Getting Real by 37signals

Caindo na Real

Bem vindos à primeira das traduções mundiais completa do livro 'Getting Real'. Totalmente em Português.

Introdução capítulo 1

O que é Caindo na Real?

Quer construir uma aplicação web de sucesso? Então é hora de Cair na Real. Caindo na Real é o menor, mais rápido e melhor caminho para construir software.

- Caindo na Real é sobre pular todas as coisas que representam a realidade (cartas, gráficos, caixas, setas, esquemas, wireframes, etc.) e realmente construir a coisa real.
- Caindo na Real é menos. Menos massa, menos software, menos funcionalidades, menos papéis, menos tudo que não é essencial (e a maioria do que você pensa ser essencial realmente não é).
- Caindo na Real é permanecer pequeno e ser ágil.
- Caindo na Real inicia com a construção da interface, ou seja, as telas reais que as pessoas irão

utilizar. Começa com as experiências reais dos clientes, construindo a partir disso para trás. Dessa forma você obtém a interface adequada antes de obter um software errado.

- Caindo na Real é sobre iterações e baixar os custos da mudança. Caindo na Real tem tudo a ver com lançamento, refinamento e melhorar constantemente, o que o torna o caminho perfeito para software baseado em web.
- Caindo na Real entrega exatamente o que os clientes precisam e elimina qualquer coisa que não precisam.

Os benefícios de Caindo na Real

Caindo na Real entrega melhores resultados porque o força a lidar com os problemas reais que está tentando resolver em vez de suas idéias sobre esses problemas. Ele o força a lidar com a realidade.

Caindo na Real pula especificações funcionais e outras documentações transitórias em favor de construir telas reais. Uma especificação funcional é para inglês ver, uma ilusão de um acordo, enquanto uma página web pronta é realidade. É isso que seus clientes irão ver e usar. É isso que importa. Caindo na Real o leva lá mais rápido. E isso signfica que está tomando decisões de software baseado na coisa real em vez de noções abstratas.

Finalmente, Caindo na Real é a maneira que se encaixa idealmente para software baseado em web. O modelo convencional de entregar software em uma caixa e então esperar um ano ou dois para entregar uma atualização está desaparecendo. Diferente de software instalado, aplicações web podem evoluir constantemente de maneira diária. Caindo na Real abre essa vantagem por tudo que ele vale.

Como Escrever Software Vigoroso

Escrita vigorosa é concisa. Uma sentença não deve conter palavras desnecessárias, um parágrafo não deve conter sentenças desnecessárias, pela mesma razão que desenhar não deve ter linhas desnecessárias e uma máquina não deve ter partes desnecessárias. Isso requer não que o escritor torne todas as sentenças curtas ou evite todos os detalhes e trate os assuntos apenas em ítens, mas sim que cada palavra fale.

—De "Os Elementos de Estilo" de William Strunk Jr.

Sem mais gordura

Da forma antiga: um processo comprido, burocrático, estamos-fazendo-isso-para-proteger-nossas-bundas. O resultado típico: software gorduroso, esquecível, vazando em mediocridade. Eca.

Caindo na Real se livra de ...

- Cronogramas que levam meses ou mesmo anos
- Especificações Funcionais Utópicas
- Debates de Escalabilidade
- Reuniões de equipe intermináveis
- A "necessidade" de contratar dúzias de funcionários
- Números de versões sem sentido
- Planejamentos cristalinos que prevêem o futuro

- Opções de preferência intermináveis
- Suporte terceirizado
- Testes de usuário irreais
- Papelada inútil
- Hierarquia de cima-para-baixo

Você não precisa de toneladas de dinheiro ou uma equipe enorme ou um ciclo de desenvolvimento longo para construir grandes softwares. Essas coisas são ingredientes para aplicações lentas, esfumaçadas, que não mudam. Caindo na Real usa o caminho oposto.

Nesse livro lhe mostraremos ...

- A importância de ter uma filosofia
- Por que se manter pequeno é uma coisa boa
- Como construir menos
- Como ir de idéia à realidade rapidamente
- Como montar sua equipe
- Por que você deve fazer design de dentro para fora
- Por que escrever é tão crucial
- Por que você deve fazer menos que sua concorrência
- Como promover sua aplicação e espalhar a palavra
- Segredos para um suporte de sucesso
- Dicas de como manter o momento depois do lançamento
- ... e muito mais

O foco é em idéia amplas. Não vamos entediá-lo com trechos de código detalhados ou truques de css. Vamos nos manter nas grandes idéias e filosofias que dirigem o processo Caindo na Real.

Esse livro é para você?

Você é um empreendedor, designer, programador ou marketeiro trabalhando em uma grande idéia.

Você percebe que as velhas regras não se aplicam mais. Distribui seu software em cd-roms a cada ano? Que 2002. Números de versão? Jogue pela janela. Você precisa construir, lançar e refinar. Então recomece e repita.

Ou talvez você ainda não esteja a bordo do desenvolvimento ágil e estruturas de negócios, mas está louco para aprender mais.

Se isso soa como você, então esse livro é para você

Nota: enquanto este livro enfatiza em construir aplicações web, um monte de idéias são aplicáveis para atividades que não são de software também. As sugestões sobre equipes pequenas, prototipação rápida, esperar iterações e muitas outras apresentadas aqui podem servir como um guia seja se estiver começando um negócio, escrevendo um livro, fazendo o design de um site web, gravando um álbum ou fazendo uma variedade de outras coisas. Uma vez que começar Caindo na Real em uma área de sua vida,

verá que esses conceitos podem ser aplicados para uma ampla variedade de atividades.

Sobre a 37signals

O que fazemos

37signals é uma pequena equipe que cria software simples e focado. Nossos produtos o ajudam a colaborar e se organizar. Mais de 350 mil pessoas e pequenos negócios usam nossas aplicações web para fazer suas coisas. Jeremy Wagstaff, do Wall Street Journal, escreveu “os produtos da 37signals são ferramentas maravilhosamente simples, elegantes e intuitivas que fazem uma tela de Outlook parecer um equivalente em software de uma câmara de tortura”. Nossos aplicativos nunca põe você no pau de arara.

Nosso modus operandi

Acreditamos que software é muito complexo. Funcionalidades demais, botões demais, coisa demais para aprender. Nossos produtos fazem menos do que a concorrência – intencionalmente. Construímos produtos que funcionam de forma mais esperta, que parecem melhor, que lhe permitem fazer suas coisas e são mais fáceis de usar.

Nossos produtos

Até a data de publicação desse livro, temos cinco produtos comerciais e um framework open source de aplicações web.

Basecamp vira a cabeça de gerenciamento de projetos. Em vez de tabelas Gantt, gráficos engraçadinhos e planilhas lotadas de estatísticas, Basecamp oferece painéis de mensagens, listas de tarefas, cronograma simples, escritas colaborativas e compartilhamento de arquivos. Até agora, centenas de milhares concordam que é a melhor maneira. Farhad Manjoo, da Salon.com disse que “Basecamp representa o futuro de software na Web”.

Campfire traz um simples chat em grupo para o contexto de negócios. As empresas conhecidas entendem quão valioso um chat persistente em tempo real pode ser. Mensagens instantâneas convencionais são ótimas para conversas entre duas pessoas, mas são miseráveis para 3 ou mais pessoas de uma só vez. Campfire resolve esse problema e muito mais.

Backpack é a alternativa para aqueles confusos, complexos “organize sua vida em 25 simples passos” gerenciadores de informações pessoais. A tirada de Backpack com páginas, anotações, lista de tarefas e avisos via telefones celulares / e-mail são idéias inovadoras em uma categoria de produtos que sofre com o status quo. Thomas Weber, do Wall Street Journal disse que é o melhor produto na sua classe e David Pogue, do New York Times o chamou de uma ferramenta de organização “muito legal”.

Writeboard deixa você escrever, compartilhar, revisar e comparar texto, sozinho ou com outros. É a alternativa refrescante dos gordurosos processadores de texto que são demais para 95% do que você escreve. John Gruber, da Daring Fireball disse “Writeboard deve ser a aplicação web mais clara e simples que já vi”. O guru de Web, Jeffrey Zeldman disse “as mentes brilhantes da 37signals fizeram novamente”.

Ta-da List mantém todas as suas listas de tarefas juntas e organizadas online. Mantenha as listas para você ou compartilhe com outros para fácil colaboração. Não existe jeito mais fácil de terminar suas coisas. Mais de 100 mil listas e perto de 1 milhão de ítens foram criadas até agora.

Ruby on Rails, para desenvolvedores, é um framework web completo, open source para escrever aplicação para o mundo real rapidamente e facilmente. Rails toma conta do trabalho pesado para que você possa

focar na sua idéia. Nathan Torkington, do império editorial O'Reilly disse que "Ruby on Rails é incrível. Usá-lo é como assistir a um filme de kung-fu, onde uma dúzia de frameworks maus se preparam para atacar o novato apenas para apanharem de uma variedade de formas imaginativas". Não tem como não gostar dessa citação.

Você pode encontrar mais sobre nossos produtos e nossa companhia no nosso site web em www.37signals.com.

Avisos, Condições e outros Ataques Antecipados

Apenas para tirar isso do caminho, aqui estão nossas respostas para algumas reclamações que ouvimos de vez em quando:

"Essas técnicas não funcionarão para mim"

Caindo na Real (Getting Real) é um sistema que funcionou excelentemente para nós. Dito isso, as idéias nesse livro não se aplicarão a todos os projetos abaixo do Sol. Se estiver construindo um sistema de armas, uma usina de controle nuclear, um sistema bancário para milhões de clientes ou outro sistema crítico vital/financeiro, você irá latir a algumas de nossas atitudes. Vá em frente e tome precauções adicionais.

E não precisa ser uma proposição do tipo tudo ou nada. Mesmo que não possa abraçar Caindo na Real completamente, devem existir pelo menos algumas idéias aqui que você possa tentar usar.

"Vocês não inventaram essa idéia"

Não estamos afirmando que inventamos essas técnicas. Muitos desses conceitos estão por aí de uma forma ou de outra há um bom tempo. Não fique nervoso de ler algum de nossos conselhos e isso o lembrar de alguma coisa que já leu mais ou menos em algum weblog ou algum livro publicado 20 anos atrás. É definitivamente possível. Essas técnicas não são todas exclusivas da 37signals. Apenas estamos dizendo como nós trabalhamos e o que tem feito sucesso para nós.

"Vocês levam tudo para uma visão muito preto-no-branco"

Se nosso tom parecer muito convencido, conviva com isso. Achamos que é melhor apresentar idéias de maneira enfática do que ser escorregadio sobre isso. Se parecer grosseiro ou arrogante, que assim seja. Preferimos ser provocativos do que diluir tudo com "isso depende …" Claro, haverá momentos quando essas regras precisão ser esticadas ou quebradas. E algumas dessas táticas podem não se aplicar à sua situação. Use seu julgamento e imaginação.

"Isso não funcionará dentro da minha empresa"

Acha que você é grande demais para Cair na Real (Getting Real)? Mesmo a Microsoft está Caindo na Real (e duvidamos que você seja maior do que eles).

Mesmo que sua empresa funcione tipicamente com cronogramas de longo prazo e com grandes equipes, ainda existem maneiras de Cair na Real. O primeiro passo é quebrar em pequenas unidades. Quando existem pessoas demais envolvidas, nada acontece. Quanto mais enxuto você for, mais rápido – e melhor – as coisas acontecem.

Entretanto, isso vai requerer alguma conversa de vendas. Venda a idéia do processo Caindo na Real na sua empresa. Mostre a eles esse livro. Mostre a eles os resultados reais que você pode atingir em menos tempo

e com equipes menores.

Explique que Caindo na Real é uma maneira de baixo risco, baixo investimento para testar novos conceitos. Veja se você pode se separar da nave-mãe em um projeto menor, como prova de conceito. Demonstre resultados.

Ou, se quiser ser corajoso, vá silenciosamente. Voe abaixo do radar e demonstre resultados reais. Essa foi a forma que a equipe da Start.com usou na Microsoft. “Eu observei a equipe da Start.com trabalhar. Eles não pedem permissão”, disse Robert Scoble, Technical Evangelist da Microsoft. “Eles tem um chefe que fornece cobertura aérea. E eles mordem um pequeno pedaço de cada vez, fazem isso e respondem a feedback”.

Lançando Start.com da Microsoft

Em uma grande empresa, processos e reuniões são normais. Muitos meses são gastos em planejamento de funcionalidades e discutindo detalhes com a finalidade de todos alcançarem um acordo sobre o que é a coisa “certa” para o cliente.

Essa pode ser a forma certa para softwares de prateleira, mas com a web nós temos uma incrível vantagem. Apenas lance! Deixe o usuário lhe dizer se é a coisa certa ou não. Ei, você pode corrigir e lançar na web no mesmo dia, se quiser! Não existe palavra mais forte do que do cliente – resista à pressão de se engajar em longas reuniões e discussões. Apenas lance e prove seu ponto.

Mais fácil falar do que fazer – isso implica:

Meses de planejamento não são necessários.
Meses escrevendo especificações não são necessários – especificações devem ter as fundações pregadas e os detalhes entendidos e refinados durante a fase de desenvolvimento. Não tente fechar todos os pontos abertos e pregar cada detalhe antes de começar a desenvolver.

Lance menos funcionalidades, mas de qualidade.
Você não precisa usar a forma big bang com todo novo lançamento e amontoados de funcionalidades. Dê aos usuários pedaços minúsculos que eles possam digerir.

Se existirem pequenos bugs, lance tão logo tenha os cenários principais pregados e disponibilize as correções dos bugs gradualmente depois disso. Quanto mais rápido tiver o feedback do usuário, melhor. Idéias podem soar ótimas no papel, mas na prática acabam sendo menos do que boas. Quanto mais cedo descobrir sobre pontos fundamentais que estão errados com uma idéia, melhor.

Uma vez que você estiver iterando rapidamente e reagindo ao feedback dos clientes, estabelecerá uma conexão com eles. Lembre-se que o objetivo é ganhar o cliente construindo o que eles querem.

— *Sanaz Ahari, Gerente de Programa da Start.com, Microsoft*

A Linha de Largada capítulo 2

Construa Menos

Faça menos que sua competição

O senso comum diz que para vencer seus competidores, você precisa estar um passo a frente. Se eles possuem quatro funcionalidades, você precisa de cinco (ou 15, ou 25). Se eles gastam X, você precisa gastar XX. Se eles têm 20, você precisa 30.

Este tipo de estratégia, a Guerra Fria de estar um passo a frente, leva a uma briga sem fim. Trabalhar assim é caro, defensivo e paranóico. Empresas defensivas e paranóicas não pensam para frente, eles pensam apenas no passado. Elas não lideram, elas seguem.

Se você quer construir uma empresa que segue, este livro não é para você.

Mas então, e ai? A resposta é menos. Faça menos que a concorrência para desbancá-los. Resolva os problemas simples, deixe os problemas cabeludos, difíceis e desesperadores para os outros. Ao invés de estar um passo a frente, esteja um passo atrás. Ao invés de se superar, tente manter-se dentro do seu potencial.

Veremos o conceito de menos durante o livro, mas para os iniciantes, menos significa:

- Menos funcionalidades
- Menos opções/preferências
- Menos pessoas e estrutura empresarial
- Menos reuniões e abstrações

Qual o Seu Problema?

Construa software para você mesmo

Uma grande maneira de escrever software é começar resolvendo seus próprios problemas. Você será o público-alvo e saberá o que é importante e o que não é. Isso lhe dá um bom adiantamento na entrega de um produto fora de série.

A chave aqui é entender que não está sozinho. Se estiver tendo problemas, é provável que centenas de milhares de outras pessoas estão no mesmo barco. Esse é seu mercado. Não foi fácil?

Basecamp se originou em um problema: como uma empresa de design precisávamos de uma maneira simples de comunicar nossos clientes sobre os projetos. Começamos fazendo isso através da extranet dos clientes, que atualizávamos manualmente. Mas modificar o HTML na mão toda vez que o projeto precisava ser atualizado simplesmente não estava funcionando. Esses sites de projetos sempre pareciam ficar travados e eventualmente eram abandonados. Era frustrante porque nos deixava desorganizados e deixava os clientes no escuro.

Então começamos a procurar outras opções. Ainda assim cada ferramenta que encontrávamos ou 1) não fazia o que precisávamos ou 2) era gorda de funcionalidades que não precisávamos – como cobrança, controles estritos de acesso, planilhas, gráficos, etc. Sabíamos que deveria haver uma maneira melhor então decidimos construir nossa própria.

Quando resolvemos nossos próprios problemas, criamos uma ferramenta que nos apaixona. E paixão é a chave. Paixão significa que realmente a usaremos e cuidaremos dela. E essa é a melhor maneira de fazer os outros se sentirem apaixonados sobre ela também.

Arranhando sua própria coceira

O mundo de Código Aberto abraçou esse mantra há muito tempo – eles chamam de “arranhando sua própria coceira”. Para os desenvolvedores de código aberto, significa que terão as ferramentas que querem, entregues da maneira que querem. Mas os benefícios vão mais a fundo.

Como designer ou desenvolvedor de uma nova aplicação, você precisa encarar centenas de micro-decisões todos os dias: azul ou verde? Uma tabela ou duas? Estática ou dinâmica? Abortar ou recuperar? Como tomamos essas decisões? Se é algo que reconhecemos como importante, poderíamos perguntar. O resto, chutamos. E todos esses chutes constroem um tipo de débito em nossas aplicações – uma rede interconectada de coisas que assumimos.

Como um desenvolvedor, detesto isso. O conhecimento de todas essas bombas-relógio em pequena escala nas aplicações que escrevo somam-se ao meu stress. Desenvolvedores de código aberto, arranhando suas próprias coceiras, não sofrem isso. Porque eles são seus próprios usuários, eles sabem a resposta correta para 90% das decisões que precisam tomar. Acho que é uma das razões que as pessoas chegam em casa após um dia duro de trabalho de codificação e ainda trabalham com código aberto: é relaxante.

— Dave Thomas, The Pragmatic Programmers

Nascido da necessidade

Campaign Monitor realmente nasceu na necessidade. Por anos nos frustramos com a qualidade das opções de marketing por e-mail que existiam por aí. Uma ferramenta fazia x e y mas nunca z, a próxima tinha y e z mas simplesmente não podia ter x direito. Não podíamos vencer.

Decidimos liberar a agenda e começar a construir nossa ferramenta de marketing por e-mail dos sonhos. Conscientemente decidimos não olhar para o que os outros estavam fazendo e em vez disso construir algo que fizesse nossas vidas, e a de nossos clientes, um pouco mais fáceis.

Depois descobrimos que não éramos os únicos que estavam infelizes com as opções que existiam. Fizemos algumas modificações ao software de forma que qualquer empresa de design pudesse usá-lo e começamos a espalhar a palavra. Em menos de seis meses, milhares de designers estavam usando Campaign Monitor para enviar informativos por e-mail para eles mesmos e seus clientes.

—David Greiner, fundador, Campaign Monitor

Você precisa de importar sobre isso

Quando você escreve um livro, precisa de mais do que uma história interessante. Precisa ter um desejo de contar a história. Precisa investir pessoalmente de alguma maneira. Se vai viver com alguma coisa por dois anos, três anos, o resto de sua vida, precisa se importar sobre isso.

—Malcolm Gladwell, autor (de Algumas Finas Fatias de Malcolm Gladwell)

Financie Você Mesmo

Dinheiro de fora é plano B

A primeira prioridade de muitas empresas iniciantes é adquirir fundos de investidores. Mas lembre-se, se nos viramos para gente de fora para fundos, teremos que responder a eles também. Crescem expectativas. Investidores querem seu dinheiro de volta – e rapidamente. O fato triste é que dinheiro entrando nem sempre significa a construção de um produto de qualidade.

Atualmente não é preciso muito para começar. Hardware é barato e uma boa parte de grandes softwares de infra-estrutura são código aberto e de graça. E paixão não vem com uma etiqueta de preço.

Então faça o que puder com o dinheiro que tem em mãos. Pense muito e determine o que é realmente

essencial e o que pode viver sem. O que pode fazer com três pessoas em vez de dez? O que pode fazer com R$ 40 mil em vez de R$ 200 mil? O que pode fazer em três meses em vez de seis? O que pode fazer se puder manter seu emprego e construir sua aplicação nas horas vagas?

Restrições forçam a criatividade

Dirija com recursos limitados e será forçado a contar com restrições mais cedo e mais intensamente. E isso é uma coisa boa. Restrições dirigem inovação.

Restrições também o forçam a liberar sua idéia para fora mais cedo em vez de mais tarde – outra coisa boa. Um mês ou dois fora das porteiras devem lhe dar uma boa idéia se você está em algo sólido ou não. Se estiver será auto-sustentável logo e não precisará de dinheiro externo. Se sua idéia estiver furada, é hora de voltar à prancheta de desenho. Pelo menos sabe disso agora em vez de meses (ou anos) para frente. E pelo menos pode voltar atrás mais facilmente. Planos de saída se tornam bem complicados quando investidores estão envolvidos.

Se estiver criando software apenas para fazer um dinheiro rápido, isso vai aparecer. Um retorno rápido é bem improvável. Então foque em construir uma ferramenta de qualidade que você e seus clientes poderão viver com por um bom tempo.

Dois caminhos

[Jake Walker começou uma companhia com dinheiro de investidores (Disclive) e um sem (The Show). Aqui ele discute as diferenças entre os dois caminhos.]

A raíz de todos os problemas não foi conseguir dinheiro, mas tudo que veio junto com ele. As expectativas são simplesmente mais altas. As pessoas começam tomando salários e a motivação é para construir e depois vender, ou encontrar outra maneira para os investidores iniciais terem seu dinheiro de volta. No caso da primeira empresa, simplesmente começamos a agir como se fôssemos muito maiores do que realmente éramos – sem necessidade …

[Com The Show] percebemos que poderíamos entregar um produto muito melhor com menos custo, apenas com mais tempo. E apostamos com um pouco de nosso próprio dinheiro que as pessoas iriam esperar por mais qualidade em vez de velocidade. Mas a empresa se manteve (e provavelmente continuará sendo) uma operação pequena. E desde esse primeiro projeto, estamos totalmente auto-financiados. Com apenas um pouco de criatividade de nossos fornecedores, nunca mais realmente precisamos colocar muito de nosso próprio dinheiro na operação. E a expectativa não era de crescer e vender, mas de crescer por crescer e continuar se beneficiando disso financeiramente.

—Um comentário de Signal vs. Noise

Fixe o Prazo e o Orçamento, Flexibilize o Escopo

Lance dentro do prazo e do orçamento

Aqui vai uma maneira fácil de lançar dentro do prazo e do orçamento: mantenha-os fixos. Nunca jogue mais tempo ou dinheiro em um problema, apenas diminue o escopo.

Existe um mito que diz o seguinte: podemos lançar no prazo, no orçamento e no escopo. Isso quase nunca acontece e quando acontece a qualidade normalmente sofre.

Se não puder encaixar tudo dentro do prazo e orçamento planejados então não aumente o tempo e o custo.

Em vez disso, puxe o escopo para trás. Sempre existe tempo para adicionar coisas mais tarde – o mais tarde é eterno, o agora está voando.

Lançar alguma coisa grande que está um pouco menor em escopo do que o planejado é melhor do que lançar alguma coisa medíocre e cheio de buracos porque precisou atingir uma janela mágica de prazo, orçamento e escopo. Deixe a mágica para Houdini. Você tem um negócio de verdade para administrar e um produto real para entregar.

Aqui vão os benefícios de fixar o prazo e orçamento e manter o escopo flexível:

- **Priorização**

  Precisaremos descobrir o que é realmente importante. O que vai chegar ao lançamento inicial? Isso força uma restrição que o pressionará a tomar decisões difíceis em vez de ficar hesitando.

