Guia da Modernização Consciente (Prévia)

Mário Melo

Published with GitBook

Tabela de conteúdos

Sobre a Prévia

Embora esta não seja a versão final do Guia da Modernização Consciente, o conteúdo mais importante já está aqui. E por ser um conteúdo de grande importância, resolvi pedir uma opinião ainda mais importante: a sua.

A versão final provavelmente terá um layout diferente, um prefácio, agradecimentos, os dois últimos tópicos que por enquanto seguem em desenvolvimento e alterações sugeridas por você.

Então, agradeço imensamente cada segundo despendido com a leitura deste guia e qualquer tipo de *feedback* que você possa fornecer.

Um abraço,

Mário

A culpa é do sistema

A culpa é sistema; essa entidade abstrata que engessa os processos da empresa e impede os funcionários de executarem seu trabalho a todo vapor. Mas esse culpado foi originalmente concebido para atingir objetivos completamente opostos, então como chegamos até esse ponto?

Quando comecei a desenvolver sistemas em 2001 percebi que o processo de concepção de um software era muito diferente do utilizado em outras áreas. Tudo acontece de maneira extremamente aberta, e as únicas limitações são o tempo e o dinheiro de quem encomenda o sistema. Sim, somente isso: **tempo e dinheiro**.

Quanto vale uma modernização?

> Por que uma empresa investe 20 milhões de dólares em um ERP?

A resposta é óbvia: **porque ela acredita que vai ganhar mais do que 20 milhões adotando esse ERP.** Como você deve imaginar, essa é uma aposta muito alta para ser feita baseada em um simples palpite. Por isso, o processo de compra de um software desse porte é extremamente complexo: **é necessário validar o retorno esperado do investimento**.

> Mas se as empresas validam o retorno do investimento, como chegamos em um sistema que não atende nossas necessidades?

Software é abstrato. E sempre há espaço para melhorias. Mas muitas dessas melhorias não justificam o investimento necessário para realizá-las. Quando pensamos racionalmente sobre isso, tudo parece muito simples. Afinal, estamos acostumados a comprar produtos e serviços, certo? Mas na maioria das situações em que nos vemos como consumidores, o processo de compra é muito diferente. Mesmo quando estamos adquirindo um serviço. O software é abstrato demais, sem nenhuma forma pré-definida. Isso nos dá uma sensação de liberdade infinita para sonhar com o produto final, e aí o foco se perde.

Mas de quem é a culpa afinal?

Sempre que um cliente chega até mim com um plano para desenvolver um software eu tento argumentar contra a criação do mesmo. E já convenci vários clientes a mudar de idéia, mesmo isso significando que minha empresa não lucraria com o desenvolvimento do projeto. Parece estupidez, mas não é. O portfólio de minha empresa deve conter sistemas que trouxeram benefícios reais para seus usuários.

Afinal, se a culpa realmente é do sistema, ela também é de quem o desenvolveu.

Os 5 Fracassos de uma Modernização

Como desenvolvedor de software, já participei de muitos projetos incríveis que acabaram fracassando. Hoje, analisando friamente esses casos consigo enxergar certos padrões e trabalhar para que eles não se repitam no futuro.

> Por quê um projeto tão incrível termina em um sistema tão medíocre?

Dificilmente a resposta será falta de capacidade técnica da equipe de desenvolvimento. Afinal, se for essa a resposta a solução é simples: **trocar o fornecedor**. Mas o problema geralmente está na concepção do software: expressar suas necessidades em software é algo muito passível de erros.

Trocar o fornecedor é uma atitude que exime a empresa contratante de qualquer culpa nos erros do processo de desenvolvimento de software. E se não somos capazes de detectar erros em partes do processo que podemos controlar, estamos assumindo nossa incapacidade de corrigí-los.

É muito comum uma empresa trocar o fornecedor e continuar com os mesmo problemas de antes. Ou aumentar o número de problemas. E isso acontece porque algum dos padrões estabelecidos nesse capítulo está acontecendo.

O Fruto do Orgulho

A maioria de nós utiliza algumas dezenas de softwares todos os dias: Windows, Word, Excel, Firefox, Safari, Android, iOS, Google Maps, Facebook, Twitter. E só existe um motivo para fazermos isso: **ganhar tempo**.

