Universidade Federal de Pernambuco
Centro de Informática
Departamento de Sistemas de Computação

Graduação em Ciência da Computação

O *TOYOTA WAY* E O DESENVOLVIMENTO ÁGIL DE SOFTWARE

Ricardo de Oliveira Cavalcanti

TRABALHO DE GRADUAÇÃO

Recife

2006

Universidade Federal de Pernambuco
Centro de Informática
Departamento de Sistemas de Computação

Ricardo de Oliveira Cavalcanti

O *TOYOTA WAY* E O DESENVOLVIMENTO ÁGIL DE SOFTWARE

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Departamento de Sistemas de Computação da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Prof. Dr. Alexandre Marcos Lins de Vasconcelos*

Recife

2006

À minha família

AGRADECIMENTOS

Valeu a pena? Tudo vale a pena
se a alma não é pequena.
Quem quere passar além do Bojador
tem que passar além da dor.
Deus, ao mar o perigo e o abysmo deu
mas nelle é que espalhou o céu.
FERNANDO PESSOA

A minha família, pelo incentivo e pelas oportunidades dadas ao longo da minha vida. A ela também dedico este trabalho que representa a conclusão desta jornada de crescimento acadêmico e pessoal. Agradeço especialmente a minha mãe, além de tudo, pela ajuda na revisão do trabalho.

Aos colegas, Martin e Nadine, pelo apoio em Hamburgo.

Aos velhos amigos e aos novos que encontrei durante esta jornada.

Ao meu orientador, Alexandre Vasconcelos, pela confiança e apoio na realização deste trabalho.

Aos professores do Centro de Informática, pela contribuição dada na minha formação.

A todos que contribuíram de alguma forma para elaboração deste trabalho.

I have no special talent. I am only passionately curious.
—ALBERT EINSTEIN

RESUMO

Em "*The Toyota Way - 14 management principles from the world's greatest manufacturer*", Jeffrey K. Liker descreve 14 princípios de gerenciamento, os quais, segundo o autor, podem ser aplicados a qualquer organização e qualquer processo de negócio, seja no ramo de serviços ou industrial. A abordagem da Toyota, à primeira vista, se assemelha à das metodologias ágeis de desenvolvimento de software no tocante à valorização das pessoas e à elevada capacidade de adaptação e reação. A proposta deste trabalho é examinar os 14 princípios do *Toyota Way* e verificar se, e como, seria possível aplicá-los em projetos de desenvolvimento de software. Cabe, portanto, a comparação entre as metodologias ágeis de desenvolvimento de software e os princípios descritos por Liker.

Palavras-chave: Processos de Software, Metodologias Ágeis, Pensamento *Lean*, *Extreme Programming*.

ABSTRACT

In "*The Toyota Way - 14 management principles from the world's greatest manufacturer*", Jeffrey K. Liker describes 14 management principles. According to the author, these principles can be applied to any business process, in service or in manufacturing. When it comes to people empowerment and the abilities of adaptation and reaction, Toyota's approach is very similar to agile methods of software development. This work intends to examine the 14 *Toyota Way*'s principles and verify if and how would it be possible to apply them to software development projects. Therefore takes place a comparative analysis between the agile software methodologies and the principles described by Liker.

Keywords: Software processes, Agile Methodologies, *Lean* Thinking, *Extreme Programming*.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

C3	*Chrysler Comprehensive Compensation*
GM	*General Motors Corporation*
TPS	*Toyota Production System* (Sistema Toyota de Produção)
TQM	*Total Quality Management*
XP	*Extreme Programming*

SUMÁRIO

Capítulo 1
INTRODUÇÃO

Na década de 1980, as empresas automobilísticas japonesas surpreenderam o mundo ao começarem a produzir veículos mais duráveis e robustos que os americanos. Nos anos 90, a Toyota, em especial, chamou a atenção com sua maneira peculiar de desenvolver e produzir carros. Ela fabricava automóveis de forma mais rápida e tinha maior credibilidade no mercado que seus concorrentes. Apesar de pagar os salários relativamente altos dos trabalhadores nipônicos, mantinha preços competitivos (LIKER, 2003). Atualmente, a Toyota é a segunda maior indústria automobilística do mundo, atrás apenas da *General Motors* (GM), com mais de seis milhões de vendas por ano em 170 países do mundo (TIERNEY, 2004; ORGANISATION INTERNATIONALE DES CONSTRUCTEURS D'AUTOMOBILES-OICA, 2005).

O impressionante desempenho da principal representante do movimento *Lean* é resultado direto da excelência operacional, a qual é baseada em ferramentas e métodos bastante conhecidos na indústria como, por exemplo, *Flow*, *Kaisen*, *Jidoka* e *Heijunka*. Apenas técnicas e ferramentas, no entanto, não são capazes de transformar empresas. O sistema de produção fundado por Taiichi Ohno, é complementado por algo de único na Toyota: os princípios que regem o gerenciamento na empresa. Enfim, o sucesso da companhia está intimamente ligado à sua habilidade de cultivar lideranças, times e uma cultura organizacional, de planejar estratégias, de construir relações estreitas e duradouras com fornecedores e, principalmente, de manter uma organização capaz de aprender, como fazia o próprio Ohno em seus passeios no chão da fábrica (LIKER, 2003).

Este conjunto de princípios é descrito por Liker (2003). Segundo o autor, os princípios de gerenciamento do *Toyota Way* podem ser aplicados a qualquer organização e qualquer processo de negócio, seja no ramo de serviços ou industrial.

Desde o fim da década de 1960, a indústria do software, por sua vez, procura tornar o seu desenvolvimento mais previsível e eficiente (NAUR; RANDELL, 1969). Na tentativa de organizar as atividades de desenvolvimento de software, surgiram as primeiras metodologias, as quais impunham processos detalhados e prescritivos, baseados fortemente em planejamento e documentação extensivos, como em outras engenharias. Estas metodologias, no entanto não

se mostraram bem sucedidas, uma vez que os problemas de estouro de prazo e orçamento nos projetos continuaram a ocorrer (STANDISH GROUP INTERNATIONAL, 2001).

Em resposta ao insucesso das primeiras metodologias de desenvolvimento de software, surgiram, na década de 1990, diversas metodologias denominadas, posteriormente "metodologias ágeis". Com ênfase em colaboração entre o time de programação e os especialistas de negócios, comunicação face a face (sendo mais eficiente do que documentação escrita), times enxutos e auto-organizados entre outras características (AGILE ALLIANCE, 2005), elas propõem um meio termo entre "nenhum processo" e "processo demais", de forma que o desenvolvimento do software não seja obstaculizado pela metodologia. A primeira impressão, no entanto, de que as metodologias ágeis de desenvolvimento simplesmente se opõem àquela burocrática, porque geram menos documentos, portanto consideradas "leves", não toca o cerne dos métodos ágeis. Duas das principais idéias por trás dos métodos ágeis, as quais os diferenciam das demais metodologias são:

1. Métodos ágeis são mais adaptativos do que preditivos. Métodos tradicionais procuram planejar em detalhe longos períodos, desta forma é da sua natureza a resistência à mudança uma vez que o planejamento esteja estabelecido. Métodos ágeis, por sua vez, acolhem a mudança a qualquer momento, ao ponto de adaptar a própria metodologia para serem bem sucedidos (FOWLER, 2005).
2. Métodos ágeis baseiam-se mais nas pessoas do que nos processos. Enquanto os métodos tradicionais procuram definir processos os quais vão funcionar com qualquer um que os execute, os métodos ágeis asseguram que nenhum processo pode superar as habilidades de um time. Desta forma, o papel do processo é dar suporte à equipe de desenvolvimento em seu trabalho (FOWLER, 2005).

As práticas aplicadas num domínio específico não irão necessariamente ser aplicáveis em outros domínios. Os Princípios, no entanto, podem ser aplicados desde que sejam traduzidos em práticas específicas para cada domínio (POPPENDIECK & POPPENDIECK, 2003). O desenvolvimento de software é uma disciplina bastante abrangente e pode ser utilizada em diferentes domínios – de criar portais de *e-commerce* a pilotar robôs remotamente. Da mesma forma que a Toyota revolucionou a indústria automobilística com seus princípios e ferramentas, as metodologias ágeis de desenvolvimento de software trazem sugestões inovadoras.

Este trabalho tem como objetivo examinar os 14 princípios do *Toyota Way* e verificar se, e como, é possível aplicá-los em projetos de desenvolvimento ágil de software. A importância de examiná-los está no fato de que somos, todos, gerentes – mesmo que em alguns processos, sejamos gerentes de nossas próprias ações. (VAN CAUWENBERGHE, 2005).

À primeira vista, as abordagens se assemelham no tocante à valorização das pessoas e à elevada capacidade de adaptação e reação. Cabe, portanto, a comparação entre as metodologias ágeis de desenvolvimento de software e os princípios descritos por Liker (2003), como forma de demonstrar a aplicabilidade dos citados princípios no desenvolvimento ágil de software. Especificamente, será analisada a *Extreme Programming* (XP), a mais popular das metodologias ágeis.

Um trabalho na mesma linha de pesquisa foi apresentado por Van Cauwenberghe (2005), e Moryen (2005).

1.1 ESTRUTURA DO TRABALHO

Além deste capítulo introdutório, este trabalho está organizado da seguinte maneira: o capítulo 2 descreve os 14 princípios de gerenciamento da Toyota, e as ferramentas e técnicas utilizadas pela empresa, na fábrica ou no setor de serviços; o capítulo 3 apresenta os valores e práticas que constituem a *Extreme Programming*; o capítulo 4 analisa a metodologia ágil à luz dos princípios previamente descritos, mostrando como suas práticas exercitam os princípios da Toyota, e sob quais aspectos pode ser incompatível com a cultura organizacional estabelecida na empresa japonesa; finalmente, o capítulo 5 tece as conclusões e possibilidades de trabalhos futuros.

Capítulo 2
O *TOYOTA WAY*

The Toyota Way *and the Toyota Production System (Toyota's manufacturing method) are the double helix of Toyota's DNA; they define its management style and what is unique about the company.*
JEFFREY K. LIKER

A história da Toyota começa com Sakichi Toyoda, funileiro e inventor. O japonês iniciou sua vida fabricando teares e em 1926 fundou a *Toyoda Automatic Loom Works*, empresa mãe do grupo Toyota. A venda dos direitos de patente de uma das invenções de Toyoda resultou no capital necessário para a fundação da *Toyota Motor Corporation* em 1930. O componente principal do modelo cujos direitos foram vendidos era um mecanismo destinado a parar automaticamente o tear quando algum fio se partia. Importante, porém, é perceber que o mecanismo é a representação prática de um dos pilares do Sistema Toyota de Produção: o *jidoka*. Essencialmente, *jidoka* quer dizer que a qualidade deve ser embutida no processo, uma falha deve provocar a parada do processo para resolver a causa do problema. Juntamente com seis outros princípios, o *jidoka* compõe o grupo de princípios relativos ao processo. Ao todo, os 14 princípios de gerenciamento estão divididos em quatro grupos: Filosofia de Longo Prazo, Processo, Pessoas/Parceiros e Resolução de Problemas. Os grupos organizam-se numa pirâmide, como mostrado na Figura 2-1 (adaptada de LIKER, 2003, p. 6, il.). A estrutura faz alusão à melhor forma de adotá-los: da base para o topo. Cada um destes princípios será analisado neste capítulo, e seus enunciados, destacados dentro de cada seção.

Figura 2-1 Pirâmide do *Toyota Way*

2.1 FILOSOFIA DE LONGO PRAZO

Desenvolver uma filosofia de longo prazo, e fazer dela a base para todas as decisões e atividades exercidas na empresa é o fundamento de todos os princípios de gerenciamento descritos pelo *Toyota Way*. O primeiro princípio, único deste grupo, propõe **basear suas decisões em ideais de longo prazo, mesmo que ao dispêndio de metas financeiras de curto prazo**.

Gerar valor para o cliente, para a sociedade e para a economia é o início. A partir da declaração da Missão da empresa, todas as funções na organização estão alinhadas. A fim de ilustrar como a Toyota se diferencia de outras grandes empresas, confrontemos as declarações de missão da *General Motors Corporation* (2006) e da *Toyota Motor Manufacturing North America*, filial americana da empresa. Observemos a Figura 2-2.

