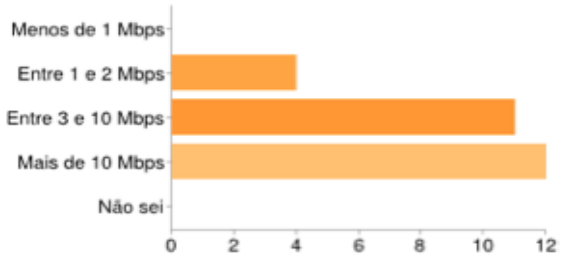
Menos de 1 ano	**12**	44%
De 1 a 2 anos	**12**	44%
De 3 a 5 anos	**3**	11%
Mais de 5 anos	**0**	0%

1.9 Com que frequência você costuma acessar o SOLAR?



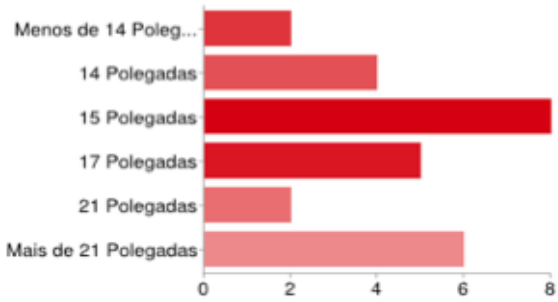
1 a 2 vezes por semana	**19**	70%
3 a 5 vezes por semana	**5**	19%
Mais de 5 vezes por semana	**1**	4%
1 acesso diário	**1**	4%

1.10 Qual é a velocidade de seu provedor de acesso à Internet?



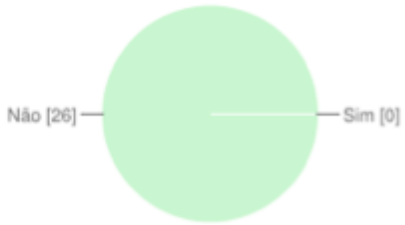
Menos de 1 Mbps	**0**	0%
Entre 1 e 2 Mbps	**4**	15%
Entre 3 e 10 Mbps	**11**	41%
Mais de 10 Mbps	**12**	44%
Não sei	**0**	0%

1.11 Qual é a o tamanho do monitor de seu computador?



Menos de 14 Polegadas	**2**	7%
14 Polegadas	**4**	15%
15 Polegadas	**8**	30%
17 Polegadas	**5**	19%
21 Polegadas	**2**	7%
Mais de 21 Polegadas	**6**	22%

1.13 Você é portador de algum tipo de necessidade especial?



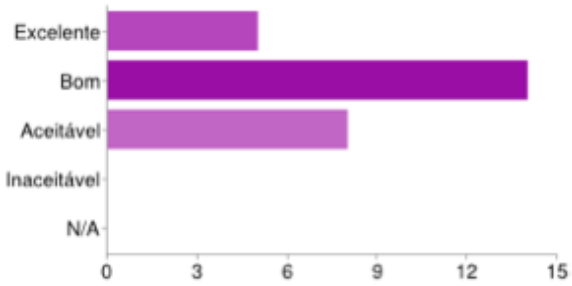
Sim	**0**	0%
Não	**26**	96%

1.14 Caso você tenha respondido "sim" na pergunta anterior, por favor, informe qual o seu tipo de necessidade especial:

Ainda não há respostas para esta pergunta.

2. Aspectos Gerais de Interface

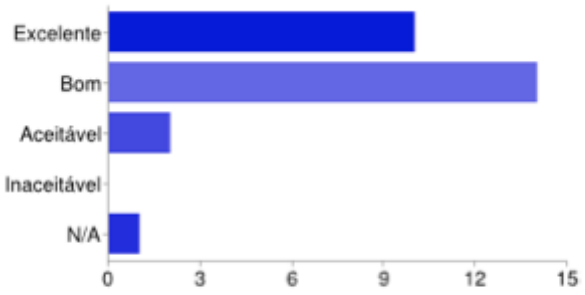
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.1 O design gráfico ((tipo de fontes, ícones, cores etc.) do ambiente SOLAR é atraente?