- **Realidade**

  Configurar expectativas é a chave. Se tentar fixar o prazo, orçamento e escopo, não será capaz de entregar com um alto grau de qualidade. Claro, provavelmente poderá entregar alguma coisa, mas "alguma coisa" é o que realmente quer entregar?

- **Flexibilidade**

  A habilidade de mudar é a chave. Ter tudo fixado torna as mudanças difíceis. Injetar flexibilidade de escopo apresentará opções baseadas em sua experiência real de construir o produto. Flexibilidade é seu amigo.

Nossa recomendação: abaixo o Escopo. É melhor fazer meio-produto do que um produto meia-boca (mais sobre isso depois).

Um, dois, três ...

Como um projeto chega a estar um ano atrasado? Um dia de cada vez.

—Fred Brooks, engenheiro de software e cientista da computação

---

Tenha um Inimigo

Pegue uma briga

Algumas vezes a melhor maneira de saber como sua aplicação deve ser é saber o que ela não deve ser. Descubra o inimigo da sua aplicação e você acenderá uma luz para onde precisa ir.

Quando decidimos criar um software de gerenciamento de projetos, sabíamos que Microsoft Project era o gorila na sala. Em vez de temer o gorila, o usamos como motivador. Decidimos que Basecamp seria algo completamente diferente, o anti-Project.

Entendemos que gerenciamento de projetos não é sobre tabelas, gráficos, relatórios e estatísticas – é sobre comunicação. Também não é sobre um gerente de projetos sentando lá no alto e distribuindo um plano de projetos. É sobre todos assumindo responsabilidades juntos para fazer o projeto funcionar.

Nossos inimigos eram os Gerentes de Projetos Ditadores e as ferramentas que eles usavam para chicotear.

Queríamos democratizar o gerenciamento de projetos – fazê-lo de forma que todos fizessem parte (incluindo o cliente). Projetos se dão melhor quando todos assumem propriedade coletiva do processo.

Quando chegou a vez do Writeboard, sabíamos que haviam competidores lá fora com toneladas de funcionalidades. Então decidimos enfatizar em um ângulo “sem frescura”. Criamos uma aplicação que permite às pessoas compartilhar e colaborar nas idéias de maneira simples, sem incomodá-las com funcionalidades não-essenciais. Se não era essencial, deixamos de fora. E em apenas três meses depois do lançamento, mais de 100 mil Writeboards foram criados.

Quando começamos no Backpack nosso inimigo era estrutura e regras rígidas. As pessoas devem ser capazes de organizar suas informações de sua própria maneira – não baseado em uma série de telas pré-formatadas ou uma montanha de campos de edição obrigatórios.

Um bônus que você recebe em ter um inimigo é uma mensagem de marketing muito clara. As pessoas estão cheias de conflitos. E também entendem um produto comparando-o com outros. Com um inimigo escolhido, você está enviando uma história que eles querem ouvir. Não só eles vão entender seu produto melhor e mais rápido, mas vão tomar um lado. E essa é uma maneira garantida de chamar a atenção e acender uma paixão.

Agora, com tudo isso dito, também é importante não ficar muito obcecado com a concorrência. Analise demais outros produtos e você vai começar a limitar sua maneira de pensar. Dê uma olhada e vá em frente para sua própria visão e suas próprias idéias.

Não siga o líder

Marketeiros (e todos os seres humanos) são bem treinados para seguir o líder. O instinto natural é descobrir o que funciona para a concorrência e então tentar superá-los – em ser mais barato que seu competidor que compete no preço, ou mais rápido que seu competidor que compete na velocidade. O problema é que uma vez que o consumidor já comprou a história de alguém e acredita nessa mentira, persuadí-lo a mudar é a mesma coisa que persuadí-lo a admitir que estava errado. E as pessoas odeiam admitir que estão erradas.

Em vez disso, você deve dizer uma história diferente e persuadir os ouvintes que sua história é mais importante do que a história que eles acreditam atualmente. Se sua competição é mais rápida, você deve ser mais barato. Se eles vendem a história de saúde, você deve vender a história da conveniência. Não apenas o posicionamento cartesiano x/y do tipo “Somos mais baratos”, mas uma história real que é completamente diferente da história que já foi contada.

—Seth Godin, autor/empresário (de Seja um Mentiroso Melhor)

Qual é o problema chave?

Uma das maneiras mais rápidas de se colocar em problemas é olhar o que seus competidores estão fazendo. Isso foi especialmente verdade para nós na BlinkList. Desde que lançamos houveram cerca de 10 outros serviços de bookmarking social que foram lançados. Algumas pessoas até começaram a gerar planilhas online com comparações funcionalidade a funcionalidade.

Entretanto, isso pode rapidamente levar ao erro. Em vez disso, permanecemos focados na figura maior e continuamos nos perguntando, qual é o problema chave que estamos tentando resolver e como podemos resolvê-lo.

—Michael Reining, co-fundador, MindValley & Blinklist

Não Deveria ser uma Rotina

Sua paixão — ou falta de — vão aparecer

Quanto menos sua aplicação se tornar uma rotina para construir, melhor será. Mantenha pequena e gerenciável para que possa realmente apreciar o processo.

Se sua aplicação não o excita, algo está errado. Se está trabalhando nela apenas para ganhar dinheiro, isso vai aparecer. Da mesma forma, se você se sentir apaixonado pela aplicação, também vai aparecer no produto final. As pessoas conseguem ler nas entrelinhas.

A presenção de paixão

Em design, onde o significado é normalmente e controversamente subjetivo ou dolorosamente indecifrável, poucas coisas são mais aparentes e lúcidas do que a presença de paixão. Isso é verdade seja quando o design do produto o agrada ou o deixa frio; em ambos os casos é difícil não detectar o investimento emocional das mãos que o construíram.

Entusiasmo se manifesta prontamente, claro, mas indiferença é igualmente inesquecível. Se seu compromisso não vem com paixão genuína para o trabalho às mãos, isso se torna um vazio que é quase impossível de conciliar, não importa o quão elaborado ou atrativo é o design.

—Khoi Vinh, Subtraction.com

A padaria

Os negócios americanos neste momento realmente são sobre desenvolver idéias, torná-las lucrativas, vendê-las enquanto são lucrativas e então sair fora ou diversificar. É justamente sobre sugar tudo. Minha idéia era: aprecie cozinhar, venda seu pão, as pessoas gostam disso, venda mais. Mantenha a padaria indo porque você está fazendo boa comida e as pessoas estão felizes.

—Ian MacKaye, membro da Fugazi e um dos donos da Dischord Records
(da Salon.com People | Ian MacKaye)

Permaneça Enxuto capítulo 3

Menos Massa

Quanto mais enxuto for, mais fácil é para mudar

Quanto mais massa tiver um objeto, mais energia é necessária para mudar sua direção. É uma verdade tanto para o mundo dos negócios como para o mundo físico.

Quando falamos em tecnologias web, mudanças devem ser fáceis e baratas. Se você não puder mudar rapidamente, perderá terreno para alguém que possa. É por isso que você deve optar por menos massa.

Massa aumenta com...

- Contratos de longo prazo
- Excesso de pessoas

- Decisões permanentes
- Reuniões sobre outras reuniões
- Processos Burocráticos
- Inventário (físico ou mental)
- Prisão em hardware, software e tecnologia
- Formatos proprietários de dados
- Passado mandando no futuro
- Planejamentos de longo prazo
- Políticas de escritório

Massa se reduz com...

- Pensamentos just-in-time
- Equipes com membros multi-tarefa
- Abraçar limitações, sem aumentá-las
- Menos software, menos código
- Menos funcionalidades
- Equipes pequenas
- Simplicidade
- Interfaces reduzidas
- Produtos de código aberto
- Formatos de dados abertos
- Uma cultura aberta que torna fácil admitir erros

Menos massa permite mudar de direção rapidamente. Você pode reagir e evoluir. Pode focar em boas idéias e derrubar as ruins. Pode ouvir e responder a seus clientes. Pode integrar novas tecnologias agora em vez de mais tarde. Ao invés de um avião de cargas, você dirige um pequeno bote. Aproveite esse fato.

Por exemplo, vamos imaginar uma empresa enxuta e com menos massa, que construiu um produto com menos código e menos funcionalidades. Do outro lado está uma empresa massuda que tem um produto significativamente com mais software e mais funcionalidades. Então, digamos que uma nova tecnologia como Ajax ou um novo conceito como tags apareçam por aí. Quem estará apto a adaptar seu produto mais rápido? A equipe com mais software e mais funcionalidades, com um planejamento de 12 meses ou a equipe com menos software, menos funcionalidade e com um processo mais organico do tipo “vamos focar no que realmente precisamos agora”?

Obviamente a empresa com menos massa está em uma posição melhor para se ajustar às demandas reais do mercado. A empresa com mais massa ainda estará discutindo as mudanças, ou empurrando-as junto ao processo burocrático, enquanto a empresa com menos massa já haver feito a troca. A empresa com menos massa está dois passos à frente enquanto a empresa com mais massa ainda está tentando entender como andar.

Negócios rápidos, ágeis, e com menos massa podem rapidamente mudar seu modelo de negócios, produtos, funcionalidades e mensagem de marketing. Eles podem cometer erros e corrigí-los rapidamente.

Podem mudar suas prioridades, misturar produtos e focar. E, mais importante, **podem mudar sua maneira de pensar**.

Diminua seu Custo de Mudança

Permaneça flexível reduzindo os obstáculos à mudança

A mudança é sua melhor amiga. Quanto mais caro for para fazer uma mudança, menos chances ela terá de ser realizada. Se seus competidores podem mudar mais rápido, você se encontra em enorme desvantagem. Se a mudança for cara demais, você está morto.

É aí que manter-se enxuto realmente ajuda. A capacidade de mudar num piscar de olhos é algo que equipes pequenas têm por natureza, e que grandes equipes nunca conseguirão ter. É nisto que os grandes invejam os pequenos. O que poderia levar semanas com uma equipe grande em uma mega-corporação pode levar apenas um dia em uma organização pequena e enxuta. Essa vantagem não tem preço. Mudanças rápidas e baratas são a arma secreta dos pequenos.

E lembre-se: Mesmo com todo o dinheiro, marketing e pessoas do mundo você não pode comprar a agilidade de ser pequeno.

Emergencia

A emergencia é um dos princípios fundamentais da agilidade, e é a coisa mais próxima da magia pura. Propriedades emergenciais não são projetadas ou vêm prontas, elas simplesmente acontecem como um resultado dinâmico do resto do sistema. “Emergencia” vem do Latim da metade do século 17, que significa “ocorrência não prevista”. Você não pode planejá-la ou agendá-la, mas pode cultivar um ambiente em que a deixe ocorrer, se beneficiando dela.

Um exemplo clássico de emergência está no comportamento dos bandos de pássaros. Uma simulação de computador pode usar apenas três regras simples (parecidas com “não colida-se com outros”) e de repente você tem comportamento complexo quando o bando vai batendo as asas graciosamente pelos céus, se remodelando em torno de obstáculos e assim por diante. Nenhum desses comportamentos avançados (como se remodelar na mesma forma ao redor de obstáculos) é especificado pelas regras; eles emergem da dinâmica do sistema.

Regras simples, como na simulação dos pássaros, leva a comportamentos complexos. Regras complexas, como com leis tributárias na maioria dos países, levam a comportamentos estúpidos.

Muitas práticas comuns de desenvolvimento de software tem o infeliz efeito-colateral de eliminar qualquer chance de comportamento emergente. A maioria das tentativas de otimização – amarrando alguma coisa muito explicitamente – reduz a extensão e escopo de interações e relacionamentos, que é a origem da emergencia. No exemplo do bando de pássaros, assim como sistemas bem-desenhados, são as interações e relacionamentos que criam os comportamentos interessantes.

Quanto mais amarramos as coisas, menos espaço deixamos para uma solução criativa e emergente. Seja tanto travando requisitos, antes de serem bem entendidos ou otimizando o código prematuramente, como inventando navegações e cenários de fluxo de trabalho complexas, antes de deixar o usuário final usar o sistema, o resultado é o mesmo: um sistema exageramente complicado e estúpido ao invés de um sistema limpo e elegante que aproveita a emergencia.

Mantenha pequeno. Mantenha simples. Deixe acontecer.

—Andrew Hunt, The Pragmatic Programmers

Os Três Mosqueteiros

Use uma equipe de três para a versão 1.0

Para a primeira versão da sua aplicação, comece com apenas três pessoas. Este é o número mágico que lhe dará força de trabalho suficiente sem lhe tirar o dinamismo e a agilidade. Comece com um desenvolvedor, um designer e um varredor (alguém que possa transitar entre ambos os mundos).

Claro, é um desafio desenvolver uma aplicação com poucas pessoas. Mas se você possuir a equipe certa, esta será valorosa. Pessoas talentosas não precisam de recursos infinitos. Elas prosperam no desafio de trabalhar com restrições e usam a criatividade para resolver problemas. Falta de recursos humanos força-o a lidar com sacrifícios mais cedo, o que é ótimo. Fará você entender suas prioridades mais cedo do que mais tarde. E você estará apto para comunicar-se sem ter constantemente que se preocupar se está deixando alguém de fora.

Se você não pode desenvolver sua primeira versão com apenas três pessoas, então ou você precisa de pessoas diferentes ou diminuir sua versão inicial. Lembre-se, tudo bem você lançar sua primeira versão pequena e consistente. Você rapidamente perceberá se sua idéia tem futuro e, se tiver, você terá uma base simples e limpa para progredir.

Lei de Metcalfe e equipes de projeto

Deixe a equipe tão pequena quanto possível. A lei de Metcalfe, “O valor de um sistema de comunicação cresce aproximadamente ao quadrado do número de usuários do sistema”, tem um corolário quando se trata de equipes de projeto: A eficiência da equipe é aproximadamente o inverso do quadrado do número de membros na equipe. Estou começando a achar que três pessoas é ótimo para a versão 1.0 de um produto… Comece por reduzir o número de pessoas que você planeja incluir na equipe, e então reduza um pouco mais.

—Marc Hedlund, entrepreneur-in-residence na O’Reilly Media

O fluxo da comunicação

A comunicação flui mais facilmente em equipes pequenas do que em grandes. Se você é a única pessoa no projeto, comunicação é simples. O único caminho de comunicação é entre você e o cliente. Com o aumento do número de pessoas em um projeto, aumenta também o número de caminhos de comunicação. E não aumenta de forma aditiva, como o número de pessoas, aumenta de forma multiplicativa, proporcional ao quadrado do número de pessoas.

—Steve McConnell, Chief Software Engineer na Construx Software Builders Inc.
(de: Less is More: Jumpstarting Productivity with Small Teams)

Abrace as Restrições

Deixe as limitações lhe guiar para soluções criativas

Nunca há suficiente para dar a volta. Sem tempo suficiente. Sem dinheiro suficiente. Sem pessoal suficiente.

Isso é uma coisa boa.

Em vez de se desesperar com essas restrições, aceite-as. Deixe que elas o guiem. Restrições incentivam inovação e forçam o foco. Em vez de tentar removê-las, use-as em seu benefício.

Quanto a 37signals estava desenvolvendo o Basecamp, nós tínhamos muitas limitações. Tínhamos:

- Uma empresa de design para administrar
- Trabalhos para clientes já existentes
- Uma diferença de 7 horas (O David estava programando na Dinamarca e o resto de nós nos Estados Unidos)
- Uma equipe pequena
- Nenhum finaciamento externo

Nós sentimos a depressão “sem suficiente”. Então mantivemos nossos pratos pequenos. Dessa maneira só poderíamos colocar até onde coubesse. Pegávamos grandes tarefas e quebrávamos em pedaços menores que atácavamos um de cada vez. Nos movemos passo a passo e priorizamos no caminho.

Isso nos forçou a chegar com soluções criativas. Baixamos nosso custo de mudança construindo sempre menos software. Demos às pessoas apenas as funcionalidades suficientes para resolver seus problemas do seu jeito – e então saíamos do caminho. A diferença de tempo e distância entre nós nos tornou mais eficientes na nossa comunicação. Em vez de nos encontrarmos em pessoa, comunicávamos exclusivamente via mensagens instantâneas e e-mail, o que nos forçava a ir direto ao ponto rapidamente.

Restrições normalmente são vantagens disfarçadas. Esqueça investimento externo, longos ciclos de lançamento e resoluções rápidas. Em vez disso, trabalhe com o que você tem.

Combata a destruição

O que já foi descrito como “elegância bizarra” é provavelmente melhor descrito como “funcionalidade destrutiva”, como um fungo em uma planta ele gradualmente elabora a embaça a verdadeira forma do produto enquanto drena suas energias. O antídoto para funcionalidade destrutiva é, claro, o “prazo final restritivo”. Isso resulta em funcionalidades serem descartadas por causa do tempo que levaria para implementá-las. Normalmente é o caso que as funcionalidades mais úteis levam a maior parte do tempo para implementar. Portanto a combinação da destruição e do prazo final gera software como conhecemos e amamos, formado de grande quantidade de funcionalidades inúteis.

—Jef Raskin, autor (de Por que Software é como é)

Seja Você Mesmo

Diferencie-se das companhias maiores sendo amigável e pessoal

Muitas pequenas empresas cometem o erro de tentarem atuar grande. É como se elas entendessem seu tamanho como uma fraqueza que precisa ser encoberta. Muito ruim. Ser pequeno pode realmente ser uma grande vantagem, especialmente quando isto representa comunicação.

Pequenas empresas gostam de menos formalidades, menos burocracia e mais liberdade. **Menores empresas são mais próximas dos clientes por padrão**. Isto significa que elas podem se comunicar com

seus clientes de forma mais direta e pessoal. Se a empresa é pequena, pode-se usar uma linguagem familiar ao invés de jargão. Seu site e seu produto podem ter uma voz humana ao invés de soar como um zumbido corporativo. Ser pequeno significa poder falar com os clientes, e não “se submeter a eles.”

Há também vantagens na comunicação interna em pequenas empresas. Você pode dispensar formalidades. Não há necessidade de processos árduos e múltiplas assinaturas para tudo. Todos no processo podem falar abertamente e honestamente. Este fluxo livre de idéias é uma das grandes vantagens de ser pequeno.

Seja orgulhoso, desafiadoramente sincero

Embora você possa pensar que um cliente pode ser logrado por exageros no número de empregados em sua companhia ou na amplitude de suas ofertas, os espertos, aqueles que realmente quer, sempre perceberão a verdade – seja por intuição ou dedução. De forma embaraçosa, Eu fiz parte de mentiras como essa no passado, e nenhuma dessas situações resultou algo que importasse para os negócios: relações duradouras, com significado e mutuamente benéficas com pessoas que possuem uma necessidade real pelos serviços oferecidos. O melhor caminho deveria ser orgulhoso, desafiadoramente sincero sobre o tamanho exato e a amplitude da companhia.

—Khoi Vinh, Subtraction.com

Sempre disponível

Não importa em qual negócio você está, um bom serviço ao cliente tornou-se o maior requisito que qualquer cliente estabelecerá. Nós demandamos isso dos serviços que usamos então por que com nossos clientes seria diferente? Desde o começo nós deixamos fácil e transparente para nossos clientes contatar-nos por toda e qualquer questão que tiverem. Em nosso website nós listamos um grande número de ferramentas gratuitas que redireciona para nossos celulares e nossos cartões de visita listam os números de cada um de nós. Nós enfatizamos para nossos consumidores que eles podem nos contatar a qualquer hora independente do problema. Nossos clientes apreciam esse nível de confiança ninguém jamais abusou deste serviço.

—Edward Knittel, Diretor de Vendas e Marketing, KennelSource

Prioridades capítulo 4

Qual é a Grande Idéia?

Diferencie-se das grandes empresas sendo pessoal e amigável

Explicitamente defina a visão principal da sua aplicação. O que a sua aplicação defende? Do que se trata? Antes de começar o design ou a codificação de qualquer coisa você precisa saber o propósito do seu produto – a visão. Pense grande. Para que ele existe? O que o torna diferente de outros produtos similares?

A visão irá guiar suas decisões e o manterá em um caminho consistente. Sempre que houver um ponto duvidoso, pergunte, “Estamos nos mantendo coerentes à visão?”

A visão deve ser breve também. Uma sentença deve ser o suficiente para espalhar a idéia. Aqui estão as visões para cada um de nossos produtos:

- **Basecamp**: Gerenciamento de Projetos é comunicação

- **Backpack**: Junte as pontas soltas da vida
- **Campfire**: Chat em grupo ao invés de Mensagens Instantâneas ruins
- **Ta-da List**: Competindo com os post-its
- **Writeboard**: Word é coisa demais

Com o Basecamp, por exemplo, a visão era “Gerenciamento de Projetos é comunicação”. Sentimos fortemente que comunicação efetiva em um projeto leva à propriedade coletiva, ao envolvimento, ao investimento e ao momento. Traz todos à mesma página trabalhando em direção a um objetivo comum. Sabíamos que se Basecamp pudesse atingir isso, todo o resto entraria na linha.

A visão nos levou a manter o Basecamp o mais aberto e transparente possível. Em vez de limitar a comunicação para dentro da empresa, demos acesso aos clientes também. Pensamos menos sobre permissões e mais sobre encorajar todos os participantes a tomar parte. A visão é o motivo porque pulamos painéis, gráficos, tabelas, relatórios, estatísticas e planilhas e ao invés disso focamos na prioridade da comunicação como mensagens, comentários, listas de tarefas e compartilhamento de arquivos. Tome a grande decisão sobre a visão logo no começo e todas as pequenas decisões futuras se tornam muito mais simples.

Filosofia do Quadro Branco

Andy Hunt e eu uma vez escrevemos um sistema de transações de cartão de débito. Um grande requisito era que o usuário de um cartão de débito não deveria ter a mesma transação aplicada à sua conta duas vezes. Em outras palavras, não importa que tipo de falha pudesse acontecer, o erro deveria ir para o lado de não processar a transação em vez de processar e duplicá-la.

Então, escrevemos isso no nosso quadro-branco compartilhado em letras grandes: Erros a favor dos usuários.

Isso se juntou a outra meia-dúzia de máximas. Juntas, elas guiaram todas aquelas decisões complicadas que se fazem quando se constrói algo complexo. Juntas, essas leis deram forte coerência interna e grande consistência externa à nossa aplicação.

—Dave Thomas, The Pragmatic Programmer

Faça um Mantra

Organizações precisam de pontos-guia. Precisam de linhas gerais; funcionários precisam saber a cada dia quando acordam porque estão indo trabalhar. Essas linhas devem ser curtas e doces, e bem compreensivas: Por que você existe? O que o motiva? Chamo isso de mantra – uma descrição de três ou quatro palavras de porque você existe.

—Guy Kawasaki, autor (de Make Mantra)

Ignore os Detalhes logo no Começo

Trabalhe do grande para o pequeno

Somos loucos por detalhes.

- O espaço entre objetos

- O espaço perfeito entre linhas
- A cor perfeita
- As palavras perfeitas
- Quatro linhas de código em vez de sete
- 90% vs 89%
- 760px vs 750px
- $39/mês vs $49/mês

Sucesso e satisfação estão nos detalhes

Entretanto, o sucesso não é a única coisa que encontrará nos detalhes. Também encontrará estagnação, desacordo, reuniões e atrasos. Essas coisas podem acabar com a moral e diminuir suas chances de sucesso.

Quantas vezes se encontrou travado em um único design ou elemento de código por um dia inteiro? Quantas vezes se deu conta de que o progresso que fez hoje não foi progresso real? Isso acontece quando você foca nos detalhes cedo demais no processo. Há tempo suficiente para ser um perfeccionista. Apenas faça isso mais tarde.

Não se preocupe com o tamanho da fonte do cabeçalho na primeira semana. Você não precisa empregar o tom perfeito de verde na segunda semana. Não precisa mover em três pixels o botão de “submeter” na terceira semana. Apenas coloque as coisas na página por enquanto. Então use. Garanta que funciona. Mais tarde você pode ajustar e aperfeiçoar.