Você pode argumentar e dizer que o Facebook te dá algo mais humano, como contato com familiares distantes. Mas ele na verdade faz você se comunicar com eles de maneira mais fácil e rápida, te poupando tempo. Foi assim que o Whatsapp ganhou espaço no mercado: poupando mais tempo dos usuários do que seus concorrentes.

Para uma empresa, tempo é dinheiro. Parece clichê, mas não é. E uma empresa compra sistemas economizar tempo, ou para:

- **Vender mais** ou fazer seu vendedor realizar mais vendas em menos tempo
- **Produzir mais** ou fazer seu funcionário gerar mais em menos tempo
- **Reduzir custos** ou reduzir o tempo de trabalho necessário para cobrir estes custos

No entanto, é bastante comum alguém dizer que precisa de um software capaz de *reforçar determinada regra da empresa*, *organizar melhor um setor* ou *tratar uma situação de exceção*. Quando essa solicitação vem de um nível hierárquico mais alto, o software desejado pode não ser a solução para um problema crônico, e sim o **fruto do orgulho** e alguém.

Como dito na introdução, **uma empresa sempre investe em modernizações esperando ter um retorno maior que o montante investido**. Quando um software é meramente um **fruto do orgulho**, as pessoas envolvidas no projeto têm dificuldade de explicar o seu objetivo final. É bem fácil evidenciar esse padrão através dos *5 Porquês*: pergunte ao responsável o porquê do software precisar ser desenvolvido, escute a resposta, e pergunte o porquê novamente mais 4 vezes. Se ele for capaz de responder todos os questionamentos, o software provavelmente tem uma motivação sólida.

Fruto do orgulho de um, fruto da vergonha de outro

Outro ponto extremamente negativo nesse padrão é que geralmente os

envolvidos no projeto não se orgulham tanto do projeto quanto seu idealizador. Essa desmotivação por parte dos envolvidos faz com que cada um tente justificar o projeto à sua maneira, e o que antes não tinha um foco passa a ter vários. Assim, ao invés de todos trabalharem em direção a um único objetivo, cada um puxa o projeto para uma direção. E ele não sai do lugar.

> Se alguém disser que precisa de um software ou modernização de processos, pergunte o porquê. Cinco vezes.

A (eterna) busca pela perfeição

Não é incomum encontrar empresas com projetos de software que estão na fase final de desenvolvimento há mais de um ano. Eu sei, você deve estar pensando: "*Nada pode estar na fase final de desenvolvimento se ainda há um ano de trabalho pela frente!*". E você acertou.

Chega um momento em todo projeto de software onde a única maneira de aprender mais sobre o mesmo é escutando seus usuários. Ver o software em execução e corrigir as falhas encontradas é geralmente mais eficaz do que ficar especulando o que pode dar errado. Na maioria dos casos, implantar um sistema que ainda não está totalmente pronto já gera algum ganho para a empresa e ainda permite que os usuários colaborem com seu desenvolvimento.

> Não se preocupe com a perfeição, você nunca irá conseguí-la. - ***Salvador Dalí***

"*Esse sistema entrará em produção daqui há 6 meses, e então teremos um período de ajustes e estabilização*". Isso significa que nos próximos 6 meses a empresa seguirá sem modernizar seus processos simplesmente porque a solução não é boa o suficiente. E o período de estabilização tende a ser grande porque os usuários jamais tiveram contato com o software, e portanto precisarão de tempo para conhecê-lo por inteiro.

Quando os usuários têm a chance de experimentar o sistema antes do fim do desenvolvimento existe a possibilidade de algumas funcionalidades serem descartadas. E podem surgir outras também. Alterações de escopo são perfeitamente normais e apenas mostram que a organização evoluiu desde a data em que o projeto foi planejado. **Não lute contra a evolução de sua organização**.

Para identificar se um projeto se encaixa nesse padrão basta se perguntar: "*alguém teria algum ganho caso o software fosse implantado do jeito que está?*". A resposta geralmente é sim. E se ninguém vai sair prejudicado, por que adiar a implantação?