Excelente	**5**	19%
Bom	**14**	52%
Aceitável	**8**	30%
Inaceitável	**0**	0%
N/A	**0**	0%

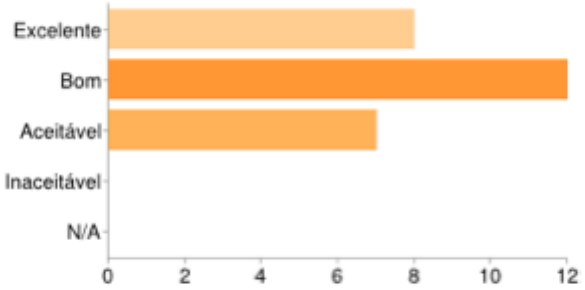
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.2 O ambiente apresenta o mesmo estilo de interface em todas as páginas?

Excelente	**10**	37%
Bom	**14**	52%
Aceitável	**2**	7%
Inaceitável	**0**	0%



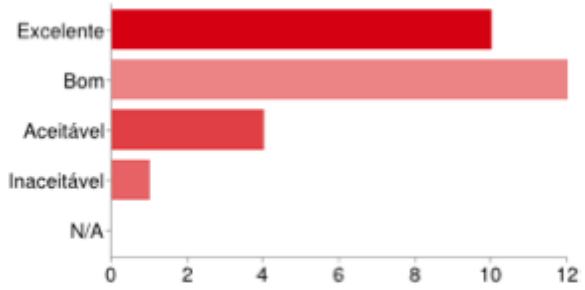
N/A	1	4%

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.3 As cores utilizadas são adequadas?



Excelente	8	30%
Bom	12	44%
Aceitável	7	26%
Inaceitável	0	0%
N/A	0	0%

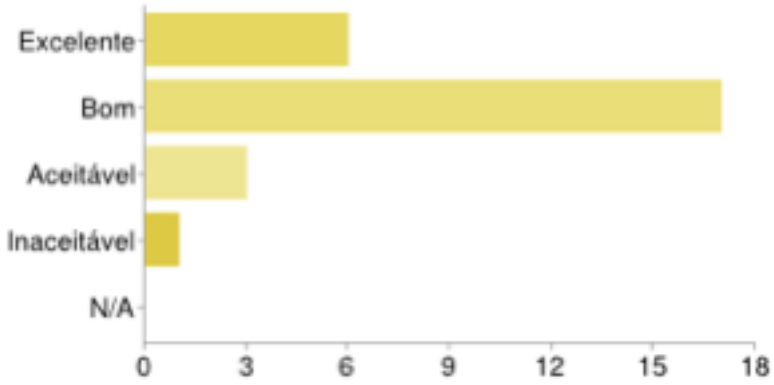
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.4 O esquema de contraste entre fontes e fundo favorecem a leitura?



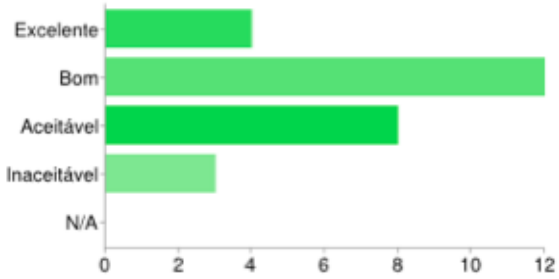
Excelente	10	37%
Bom	12	44%
Aceitável	4	15%
Inaceitável	1	4%
N/A	0	0%

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.5 O tamanho das letras e o espaçamento entre elas favorecem a leitura no ambiente?

Excelente	**6**	22%
Bom	**17**	63%
Aceitável	**3**	11%
Inaceitável	**1**	4%
N/A	**0**	0%

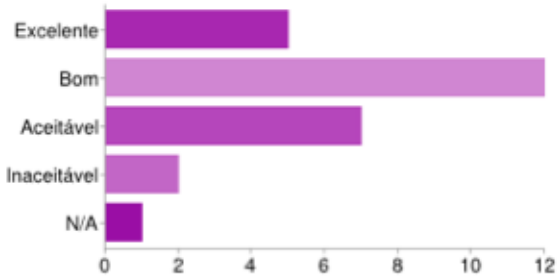


Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.6 Os elementos gráficos são úteis para que você possa se situar no ambiente (contexto) e por ele navegar?



