Antiágil

Forças contrárias à agilidade nas organizações e como evitá-las

Implantar a agilidade em organizações tradicionais não é fácil. Além de dicas e soluções, este artigo traz uma forma poderosa de enxergar o problema.

Klaus Wuestefeld

Fundador da Objective e já trabalhou em par com Kent Beck, Vinicius Teles e Scott Ambler. Participou da primeira conferência internacional de XP, em 2000, na Itália. Realizou os eventos "Extreme Programming Brasil 2002 e 2004" com a presença de Beck, Ambler, Teles, Tom e Mary Poppendieck, entre outros. É o culpado pelo Prevayler e pelo manifesto da computação soberana. Lidera projetos de desenvolvimento e treinamento ágil para empresas como TV Globo, Siemens, TIM, GVT, Tribunal de Contas da União e Banco Central do Brasil.

Tentar mudar hábitos e processos de uma organização é sempre um desafio. Poucos são os membros que se animam com a mudança: inovadores e os chamados "early adopters". A maioria fica com medo da obsolescência de seus conhecimentos e aqueles que têm alguma posição de influência sentem-se ameaçados. A implantação de uma cultura de software ágil não é diferente. Vamos examinar as forças boas e ruins que atuam especificamente em organizações de software; e tentar "jogar um pouco de luz no lado negro".

Enxergando as Coisas

Vamos usar um diagrama de feedback simplificado para enxergar as coisas, mostrando apenas o

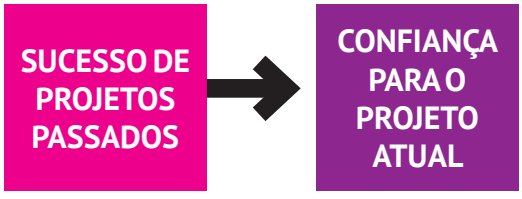

Figura 1. Exemplo de feedback positivo.

feedback positivo, "re-enforçante". Trata-se de um diagrama onde podemos ver as coisas que causam, que fomentam, que reforçam outras coisas. Podemos ler a figura 1, por exemplo, como "Sucesso de Projetos Passados causa/reforça a Confiança para o Projeto Atual".

Mil Maravilhas

Quando abordamos um projeto de software com confiança, com coragem, como na figura 2, isso faz com que tenhamos a atitude de "Play To Win" (jogar pra ganhar). As celebrações da equipe são causadas pelos sucessos a cada entrega incremental do projeto. Celebrações favorecem a amizade que, por sua vez, reforça ainda mais a confiança e a colaboração, num "ciclozinho" virtuoso local.

Jogar pra ganhar favorece também a colaboração do trabalho em equipe e motiva investimentos. Investimentos favorecem a proximidade, pois podem ser usados para conseguir links de dados e equipamentos bons para conferências remotas, para pagar viagens de reunião e, principalmente, para alugar espaço físico para trabalho co-localizado.

A proximidade é o fator crítico para favorecer a colaboração. Dela, principalmente da proximidade com o cliente, vem o entendimento e o aprendizado, que também pode ser acelerado por investimentos em treinamentos, eventos e consultoria.

Sem uma propensão da organização para investir é muito difícil alocar tempo da equipe para refatoração e melhorias de processo, como automação do processo de build e dos testes. Refatoração visa melhorar a simplicidade do código, impactando diretamente a facilidade do seu aprendizado. O aprendizado leva à competência que, claro, ajuda em muitas coisas como o design simples e a corretude do código. A competência é o requisito necessário para uma boa automação e para se fazer refactoring.

A automação favorece o que? Ela dá respaldo para a refatoração, através de uma suíte segura de testes automatizados, que serve também para evitar bugs e garantir a corretude do produto. A eliminação da etapa dos testes manuais permite entregar o software com baixo custo e tempestividade. Aprendi essa palavra, tempestividade, na GVT, meu último cliente de consultoria. Se você não a conhece, vale a pena uma "googlada".