> Durante a execução de um projeto de modernização, muita coisa pode mudar. Dê ao projeto a chance de se adaptar o mais cedo possível.

O benchmaring equivocado

> A empresa X tem um brinquedo novo. E eu quero um igual.

Essa frase resume bem o significado de **benchmarking** **equivocado**. Antes de copiar aquilo que acreditamos ser a chave do sucesso alheio, precisamos entender melhor o contexto desse sucesso.

Softwares personalizados são muito mais do que simples produtos de prateleira. São frutos de um estudo prévio que muitas vezes envolve consultorias e até mesmo alguns anos de pesquisa. Para entender o sucesso de uma solução adotada por uma determinada empresa é necessário entender também a sua cultura, afinal, muitas vezes um software é adotado para reforçar um processo que já existia por ali há algum tempo. Comprar ou desenvolver um software igual e esperar o mesmo desempenho é como comprar um carro de fórmula 1 e acreditar que isso lhe transformará no Schumacher.

Um exemplo clássico

A **S.C.Johnson** era líder no mercado de ceras para pisos nos Estados Unidos. Ao tentar exportar esse mesmo produto para o Japão, esqueceu-se de considerar as diferenças culturais. Simplesmente copiou e colou. Mas os japoneses não usam sapatos dentro de casa, e a cera deixava o piso muito escorregadio. Resultado: fracasso nas vendas do mesmo produto que era um sucesso nos Estados Unidos.

Quando falamos de software, falamos implicitamente das pessoas que o estão utilizando. Não são as linhas de código que o concorrente comprou que o fazem mais produtivo; é o que os trabalhadores de lá conseguem extrair do software adquirido.

> Benchmarkings são importantes, mas devem ser levados a sério desde o início. Questione respostas simples sempre que elas existirem.

A Restrição ao Especialista

Fornecedores de software e consultorias são contratados para suprir uma necessidade da contratante: a falta de *know-how* em uma atividade que não é o foco da empresa.

É algo semelhante a contratar um pintor ou pedreiro para reformar sua casa. Nessa situação temos uma ideia do que queremos, mas os conselhos dos especialistas à respeito de qual tinta ou argamassa usar são fundamentais. Podemos até ignorá-los e comprar a tinta mais barata, mas corremos o risco de não ter o resultado final tão bom quanto o esperado. E o motivo é óbvio: **não somos especialistas** em tintas e argamassas.

Motivação e produtividade

O problema é um pouco pior que simplesmente não ter o resultado esperado. Quando restringimos o especialista contratado também diminuímos drasticamente a sua produtividade e motivação com relação ao projeto. A queda de produtividade é bem simples de se explicar: houve uma restrição que o impediu de utilizar as melhores ferramentas para executar o trabalho. A desmotivação é uma mera consequência disso. O especialista sabe que poderia fazer um trabalho melhor em menos tempo, então o projeto passa a ser visto (pelo menos em parte) como uma grande perda de tempo.

> Um sushiman sem uma faca afiada também é capaz de fazer sushi. Mas ele vai gastar mais tempo e os cortes serão grosseiros.

De volta ao mundo da tecnologia

Por vezes clientes me solicitaram que um projeto fosse desenvolvido em uma linguagem específica ou utilizando um determinado *framework*. Geralmente essas condições são impostas por regras organizacionais que visam a manutenção do sistema por uma equipe interna ou a padronização de tecnologias utilizadas.

Novamente caímos em uma questão de custo-benefício: vale a pena pagar mais caro, demorar mais tempo para ter uma solução pronta e não poder extrair todo o benefício dela? O que se ganha em troca? Se você se encontrar em uma situação onde precisa fazer uma **restrição ao especialista**, coloque os números

na ponta do lápis.

> Sua empresa contratou um especialista em tecnologias. Se não puder confiar nas opiniões dele sobre tecnologias, chame alguém mais confiável. Mas não restrinja o trabalho do especialista.

O ataque à exceção

Imprevistos. Não importa quanto tempo gastamos planejando algo, eles sempre vão acontecer. Isso porque geralmente pensamos nos padrões e nos esquecemos das exceções, que precisam acontecer primeiro para depois serem tratadas.

