**Experiência de uso do *framework* Scrum no Desenvolvimento de um Sistema Integrado de Apoio a Projetos alinhado ao CMMI**

Eli Rodrigues – contato@elirodrigues.com

**Desafio**

Aplicar o *framework* Scrum para o gerenciamento de um programa de desenvolvimento de software, cujo objeto era um sistema integrado de apoio a projetos alinhado ao CMMI.

**Organização**

A equipe era composta por 12 profissionais dos perfis de analista, programador, testador e líder de configuração. Foram organizados grupos menores para desenvolver os softwares em paralelo. *Scrum* *Master (SM)* e testadores eram compartilhados entre as equipes.

O tamanho das equipes variou conforme a fase de projeto, nas fases preliminares havia analistas e testadores, Já na fase de codificação apenas os desenvolvedores com participação pontual dos demais. O gerenciamento das alocações foi facilitado, pois o *SM* tinha autonomia para decidir onde e quando alocar cada profissional.

Os projetos eram internos e o *Product Owner* (PO) representava o departamento de Qualidade, maior interessado nos projetos, isto permitiu sua participação contínua nas reuniões de revisão do *product backlog* e até nas *daily meetings,* quando necessário.

As cerimônias seguiram o fluxo normal do Scrum, com definição e priorização de requisitos feita pelo *PO,* planejamento dos *sprints* realizado pela equipe e reuniões de acompanhamento. Algumas adaptações foram realizadas.

**Projetos**

Todos os projetos usaram *Scrum* e compartilharam um pool de recursos humanos. Também usavam a mesma tecnologia (.NET e SharePoint), exceto o projeto B, que foi construído em Java. Os testes realizados foram funcionais e manuais, havia dois testadores na equipe, que se alternavam entre os projetos. Os softwares desenvolvidos foram:

|  |  |  |  |
| --- | --- | --- | --- |
| Projeto | Nome | Descrição | Sprints |
| A | Cadastro de Projetos | Cadastro das características dos projetos, que serviu como base associativa aos outros softwares | 3 |
| B | Rastreabilidade de requisitos | Cadastro do ciclo de vida dos requisitos, da necessidade à tela implementada | 3 |
| C | Controle de versão | *Upload* de documentos e permitia a edição integrada com a ferramenta, garantindo o versionamento, estabelecimento de *baselines* e aprovações | 5 |
| D | Gestão da qualidade | Máquina de estados para cadastro de não-conformidades de todos os tipos, incluindo *bugs* | 1 |
| E | Controle de mudanças | Registro de solicitações de mudança no escopo de projetos, com campos para análises e aprovações. | 2 |
| F | Apontamento integrado | Por ser uma empresa com estrutura projetizada, todos os funcionários apontavam horas em uma planilha. O software foi criado para substituir a planilha, integrando todos os projetos e gerando relatórios. | 5 |
| G | Integração | Houve um esforço de ajustes em base de dados, interfaces e padrões de codificação, que foi nomeado “integração” | 1 |

**Planejado x Realizado**

A ordem cronológica executada divergiu bastante da planejada, devido a construção iterativo-incremental e a impedimentos ocorridos. Alguns projetos como A e G, não estava no planejamento inicial e a percepção de necessidade ocorreu apenas com o uso dos demais softwares. O quadro abaixo mostra as diferenças entre os prazos planejados e realizados:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Projeto** | **Meses** | | | | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| A |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | NP | NP | NP |  |  |  |  |
| B |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| F |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| G |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | NP |

Planejado Executado NP = Não planejado

Já o projeto B usava uma tecnologia (Java) que não era de domínio do grupo, apenas de uma pessoa do grupo, que acabou adoecendo e, por falta de recursos, o projeto ficou parado por 2 meses.

No “projeto C” foi necessário contratar uma consultoria para capacitar a equipe, foram cerca de 500 horas de apoio, o que trouxe um custo adicional significativo, mas gerou um “framework” de desenvolvimento útil aos demais projetos.

As atividades de capacitação, consultoria, codificação conjunta e revisão de código foram registradas nos *sprint backlogs*, que inclusive foram utilizados como insumo para os pagamentos do fornecedor (contrato de tempo e material). Os consultores participaram apenas de algumas *daily meetings*, pois não ficavam alocados em tempo integral na empresa.

O “projeto D” tinha requisitos simples e utilizou o framework gerado pelo “projeto C”, o que acabou reduzindo seu tempo pela metade e reduzindo também o processo de correção de bugs.

O projeto “E” foi realizado dentro do planejado, sem anormalidades relevantes.