Toyota Motor Manufacturing North America	*General Motors Co.*
1. Por ser uma empresa americana, contribuir para o crescimento econômico da comunidade e dos Estados Unidos. 2. Por ser uma empresa independente, contribuir para a estabilidade e bem estar dos membros da equipe. 3. Por fazer parte do Grupo Toyota, contribuir para o crescimento geral da Toyota gerando valor para nossos clientes.	1. Ser o líder mundial em produtos e serviços relacionados ao transporte. 2. Nós vamos conquistar o entusiasmo do cliente através da melhora contínua orientada pela integridade, trabalho em equipe e inovação dos que fazem parte da GM.

Figura 2-2 Missão da Toyota *versus* GM

Enquanto a missão da GM fala sobre ser líder de mercado e melhorar seus produtos e serviços, a Toyota não menciona a liderança de mercado, mesmo sendo a segunda maior companhia de veículos do planeta, tendo ultrapassado recentemente a Ford em volume de vendas (TIERNEY, 2004; OICA, 2005); nem fala sobre a qualidade dos seus produtos, mesmo sendo famosa pela busca contínua pela qualidade. A missão da Toyota, na verdade, se divide em três partes: o trato com os *stakeholders* externos (o país onde está situada), o trato com os *stakeholders* internos (os membros da equipe), e a contribuição para o crescimento geral da Toyota. A busca contínua pela qualidade e a geração de valor para os investidores são, na verdade, requisitos para conseguir executar a missão.

Isso, no entanto, não quer dizer que a empresa japonesa não se preocupa, por exemplo, com corte de gastos. Muito pelo contrário, ao fim da II Guerra Mundial, a companhia estava beirando a falência, e a busca liderada por Taiichi Ohno para eliminar desperdícios nos processos, esclarecida na próxima seção, também teve suas implicações no controle de custos. Ao invés de demitir funcionários, operários foram movidos para outras funções, para que não fosse mais necessário contratar outro funcionário para aquela função antiga.

Respeito, dedicação, inovação, persistência e colaboração são valores praticados diariamente na Toyota. Eles formam a base filosófica que não será ab*andon*ada, mesmo que um desastre na linha de produção desligue toda a fábrica. A única coisa que poderia modificar as diretrizes da companhia seria uma mudança tal que ameaçasse a sobrevida da empresa no longo prazo. Mesmo assim, uma transformação desse porte não seria posta em execução antes de uma profunda análise.

2.2 PROCESSO

A Toyota acredita que o processo correto irá levar ao resultado correto. Neste grupo encontram-se sete dos 14 princípios do *Toyota Way*. Para pô-los em prática, a empresa lança mão de várias das famosas técnicas e ferramentas do *Lean*, como o *flow*, o *jidoka*, o *kanban* e o *andon*.

Também é neste ponto que a maioria das empresas que procuram inserir o *Lean* em seu dia-a-dia permanece estagnada. Elas procuram melhorar seus processos através do uso das técnicas difundidas na Toyota, mas não conseguem alcançar o nível de excelência da companhia japonesa. Acontece que apesar de poderosas, as técnicas e ferramentas do *Just-In-Time* são apenas táticas e operacionais. O *Pensamento Lean* envolve uma mudança cultural muito mais profunda e ampla. O apoio de uma filosofia de longo prazo deve ser a base para sua implantação.

2.2.1 Fluxo Contínuo

Criar um fluxo de processo contínuo para trazer os problemas à superfície. Este é o enunciado do segundo princípio do *Toyota Way*. A idéia por trás é elaborar seus processos de negócio de forma a diminuir ao máximo o desperdício, obtendo, desta forma, um fluxo contínuo composto apenas de processos que acrescentam valor ao seu objeto de negócio. Nenhum trabalho em andamento pode ficar esperando por alguém que vá trabalhar nele. Materiais e informações devem fluir de forma rápida, assim, qualquer falha será logo percebida.

A criação de um fluxo contínuo nos processos de negócio irá forçar uma diminuição no estoque de matéria-prima ou de informação. Essa diminuição expõe ineficiências que

demandarão soluções imediatas, caso contrário todo o fluxo irá parar por mais tempo e o prejuízo será maior ainda. Quando os processos estão ligados uns aos outros, há mais trabalho em equipe, feedback mais rápido sobre os problemas, maior controle sobre o processo e pressão constante sobre as pessoas para resolver problemas, pensar e crescer.

A produção em grandes lotes não era compatível com a Toyota por causa do reduzido mercado japonês do pós-guerra. Taiichi Ohno não assumiu que o tamanho do lote deveria ser o mais eficiente para cada processo. Ao invés disso, ele organizou a fábrica em células de trabalho que eram agrupadas por produto, e não por processo. Ele pode, então, estabelecer quais processos acrescentavam valor ao produto de cada célula, e elevar a eficiência da fábrica.

Neste sentido, Ohno identificou os sete principais tipos de desperdício na indústria:

1. Superprodução;
2. Espera;
3. Transporte desnecessário (dos produtos e matérias primas);
4. Processamento exagerado ou incorreto;
5. Estoque excessivo;
6. Movimento desnecessário (dos trabalhadores) e
7. Defeitos.

Para ele, superprodução é a forma fundamental de desperdício, porque causava a maioria das outras perdas. Produzir mais do que o cliente encomendou, conduzirá a montanhas de estoque à espera do próximo processamento. Quando não há um fluxo que expõe as dificuldades, lotes inteiros de peças defeituosas podem ser jogados fora; ou a manutenção preventiva de uma máquina pode ser deixada de lado simplesmente porque sua parada não irá afetar direta e imediatamente a linha de produção.

Através da implantação do fluxo contínuo, alguns dos benefícios conquistados são os seguintes:

- Flexibilidade: dedicar um equipamento a uma linha de produção leva à perda em flexibilidade para alocá-lo para outros fins. Entretanto, se o tempo de confecção de um produto é curto, pode-se responder às mudanças e fazer aquilo que o cliente realmente deseja.

- Qualidade embutida no processo: é mais fácil embutir o controle de qualidade no processo. Cada operador vira um inspetor da qualidade e a causa do defeito pode ser rapidamente corrigida causando o mínimo desperdício.
- Maior produtividade: numa célula de trabalho em fluxo contínuo, há pouquíssima atividade que não acrescenta valor ao produto, como o movimento desnecessário dos trabalhadores. Pode-se facilmente determinar quem está ou não ocupado e, conseqüentemente é fácil calcular quantas pessoas são necessárias para alcançar uma determinada taxa de produção. Em todos os casos que a Toyota promoveu a mudança da linha de produção de seus fornecedores, foi registrada, no mínimo, 100% de melhora na produtividade.

2.2.2 Sistemas *Pull*

Em conjunto com o fluxo contínuo de produção, aparece o princípio de **utilizar sistemas *pull* para evitar superprodução**. Desta forma, um consumidor só deve receber o produto desejado, na quantidade desejada quando ele demandá-lo.

Em oposição, tradicionalmente, as fábricas trabalham com sistemas *push*: planos feitos com antecedência definem a agenda de produção, baseada, portanto, numa projeção da demanda. Caso a demanda diminua, dificilmente a fábrica conseguirá se adaptar e irá produzir em excesso. Quando se produz em massa, normalmente um departamento irá produzir o máximo possível antes que seja preciso realizar as mudanças necessárias para se gerar um outro tipo de produto. A fábrica é organizada por processo de produção, como explicado anteriormente. Entre departamentos, então, haverá grandes reservas de estoque para manter todos ocupados e produzindo. Assim, cada departamento pode obedecer a uma agenda independente para produzir o máximo possível.

Mas, se por um lado departamentos produzindo com base num sistema *push* irão produzir em demasia naturalmente, a presença do estoque pode facilitar a continuidade do fluxo de produção. Taiichi Ohno então decidiu criar pequenas operadoras de estoque entre determinadas atividades. Quando um consumidor retira determinados itens, eles são repostos. Itens não utilizados continuam no estoque e não são repostos. Desta forma, o estoque pode ser tão pequeno quanto uma prateleira, e há uma ligação direta entre a produção e a demanda. Como as fábricas são grandes e os fornecedores podem estar situados longe, criou-se na

Toyota uma maneira simples de sinalizar que determinados itens foram utilizados e necessitam reposição: *kanban*.

Kanban significa sinal. Pode ser uma caixa vazia que precisa voltar cheia com uma determinada quantidade de peças. Cartões e outros tipos de sinais também são utilizados para desencadear a produção e entrega de peças. A simplicidade do *kanban* não deve ser subestimada. A eficiência e riqueza visual da técnica fazem com que o estoque diminua, e cresça a satisfação do consumidor, em geral interno à fábrica. Além disso, seu uso possibilita, ou até força, o desenvolvimento de regras para coordenar e executar a reposição de peças baseada nos dados mais atuais sobre o consumo.

A utilização de uma ferramenta simples como o *kanban* pode ser comparada com situações diárias de reposição baseadas em sistemas *pull*. Ao encher o tanque do carro, por exemplo, dificilmente há uma agenda a seguir, por exemplo, encher o tanque toda terça-feira. Caso haja tal regularidade, fatalmente haverá dias nos quais não seria necessário completar o tanque e outros em que o tanque estaria vazio, se não tivesse sido abastecido no domingo anterior.

Nem tudo pode ser reposto através de sistemas *pull*. Alguns produtos não são necessários imediatamente, e podem seguir uma agenda fixada previamente. Entretanto, imagine se o carro citado acima não tivesse um medidor para o tanque de combustível; seria necessário obedecer a uma agenda. Talvez até por precaução, parar-se-ia no posto duas vezes por semana. É importante perceber que uma situação crítica como a do tanque de combustível levaria, com certeza, ao desenvolvimento de um mecanismo que substituísse o cronograma pelo alerta da necessidade de reposição. Isso ocorre frequentemente na Toyota.

É importante mencionar que a Toyota sabe da importância de sistemas de reposição baseados em planos e agendas e também os utiliza em diversas ocasiões. Para os japoneses, no entanto, em primeiro lugar está o fluxo contínuo de produção. Quando não é possível estabelecê-lo, utilizam-se sistemas *pull* para evitar a superprodução. Quando esses não são viáveis, planos e *deadlines* entram em jogo.

2.2.3 Ritmo Sustentável (*heijunka*)

O desperdício representa apenas um dos males que devem ser eliminados para alcançar o sucesso. Na realidade, o foco exclusivo em eliminar o desperdício pode até

prejudicar a produtividade. Os outros dois grandes males da produção são sobrecarga e incerteza.

A sobrecarga, obviamente, compromete a segurança dos trabalhadores e a qualidade da produção. A incerteza na produção é o reflexo de planos irregulares e flutuações no volume de produção devido a problemas internos, como defeitos ou falta de peças. Como resultado direto da incerteza vem o desperdício. Para manter uma produção incerta é preciso manter material, equipamento e pessoas prontas para produzir em nível máximo, mesmo que a necessidade média esteja abaixo deste nível.

Pensando nesses males, o quarto princípio estabelece **regularizar a carga de trabalho**.

Normalmente, as empresas que tentam introduzir o *Lean* em seu dia-a-dia concentram-se em estabelecer um fluxo contínuo, com pouco estoque, onde os trabalhadores dão tudo de si para o sistema ser o mais eficiente possível. A facilidade em identificar e eliminar o desperdício cria um falso fluxo contínuo que é irregular, no qual os trabalhadores e as máquinas terminam sobrecarregados. Após alguns picos de melhoras na produtividade, tudo sucumbe.

A maioria das empresas falha porque não consegue estabelecer um fluxo regular e balanceado de produção. Estabelecer um ritmo sustentável, ou *heijunka*, é essencial para eliminar a incerteza, o que, por sua vez, é essencial para eliminar o desperdício e a sobrecarga.

Para estabelecer um ritmo sustentável, regular e balanceado, a demanda do consumidor é analisada e a produção é definida. Problemas que impeçam o balanceamento da produção, como a dificuldade de fabricar diferentes peças, por exemplo, geralmente podem ser contornados. Depois de estabelecido o *heijunka*, os seguintes benefícios podem ser observados: maior flexibilidade para produzir de acordo com a demanda do consumidor, menos risco de ter itens não consumidos, trabalhadores e equipamentos utilizados de forma balanceada, além de regularidade também para os fornecedores.

O trabalho regular e padronizado é mais simples, barato e rápido de gerenciar. Torna-se mais fácil observar desperdícios e defeitos. Sem regularidade é praticamente impossível estabelecer tarefas padronizadas e melhoria contínua. O desperdício aumenta naturalmente, quando as pessoas e máquinas têm de ora trabalhar em demasia e ora parar e esperar.