Alguns exemplos clássicos de exceção são:

- O funcionário que passa mais tempo no Facebook do que trabalhando
- O local de trabalho mais afastado onde não existe sinal de telefone ou WiFi.
- O projeto que precisa de uma etapa extra de aprovação a cada entregável
- O usuário que trabalha com um *hardware* ou *software* diferente

E alguns respectivos **ataques à exceção** clássicos seriam:

- Bloquear o acesso de todos funcionários ao Facebook
- Alterar o software utilizado para funcionar apenas em modo offline
- Alterar o processo de aprovação de todos os projetos para atender o novo cliente
- Investir em uma adaptação que pode ser prejudicial aos outros funcionários

Existe um dano colateral em todas as soluções propostas acima. O caso clássico sobre bloquear redes sociais vai contra a filosofia de empresas como a Google, e a explicação é bem simples: se um funcionário passa tempo demais no Facebook, fale com ele. Se isso realmente afetar sua produtividade e não houver perspectiva de mudança, o certo seria substituí-lo e não prejudicar todos os outros funcionários. Veja bem: estamos criando uma regra que prejudica bons funcionários (o que é ruim) para criar um ambiente onde é possível ter funcionários relapsos (o que é pior).

Não digo que as soluções propostas acima são maléficas em qualquer situação. Mas é muito importante se atentar para qualquer **efeito colateral** que os ataques à exceção podem gerar. Manter o funcionamento da regra é geralmente mais importante do que tratar exceções isoladas. Mas, por mais engraçado que seja, existem exceções a essa regra também.

No final, todos os principais motivos de fracasso podem ser corrigidos de uma maneira simples: basta estimar o quanto se ganha, o quanto se perde e o quanto

se gasta ao implementar uma modernização. Mas é necessário fazer isso de maneira concreta, com **números**.

> Confira sempre se um ataque à exceção não tem efeitos colaterais indesejados. Pense em números.

Os 4 Passos para um software eficiente

Vi muitos projetos falharem quando se enquadraram em algum dos padrões citados anteriormente. Mas participei de mais projetos que quebraram esses padrões e geraram benefícios incríveis para organização responsável. Essa convivência com os dois lados da moeda ao longo de mais de 10 anos de carreira me possibilitou enxergar também os padrões de sucesso.

A cultura do questionamento

Existe algo em comum entre os **quatro passos para o software eficiente**: o questionamento. Em toda e qualquer modernização adotada por uma organização, os participantes precisam estar cientes dos benefícios e dos valores envolvidos no projeto. O senso crítico e a postura de cada envolvido é essencial para que o direcionamento do projeto possa sempre tomar o rumo mais produtivo.

> Questione. Sempre. E fomente esse hábito em todos os envolvidos no projeto.

A Necessidade Real

Toda modernização é fruto de uma necessidade que de alguma forma emergiu dos processos da organização.

> A necessidade é a alavanca que move a humanidade. - ***Eric Klain***

Embora a necessidade surja naturalmente, alguém sempre é repońsável por identificá-la. Essa pessoa tende a ter uma certa tendência a abraçar o projeto de modernização necessário como um filho e cair no padrão *fruto do orgulho*__. Não seja essa pessoa; questione essa necessidade.

Desejos desnecessários

Como utilizo metodologias ágeis no desenvolvimento de meus projetos, estou sempre aberto a sugestões de todos os envolvidos no projeto. E como nos reunimos periodicamente para acompanhar os avanços do projeto, essas sugestões costumam surgir frequentemente.

- E se tívessemos um mapa onde eu pudesse ver em tempo real onde está cada um dos usuários do sistema?

Realmente é uma funcionalidade que salta aos olhos. Mas quem ganha o quê com isso? Qual é a real necessidade de se gastar tempo e dinheiro construindo algo assim?

- O que acontece se não houver energia elétrica por 10 dias consecutivos em nosso Data Center? Como podemos nos precaver para que isso não afete o funcionamento do sistema?

Em uma situação extrema como essa o caos é inevitável. Vale a pena gastar tempo e dinheiro contornando uma situação que provavelmente nunca vai acontecer? E, caso aconteça, o funcionamento do sistema provavelmente será o menor dos problemas a serem resolvidos.

