PROJETOS ÁGEIS E OS 12 PONTOS DE ALAVANCAGEM SISTÊMICA

por Alisson Vale

Uma das coisas mais interessantes em se estudar e trabalhar com o modelo ágil é ser capaz de estabelecer novas formas de entendimento sobre os mecanismo de funcionamento de um projeto de software. Entender o modelo ágil nos leva a ver o potencial de um projeto quando este se torna uma entidade sistêmica, adaptativa e auto-organizável. É um ótimo primeiro passo na direção de ser capaz de influenciá-lo quando o intuito é produzir melhores resultados.

A teoria de sistemas preenche as grades curriculares das universidades já a bastante tempo. Na verdade, desde que Ludwig von Bertalanffy formalizou os primeiros estudos nessa área na década de 1930, a ciência não é mais a mesma. Desde então, um sistema é considerado e estudado como um conjunto de agentes que interagem entre si produzindo um todo maior do que a simples soma de suas partes. Já na década de 1990, Peter Senge trouxe novas idéias para a aplicação do estudo sistêmico em organizações (Senge, 1990), agarrando-se no fato de que as organizações modernas são sistemas complexos que não podem ser estudados sob o ponto de vista de simples análises de causa-e-efeito. Muito pelo contrário, na maioria das vezes causa e efeito aparecem tão distantes entre si que é difícil relacioná-las facilmente.

Sistemas se tornam interessantes quando apresentam comportamentos e características generalizáveis, tornando-os passíveis de estudo e classificação. A análise sobre o funcionamento de projetos de software, quando realizada sob uma perspectiva sistêmica, pode nos oferecer elementos importantes capazes de nos ajudar a não só entendê-los melhor, mas também a influenciar o seu comportamento na direção que desejamos.

Recentemente tive contato com o trabalho de Donela H. Meadows , cientista e estudiosa da teoria sistêmica que propôs, ao final da década de 90, aqueles que ficaram conhecidos como os 12 pontos de alavancagem de um sistema (Meadows, 1999). Cada ponto revela uma forma de atuar sistemicamente de forma a influenciar o seu comportamento e, conseqüentemente, os resultados que eles geram. Qual foi a minha surpresa quando, depois de uma rápida análise em seus argumentos, me deparei com pontos de alavancagem extremamente coerentes com os princípios e práticas utilizadas pelo modelo ágil para desenvolvimento de software. Nesse artigo, eu procuro explorar cada um desses pontos contextualizando-os com as práticas e princípios sugeridos pelo Movimento Ágil. Para o leitor já familiarizado com esse novo paradigma, há nesse texto uma valiosa oportunidade para absorver conceitos que podem efetivamente ajudar na melhoria de seus projetos.

Visão Sistêmica em Projetos Ágeis

Jim Highsmith fundamentou sua abordagem para desenvolvimento de software na teoria dos sistemas complexos adaptativos. O livro *Adaptive Software Development – A Collaborative Approach to Managing Complex Systems* (Highsmith J. , 2000), publicado ainda antes do Manifesto Ágil (AgileAlliance, 2001), é uma fonte preciosa de elementos capazes de justificar as idéias do Movimento Ágil. A dinâmica organizacional em um projeto de software, segundo Highsmith, pode ser compreendida por meio da ciência dos sistemas adaptativos complexos. Um sistema adaptativo complexo possui características bem similares àquelas que encontramos hoje em projetos ágeis: (1) São complexos globalmente, mas apresentam grande simplicidade local; (2) Não são determinísticos, mas é possível prever seu comportamento em uma escala de incerteza aproveitável; (3) Há grande sinergia entre seus componentes, o que torna o todo maior que a soma de suas partes; (4) Se adaptam a mudanças ambientais; (5) São auto-funcionais, pois não possuem uma coordenação central. Se há uma liderança, esta não comanda o sistema, apenas se inter-relaciona com ele.