No projeto “F” houve um problema de gerenciamento de expectativas, o cliente do software era a pessoa responsável pela coleta das planilhas de horas, que cultivava a expectativa de receber o software “redondo” na primeira iteração. O mesmo aconteceu com as pessoas que participaram da execução-piloto, houve muitas reclamações sobre a usabilidade em termos de interface e fluxo de operação, além da ausência (na primeira iteração) dos relatórios esperados. Apesar disso, os dados coletados no piloto e nos testes de aceitação foram reinseridos no processo de desenvolvimento e implementados nas iterações seguintes.

**Adaptações realizadas**

|  |
| --- |
| **Adaptação #1:** Houve *sprints* que não geraram nenhum produto funcional, mas documentação de requisitos e casos de testes a serem validados pelo *PO*.  **Como é na teoria:** Cada *sprint* deve gerar um produto funcional  **Impactos:**   * Positivos: Puderam-se definir as interfaces entre os sistemas antes de iniciar * Negativos: O primeiro projeto atrasou 2 meses e mesmo com as entregas intermediárias, houve impacto no trabalho de implantação do modelo CMMI na empresa, que era o objetivo final de todo projeto. |
| **Adaptação #2**: O quadro de acompanhamento foi construído em uma planilha, de modo que a própria equipe pudesse fazer as atualizações de “horas restantes” antes da *daily meeting*. Também lhes era permitido inserir novas atividades livremente, fazendo comentários nas reuniões. O gráfico de *burndown* era atualizado automaticamente pela planilha.  **Como é na teoria**: Atualizar o quadro de acompanhamento durante a *daily meeting*.  **Impactos**:   * Positivos: Agilidade e análise do todo durante a reunião * Negativos: Eventualmente algum recurso precisava discutir com o grupo antes de atualizar |
| **Adaptação #3**: As reuniões de acompanhamento aconteceram duas vezes por semana para cada projeto. No entanto, a equipe tinha acesso ao *SM* para reportar impedimentos a qualquer instante.  **Como é na teoria**: As reuniões devem acontecer diariamente  **Impactos**:   * Positivos: Foi possível reunir em separado as equipes de cada projeto e com isso ter maior visão técnica, possibilitando integrações e reusos. * Negativos: Eventualmente algum recurso entrava em impedimento e não reportava ao *SM.* |
| **Adaptação #4:** Em ocasiões específicas foram permitidas alterações nos requisitos ao longo dos *sprints*. Exemplos: Alteração de layout de tela, ajuste de fluxo de trabalho, inserção de um novo campo.  **Como é na teoria:** Mudanças de requisitos ocorrem no nível de *Product Backlog,* antes do planejamento dos *sprints*.  **Impactos:**   * Positivos: Não precisar cancelar o *sprint* ou esperar o próximo para realizar uma alteração simples. * Negativos: Cliente passou a solicitar alterações ao longo dos sprints. |
| **Adaptação #5:** Adição de *buffer* nos *sprints* para correção de bugs e pequenas alterações de requisitos.  **Como é na teoria:** Não especifica  **Impactos:**   * Positivos: Permitiu que houvesse tempo suficiente para correção dos *releases* anteriores e viabilizou pequenos ajustes de requisitos. * Negativos: Nenhum observado. |

**Benefícios observados**

O framework Scrum possui como característica mais forte a adaptabilidade às mudanças para atendimento das necessidades dos clientes. A tradução da palavra *framework* é arcabouço, que significa uma espinha dorsal. Isso deixa claro que são permitidas adaptações e adições também no processo de trabalho.

Os benefícios observados ao longo destes projetos foram:

* Capacidade de mudar o planejamento dos projetos ao longo do desenvolvimento, incluindo novos módulos conforme a necessidade foi observada.
* As entregas intermediárias ao longo do programa permitiram que os clientes e usuários interagissem com a equipe, dando novos *inputs* para melhoria do produto
* Controle e Visibilidade do todo através do conceito de *time Box*
* Comprometimento do cliente na criação do “melhor produto possível” dentro do custo aceitável
* Oportunidade de aprendizado contínuo e incremental da equipe (como um módulo influenciou no desenvolvimento de outro)
* Comprometimento e participação da equipe como verdadeiros “donos” do negócio, interagindo com clientes, usuários, sugerindo mudanças e fazendo os *sprints* acontecerem.
* Desenvolvimento de novos líderes dentro das equipes

Por ter uma “fórmula” simples de implantação, o Scrum viabiliza que conceitos importantes da engenharia de software sejam aplicados de forma empírica, como a gestão de requisitos, controle de qualidade, planejamento e monitoramento de projetos, estimativas de esforço e prazo e até métricas. Todas estas práticas fazem parte do modelo CMMI e funcionam na prática usando-se o Scrum.