O entendimento do problema do cliente é o que

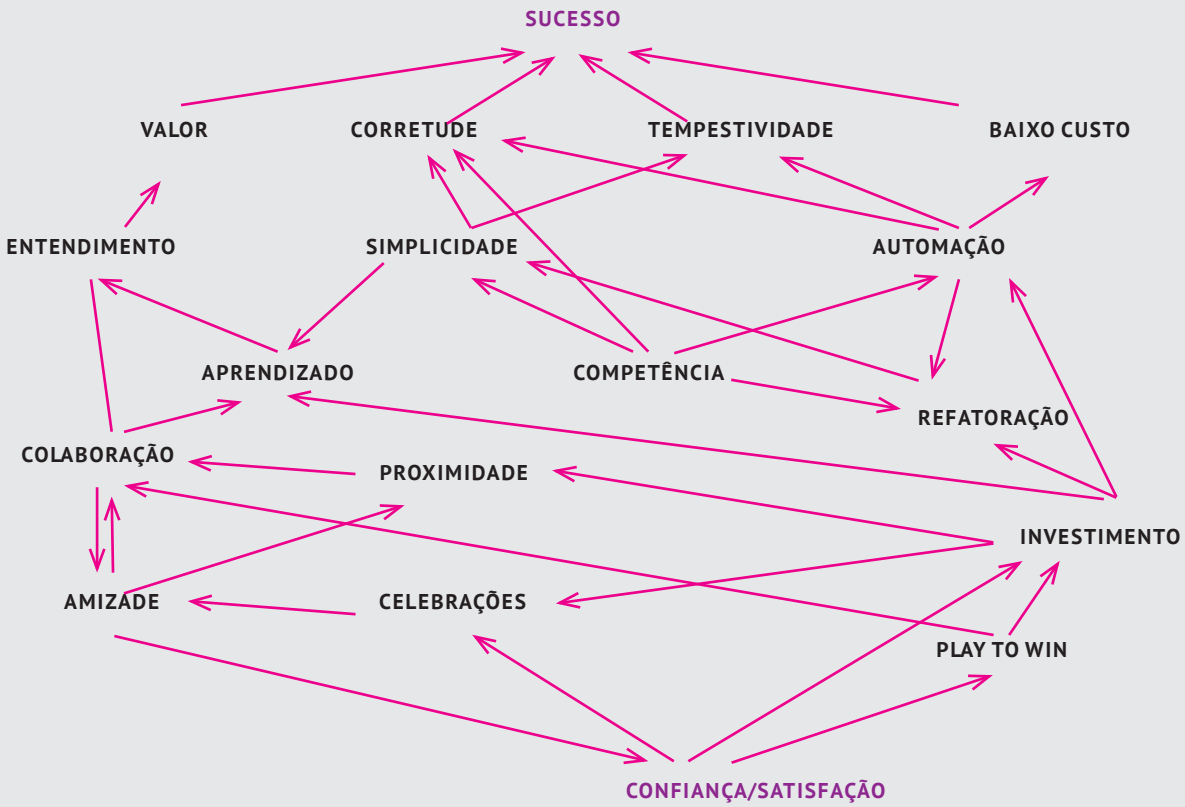

Figura 2. O Ciclo Virtuoso dos Projetos de Software.

garante que o time vai entregar soluções de valor e evitar a famosa frase "não foi nada disso que eu pedi".

Por fim, o valor, a corretude, a tempestividade e o baixo custo são o que determinam o sucesso do projeto. O sucesso a cada fase alimenta, para a próxima, a confiança e a satisfação, que foram nosso ponto de partida, lá embaixo no diagrama, fechando assim o ciclo virtuoso.

O Lado Negro das Forças

Nem tudo são flores no desenvolvimento de software. Além dos fatores positivos que vimos acima, atuam uma série de fatores que, sem cuidado, nos levam ao fracasso.

Tudo começa, na figura 3, com o medo de fracassar, como tantos projetos passados, e com a desconfiança generalizada entre o cliente, escaldado pelos charlatões do mercado, os desenvolvedores, cansados de trabalharem horas extras em sistemas legados complexos e "porcos", e a gestão, frustrada por ter que correr atrás de prazos insanos sem poder fazer muito mais que pressionar a equipe.

O medo e a desconfiança fazem as pessoas jogarem na retranca. Clientes pedem contratos de escopo fechado, de preferência com pesadas multas. Gerentes pedem às equipes ridículas estimativas de curto prazo (sprint planning) e fazem tudo para maximizar a "eficiência" dos "recursos" através de avaliações individuais.

A tática de jogo das pessoas nesse clima de desconfiança e caça às bruxas é sempre o famigerado CYA (Cover Your Ass), traduzido com "tirar o seu da reta". Tanto o escopo fechado quanto o CYA arrastam consigo, desde o começo do projeto, seu pesado fardo de documentação.

Outros problemas causadas pela psicose do escopo fechado são a especulação de requisitos que o cliente é obrigado a fazer com meses ou anos de antecedência e a "waterfallização" do projeto, que já tem que começar com uma gigantesca fase de análise.