2.2.4 Qualidade Embutida (*jidoka*)

Parar a linha de montagem pode ser considerado o pior pecado para as empresas automotivas tradicionais. Para a Toyota, no entanto, é dever dos líderes de grupo interromper a linha de montagem caso necessário. Só desta forma a melhoria contínua pode ser alcançada.

Como já explicado brevemente, é chamado de *jidoka* o conceito de qualidade embutida no processo. O conceito também se refere à "autonomação" (autonomia + automação). A idéia de que o equipamento é capaz de reconhecer comportamento anormal em si mesmo e parar. É muito mais efetivo e barato prevenir problemas os quais podem se estender por toda a linha do que inspecionar e consertá-los depois do ocorrido.

A Toyota defende **criar uma cultura de parar para corrigir problemas**. Quando se tem cotas baixas de estoque, não há muitas chances para errar. Caso uma seção da linha de montagem seja cessada, a seção seguinte irá fatalmente parar. Há então um sistema de sinais e alarmes, denominado *andon*, que alerta aos líderes a parada de equipamentos ou a necessidade de ajuda para resolver um problema. Quando alertados sobre os problemas, os líderes de equipe têm um tempo hábil para ponderar sobre a necessidade de parar aquele segmento pelo qual ele é responsável. Na realidade, raramente uma linha de montagem inteira para. As fábricas são divididas em segmentos com reservas de carros entre eles. Por causa das reservas, um segmento pode continuar trabalhando cerca de 10 minutos antes de interromper seu funcionamento por causa da parada de um segmento anterior.

Em oposição às grandes indústrias de produção em massa, a Toyota não funciona sempre com 100% da capacidade. Acredita-se que se a linha de montagem nunca para, é porque não há problemas; mas como toda fábrica tem problemas, eles devem estar encobertos. Os japoneses perceberam que deter a linha e solucionar a causa do problema é muito mais rentável no fim das contas, porque trará ganhos na qualidade do produto e do processo.

2.2.5 Atividades Padronizadas

Padronização é a chave para a melhoria contínua e a qualidade. As idéias de administração científica de Taylor afastam-se da questão da alienação do trabalho a partir do momento no qual a criação dos padrões torna-se um esforço conjunto do capataz e dos trabalhadores sob sua chefia. A abordagem da Toyota à padronização foi parcialmente

influenciada pelas idéias de Henry Ford (1988 apud LIKER, 2003, p. 141, tradução nossa), escritas em 1926:

> A padronização de hoje, ao invés de ser uma barreira contra o aprimoramento é o alicerce necessário sobre o qual o aprimoramento de amanhã será baseado.
> Se você pensar em 'padronização' como o que você conhece de melhor hoje, mas que será melhorado amanhã – você chegará a algum lugar. Mas se você pensar em padrões como confinantes, então o progresso pára.

Contudo, a própria Ford, primeira gigante da indústria automotiva a perceber o poder da padronização na linha de montagem, tornou-se rígida e burocrática, seguindo as idéias de Taylor de maneira inadequada: obtendo medidas de tempo e produtividade, desenvolvendo tarefas padronizadas independentemente dos trabalhadores e impondo os padrões aos profissionais (LIKER, 2003). A Toyota alia alta padronização e burocracia a um modelo gerencial permissivo, o qual integra os trabalhadores na construção do processo. Esta mistura possibilita liberdade para inovar e ser criativo ao mesmo tempo em que mantém a rigidez dos processos.

É extremamente importante envolver os trabalhadores na criação e no desenvolvimento de padrões. O poder dado aos trabalhadores é refletido no comprometimento deles com a qualidade.

Enquanto tarefas repetitivas são descritas de forma bastante específica, descrições mais maleáveis podem ser usadas em outras funções. Além disso, é importante que a própria pessoa que realiza a tarefa se empenhe para melhorar o padrão. Por diversas razões: primeiramente, não há como manter engenheiros em toda a fábrica analisando e reescrevendo padrões; adicionalmente, regras impostas podem ser facilmente vistas como coercivas, ao passo que a possibilidade de contribuir funciona como uma alavanca motivacional. Define-se, destarte, o sexto princípio do *Toyota Way*: **o uso de tarefas padronizadas é a base para a melhoria contínua e o empoderamento das pessoas**.

2.2.6 Controle Visual

Dispositivos que indicam de imediato como a tarefa deve ser feita ou se há algo de errado são largamente utilizados não apenas na linha de produção. Eles podem mostrar onde devem ser guardadas ferramentas, qual o procedimento para executar uma tarefa ou qual o progresso da tarefa em andamento. De uma forma mais geral, o controle visual é utilizado

para divulgar informações de todo tipo para garantir que as operações e processos sejam executados de forma correta. Como semáforos nas ruas, que regularizam o trânsito.

São várias as ferramentas utilizadas pela Toyota para por em prática o princípio de **usar controle visual para nenhum problema ficar escondido**. *Andon*, *kanban* e até a própria célula de trabalho em fluxo contínuo e o trabalho padronizado são exemplos de controle visual. O trabalho padronizado é publicado, para ficar claro qual a melhor forma conhecida de se realizar uma determinada tarefa. Uma célula de trabalho irá mostrar imediatamente como está o andamento da produção, em comparação ao padrão estabelecido do progresso da tarefa.

2.2.7 Tecnologia a Serviço das Pessoas

Por incrível que possa parecer, a Toyota é considerada lenta e conservadora quando o assunto é adquirir nova tecnologia. Isso ocorre porque antes de ser incorporada, uma nova tecnologia deve passar pelo rigoroso teste de ser aderente às pessoas, ao processo e aos valores da empresa. O receio de adotar novas tecnologias se deve ao fato de que geralmente elas não são confiáveis, e por isso podem por em risco o fluxo contínuo. Desta forma, processos mais tradicionais, mas de eficácia comprovada, vão ser utilizados até que a nova tecnologia seja rigorosamente testada.

O oitavo princípio de gerenciamento descrito diz: **use apenas tecnologia confiável e largamente testada que sirva a seu pessoal e a seus processos**. Geralmente, a introdução de novas técnicas irá ser precedida de um projeto piloto na tentativa de melhorar o processo mantendo tecnologia, equipamentos e pessoas. Quando, e se, for estabelecido que não se possa mais aperfeiçoar o processo atual, surge o inquérito para decidir se a novidade trará melhorias. Ela deve estar em conformidade com os princípios vigentes:

- A valorização das pessoas em detrimento da tecnologia;
- A tomada decisão em consenso; e,
- O foco na eliminação do desperdício.

Caso não haja conformidade com a cultura organizacional, ela pode atrapalhar a estabilidade, fidelidade e a previsibilidade do processo e deve ser, portanto, modificada ou até rejeitada. Uma vez sabatinada e aprovada, a nova tecnologia deve ser implantada rapidamente.

Muitas vezes, ferramentas simples e manuais como um quadro negro, por exemplo, são utilizadas no lugar de modernos sistemas de informação. Em primeiro lugar, porque trazer o trabalhador para frente da tela do computador irá isolá-lo, retirá-lo do trabalho em equipe e afastá-lo do trabalho real que está sendo executado. Como explicado anteriormente, todos os esforços são voltados para a criação de controles visuais para o processo. Além disso, o corte de custos através da introdução de sistemas de informação pode trazer impactos profundos à cultura organizacional, caso os sistemas não apóiem as pessoas e os processos.

2.3 PESSOAS E PARCEIROS

O caminho para se tornar e manter-se uma organização bem sucedida passa, obrigatoriamente, pelas pessoas que fazem parte do empreendimento. Esta seção expõe como o trabalho em equipe é aquilatado dentro da Toyota. Para criar equipes que excedem os parâmetros de desempenho, é preciso, além de indivíduos com excelente formação técnica, líderes que sirvam de exemplo técnica e filosoficamente para seus times.

O espírito do Sistema de Produção da Toyota não se atém aos empreendimentos da gigante automobilística. Os fornecedores e companhias parceiras também devem compartilhar dos mesmos valores e princípios. Também é propósito desta seção descrever como a Toyota lida com seus parceiros.

2.3.1 Líderes

Ao contrário de outras empresas, a Toyota prefere criar e amadurecer seus líderes em casa, ao invés de contratá-los de outras empresas. A grande razão: um líder deve ser o espelho da cultura organizacional da empresa.

Primeiramente, é esperado de um líder que ele ensine a seus subordinados a cultura da empresa. Ele precisa ser um modelo para os outros trabalhadores, para isso tem que saber como o trabalho é realizado em cada parte da fábrica que está sob seu comando. O entendimento sobre as tarefas, inclusive as mais básicas, também está relacionado ao *genchi genbutsu*. Como será explicado posteriormente, o *genchi genbutsu*, a habilidade de observar e entender profundamente uma situação, é um dos pilares da cultura Toyota. Nesta cultura, uma impressão superficial da situação implicará decisões e aprendizado não efetivos.

Além disso, os esforços feitos pelos líderes para criar uma organização que aprende não podem ser desperdiçados trazendo um estranho. A introdução de um líder adventício, com certeza, irá abalar a lealdade e proximidade dos trabalhadores em relação a ele. Dessa forma, menores são as chances do novo líder promover as mudanças necessárias.

Para caracterizar melhor o líder dentro da Toyota, observemos a Figura 2-3 (adaptada de LIKER, 2003, p. 181, il.). Pode-se perceber no eixo vertical uma forma mais diretiva de liderança, na qual os líderes procuram dar ordens, e a forma denominada *Bottom-up*, na qual o líder se envolve e procura desenvolver as pessoas para que elas possam tomar decisões independentemente. O eixo horizontal diz respeito à profundidade de conhecimento do líder sobre o trabalho a ser realizado.

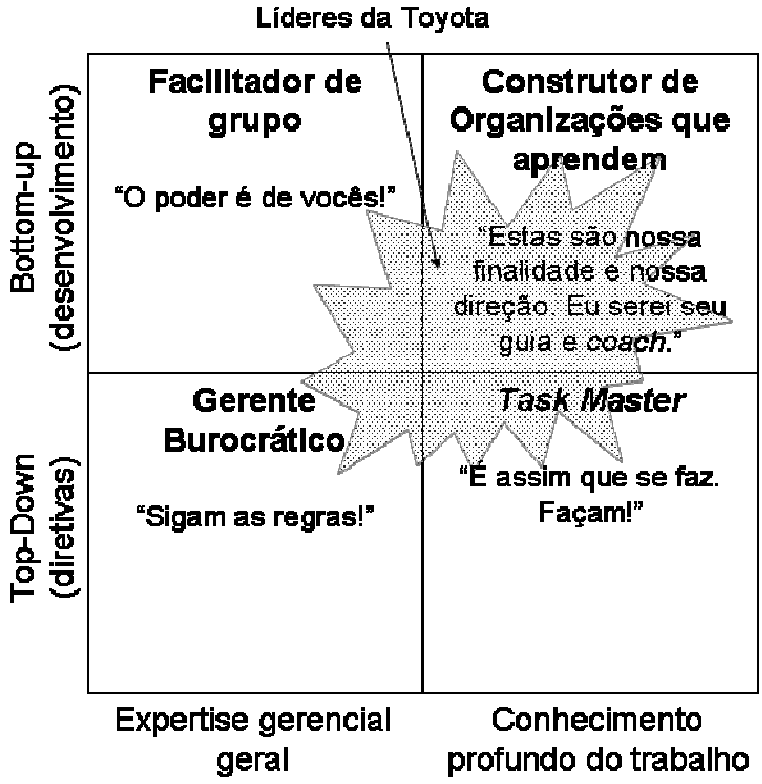

Figura 2-3 Matriz de Liderança

De acordo com o modelo, o gerente menos eficaz é aquele que procura controlar e comandar sem deter o conhecimento pleno de como as tarefas serão realizadas. Sua única chance de sucesso é estabelecer regras e políticas, além de formas de medir o desempenho de sua execução. O foco em satisfazer o cliente e criar uma organização que evolui será desvirtuado através de uma liderança baseada em números.

Um gerente com grande conhecimento sobre o trabalho, mas sem grandes habilidades sociais torna-se um *Task Master*. Ele dá ordens de forma que todos executem suas tarefas da maneira correta e na ordem apropriada. O problema é que uma pequena falha pode comprometer todo o trabalho. Um líder com estas características, provavelmente, não confiará em novatos.

O terceiro tipo de gerente é chamado de Facilitador de Grupo. Ele possui grandes habilidades motivacionais e fará o grupo se desenvolver e trabalhar junto. Ele funciona como grande catalisador, mas não é capaz de ensinar novos recrutas sobre o conteúdo do trabalho. Na realidade, sua falta de experiência na atividade em andamento também o torna incapaz de julgar realizações excepcionais dos subordinados.