Excelente	**4**	15%
Bom	**12**	44%
Aceitável	**8**	30%
Inaceitável	**3**	11%
N/A	**0**	0%

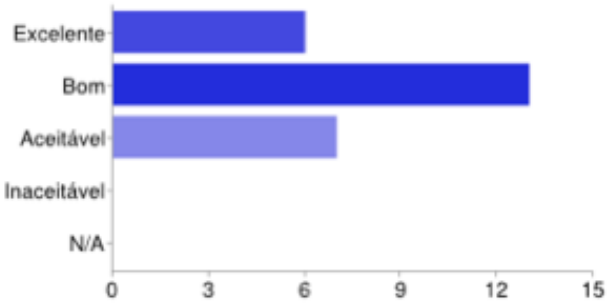
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.7 Os ícones e outras imagens têm significação clara?



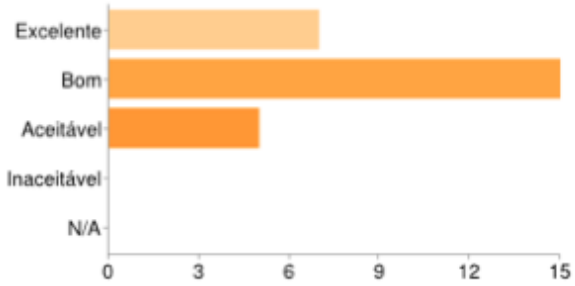
Excelente	**5**	19%
Bom	**12**	44%
Aceitável	**7**	26%
Inaceitável	**2**	7%
N/A	**1**	4%

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.8 O layout e uso de cores, fontes e imagens é consistente?

Excelente	**6**	22%
Bom	**13**	48%
Aceitável	**7**	26%
Inaceitável	**0**	0%
N/A	**0**	0%

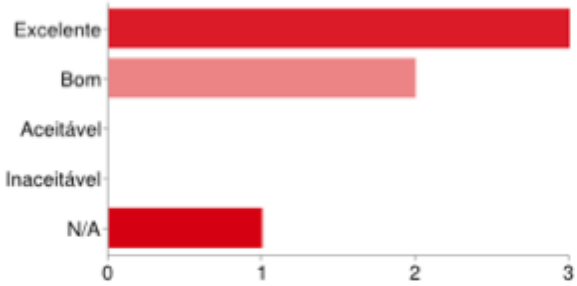


Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 2.9 A linguagem usada no ambiente é clara?



Excelente	**7**	26%
Bom	**15**	56%
Aceitável	**5**	19%
Inaceitável	**0**	0%
N/A	**0**	0%

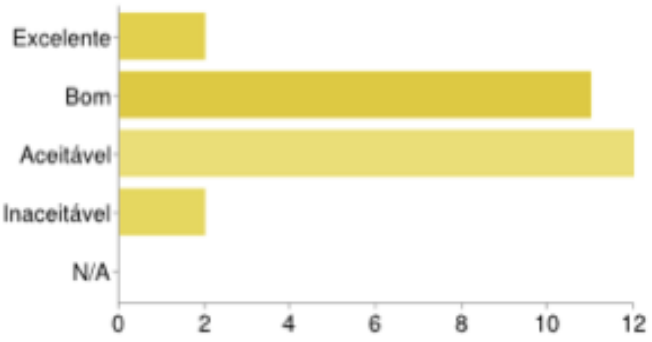
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: -



Excelente	**3**	11%
Bom	**2**	7%
Aceitável	**0**	0%
Inaceitável	**0**	0%
N/A	**1**	4%

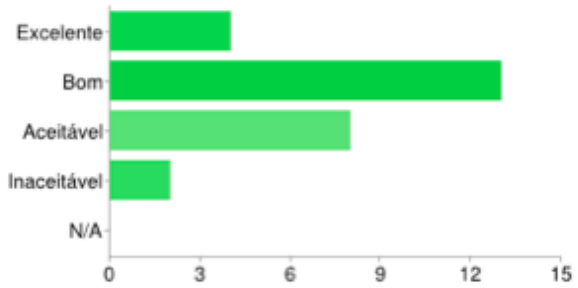
3. Navegação/Organização da Informação

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 3.1 A navegação no ambiente é intuitiva?