O projeto ser waterfall implica em ciclos longos e demorados de entrega, o que já realimenta a especulação de requisitos. Ciclos longos permitem que se instaure a lei de Parkinson, na qual "todo trabalho se expande para ocupar o tempo a ele alocado", através do preciosismo, descaso e a síndrome do estudante. Isso causa atrasos que realimentam ciclos cada vez mais longos.

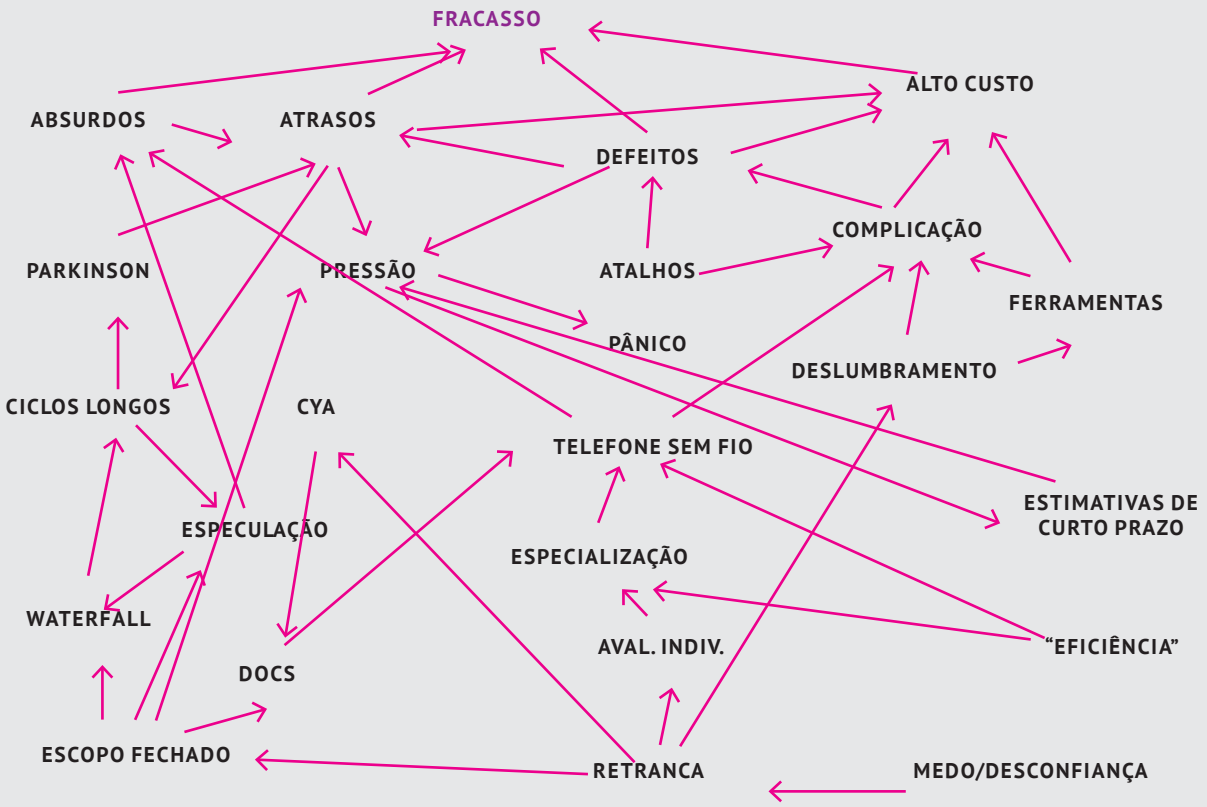

Figura 3. Forças contrárias à agilidade.

Essa especulação toda é uma das responsáveis pelas features absurdas com as quais o cliente acaba "micando" na mão (ver "Jogo do Mico"), em vez de coisas realmente úteis a ele. Os absurdos e os atrasos são os primeiros dois fatores que resultam no fracasso do projeto.

O escopo fechado e os atrasos obrigam diretores e gerentes a aplicar a supersofisticada técnica de botar pressão na equipe, através de gritos de "motivação". Essa pressão pede estimativas de curto prazo que, por sua vez, servem para que? Para fazer mais pressão. A pressão faz então as pessoas buscarem atalhos de qualidade, que sempre funcionam bem, como tirar a calça jeans sem tirar o tênis. Ou seja, trazem defeitos, complicação e, portanto, alto custo.