Os detalhes se revelam ao se usar o que está construindo. Você verá o que precisa de mais atenção. Sentirá o que está faltando. Saberá quais crateras pavimentar porque ficará sempre caindo nelas. É quando precisa prestar atenção, e não antes.

O Diabo está nos Detalhes

Quase me cansei da atitude “entre nos detalhes imediatamente” depois de tomar algumas aulas de desenho … Se começar a desenhar os detalhes imediatamente pode ter certeza que o desenho será uma droga. De fato, você está perdendo completamente o ponto.

Você deve começar pegando as proporções corretas da cena toda. Então rascunha os grandes objetos na sua cena, indo até os menores. O rascunho deve ser bem vago nesse ponto. Então pode proceder sombreando, o que consiste em dar volume à vida. Você começa com apenas três tons (claro, médio, escuro). Isso dá um rascunho de tons. Então, para cada porção do seu desenho reavalia três tons e os aplica. Faça isso até os volumes aparecerem (requer múltiplas iterações) ...

Funciona do grande para o pequeno. Sempre.

—Patrick Lafleur, Creation Object Inc. (de Signal vs. Noise)

Só é um Problema Quando é um Problema

Não desperdice tempo com problemas que você ainda não tem

Você precisa realmente se preocupar em escalar para 100.000 usuários hoje se vai levar dois anos para

chegar lá?

Você tem mesmo que contratar oito programadores se hoje você só precisa de três?

Você precisa realmente de 12 servidores top-de-linha agora se dá para rodar em dois por um ano?

Apenas se vire

As pessoas costumam gastar tempo demais logo de cara tentando resolver problemas que elas ainda nem têm. Não faça isso. Poxa, nós lançamos o Basecamp sem a habilidade de cobrar os clientes! Como o produto é cobrado mensalmente, sabíamos que teríamos um intervalo de 30 dias para dar um jeito. Usamos aquele tempo para resolver problemas mais urgentes e então, após o lançamento, enfrentamos a cobrança. Deu certo (e nos forçou a adotar uma solução simples, sem firulas desnecessárias).

Não esquente com uma coisa até que você tenha de fato que fazê-lo. Não desenvolva demais. Aumente hardware e software de sistema conforme necessário. Se ficar lento por uma ou duas semanas não será o fim do mundo. Apenas seja honesto: explique para os seus clientes que você está passando por dores de crescimento. Eles podem não ficar empolgados mas apreciarão a franqueza.

Resumo da Ópera: Tome decisões só no momento necessário, pois aí você terá acesso à informação real de que precisa. Entrementes você estará em condições de prestar atenção às coisas que requerem cuidado imediato.

Contrate os Clientes Certos

Encontre o nicho de mercado para seu aplicativo e concentre-se somente nele

O cliente nem sempre tem razão. A verdade é que você terá que separar **quem é certo e quem é errado** para seu aplicativo. A boa notícia é que a Internet torna mais fácil do que nunca encontrar as pessoas certas.

Se você tentar agradar todo mundo, não irá agradar ninguém

Quando nós desenvolvemos o Basecamp, focamos nosso marketing em firmas de design. Por restringir nosso mercado desta forma, ficou mais fácil atrair clientes passionais que, por sua vez, iriam evangelizar o produto. Saiba para quem seu aplicativo realmente se destina e foque-se em agradar este público.

A Melhor Decisão Que Já Tomamos

A decisião de direcionar o Campaign Monitor estritamente para o mercado de web design foi **a melhor escolha** que já fizemos. Ela nos permitiu identificar facilmente quais recursos seriam genuinamente úteis e, mais importante, quais recursos deixar de fora. Não só atraímos mais clientes por mirar em um grupo menor de pessoas, como todos esses clientes tinham necessidades similares que tornavam nosso trabalho muito mais fácil. Há um monte de recursos no Campaign Monitor que seriam inúteis para qualquer um exceto um web designer.

Focar um nicho de mercado também torna muito mais fácil divulgar seu software. Agora que temos um público bem definido, podemos fazer anúncios em lugares da web que este público freqüenta, publicar artigos que eles podem achar interessantes e em geral formar uma comunidade em torno do produto.

—David Greiner, fundador, Campaign Monitor

Escale mais Tarde

Você ainda não tem um problema de escalabilidade

”Conseguirei escalar minha aplicação quando milhões de pessoas começarem a usá-la?”

Quer saber? Espere até que isso aconteça de fato. Se você tiver um número gigante de pessoas sobrecarregando seu sistema, magnífico! Que ótimo problema para se ter. A verdade é que a maioria esmagadora das aplicações web nunca alcançará esse estágio. E mesmo se você começar a ser sobrecarregado isto tipicamente não é uma questão de tudo ou nada. Você terá tempo para ajustar-se e responder ao problema. Além disso, depois de lançar você terá mais dados reais e benchmarks que podem ser usados para descobrir as partes que precisam ser revisadas.

Por exemplo, nós rodamos o Basecamp em um único servidor durante o primeiro ano. Por termos iniciado com uma configuração simples, conseguimos implementar em uma única semana. Nós não começamos com um aglomerado de 15 computadores nem gastamos meses nos preocupando com escalabilidade.

Tivemos problemas? Alguns. Mas também descobrimos que a maior parte do que temíamos, como um breve período de lentidão, não era o fim do mundo para os clientes. Desde que você mantenha as pessoas informadas, e seja honesto sobre a situação, elas entenderão. Em retrospecto, somos contentes por não termos atrasado o lançamento em meses para criar “a configuração perfeita”.

No começo, priorize construir um produto sólido em vez de obsecar-se com escalabilidade ou fazendas de servidores. **Crie uma grande aplicação e depois se preocupe com o que fazer quando ela se tornar animalmente bem-sucedida.** Do contrário você o corre o risco de desperdiçar energia, tempo e dinheiro se prendendo a algo que nunca acontecerá.

Acredite ou não, o maior problema não é escalar, é chegar ao ponto de ter de fazê-lo. Sem o primeiro problema, você não terá o segundo.

Você terá que revisar de qualquer jeito

A verdade é que todo mundo tem problemas de escalabilidade, ninguém lida com a transição de zero para alguns milhões de usuários sem revisar quase todos os aspectos do design e arquitetura da aplicação.

—Dare Obasanjo, Microsoft (de Scaling Up and Startups)

Faça Software que tem Opinião

Seu aplicativo deve tomar partido

Algumas pessoas defendem que o software deve ser agnóstico. Dizem que é arrogante da parte dos desenvolvedores limitar a funcionalidade ou ignorar pedidos de novos recursos. Dizem que o software deve ser sempre o mais flexível possível.

Para nós isso é papo-furado. O melhor software traz consigo uma visão. O melhor software toma partido. Quando alguém usa um software, não está procurando apenas recursos, está procurando uma abordagem. Está procurando uma visão. Decida qual é sua visão e atenha-se a ela.

E lembre, se não gostarem da sua visão há um monte de outras visões por aí. Não corra atrás de quem você nunca irá contentar.

Um ótimo exemplo é o projeto original do wiki. Ward Cunningham e seus amigos deliberadamente desproveram o wiki de muitos recursos que no passado eram considerados parte indispensável da colaboração de documentos. Em vez de atribuir cada mudança do documento a uma pessoa determinada, eles removeram muito da representação visual de propriedade. Eles tornaram o conteúdo atemporal e destituído de ego. Eles decidiram que não importava quem escreveu o conteúdo ou quando ele foi escrito. E isso fez toda a diferença. Essa decisão despertou nas pessoas um senso de comunidade e foi peça-chave no sucesso da Wikipédia.

Nossos aplicativos trilharam um caminho parecido. Eles não tentam ser todas as coisas para todas as pessoas. Eles têm uma atitude. Eles vão atrás de clientes que são no fundo parceiros. Eles têm apelo para as pessoas que partilham de nossa visão. Ou se está do lado de dentro ou se está do lado de fora.

Seleção de Funcionalidades capítulo 5

Meio, Não Meia-Boca

Faça meio produto e não um produto meia-boca

Cuidado com a visão “faz-tudo” no desenvolvimento de uma aplicação web. Considere todas as boas idéias que aparecerem ao longo do processo e você acabará apenas com uma versão meia-boca do seu produto. O que você realmente precisa é montar meio produto que detone.

Atenha-se ao que é verdadeiramente essencial. Boas idéias podem ser tiradas da gaveta. **Pegue tudo que você acha que seu produto deve ser e corte pela metade**. Remova funcionalidades até que você obtenha apenas o essencial. E então, repita o processo.

Começamos o *Basecamp* apenas com a seção de mensagens. Nós sabíamos que isso era o coração do aplicativo, então, de início, ignoramos as *milestones*, listas de tarefas e outros itens. Isso nos permitiu embasar as decisões futuras no uso real e não em palpites.

Comece com um aplicativo simples e inteligente e deixe-o ganhar impulso. Só então pense em adicionar coisas à fundação sólida que você já construiu.

Isso Simplesmente Não Importa

Apenas o essencial

Nossa resposta favorita à pergunta *”porque você não fez isso ou porque você não fez aquilo?”* é sempre: “Porque isso simplesmente não importa.” Essa afirmação representa o que torna genial um produto. Descobrir o que importa e deixar de fora o resto.

Quando lançamos o *Campfire*, nós recebemos essas perguntas das pessoas que usavam o produto pela primeira vez:

”Porque marcar o horário apenas a cada 5 minutos? Porque não marcar a hora de cada linha do batepapo?”

Resposta: Porque não importa. Com que frequência você precisa ter controle de uma conversa segundo a segundo, ou mesmo minuto a minuto? Certamente não é 95% do tempo. Marcar o horário a cada 5 minutos são suficientes porque qualquer outra frequência mais específica não importa.

"Porque vocês não permitem negrito ou itálico ou formatação colorida nos batepapos?" Resposta: Porque não importa. Se você necessita enfatizar algo, use a boa e velha trava de maiúsculas ou coloque alguns * em volta da palavra ou frase. Essas soluções não requerem nenhum *software* adicional, suporte técnico, capacidade de processamento ou curva de aprendizado. Além disso, formatação de texto num simples batepapo baseado em texto não importa.

"Porque vocês não mostram o total de pessoas na sala?" Resposta: Porque não importa. O nome de todos está listado, então você sabe quem está aí, que diferença faz saber se há 12 ou 16 pessoas? Se isso não muda o seu comportamento então isso não importa.

Seria legal ter essas coisas? Claro. Elas são essenciais? Elas realmente importam? Não. Por isso foram deixadas de fora. Os melhores designers e os melhores programadores não são os com as melhores habilidades, ou os dedos mais ágeis, ou os que podem assoviar e chupar cana com o Photoshop ou sua plataforma preferida, e sim aqueles que podem determinar o que não importa. É aí onde os ganhos reais são feitos.

A maior parte do tempo que você gasta é perdido em coisas que não importam. Se você puder cortar o tempo pensando no que não importa, você atingirá níveis de produtividade que você jamais imaginou.

Comece com Não

Faça com que as funcionalidades dêem duro para ser implementadas

O segredo de criar meio produto ao invés de um produto meia-boca é dizer não.

Cada vez que você diz sim para uma funcionalidade, você está adotando um filho. Você tem que levar seu bebê através de toda uma cadeia de eventos (exemplo: *design*, implementação, testes etc.). Uma vez que está funcionalidade está lá, você está preso a ela. Apenas tente removê-la e veja o quão irados ficarão os clientes.

Não concorde com tudo

Faça com que cada funcionalidade dê duro para ser implementada. Ponha cada uma delas à prova e mostre que é uma sobrevivente. É como no filme "O Clube da Luta". Você deveria considerar apenas funcionalidades que estejam dispostas a ficar aguardando na porta por três dias para serem aceitas.

É por isso que você tem que começar com um não. Cada novo pedido de funcionalidade que vem até nós – ou de nós – encontra um não. Nós ouvimos mas não agimos. A resposta inicial é "agora não". Se o pedido continua a aparecer, então sabemos que é hora de um olhar mais profundo. Somente então nós começamos a pensar na funcionalidade de fato.

E o que dizer às pessoas que reclamam quando nós não adotamos a sua idéia? Lembre-os do porque eles gostam da aplicação em primeiro lugar. "Você gosta dele porque nós dizemos não. Você gosta dele porque ele não faz outras 100 coisas. Você gosta dele porque ele não tenta agradar a todos sempre."

"Nós não queremos milhares de funcionalidades"

Steve Jobs deu uma pequena apresentação sobre o *iTunes Music Store* para um pessoal de uma gravadora

independente. Minha fala favorita nesse dia foi quando as pessoas insistiam em levantar a mão perguntando, “Ele faz [x]?”, “Você planeja adicionar [y]?”. Finalmente Jobs disse, “Calma, calma – abaixem seus braços. Ouçam: Eu sei que vocês tem milhares de idéias de funcionalidades bacanas para o iTunes. Nós também. Mas não queremos milhares de funcionalidades. Isso seria horrível. Inovação não é dizer sim para tudo. É dizer NÃO para tudo exceto as funcionalidades mais cruciais.”

—-Derek Sivers, presidente e programador, CD Baby e HostBaby
(from Diga NÃO por padrão)

Custos Ocultos

Exponha o preço das novas funcionalidades

Mesmo que uma funcionalidade passe o estágio do “não”, você ainda precisa expor seus custos ocultos.

Por exemplo, fique de alerta com *loops* de funcionalidades, ou seja, funcionalidades que levam a mais funcionalidades. Nós recebemos pedidos para adicionar uma aba de reuniões ao *Basecamp*. Parece simples até que você examine com mais cautela. Pense em todos os diferentes itens que uma aba de reuniões precisaria: localização, hora, sala, pessoas, convites por e-mail, integração com o calendário, documentação de suporte, etc. Isso sem mencionar que nós teríamos que modificar as imagens de promoção, as páginas do tour, páginas do faq/ajuda, contrato de prestação de serviço e mais. Antes que você note, uma idéia simples pode ser tornar uma dor de cabeça enorme.

Para cada nova funcionalidade você precisa…

- 1. Dizer não.
- 2. Forçar a funcionalidade a provar seu valor.
- 3. Se “não” novamente, pare aqui. Se “sim”, continue…
- 4. Esboce as telas/UI.
- 5. Crie as telas/UI.
- 6. Programe-as.
- 7-15. Teste, aperfeiçoe, teste, aperfeiçoe, teste, aperfeiçoe…
- 16. Cheque para ver se o texto da ajuda precisa ser modificado.
- 17. Atualize o tour do produto (se necessário).
- 18. Atualize a cópia de marketing (se necessário).
- 19. Atualize o Termo de Prestação de Serviço (se necessário).
- 20. Cheque se alguma promessa foi quebrada.
- 21. Cheque se a estrutura de custos foi afetada.
- 22. Publique.
- 23. Cruze os dedos.

Você Pode Lidar com Isso?

Crie algo que você possa gerenciar

Se você lançar um programa de filiação, você teria os sistemas prontos para fazer a contabilidade e os pagamentos? Talvez seria melhor você deixar as pessoas ganhar descontos nas suas taxas de filiação ao invés de escrever, assinar e enviar cheques todos os meses.

Você consegue dar 1 GB de espaço de graça, só porque o Google dá? Talvez você deva começar pequeno, em 100 mb, ou ceder espaço apenas para as contas pagantes.

Conclusão: Crie produtos e ofereça serviços que você possa gerenciar. É fácil fazer promessas. É bem mais díficl mantê-las. Tenha certeza de que, seja lá o que você fizer, seja algo que você possa realmente sustentar – organizacional, estratégica e financeiramente.

Soluções Humanas

Crie *softwares* voltados para conceitos gerais e incentive as pessoas a criar suas próprias soluções

Não force convenções. Ao invés disso, faça seu *software* de modo generalista, assim todos podem encontrar suas próprias soluções. Dê às pessoas só o suficiente para resolver os problemas delas ao modo delas. E então saia do caminho.

Quando criamos a *Ta-da List*, nós intencionalmente omitimos uma série de coisas. Não há como designar uma tarefa para alguém, não há como marcar uma data de término, não há como categorizar os itens, etc.

Nós mantivémos a ferramenta limpa e sem frescuras deixando as pessoas serem criativas. As pessoas descobriram como resolver seus problemas sozinhas. Se elas quisessem adicionar uma data para uma tarefa, elas poderiam inserir (Prazo: 7 de Abril de 2006) na frente do item. Se elas quisessem categorizar, poderiam simplesmente colocar [Livros] na frente do item. Ideal? Não. Infinitamente flexível? Sim.

Se tentássemos criar *software* para, especificamente, cuidar desses casos, nós estaríamos o tornando menos útil para todos os casos onde essas preocupações não se aplicam.

Faça o melhor trabalho que você puder com a raiz do problema e então saia de fininho. As pessoas vão encontrar suas próprias soluções e convenções dentro de sua estrutura geral.

Esqueça Pedidos de Funcionalidades

Deixe os clientes informarem o que é importante

Os clientes querem absolutamente tudo. Eles virão com uma avalanche de pedidos de funcionalidades. Dê uma olhada nos fóruns de nossos produtos; A categoria ‘pedido de funcionalidade’ sempre sobrepuja as com larga vantagem.

Nós vamos ouvir sobre “essa pequena funcionalidade extra” ou “não pode ser difícil” ou “não seria fácil

colocar isso” ou “vai levar apenas uns segundos para inserí-la” ou “se você adicionar isso, eu pagaria o dobro” e assim por diante.

Claro que não podemos culpar as pessoas por pedir funcionalidades. Nós as encorajamos e queremos ouvir o que elas tem a dizer. A maior parte das funcionalidades que inserimos em nossos produtos começaram como sugestões de nossos clientes. Mas, como dissemos antes, sua primeira resposta deve ser um não. Então o que você faz com todos esses pedidos? Onde você os guarda? **Como você os gerencia? Você não faz isso. Você apenas os lê e então os joga fora**.

Sim, leia, jogue fora e esqueça-os. Pode soar como heresia mas os realmente importantes irão, com certeza, reaparecer. Esses são os únicos que você precisa se lembrar. Esses são os realmente esseciais. Não se preocupe em organizar e guardar cada pedido que aparecer. Deixe seus clientes serem sua memória. Se a funcionalidade for realmente necessária, eles te lembrarão até que você não consiga esquecer.

Como nós chegamos à essa conclusão? Quando nós lançamos o *Basecamp* nós colocávamos todos os pedidos de novas funcionalidades em uma lista de tarefas. A cada repetição de um pedido, nós marcávamos com um “I” extra (II, III ou IIII etc). Nós imaginávamos que um dia iríamos revisar a lista e trabalhar indo das mais para as menos populares.

Mas a verdade é que nunca olhamos para a lista novamente. Nós já sabíamos o que precisava ser feito porque nossos clientes nos lembravam constantemente fazendo sempre os mesmos pedidos. Não havia necessidade de uma lista ou grupos de análise, porque tudo estava acontecendo em tempo real. Você não pode esquecer o que é importante quando você é lembrado todos os dias.

E mais uma coisa: Só porque x pessoas pedem algo, não significa que você tem que incluí-la. Algumas vezes é melhor apenas dizer não e manter sua visão de produto.

Segure as Rédeas

Pergunte o que as pessoas *não* querem

A maior parte das enquetes e pesquisas sobre *software* são focalizadas em o que as pessoas querem num produto. “Qual funcionalidade você acha que está faltando?”, “Se você pudesse adicionar mais uma única coisa, o que seria?”, “O que tornaria esse produto mais útil para você?”

E o outro lado da moeda? Porque não perguntar o que as pessoas nao querem? “Se você pudesse remover algo, qual seria?” “O que você não usa?” “O que mais te atrapalha?”

Mais não é a resposta. Algumas vezes o maior favor que você pode fazer para os clientes é deixar algo de fora.

Inovação aparece quando dizemos não

[Inovação] aparece quando dizemos não à 1000 coisas para termos certeza que não estamos seguindo o caminho errado ou tentando fazer coisas demais. Nós estamos sempre pensando em novos mercados para entrar, mas somente dizendo não para isso que você pode se concentrar no que realmente importa.

—Steve Jobs, CEO, Apple (de A Semente de Inovação da Apple)

Processo capítulo 6

Corra para Rodar o Software

Pegue algo real e ponha-o para rodar rapidamente

Rodar o software é a melhor maneira de fazer acontecer, reúna sua equipe e jogue fora idéias que não funcionam. Esta deve ser sua prioridade número um.

Não tem problema fazer menos, pular detalhes e encontrar atalhos em seus processos se o levarem a software que funciona mais rápido. Uma vez lá, você será recompensado com perspectivas significativamente mais precisas de como proceder. Histórias, rascunhos de estrutura e mesmo protótipos html são apenas aproximações. Software rodando é real.

Com software real rodando todos ficam perto da compreenção e do acordo. Você evita argumentos acalorados sobre esboços e parágrafos que não resolveriam nada de importante. Você percebe que coisas que considerava triviais são na verdade bem cruciais.

Coisas reais levam a reações reais. E é assim que você chega à verdade.

As coisas reais levam ao acordo

Quando um grupo de pessoas diferentes tentam encontrar o que é harmonioso… suas opiniões tendem a convergir se eles estão rascunhando coisas reais em larga escala. Claro, se estão fazendo um esboço ou gerando idéias, não vão concordar. Mas se você começa fazendo coisas reais, tendem a chegar a um acordo.

—Christopher Alexander, Professor de Arquitetura
(de Contrastando Conceitos de Harmonia na Arquitetura)

Faça Funcionar o Mais Rápido Possível

Eu não lembro de nenhum projeto de software que estive envolvido – grande ou pequeno – que tenha tido sucesso em termos de prazos, custos ou funcionalidades que tenha começado com um longo período de planejamento e discussão, e sem desenvolvimentos concorrentes. É simplesmente muito fácil e às vezes divertido, jogar o precioso tempo fora inventando funcionalidades que acabam sendo desnecessárias ou que não serão implementáveis.

Isso se aplica a qualquer tipo de desenvolvimento e chegar a alguma coisa real e rodando é um mantra fractal. Isto não se aplica apenas a projetos como um todo, é pelo menos igualmente aplicável ao desenvolvimento de componentes de pequena escala, dos quais a aplicação é feita.

Quando há trabalho de implementação de um componente chave, desenvolvedores querem entender como vai ou não trabalhar em conjunto com suas partes da aplicação e irão geralmente tentar usá-lo assim que puderem. Mesmo que a implementação não esteja perfeita ou completa no início, estas colaborações prematuras normalmente levam a interfaces bem definidas e funcionalidades que fazem exatamente o que se espera delas.

—Matt Hamer, desenvolvedor e gerente de produtos, Kinja

Enxague e Repita

Trabalhe por iterações

Não espere fazer certo na primeira vez. Deixe a aplicação crescer e se apresentar a você. Deixe ela mutar e envolver. Com softwares baseados na web não há necessidade de entregar a perfeição. Projete as telas, use-as, analize-as e então comece de novo.

Ao invés de querer tudo certo desde o início, o processo iterativo lhe permite continuar tomando decisões informadas no percurso. Você ainda terá uma aplicação ativa e rodando rapidamente, desde que não fique obcecado em atingir a perfeição logo de cara. O resultado é um feedback e um guia real sobre o que requer sua atenção.

Iterações levam à liberação

Você não precisa almejar a perfeição na primeira tentativa se sabe que isto será feito de novo mais tarde. Saber que revisitará problemas é um grande motivador para apenas lançar as idéias e ver se voam.

Talvez você seja mais esperto do que eu

Talvez você seja BEM mais esperto do que eu.

Isto é totalmente possível. De fato, isto é provável. De qualquer maneira, se você é como a maioria das pessoas, então como eu, tem problemas imaginando aquilo que não consegue ver e tocar.

Seres humanos são extremamente bons em responder a coisas do ambiente. Nós sabemos entrar em pânico quando um tigre entra na sala e como limpar tudo depois de uma enchente devastadora. Infelizmente somos terríveis em planejar adiante, em compreender ramificações das nossas ações e em priorizar as coisas que realmente tem importância.