A teoria de sistemas complexos, quando aplicadas a projetos de software, revela novos modelos conceituais que corroboram de maneira significativa com as abordagens ágeis para gestão de projetos de software. Highsmith ressalta alguns pontos principais que devem ser considerados em ambientes altamente competitivos:

a. Adaptação é mais importante que otimização;

b. Adaptação depende mais de liderança e colaboração do que de comando e controle;
c. Gerir exceções a um processo linear é mais comum do que gerir o próprio processo;
d. O ciclo de vida de software e seu gerenciamento devem estar baseados em *workstate* ao invés de *workflow*. Onde *workflow* foca em se gerir o andamento de tarefas e *workstate* em se gerir *deliverables* e seu estado de conclusão.
e. Emergência, ou seja, a habilidade de sistemas complexos em se produzir resultados por meio da interação espontânea de elementos auto-organizáveis, é o motor do processo de adaptação e evolução;
f. Colaboração é o elemento primordial para geração de emergência.
g. O maior risco em desenvolvimento de software é superestimar nosso próprio conhecimento e nossas próprias suposições sobre o que vamos desenvolver e sobre como vamos desenvolver.

Métodos e metodologias seguidoras do paradigma ágil estão alinhadas a esses conceitos teóricos e estabelecem diferentes modelos de aplicação no contexto de um projeto de software. A análise sistêmica segue um padrão que é indiferente quanto à escolha metodológica. Em geral, todas as metodologias ágeis participantes do Manifesto original são aderentes aos conceitos expostos nos itens de (a) a (g). Uma representação simplificada do funcionamento sistêmico no modelo ágil pode ser visualizada na Figura 1. Nesse modelo, observamos um sistema composto basicamente por tudo aquilo que é necessário para transformar um conjunto de requisitos, desejos e idéias (na figura organizados segundo a técnica MOSCOW do DSDM para maior clareza) e a geração de Business Value (na maioria das vezes materializado em termos de software). Tudo que é gerado pelo sistema que não é Business Value, é desperdício. Assim como, em um motor de combustão, nem todo combustível pode ser transformado em energia, em projetos de software nem toda as atividades geram diretamente business value. É um subproduto necessário e inevitável. Por isso, o Lean Software Development (Poppendieck & Poppendieck, 2003) é um bom complemento para abordagens ágeis na medida em que permite o uso de práticas e ferramentas capazes de reduzir e controlar os níveis de desperdício em um projeto de software.

O estado do sistema em um projeto ágil é representado pela discrepância entre o volume de demandas que ainda não entraram no sistema e o volume de business value que sai. Gráficos como os de Burndown, muito utilizados em projetos ágeis, são bons exemplos de representações do estado do sistema. O estado do sistema também pode ser representado por medidas mais abstratas como a satisfação do cliente ou a motivação da equipe. Quando a discrepância, ou seja a diferença entre a meta do sistema e seu estado atual chega a zero, o sistema então cumpriu seu objetivo.

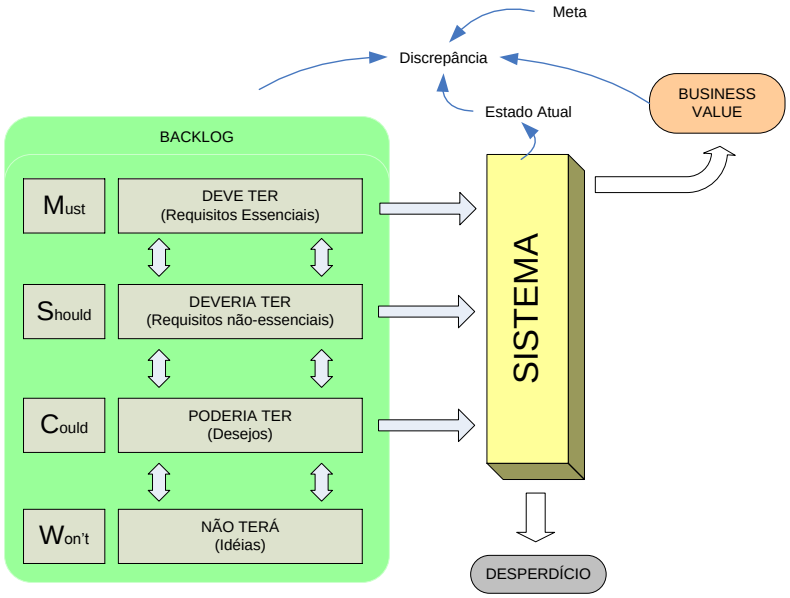

Figura 1: Modelo sistêmico simplificado em um projeto ágil.