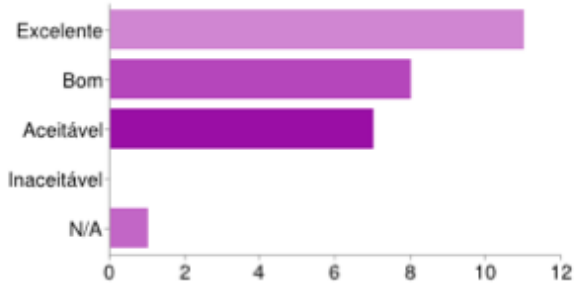
Excelente	**2**	7%
Bom	**11**	41%
Aceitável	**12**	44%
Inaceitável	**2**	7%
N/A	**0**	0%

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 3.2 A navegação no ambiente é rápida e requer no mínimo três cliques para se chegar à informação desejada?



Excelente	**4**	15%
Bom	**13**	48%
Aceitável	**8**	30%
Inaceitável	**2**	7%
N/A	**0**	0%

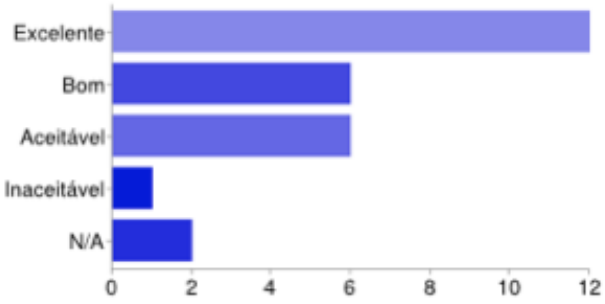
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 3.3 O usuário tem a possibilidade de ver o caminho percorrido?



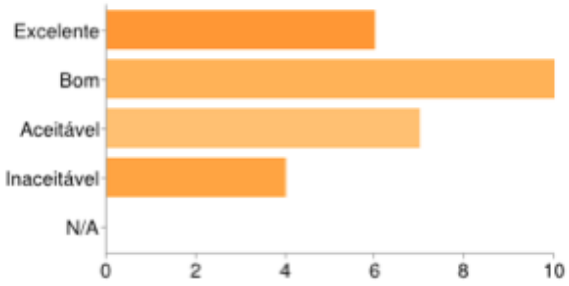
Excelente	**11**	41%
Bom	**8**	30%
Aceitável	**7**	26%
Inaceitável	**0**	0%
N/A	**1**	4%

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 3.4 A navegação permite voltar a níveis anteriores?

Excelente	**12**	44%
Bom	**6**	22%
Aceitável	**6**	22%
Inaceitável	**1**	4%
N/A	**2**	7%

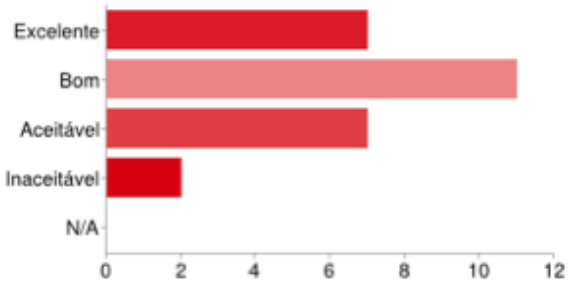


Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 3.5 Os menus são organizados de maneira lógica, ou seja, você consegue perceber como chegar em seu objetivo no sistema de forma clara e direta usando os menus?



Excelente	**6**	22%
Bom	**10**	37%
Aceitável	**7**	26%
Inaceitável	**4**	15%
N/A	**0**	0%

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 3.6 O ambiente apresenta a informação de maneira clara e em ordem lógica?