Talvez você seja um dos poucos indivíduos que consegue manter isto tudo em sua cabeça. Isto realmente não interessa.

Web 2.0, o mundo que começamos assumindo que todo mundo já usa a web, dá permissão a desenvolvedores espertos colocarem essas fraquezas humanas a trabalhar para elas. Como? Permitindo que seus usuários lhes contem o que pensam enquando ainda há tempo de fazer algo a respeito.

E esta última frase explica porque você deve desenvolver desta maneira e como pode querer promover/lançar.

Torne sua história direta. Garanta que as peças funcionem. Então lance e revise. Ninguém é tão esperto quanto todos nós juntos.

—*Seth Godin, autor/empreendedor*

Da Idéia à Implementação

Vá do brainstorm à esboços à HTML à codificação

Aqui vai o processo que usamos para Cair na Real:

Brainstorm

Traga idéias à tona. O que este produto irá fazer? Para o Basecamp, nós olhamos para nossas próprias necessidades. Queríamos publicar atualizações de projeto. Queríamos participação dos clientes. Sabíamos que projetos tinham datas-chave. Queríamos centralizar arquivos para que as pessoas pudessem revisar coisas antigas com facilidade. Queríamos ter uma visão da figura maior, uma vista aérea do que estava acontecendo com todos os nossos projetos. Juntas, estas premissas e algumas outras, serviram como nossa fundação.

Esse estágio nao é sobre os mínimos detalhes. É sobre grandes questões. O que a aplicação precisa fazer? Como saberemos quando será útil? O que exatamente faremos? Isso é sobre idéias de alto nível, nao discussões no nível dos pixels. Nesse estágio, esses tipos de detalhe simplesmente não têm sentido.

Papel de Padeiro

Esboços são rápidos, sujos e baratos e é exatamente como você quer começar. Desenhe coisas. Rabisque coisas. Caixas, círculos, linhas. Arranque as idéias da cabeça para o papel. O objetivo nesse ponto deve ser converter conceitos em designs grosseiros de interface. Esse passo é apenas sobre experimentação. Não há respostas erradas.

Crie telas HTML

Faça uma versão HTML dessa funcionalidade (ou seção, ou fluxo, se for mais apropriado). Pegue algo real e publique para que todos possam ver como fica na tela.

Para o Basecamp, primeiro fizemos a tela de “postar mensagens”, então a tela de “editar mensagens” e a coisa prosseguiu daí.

Não escreva nenhum código de programação ainda. Apenas faça um protótipo em html e css. A implementação vem depois.

Codifique

Quando o protótipo parecer bom e demonstrar o suficiente das funcionalidades necessárias, vá em frente e conecte o código de programação.

Durante todo esse processo, se lembre de permanecer flexível e esperar múltiplas iterações. Você deve se sentir livre para jogar fora qualquer parte entregável de qualquer passo particular e começar novamente se ela se mostrar lixo. É natural passar por esse ciclo múltiplas vezes.

Evite Preferências

Decida sobre os pequenos detalhes para que seus clientes não precisem

Nos deparamos com uma decisão difícil: quantas mensagens incluímos em cada página? A primeira inclinação pode ser em dizer, “Vamos apenas tornar isso uma preferência onde as pessoas possam escolher 25, 50 ou 100”. Entretanto, essa é a forma mais simples. Apenas tome a decisão.

Preferências são uma maneira de evitar tomar decisões difíceis

Em vez de usar nossa experiência para escolher o melhor caminho, deixamos nas mãos dos clientes. Pode parecer que estamos fazendo um favor a eles mas apenas estamos lhes dando mais trabalho (e

provavelmente eles já são ocupados o suficiente). Para os clientes, telas de preferência com uma quantidade infinita de opções são uma dor de cabeça, não uma bênção. Clientes não deveriam ter que pensar sobre cada ínfimo detalhezinho – não coloque esse peso sobre eles quando deveria ser sua responsabilidade.

Preferências também são más porque criam mais software. Mais opções requerem mais código. E também ainda tem todo o teste extra e design que precisamos fazer. E ainda vamos acabar com permutações de preferências e telas que nunca usaremos. Isso significa bugs que não sabemos a respeito: layouts quebrados, tabelas explodindo, problemas estranhos de paginação, etc.

Tome a decisão

Tome as decisões simples no lugar dos clientes. É o que fizemos no Basecamp. O número de mensagems por página é 25. Na página de resumo, as últimas 25 são mostradas. Mensagens são ordenadas em ordem cronológica reversa. Os cinco projetos mais recentes são mostrados no painel. Não existem opções. É o jeito que tem que ser.

Sim, podemos tomar uma decisão ruim. Mas e daí? Se fizermos isso, as pessoas vão reclamar e nos dizer sobre isso. Como sempre, podemos ajustar. Cair na Real é justamnete sobre ser capaz de mudar em tempo real.

Preferências Têm um Custo

No fim das contas preferências tem um custo. Claro, algumas preferências também têm benefícios importantes – e podem ser funcionalidades de interface cruciais. Mas cada um tem um preço e temos que considerar cuidadosamente seu valor. Muitos usuários e desenvolvedores não entendem isso e acabam com muito custo e pouco valor por seus dólares de preferências … acho que se formos duramente disciplinados sobre ter bons padrões Que Apenas Funcionam, em vez de adicionar preferências folgadamente, isso naturalmente leva a interface de usuário como um todo na direção certa.

—Havoc Pennington, líder técnico, Red Hat (de Software Livre e boas interfaces de usuário)

"Feito !"

Decisões são temporárias então faça a escolha e siga em frente

Feito. Comece a pensar nisso como uma palavra mágica. Quando algo está feito significa que algum objetivo foi atingido. Uma decisão foi tomada e podemos seguir em frente. Feito significa que estamos construindo momento.

Mas espere, e se ferramos e tomamos a decisão errada? Tudo bem. **Isso não é cirurgia de cérebro, é uma aplicação web**. Como estamos dizendo, provavelmente teremos que rever funcionalidades e idéias múltiplas vezes durante o processo de qualquer forma. Não importa quanto planejamos, com certeza estaremos meio errados de qualquer jeito. Então não faça essa coisa de “pausa para análise”. Isso apenas desacelera o progresso e compromete a moral.

Em vez disso, avalie a importância de seguir em frente. Entre no ritmo de tomar decisões. Tome uma decisão rápida e simples e então retorne e mude se não funcionar.

Aceite que decisões são temporárias. Aceite que erros vão acontecer e entenda que não tem nada demais enquanto estivermos fazendo correções rapidamente. Execute, construa momento, e siga em frente.

Seja um Executador

É tão engraçado quando ouço sobre pessoas protegendo tanto suas idéias. (Pessoas que querem que eu assine um contrato de sigilo antes de me contar a mais simples das idéias).

Para mim, idéias não valem nada até serem executadas. São apenas multiplicadores. Execução vale milhões.

Explicação:

- Idéia Péssima = -1
- Idéia Fraca = 1
- Idéia mais ou menos = 5
- Boa Idéia = 10
- Grande Idéia = 15
- Brilhante Idéia = 20
- Nenhuma execução = $1
- Execução Fraca = $1.000
- Execução mais ou menos = $10.000
- Boa Execução = $100.000
- Grande Execução = $1.000.000
- Brilhante Execução = $10.000.000

Para fazer negócios, você precisa multiplicar os dois.

A idéia mais brilhante, sem nenhuma execução, vale $20. A idéia mais brilhante necessita de grande execução para valer $20.000.000.

Esse é o motivo porque não quero ouvir idéias de outras pessoas. Não estou interessado até ver suas execuções.

—Derek Sivers, presidente e programador, CD Baby e HostBaby

Teste ao Ar Livre

Teste sua aplicação com uso do mundo real

Não há substituto para pessoas reais usando sua aplicação de maneiras reais. Pegue dados reais. Receba feedback real. Então aprimore baseado nessa informação.

Testes de usabilidade formais são muito duros. Configurações de laboratório não refletem a realidade. Se ficarmos sobre os ombros de alguém, teremos alguma idéia do que está funcionando ou não mas as pessoas normalmente não agem bem de frente para as câmeras. Quando outra pessoa está olhando, as pessoas são especialmente mais cuidadosas para não cometer erros – sendo que erros são exatamente o que estamos procurando.

Em vez disso, lance versões beta para alguns poucos selecionados dentro da própria aplicação real. Faça com que usem as funcionalidades do beta ao lado das funcionalidades lançadas. Isso irá expôr essas funcionalidades a dados reais das pessoas e a fluxos verdadeiros. E é aí que teremos resultados reais.

Além disso, não tenha uma versão de lançamento e outra beta. Elas devem ser sempre a mesma coisa. Uma versão beta separada só receberá uma leve navegação. A versão real, com algumas funcionalidades beta misturadas, receberão o fluxo de uso completo.

O Livro Beta

Se desenvolvedores estão nervosos liberando seus códigos, então editores e autores estão assustados lançando seus livros. Uma vez que o livro está fixo no papel, é visto como uma coisa cabeluda mudar (na verdade não é, mas percepção e lembranças de problemas com velhas tecnologias ainda persistem na indústria). Então, editores passam por vários problemas (e custos) tentando fazer os livros ficarem “certos” antes de serem lançados.

Quando escrevi o livro Agile Web Development With Rails, houve muita demanda entre os desenvolvedores: nos dê o livro agora – queremos aprender Rails. Mas eu caí no pensamento de um editor. “Não está pronto ainda”, eu dizia. Mas a pressão da comunidade e trocas de idéias com David Heinemeier Hansson mudaram minha forma de pensar. Lançamos o livro em formato pdf cerca de 2 meses antes de ficar completo. Os resultados foram espetaculares. Não apenas vendemos um monte de livros, mas recebemos feedback – muito feedback. Configurei um sistema automatizado para capturar os comentários dos leitores, e no final tivemos quase 850 relatos de erros de digitação, erros técnicos e sugestões para novo conteúdo. Quase todos encontraram seu caminho para o livro final.

Foi uma situação ganha-ganha: consegui entregar um livro muito melhor e a comunidade teve acesso antecipado a algo que eles queriam. E se você está numa corrida competitiva, ter algo antecipado ajuda as pessoas a se comprometerem com você e não com sua competição.

—Dave Thomas, The Pragmatic Programmers

Faça isso rápido

- 1. Decida se vale a pena fazer, e se for:
- 2. Faça rápido — Não perfeito. Apenas faça;
- 3. Grave. Faça upload. Publique;
- 4. Veja o que as pessoas acham.

Embora eu esteja sempre relutante em adicionar novas funcionalidades às coisas, quando tenho aquele momento "isso!" de decisão de que alguma coisa vale a pena fazer, normalmente está no ar no website algumas horas depois, com falhas mas lançado, deixando o feedback guiar o futuro dos refinamentos nele.

—Derek Sivers, presidente e programador, CD Baby e HostBaby

Encolha Seu Tempo

Quebre

Estimativas que esticam em semanas ou meses são fantasias. A verdade é que simplesmente não sabemos o que vai acontecer daqui tanto tempo à frente.

Então encolha seu tempo. Continue quebrando seu cronograma em pedaços menores. Em vez de um projeto de 12 semanas, pense nele como 12 projetos semanais. Em vez de chutar tarefas que levam 30 ou mais horas, quebre em pedaços mais realistas de 6 a 10 horas. Então proceda, um passo de cada vez.

A mesma teoria se aplica para outros problemas também. Você está enfrentando um problema tão grande que não cabe na sua cabeça? Quebre. Continue dividindo os problemas em pedaços cada vez menores até que você seja capaz de digerí-los.

Tarefas Menores e Cronogramas Menores

Desenvolvedores de software são uma espécie especial de otimistas: quando apresentados a uma tarefa de programação, eles pensam, “Isso será fácil! Não vai levar tanto tempo, afinal de contas”.

Então, dê três semanas a um programador para completar a enorme tarefa, e ele gastará duas semanas e meia procrastinando, e então uma programando. O atraso no cronograma provavelmente encontrará os requisitos errados, porque a tarefa se mostrou mais complexa do que parecia. Além disso, quem vai lembrar o que foi acordado entre a equipe três semanas atrás?

Dê a um programador uma tarde para codificar um módulo pequeno, específico e ele vai devorá-lo, pronto para ir para o próximo.

Tarefas menores e cronogramas menores são mais gerenciáveis, escondem menos requisitos mal entendidos e custam menos para você mudar de idéia ou refazer. Cronogramas menores mantém os desenvolvedores engajados e lhes dá mais oportunidades para aproveitar um senso de conquista e menos razões para pensar, “oh, eu tenho tempo suficiente para fazer isso. Por ora, vou terminar de categorizar minhas músicas no meu iTunes”.

—Gina Trapani, desenvolvedora web e editora da Lifehacker, o guia da produtividade e software

Fatores Verdadeiros

Da próxima vez que alguém o pressionar por uma resposta exata a uma questão desconhecida — seja sobre uma data de entrega, o custo final do projeto ou o volume de leite que caberia no Grand Canyon — apenas comece tirando o ar da sala: diga, "Eu não sei".

Longe de danificar sua credibilidade, isso demonstra o cuidado que você trás à sua tomada de decisões. Você não está dizendo palavras apenas para parecer esperto. Isso também nivela o campo de jogo reformulando a questão como uma conversa colaborativa. Sabendo quão exata sua estimativa precisa ser (e porque), você pode trabalhar junto para desenvolver um entendimento compartilhado sobre os verdadeiros fatores por trás dos números.

—Merlin Mann, criador e editor da 43folders.com

Resolva Aquele Problema Que Está Te Encarando na Cara

Minha coisa absolutamente favorita que acontece na web em tempos recentes é o lançamento e adoção do atributo "nofollow" ("não siga"). Ninguém falou sobre isso de antemão. Não houve conferências ou comitês onde um bando de gente poderia debater a semântica ou natureza gramatical. Nenhuma RFC que poderia tornar uma idéia simples em um pedaço de XML de 20 linhas que eu teria que ler de cabo a rabo apenas para entender como usá-lo, e então não usar porque eu não teria certeza se estava formatado para a versão .3 ou 3.3b

É simples, é efetivo, forneceu uma opção às pessoas que queriam uma opção — e isso é muito mais importante ao lidar com uma população da web que não se importa com especificações ou deferências.

Algumas vezes resolver os próximos vinte problemas não é tão útil ou prudente quanto resolver aquele um que está nos encarando diretamente na nossa cara. Não foi apenas uma pequena vitória contra o SPAM (todas as vitórias contra SPAM são pequenas), mas uma vitória para aqueles de nós que apreciam os resultados simples e diretos sobre ser um desenvolvedor web.

—Andre Torrez, programador e VP de Engenharia na Federated Media Publishing

---

A Organização capítulo 7

Unidade

Não quebre em áreas

Muitas empresas separam design, desenvolvimento, redação, suporte e marketing em áreas isoladas. Enquanto a especialização tem suas vantagens, também cria uma situação em que os funcionários só enxergam seus próprios mundos ao invés da aplicação web como um todo.

Integre sua equipe ao máximo para que exista um diálogo contínuo em todas as etapas do processo. Faça um sistema de verificações e balanços. Não deixe que coisas se percam nas transcrições. Tenha redatores trabalhando com designers. Faça com que os desenvolvedores tenham ciência dos pedidos de suporte.

Melhor ainda, contrate pessoas com múltiplos talentos, que podem atuar em diversas frentes. O resultado final será um produto mais harmonioso.

Tempo Sozinho

Pessoas precisam de períodos sem interrupções para terminar o trabalho

37signals está espalhada por quatro cidades e oito fusos-horário. De Provo, Utah a Copenhagen, na Dinamarca, cinco de nós estamos oito horas distantes. Um dos efeitos positivos de se ter oito horas de diferença é o tempo em que podemos ficar sozinhos.

Apenas 4 a 5 horas por dia estamos trabalhando juntos. Em algumas horas, a equipe norte-americana está dormindo enquanto David, que está na Dinamarca, está trabalhando. Nas horas restantes, estamos trabalhando enquanto David está dormindo. Isso nos dá meio dia juntos e meio dia sozinhos.

Adivinhe qual a parte do dia em que somos mais produtivos? O tempo em que estamos sozinhos. E isso não é nenhuma surpresa. Muitas pessoas preferem trabalhar logo de manhãzinha ou tarde da noite – nas horas em que não são incomodados.

Quando você tem um longo período em que não é incomodado, consegue se concentrar. E concentrado se é mais produtivo. É quando você não tem que dividir a cabeça com outros assuntos ou tarefas. É quando não se é interrompido para responder algo, ou procurar alguma coisa, ou enviar um email, ou responder mensagens instantâneas. O tempo sozinho é onde progressos de verdade acontecem.

Se concentrar leva tempo. E é exatamente por isso que a interrupção é seu maior inimigo. É como pegar no sono profundo – não se entra no sono profundo do nada, tem que se deitar, dormir e então entrar no sono profundo. Qualquer interrupção o força a começar tudo de novo. O sono profundo é onde a mágica do sono

acontece de verdade. A concentração é onde a mágica da produtividade acontece de verdade.

Estabeleça uma regra: faça que metade do dia seja de horas onde você fica sozinho. Das 10 da manhã até às 2 da tarde, ninguém pode falar com ninguém mais (exceto durante o almoço). Ou então, faça com que a manhã ou a tarde seja o seu tempo para ficar sozinho. O importante é que o período seja contínuo para evitar interrupções que matam sua produtividade.

Um período de tempo sozinho significa largar o vício da comunicação. Durante o tempo que ficar sozinho, esqueça as mensagens instantâneas, ligações telefônicas, reuniões. Evite qualquer conversa por e-mail que exija respostas imediatas. Em resumo: cale a boca e trabalhe.

Se Concentrando

Todos sabemos que profissionais sábios trabalham melhor entrando no “clima”, também chamado de “se concentrar”, onde ficam totalmente concentrados em seus trabalhos e completamente desligados dos seus ambientes. Eles perdem a noção do tempo e produzem muito mais através de concentração absoluta… o problema é que é muito fácil perder a concentração. Barulho, telefonemas, saída para o almoço, ter que dirigir por 5 minutos pra comer um pão de queijo e interrupções por colegas de trabalho – especialmente interrupções por colegas—tudo te tira da zona de concentração. Se você parar por 1 minuto para responder a uma pergunta de um colega de trabalho, e isso tirar sua concentração o suficiente para levar meia hora pra voltar a ser produtivo novamente, sua produtividade geral está em sérios problemas.

—Joel Spolsky, desenvolvedor de software, Fog Creek Software
(de De Onde Essas Pessoas Tiram Essas Idéias (Não Originais)?)

Reuniões São Tóxicas

Não tenha reuniões

Você precisa mesmo de reuniões? Reuniões geralmente acontecem quando um conceito não está claro o suficiente. Ao invés de recorrer a uma reunião, tente simplificar o conceito, para que você possa discutí-lo rapidamente por email ou IM ou Campfire. O objetivo é evitar reuniões. Cada minuto que você gasta em uma reunião é um minuto que você poderia estar trabalhando.

Não existe nada mais tóxico à produtividade do que uma reunião. Aqui vão alguns motivos:

- Elas quebram seu trabalho diário em pequenos períodos, que acabam por quebrar o fluxo do trabalho
- Elas geralmente tratam apenas de palavras e conceitos abstratos, não de coisas reais (como um trecho de código ou algum detalhe do design de interface)
- Elas geralmente tratam de uma pequena quantidade de informações por minuto
- Elas quase sempre tem uma pessoa que inevitavelmente vai fazer com que todos percam o tempo com assuntos não relacionados
- O assunto principal vai embora muito facilmente
- Freqüentemente tem pautas tão vagas que ninguém tem certeza do assunto principal
- Requerem uma preparação prévia, que quase ninguém faz

Em casos em que reuniões são *realmente* necessárias (faça disso um raro evento), siga estas regras simples:

- Coloque um alarme pra 30 minutos. Assim que ele tocar, a reunião acabou. Ponto final.
- Chame o menor número de pessoas possível.
- Nunca tenha uma reunião sem uma pauta bem clara.

Tenha menos reuniões

Existem muitas reuniões. Fazer reuniões que não fazem sentido é uma tarefa improdutiva. Só agende uma reunião quando você tem um assunto muito importante para discutir e você quer ou precisa de uma idéia, aprovação ou aval. Mesmo assim, resista bravamente à tentação de convidar todo mundo – não desperdice o tempo das outras pessoas sem necessidade.

—Lisa Haneberg, autora (de Não Deixe as Reuniões Ditarem as Regras!)

Quebre-as

Conforme o projeto cresce, o acréscimo de pessoas diminui a produtividade. Uma das razões mais interessantes é o aumento do número de canais de comunicação. Duas pessoas podem falar entre si; é um canal de comunicação único. Três pessoas tem três canais de comunicação; 4 tem 6. Na verdade, o crescimento dos canais é exponencial… Logo, memorandos e reuniões vão acabar consumindo o tempo todo.

A solução é clara: quebre as equipes em unidades pequenas, autônomas e independentes, para reduzir os canais de comunicação.

Da mesma forma, quebre os programas em unidades pequenas. Uma grande parte do problema vêm de dependências externas (variáveis globais, dados passados entre funções, hardware compartilhado, etc…), encontre um jeito de quebrar o programa para eliminar – ou minimizar – as dependências entre as unidades.

—Grupo Ganssle (de Mantenha Pequeno)

Procure e Celebre Pequenas Vitórias

Entregue algo hoje

A coisa mais importante em desenvolvimento de software é motivação. Motivação é localizada—se você não está motivado pelo que está trabalhando neste instante, as chances de que isso não saia tão bom são grandes . Na verdade, isso provavelmente vai ficar ruim.

Ciclos de entregas longos e demorados são assassinos de motivação. Eles separam muito o tempo entre as comemorações. Por outro lado, conquistas rápidas que você pode comemorar são grandes motivadores. Se você deixar que as entregas demorem a acontecer, você estará matando a motivação. E isso pode matar o produto.

Portanto, se você está no meio de um ciclo de entrega que demora meses, dedique um dia por semana (ou a cada duas semanas) para pequenas vitórias. Pergunte-se “o que nós podemos fazer e entregar em 4 horas?”. E então, faça isso. Este tipo de trabalho poderia ser ...

- Uma funcionalidade simples
- Um pequeno ajuste a algo que já existe

- Reescrever algum texto de ajuda, que reduz o número de pedidos de suporte
- Em alguns formulários, remova alguns campos que você não precisa

Quando conseguirem esta vitória de 4 horas, você vai encontrar a comemoração. E isso constrói a moral, aumenta a motivação e assegura que a equipe está na direção certa.

Contratando capítulo 8

Contrate Menos e Contrate Mais Tarde

Adicione devagar para andar rápido

Não existe necessidade de ficar grande logo – ou depois de algum tempo. Mesmo se você tem acesso às 100 melhores pessoas, não é bom tentar contratá-las todas de uma vez. Você não conseguirá fazer tanta gente assim assimilar a cultura da sua empresa. Você terá problemas no treinamento, disputas pessoais, problemas de comunicação, pessoas indo em direções opostas e muito mais.

Portanto, não contrate. Falando sério, não contrate. Procure outra saída. O trabalho está tão puxado assim a ponto de você realmente precisar de mais gente? Porquê você mesmo não faz? Você pode resolver o problema com algum software ou uma mudança nas práticas?

Toda vez que Jack Welch, antigo CEO da GE, precisava demitir alguém, ele não contratava alguém de imediato para substituí-la. Ele queria ver por quanto tempo a GE poderia sobreviver sem aquela pessoa e sem aquele cargo. Claro, não estamos incentivando ninguém a demitir pessoas para testar esta teoria, mas nós achamos que Jack acertou em algo: Você não precisa de tanta gente quanto você pensa.

Se não tem outro jeito, então pense em contratar. Mas você deve saber exatamente o que precisa, apresentar os candidatros ao trabalho e mostrá-los o tipo de sofrimento você quer que eles tenham.