Os líderes da Toyota têm como característica dominante a combinação entre habilidades de auxiliar, ensinar, motivar e liderar pessoas e o respeito conquistado através do profundo conhecimento técnico possuído. Eles têm o papel de *coach*, lideram o grupo e são profundos conhecedores dos processos e da cultura organizacional. Além disso, como mostra a área em destaque na figura, os líderes também assumem papéis mais burocráticos e autoritários, mas sua principal função é de possibilitar a evolução de toda a organização através do aprendizado contínuo.

Ao invés de contratar lideranças para resolver problemas financeiros ou decidir em situações isoladas, como esperado, os líderes impregnados dos princípios do *Toyota Way* devem ter sempre em mente os objetivos de longo prazo, além da habilidade de desenvolver nos subordinados a capacidade de entender como realizar o trabalho com excelência para ainda poderem aprimorar a maneira como realizam suas tarefas. **Cultivar líderes que entendam o trabalho, vivam a filosofia e ensinem-na aos outros**, como diz o nono princípio de gerenciamento, é uma das bases para o sucesso em longo prazo.

2.3.2 Pessoas e Times

O décimo princípio da filosofia Toyota trata do desenvolvimento das pessoas e dos times dentro da empresa. O **desenvolvimento de pessoas e times excepcionais que seguem a filosofia da empresa** é realizado de acordo com diversas teorias motivacionais, convenientemente adaptadas aos princípios e práticas da companhia. Por exemplo, segundo a Administração Científica de Taylor, as pessoas vão ao trabalho para ganhar dinheiro. Para

motivar os trabalhadores é preciso estabelecer padrões e medidas, ensinar-lhes a melhor forma de realizar uma tarefa, e dar gratificações caso ele consiga exceder as metas. Para Taylor, as metas eram sempre de quantidade, e não de qualidade.

A aplicação japonesa desta teoria motivacional envolve o estabelecimento de padrões, como já explicado anteriormente, porém, distancia-se das idéias Tayloristas porque os próprios trabalhadores são responsáveis pela melhora dos processos; bonificações são baseadas no desempenho do time, não de cada indivíduo; além da conhecida preocupação da Toyota com a qualidade.

O estabelecimento de metas a partir do topo da hierarquia aliado ao controle visual e à medição constante também é um bom exemplo de como a Toyota motiva seus funcionários. O controle visual e as medições constantes informam o trabalhador sobre seu progresso em relação a uma meta. Seguindo o *hoshin kanri*, como é conhecida a metodologia de desdobramento de políticas, as metas são estabelecidas nos níveis mais altos do organograma da empresa e desdobradas para os níveis mais baixos sempre refletindo as necessidades do alto escalão em forma de objetivos mais específicos.

A imersão dos colaboradores na cultura organizacional faz com que assumam o compromisso não apenas de seguir como também de passar adiante a filosofia que rege a corporação.

O décimo princípio também abrange o equilíbrio entre o trabalho individual e o trabalho em grupo. A Toyota acredita que "ao fazer do trabalho em equipe a base da companhia, cada indivíduo que faz parte do time irá dar seu máximo para que a empresa alcance o sucesso" (LIKER, 2003, p. 186, tradução nossa). Por isso, todas as ferramentas e técnicas são utilizadas para valorizar o trabalho em equipe. A sinergia, principalmente de times multidisciplinares, aumenta a motivação e o aprendizado de cada indivíduo além de refletir diretamente na qualidade e produtividade.

Por outro lado, quando observamos quão detalhado e demorado é o processo seletivo da Toyota, percebemos que, mesmo acreditando que "o todo pode ser maior que a soma das partes", não é ignorado o fato de que a excelência individual do trabalhador é necessária para o time ultrapassar os limites de qualidade e produtividade. A preocupação com o desenvolvimento técnico de cada colaborador é notável.

2.3.3 Respeito aos parceiros e fornecedores

Tão importante é a filosofia e como ela rege os negócios, que seus reflexos podem ser observados inclusive nos parceiros e fornecedores da empresa. Faz parte da maneira de gerenciar da gigante japonesa escolher rigorosamente seus fornecedores, buscando parcerias sólidas que possibilitem mútuos benefícios. É imprescindível que eles compartilhem da filosofia e cultura organizacionais pelo menos compatíveis, ou que estejam dispostos a ingressar no mundo *Lean* para serem parceiros da Toyota.

Depois de escolhido um fornecedor, todos os esforços são feitos para que a ligação seja duradoura. Bons resultados obtidos durante a relação contribuem para a construção de um vínculo baseado em respeito e confiança. Não faz parte do espírito do Sistema de Produção Toyota escolher um concorrente de um parceiro já estabelecido por causa de ninharias percentuais no preço dos produtos. Isso não significa, contudo, que a austeridade da contratante diminua após a introdução de um novo fornecedor à cadeia de suprimentos. Pelo contrário, o **respeito pelos parceiros e fornecedores é traduzido em desafios**. A Toyota sempre estabelece metas ousadas da mesma forma que age com os próprios trabalhadores. Como se fizessem parte da corporação, os parceiros são auxiliados para alcançar os objetivos estipulados. Para as empresas fornecedoras, fazer parte da cadeia de suprimentos da Toyota é certeza de novos desafios, mas também de melhoras nos negócios, inclusive com outros clientes.

O ponto crucial da abordagem Toyota para o gerenciamento da cadeia de fornecimento é a simbiose entre a empresa japonesa e seus parceiros. Essa interação possibilita o aprendizado e o crescimento mútuo e redunda em uma organização que aprende, que ultrapassa os limites tradicionais.

2.4 RESOLUÇÃO DE PROBLEMAS

A seguir, os três últimos princípios que regem o gerenciamento dentro das instalações da Toyota. Nesta seção será descrito como a empresa japonesa identifica problemas através do *genchi genbutsu*. Além disso, será explicado o princípio que governa a tomada de decisão, o *nemawashi*, e, finalmente, a importância da reflexão e o famoso *kaizen*, o processo de melhoria contínua.

2.4.1 Veja com seus Próprios Olhos (*genchi genbutsu*)

O mentor do Sistema de Produção Toyota, Taiichi Ohno, costumava realizar visitas diárias à linha de montagem para checar o andamento das operações e era capaz de identificar problemas simplesmente observando a situação. Apesar de gerente, por conseguinte sempre munido de relatórios sobre os resultados das operações, Ohno sabia que o aprendizado adquirido em seus passeios pela fábrica não poderia ser conquistado através de números. As tabelas e relatórios são importantes, pois informam os resultados, mas não se pode ter noção do que ocorre no dia-a-dia da fábrica apenas com eles. "A abordagem de Ohno era semelhante à de um cientista forense, investigando a cena do crime." (LIKER, 2003, p. 226, tradução nossa).

É chamada de *genchi genbutsu* a atividade de **ir ao local, ver, e entender o que está ocorrendo**. É imprescindível, até mesmo para os níveis mais altos da gerência, a observação *in loco* e o entendimento das situações. Como conseqüência imediata, um funcionário da Toyota sempre sabe do que está falando, porque suas informações foram obtidas em primeira mão, e não através de terceiros. Está implícito no *genchi genbutsu* que cada um é responsável pela informação passada adiante, portanto, os fatos devem ser confirmados pessoalmente. Após alguns anos de prática, é possível ao observar, realizar uma avaliação crítica da situação, a qual conduzirá à causa primordial do problema. É extremamente importante não apenas solucionar o problema, mas também impedir que ele ocorra novamente. Por isso, sempre são atacadas as causas dos problemas.

2.4.2 Tomada de Decisão (*nemawashi*)

Uma das facetas do processo de resolução de problemas executado na Toyota representa mais uma peculiaridade com a qual os novatos têm que se familiarizar. Diferente da maioria das empresas ocidentais, a Toyota acredita ser mais adequado o processo de tomada de decisão baseado no consenso. O processo completo de tomada de decisão inclui os seguintes pontos principais (LIKER, 2003):

1. Descobrir o que realmente está acontecendo.
2. Entender as causas primordiais que explicam o aparecimento de problemas.
3. Considerar de maneira abrangente alternativas de soluções e justificar de forma detalhada a solução eleita.

4. Obter consenso dentro do time, considerando, inclusive, opiniões de colaboradores externos à equipe, e até externos à empresa.
5. Utilizar veículos de comunicação eficientes para realizar os quatro pontos anteriores.

Para delinear a situação, uma das ferramentas usadas é o *genchi genbutsu*, como explicitado. Diversas implementações dos princípios anteriores também auxiliam a identificação de problemas e a caracterização da situação. Além da utilização de controle visual, pode-se destacar o fluxo contínuo e os sistemas *pull* como poderosas formas de fazer os problemas virem à tona: basta diminuir mais ainda as reservas entre os segmentos da linha de montagem.

É extremamente valioso atacar as causas dos problemas, e embutir o controle de qualidade no processo, para que contratempos antigos não advenham novamente. Para trazer à luz as causas primordiais dos problemas evidenciados, ferramentas como a *Root Cause Analysis* (RCA), a técnica de perguntar cinco vezes o porquê, são utilizadas. A RCA será explicada na próxima seção, por ser uma ferramenta essencial também no processo de melhoria contínua.

O terceiro ponto é considerado crítico para os gerentes de mais alto escalão, e o mais difícil de transmitir aos novatos. A importância de considerar diversas opções, adiando ao máximo a escolha, é que várias alternativas vindas à tona poderiam não ser expostas. Ao invés de escolher a primeira possibilidade, e iterativamente aprimorá-la, é preferível desenvolver diversas alternativas paralelamente. Logo algumas irão se destacar. Esta é uma das principais diferenças entre a Toyota e outras empresas, é o que Ward (1995) chamou de "o segundo paradoxo da Toyota" (o primeiro seria o Sistema de Produção em si): no desenvolvimento de produtos, antes de decidir a solução a ser posta em prática, utilizando-se extensivamente de protótipos, os gerentes e engenheiros da Toyota consideram diversas alternativas, o que, geralmente, demanda tempo precioso para o desenvolvimento da solução; mesmo assim, a Toyota consegue desenvolver novos produtos mais rapidamente do que seus concorrentes.

É chamado de *nemawashi* o processo de **tomar decisões devagar e por consenso, considerando todas as opções e executar a escolha rapidamente**. A tomada de decisão na Toyota pode ocorrer de diversas maneiras, de uma decisão unilateral por parte do gerente até uma decisão tomada completamente em grupo, através de consenso. A forma preferida é o consenso, mas com aprovação do gerente. Sempre que o grupo não conseguir chegar ao

consenso, o chefe entra em cena e pode decidir. Da mesma forma, caso uma decisão mais rápida seja necessária, o gerente pode buscar informações com outros e tomar a decisão sozinho. O segredo para a execução rápida é o planejamento extensivo anterior à realização. De acordo com Alex Warren (apud LIKER, 2003, p. 237, tradução nossa), antigo vice-presidente de uma das fábricas americanas, "num projeto de um ano, 10 meses são utilizados para planejar, depois a implementação pequena é realizada, como uma produção piloto, e no fim do ano é totalmente executada, com praticamente nenhum problema restante".

Para permitir que o processo flua suavemente, os esforços são realizados para melhorar a comunicação entre as partes. A abordagem visual é a preferida, com base no fato de que as pessoas são mais bem orientadas visualmente. Quando recém contratados, os funcionários são treinados a se expressarem de forma sintática e com auxílio visual.

O *nemawashi* tem como principais resultados: a exposição de possíveis riscos e problemas que de outro modo não seriam descobertos e demandariam grandes esforços para voltar atrás; a opinião de todas as partes envolvidas pode ser alinhada de maneira que qualquer resistência à alternativa escolhida pode ser trabalhada antes do início da execução; o aprendizado alcançado antes mesmo do início das atividades de planejamento e implementação.

2.4.3 Organização que Aprende (*hansei*) e Melhora Continuamente (*kaizen*)

Evoluir ou aprender continuamente, no contexto organizacional, significa ser capaz de crescer baseado no passado, e seguir em frente gradualmente, ao invés de recomeçar tudo com novo pessoal e novos projetos. Além dos processos estáveis, é preciso ter estabilidade também na mão-de-obra empregada, promoções lentas e sistemas de sucessão cautelosos, que protejam a base de conhecimento da organização (LIKER, 2003). O *kaizen*, ou melhoria contínua, constitui-se na maneira de pensar e agir de todos os que fazem parte da organização: a atitude de um indivíduo reconhecer abertamente os problemas existentes, e até mesmo a culpa, e propor medidas corretivas para prevenir a recorrência dos erros.