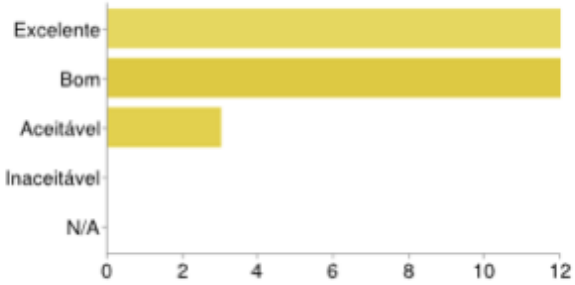


Excelente	**7**	26%
Bom	**11**	41%
Aceitável	**7**	26%
Inaceitável	**2**	7%
N/A	**0**	0%

4. Aspectos Gerais do Sistema

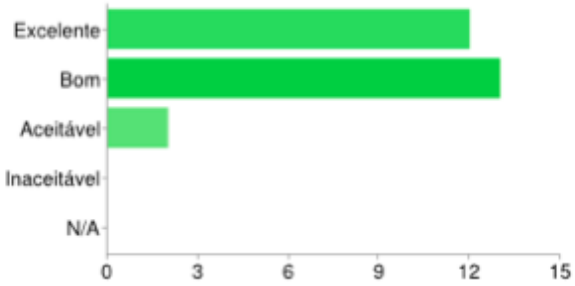
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 4.1 Qual a qualidade de resposta do sistema (agilidade, rapidez)?

Excelente	**12**	44%
Bom	**12**	44%
Aceitável	**3**	11%
Inaceitável	**0**	0%
N/A	**0**	0%

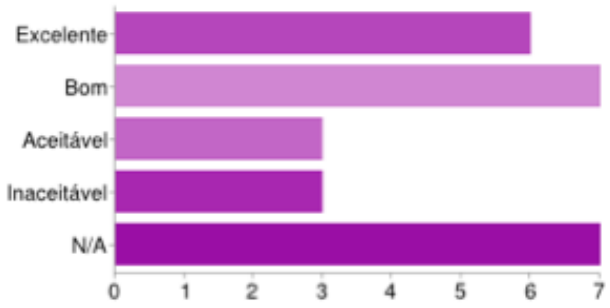


Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 4.2 Qual a qualidade do tempo de carregamento do sistema?

Excelente	**12**	44%
Bom	**13**	48%
Aceitável	**2**	7%
Inaceitável	**0**	0%
N/A	**0**	0%



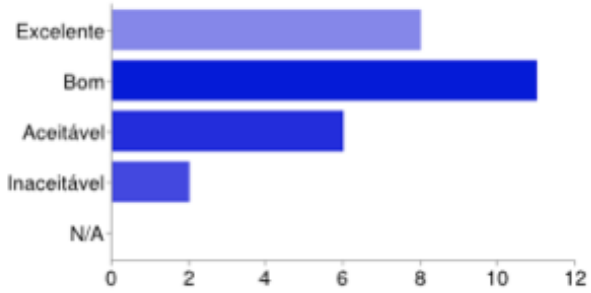
Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 4.3 Caso tenha encontrado algum erro, a notificação apresentada foi clara e de fácil compreensão?



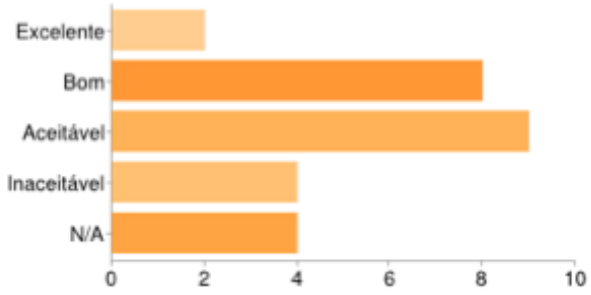
Excelente	**6**	22%
Bom	**7**	26%
Aceitável	**3**	11%
Inaceitável	**3**	11%
N/A	**7**	26%

Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 4.4 O uso das ferramentas do ambiente é de fácil compreensão ou requerem mais explicações?

Excelente	**8**	30%
Bom	**11**	41%
Aceitável	**6**	22%
Inaceitável	**2**	7%
N/A	**0**	0%



Por favor, assinale a opção que melhor corresponde à sua opinião, referente ao grau de satisfação com os itens descritos a seguir: - 4.5 O usuário encontra diferentes maneiras de executar uma mesma tarefa?