Lei de Brooks

Adicionar pessoas a um projeto de software atrasado vai atrasá-lo ainda mais.

—Fred Brooks

Programação e Requiem de Mozart

Um único bom programador trabalhando em uma única tarefa não tem excesso de coordenação ou comunicação. Cinco programadores trabalhando na mesma tarefa precisam se coordenar e se comunicar. E isso toma muito tempo… O problema em usar muitos programadores medianos ao invés de alguns bons programadores é que, não importa o tempo que eles tomem, o resultado nunca vai ser tão bom quanto o que os bons programadores vão produzir. Cinco Antonio Salieri's não vão produzir um Requiem, de Mozart. Nem se trabalhassem por 100 anos.

—Joel Spolsky, desenvolvedor de software, Fog Creek Software (de Acertando as Notas Maiores)

Chute os Pneus

Trabalhe com possíveis funcionários na base do “teste antes”

Uma coisa é olhar o portfólio, curriculum, exemplo de código ou trabalhos anteriores. Outra coisa é efetivamente trabalhar com alguém. Sempre que possível, faça um “test-drive” com possíveis novos membros da equipe.

Antes de contratar alguém, passe a ele um pequeno projeto antes. Vamos ver como ele assume o projeto, como ele se comunica, como ele trabalha, etc. Trabalhar com alguém conforme ele faça o design ou codifique algumas telas vai te dar uma boa idéia sobre a pessoa. Você verá rapidamente se a pessoa é ou não o que você precisa.

O tempo de trabalho para este projeto pode ser pequeno. Mesmo que sejam 20 ou 40 horas, é melhor do que nada. Se o tempo é ou não suficiente, isso se tornará óbvio. Em todo caso, ambos os lados se livrarão do risco e dor de cabeça testando antes.

Comece com pouco

Faça um pequeno teste no começo. Não jogue todo o trabalho para a pessoa de uma vez. Dê a seu novo [assistente virtal] um ou outro projeto de teste e veja como a química se desenvolve. Neste começo, é mais fácil de detectar problemas em potencial. Deixe claro de que este é um período de experiência.

—Suzanne Falter-Barns, autora/especialista em criatividade
(de Como Encontrar e Manter o Assistente Virtual Perfeito)

Ações, Não Palavras

Julgue potenciais contratações de tecnologia em contribuições open source

O método tradicional de contratação para cargos técnicos—baseados em faculdades, curriculums, etc. —é falho por diversas razões. Realmente importa onde a pessoa é formada ou suas notas? Podemos confiar mesmo em um curriculum ou indicação?

Open source é uma dádiva para aqueles que precisam contratar técnicos. Com open source, pode-se checar o trabalho e contribuições de alguém—pra bem ou mal—com um bom intervalo de tempo.

Isso significa que você pode julgar pessoas pelas ações ao invés de apenas palavras. Você pode tomar decisões com base no que realmente importa:

- **Qualidade do trabalho**
  Muitos programadores falam bonito, mas afinam na hora do “vamos ver”. Com open source, você consegue ver com detalhes as práticas e conhecimentos de programação de uma pessoa.
- **Perspectiva cultural**
  Programar é tomar decisões. Muitas delas. Decisões são tomadas com base na cultura, nos valores e em ideais. Veja as decisões específicas feitas por um candidato enquanto está programando e testando, e veja seus argumentos na comunidade para ver se o candidato está dentro do que a empresa espera. Se não se encaixa na empresa, as decisões podem parecer erradas.
- **Nivel de paixão**
  Por definição, envolvimento em projetos open source requerem um nível mínimo de paixão. Se não,

porque outro motivo a pessoa perderia tempo na frente de um monitor? O tamanho do envolvimento em movimentos open source mostra quanto um candidato realmente se importa com programação.

- **Porcentagem de finalização**
Toda a inteligência, toda a cultura e paixão não se transformam em software de valor se o candidato não consegue terminá-lo. Infelizmente, muitos programadores não terminam seus projetos. Então, procure a exceção. Contrate aquele que consegue sair pela porta e está disposto a fazer as trocas pragmáticas que o trabalho exige.

- **Lado social**
Trabalhar com alguém por um bom período de tempo, durante tanto as horas de stress e descontração e altos e baixos vão mostrar a verdadeira personalidade do candidato. Se alguém não tem modos ou um lado sociável, deixe-os de lado.

Quando estamos falando de programadores, somente contratamos pessoas que nós conhecemos através do open source. Nós acreditamos que se adotarmos qualquer outro método, estamos sendo irresponsáveis. Contratamos Jamis porque nós gostamos de seus releases e participação na comunidade Ruby. Ele se superou em todas as áreas que acima. Não precisamos verificar mais nada, já que nós pudemos julgá-lo com base no que realmente importa: qualidade do seu trabalho.

E não se preocupe se as atividades extra-curriculares roubarem o foco e paixão do trabalho diário dos funcionários. Como diz aquele velho ditado: se quer algo feito, peça à pessoa mais ocupada que você conhece. Jamis e David são dois dos maiores contribuidores do Rails e ainda conseguem dirigir tecnicamente a 37signals. Pessoas que amam programar e terminar seus projetos são exatamente o tipo de pessoa que você quer em sua equipe.

Paixão open source

O que você mais quer de um novo funcionário é paixão pelo que ele faz, e não há jeito melhor de mostrar isso do que a trllha de comprometimento em projetos open source.

—Jarkko Laine, desenvolvedor de software
(de Reduza o risco, contrate de open source)

Procure Indivíduos Equilibrados

Procure por generalistas que aprendem rápido em vez dos especialistas limitados

Nunca contrataremos alguém que seja um arquiteto de informação. É simplesmente específico demais. Com uma equipe pequena como a nossa, não faz sentido contratar pessoas com um conjunto de conhecimento tão limitado.

Equipes pequenas precisam de pessoas que possam vestir diferentes chapéis. Precisamos de designers que saibam escrever. Precisamos de programadores que entendam de design. Todos devem ter noção de como arquitetar informação (seja lá o que isso signifique). Todos precisam ter mentes organizadas. Todos precisam saber se comunicar com clientes.

E todos precisar querer e serem capazes de diminuir a marcha pela estrada. Tenha em mente que equipes pequenas eventualmente precisam mudar de direção rapidamente. Queremos alguém que possa se ajustar, aprender e fluir ao contrário de um pé-na-lama que só consegue fazer uma coisa.

Você Não Pode Falsificar Entusiasmo

Vá com feliz e mediano em vez de frustrado e grande

Entusiasmo. É um atributo que simplemente não se pode falsificar. Quando chega a hora de contratar, não pense que precisa de um guru ou uma celebridade high-tech. Normalmente, de qualquer maneira são apenas divas. Um empregado mediano, mas feliz é melhor do que um expert que fica grunhindo.

Encontre alguem entusiasmado. Alguém em quem possa confiar para fazer as coisas quando deixado sozinho. Alguém que sofreu em uma empresa grande, devagar e deseja um novo ambiente. Alguém que está excitado para construir o que você está construindo. Alguém que odeia as mesmas coisas que você. Alguém que mal consegue esperar para subir a bordo do seu trem.

Pontos extras por fazer perguntas

Observe se um candidato em potencial faz muitas pergumtas sobre seu projeto. Programadores apaixonados querem entender um problema tão bem quanto possível e rapidamente irão propor soluções potenciais e melhorias, o que leva a muitas perguntas. Perguntas esclarecedoras também revelam um entendimento de que seu projeto poderia ser implementado de milhares de maneiras diferentes e é essencial detalhar o mais explicitamente quanto for possível como você imagina sua aplicação web funcionando. À medida que vai cavando nos detalhes, você desenvolverá um senso se a pessoa é bem aculturada.

—Eric Stephens, BuildV1.com

Artesãos de Palavras

Contrate bons escritores

Se está tentando decidir entre poucas pessoas para preencher uma posição, sempre contrate o melhor escritor. Não importa se essa pessoa é um designer, programador, marketing, vendedor ou o que for, essa habilidade leva a escrever mais efetivamente e concisamente código, design, emails, mensagens instantâneas e mais.

Isso porque ser um bom escritor é mais do que apenas palavras. Bons escritores sabem como se comunicar. Eles tornam as coisas mais fáceis de entender. Eles podem se colocar no lugar dos outros. Eles sabem o que omitir. Eles pensam claramente. E essas são as qualidades que você precisa.

Uma Mente Organizada

Boas habilidades de escrita são um indicador de uma mente organizada que é capaz de arranjar informação e argumentos de uma maneira sistemática e também ajudar (não fazer) outras pessoas a entender as coisas. Isso aparece no código, comunicação pessoal, mensagens instantâneas (para aqueles colaboradores de longa distância) e até esses conceitos exotéricos como profissionalismo e confiança.

—Dustin J. Mitchell, developer (de Signal vs. Noise)

Escrita Clara leva a Pensamento

Escrita clara leva a pensamento claro. Você não sabe o que sabe até tentar expressar esse conhecimento. Boa escrita é em parte uma questão de caráter. Em vez de fazer o que é fácil para você, faça o que é mais fácil para seu leitor.

—Michael A. Covington, professor de ciências da computação da Universidade da Geórgia
(de Como Escrever mais Claramente, Pensar mais Claramente e aprender Material Complexo mais Facilmente)

Design de Interface capítulo 9

Primeiro a Interface

Desenhe a interface antes de começar a programar

Muitos aplicativos começam com a mentalidade de programar primeiro. Isso é uma má idéia. Programação é o componente mais pesado de construir em um aplicativo, significando ser o mais caro e mais difícil de mudar. Ao invés disso, comece desenhando primeiro.

Design é relativamente leve. Um esboço de papel é barato e fácil de mudar. Rascunhos HTML são relativamente simples de modificar (ou jogar fora). Isso não é verdade na programação. Desenhar antes deixa você flexível. Programar primeiro prende você e gera custos adicionais.

Outra razão para projetar primeiro é que a interface é o seu produto. O que as pessoas vêem é o que você está vendendo. Se você somente rabiscar uma interface no final, os buracos vão aparecer.

Nós começamos pela interface para que possamos ver como o aplicativo será desde o começo. Este será constantemente revisado no decorrer do processo. Isso faz sentido? É fácil de usar? Ele resolve um problema de imediato? Existem perguntas que você só poderá realmente responder quando você lidar com telas reais. Desenhar antes deixa você flexível e o leva para essas respostas no processo mais cedo do que mais tarde.

A Caneta Laranja que Iniciou Blinksale

Assim que eu me toquei da minha frustação com o software de cobranças de prateleira, eu decidi desenhar como eu gostaria que minha solução de cobrança funcionasse. Eu retirei uma caneta laranja, porque era a única em mãos naquela noite e tive aproximadamente 75 por cento da UI desenhada em algumas horas. Eu mostrei para minha esposa, Rachel, que estava passando roupa no momento, e perguntei, “O que você acha?” E ela respondeu com um sorriso, “Você precisa fazer isso. Pra valer.”

Nas duas semanas seguintes eu refinei o design, e criei quase todos os protótipos HTML estáticos para a primeira versão do que se tornaria Blinksale. Nós nunca fizemos wireframes além daqueles rabiscos de caneta laranja, e chegando direto no design HTML nos ajudou a ficar excitados sobre o quão/como “real” o projeto estava tornando, mesmo que nós realmente não sabíamos o que estávamos começando.

Uma vez que os protótipos do HTML foram terminados, nós apresentamos ao nosso desenvolvedor Scott, a idéia para Blinksale. Ter a maioria da Interface de Usuário (UI, User Interface em inglês) projetada anteriormente foi extremamente benéfico em diversos níveis. Primeiramente, demos a Scott uma visão real e um clareamento para onde nós iríamos. Era muito mais do que apenas uma idéia, ele era real. Em seguida, ele nos ajudou a medir exatamente quanto do esforço e tempo seria necessário para tornar o design/projeto em uma aplicação funcionando. Quando você está amarrado financeiramente um projeto, quanto mais cedo você pode predizer exigências do orçamento melhor. O projeto de UI tournou-se nossa

marca de nível para o escopo inicial do projeto. Finalmente, o projeto do UI serviu como um guia para nos lembrar sobre o que a aplicação era quando progredíamos mais no desenvolvimento. Enquanto nos era tentador adicionar características novas, nós não poderíamos simplesmente dizer, “Certo, vamos adicionar isso!” Tivemos que voltar para trás ao projeto e perguntar a nós mesmos onde essa característica nova iria, e se não tivesse um lugar, não seria adicionada.

—Josh Williams, fundador, Blinksale

Design de Epicentro

Comece do núcleo da página e construa para fora

Design de epicentro foca na verdadeira essência da página — o epicentro — e então constrói para fora. Isso significa que, no começo, você ignora as extremidades: a navegação/menus, rodapé, cores, barra lateral, logotipo, etc. Em vez disso, você começa o epicentro e faz o design das peças de conteúdo mais importantes primeiro.

Seja qual for a página ela não pode viver sem seu epicentro. Por exemplo, se estiver fazendo o design de uma página que mostra a publicação de um blog, a publicação por si é o epicentro. Não as categorias na barra lateral, não o cabeçalho no topo, não o formulário de comentários embaixo, mas a unidade de publicação de mensagem do blog. Sem essa unidade de publicação, a página não é a publicação de um blog.

Somente quando essa unidade está completa você começaria a pensar no segundo elemento mais crítico da página. Então, depois desse segundo elemento mais crítico, se moveria para o terceiro, e assim por diante. Isso é design de epicentro.

Design de epicentro evita o tradicional modelo "vamos construir a moldura então jogar o conteúdo dentro". Nesse processo, o formato da página é construída, então a navegação é incluída, então as "coisas" de marketing são inseridas e então, finalmente, o núcleo da funcionalidade, o verdadeiro propósito da página, é enfiado em um espaço qualquer que tenha sobrado. É um processo de trás para frente que tira o que deveria ser a prioridade principal e deixa isso para o fim.

Design de Epicentro vira esse processo e permite que você foque no que realmente interessa no dia um. Essenciais primeiro, extras em segundo. O resultado é uma tela mais amigável, focada e usável para os clientes. Além disso, permite que você comece o diálogo entre designer e desenvolvedor logo de cara em vez de esperar por todos os aspectos da página caírem na linha primeiro.

Solução de Três Estados

Faça design para os estados regular, branco e erro

Para cada tela, você precisa considerar três estados possíveis:

- **Regular**
  A tela que as pessoas vêem quando tudo está funcionando bem e sua aplicação é preenchida com dados.
- **Branco/Vazio**

A tela que as pessoas vêem quando estão usando a aplicação pela primeira vez, antes de dados serem inseridos.

- **Erro**
  A tela que as pessoas vêem quando alguma coisa dá errado.

O estado regular é trivial. É a tela onde você vai gastar a maior parte do tempo. Mas não se esqueça de investir tempo nos outros estados também (veja os artigos seguintes para mais sobre isso).

A Tela em Branco

Supere as expectativas com uma primeira experiência convincente

Ignorar o estado de superfície branca é um dos maiores erros que você pode cometer. O estado branco é a primeira impressão de sua aplicação e você nunca terá uma segunda ... bem, você sabe.

O problema é que quando se faz o design da interface de usuário, normalmente ela está preenchida de dados. Designers sempre preenchem os desenhos com dados. Cada lista, cada publicação, cada campo, cada canto e espacinho tem coisa dentro. E isso significa que a tela parece pronta e funciona muito bem.

Entretanto, o estado natural da aplicação é vazia de dados. Quando alguém se cadastra, eles começam com uma tela em branco. Muito parecido com um weblog, cabe a eles popularem — o visual geral não toma forma até as pessoas colocarem seus dados: publicações, links, comentários, horas, informação de barra lateral ou o que for.

Infelizmente, o cliente decide se a aplicação vale a pena pelo estado inicial de sua tela em branco — o estado onde há a menor quantidade de informação, design e conteúdo sobre o qual julgar a utilidade geral da aplicação. Quando você falha no design adequado deste estado em branco, as pessoas não sabem o que estão perdendo porque tudo está faltando.

Mesmo assim a maioria dos designers e desenvolvedores ainda subestima esse estado. Eles falham em gastar um bom tempo no design de telas vazias porque quando eles desenvolvem/usam a aplicação, esta já está cheia de dados que eles colocaram para propósitos de testes. Eles nem mesmo encontram a tela em branco. Claro, eles podem fazer o login como uma nova pessoa algumas vezes, mas a maioria das vezes é gastam nadando em uma aplicação que está cheia de dados.

O que você deve incluir em uma tela em branco que ajuda?

- Use como uma oportunidade de inserir pequenos tutoriais e caixas de ajuda.
- Dê uma foto da tela final como exemplo da página populada com dados para que as pessoas saibam o que esperar (e porque devem ficar por lá).
- Explique como começar, como a tela vai ficar exatamente, etc.
- Responda as perguntas-chave que visitantes de primeira viagem fazem: O que é esta página? O que faço agora? Como essa tela vai ficar quando estiver cheia?
- Supere as expectativas e ajude a reduzir frustrações, intimidações e a confusão em geral.

Primeiras impressões são cruciais. Se você falhar em fazer o design de uma tela em branco bem pensada, criará impressão negativa (e falsa) da sua aplicação ou serviço.

Você Nunca Ganha uma Segunda Chance ...

Outro aspecto da interface do Mac OS X que acho que foi tremendamente influenciado por [Steve] Jobs é a configuração e primeira experiência. Acho que Jobs sabe muito bem da importância das primeiras impressões ... Acho que Jobs olha para a primeira experiência e pensa, pode ser apenas um milionésimo da experiência geral de um usuário com a máquina, mas é o milionésimo mais importante, porque é o primeiro milionésimo, e isso supera suas expectativas e impressões iniciais.

—*John Gruber, autor e desenvolvedor web (de Entrevista com John Gruber)*

Torne-se Defensivo

Faça Design para quando as coisas derem errado

Vamos admitir: As coisa vão dar errado online. Não importa o quão cuidadoso você faça o design de sua aplicação, não importa quanto teste fizer, os clientes ainda vão encontrar problemas. Então como você gerencia essas quedas inevitáveis? Com design defensivo.

Design defensivo é como direção defensiva. Da mesma maneira como motoristas devem estar sempre atentos para estradas escorregadias, motoristas imprudentes e outros cenários perigosos, construtores de sites devem procurar constantemente por pontos de problema que causem confusão e frustração aos visitantes. Um bom site defensivo por criar ou quebrar a experiência do cliente.

Poderíamos encher um livro separado com todas as coisas que temos a dizer sobre design defensivo. De fato, já fizemos. "Design Defensivo para a Web" é o título e é um grande recurso para qualquer um que queira aprender como melhorar telas de erros e outros pontos críticos.

Lembre-se: Sua aplicação pode funcionar muito bem 90% do tempo. Mas se você abandonar seus clientes no momento em que mais precisam, é improvável que eles se esqueçam disso.

Contexto Sobre Consistência

O que faz sentido aqui pode não fazer sentido alí

As ações devem ser botões ou links? Depende da ação. O calendário deve ser visto em forma de lista ou em grade? Depende de onde ele será visto e quão longo é o período exibido. Todo link de navegação global deve estar em todas as páginas? Você precisa mesmo de uma caixa de busca em todo lugar? Você precisa do mesmo rodapé em cada página? A resposta: “Depende”.

Isto porque contexto é mais importante que consistência. Tudo bem ser inconsistente se o seu design faz mais sentido dessa maneira. Forneça às pessoas apenas o que importa. Ofereça a eles o que eles precisam e livre-se de tudo o que não for necessário. É melhor ser correto do que ser consistente.

Inconsistência Inteligente

Consistência não é necessária. Por muitos anos, estudantes de Design foram ensiados que consistência na interface é uma das regras-chave no design. Talvez isso sirva pra software, mas na Web, não é verdade. O que importa na Web é que, em cada página, os usuários possam fácil e rapidamente avançar para o próximo passo no processo.

Na Creative Good, nós chamamos isso de “inconsistência inteligente”: certeza de que cada página no processo dá ao usuário precisamente o que eles precisam naquele ponto do processo. Adicionar elementos navigacionais supérfluos, só porque são consistentes com o restante do site, é pura bobagem.

—Mark Hurst, fundador da Creative Good e criador da Goovite.com
(de O Paradigma da Página)

Direitos Autorais é Design de Interface

Cada palavra importa

Direitos Autoriais é Design de Interface. Grandes interfaces são escritas. Se você pensar que cada pixel, cada ícone, cada fonte importa, então precisa acreditar que cada palavra importa. Quando está escrevendo sua interface, coloque-se sempre no lugar da pessoa que está lendo sua interface. O que eles precisam saber? Como você pode explicar suscintamente e claramente?

Você etiqueta um botão como Submeter ou Salver ou Atualizar ou Novo ou Criar? Isso é Direito Autoral. Você escreve três sentenças ou cinco? Explica com exemplos gerais ou com detalhes? Etiqueta seu conteúdo como Novo ou Atualizado ou Atualizado Recentemente ou Modificado? Será Existem novas mensagens: 5 ou Existem 5 novas mensagens ou é 5 ou cinco ou mensagens ou publicações? Tudo isso importa.

Você precisa falar a mesma linguagem que sua audiência também. Só porque está escrevendo uma aplicação web não significa que pode sair por aí com jargão técnico. Pense sobre seus clientes e pense sobre o que significam aqueles botões e palavras para eles. Não use acrônimos ou palavras que a maioria das pessoas não entende. Não use linguagem interna. Não pareça como um engenheiro falando com outro engenheiro. Mantenha curto e doce. Diga o que precisa e não mais.

Boa escrita é bom design. É rara a exceção de palavras não acompanharem o design. Ícones com nomes, campos de formulários com exemplos, botões com etiquetas, instruções passo a passo em um processo, uma explicação clara das suas políticas de reembolso. Tudo isso é design de interface.

Uma Interface

Incorpore funções administrativas em interfaces públicas

Telas administrativas — as telas usadas para gerenciar preferências, pessoas, etc. — tem uma tendência de parecer um lixo. Isso porque a maioria do tempo de desenvolvimento é gasto na aparência pública das interfaces.

Para evitar a síndrome da tela-administrativa-lixo, não construa telas separadas para lidar com funções administrativas. Em vez disso, construa essas funções (isto é, editar, adicionar, deletar) na interface regular da aplicação.

Se tiver que manter duas interfaces separadas (isto é, uma para o pessoal regular outra para administradores), ambas vão sofrer. Em efeito, você acaba pagando a mesma taxa duas vezes. Você é forçado a se repetir e isso significa que você aumenta as chances de ficar desleixado. Quanto menos telas tiver, menos terá para se preocupar e melhor as coisas saem.

Sem Interface Separada

A aplicação é tudo. Qualquer coisa pode ser modificada, adicionada ou ajustada diretamente através da área de gerenciamento da aplicação. Isso nos permite ver exatamente o que nossos clientes vêem para ajudá-los através de qualquer problema ou questões que tiverem. E nossos clientes não precisam se preocupar em fazer login em uma interface separada para fazer tarefas diferentes. Em um minuto podem estar lidando com compromissos para seus clientes e no próximo minuto poderiam estar adicionando um novo empregado. Eles não podem ser incomodados pulando entre aplicações diferentes, e mantendo a interface consistente eles são capazes de se adaptar à aplicação ainda mais rápido.

—Edward Knittel, Diretor de Vendas e Marketing , KennelSource

---

Código capítulo 10

Menos Software

Mantenha seu código o mais simples possível

É bastante razoável pensar que um software com o dobro de código seria apenas duas vezes mais complexo. Mas na verdade, **à medida que se aumenta a quantidade de código, a complexidade do software tende a crescer exponencialmente**. Cada pequena adição, cada nova interdependência, cada nova preferência amplia o efeito cascata. Adicione código desenfreadamente à sua aplicação, e antes que você perceba, você terá criado uma grande e desgovernada bola de neve.

A melhor maneira de se enfrentar a complexidade é com menos código. Menos software significa menos funcionalidades, menos código, menos desperdício.