Com essa básica noção do modelo sistêmico em projetos ágeis, podemos seguir na direção de conhecer os 12 pontos de alavancagem descritos a seguir. A contextualização desses pontos no universo dos projetos ágeis tem dois propósitos básicos: o primeiro é oferecer uma nova perspectiva de entendimento do efeito sistêmico que algumas práticas ágeis geram; o segundo é abrir o leque de possibilidades para intervenções que gerem melhoria sustentável e significativa em nossos projetos de software.

Os 12 Pontos de Alavancagem

Pontos de Alavancagem são elementos sistêmicos que, quando manipulados ou influenciados de certa maneira, podem gerar uma grande transformação no sistema como um todo. Como cada sistema possui suas próprias características de funcionamento, estabelecer elementos genéricos que causem esse tipo de transformação não é muito fácil. No entanto, os 12 pontos apresentados aqui representam um bom começo. Eles surgiram em 1997 em uma reunião onde se tentava organizar um novo regime de comércio mundial (envolvendo NAFTA, GATT e a OMC) que

promovesse um mundo melhor por meio do progresso econômico. A cientista ambiental Donella Meadows estava presente e, depois de se frustar com o rumo das soluções discutidas, apresentou um rascunho daquilo que se tornaria, mais tarde, os 12 pontos de alavancagem sistêmica. Cada ponto ilustra uma maneira de influenciar o funcionamento de um sistema. Eles são apresentado em ordem crescente de potencial de efetividade, ou seja, o ponto 12 tem um poder pequeno de alavancagem, enquanto que o ponto 1 terá um enorme poder de alavancagem. Tive o cuidado de mantê-los na ordem original sugerido pela autora, mas você poderá observar que, no caso de projetos de software, essa ordenação poderia ser outra. Isso porque a posição de muitos deles levou em conta a sua dificuldade de aplicação, que em casos de software pode ser bem menor. Por exemplo, o ponto 10 trata sobre modificações na estrutura do fluxo de inventário. Mudar toda uma estrutura de um sistema de tráfego em uma cidade é uma tarefa incrivelmente complexa. Assim como seria difícil mudar toda a estrutura de encanamento do sistema hidráulico de uma casa. No entanto, mudar a estrutura operacional e de trabalho em um projeto de software não é tão complicado assim. Portanto, a ordem aqui não é sempre relevante, apesar de ainda manter alguma relevância como será visto no texto.

Ponto 12 - Constantes, Parâmetros e Números. Parâmetros medem a eficiência do seu sistema em transformar o que ele recebe no que ele gera. Em projetos ágeis, um parâmetro chamado Velocidade é muito utilizado. Ele mede o quanto você é capaz de tranformar requisitos em software dado um certo período de tempo (1 iteração). Alterar esse parâmetro (aumentando o número de pessoas na equipe, por exemplo) é uma forma de alavancar o sistema. Você pode aumentar o número de desenvolvedores de 4 para 6 por exemplo para conseguir uma velocidade maior. Esse é um tipo alavancagem focada na produção de indicadores, não na melhoria efetiva do seu sistema. Quando você faz isso, você certamente terá algum resultado, mas isso *não gerará nenhuma modificação no comportamento sistêmico do seu projeto*. Números não mudam sistemas, eles são apenas capazes de demonstrar tendências, facilitar análises e diagnósticos. Por isso, esse é o ponto de alavancagem número 12, o menos efetivo de todos. Outros exemplos: Demitir e recontratar esperando melhorar o sistema (o sistema não mudará com uma nova pessoa, a não ser que essa pessoa mude o sistema); Aumentar o número de horas gastos com testes para aumentar os níveis de qualidade; aumentar o salário da equipe técnica para aumentar a produtividade ou a motivação; definir um número mínimo de bugs aceitável por release. Essas medidas podem causar algum impacto a curto prazo, mas elas não induzem mudanças sistêmicas permanentes. Para Meadows, atuar sobre parâmetros numéricos é a forma menos efetiva de causar mudanças satisfatórias em sistemas.

