

## Princípios LEAN aplicados ao desenvolvimento de software

O objetivo deste artigo é mostrar como os princípios Lean, empregados pela primeira vez em otimização de processos na indústria automobilística japonesa, podem contribuir para o desenvolvimento de software.

Os problemas que acontecem no desenvolvimento de software costumam se repetir nos mais variados tipos de projetos e de organizações no mundo todo. Existe a percepção, correta, de que pessoas capacitadas atuando em um processo adequado produzem um produto (software) de qualidade. Mas o que é um processo adequado?

Podemos considerar três linhas de processos: tradicional (em cascata), unificado (RUP-like) e ágil (XP é o exemplo mais típico).

O processo unificado foi o primeiro a consolidar três características que são extremamente benéficas e que estão incorporadas em todos os processos mais modernos: iterativo, incremental e "time-boxed".

O processo tradicional tem sido formalmente abandonado, embora as empresas o pratiquem sob outros rótulos.

Seus maiores problemas são:

- Assumir que pode se saber tudo o que se quer do software num primeiro momento.

- Deixar os testes para os últimos momentos.

O processo unificado se divide em iterações (pequenos pedaços de tempo) nas quais são gerados incrementos de software pronto (construído, testado, integrado).

As iterações são "time-boxed", ou seja: as datas de término previsto não podem ser adiadas e representam uma oportunidade de atuação gerencial que coloca em prática o modelo: planeja, executa, mede, aprende.

Este processo permite um aprendizado crescente dos analistas em relação ao negócio e dos clientes em relação ao sistema.

Os processos mais ágeis são iterativos, incrementais e "time-boxed". Mas, em busca de maior rapidez nas entregas, eliminam parte do formalismo na mesma medida em que agregam maior disciplina. Não contrariam radicalmente o processo unificado, mas enfatizam alguns pontos de maneira diferente; não seria errado, portanto, pensar nestes processos como sendo uma "leitura ágil" do processo unificado.

O interessante na fotografia atual da indústria mundial de software é que as organizações estão buscando o posicionamento adequado de seus processos ("aqui se faz assim...") situando-se em algum ponto intermediário.

- Processos "RUP-like" estão se tornando mais "leves". De maneira a serem mais ágeis.

- Processos ágeis estão buscando práticas que permitam atacar problemas e projetos mais amplos, para além dos casos de sucesso iniciais.

Mas onde encontrar os princípios que podem nos ajudar a tornar nossas práticas de desenvolvimento de software adequadas a este novo entendimento resultante de algumas décadas de procura?

Na forma Lean de pensar processos.

As origens do Lean vêm da indústria e tratam essencialmente de três palavras e conceitos:

"Muda": desperdício, aquilo que não agrega valor ao produto;

"Mura": operação não linear;

"Muri": estresse de pessoas e equipamentos.

Operação não linear (um pico de demanda, por exemplo) gera estresse, o que baixa a qualidade, o que exige mais operação não linear para corrigir falhas; sendo que tudo isso não só afeta nossos esforços de eliminar desperdício como aumenta o desperdício pela utilização de iniciativas desnecessárias (mais controles extras, por exemplo).

Os princípios Lean aplicados ao desenvolvimento de software geram um processo semelhante ao de desenvolvimento de um produto:

1. Descobrir o quê;
2. Descobrir como; e
3. Construir.

Seus objetivos são entregar produtos que agregam valor para o negócio mantendo a habilidade de continuar adicionando valor rapidamente no futuro.

O processo resultante é um fluxo de produção rápido e flexível (Fast, Flexible Flow) e que tem as seguintes características:

- Desenvolve uma visão de produto, a ser entregue através de releases,

- Cada release acontece de maneira iterativa e incremental,
- Cada iteração é "time-boxed".

Os princípios Lean que devem ser entendidos e aplicados para se encontrar um processo adequado são os seguintes:

### 1. Respeite as pessoas

- Os softwares são construídos por pessoas trabalhando em equipes, portanto é necessário integrar pessoas das diversas áreas envolvidas, facilitar a comunicação e a colaboração entre elas e mantê-las motivadas;
- É importante acreditar que as pessoas são boas e querem entregar um produto de qualidade, portanto é necessário dar condições para que realizem seu trabalho;
- Os erros ocorrem mais frequentemente por defeitos de processo do que por falha humana num processo correto;
- As pessoas devem estar capacitadas e autorizadas para customizarem o processo (on-the-fly) conforme as necessidades.