As lideranças, como retratado no nono princípio do *Toyota Way*, devem ser cultivadas dentro da empresa, e não trazidas de fora, e os times devem ser formados por trabalhadores educados dentro da cultura organizacional vigente, o que significa que eles também são responsáveis por ensinar aquilo que aprenderam aos próximos.

Conforme explanado anteriormente, o estabelecimento de processos estáveis é o primeiro passo para a evolução contínua. Só é possível alcançar uma meta com segurança sabendo-se onde se está. Por isso, depois da instituição e padronização de processos, é preciso observar onde podem ser melhorados. Por vezes, as ineficiências serão facilmente visíveis, mas após algum tempo, os problemas tenderão a se esconder. É necessário, então, trazê-los à luz, encontrar suas causas e aplicar medidas corretivas.

Na busca de eliminar o desperdício da linha de produção, Taiichi Ohno estabeleceu o fluxo contínuo como a forma mais eficiente de se produzir. Como nem sempre é possível instituir a continuidade do fluxo, reservas são utilizadas entre os segmentos produtivos para manter a suavidade do fluxo. Os estoques, por sua vez, representam desperdício. Para diminuir esse desperdício e promover uma ligação direta entre a produção e a demanda, sistemas *pull* são utilizados. É interessante perceber o fluxo contínuo e os sistemas *pull* como uma ferramenta espetacular para a exposição de problemas. Basta diminuir o estoque entre dois segmentos, e observar onde o processo falha, e pode, portanto, ser melhorado. Exposto o problema, é preciso utilizar um processo para sua resolução. Na Toyota, não basta solucionar o problema aparente, é preciso atacar suas causas.

Para Ohno, o processo de resolução de problemas requer identificar a causa ao invés da fonte do problema. O importante é perguntar "por que o problema ocorre". Um dos processos mais interessantes dentro da Toyota chama-se *Root Cause Analysis*, ou "os cinco porquês" (*five-Why*). A RCA resume-se em: perguntar por que aconteceu um determinado problema, encontrar a resposta, e perguntar o porquê novamente para descobrir sua causa. Em geral, as respostas vão levar aos níveis mais elevados da organização, e mais abstratos do processo. Como exemplificado na Figura 2-4 (adaptada de LIKER, 2003, p. 253 il.):

	Problema	**Medida Corretiva**
	Há uma poça de óleo no chão da fábrica	Limpar o óleo
Por quê?	Porque uma máquina está vazando	Consertar a máquina
Por quê?	Porque a vedação está deteriorada	Substitua a vedação
Por quê?	Porque foi comprada vedação de qualidade inferior	Mudar as especificações da vedação
Por quê?	Porque foi feito um bom acordo (preço) na compra da vedação	Mudar as políticas de compra
Por quê?	Porque os responsáveis pelas compras baseiam suas decisões em economias de curto-prazo	Mudar a política de avaliação dos responsáveis pelas compras

Figura 2-4 *Five-Whys*

Como podemos observar, correções superficiais não impedirão os problemas de acontecer novamente, elas são apenas paliativas. Quanto mais se aprofunda em busca das causas reais de um problema, mais duradoura e abrangente é a medida corretiva a ser tomada. Naturalmente que o óleo que suja o chão deve ser limpo e a máquina deve ser consertada, mas o mais importante é a mudança na política de avaliação e aquisição. Esta medida refletirá em benefícios para outras partes da empresa cujos problemas ainda não se fizeram expor.

Além da padronização das atividades, a exposição dos problemas e de suas causas, a execução de medidas corretivas, o processo de melhoria contínua também envolve a verificação da eficácia das medidas tomadas. O *hansei*, traduzido no ocidente como "reflexão", é uma das ferramentas utilizadas pelos japoneses para refletir sobre a eficácia do processo. São recomendadas reuniões de *hansei* sempre em marcos importantes (*milestones*) e ao fim de cada projeto para que problemas no decorrer do projeto sejam expostos e suas causas possam ser corrigidas. Mais importante ainda, para que haja aprendizado sobre os erros e as boas práticas sejam seguidas e incorporadas aos padrões.

O *kaizen*, pelo qual o Sistema Toyota de Produção é conhecido mundialmente, envolve os diversos princípios e ferramentas descritos até agora: as atividades padronizadas, como ponto de partida para a melhoria contínua; o fluxo, e seu uso para trazer os problemas à tona; o entendimento da situação e a *Root Cause Analysis* para estabelecer as medidas corretivas; e a reflexão, para verificar a eficácia das medidas. Como podemos notar, as idéias difundidas por W. Edwards Deming influenciaram bastante as noções de controle de

qualidade e também a Toyota e seus processos. O ciclo pode ser percebido como mostra a Figura 2-5 (adaptada de LIKER, 2003, p. 264, il.):

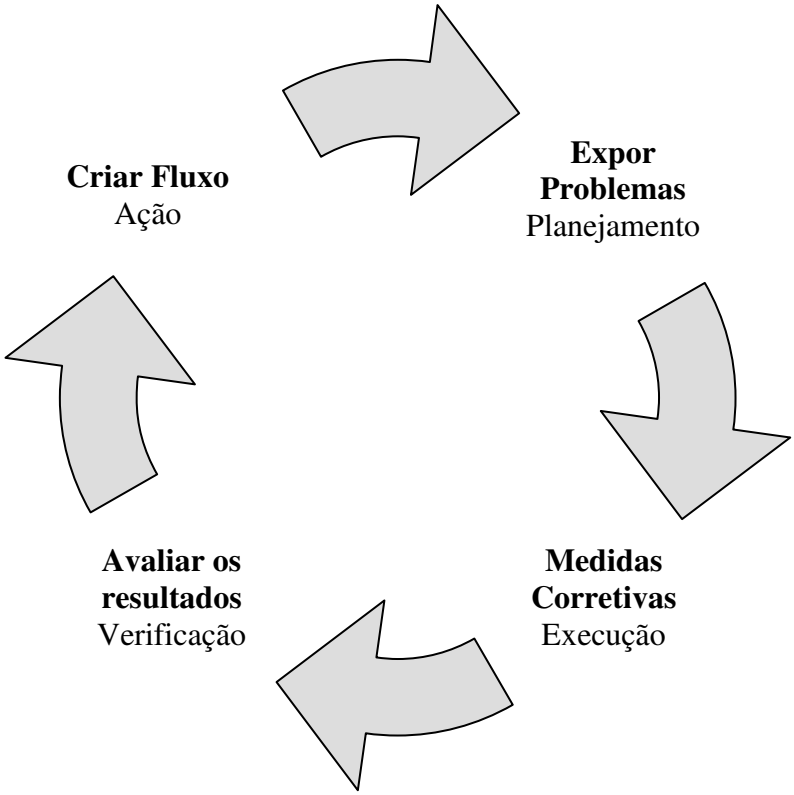

Figura 2-5 O Ciclo PDCA na Toyota

Kaizen quer dizer mudança para melhor. Para pô-lo em prática pode-se lançar mão de ferramentas como o *hansei*, o desdobramento de políticas (*hoshin kanri*), *Root Cause Analysis* etc. **Tornar-se uma organização que aprende através da melhoria contínua** não é uma tarefa fácil de executar. A própria Toyota levou algumas décadas para chegar ao nível em que se encontra. Mais importante ainda, o processo de melhoria nunca pode parar, como sugere a forma cíclica dos ensinamentos de Deming, e deve ser aplicado em todos os níveis do organograma da empresa, e até envolver empresas parceiras.

2.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram descritos os 14 princípios que regem o dia-a-dia dos integrantes da Toyota. É importante observar que os princípios representam bases filosóficas para as atividades da empresa, e podem ser observados não apenas na linha de montagem, mas também nos segmentos de serviços e até nas empresas que firmam parcerias com a gigante

japonesa. Também é possível observar a influência sofrida pela empresa de diversos métodos de gestão da qualidade, como o 5S, e o TQM.

É preciso, novamente, deixar claro a distinção entre o TPS e os princípios de gerenciamentos acima descritos: o TPS representa o que pode ser alcançado através da concretização dos princípios de gerenciamento difundidos pela Toyota. Unido à base filosófica, o TPS representa muito mais do que um conjunto de ferramentas para solucionar problemas em curto prazo. Ele representa até bem mais do que um simples sistema de produção automobilística. A maneira Toyota de produzir representa uma forma diferenciada de pensar, considera a qualidade como parte integrante de todo o processo produtivo de forma abrangente e organizada. Ela mantém claro o objetivo de satisfazer o cliente e sempre procura evolver seus produtos e serviços e bem como a forma como eles são, respectivamente, fabricados e realizados.

A seguir será descrita a *Extreme Programming*, uma abordagem para o desenvolvimento de software, que também tem como foco principal a satisfação do cliente.

Capítulo 3
EXTREME PROGRAMMING

Replacing an on-site customer with some use cases is about as effective as replacing a hug from your Mom with a friendly note.
RON JEFFRIES

Optimism is an occupational hazard of programming: feedback is the treament.
KENT BECK

Em 1995, a *Chrysler Corporation* iniciava o desenvolvimento do projeto *Chrysler Comprehensive Compensation* (C3), cuja meta era substituir os diversos sistemas de folha de pagamento por um sistema novo. O C3 também tinha a função de piloto na adoção de linguagens orientadas a objeto em substituição às procedurais, e *Smalltalk* foi escolhida para a implementação. Cerca de um ano depois do início, o projeto sofria sérios problemas de desempenho, quando Kent Beck foi contratado por sua experiência no desenvolvimento com *Smalltalk*. Em seus primeiros contatos com a equipe, ficou claro que o problema do projeto ia além do desempenho do sistema: após quase um ano de desenvolvimento, o C3 não era sequer capaz de imprimir um contracheque corretamente! Após uma reunião com o CIO da empresa, seduzido pelo desafio, Beck foi responsável pela continuação do projeto.

No comando, Kent Beck trouxe Ron Jeffries para atuar como *coach* do time. Martin Fowler, que já realizava consultoria no projeto no que dizia respeito à análise e aos testes, também foi uma das peças fundamentais para a adoção do conjunto de práticas de desenvolvimento introduzidas pelo novo líder do projeto. As idéias envolviam práticas antigas do desenvolvimento de software e inovações estudadas em conjunto por Beck e Cunningham, como a utilização de cartões CRC (BECK; CUNNINGHAM, 1989) e programação em pares (WILLIAMS et al., 2000).

Em abril de 1997 o sistema entrou em produção. Beck (1999) publicou a primeira edição de *Extreme Programming Explained: Embrace Change*, que trazia uma compilação dos valores, princípios e práticas defendidos por ele, desenvolvidos por toda a equipe do projeto C3 e amplamente discutidos no *WikiWikiWeb* (CUNNINGHAM, 1995-). Diversas publicações surgiram e uma ampla comunidade foi criada em torno da metodologia para sua discussão e aprimoramento.

Cinco anos após o lançamento primeiro livro, Kent Beck revisita sua obra e reescreve todo o livro, levando em consideração as discussões realizadas pela comunidade XP. Na nova edição, características que se mostraram ineficazes foram abandonadas e outras introduzidas. A seguir, será descrita a metodologia, seus valores, princípios e práticas.

3.1 VISÃO GERAL

Extreme Programming (XP) é uma abordagem ágil para desenvolvimento de software. Por "ágil" entende-se que ela compartilha dos mesmos princípios que o Manifesto Ágil (BECK et al., 2001), que são, entre outros: comunicação intensa, trabalho em equipe e proximidade entre os envolvidos no processo. Ela inclui:

- uma filosofia de desenvolvimento baseada nos valores de comunicação, simplicidade, feedback, coragem e respeito;
- um conjunto de práticas de desenvolvimento de software que se complementam e exercitam os valores, como a programação em pares, estórias de usuários, programação orientada a testes etc.; e,
- um conjunto de princípios que servem como ponte entre os valores e as práticas, como fluxo, benefício mútuo, responsabilidade aceita etc. (BECK; ANDRES, 2004).

Para Beck, valores representam um nível mais abstrato de conhecimento e compreensão, são como guias em nível estratégico para julgar o que se vê, pensa e faz. Eles devem representar os interesses da organização. Todos os que fazem parte da companhia devem estar alinhados com os valores em vigor. Eles são o motivo para realizar uma determinada prática.