A chave está em repensar qualquer problema difícil que venha a necessitar de uma grande quantidade de componentes para ser solucionado em um problema mais fácil, que requeira muito menos. Você pode acabar não solucionando exatamente o mesmo problema, mas tudo bem. Resolver 80% do problema original despendendo 20% do esforço é uma vitória e tanto. O problema original raramente é tão crítico de forma a realmente merecer cinco vezes mais esforço em sua solução.

Menos software é a melhor maneira para aposentar a sua bola de cristal. Em vez de tentar prever problemas futuros, lide apenas com os problemas de hoje. Por quê? A maioria dos medos que você tem a respeito do futuro raramente tornam-se reais. Não perca seu tempo tentando solucionar estes problemas-fantasma.

Desde o início, desenvolvemos nossos produtos ao redor do conceito de pouco software. Sempre que possível, simplificamos os problemas mais difíceis. E descobrimos que a solução para problemas mais simples não é somente mais fácil de implementar e suportar, como também de entender e usar. É tudo parte de uma estratégia para diferenciar-se dos competidores: em vez de focar-se em produtos que fazem mais, construimos produtos que fazem menos.

- Menos software é mais fácil de se gerenciar.
- Menos software reduz a quantidade de código e isso significa
- menor carga de trabalho de manutenção (e uma equipe mais feliz).
- Menos software reduz os custos de mudança, de forma que você pode adaptar-se rapidamente. Você pode mudar de idéia sem ter que mudar milhões de linhas de código.
- Menos software resulta em menos bugs.

- Menos software significa menos suporte.

A escolha de quais funcionalidades incluir ou omitir também tem muito a ver com a quantidade de software. Não tenha medo de dizer não a solicitações de funcionalidades difíceis de se implementar. A menos que elas sejam absolutamente essenciais, economize tempo, esforço e muita confusão deixando-as de fora.

Vá devagar, também. Quando surgir uma nova idéia, não tome nenhuma ação por uma semana, e ao final veja se a idéia ainda parece tão brilhante. O tempo extra em “banho maria” geralmente ajudará seu cérebro a pensar em uma solução mais simples.

Encoraje programadores a pensar em contra-propostas
O que se deseja ouvir é: “A maneira como você sugeriu levará 12 horas para ser implementada. Mas há um jeito de fazer que vai levar só uma hora. Não vai fazer x, mas vai fazer y.”. **Deixe o software dizer "não".**. Encoraje os programadores a lutarem pelo que eles pensam ser a melhor maneira.

Também busque por alternativas a ter que escrever mais software. Seria possível mudar um fluxo de telas de modo que elas sugiram uma rota alternativa para os usuários que não requeira mudanças no modelo do software? Por exemplo, seria possível sugerir que as pessoas façam upload de imagens de um tamanho específico, em vez de ter que manipular as imagens no lado do servidor?

Para cada nova funcionalidade de sua aplicação, pergunte-se se não existe uma maneira de se produzir o mesmo resultado e que não requeira tanto software. Escreva apenas o código que precise, e nada mais. Sua aplicação acabará bem mais magra e saudável.

NÃO existe código mais flexível do que NENHUM código!

O “segredo” do bom projeto de software não estava em saber o que codificar; estava em saber o que NÃO codificar. Estava em saber o que deixar de fora! Estava em perceber quais os pontos que necessitariam de mais flexibilidade e quais não, e então deixar espaço para futuras mudanças, em vez de tentar expandir mais e mais o design.

—Brad Appleton, engenheiro de software
(de There is No CODE that is more flexible than NO Code!)

Complexidade não aumenta linearmente com o tamanho

A regra mais importante da engenharia de software é também a menos conhecida: a complexidade não cresce linearmente com o tamanho… Um programa de 2000 linhas requer mais do dobro do esforço de desenvolvimento que um programa com a metade do seu tamanho.

—The Ganssle Group (de Keep It Small)

Otimize para Felicidade

Escolha ferramentas que estimulem e motive o seu time

Um programador feliz é um programador produtivo. É por isso que nós otimizamos para felicidade e você deveria fazer o mesmo. Não escolha as ferramentas e práticas baseado simplesmente no padrão do mercado ou métricas de desempenho. Avalie os atributos intangiveis: a ferramenta foi criada com paixão, orgulho e dedicação?. Você seria feliz trabalhando neste ambiente oito horas por dia?

Isto é especialmente importante quando você estiver escolhendo uma linguagem de programação. Apesar

da percepção do público sobre o conteário, elas não são criadas iguais. Enquanto qualquer linguagem pode produzir qualquer tipo de aplicação, as melhores para o caso não seriam somente possíveis ou aceitáveis, elas seriam prazeirosas e revigorantes. É simplesmente tornar os pequenos detalhes do trabalho diário mais divertidos.

A felicidade tem um efeito em cascata. Programadores felizes trabalham da maneira correta. Eles escrevem códigos simples e de fácil leitura. Eles abordam o problema de uma maneira elegante, expressiva e de fácil entendimento. Eles se divertem.

Nós encontramos o ecstasy da programação na linguagem Ruby e o passamos adiante através do nosso framework, Rails. Ambos compartilham do mesmo objetivo de otimizar para humanos e sua felicidade. Nós o aconselhamos a dar uma chance a essa combinação.

Resumindo, sua equipe necessita trabalhar com ferramentas de que eles gostem. Nós citamos exemplos no contexto de linguagens de programação, mas o conceito se aplica à aplicações, plataformas, e praticamente a tudo. Escolha a fusão que deixa as pessoas excitadas. Você vai criar mais motivação e excitação e consequentemente um melhor resultado.

Os tipos de engenheiros que você quer

O motivo número um de eu ter escolhido Ruby on Rails para criar nossa aplicação é a sua elegância, produtividade e a beleza de sua arquitetura. Ele tem a tendência de atrair o tipo de engenheiros que se preocupam com esse tipo de coisa… exatamente o tipo de engenheiros que você quer no seu time, porque eles criam softwares atrativos, elegantes e produtivos que você precisa para ganhar o mercado.

—Charles Jolley, Diretor gerencial da Nisus Software (de Signal vs. Noise)

O Código Fala

Ouça quando seu código diz "não"

Ouça seu código. Ele oferecerá sugestões. Ele irá dizer "não". Ele lhe dirá onde ficam as armadilhas. Ele irá sugerir novas maneiras de fazer as coisas. Ele irá ajudá-lo a se manter em um modelo de menos software.

Uma nova funcionalidade está requerendo semanas de tempo e milhares de linhas de código? Isso é seu código lhe dizendo que provavemente existe uma maneira melhor. Existe uma maneira simples de codificar alguma coisa em uma hora em vez de uma maneira complicada que consumirá dez horas? Novamente, esse é seu código o guiando. Ouça.

Seu código pode guiá-lo a consertos que são baratos e leves. Preste atenção quando um caminho mais fácil emerge. Claro, a funcionalidade que é fácil de fazer pode não ser exatamente a mesma que você originalmente tinha em mente, mas e daí? Se funciona bem o suficiente e lhe dá mais tempo para trabalhar em outra coisa, é um ganhador.

Ouça

Não se preocupe com o design, se ouvir seu código um bom design vai aparecer ... Ouça as pessoas técnicas. Se eles estão reclamando sobre a dificuldade de fazer mudanças, então leve essas reclamações a sério e lhes dê tempo para consertar as coisas.

—Martin Fowler, Cientista Chefe, ThoughtWorks (de Is Design Dead?)

Se Programadores Fossem Pagos para Remover Código ...

Se programadores fossem pagos para remover código do software em vez de escrever novo código, seria muito melhor.

—Nicholas Negroponte, Professor de Tecnologia de Mídia no MIT
(de And, the rest of the (AIGA Conference) story)

Gerencie Débitos

Pague a seu código e as "contas" de design

Normalmente pensamos em débito na forma de dinheiro mas ele vem em outras formas também. Você pode facilmente construir código e débito de design.

Grude junto alguns códigos ruins que são funcionais mas ainda assim um pouco cabeludos e estará construindo débito. Jogue junto um design que é bom o suficiente mas não realmente bom e estará se endividando novamente.

Não tem problema fazer isso de vez em quando. De fato, às vezes é uma técnica necessária que o ajuda a chegar mais rápido ao esquema Caia-Na-Real-RÁPIDO. Mas você ainda precisa reconhecê-lo como débito e pagá-lo em algum ponto limpando o código cabeludo ou redesenhando aquela página mais ou menos.

Da mesma forma que você deve regularmente colocar de lado uma parte do seu salário para impostos, regularmente coloque uma parte do seu tempo para pagar seu código e débito de design. Se não fizer isso, apenas estará pagando juros (consertando grudes) em vez de pagar o montante (e movendo-o adiante).

Abra as Portas

Publique dados para o mundo via RSS, APIs, etc.

Não tente prender seus usuários. Deixe que eles possam ter acesso a suas informações quando quiserem, da forma que preferirem. Para tal, você precisa deixar de lado a idéia de manter os dados de seus usuários trancados a sete chaves. Em vez disso, deixe que a informação flua. Garanta o acesso à informação através de feeds RSS. Ofereça APIs que permitam a terceiros construir aplicações integradas à sua. Tais atitudes tornarão a vida dos usuários mais conveniente e expandirão as possibilidades do que sua aplicação é capaz de fazer.

No passado, as pessoas acostumaram-se a pensar nos feeds RSS apenas como uma boa maneira de se agregar conteúdo de sites de blogs e sites de notícia. Contudo, os feeds são mais poderosos que isto. Eles também podem permitir ao usuário manter-se atualizado sobre mudanças internas à aplicação sem a necessidade de logar-se repetidas vezes. Através do site do Basecamp, por exemplo, o usuário pode cadastrar sua url em um agregador de RSS e assim receber notificações de mensagens de projetos, listas de tarefas e objetivos sem a necessidade de conectar-se constantemente ao site em busca de informações atualizadas.

APIs permitem que desenvolvedores construam plugins adicionais à sua aplicação, que geralmente agregam valor ao seu produto. Por exemplo, a API disponibilizada pelo Backpack foi utilizada pela Chipt Productions na construção de um widget para o Mac os X. A pequena aplicação permite aos usuários

adicionar e editar lembretes, listagens de items e muito mais a partir de seus desktops. Muitos usuários apontaram o widget como uma ótima ferramenta, e alguns mesmo apontaram-no como um fator decisivo na escolha da utilização do Backpack.

Outros bons exemplos de empresas que liberaram dados como uma maneira de conseguir um 'efeito bumerangue':

- A API do **Google Maps** permitiu o surgimento de toda sorte de pequenas aplicações que recuperam dados de outras fontes (ex.: uma listagem de apartamentos) e os exibem em um mapa.
- Linkrolls oferece aos usuários exibir seus últimos bookmarks do **del.icio.us** em seu próprio site.
- O **Flickr** permite que outros negócios acessem as suas APIs comerciais, de forma a permitir aos usuários comprar livros de fotos, posters, backups em DVD e selos. "O objetivo é manter as portas completamente abertas e permitir o maior número possível de possibilidades de utilização de suas fotos", diz Stewart Butterfield, do Flickr.

Um Widget Faz a Diferença

Quando a 37signals lançou o Backpack, há algum tempo atrás, minha primeira impressão foi… er... bem...

Ocorreu mais ou menos na época em que a Chipt Productions lançava um widget Backpack para o Sistema Operacional Tiger — que parecia interessante demais para passar despercebido — com isso dei uma segunda olhada no Backpack. O resultado? Uma grande diferença.

Hoje, sempre que uma nova idéia surge, abro o widget, digito e salvo — e pronto. Recebo algum e-mail com algo que devo fazer? Abro o widget, digito e salvo — e pronto. O widget tornou-se um tipo de bloco de notas indispensável, que instalo em todo Mac que uso. E por se tratar de uma aplicação totalmente web, não há necessidade de nenhum tipo de controle de versão ou sincronizaçao de dados — apenas a fluidez de digitar-se dados sem ter que se preocupar em saber para onde os dados foram, nem como acessá-los mais tarde.

—Todd Dominey, fundador, Dominey Design
(de Trying on Backpack)

Palavras capítulo 11

Não Há Nada de Funcional em uma Especificação Funcional

Não escreva um documento de especificações funcionais

Estas plantas baixas de projeto geralmente acabam por descrever algo completamente diferente do produto final. Aí vai o motivo:

Especificações funcionais são fantasias

Especificações não refletem a realidade. Uma aplicação não é real enquanto seus desenvolvedores não começarem a construção; seus designers começarem a traçar a desenhá-la; os usuários começarem a testa-la e experimentar o produto. Especificações funcionais são apenas palavras em papel

Especificações funcionais são apenas uma forma de acalmar os ânimos.

Elas servem para fazer com que todos sintam-se envolvidos e felizes com o projeto – e embora a sensação de segurança seja reconfortante, ela não traz nenhum benefício ao desenvolvimento. Estas especificações nunca tocam nos pontos mais críticos do projeto ou mesmo avaliam seus custos, fatores que nunca devem ser esquecidos na construção de uma grande aplicação.

Especificações funcionais apenas levam à ilusão de um acordo

O fato de um punhado de pessoas concordarem sobre alguns parágrafos de texto não implica em dizer que todos chegaram a um entendimento comum: todos podem estar lendo o mesmo texto, mas chegando a conclusões completamente diferentes. Estas diferenças inevitavelmente aparecem no decorrer do projeto: “Espere, não era isso que eu tinha entendido.” “Hein? Não foi assim que nós descrevemos.” “Sim, foi isso que concordamos – você aprovou que fosse desse jeito!”. Você sabe a encrenca que é.

Especificações funcionais forçam a tomada das decisões mais importantes justamente quando se tem o mínimo de informações sobre o todo.

É normal saber-se pouco sobre qualquer coisa antes de começar a construção. Quanto mais se avança no projeto, quanto mais se usa o produto, mais se entende sobre ele. É neste ponto em que as decisões deveriam ser feitas – quando se tem mais informação, não menos.

Especificações funcionais geram excesso de funcionalidades

Não há impeditivos na fase de especificação. Não há custo nenhum em adicionar mais um tópico a uma lista de requisitos. Você pode agradar aos críticos mais chatos do projeto adicionando à especificação aquela “funcionalidade de estimação” que eles tanto gostariam de ver implementada. E no fim das contas, a equipe acabará desenvolvendo uma aplicação que satisfará uma lista de requisitos em uma folha de papel – não seres humanos. E assim você acabará com um site sobrecarregado, com um milhão de abas, menus e opções espalhadas por uma página indecifrável.

Especificações funcionais não deixarão que você evolua, mude ou rearranje

Cada funcionalidade é acordada e aprovada. Mesmo que você perceba durante o desenvolvimento que esta funcionalidade é uma má ideia, não há mais nada que possa ser feito. Especificações não ajudarão a lidar com a realidade quando o desenvolvimento começar, e tudo mudar de uma hora para a outra.

Então o que deve ser feito em vez de uma especificação funcional? Comece por uma alternativa mais breve, que possa transformar-se mais rapidamente em algo real. Escreva uma página descrevendo o que a aplicação precisa fazer. Use linguagem coloquial e faça isso rápido. Se você precisar de mais de uma página para explicar o conceito, então ele é provavelmente muito complexo. Este processo de “especificação” não deve tomar mais que um dia.

Comece então a construção da interface – ela será o substituto da sua especificação funcional. Desenhe alguns esboços rápidos em papel, então comece a transformar o esboço em código HTML. Diferentemente de parágrafos de texto abertos a interpretação, a interface da aplicação é um corpo comum, que tenta representar ao máximo a versão desejada da aplicação – sem interpretações subjetivas.

A confusão tende a desaparecer quando todos começam a usar as mesmas telas. Construa uma interface que permita que todos naveguem, usem e “sintam” a aplicação antes mesmo de começar a se preocupar com o código de back-end. Coloque-se na pele do usuário o máximo possível.

Esqueça as especificações congeladas e imutáveis. Elas forçam a tomada de decisões importantes muito

cedo no processo. Pule a etapa de especificação funcional e mantenha o custo das mudanças em baixa e a flexibilidade em alta.

Especificações inúteis

Uma “especificação” é um documento quase que completamente inútil. Eu nunca vi uma especificação detalhada o suficiente para que seja útil e precisa ao mesmo tempo.

E eu já vi muito lixo construído com base em especificações. Desenvolver com base em especificações é a pior maneira de se escrever software, pois por definição, trata-se de programar para satisfazer uma teoria, não a realidade.

—Linus Torvalds, Criador do Linux (from: Linux: Linus Sobre Especificações)

Enfrente os “atrasadores de projeto”

Eu cheguei à conclusão de que muitas das pessoas que insistiam em uma lista extensiva de requisitos antes de começar qualquer design tratavam-se de meros “atrasadores” tentando frear o processo (e que geralmente estas pessoas não tinham nada a contribuir no design, nem qualquer idéia inovadora para compartilhar).

Todo nosso melhor trabalho foi feito com alguns conceitos na cabeça sobre melhorar o site, alguns protótipos rápidos (estáticos), pequenas alterações no design e, enfim, com a construção de um protótipo funcional com dados reais. Após nos prepararmos com esse protótipo, geralmente tínhamos um projeto real em curso e um bom resultado.

—Mark Gallagher, desenvolvedor de intranets corporativas (de Signal vs. Noise)

Não Faça Documentos Mortos

Elimine a papelada desnecessária

Evitar especificações funcionais é um bom começo, mas não é tudo: é necessário evitar o excesso de papelada em todo o projeto. A menos que um documento vá efetivamente transformar-se em algo real, ele não deve ser escrito.

Construa, não escreva. Se você precisar explicar alguma coisa, tente construir um protótipo em vez de redigir um longo documento. Uma interface de verdade ou um protótipo tende a seguir caminho em direção a um produto real. Um punhado de folhas de papel, por sua vez, somente seguirá caminho rumo a uma lixeira.

Se um documento provavelmente nunca evoluirá para um design real, não se preocupe em escrevê-lo. Se o intuito de um documento é transformar-lo em um modelo, vá em frente.

Documentos que existem separadamente da aplicação são inúteis. Eles não levarão a lugar nenhum. Todos os esforços de projeto devem ser usados na evolução do produto real. Se um documento é congelado antes de se tornar uma peça real, ele está morto.

Ninguém nunca irá lê-lo

Eu sequer consigo lembrar quantas especificações ou documentos de requerimentos de negócio ficaram de escanteio, juntando poeira enquanto minha equipe de desenvolvimento codificava, discutia problemas, fazia

perguntas e conduzia testes de usabilidade ao longo de nossos projetos. Também já trabalhei com desenvolvedores que desperdiçaram horas escrevendo longos e minuciosos emails ou documentos de padrões de codificação que sempre acabaram não sendo lidos por ninguém.

Aplicações web não avançam graças a um grande punhado de documentos. O desenvolvimento de software é um processo em constante evolução e que envolve iterações e decisões rápidas, à medida que problemas imprevisíveis aparecem pelo caminho. Nada disso pode ou deveria ser registrado em folhas e folhas de papel.

Não desperdiçe seu tempo escrevendo aquele longo e visionário documento: ninguém irá lê-lo. Console-se com o fato de que, se o seu produto tiver espaço o suficiente para crescer adequadamente, no final ele nem de longe parecerá com qualquer coisa que você tenha escrito sobre ele.

—Gina Trapani, desenvolvedora web e editora do Lifehacker, o guia de produtividade e software

---

Me Conte uma História Rápida

Escreva histórias, não detalhes

Sempre que faltarem palavras para explicar uma nova funcionalidade ou conceito, deve-se escrever uma pequena história sobre a idéia. Sem entrar em detalhes técnicos ou de design, a história deve ser escrita para humanos – como que em um diálogo qualquer com outras pessoas.

A história tão pouco precisa ser um ensaio elaborado. Um punhado de orientações sobre o fluxo das coisas geralmente é suficiente. Ainda melhor se for possível contextualizar a história com um punhado de projetos de telas.

Ao se expressar novos conceitos através de histórias, deve-se pensar na experiência, em vez de perder-se em detalhes. Focar na estratégia, não na tática. As táticas aparecerão uma vez que a aplicação comece a ser construída. Até aqui, tudo que se deseja é uma história capaz de iniciar uma discussão, e colocar o projeto no trilho certo.

Use Palavras de Verdade

Use texto real em vez de lorem ipsum

O clássico Lorem ipsum dolor é um amigo fiel de muitos designers. Textos falsos, “de enchimento”, ajudam a ter uma idéia de como o design ficará, uma vez finalizado. Mas a utilização de textos de enchimento pode ser perigosa, também.

O lorem ipsum muda a forma como o texto é visualizado no todo. Ela reduz o conteúdo textual do site a um mero elemento visual – uma “forma de texto” – em vez do que ele realmente deveria ser: informações valiosas que deverão ser lidas e/ou digitadas. A utilização de textos de enchimento acaba por esconder as inevitáveis variações que aparecerão uma vez que informações reais sejam utilizadas. Ela dificulta a percepção de como o design realmente se comportará quando dados reais forem digitados. Textos de enchimento são um abismo entre o design e a realidade.

São precisos dados reais para que se possa definir o tamanho ou forma de certos campos. São precisos dados reais para perceber como tabelas irão se expandir ou contrair. São precisos dados reais para

visualizar a aplicação.

Palavras relevantes devem ser usadas o mais cedo possível. Se o site requerer entrada de dados, dados reais devem ser fornecidos. Mais que isso, os dados devem ser realmente digitados – não somente copiados e colados de outra fonte. Se o sistema solicitar um nome, utilize um nome real. Se solicitar uma cidade, utilize uma cidade real. Se solicitar a digitação de uma senha e sua confirmação, digite duas vezes.

Claro que é muito mais fácil percorrer todos os formulários e preencher os campos com lixo (“asdsadklja” “123usadfjasld” “snaxn2q9e7”), de forma a percorrê-los rapidamente. Mas estes dados não são reais. Não é isso que os clientes farão. Não é sábio testar o sistema através de um atalho, enquanto os usuários serão forçados a tomar o caminho mais longo. Quando apenas se digitam dados falsos com a velocidade de uma metralhadora, perde-se a percepção de como realmente se preenche tais formulários.

Fazer como os usuários fariam é uma maneira de entendê-los melhor. E uma vez que eles sejam entendidos, uma vez que se sinta o que os usuários sentem, sua equipe construirá uma interface melhor.

Lorem Ipsum Lixum

Quando o design deixa de levar em consideração como o conteúdo do site “deveria ser”, o impacto sobre o resultado final é grande. O significado das páginas se perde por ser “apenas texto”; o entendimento é comprometido por ninguém perceber que o tal texto deve estar ali supostamente para ser lido. Oportunidades são perdidas porque o texto lorem ipsum usado no lugar de texto real não sugere novas oportunidades ou funcionalidades. E se o texto é tão desnecessário e não está ali para ser usado, podemos então apenas substituí-lo por um adorável espaço em branco.

—Tom Smith, designer e desenvolvedor (de Eu Odeio Lorem Ipsum e Usuários Lorem Ipsum)

Dê Personalidade a Seu Produto

Qual o tipo de personalidade do seu produto?

Produtos devem ser pensados como se fossem pessoas. Que tipo de pessoa seu produto deve ser? Educada? Divertida? Séria? Relaxada? É melhor que ela seja lembrada como uma aplicação confiável ou excessivamente segura? As a know-it-all? Modesta? Simpática?

Uma vez decidida a personalidade da aplicação, tais características devem ser sempre lembradas durante a construção do produto. Elas devem ser usadas para guiar a construção da interface, a escolha das funcionalidades e até mesmo o texto de copyright. Sempre que qualquer mudança for feita à aplicação, deve-se pensar primeiro se tal mudança se adequa à personalidade da aplicação.

Cada produto tem uma voz – e ela fala com os clientes 24 horas por dia.