Ponto 11 - Tamanho de buffers ou de inventário acumulado. Projetos ágeis mantém inventário acumulado fora do sistema até que ele realmente precise entrar para ser processado. Isso visa o aumento de eficiência, de flexibilidade e de potencial de adaptabilidade. Em projetos de software, inventário é qualquer demanda não implementada ou parcialmente implementada: demandas detalhadas e não implementadas, código não integrado, código não testado, requisitos não liberados para uso e outros. Grandes níveis de inventário aumentam a complexidade e os controles necessários para mantê-los e atualizá-los, além disso eles ocultam defeitos no produto e problemas no processo. Grandes níveis de inventário dificultam, assim, a geração de business value diminuindo a eficácia do sistema. Esses pontos de represamento existentes nos sistemas são também conhecidos como *buffers*. Trabalhar com buffers pequenos está na essência dos projetos ágeis. "Se um buffer é muito grande, o sistema se torna inflexível. Ele reage muito lentamente", diz Meadows. Mudar o tamanho dos buffers é um ótimo ponto para alavancagem em sistemas. Em projetos ágeis, há um buffer em cada iteração e outro para todas as outras demandas que ainda não entraram no sistema. Iterações maiores resultarão em tamanhos de buffers maiores. Assim, diminuir o tamanho de uma iteração pode ser um ponto de alavancagem a se considerar. A redução extrema de buffers ocorre quando a equipe resolve eliminar completamente as iterações estabelecendo um fluxo contínuo de entrada e saída de demandas. Mas ainda há muita polêmica sobre as vantagens e os perigos dessa abordagem.

Ponto 10 - A estrutura do fluxo de inventário. A estrutura pela qual o inventário é conduzido dentro do sistema é o ponto de alavancagem número 10. Em projetos ágeis, essa estrutura é definida pelo trabalho colaborativo sendo comum o uso de kanban's para otimizá-la. Os kanban's são instrumentos que permitem controlar o chamado *Work In Process*. Esse tipo de controle permite que a equipe monitore o que está acontecendo no dia-a-dia, detecte gargalos e atue rapidamente no caso de problemas. Ele sinaliza a movimentação do inventário desde que ele entra no sistema (planejamento do trabalho) até o momento em que ele sai (entrega). Projetos ágeis costuma utilizar esse recurso para otimizar essa movimentação e com isso influenciar positivamente o sistema.

Ponto 09 - Tamanho dos tempos de resposta. Um fator crítico ao atuar sobre um sistema é o tempo que se leva para perceber seu nível de discrepância depois que seu estado é alterado. Pense em um chuveiro. Se o tempo entre girar o registro de água e perceber o seu efeito por meio da mudança de temperatura for muito longo, será difícil e chato de se chegar a temperatura desejada. Em projetos ágeis, isso é tratado com o conceito de feedback, e é estimulado em vários níveis (técnicos e de gestão). O tempo de resposta para se avaliar a evolução do escopo é o de uma iteração. O tempo de resposta para se avaliar intervenções no processo (realizadas durantes as retrospectivas) também são de uma iteração. Enfim, no processo de gestão as iterações ditam os tempos de respostas. Mais um motivo para mantê-las curtas.

Ponto 08 - A Força dos ciclos de feedback negativo. Os ciclos de feedback negativo são comportamentos sistêmicos que atuam estabilizando o sistema quando este se afasta do seu estado desejado. Um exemplo simples é o sistema de regulação de temperatura do nosso corpo. Quando ele aquece ou esfria fora de faixas seguras, esse sistema detecta essa variação e aciona mecanismos para restabelecer a temperatura. Por isso suamos quando está muito calor e nossos pêlos ficam eriçados quando fica frio. O piloto automático de um carro é outro exemplo. Quando este detecta que está fora da velocidade configurada, ele é capaz de acelerar ou desacelerar o motor para que o carro se mantenha na velocidade desejada. Um ciclo de feedback negativo é, assim, composto de monitoramento, controle e ações de reforço. Em projetos ágeis, há vários ciclos de feedback negativo. Pense, por exemplo, no planejamento iterativo. É uma forma de feedback negativo, onde pode-se utilizar o aprendizado conquistado para agir restaurando o projeto para a direção correta. Um outro exemplo são as reuniões diárias. As próprias reuniões de retrospectiva atuam como ciclos de feedback negativo, pois são capazes de monitorar e ajustar as práticas utilizadas quando estas perdem efetividade. Integração Contínua e TDD são outros exemplos de uso desse conceito a favor dos projetos de software. Se uma build não tem qualidade, o monitor (build server) para o processo e dá o alerta. A equipe então se mobiliza para re-estabilizar o sistema para os níveis desejados (build com qualidade). Esse tipo de controle de estabilização, ou feedback negativo, são muito poderosos para alavancagem de eficácia dos sistema. Talvez seja graças ao imenso poder que práticas como Integração Contínua, TDD, Testes funcionais automatizados, planejamento iterativo, retrospectivas e daily meetings tem para manter o projeto estabilizado que as tornam tão populares e eficazes.