Excelente	**2**	7%
Bom	**8**	30%
Aceitável	**9**	33%
Inaceitável	**4**	15%
N/A	**4**	15%

5. Aponte o que você gosta no ambiente SOLAR 2.0

teste Foruns e portfólios Visual, navegabilidade, facilidade de achar as funcionalidades e uso de novas ferramentas modernas da Web. É um ambiente interativo que possíbilita aos integrantes da rede discutirem e trocarem ideias sobre diversos assuntos. A página inicial de cada disciplina que contém: Unidade Curricular, Responsável, Aulas, Mensagens (com contador das novas), Fórum e Agenda. Acho bem legal. Ambiente propício para atividades assíncronas. A forma de integração com o professor e demais colegas A interface é limpa e bem organizada. O uso das cores também deixa o visual mais limpo, já que ...

6. Quais os maiores problemas que ocasionam dificuldades de uso do SOLAR 2.0, segundo sua opinião?

teste não deixar acessar mais de uma disciplina ao mesmo tempo. Tive dificuldades de encontrar o que eu queria, algumas informações não estavam claras o suficiente. As funcionalidades do menu estavam um pouco confusas. Acredito que os menus em que estão distribuídos não são intuitivos. Interface, em alguns pontos, é pouco intuitiva. Em um primeiro acesso é difícil encontrar como acessar os matériais de um curso. Poderia ser acessado diretamente no menu de cursos oferecidos, caso eu esteja inscrito no curso. Acessar os comentários do professor no portfolio entregue. O ícone do balão fica com as ...

6. Quais os maiores problemas que ocasionam dificuldades de uso do SOLAR 2.0, segundo sua opinião?

teste não deixar acessar mais de uma disciplina ao mesmo tempo. Tive dificuldades de encontrar o que eu queria, algumas informações não estavam claras o suficiente. As funcionalidades do menu estavam um pouco confusas. Acredito que os menus em que estão distribuídos não são intuitivos. Interface, em alguns pontos, é pouco intuitiva. Em um primeiro acesso é difícil encontrar como acessar os matériais de um curso. Poderia ser acessado diretamente no menu de cursos oferecidos, caso eu esteja inscrito no curso. Acessar os comentários do professor no portfolio entregue. O ícone do balão fica com as ...

7. O que você mudaria ou o que falta no SOLAR 2.0?

teste Falta permitir o acesso a mais de uma disciplina. Possibilidade de dar um feedback de dentro do ambiente. Acredito que seria interessante rever a parte do menu, possibilitando mais clareza para o usuário encontrar o que deseja. Na minha opinião, em alguns casos parecia um "labirinto". Poder adicionar alguma mensagem ao professor ou poder respondê-lo após a entrega de um trabalho. A interface poderia ser mais limpa. Estilo os novos serviços do Google. Os dois casos registrados na questão 6. Integração com e-mail pessoal, informando acontecimento novo no ambiente. Não faria mudanças drásticas, ...

8. Caso tenha tido experiência com a versão 1.2 do SOLAR (versão anterior): Aponte o que está melhor na versão 2.0 do SOLAR, em comparação à versão anterior.

teste Cor: Aquele laranja me dava uma agonia, doía na vista e como provavelmente você que vai ler esses resultados da pesquisa sabe (hehehe), as cores possuem certa sinestesia, então creio que o laranja não foi bem empregado. O branco predominante com o azul são cores mais relacionadas com tecnologia, enfim, têm mais a ver com um ambiente de aprendizagem online. Navegação: Aqueles menus laterais do antigo eram confusos, eles não estavam agrupados de forma amigável. Era difícil achar a informação pois eles estavam todos à mostra, e quanto mais elementos estão visíveis, mais difícil é a identi ...

9 Caso tenha tido experiência com a versão 1.2 do SOLAR (versão anterior): Aponte o que deve ser mantido na versão 2.0 do SOLAR, em comparação à versão anterior.

teste Acho que não tem nada que se aproveite do solar antigo. :D N/A Nada. Parece simplista? Não é. Quando algo "tem um defeito de fábrica" como o Solar atualmente em uso, não adianta tentar "melhorar" o produto (é assim que algumas grandes empresa perdem bilhões) mas sim... desenvolver um NOVO produto (novo mesmo - sem referencias ou melhoramentos pontuais). As cores, por favor. Se possível nada! Uma reforma geral é o mais adequado. Envio de arquivos via forúns, assim como imagens. O menu lateral.

Número de respostas diárias