### 2. Elimine desperdício

- Desperdício é tudo que não agrega valor ao produto ou que não prepara para uma próxima agregação de valor;
- Uma fonte comum de desperdício é a comunicação entre equipes;
- Outra fonte é construir mais do que foi solicitado ou agregar complexidade às soluções de tecnologia;
- A correção de erros também é desperdício, não podemos deixar de corrigi-los, mas estamos aumentando o custo de um valor que já deveria ter sido agregado. Os erros devem ter sua causa de processo investigada;
- Controles e documentos devem ser bastante analisados para verificar sua contribuição. Deve-se trocar formalismo por disciplina;

### 3. Postergue compromissos

- O processo de desenvolvimento de software é muito sujeito a mudanças. Deve-se aceitar que o usuário só conhece 20 a 25% dos requisitos, mas que este valor certamente representa o "core" do sistema;
- Implementando este "core" se terá oportunidade para descobrir o restante (vide princípio seguinte);
- Sobre requisitos: é necessário ter a visão de produto (ampla, mas não detalhada) e depois efetuar just-in-time

requirements ao longo do ciclo de desenvolvimento;

Sobre arquitetura: é indispensável, mas deve se praticar Emergent Design que permite elaborar uma arquitetura flexível sem necessidade de antecipar definições sobre as futuras demandas.

### 4. Crie conhecimento

- Pela utilização de um processo iterativo e incremental se promove a aprendizagem de todo os envolvidos (usuários, desenvolvedores, etc.) no projeto.

### 5. Entregue rapidamente

- Deve se entregar software rápido, mas isto se obtém mais com a eliminação de desperdícios e a manutenção do time em um ritmo adequado de produção do que satisfazendo a ansiedade de "sair fazendo...". Deve se produzir em uma "velocidade de cruzeiro";
- Deve se planejar um release de maneira a entregar um produto parcial que Possua "Minimal Marketable Features".

### 6. Construa com qualidade (embutida)

- As pessoas não gastam muito tempo corrigindo erros, gastam muito tempo achando erros!
- É um desperdício muito grande gastar esse tempo nas últimas iterações;
- As tarefas de testes devem começar muito cedo e seguir em paralelo com o desenvolvimento;
- Quem define os requisitos define os critérios de testes da implementação desses requisitos.

### 7. Otimize o todo

- Qual a experiência do usuário: os seis meses desde a solicitação até a entrega ou os 40 dias de programação?
- Todas as partes do processo devem ser analisadas e otimizadas;
- Deve-se usar VSM-Value Stream Maps, identificando o fluxo de tarefas, o tempo de trabalho e o tempo decorrido em cada passo, o tempo de espera entre passos e o tamanho da fila para cada passo. Assim é possível calcular a eficiência do processo.

Resumindo, os benefícios da aplicação de princípios Lean ao desenvolvimento de software são:

- Agregar valor rapidamente ao negócio
- Obter clareza sobre as necessidades do cliente

- Melhorar a gerência dos projetos
- Dar maior visibilidade sobre o andamento dos projetos
- Desenvolver com foco no produto
- Manter o time motivado

Para obter esses resultados é necessário vencer alguns desafios

- Obter maior envolvimento do cliente
- Clientes fazem parte do time
- Aceitar as consequências do iterativo e incremental
- O processo tradicional em cascata dá uma falsa certeza inicial com a qual as organizações estão acostumadas
- Praticar maior disciplina
- Para eliminar o desperdício representado por formalismos e controles é necessário o compromisso com a comunicação e a disciplina
- Aceitar "planejar" mais
- Na prática se planeja mais frequentemente que no tradicional: plano do produto, plano do release, plano da

iteração, revisão diária das tarefas do dia (plano "on-the-job")

- Trabalhar com um time "de times"
- Pessoas se intercomunicando o tempo todo buscando um objetivo comum conhecido

No caso de se terceirizar partes do processo, os seguintes desafios devem ser enfrentados:

- Adaptar os contratos ao processo
- Pode ser difícil, mas vale lembrar que o que se quer é software funcionando..., não o pagamento de multas.
- Resolver os problemas que se apresentarão nas "passagens de bastão"
- É utópico pensar que se pode desenvolver uma "especificação" 100% de forma a poder cobrar uma "implementação" 100%; desenvolvimento de software é um processo de descoberta!

**Continuação:** num próximo artigo abordaremos os princípios do SCRUM dentro da perspectiva de utilizá-lo no gerenciamento das iterações de um processo definido com base nos princípios Lean descritos neste artigo.

Por: Domingo Chabalgoity