Voltando para a parte de baixo da figura 3, vemos avaliações individuais de produtividade e busca por uma "eficiência" taylorista míope motivarem a especialização de cargos, como se desenvolver software fosse um processo repetitivo fabril. A especialização agrava o telefone-sem-fio instaurado pela dependência da documentação baseada em texto, pior forma de comunicação humana em uso corrente, contribuindo para os absurdos e para a complicação do produto e do processo.

Um fenômeno interessante que acontece em organizações que têm desconfiança em relação a seus técnicos e cujos processos não favorecem o aprendizado é certo deslumbramento. A gestão, burrificada como a organização, passa a acreditar somente nas coisas que lê em brochuras brilhosas de grandes empresas, dinossauros do mercado. Essas empresas não têm interesse em simplificar o processo de seus clientes porque vendem ferramentas e quanto mais complicado o processo, maior a dependência em ferramentas e maior seu custo.

A complicação, claro, resulta em defeitos e alto custo, que condenam de vez o projeto.

Além de causar seus próprios problemas, o ciclo vicioso ainda sangra as energias positivas do projeto. Na figura 4 vemos esse efeito nas setas vermelhas.

A complicação, claro, detona a simplicidade. O telefone-sem-fio acaba com as chances de entendimento. Os ciclos longos eliminam o feedback rápido, necessário para o aprendizado. A especialização e as avaliações individuais sangram a colaboração e a amizade.

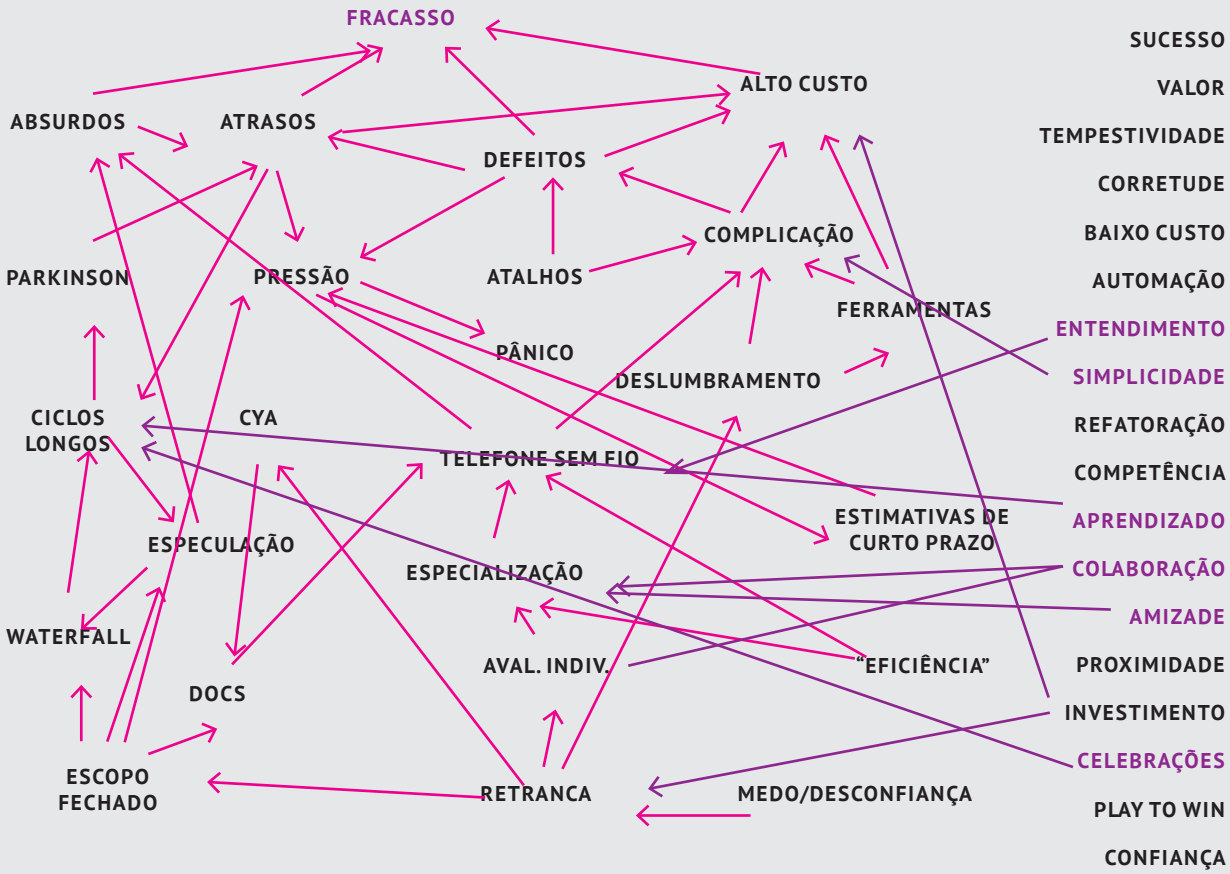

Figura 4. Ciclo vicioso drenando as forças positivas do projeto.