- Seria realmente importante termos uma versão móvel desse aplicativo.

Os usuários têm acesso a esses dispositivos no ambiente onde o sistema será utilizado? Eles estão adaptados e aptos a usar esse tipo de ferramenta? A organização está disposta a investir em hardware também?

Idéias são sempre bem vindas em um projeto de modernização. E é natural que idéias sem valor apareçam, principalmente quando as pessoas estão empolgadas com o projeto. Faz parte do trabalho filtrar tudo aquilo que não atende uma **necessidade real**.

> Identificar se a necessidade de mudança é mesmo real nos ajuda a eliminar **3** padrões de fracassos: o fruto do orgulho, o benchmarking equivocado e o ataque à exceção.

O Objetivo Principal

Todo projeto precisa ter um objetivo principal. Algo simples que dê um norte para todos os envolvidos no projeto. Vejamos a descrição de alguns softwares famosos:

- **Facebook**: *No Facebook você pode se conectar e compartilhar o que quiser com quem é importante em sua vida.*
- **Twitter**: *Get in-the-moment updates on the things that interest you*
- **Youtube**: *Flickr for videos*

Lean Canvas

Uma boa prática para identificar um objetivo principal é elaborar um *Lean Canvas* do projeto. Como o preenchimento do quadro não leva muito tempo, rapidamente podemos chegar ao objetivo principal através de algum debate com os envolvidos no projeto.

Além de ajudar a definir **o objetivo principal** do projeto, o *Lean Canvas* é útil para identificar possíveis riscos, oportunidades e documentar expectativas. Para quem é fluente em inglês, a Udemy possui um mini curso de pouco mais de 1 hora de duração sobre o assunto: https://www.udemy.com/lean-canvas-course/.

Como definir um bom objetivo principal

Evite o blablablá; seja conciso. O importante é que a mensagem seja compreendida por todos os envolvidos, afinal não estamos criando um *slogan super esperto*. Se o objetivo estabelecido for atingido, a empresa terá um benefício real? Se a resposta for sim, vamos em frente!

> Definir o objetivo principal nos ajuda a combater os padrões **busca pela perfeição**, **benchmarking equivocado** e até mesmo o **ataque à exceção**

A Análise do Processo

Analisar o processo é um trabalho que precisa ser contínuo. Precisamos assumir que todo processo está em uma constante evolução que deve ser acompanhada.

Antes: O processo atual

Se você foi capaz de definir uma **necessidade real** no primeiro passo, já começou com o pé direito e fez alguma análise do processo atual. Ter uma noção real de como as coisas estão é essencial para que possamos estimar o que podemos alcançar através da modernização, e isso nos ajuda a pontuar o momento de "virar a chave".

Durante: O momento de evoluir

Caso um **objetivo principal** também tenha sido encontrado, estamos muito perto de definir o momento de evoluir. Ou de implantar o sistema.

O melhor momento de aplicar a modernização (implantar o sistema, atualizar uma versão, redefinir práticas, etc) é quando se torna possível perceber um mínimo ganho sobre o processo atual. Se a análise do cenário atual foi bem feita, fica fácil saber se um sistema (mesmo incompleto) já é capaz de gerar ganhos se implantado.

Nesse ponto, o mais importante é não se deixar levar pela inércia do *status quo*. Se não corrermos o risco de piorar as coisas, nunca vamos melhorá-las.

Depois: A melhoria contínua

Qual era o **objetivo real**? Vamos supor que fosse *permitir que os vendedores realizem uma venda em 5 minutos*, e que o tempo estimado de uma venda anteriormente fosse de 10 minutos.

Se um sistema foi criado com esse objetivo, é de fundamental importância que ele colete métricas relativas ao tempo de venda. Conseguimos mesmo reduzir o tempo para 5 minutos? Qual é a etapa mais demorada do processo? É possível reduzir esse tempo para 3 minutos?

E se o projeto falhou em algum ponto e o tempo gasto com as vendas aumentou ao invés de diminuir?