Práticas são atividades concretas e bem descritas. Elas devem ser evidências da adoção dos valores e da importância dada a eles. Programação em pares, por exemplo, promove comunicação entre os membros da equipe; ciclos curtos de desenvolvimento promovem feedback; e assim por diante.

Para preencher a lacuna entre a abstração dos valores e a concretude das práticas estão os princípios. O princípio da auto-semelhança defende que soluções que deram certo devem ser aplicadas em outras ocasiões sempre que possível e adequado, e pode ser observado em

diferentes níveis do ciclo de desenvolvimento XP: testes são escritos antes da implementação de novas funcionalidades em seu nível mais granular, sob a forma de testes unitários, e em mais alto nível, os testes funcionais.

As principais características de XP, que a diferencia de outras abordagens, são: os ciclos curtos de desenvolvimento, que acarretam feedback rápido e contínuo; a abordagem incremental para o planejamento; emprego de testes automatizados escritos pelos desenvolvedores e pelos clientes; o emprego de processos evolutivos de análise e projeto de sistemas; a capacidade de acolher mudanças nos requisitos a qualquer momento do projeto para responder melhor às mudanças do negócio. (BECK; ANDRES, 2004).

Apesar de conhecida, em seus primórdios, por ser uma metodologia voltada para equipes pequenas que trabalham com definições de requisitos de software vagas e susceptíveis à mudança, há diversos relatos de aplicação da metodologia em condições diferentes às da definição, como a aplicação de XP em equipes com cerca de 40 desenvolvedores, 15 analistas de negócio e 10 analistas de qualidade (ELSSAMADISY, 2001).

3.2 VALORES

Valores representam bases filosóficas que servem para alinhar as atividades da equipe. Práticas sem ter valores maiores não têm razão para serem realizadas. Quando os valores são deixados de lado, o que se têm são comportamentos idiossincráticos, que não compartilham dos interesses maiores da equipe ou da organização. Além dos cinco valores a serem descritos abaixo, outros podem fazer parte da fundamentação filosófica da equipe ou da organização para levar o desenvolvimento de software ao patamar mais elevado possível, o importante é que o comportamento dos integrantes da equipe reflita os valores adotados por eles. Beck e Andres (2004) citam segurança e qualidade de vida como possíveis valores adicionais.

Os valores também se relacionam: melhoras na comunicação entre a equipe e o cliente podem promover simplicidade quando requisitos desnecessários podem ser deixados de lado; a coragem para falar a verdade promove comunicação e confiança; a busca pela simplicidade pode ser auxiliada pelo feedback intenso. (BECK; ANDRES, 2004).

3.2.1 Comunicação

Sem dúvida, comunicação é a questão mais importante do desenvolvimento de software. Grande parte dos problemas que ocorrem durante um projeto é causada por falta de comunicação ou por comunicação não efetiva. Beck e Andres (2004) sugerem que sempre que um problema emergir, deve ser questionado se foi causado por falha na comunicação. Neste caso, deve ser encontrada uma maneira para corrigir esta falha.

Em geral, sempre há alguém na equipe que já passou por uma dificuldade semelhante, mas nem sempre é possível encontrar essa pessoa e promover a cooperação adequada. "A transmissão de conhecimento tácito representa um desafio significativo para as equipes de desenvolvimento, o qual pode ser solucionado de forma mais ou menos eficaz dependendo dos mecanismos de comunicação adotados no projeto." (TELES, 2005b, p. 61).

No ambiente de desenvolvimento de software, uma das coisas para a qual se deve atentar é a riqueza dos meios de comunicação. Cockburn (2002) classifica a eficácia da comunicação em relação à riqueza, ou temperatura do canal de comunicação utilizado. Para o autor, na maioria das vezes, canais com via única, como impressos, são menos eficazes que os de mão dupla, como conversas ao telefone. Em relação à riqueza dos canais: a comunicação através de e-mails é mais "fria" que a interação face a face.

> [comunicação face a face é] o canal de comunicação mais barato e rápido para troca de informação. [...]
> Em geral, devemos preferir o uso de canais de comunicações mais *quentes* no desenvolvimento de software, sempre que estivermos interessados em reduzir o custo de detecção e transferência da informação. (COCKBURN, 2002, p. 149, tradução nossa, grifo nosso).

3.2.2 Simplicidade

"Simplicidade – a arte de maximizar a quantidade de trabalho **não** feito" (BECK et al., 2001, tradução nossa, grifo nosso) é um dos valores mais difíceis de dispor. Ter simplicidade como um valor significa realizar sempre o mais simples necessário hoje, e melhorar gradativamente quando necessário. A idéia não é ser simplista, mas eliminar toda a complexidade desnecessária.

> [...] Num projeto ágil, é particularmente importante usar abordagens simples, porque elas são mais fáceis de se mudar. É mais fácil acrescentar algo a um processo muito simples do que retirar algo de um processo muito complexo. [...] Incluía **apenas o que todo mundo precisa** invés de **qualquer coisa que alguém precise**, para facilitar o acréscimo de algo para tratar de uma necessidade particular do time. (FOWLER; HIGHSMITH, 2001, tradução nossa, grifo nosso).

Simplicidade sempre está inserida num contexto, também está relacionada com a audiência. Tarefas extremamente complexas para novatos podem ser simples para um grupo de especialistas (BECK; ANDRES, 2004).

3.2.3 Feedback

Feedback é uma parte crítica da comunicação. Através dele é possível obter a capacidade de adaptação e mudança defendida pelo movimento ágil. Dificilmente num projeto de software as circunstâncias permanecerão estáveis por muito tempo.

Em contraste à tradicional curva exponencial de custo de mudança durante o desenvolvimento de software, Beck (1999) defende que ao aplicar XP, o custo para se realizar uma mudança no software mantém-se constante ao longo do tempo. "Fundamentalmente, a razão pela qual o a curva de custo de mudança ágil tornou-se constante é porque nós seguimos técnicas que reduzem o ciclo de feedback" (AMBLER, 2006b).

Ambler (2006c) relaciona técnicas de desenvolvimento de software com a curva de custos de mudança, como mostra a Figura 3-1 (adaptada de AMBLER, 2006c, il.). As técnicas tradicionais promovem um ciclo de feedback longo, que variam de dias a meses, enquanto as técnicas ágeis, grifadas na figura, proporcionam ciclos de horas, minutos e até segundos. Problemas encontrados através do uso de técnicas tradicionais, portanto, acarretam custos maiores de correção.

> Uma das muitas razões por que as técnicas ágeis são tão efetivas, em minha opinião, é que elas reduzem o ciclo de feedback entre a geração de uma idéia (talvez um requisito ou uma estratégia de design) e a realização dessa idéia. Isso não apenas minimiza o risco do entendimento equivocado, também reduz o custo para atender de quaisquer erros. (AMBLER, 2006c).

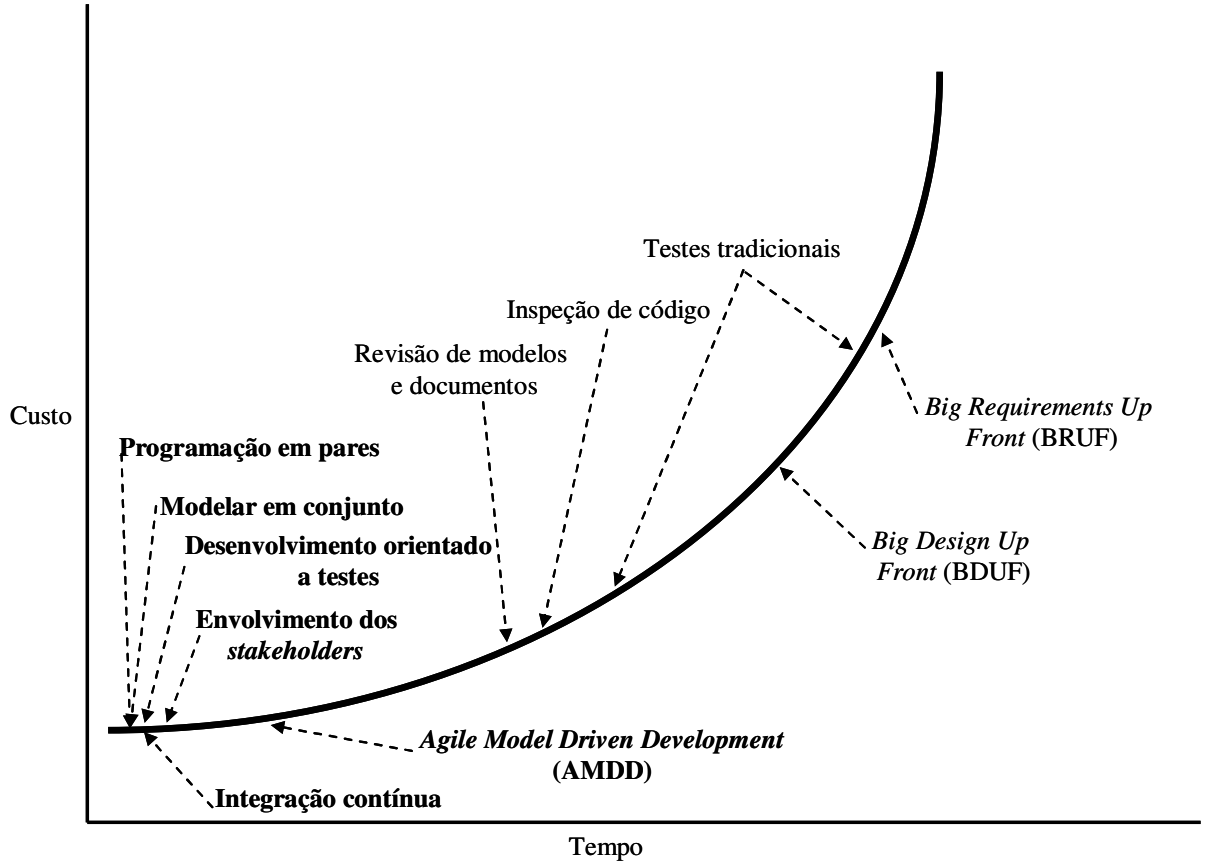

Figura 3-1 Comparação do ciclo de feedback de técnicas de desenvolvimento de software

3.2.4 Coragem

É preciso coragem para fazer a coisa certa: seguir a metodologia inclusive nos momentos críticos, restringir o escopo no andamento da iteração para poder alcançar o prazo estipulado, manter a comunicação efetiva e transparente com o cliente, apontar falhas e sugerir e realizar transformações no próprio processo de desenvolvimento. Enquanto a coragem sozinha pode ser inconseqüente, em associação com outros valores pode trazer muitos benefícios, por exemplo, ter coragem para abandonar soluções antigas em busca de outras mais simples, trará benefícios para a manutenção do projeto.

> Ter coragem em XP significa ter **confiança** nos mecanismos de segurança utilizados para proteger o projeto. Ao invés de acreditar que os problemas não ocorrerão e fazer com que a coragem se fundamente nesta crença, projetos XP partem do princípio de que problemas irão ocorrer, inclusive, aqueles mais temidos. Entretanto, a equipe utiliza redes de proteção que possam ajudar a reduzir ou eliminar as conseqüências destes problemas. (TELES, 2005b, p. 68, grifo do autor).
> [...] Acho que coragem é uma atitude pessoal importante, mas que beira a inutilidade quando não existem mecanismos práticos de proteção que possam confortar aqueles que decidam ser corajosos. [...] (TELES, 2005a, mensagem pessoal).

A coragem em XP não deve ser vista como valentia ou ousadia, mas como confiança e responsabilidade. Os desenvolvedores empregam mecanismos sérios de proteção para poderem ter confiança, por exemplo, para acolher mudanças nos requisitos a qualquer momento.

3.2.5 Respeito

O respeito é necessário para que todos os outros valores façam sentido. Sem respeito pelos colegas não há trabalho em equipe, sem levar o projeto a sério, não há como alcançar o sucesso. A importância de cada integrante da equipe deve ser levada em conta, e cada um, como ser humano, e cada opinião devem ser respeitados.

O respeito não era um valor descrito nas primeiras publicações sobre a metodologia, mas a discussão gerada pela comunidade em torno de XP trouxe à luz esse valor que era praticado em cada projeto bem sucedido.

3.3 PRINCÍPIOS

Os princípios aproximam os valores e as práticas. Eles também são guias úteis para serem seguidos quando ainda não há uma prática específica para solucionar determinado problema. Os princípios descritos a seguir guiam as práticas de XP alinhadas aos valores já descritos. Naturalmente, em certas ocasiões, outros princípios podem ser seguidos adicionalmente, desde que não sejam conflitantes.