Por fim, a atitude retranqueira e o alto custo minam a propensão e a capacidade de investimento.

E agora?

A primeira coisa que podemos fazer é ter uma atitude ética em relação ao nosso trabalho. Isso significa resistir à pressão e simplesmente negar-se a tomar atalhos, assim como um médico se nega a fazer uma cirurgia "rapidinho" para liberar mais cedo a sala de cirurgia. Desenvolver software sem uma boa cobertura de testes automatizados também é antiético, como um médico fazendo uma operação crítica no século 21 sem plugar os aparelhos que medem os sinais vitais do paciente.

Pedir para o cliente priorizar as tarefas de qualidade é como deixar o paciente decidir se os instrumentos cirúrgicos devem ser esterilizados ou se ele prefere uma cirurgia "mais rápida e mais barata". Trabalhar com qualidade não é uma decisão de gestão e muito menos do cliente. É uma decisão ética individual. Sendo um desenvolvedor de software competente, a última coisa que você precisa temer no mercado atual é ficar sem emprego.

Vou falar agora de pontos de alavancagem que são um pouco menos óbvios, mais contraintuitivos: coisas que muitas vezes iludem as equipes e sua gestão.

A Falsa Eficiência

Preocupar-se em manter todo mundo 100% ocupado é uma falácia. Garantir alocação total das pessoas não é um fim em si. Pelo contrário, manter as pessoas sempre ocupadas dentro de um processo complexo é como querer que o trânsito flua melhor enchendo de carros todas as vias; como querer que seu computador fique rápido e responsivo deixando sua CPU sempre a 100%; como querer que os membros de um time de corrida de revezamento fiquem fazendo flexão de braço enquanto o bastão não chega.

Concentre-se no fluxo rápido de entregas de features completas e correções de defeitos, ignorando completamente a taxa de alocação individual. Você vai observar que o fluxo máximo de produtividade da equipe é atingido quando se tem certo nível de ociosidade.

Para saber mais sobre isso, recomendo a vasta literatura disponível sobre Kanban.

Escopo Especulado

O escopo fechado não garante que o software entregue vai ser o que o cliente precisa. Pelo contrário, por causa da pesada especulação necessária e do compromisso de entregar o que foi especulado, fechar o escopo é receita certeira para entregar algo que não é o que o cliente realmente gostaria de ter ao final do projeto, depois de ter aprendido um monte sobre seu negócio e sobre a tecnologia.

O melhor contra isso é, primeiro, negociar fases menores em vez de um projeto gigante e, depois, ganhar aos poucos a confiança do cliente. À medida que o cliente confia mais na equipe, os requisitos podem ser definidos de forma cada vez mais vaga, mais abstrata, até se tornarem meros objetivos de negócios de alto-nível.

É isso que traz à equipe e ao cliente a humildade e a flexibilidade para aprenderem juntos e entregar um produto de muito maior valor no decorrer do projeto.

Encurte os Ciclos

É muito comum equipes de Quality Assurance, que têm responsabilidade por garantir corretude de software, introduzirem fases de alfa-teste e beta-teste cada vez mais extensas. Estão simplesmente represando e deixando apodrecer e fermentar mais defeitos a cada release. Quanto maiores os ciclos, piores os problemas que podem se esconder no processo.

Se o tempo médio de um release da sua equipe é de meses, lute para reduzir para semanas; se é de semanas, lute para reduzir para dias. Se é de dias, reduza para horas; e por aí vai.

Não discuta. Reduza.

Play To Win

Por fim e mais importante: tenha coragem. Faça de tudo para superar o medo, para sair da retranca, do clima de desconfiança generalizada e trazer para sua equipe o espírito de jogar pra ganhar.

Conclusão

Na próxima vez em que você contar a situação da sua empresa para um amigo agilista e perguntar o que deve fazer, não estranhe se ele levar alguns segundos pra responder. Ele precisa montar na cabeça dele um cenário como os acima, buscar as causas raiz e, só então, te mostrar os melhores pontos de alavancagem para reverter as forças do lado negro em forças do ciclo virtuoso.

Na próxima vez em que você contar a situação da sua empresa para um amigo agilista e perguntar o que deve fazer, não estranhe se ele levar alguns segundos pra responder.