Ponto 07 - Dirigindo ciclos de feedback positivo. "Um ciclo de feedback negativo é auto-corretivo; um ciclo de feedback positivo é auto-reforçante". O que isso significa? Enquanto que o primeiro re-estabiliza um sistema que saiu do seu estado normal, o outro reforça cada vez mais o seu comportamento. Em uma epidemia, quanto mais gente fica doente, mais gente é capaz de transmitir a doença. Quanto mais problemas de qualidade você tem no seu produto, mais tempo você vai gastar resolvendo-os e menos tempo restará para investir em garantia de qualidade, o que aumentará ainda mais os problemas de qualidade. Quanto mais código com design ruim é produzido, mais difícil ficará para mantê-lo e mais difícil será evitar novos trechos de código com problema de design. Em resumo, o feedback positivo é caracterizado pela potencialização crescente do aparecimento de um determinado evento (bom ou ruim) pelo simples fato de ele existir. Um sistema pode implodir negativamente ou explodir positivamente por essa força sistêmica. Saber evitar a implosão e criar mecanismos de explosão certamente afetará e muito o potencial de alavancagem sistêmica do seu projeto.

Ponto 06 – A Estrutura do Fluxo de Informações. O exemplo de Donna Melow é bem ilustrativo: Imagine dois conjuntos de casas cujo consumo de energia é bem semelhante. Depois de certo tempo, coloca-se no primeiro deles o relógio de medição de energia bem na porta de entrada, enquanto que no outro conjunto ele continua instalado em local de difícil acesso. Você se espantaria se o primeiro conjunto de casas passasse a consumir 30% menos energia do que o segundo? Estamos falando de um tipo de mudança ambiental capaz de fazer com que as pessoas comecem a se comportar de modo diferente apenas porque um novo fluxo de informação foi criado. Acho que nem preciso dizer como um projeto ágil se beneficia desse ponto de alavancagem. Ambientes informativos tem exatamente esse propósito: gerar novos fluxos de informação e com isso influenciar o comportamento das pessoas.

Ponto 05 - As regras do sistema. Todo o sistema tem suas regras. Em Projetos Ágeis, as regras são formalizadas em termos de práticas. Equipes ágeis que avaliam e aperfeiçoam continuamente as práticas que utilizam estão, na verdade, criando oportunidades de alavancagem no sistema por meio da redefinição das suas regras. "Se você quer entender os sistemas, preste atenção às suas regras, e a quem tem o poder sobre elas".

Ponto 04 – O poder da auto-organização. Com o recente sucesso e crescimento do Scrum, muito se tem falado sobre times auto-organizáveis. A idéia é de que o time de projeto tem que ser

capaz de se auto-organizar para alcançar metas e vencer desafios. Ser capaz de se auto-organizar é realmente uma das mais poderosas formas para se manter a sustentabilidade e a evolução constante de um sistema. Mas não está relacionada apenas com a capacidade de organização para o cumprimento de objetivos. No sentido sistêmico, a auto-organização é uma força consciente, com regras próprias. Ela é capaz de procurar por cada um dos pontos de alavancagem descritos até aqui e aplicá-los sem a intervenção de uma força superior manipuladora. Um sistema auto-organizado consegue adaptar-se por si próprio, regular-se a si mesmo, responder a ciclos de feedback positivo e negativo sem um comando externo.