3.3.1 Humanidade

Seguir o princípio da humanidade significa estar ciente de que o software é desenvolvido por pessoas. Fatores humanos são as principais causas para o sucesso ou falha de um projeto. As práticas de XP procuram balancear as necessidades pessoais de cada membro da equipe e as necessidades da organização. Desequilíbrios podem causar, por um lado, baixa produtividade, se as necessidades na organização não são levadas em conta; ou trabalho excessivo, insatisfação e alta taxa de rotatividade de pessoal.

Em geral, as necessidades pessoais dos trabalhadores que podem ser supridas no ambiente de trabalho são: a noção de segurança ligada à manutenção do emprego; o

sentimento de realização pessoal na execução do seu trabalho; o sentimento de pertinência à equipe; a oportunidade de evolução das suas competências e perspectivas; e a habilidade de entender e ser entendido pelos outros. Outras necessidades, como descanso e socialização, não precisam, necessariamente, ser atendidas pela organização. É importante perceber que um projeto de desenvolvimento de software é extremamente dependente da capacidade criativa dos integrantes da equipe. Eles também precisam de tempo fora do trabalho para recarregar as energias. Como veremos posteriormente, várias práticas, como Trabalho Energizado, estão muito ligadas ao princípio da Humanidade.

3.3.2 Economia

Este princípio está intimamente ligado às necessidades do negócio. O desenvolvimento de software não pode ser considerado bem sucedido apenas do ponto de vista técnico. É preciso estar sempre orientado pelos objetivos da empresa e buscar gerar o máximo de valor, o mais cedo possível, através do desenvolvimento do software.

Primeiramente, é preciso ter em mente que, quanto mais cedo se obtiver retorno financeiro com a utilização do software e quanto mais tarde se precisar gastar com ele, melhor. As práticas que regem o planejamento do projeto procuram sempre priorizar as funcionalidades a serem implementadas de acordo com as necessidades do negócio. Aliadas a ciclos curtos de desenvolvimento e a produção incremental, possibilitam o retorno sobre o investimento em menor período. Adiar decisões de design, até que elas sejam terminantes, também protela os gastos até o último momento possível.

3.3.3 Benefício Mútuo

Para todos os problemas devem se buscar soluções que beneficiem os envolvidos, agora e no futuro. As famigeradas situações ganha-ganha. Em situações críticas, soluções unilaterais mais simples podem ser tentadoras, mas também podem provocar desgaste no relacionamento dos envolvidos.

A documentação extensiva é uma prática que pode violar o princípio do benefício mútuo. Ela diminui a velocidade de desenvolvimento para facilitar a manutenção futura do código. Ela beneficia o futuro mantenedor, mas não gera benefícios no presente. (BECK;

ANDRES, 2004). Como veremos, práticas como a programação orientada a testes, refatoração e os padrões de codificação trabalham a questão da comunicação futura dentro de XP.

3.3.4 Auto-semelhança

Como já explicado brevemente, o princípio da auto-semelhança defende que se deve procurar repetir soluções bem sucedidas. Não significa que soluções únicas são ruins, ou que aquelas já implantadas irão funcionar em qualquer contexto.

Seguindo este princípio, a própria metodologia evoluiu. A proposta inicial para o ciclo semanal assemelha-se ao modelo Cascata de Desenvolvimento, no qual os testes de sistema são realizados após a implementação das funcionalidades (BECK, 1999). Posteriormente, é reportado que utilizar uma abordagem orientada a testes, como já era a sugestão para os testes unitários, também para testes de sistema, simplifica o design e melhora o feedback (BECK; ANDRES, 2004).

3.3.5 Aprimoramento

A busca pela melhoria deve ser diária. Mesmo ciente de que a perfeição não pode ser alcançada, sua simples busca trará benefícios. O princípio do aprimoramento pode ser observado em práticas como o design incremental e o uso de refatoração.

Os ciclos trimestrais também são excelentes oportunidades para realizar reflexões. (BECK; ANDRES, 2004; TELES, 2004).

3.3.6 Diversidade

"A diversidade é expressa em práticas como Equipe Completa, onde participantes com diversas habilidades e perspectivas são colocados no mesmo time". (BECK; ANDRES, 2004, p. 29, tradução nossa). É importante montar um time diversificado, pela versatilidade potencial da equipe. Um time heterogêneo pode ser uma fonte de conflitos, mas equipes maduras, que têm como valor o Respeito, vêem conflitos como oportunidades, não como brigas. Resolver conflitos de personalidade quando eles estão atrapalhando a equipe é função do gerente.

3.3.7 Reflexão

Enquanto os ciclos semanais e trimestrais reservam espaço para reflexão, não se deve esperar as reuniões oficiais para se refletir sobre como o trabalho está sendo realizado. Incluído no princípio da reflexão está a coragem, necessária para expor as falhas no processo. A importância de se utilizar dinâmicas de reflexão no processo de desenvolvimento de software já foi descrita por Cockburn (2002) e Teles (2004), além de ser um princípio compartilhado pelo Manifesto Ágil (BECK et al., 2001).

3.3.8 Fluxo

Ao seguir o princípio do fluxo, a equipe dá preferência às entregas (*deployment*) freqüentes e regulares ao invés de versões com grandes mudanças. Este princípio está intimamente ligado à utilização de passos pequenos, descrito posteriormente. XP não tem fases definidas de desenvolvimento, ao contrário do modelo Cascata de Desenvolvimento, as atividades são executadas simultaneamente e de forma integrada.

O fluxo também é importante para expor problemas. Falhas mais sérias, que interrompam o fluxo do desenvolvimento devem ser sanadas o quanto antes para evitar maiores prejuízos, depois o fluxo pode ser retomado.

3.3.9 Oportunidade

Segundo Beck (2004), adotar XP também está relacionado com transformar problemas em oportunidades: de crescimento pessoal, de aprofundamento das relações e de melhoria do software produzido. A atitude de simplesmente resolver os problemas não é suficiente para chegar ao patamar de excelência na produção de software. É preciso aprender a enxergar as oportunidades de mudança e melhoria ao transpor um obstáculo.

3.3.10 Redundância

Apesar de parecer desperdício, a redundância pode ter seu valor e deve ser utilizada sempre que necessário. Quando se trata de problemas críticos, o preço pago pela redundância é mais do que compensado pelo potencial prejuízo causado pela falha. Na metodologia podemos observar, por exemplo, que a busca por defeitos é tratada de forma redundante.

Programação em pares, integração contínua, programação orientada a testes, envolvimento do cliente, todas são práticas que auxiliam a busca e a eliminação dos defeitos. O tratamento redundante para este problema é contrabalanceado pelo ganho em qualidade do produto.

3.3.11 Falha

É melhor tentar e falhar do que gastar tempo tentando realizar tudo certo da primeira vez. É preciso compreender que se há aprendizado, a falha não pode ser considerada desperdício. Obviamente, se uma solução já é conhecida, a falha não pode ser desculpada, mas, várias vezes, o erro inicial é o caminho para a solução.

3.3.12 Qualidade

A qualidade não deve ser usada para controlar o projeto. Deve-se sempre manter padrões altos de qualidade para estimular a economia e a produtividade. A qualidade elevada também tem seus efeitos positivos no campo motivacional, haja vista a necessidade de realização pessoal dos membros da equipe.

Como a qualidade não deve ser uma variável no controle do projeto, e como custo e prazo em geral são fixos, o controle do desenvolvimento é feito variando o escopo. Como, normalmente, o escopo ainda é vago no início do projeto, ele vai sendo mais bem definido no decorrer do projeto, durante as reuniões dos ciclos semanal e trimestral. (BECK; ANDRES, 2004).

3.3.13 Pequenos passos

Devem ser utilizados passos pequenos para realizar mudanças. Grandes mudanças, realizadas com grandes passos, podem ser perigosas. Caminhar em passos pequenos é mais seguro, porque se pode facilmente mudar a direção sem grandes prejuízos. Práticas como a Integração Contínua demonstram como utilizar passos pequenos no desenvolvimento de software, no caso, integrar pequenas parcelas de código várias vezes por dia (FOWLER, 2006).

Incrementos pequenos não significam baixa velocidade, é absolutamente viável trabalhar com passos pequenos, mas rápidos, que parecerão largos saltos externamente.

Segundo Fowler (2006, tradução nossa), "várias equipes concluem que essa abordagem [da integração contínua] acarreta significativamente menos problemas de integração e permite o desenvolvimento de software coeso mais rapidamente".

3.3.14 Responsabilidade aceita

A responsabilidade deve ser aceita, não imposta. Este princípio pode ser observado em diversas práticas que envolvem planejamento, como o ciclo semanal, quando os desenvolvedores tomam a responsabilidade sobre a realização de uma tarefa. Ao analisarmos o papel do gerente na equipe XP, posteriormente, veremos que sua função aproxima-se mais de um facilitador do que a de impositor. A imposição de tarefas pode ocorrer, mas não é ideal.

3.4 PRÁTICAS

As práticas representam a porção mais detalhada e concreta da metodologia. Elas são divididas em dois grandes grupos: as primárias e as do corolário. As práticas primárias são mais simples e devem ser adotadas antes. As outras devem ser utilizadas apenas quando todas as primárias já estiverem em uso pleno e servem para tratar de problemas específicos. Não há uma ordem ideal para se adotar as diversas práticas. A sugestão é usar a prática mais adequada para auxiliar no tratamento de um problema, no momento em que for identificado durante o processo de desenvolvimento. (BECK; ANDRES, 2004).

> Iniciar com XP é como entrar numa piscina [...] você pode colocar apenas o dedo; você pode sentar na beira e balançar os pés na água; você pode descer pela escada; você pode dar um mergulho suave e potente como um nadador; ou um *cannonball*, fazendo muito barulho e molhando todos ao seu redor. Não há um jeito certo de entrar na água. (BECK; ANDRES, 2004, p. 139-140, tradução nossa, grifo nosso).

Cada prática sozinha promove melhorias no processo de desenvolvimento que podem ser observadas rapidamente. À medida que se introduzem mais práticas no dia-a-dia da equipe, também se pode notar como elas influenciam positivamente umas às outras.

A seguir, serão descritas as treze práticas primárias propostas por Beck e Andres (2004), além de uma visão geral das práticas do corolário.

3.4.1 Sentando Juntos

A prática sugere que toda a equipe de desenvolvimento seja acomodada num espaço comum, de forma a favorecer a comunicação entre os integrantes do time. Quando não há a disponibilidade permanente de um espaço adequado, utilizar a sala de reunião eventualmente para realizar tarefas de desenvolvimento pode ajudar.

Além de facilitar a reunião para discussão em conjunto, permitir que a equipe trabalhe junta, na mesma sala favorece o fenômeno da *comunicação osmótica* (COCKBURN, 2002). A informação é distribuída até para aqueles que não estão participando ativamente de uma conversa, por exemplo.

A organização do local de trabalho pode influenciar bastante os custos de comunicação num projeto, uma vez que: minimiza a falta de oportunidade de realizar uma pergunta e facilita o processo de detectar e transferir informações, possibilitado pela comunicação osmótica. Enquanto equipes que trabalham juntas se beneficiam dos efeitos citados, equipes cujos integrantes estão separados sofrem com a falta deles.

Wake (2001) descreve como ambientes de trabalho abertos, recomendados por XP, podem ser organizados sem excessos. Não é interessante juntar integrantes do projeto que não estejam na mesma fase do projeto, ou programadores de projetos diferentes. Nestes casos, ambientes muito abertos começarão a atrapalhar a comunicação e o fluxo de informações interessantes aos projetos.

Mudanças radicais no ambiente de trabalho não são sugeridas, pois a equipe pode ainda não estar preparada para tanto contato. Neste caso, a adoção desta prática pode ser contraproducente. “Se o senso de segurança dos membros da equipe estiver relacionado a ter seu próprio espaço, removê-lo antes de sua substituição pela segurança da realização comum provavelmente trará ressentimento e resistência.” (BECK; ANDRES, 2004, p. 38, tradução nossa).

3.4.2 Equipe Completa

Procure construir um time com todas as habilidades necessárias para alcançar o sucesso. As habilidades dos membros da equipe devem ser, na medida do possível, complementares. Todos os integrantes devem estar dispostos a interagir para que o aprendizado seja promovido.

Também se deve ter em mente que a idéia de uma equipe completa é dinâmica: caso um determinado conjunto de habilidades seja necessário, traga alguém com tais características para a equipe; caso não seja mais necessário, mude sua função.