Coletar métricas é vital para que possamos ter um entendimento mais concreto do processo e atacar os pontos mais críticos. Se for possível fazer isso de maneira automática, melhor ainda. Algumas ferramentas, como o Mixpanel (http://mixpanel.com) podem auxiliar muito na captação de dessas métricas e facilitar a identificação de pontos de melhoria.

> A análise do processo nos ajuda a quebrar os padrões **fruto do orgulho** e **busca pela perfeição**.

Os Números

Se alguém te sugerisse mudar para o outro lado da cidade, qual seria o argumento mais convincente?

1- *Tenho certeza de que você vai economizar algum tempo e dinheiro*

2- *Ao viver lá você vai economizar R$ 1.500,00 por mês e cerca de 25 minutos no deslocamento até seu trabalho*

Aplicar uma modernização implica em mudanças fortes na organização. Para justificar o esforço necessário para a mudança, precisamos de justificativas sólidas. Como **números**.

Expectativas

Escrever as expectativas do projeto de modernização juntamente com seu **objetivo real** é essencial.

Quero reduzir o tempo gasto com vendas para 5 minutos, permitindo um aumento de 40% nas vendas realizadas. Isso totaliza um aumento de R$ 23.000,00 no faturamento mensal do setor.

Esse é um plano concreto. Com **números**. Escrevê-lo antes de começar os trabalhos de desenvolvimento nos permite praticar o ciclo ***PDCA*** no projeto. Mais tarde, se não for possível atingir essas métricas podemos nos perguntar o porquê e caminhar para mais uma evolução.

Números e fornecedores

Alguns fornecedores podem oferecer benefícios imensuráveis, como uma tecnologia supostamente melhor ou um menor tempo de resposta a chamados de suporte. A verdade é que todos os benefícios são, de alguma forma, mensuráveis. Mesmo que precisemos estimá-los.

Implantar uma modernização antecipadamente pode gerar uma economia muito significativa, mas é necessário calculá-la. Se conseguirmos imaginar como estará o caixa da empresa daqui há um ano para cada uma das opções oferecidas pelos fornecedores, o processo de tomada de decisão fica cristalino.

Artifícios como a *Árvore de Decisão* podem ajudar a ordenar os pensamentos e enxergar a melhor escolha mesmo em casos muito duvidosos.

> Utilizar números sempre nos ajudará a quebrar todos os padrões de fracasso. Mas para chegar até números confiáveis, precisamos executar os outros 3 passos anteriores meticulosamente. **Expectativas e resultados só têm relevância quando expressados em números confiáveis**.

Sugestões de Software

Em progresso...

Atualizações, MATRIX e outros

Em progresso...

Sobre o Autor

Mário Melo é formado em Sistemas de Informação pela Universidade Federal de Minas Gerais e tem mais de 10 anos de experiência com modernizações. Iniciou seu primeiro estágio aos 18 anos e desde então se mantém ativo no mercado de desenvolvimento de software, hoje com sua própria empresa: a **Facta**.

Após aprender e aplicar técnicas de desenvolvimento ágil na Facta e em outras empresas, tornou-se também *coach* e instrutor do curso ***Certified Scrum Developer*** pela **GoToAgile**.

Certificações

- Sun Certified Java Developer 5.0
- Certified ScrumMaster
- Certified Scrum Product Owner
- Certified Scrum Developer
- Certified Scrum Professional
- ScrumAlliance Registered Education Provider

Contato

- Twitter: @melomario
- Email: mariofdemelo@gmail.com

Glossário

Metodologias Ágeis

Metodologias de desenvolvimento de software que seguem os princípios definidos pelo AgileManifesto (http://www.agilemanifesto.org/iso/ptbr/).

3.1. A necessidade real

Benchmarking

Processo de busca das melhores práticas numa determinada indústria e que conduzem ao desempenho superior. Quando realizado de maneira competitiva, envolve comparar as práticas dos concorrentes às suas.

3.1. A necessidade real 2.3. O benchmarking equivocado
3.2. O objetivo principal

Lean Canvas

Um quadro utilizado como ferramenta para exercitar idéias de novos projetos.

3.2. O objetivo principal

PDCA

Plan. Do. Check. Act. Trata-se de um método interativo de gestão de quatro passos, utilizado para o controle e melhoria contínua de processos e produtos.

3.4. Os números