Auto-organização não é anarquia (Highsmith J. , 2008). Equipes de projetos ágeis procuram se auto-organizar em torno de uma liderança. Esta faz parte do sistema, não o comanda. É como um juiz de futebol que precisa deixar o jogo ser jogado sem aparecer muito. Quanto menos ele aparece, mais eficiente ele é.

Ponto 03 – As metas do sistema. Todo o sistema tem que ter um propósito, e se ele não estiver claro, o propósito do sistema será apenas manter-se a si mesmo. Projetos ágeis se preocupam muito com isso. A aderência ao chamado "business purpose" é fundamento de métodos como o DSDM e o Lean Software Development. A meta não é implementar especificações, mas produzir soluções aderentes ao propósito de negócio do cliente. No Scrum, metas claras precisam ser definidas para cada sprint de forma a manter o foco. Kent Beck também dá sua contribuição quando defende o uso de "Temas" para estabelecer constância de propósito e foco nos objetivos de negócio. De fato, o grande diferencial nas abordagens ágeis é ser enfático em sugerir que as metas do projeto estejam claras e orientadas ao valor de negócio do cliente. Quando um propósito definitivamente se estabelece em um sistema, este é todo contaminado pela necessidade de atingi-la. O potencial de alavancagem, nesse cenário, será imenso.

Ponto 02 – O paradigma sobre o qual o sistema se apoia. Os paradigmas são pré-concepções sobre determinado assunto. Quando essas pré-concepções são desafiadas a ponto de se estabelecer uma nova abordagem para lidar com o assunto, um graden potencial de alavancagem poderá ser criado. É o que acontece, por exemplo, quando se substitui o paradigma de gestão, ou o paradigma de qualidade, ou ainda paradigmas de arquitetura, de design ou paradigmas metodológicos.

Ponto 01 – O poder de transcender os paradigmas. Existe ainda um último ponto de alavancagem ainda mais intenso do que a mudança de paradigma propriamente dita. É conseguir se manter desacoplado dos paradigmas. Saber que nenhum paradigma é totalmente verdadeiro, e que há um imenso universo de potencialidades em todas as áreas. A maioria das grandes conquistas apareceram devido a capacidade de pessoas que souberam transcender os paradigmas vigentes e estabelecer suas próprias verdades.

Notas Finais

Os 12 pontos não são balas de prata, nem guias para influenciar projetos. São referências para nos ajudarem a entender porque estamos intervindo ou ajudando a intervir em nossos projetos.

Sistemas da vida real são complexos e estão sujeitos a inúmeras variáveis. Algumas delas nem conseguimos perceber sem muito estudo. No entanto, a diferença entre o caos e a emergência pode estar exatamente na nossa capacidade de perceber essas nuâncias e trabalhar a favor delas.

Referências

AgileAlliance. (13 de Fev de 2001). *Agile Software Development Manifesto*. Fonte: http://www.agilemanifesto.org

DSDMConsortium. (2007). Acesso em 18 de 11 de 2007, disponível em DSDM.org: http://www.dsdm.org

Highsmith, J. (2000). *Adaptive Software Development.* Dorset House Publishing.

Highsmith, J. (13 de Set de 2008). *No More Self-Organizing Teams*. Acesso em 05 de Jan de 2008, disponível em The Cutter Blog: http://blog.cutter.com/2007/09/13/no-more-self-organizing-teams/

Meadows, D. (1999). Fonte: Sustainability Institute: http://www.sustainabilityinstitute.org/pubs/Leverage_Points.pdf

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Addison Wesley.

Senge, P. (1990). *The Fifth Discipline.* New York: Doubleday.

Sobre o Autor

Alisson Vale tem mais de 15 anos de experiência com desenvolvimento de software e a mais de 6 anos lidera projetos de software dos mais variados tipos e tamanhos. Hoje é Líder de Projeto e Diretor da Phidelis Tecnologia, onde lidera a equipe de desenvolvimento da solução acadêmica oferecida pela empresa.

E-mail : alisson.vale@phidelis.com.br
http://www.phidelis.com.br/blogs/alissonv