Precificação e Assinatura capítulo 12

Amostra Grátis

Dê alguma coisa de graça

É um mundo barulhento lá fora. Para que as pessoas o notem no meio da multidão, dê alguma coisa de graça.

Empresas espertas sabem que dar brindes é uma excelente maneira de fisgar clientes. Veja a Apple. Eles oferecem o software iTunes de graça de forma a gerar demanda para o iPod e a loja de música iTunes. No mundo offline, as lojas fazem a mesma coisa. A Starbucks diz que uma nova compra é estimulada para cada cinco amostras de bebidas que eles dão aos clientes. Nada mau.

Para nós, *Writeboard* e Ta-da list são aplicativos completamente grátis que usamos para colocar as pessoas no caminho para usar nossos outros produtos. Adicionalmente, sempre oferecemos algum tipo de versão grátis de todos os nossos aplicativos.

Queremos que as pessoas experimentem o produto, a interface, a utilidade do que construímos. Uma vez fisgados, eles são muito mais propensos a atualizar para um dos planos pagos (que permitem mais projetos ou páginas e dá acesso a funcionalidades adicionais como upload de arquivos e encriptação de dados com SSL).

Pedacinhos

Faça pedacinhos: crie ofertas especializadas, pequenas para que os clientes mordam. Subdivida pelo menos um produto ou serviço em pedacinhos que são baratos, fáceis ou divertidos.

—Ben McConnell e Jackie Huba, autores do Church of the Customer Blog
(de What is customer evangelism?)

Dê Sua Música de Maior Sucesso

Considere doar uma de suas músicas (por álbum) como download gratuito promocional para o mundo – para ser como um trailer de cinema – como o single de sucesso enviado ao rádio – a música que faz as pessoas quererem comprar sua música.

Não se preocupe com pirataria dessa música. Deixe as pessoas tocarem, copiarem, compartilharem. Tenha a confiança que, se o mundo a ouviu, irão pagar por mais.

—Derek Sivers, presidente e programador, CD Baby e HostBaby (de Free Promo Track)

Fácil entrar, fácil sair

Torne assinatura e cancelamento processos indolores

Torne simples o processo de assinar – e cancelar – o seu serviço.

Se eu sou um cliente que quero usar seu aplicativo, espero que seja um processo indolor e óbvio. Providencie um botão de assinatura grande, claro, que pula e coloque-o em cada página do seu website de marketing. Anuncie às pessoas como é fácil: “Da assinatura ao login em apenas 1 minuto!”

Sempre deve existir uma opção grátis para que os clientes possam experimentar o aplicativo sem entrar com informações de cartão de crédito. Alguns de nossos competidores requerem uma ligação de retorno, um compromisso, ou uma senha especial para poder experimentar seus produtos. Qual é o problema disso? Nós deixamos qualquer um experimentar nossos aplicativos de graça a qualquer hora.

Mantenha o formulário de assinatura o mais curto possível. Não pergunte coisas que não precisa e não jogue

um longo e assustador questionário nas pessoas.

Os mesmos princípios permanecem verdadeiros para o processo de cancelamento. Não queremos “prender” as pessoas dentro de nosso produto. Ao mesmo tempo que sentimos muito quando as pessoas decidem cancelar suas contas de Basecamp, nunca fazemos desse processo algo intimidante ou confuso. “Cancele minha conta” é um link tão claro quanto o dia na página da conta da pessoa. Não deve existir nenhum e-mail a ser enviado, formulário especial a ser preenchido ou questões a serem respondidas.

E também garanta que as pessoas possam levar seus dados se decidirem sair. Nós garantimos que os clientes possam facilmente exportar todas as mensagens e comentários em formato XML a qualquer momento. São seus dados e eles devem poder fazer com eles o que quiserem.

Isso é crucial porque dar às pessoas o controle de suas próprias informações constrói confiança. Estamos lhes dando uma ponte para suas ilhas de dados. Permitimos que saiam sem nenhum prejuízo se encontrarem uma oferta melhor. É a coisa certa a se fazer e isso gera boa vontade.

Sair com Facilidade

Não segure usuários contra suas vontades. Se querem sair, deixe-os pegar todo o conteúdo que criaram enquanto estiveram no seu website e sairem … de graça … Temos que deixar as portas abertas e focar em manter nosso cliente alimentado, de forma que ele queira voltar, em vez de voltar porque está preso.

—Charlie O'Donnell, analista, Union Square Ventures
(de 10 Steps to a Hugely Successful Web 2.0 Company)

Coelho Bobinho, Truques são para Crianças

Evite contratos de longa duração, taxas de assinatura, etc.

Ninguém gosta de contratos de longa duração, taxas de cancelamento ou cobranças para abertura de conta. Então, evite tais cobranças. Nosso produto é cobrando na forma de assinatura mensal. Não existe contrato de tempo mínimo de utilização e pode-se cancelar a assinatura a qualquer momento, sem penalidades. E não existem, nunca, quaisquer taxas de adesão.

Não tente encontrar modos “alternativos” de conseguir mais dinheiro. Faça por merecê-lo.

Batendo de Leve

Reduza o impacto das más notícias com avisos prévios e tratamento privilegiado

Você precisa divulgar uma má notícia aos usuários – como um aumento de preços? Faça o anúncio o mais indolor possível, dando aos usuários um aviso prévio. Considere também a possibilidade de um período de bônus, isentando usuários antigos por um certo período de tempo. Estes usuários são seu ganha-pão, e é seu interesse fazê-los sentirem-se valorizados, não explorados.

Promoção capítulo 13

Lançamento de Hollywood

Vá de Trailer para a Prévia para o Lançamento

Se uma aplicação é lançada numa floresta e não há ninguém lá para usá-la, ela faz barulho? O ponto aqui é que se fazemos o lançamento da nossa aplicação sem nenhum tipo de anúncio antecipado, as pessoas não saberão sobre ela.

Para construir euforia e antecipação, vá com um lançamento holywoodiano: 1) Trailer, 2) Prévia e 3) Lançamento.

Trailer

Alguns meses antes do tempo, comece soltando dicas. Faça as pessoas saber no que está trabalhando. Publique um logotipo. Publique sobre o desenvolvimento no seu blog. Mantenha-se vago mas plante a semente. Além disso levante um website onde poderá coletar e-mails das pessoas interessadas.

Nesse estágio, devemos começar a seduzir gurus e insiders. Essas são as pessoas que estão à frente. São os formadores de opinião. Apele para suas vaidades e status como pontos-fora-da-curva. Diga-lhes que estão tendo uma breve prévia exclusiva. Se um site como Boing Boing, Slashdot ou Digg criam links para sua aplicação, terá um monte de tráfego e seguidores. Mais ainda, seu page rank no Google subirá também.

Prévia

Algumas semanas antes do lançamento, comece a demonstrar funcionalidades. Dê acesso por-trás-das-câmeras às pessoas. Descreva o tema do produto. Para o Basecamp, publicamos fotos de tela e marcamos os alertas, milestones e outras funcionalidades.

Também diga às pessoas sobre as idéias e princípios por trás da aplicação. Para o Backpack, publicamos nosso manifesto antes do lançamento. Isso levou as pessoas a pensar e falar sobre a aplicação.

Também podemos oferecer algum tipo de “ingresso especial” para algumas poucas pessoas para que possam começar a usar a aplicação antes do tempo. Ainda ganhamos o benefício de ter pessoas testando como beta enquanto sentem aquele brilho especial que todos de vanguarda sentem.

E novamente, encoraje as pessoas a se cadastrar para termos uma fundação de e-mails a ser usado no lançamento. Quando lançarmos nossa aplicação, teremos milhares de e-mails para pingar, o que é uma grande ajuda para ganhar tração.

Lançamento

É hora do lançamento. Agora as pessoas podem realmente ir ao “cinema” e ver nossa aplicação. Envie e-mails para aqueles que se cadastraram. Lance seu site de marketing completo. Espalhe a palavra tanto quanto possível. Faça blogs criarem links para você. Publique sobre seu progresso: quantas pessoas se cadastraram? Que atualizações/refinamentos foram feitas? Entre no embalo e mantenha-se nele.

A Estrada para o Dia do Lançamento

Tão logo soubemos que Blinksale iria acontecer, começamos a soltar alguns trailers em nossa lista de e-mails. Essas eram as pessoas que pediram para receber informação de nós sobre nosso projeto. Esses

são nossos fãs, se quiser chamar assim. Se você já tem permissão para falar para um grupo de pessoas, esse é o melhor lugar para começar.

A segunda coisa que fizemos foi conseguir permissão para falar a mais pessoas sobre nosso produto. Umas seis semanas antes do lançamento do Blinksale pusemos uma página de trailer em nosso site que proclamava a chegada de uma maneira mais fácil de enviar faturas online. A página dava informação apenas suficiente para construir excitação e suspense, sem entregar detalhes sensíveis que precisavam se manter confidenciais. Prominentemente mostrado na página estava um formulário de cadastro para um newsletter, pedindo não mais do que um e-mail (mantenha-se simples) para que os interessados pudessem ser notificados quando o produto fosse lançado.

Espalhamos a palavra para cerca de uma dúzia de amigos e colegas que achamos que se interessaríam pelo produto e eles começaram a espalhar a palavra sobre a página de trailer através de seus blogs e websites. Dentro de alguns dias, tivemos milhares em nossa lista de e-mails. Essas eram pessoas extremamente importantes – pessoas que estavam pedindo para saber mais sobre nosso produto e que queriam saber quando lançaríamos.

Finalmente, cerca de duas semanas antes do lançamento, convidamos vários amigos, colegas e gurus da indústria para nos ajudar nos testes beta do Blinksale. Isso nos permitiu colocar o produto na frente de pessoas que sentimos que poderiam se beneficiar dele que poderiam nos ajudar a espalhar a palavra sobre o produto quando lançássemos. É importante notar que não forçamos ninguém a escrever sobre o produto. Simplesmente queríamos que fosse visto e que falassem sobre ele quando fosse lançado. No fim, se for construir euforia dessa maneira, é melhor ter certeza que seu produto faz o que diz. Caso contrário, é como nuvens sem chuva.

Quando o dia do lançamento chegou, enviamos e-mails para nossa lista, notificamos nossos amigos dos blogs e encorajamos o pessoal do teste beta a falar o que realmente acharam. E para nossa grande alegria, o esforço pagou grandes dividendos. Logo depois do lançamento dezenas de milhares visitaram nosso site e milhares deles se cadastraram para usar o produto.

—Josh Williams, fundador, Blinksale

Um Poderoso Site Promocional

Vá do Trailer para a Prévia para o Lançamento

A melhor ferramenta promocional é um grande produto. A palavra vai se espalhar se tivermos uma aplicação que as pessoas acham realmente útil.

Ainda assim, precisamos de um bom site promocional também. O que devemos incluir nesse site? Algumas idéias:

- **Apresentação**: Explique sobre a aplicação e seus benefícios.
- **Turismo**: Guie as pessoas pelas várias funcionalidades
- **Fotos de tela e vídeos**: Mostre às pessoas como sua aplicação realmente se parece e como usá-la.
- **Manifesto**: Explique a filosofia e idéias por trás dela.
- **Estudos de Caso**: Dê exemplos reais que mostram o que é possível.
- **Euforia**: Frases testimoniais de clientes, revisões, imprensa, etc.

- **Fórum**: Ofereça um local para membros da comunidades se ajudarem uns aos outros.
- **Precificação e Assinatura": Leve as pessoas à aplicação o mais rápido possível.**
- Weblog": Blogs mantém seu site atualizado com notícias, dicas, etc.

Cavalgue pela Onda dos Blogs

Blogar pode ser mais efetivo do que propaganda (e é muito mais barato)

Propaganda é caro. E calcular a eficácia de vários tipos pode acabar sendo ainda mais caro do que a propaganda em si. Quando não tiver o tempo ou o dinheiro para ir pela rota tradicional de propaganda, em vez disso considere a promoção-via-blog.

Comece criando um blog que não apenas fale sobre seu produto mas oferece bons conselhos, dicas, truques, links, etc. Nosso blog Signal vs. Noise recebe milhares de leitores únicos por semana graças aos pedaços que ajudam, informam e são interessantes e às anedotas que publicamos quase diariamente.

Então, quando chegou a hora de promover nosso primeiro produto, Basecamp, começamos lá. Liberamos a palavra sobre o SvN e ela começou a se espalhar. Pessoas como Jason Kottke, os BoingBoingers, Jim Coudal e uma variedade de pessoas com blogs populares ajudaram a crescer a visibilidade e as coisas fluíram.

Ta-da Lists é outro grande exemplo do poder do marketing baseado em blogs. Lançamos Ta-da com uma única publicação no Signal vs. Noise e algumas semanas depois ela foi mencionada em mais de 200 blogs e mais de 12 mil pessoas se cadastraram para suas próprias contas Ta-da. Palavra sobre Backpack se espalhou ainda mais rápido. Dentro de 24 horas do lançamento, mais de 10 mil haviam se cadastrado.

Solicite Antecipadamente

Consiga euforia antecipada e cadastros acontecendo o mais rápidos possível

Já falamos sobre isso mas vale a pena repetir: consiga algum tipo de site no ar e comece a coletar e-mails o mais depressa quanto for possível. Pegue seu nome de domínio e coloque um logotipo e talvez uma sentença ou duas que descreva, ou pelo menos dê dicas sobre o que sua aplicação fará. Então deixe as pessoas dar seus endereços de e-mail. Agora você está no caminho de ter uma fundação de pessoas prontas e esperando para serem notificadas no seu lançamento.

Promova Através da Educação

Compartilhe seu conhecimento com o mundo

Quando um professor aparece como competidor no programa americano de perguntas e respostas, Jeopardy, o apresentador Alex Trebek comenta que é uma "nobre profissão". Ele está certo. Existe

definitivamente alguma coisa maravilhosa e recompensadora sobre dividir seu conhecimento com os outros. **E quando o assunto que está ensinando é sua aplicação, ela serve um duplo propósito: Você pode dar alguma coisa de volta à comunidade que o suporta, e marcar uma boa exposição promocional ao mesmo tempo**.

Como uma técnica de promoção, educação é um jeito sutil de ter seu nome – e o nome de seu produto – na frente de mais pessoas. E em vez de uma aproximação dura de vendas do tipo “compre este produto”, você está conseguindo atenção fornecendo um serviço de valor. Isso cria euforia positiva que técnicas tradicionais de marketing não conseguem igualar. Pessoas que você educa se tornarão seus evangelistas.

Educação pode vir de diversas formas. Publique dicas e truques no seu site que as pessoas irão querer compartilhar com os outros. Fale em conferências e fique até depois para se encontrar e agradecer os participantes. Conduza sessões práticas de demonstração para que fãs curiosos possam aprender mais e falar com você ao vivo. Dê entrevistas para publicações. Escreva artigos que compartilhem informações úteis. E escreva livros. ;)

Um exemplo de nossa própria história é a Técnica do Amarelo que Desvanesce, um método que inventamos para sutilmente iluminar uma área que recentemente mudamos em nossa página. Escrevemos uma publicação sobre isso na Signal vs. Noise. Essa publicação circulou e trouxe milhares e milhares de visitas à página (até hoje está fazendo um grande tráfego).

A publicação funcionou tanto no nível educacional quanto promocional. Uma lição foi aprendida e muitas pessoas que nunca saberiam sobre nossos produtos foram expostos a eles. Outro exemplo: durante nosso desenvolvimento de Ruby on Rails, decidimos tornar a infra-estrutura como código aberto. Acabou sendo um movimento sábio. Demos alguma coisa de volta à comunidade, construímos boa vontade, ganhamos reconhecimento para nossa equipe, recebemos respostas úteis e começamos a receber correções e contribuições de programadores por todo o mundo.

Ensinar tem tudo a ver com bom karma. Pagamos antecipadamente. Ajudamos os outros. Ganhamos alguma promoção saudável. E podemos até mesmo nos sentir mais nobres. Portanto, o que você sabe que o mundo gostaria de ouvir?

Pague Antecipadamente

A seção de artigos e dicas em nosso blog é uma das mais populares de nosso site. Passar nosso conhecimento sobre marketing por e-mail garante que nossos clientes tirem o máximo de nosso software. Se eles podem fornecer um serviço melhor a seus clientes, é mais provável que tenham mais negócios, que por sua vez cria mais negócios para nós – todos ganham.

Dividir gratuitamente nosso conhecimento também ajuda a nos posicionar como especialistas na indústria e reforça nossos relacionamentos com os clientes atuais. Eles sabem que nos importamos sobre qualidade do nosso trabalho. Finalmente, recebemos montanhas de tráfego direcionado a partir de sites de pesquisa e bloggers que dividem nossos artigos com seus leitores. Essas são pessoas que nunca teriam ouvido falar de nosso software se não tivéssesmos escrito esse artigo.

—David Greiner, fundador, Campaign Monitor

Comida-Funcionalidade

Eles estão famintos por isso então sirva-os

Funcionalidades novas ou interessantes são uma grande maneira de gerar euforia para sua aplicação. Grupos de interesse especiais amam mastigar “comida de funcionalidade” e cuspir de volta à comunidade.

Tudo bem, essa foi uma analogia não muito boa mas você entendeu o ponto.

Por exemplo, usando Ruby on Rails, uma nova plataforma de desenvolvimento, geramos uma tonelada de atenção para o Basecamp dentro da comunidade de desenvolvedores.

Os elementos Ajax que usamos em nossa aplicação recebeu montanhas de euforia e até mesmo levou a revista Business 2.0 a nomear a 37signals um “competidor chave em Ajax” junto com grandes nomes como Google, Yahoo, Microsoft e Amazon.

Outro exemplo: Bloggers tomaram nota do suporte RSS do Basecamp já que foi um dos primeiros exemplos de negócios com RSS.

Integração com iCal, uma funcionalidade menor à primeira vista, nos levou às notícias em uma tonelada de sites relacionados a Mac, que caso contrário provavelmente nunca teriam mencionado nossa aplicação.

Equipes pequenas tem uma perna maior na integração de novas idéias com software. Enquanto grandes empresas precisam lidar com afunilamentos de burocracia, podemos rapidamente implementar novas idéias e ganhar atenção por usá-las.

Cavalgar junto com as tecnologias mais recentes e que mais fazem barulho é um jeito efetivo e barato de construir euforia. Dito isso, não vá adicionando a mais recente e obscura tecnologia apenas para ganhar mais atenção. Mas se estiver usando alguma coisa nova e merecedora de atenção, vá em frente e anuncie isso para grupos de interesses especiais.

Monitore Seus Logs

Estude seus logs para monitorar a euforia

Você precisa saber quem está falando sobre você. Cheque seus logs e encontre de onde está vindo a euforia. Quem está com links para você? Quem está falando mal de você? Que blogs listados no Technorati, Blogdex, Feedster, Del.icio.us and Daypop estão quentes na sua trilha?

Descubra e então faça sua presença ser sentida. Deixe comentários nesses blogs. Agradeça as pessoas por publicarem links. Pergunte se eles querem ser adicionados à sua lista avançada especial para que estejam entre os primeiros a saber sobre lançamentos futuros, atualizações, etc. Colete elogios e crie uma página de “euforia” no seu site. Testemunhos são uma grande maneira de promover sua aplicação uma vez que elogios dos outros é mais confiável para a maioria das pessoas.

Se os comentários são negativos, ainda assim preste atenção. Mostre que está ouvindo. Responda às críticas com reflexão. Algo do tipo: “Agradecemos as opiniões mas fizemos dessa forma porque ..” ou “Você levantou um ponto importante e estamos trabalhando nisso”. Você irá amaciar a crítica e colocar um rosto humano em seu produto. É incrível como um comentário bem refletido em um blog pode dissolver pessoas negativas e mesmo transformar quem reclamava em evangelistas.

Vendas Internas Pró-Ativas

Promova oportunidades de atualização dentro de sua aplicação

Todo mundo sabe ser agudo no site de marketing. Mas as vendas não devem parar lá. Se tiver um plano de

preços por níveis (ou uma versão livre de sua aplicação), não se esqueça de chamar para oportunidades de atualização de dentro do produto.

Diga as pessoas que você removerá as barreiras se atualizarem. Por exemplo, no Basecampo não se pode enviar arquivos se tiver uma conta grátis. Quando alguém tentar enviar um arquivo, não simplesmente negamos. Explicamos porque o envio de arquivos não está disponível e os encorajamos a atualizar para uma versão paga além de explicar porque essa é uma boa idéia. A mesma aproximação é usada para encorajar clientes já existentes a atualizar para uma conta de nível maior quando chegam ao máximo de seu plano atual.

Clientes existentes são suas melhores apostas de vendas. Não se sinta envergonhado em tentar repetir negócios com pessoas que já conhecem e usam seus produtos.

Nome-Gancho

Dê um nome à sua aplicação que seja fácil de lembrar

Um grande erro que muitas pessoas fazem é pensar que o nome de sua aplicação precisa ser ultra-descritiva. Não se preocupe em escolher um nome que vividamente descreva o propósito de sua ferramenta; isso normalmente leva apenas a um nome genérico e esquecível. Basecamp é um nome melhor do que algo como Centro de Gerenciamento de Projetos ou ProjectExpress. Writeboard é melhor do que CollaborEdit.

Além disso, não foque muito em grupos ou comitês para o processo de nomeação. Escolha um nome curto, que pegue, seja memorável e então vá com ele.

E não se preocupe se não conseguir o nome de domínio exato que quer. Você sempre pode ser criativo e chegar perto com um pouco mais de letras (ex. backpackit.com ou campfirenow.com).

Fácil é o Melhor

Será que a indústria de tecnologia não percebe que pensar em nomes que peguem e que sejam auto-explicativos os beneficiariam da mesma maneira em última instância? Eles venderiam mais do que quer que seja, porque não assustariam os consumidores que pensam que estão sendo mantidos fora do club high-tech por um punhado de engenheiros arrogantes. A tecnologia avançaria mais rápido também. O novo produto seria mais fácil de descrever, mais fácil de usar e mais fácil de comprar – o que, para as empresas, significa mais fácil de vender.

—David Pogue, colunista, New York Times (de O que há no nome de um produto?)

Suporte capítulo 14

Sinta a Dor

Derrube as paredes entre suporte e desenvolvimento

No negócio de restaurantes, existe uma enorme diferença entre aqueles que trabalham na cozinha daqueles

que estão na linha de frente lidando com clientes. É importante para ambos os lados entender e simpatizar com o outro. É por isso que escolas de culinária e restaurantes normalmente terão *chefs* trabalhando como garçons para que a equipe da cozinha possa interagir com clientes e ver como é realmente estar na linha de frente.

Muitas empresas desenvolvedoras de software tem uma divisão similar. Designers e programadores trabalham na "cozinha" enquanto o suporte lida com clientes. Infelizmente, isso significa que *chefs* de software nunca ouvem o que o cliente realmente está dizendo. Isso é problemático porque ouvir clientes é a melhor maneira de se ligar nas partes fortes e fracas do seu produto.

A solução? Evite construir paredes entre seus clientes e a equipe de desenvolvimento/design. **Não terceirize o suporte a seus clientes**. Faça você mesmo o suporte. Você e sua equipe inteira, devem saber o que seu cliente está dizendo. Quando seu cliente está incomodado, você precisa saber disso. Você pecisa ouvir as reclamações. Você precisa ficar incomodado também.

Na 37signals, todos os e-mails de suporte são respondidos pessoalmente pelo pessoal que realmente construiu o produto. Por que? Primeiro, isso fornece melhor suporte aos clientes. Eles estão recebendo uma resposta diretamente do cérebro de alguém que construiu a aplicação. Além disso, isso nos mantém em contato com a pessoa que usa nossos produtos e com os problemas que estão encontrando. Quando estão frustrados, nós ficamos frustrados. Podemos dizer sinceramente que "eu sinto sua dor".

Pode ser tentador se apoiar em análises estatísticas para revelar seus pontos problemáticos. Mas estatísticas não são como vozes reais. Você precisa eliminar a maior quantidade possível de atravessadores entre você e as vozes reais de seus clientes.

As linhas de frente são onde a ação está. Vá até lá. Faça seus *chefs* trabalharem como garçons. Leia e-mails de clientes, ouça suas frustrações, escute suas sugestões e aprenda com elas.

Corte fora os intermediários

Quase todo desenvolvimento do Campaign Monitor, suporte e marketing são feitos por duas pessoas. Mesmo que fôssemos forçados a expandir a equipe, nunca separaríamos a equipe suporte do time de desenvolvimento. Quando respondemos pessoalmente cada requisição, nos forçamos a nos colocar no lugar de nossos clientes e vemos as coisas de sua perspectiva.

É importante entender porque seus clientes precisam de alguma coisa, não apenas o que eles precisam. Esse contexto normalmente tem um impacto direto em como desenhamos alguma coisa. Corte fora os intermediários. É muito mais fácil dar a seus clientes o que querem quando é possível ouví-los diretamente.

Discuti essa configuração com muitas pessoas e a primeira resposta normalmente é "você não deveria contratar um estagiário para lidar com seu suporte?" Ponha-se no lugar de seu cliente. Se quiser um bife cozinhado do seu jeito, você preferiria falar com o motoboy ou o chef que irá cozinhá-lo?

—David Greiner, fundador, Campaign Monitor

Treinamento Zero

Use ajuda em contexto e FAQs para que seu produto não precise de um manual ou treinamento

Você não precisa de um manual para usar o Yahoo! ou Google ou Amazon. Então por que você não pode construir um produto que não requer manual? Se esforce para construir uma ferramenta que requer

treinamento zero.

Como fazer isso? Bem, como mencionamos antes, você começa mantendo tudo simples. Quanto menos complexa for sua aplicação, menos precisará ajudar as pessoas sem necessidade. Depois disso, uma grande maneira de suporte pró-ativo é usando ajuda em contexto e FAQs em potenciais pontos de confusão.

Por exemplo, oferecemos suporte pró-ativo na tela que permite as pessoas a fazer upload de seus logotipos ao Basecamp. Algumas pessoas experimentaram um problema onde continuavam vendo um logotipo antigo por causa do cache do browser. Então, próxima à área de “envie seu logotipo”, adicionamos um link a um FAQ que instruía os clientes a forçar um recarregamento de seus browsers para ver o novo logotipo. Antes de fazermos isso recebíamos 5 e-mails por dia sobre esse problema. Agora, não recebemos nenhum.

Resposta Rápida

Tempo rápido de atendimento em consultas de suporte devem ser prioridade máxima

Os clientes se alegram quando você responde suas questões rapidamente. Eles estão tão acostumados a respostas enlatadas que aparecem dias depois (se muito), que você pode realmente se diferenciar dos concorrentes oferecendo uma resposta bem pensada, imediatamente. Durante o horário comercial, respondemos 90% de nossos e-mails de requisição de suporte dentro de 90 minutos – e normalmente dentro de meia hora. E as pessoas amam isso.

Mesmo que não tenha uma resposta perfeita, diga alguma coisa. Você pode comprar boa vontade com uma resposta entregue rapidamente de forma aberta, honesta. Se alguém está reclamando sobre um problema que não pode ser consertado imediatamente, diga algo como “ouvimos o que está dizendo e estaremos trabalhando nisso no futuro”. É uma grande maneira de diluir uma situação potencialmente negativa.

Clientes gostam de coisas diretas e normalmente mudam de irritados para educados se responder rapidamente e de maneira direta.

Um Exército de Muitos

Como pode uma equipe pequena de apenas três desenvolvedores criar um produto inovador e competir com sucesso com os caras grandes? A resposta é alistar um numeroso exército.

Lembre-se no primeiro dia que seus clientes são seu patrimônio mais importante e que são absolutamente vitais para o sucesso de longo prazo; portanto trate sua comunidade de usuários como a realeza. A maneira de competir com os caras grandes é começando pequeno e prestando atenção a cada um de seus clientes.

É seu cliente o primeiro que irá alertá-lo de bugs, o primeiro que irá alertá-lo de necessidades que não foram cumpridas e são seus primeiros clientes que carregarão a bandeira e espalharão sua mensagem.

Isso não significa que seu produto tenha que ser perfeito quando for lançado. Muito pelo contrário, lance cedo e freqüentemente. Entretanto, quando seus clientes encontrarem bugs, garanta o envio de uma resposta rápida agradecendo pela sua informação.

Os clientes não esperam que seu produto seja perfeito e não esperam que todas as suas funcionalidades serão implementadas. Entretanto, esperam que esteja ouvindo e mostrando que se importa. Essa é uma área onde a maioria da grandes empresas mostra um grande descaso, portanto desenvolva um senso de comunidade cedo.

Na Blinklist, cada um dos e-mails de cliente é respondido, normalmente dentro da primeira hora (a menos que estejamos dormindo). Também temos um fórum online e garantimos que cada postagem e comentário seja entendido.

Igualmente importante, todos os nossos desenvolvedores recebem o feedback dos clientes e são participantes ativos nos fórums de discussão online. Dessa maneira estamos, lentamente mas, certamente construindo uma comunidade ativa e leal na BlinkList.

—Michael Reining, co-fundador, MindValley & Blinklist

Amor áspero

Esteja preparado para dizer não a seus clientes

Quando falamos de requisição de funcionalidades, o cliente nem sempre está certo. Se adicionássemos cada uma das coisas que nossos clientes pediram, ninguém iria querer nossos produtos.

Se fôssemos obedecer cada choro de nossos clientes, o Basecamp teria: gerenciamento de tempo completo, cobrança completa, cronograma de reuniões completo, sistema de calendário completo, sistema de dependência de tarefas completo, chats via mensagens instantâneas completo, funcionalidade de wiki completo, e tudo-mais-que-puder-imaginar completo.

Ainda assim, a requisição número 1 que recebemos nas pesquisas com os clientes é manter o Basecamp simples

Aqui vai outro exemplo: Apesar de algumas reclamações, decidimos não suportar o Internet Explorer 5 (IE5) em nossos produtos. Isso era 7% do mercado que estávamos endereçando. Mas decidimos que era mais importante nos preocupar com os outros 93%. Consertar bugs e testar para IE5 simplesmente não valia a pena. Em vez disso fizemos um produto melhor para todo o resto.

Como uma empresa de desenvolvimento de software, você precisa agir como um filtro. Nem tudo que todo mundo sugere é a resposta correta. Consideramos todas as requisições mas o cliente nem sempre está certo. Haverá tempos em que você precisará simplesmente deixar alguém irritado. *C'est la vie*.

Relacionado a isso, é crítico que você, enquanto empresa de desenvolvimento, ame seu produto. E você não ama seu produto se estiver cheio de um monte de coisas com as quais não concorda. Essa é outra justificativa para vetar requisições de clientes que não acredita que sejam necessárias.

Em Fórum Afinado

Use fórums ou chats para deixar os clientes se ajudarem

Fórum e chats de grupo baseados na web são uma grande maneira de deixar clientes fazerem perguntar e ajudar uns aos outros. Eliminando o intermediário – esse é você – você fornece uma linha aberta de comunicação e economiza seu tempo no processo.

Em nossos fóruns de produtos, os clientes publicam dicas e truques, requisições de funcionalidades, histórias e mais coisas. Nós aparecemos de tempos em tempos para oferecer assistência, mas os fóruns são principalmente um lugar para a comunidade se ajudar e compartilhar experiências com o produto.

Você ficará surpreso com quantas pessoas querem se ajudar.

Publique suas burradas

Coloque as más notícias lá fora e fora do caminho

Se alguma coisa vai errado, diga às pessoas. Mesmo que elas nem tenham visto.

Por exemplo, Basecamp ficou fora do ar uma vez por algumas horas no meio da noite. 99% de nossos clientes nunca saberiam, mas ainda assim publicamos uma notificação de “saída do ar inesperada” no nosso blog Everything Basecamp. Achamos que nossos clientes mereciam saber.

Aqui vai uma amostra do que publicamos quando alguma coisa vai errado: “Nos desculpamos pela saída do ar nessa manhã – tivemos problemas de banco de dados que causaram grandes lentidões e saídas do ar para algumas pessoas. Consertamos o problema e estamos tomando precauções para garantir que isso não aconteça novamente … Obrigado pela sua paciência e, mais uma vez, nos desculpem pela saída do ar”.

Seja tão aberto e transparente quanto for possível. Não mantenha segredos ou se esconda. Um cliente informado é seu melhor cliente. Mais do que isso, você perceberá que a maioria dos seus deslizes nem são tão ruins assim, na interpretação de seus clientes. Eles normalmente estão felizes em lhe dar um pouco de respiro enquanto souberem que está sendo honesto com eles.

Uma observação sobre entregar notícias, ruins ou boas: quando notícias ruins chegam, abra tudo de uma vez. Boas notícias, por outro lado, devem ser desenroladas aos poucos. Se puder prolongar as boas vibrações, faça isso.

Seja Ágil, Direto e Honesto

Pode soar estranho, mas o cenário de melhor caso é quando a própria empresa relata as más notícias. É a pró-atividade que previne sua empresa de ser colocada em uma posição fraca e defensiva.

—Greg Sherwin, Vice Presidente de Tecnologia de Aplicação, CNET, e Emily Avila, Diretora, Calypso Communications (de A Primer for Crisis PR)

Pós-Lançamento capítulo 15

Um Mês para melhorias

Lance uma grande atualização 30 dias após o lançamento

Uma atualização rápida mostra embalo. Mostra que estamos ouvindo. Mostra que temos mais cartas na manga. Nos dá uma segunda onda de burburinho. Reafirma os bons sentimentos do começo. Nos dá alguma coisa sobre o que falar e para que os outros possam publicar nos blogs.

Saber que uma rápida atualização está chegando também nos faz focar nos componentes mais cruciais de antes do lançamento. Em vez de tentar espremer mais algumas coisas, podemos começar aperfeiçoando apenas o conjunto principal de funcionalidades. Então podemos lançar o produto no mundo real. Uma vez lá

fora podemos começar a receber opiniões de volta dos clientes e saberemos que áreas precisam de mais atenção.

Esse estilo de um-passo-de-cada-vez funcionou bem para o Backpack. Lançamos o produto básico primeiro e então, algumas semanas depois, adicionamos funcionalidades como Backpack Mobile para computadores portáteis e tags uma vez que essas coisas eram o que os clientes nos disseram que mais queriam.

Mantenha os Posts Chegando

Mostre que seu produto está vivo mantendo um blog operacional do desenvolvimento do produto após o lançamento

Não pare de blogar depois de lançar. Mostre que seu produto é uma criatura viva mantendo um blog dedicado e atualizado freqüentemente (pelo menos uma vez por semana, e com mais freqüência se puder).

Coisas a incluir:

- Faq (Perguntas e Respostas Freqüentes)
- How-tos (Instruções passo-a-passo)
- Dicas & Truques
- Novas Funcionalidades, atualizações e correções
- Burburinho/Imprensa

Um blog não mostra apenas que seu aplicativo está vivo, mas faz sua empresa parecer mais humana. Novamente, não tenha medo de manter o tom da conversa amigável e pessoal. Às vezes, equipes pequenas sentem que precisam soar grandes e ultra-profissionais o tempo todo. É quase como uma versão de negócios do Complexo de Napoleão. Não sue a camisa soando pequeno. Deleite-se com o fato de conseguir conversar com os clientes como amigos.

Está Vivo

Um blog com atualizações frequentes sobre um produto é o melhor indicador de que essa aplicação web está com desenvolvimento ativo, é um produto adorado e que existe uma luz acesa em casa. Um blog abandonado de um produto é um sinal de um produto abandonado e diz que as pessoas responsáveis estão dormindo no ponto.

Mantenha as conversas andando com seus usuários no blog de seu produto e seja transparente e generoso com as informações que compartilha. Deixe a filosofia de sua empresa brilhar. Link e discuta abertamente sobre concorrentes. Dê dicas de funcionalidades chegando e mantenha os comentários abertos para opiniões de retorno.

Um produto vivo é uma coisa que fala e escuta seus usuários. Um blog frequentemente atualizado sobre um produto promove transparência, um senso de comunidade e lealdade com a marca. Publicidade extra e de graça são bônus.

Como editora da Lifehacker, eu vasculho constantemente os blogs de produtos que amo – como os blogs de produtos do Google, Flickr, Yahoo, Del.icio.us e 37signals. Eu sou muito mais propensa a mencioná-los do que aplicações web que apenas enviam propaganda de imprensa unidirecional do nada e não mantém uma conversa aberta com seus usuários e fãs.

—Gina Trapani, desenvolvedora web e editora da Lifehacker, o guia de produtividade e software

---

Melhor, não Beta

Não use “beta” como uma desculpa

Ultimamente parece que tudo está em um estágio beta permanente. Isso é um escapismo. Um interminável estágio beta indica aos clientes que não estamos realmente comprometidos a liberar um produto finalizado. Isso diz, “use, mas se não estiver perfeito, não é nossa culpa”.

Beta repassa a conta ao cliente. Se não estamos confiantes o suficiente sobre nosso lançamento então como podemos esperar que o público esteja? Tudo bem com betas privados. Betas públicos são grandes bobagens. Se não está bom o suficiente para o consumo público então não o dê ao público para consumí-lo.

Não espere seu produto atingir a perfeição. Isso não vai acontecer. Assuma responsabilidade sobre o que está lançando. Coloque para fora e chame de lançamento. Do contrário, está apenas dando desculpas.

Beta não tem Sentido

Culpe o Google, e outros, por causar problemas como esse. Agora, usuários foram treinados por um monte de desenvolvedores a achar que “beta” realmente não significa nada.

—Mary Hodder, arquiteta de informação e designer de interação (de The Definition of Beta)

Todo o Tempo

Sou apenas eu ou estamos todos em beta, o tempo todo?

—Jim Coudal, fundador, Coudal Partners

Bugs: cada caso é cada caso

Priorize seus bugs (e até mesmo ignore alguns deles)

Só porque descobrimos um bug em nosso produto não significa que é hora de entrar em pânico. Todo software tem bugs – é apenas um fato da vida.

Não precisamos corrigir cada bug instantaneamente. A maioria dos bugs é incômoda, não destrutiva. Incômodos podem ser tolerados um pouco. Bugs que resultam em erros que “não parecem certos” ou outras pequenas indicações podem ser colocadas de lado de maneira segura por enquanto. Se um bug destrói seu banco de dados, no entanto, obviamente precisamos corrigí-lo imediatamente.

Priorize seus bugs. Quantas pessoas são afetadas? Quão ruim é o problema? Esse bug merece atenção imediata ou pode esperar? O que podemos fazer agora mesmo que terá o maior impacto para o maior número de pessoas? Algumas vezes adicionar uma nova funcionalidade pode ser mais importante para seu aplicativo do que corrigir um bug existente.

Finalmente, não crie uma cultura de medo ao redor de bugs. Bugs acontecem. Não fique constantemente

procurando alguém para culpar. A última coisa que queremos é um ambiente onde bugs são varridos para baixo do tapete em vez de discutidos abertamente.

E lembre-se do que dissemos antes sobre a importância da honestidade: se clientes reclamam sobre um bug, seja direto com eles. Diga-lhes que notaram o assunto e estão lidando com ele. Se não forem resolvê-lo imediatamente, diga porque e explique que está focando em áreas do produto que afetam um número grande de pessoas. Honestidade é a melhor política.

Cavalgue para Fora da Tempestade

Espere até que as reações impulsivas causadas por mudanças cessem antes de tomar uma atitude

Quando balançamos o barco, haverá ondas. Depois de apresentar novas funcionalidades, mudar a política ou remover alguma coisa, reações impulsivas, às vezes negativas, vão transbordar.

Resista à vontade de entrar em pânico ou mudar rapidamente as coisas em resposta. Paixões se acendem no começo. Mas se cavalgarmos para fora desse período inicial de 24 a 48 horas, as coisas provavelmente vão se resolver sozinhas. A maioria das pessoas respondem antes de realmente ir à fundo e usar seja lá o que foi adicionado (ou se acostumarem com o que foi retirado). Então sente-se, absorva tudo e não faça nenhum movimento até que algum tempo tenha se passado. Aí sim você será capaz de oferecer uma resposta mais adequada.

Também se lembre que reações negativas são quase sempre mais altas e mais passionais do que as positivas. De fato, você pode acabar ouvindo somente vozes negativas quando a maioria da sua base de usuários está feliz com a mudança. Garanta que não estará dando um passo para trás à toa em uma decisão necessária, mas controversa.

Fique Esperto com os Vizinhos

Assine feeds de notícias sobre seus concorrentes

Assine feeds de notícias (news feeds) sobre ambos seu produto e de seus concorrentes (é sempre sábio conhecer os caminhos de seus inimigos). Use serviços como PubSub, Technorati, Feedster e outros para se manter atualizado (para palavras-chave, use nomes de empresas e produtos). Com RSS, essa informação em constantes mudanças será entregue diretamente a você, e assim estará sempre preparado.

Cuidado com o Monstro da Gordura

Mais maduro não precisa significar mais complicado

Da forma como as coisas progridem, não tenha medo de resistir à gordura. A tentação será para aumentar. Mas não precisa ser desse jeito. Só porque alguma coisa fica velha e mais madura, não precisa significar que tem que ficar mais complicada.

Não precisamos nos tornar 'canetas de outro planeta que escrevem de ponta-cabeça'. Algumas vezes está ótimo ser apenas um lápis. Não precisamos ser canivetes suíços. Podemos apenas ser uma chave-de-fenda. Não precisamos construir um relógio de mergulho que suporta até 5 mil metros se seus clientes são amantes da terra que apenas querem saber que horas são.

Não infle tão somente para inflar. É assim que aplicativos ganham gordura.

Novo nem sempre significa melhorado. Algumas vezes existe um ponto onde devemos apenas deixar o aplicativo ser como é.

Esse é um dos benefícios-chave de construir software baseado em web em vez de software tradicional de desktop. Produtores de software de desktop como Adobe, Intuit e Microsoft precisam lhe vender novas versões todo ano. E como não podem simplesmente lhes vender a mesma versão, precisam justificar o custo adicionando novas funcionalidades. É onde começa a gordura.

Com software baseado em web e um modelo de assinatura, as pessoas pagam uma mensalidade para usar o serviço. Não precisamos ficar vendendo com a adição de mais e mais e mais, apenas precisamos providenciar um serviço contínuo de valor.

Siga o Fluxo

Esteja aberto a novos caminhos e mudanças de direção

Parte da beleza de uma aplicação web é sua fluidez. Não a empacotamos em uma caixa, entregamos e então esperamos anos para o próximo lançamento. Podemos refinar e mudar na medida em que avançamos. Esteja aberto ao fato que sua idéia original pode não ser sua melhor idéia.

Veja o Flickr. Ele começou como um jogo online para múltiplas pessoas chamado "O Jogo que Nunca Acaba" (The Game Neverending). Seus criadores logo entenderam que o aspecto de compartilhamento de fotos do jogo era um produto mais plausível do que o próprio jogo em si (que foi eventualmente engavetado). Esteja pronto para admitir erros e mudar o curso.

Seja um surfador. Observe o oceano. Descubra onde as ondas grandes estão quebrando e ajuste-se de acordo.

Conclusão capítulo 16

Liguem seus Motores

Feito!

Tudo certo, você conseguiu! Se tudo deu certo está psicologicamente preparado para começar Caindo na Real com sua aplicação. Realmente nunca houve uma época melhor para fazer grandes softwares com recursos mínimos. Com a idéia certa, paixão, tempo e habilidade, o céu é o limite.

Alguns pensamentos de conclusão:

Execução

Qualquer um pode ler um livro. Qualquer um pode chegar com uma idéia. Qualquer um tem um primo que é um web designer. Qualquer um pode escrever um blog. Qualquer um pode contratar alguém para grudar algum código.

A diferença entre você e qualquer um será quão bem você executa. Sucesso tem tudo a ver com uma grande execução.

Para software, isso significa fazer um monte de coisas certas. Você não pode somente ter uma boa escrita mas falhar em entregar as promessas na sua prosa. Design limpo de interface não vai dar certo se seu código é cheio de gambiarras. Uma grande aplicação não vale nada se promoção pobre significa que ninguém saberá sobre ela. Para pontuar grande, precisa combinar todos esses elementos.

A chave é balanço. Se for longe demais em uma direção, está caminhando para o fracasso. Constantemente procure seus pontos fracos e foque neles até estar nivelado.

Pessoas

Vale a pena enfatizar a coisa que achamos que é o ingrediente mais importante quando falamos em construir uma aplicação web de sucesso: as pessoas envolvidas. Mantras, designs de epicentro, menos software e todas essas idéias maravilhosas não vão realmente importar se não tiver as pessoas certas a bordo para implementá-las.

Você precisa de pessoas que são apaixonadas pelo que fazem. Pessoas que se importam pela seu artesanato – e que realmente acham que é um artesanato. Pessoas que se orgulham do seu trabalho, independentemente da recompensa monetária envolvida. Pessoas que suam nos detalhes mesmo que 95% das pessoas nem saibam distinguir as diferenças. Pessoas que querem construir alguma coisa grande e não se conformam com menos. Pessoas que precisam de pessoas. Ok, não necessariamente essa última coisa mas não iríamos resistir não jogar um pouco de Streisand na mistura. De qualquer forma, quando encontrar essas pessoas, segure-se nelas. No final, as pessoas da sua equipe farão ou quebrarão seu projeto – e sua empresa.

Mais Que Apenas Software

Também vale a pena notar que o conceito de Caindo na Real não se aplica apenas a construir aplicações web. Uma vez que você começa a tocar nas idéias envolvidas, as encontrará em todos os lugares. Alguns exemplos:

- Forças de Operações Especiais, como os Boinas Verdes ou Navy Seals usam equipes pequenas e entrega rápida para atingir tarefas onde outras unidades são grandes demais ou lentas demais para cumprir.
- Os White Stripes abraçam restrições seguindo uma fórmula simples: duas pessoas, músicas enxutas, baterias infantis, manter o tempo de estúdio ao mínimo, etc.
- O iPod da Apple se diferencia da concorrência não oferecendo funcionalidades como rádio FM embutido ou gravador de voz.
- No futebol americano, jogadas rápidas ajudam a ganhar terreno rapidamente, eliminando a “burocracia” das jogadas ensaiadas.
- Ernest Hemingway e Raymond Carver usavam linguagem simples e limpa e ainda assim entregavam impacto máximo.
- Shakespeare revelou, nas limitações dos sonetos, poemas líricos de catorze linhas em pentâmetro iâmbico.

- E assim por diante …

Claro, Caindo na Real é sobre construir grandes softwares. Mas não há razão para parar por aí. Pegue essas idéias e tente aplicá-las em diferentes aspectos de sua vida. Você pode acabar atingindo resultados interessantes.

Mantenha Contato

Nos deixe saber como Caindo na Real funcionou para você. Mande e-mail para gettingreal [at] 37signals [ponto] com.

Além disso, mantenha-se atualizado sobre as últimas ofertas da 37signals visitando Signal vs. Noise, nosso blog sobre Caindo na Real, usabilidade, design e um monte de outras coisas.

Obrigado por ler e boa sorte!

37signals Resources

- 37signals site
- Signal vs. Noise weblog
- Basecamp — Web-based project collaboration
- Campfire — Web-based group chat for business
- Backpack — Web-based information organizer
- Writeboard — Web-based collaborative writing
- Ta-da List — Web-based dead-simple to-do lists
- Ruby on Rails — Open-source web application framework

Tradução

Organização: Fabio Akita

Agradecimentos aos seguintes tradutores: Herval Freire, Juraci Krohling Costa, Marcello Rocha, Diogo Bispo, Adriano Mitre, Ricardo Augusto, Rodrigo Kochenburger

E também aos revisores: Mateus Del Bianco, Diogo Bispo, Davis Zanetti Cabral, Gustavo Cardoso, Ricardo Augusto