O estabelecimento de uma equipe completa está muito relacionado ao princípio da humanidade e às necessidades pessoais de cada membro da equipe: o sentimento de pertinência ao time e a habilidade de entender e ser entendido pelos outros.

É importante também o papel do *Coach* na equipe XP. Ele representa uma liderança dentro da equipe de desenvolvimento. Ele deve ter bastante conhecimento técnico e também muito conhecimento sobre a metodologia. Como XP é uma metodologia que exige alta disciplina dos seus praticantes, deve haver mecanismos para alinhar os integrantes da equipe ao processo (COCKBURN, 2002). Essa é uma das funções do *Coach*.

Ademais, suas habilidades levam-no naturalmente ao posto de liderança para promover mudanças e melhorias. Sua experiência ajuda-o a evidenciar os problemas e suas causas, para que possam ser apontadas soluções à maneira ágil, sem romper com a metodologia.

3.4.3 Ambiente de Trabalho Informativo

O ambiente de trabalho deve refletir o andamento do projeto. Deve ser possível para um visitante ficar a par do progresso da equipe. Através de observação um pouco mais aprofundada, problemas reais e potenciais podem ser descobertos (BECK; ANDRES, 2004).

Enquanto a comunicação face a face ainda é a mais apropriada e eficaz na maioria dos casos, cartazes com gráficos podem ser bastante úteis para manter a equipe, ou qualquer visitante, informados. Problemas recorrentes devem sempre ser acompanhados de perto e divulgados. Sugestões incluem: funcionalidades implementadas, problemas corrigidos, e eficácia dos testes funcionais (JEFFRIES, 2004).

É importante que se utilizem painéis pela sala ao invés de páginas na intranet da empresa, porque as últimas exigem que o membro da equipe vá à busca da informação. Painéis nas paredes comunicam sempre (JEFFRIES, 2004). Para Cockburn (2002, p. 85, tradução nossa):

> [...] da mesma forma que aquecedores irradiam calor pela sala, pôsteres irradiam informação. Eles são excelentes para passar adiante informações de forma silenciosa, com pouco esforço e sem atrapalhar as pessoas para as quais o status está sendo comunicado.

O espaço deve propiciar comunicação entre os membros da equipe, e informar a todos os interessados o andamento do projeto. Além disso, fazem parte do local de trabalho itens relacionados a outras necessidades humanas:

- Água, café e comida;
- Limpeza e higiene;
- Jogos e brinquedos;
- Privacidade.

Bebida e comida promovem maior interação entre os membros da equipe, e como os jogos e brinquedos ajudam a relaxar a mente.

Apesar de as atividades de desenvolvimento ocorrerem idealmente em conjunto, cada um precisa de privacidade. Espaços privados reservados na sala podem ser utilizados ou horários de trabalhos limitados podem suprir as necessidades de privacidade dos membros da equipe (BECK; ANDRES, 2004). A organização recomendada, denominada *caves and common* (COCKBURN, 2002, p. 89-90), inclui um espaço comum, propício para a programação em pares, irradiação da informação através de cartazes e quadro brancos nas paredes, etc., e espaços privados, onde os desenvolvedores podem checar e-mail e suprirem suas necessidades de privacidade durante o horário de trabalho. Na Figura 3-2 (adaptada de WAKE, 2001, p. 57, il.), podemos observar um exemplo de ambiente de trabalho aberto.

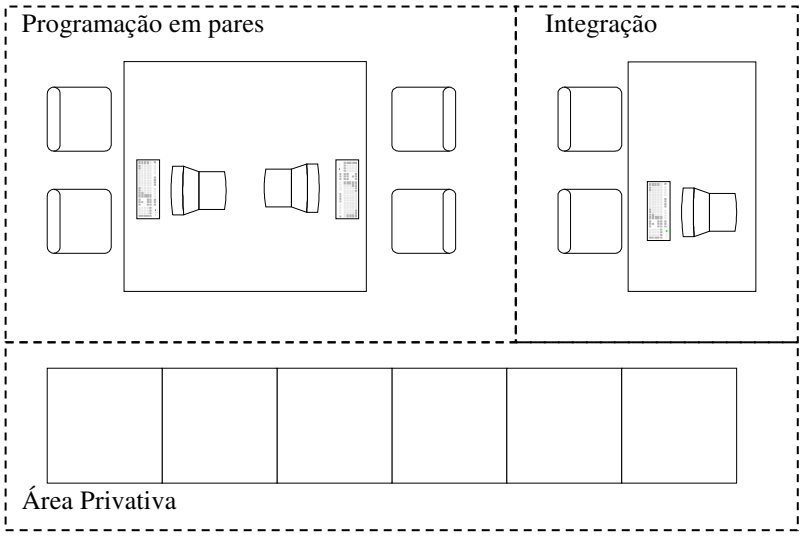

Figura 3-2 Ambiente de trabalho aberto

3.4.4 Trabalho Energizado

Para que o trabalho flua, só raramente se deve fazer uso de horas-extras. O horário de trabalho deve ser regular, de oito horas diárias. Cobrar exageradamente de um membro da equipe só vai desgastá-lo ao longo do tempo, acarretando até problemas de saúde, que podem levá-lo a se ausentar. A atividade de desenvolvimento requer criatividade, a qual, por sua vez, exige corpo e mente descansados e saudáveis.

Tradicionalmente, as horas-extras são usadas extensivamente para alcançar prazos combinados, mas segundo Demarco (2001 p. 63, apud TELES; 2005, p. 123-124, tradução nossa), "hora-extra por um longo período é uma técnica de redução de produtividade. Reduz o efeito de cada hora trabalhada."

Também fazem parte da prática do trabalho energizado os prazos fixos. Quando se trabalha com prazos fixos, é preciso ter ciência da capacidade de realização da equipe durante o tempo estipulado. Estas medidas são aferidas e atualizadas regularmente, durante os ciclos semanal e trimestral, e podem ser divulgadas através de cartazes, compondo o ambiente de trabalho informativo.

Faz parte do processo de estimativa e planejamento o que Beck e Fowler (2001) chamam de *Yesterday's weather* (a previsão do tempo de ontem). A equipe de desenvolvimento só pode se comprometer a realizar a quantidade de trabalho que ela conseguir fazer na iteração passada.

Como explicado anteriormente, XP utiliza o escopo do projeto como variável de controle. Quando as coisas dão errado e o prazo não pode ser alcançado, as tarefas a serem realizadas são priorizadas novamente pelo especialista de negócio, e o escopo é ajustado para que o prazo seja alcançado. "Quando estiver sobrecarregado, não pense nisso como se não tivesse tempo suficiente; pense nisso como tendo coisas demais para fazer. Você não pode se conceder mais tempo, mas pode se permitir fazer menos coisas (BECK; FOWLER, 2001, p. 25, tradução nossa)."

3.4.5 Programação em Par

A programação em pares consiste em "dois desenvolvedores trabalhando lado a lado em um computador, colaborando para o mesmo design, algoritmo, código ou teste".

(WILLIAMS et al. 2000, p. 19, tradução nossa). Algumas regras são estipuladas para se programar em pares:

1. Dois papéis devem ser observados: o piloto e o navegador, como num rali. O piloto tem o teclado e mouse em mãos; o navegador observa, corrige, sugere e toma a iniciativa quando se sentir mais a vontade. Os postos devem ser trocados regularmente.

2. Dentro da equipe, os pares devem se revezar, o que é chamado de *move people around*.

A revisão constante de código acarreta sensível redução na taxa de defeitos e os brainstorms que ocorrem entre a dupla, durante a codificação, eleva a qualidade do design e possibilita a resolução de problemas mais rapidamente. Além disso, por estarem trabalhando em pares, cada um é responsável por chamar a atenção do outro para se concentrarem, elevando a quantidade de tempo efetivo de trabalho. Checagem de e-mail e mensagens instantâneas são menos freqüentes e há maior satisfação por parte dos programadores. (COCKBURN; WILLIAMS, 2001).

O revezamento dos postos no par e entre os pares promove a difusão do conhecimento acerca do projeto além de aprimorar as habilidades de cada integrante do time. Cada um também pode contribuir com sua especialidade para o projeto, e ao mesmo tempo, ensinar um pouco do que sabe ao parceiro de programação. Isso ajuda a diminuir o impacto caso algum integrante da equipe venha a deixar o projeto, da mesma forma que auxilia a integração de um novo membro. Também pode se observar melhora na capacidade de comunicação dos membros da equipe (COCKBURN; WILLIAMS, 2001).

O que para gerentes tradicionais pode parecer contraproducente tem se mostrado qualitativamente vantajoso para as equipes que utilizam XP como abordagem para desenvolver software. Como demonstram Cockburn e Williams (2001), o aumento no custo de desenvolvimento para estes benefícios é de aproximadamente 15%. Este custo, no entanto é compensado em pouco tempo pelo barateamento nos custos de testes, QA e suporte técnico.

3.4.6 Estórias de usuário

As estórias de usuário são importantes instrumentos para lidar com os requisitos quando se trabalha com XP. Elas reúnem duas das principais características do desenvolvimento ágil de software: o direito dos clientes de receberem dos programadores a

maior valor agregado possível e o direito dos programadores de saberem o que o cliente necessita. Elas consistem na descrição sucinta de alguma funcionalidade, comportamento ou característica do sistema em desenvolvimento e devem ser escritas e priorizadas pelos próprios clientes, geralmente em cartões de papel. (JEFFRIES; ANDERSON; HENDRICKSON, 2001).

Os cartões de estórias de usuário servem para descrever tanto requisitos funcionais como requisitos não funcionais de software.

O uso de cartões para se registrar as estórias procura promover maior interação entre a equipe de desenvolvimento e o cliente, e são usados durante todo o ciclo de desenvolvimento. Num primeiro momento, o cliente escreve as estórias. Posteriormente, as estórias são utilizadas para definir testes de aceitação, que determinam a conformidade das descrições do cliente com a implementação dos desenvolvedores. No decorrer das iterações, os cartões também são utilizados para informar o progresso do projeto, normalmente, as equipes elaboram quadros informativos organizando os cartões, por exemplo, em “concluído”, “em andamento” e “por fazer”. (POPPENDIECK; POPPENDIECK, 2003).

3.4.7 Ciclo semanal

Uma semana é o tempo recomendado para uma iteração. No início da semana a equipe e o cliente irão:

1. Verificar o progresso do projeto, e estipular a capacidade da equipe para a próxima iteração.
2. O cliente deve selecionar estórias possíveis de serem implementadas dentro da próxima semana.
3. As estórias selecionadas pelo cliente devem ser transpostas em tarefas pelos membros da equipe. Cada membro estima e assume a responsabilidade por algumas tarefas.

Como citado anteriormente, os testes de aceitação servem para verificar a conformidade entre a descrição do cliente e a realização dos programadores. Verificar o progresso atual e planejar a iteração seguinte envolve, portanto, escrever testes de aceitação, de preferência automatizados, que deverão ser executados sem problemas ao fim da iteração. Os desenvolvedores devem auxiliar os clientes a escrever estes testes. Ferramentas como FIT

(*Framework for Integrated Test*) auxiliam a comunicação entre os clientes e os desenvolvedores na sua elaboração (WESTPHAL, 2005).

A meta principal é obter, ao fim de cada semana, uma versão funcional do software, que pode ser instalada facilmente. (BECK; ANDRES, 2004).

A priorização e seleção das estórias a serem implementadas devem ser feitas pelo cliente, ou por um especialista de negócio. Desta forma, é garantido que o cliente irá obter retorno maior e mais rápido sobre seu investimento.

Os cartões também são utilizados no processo de estimativa e priorização da execução das atividades. Uma estória não deve custar ao programador mais tempo que uma iteração, geralmente uma semana. Estórias grandes devem ser divididas em estórias menores. Os ciclos semanais, explicados a seguir, promovem o encurtamento do ciclo feedback. XP considera que os processos de especificação e análise de requisitos não devem ser realizados todos no início do projeto, mas sim de forma gradual e contínua. Esta abordagem, baseada no feedback rápido e regular, auxilia o aprendizado sobre o software a ser produzido, tanto por parte dos desenvolvedores como do cliente.

Após definidas as tarefas que, uma vez completas, trarão a estória "à vida", cada integrante deve assumir a responsabilidade pelas tarefas que estimou. É a maneira mais sensata de se proceder. Cada programador assume o compromisso de realizar suas atividades de acordo com sua capacidade. Por sua vez, a capacidade da equipe e de cada desenvolvedor é definida pelo seu desempenho na iteração passada. Naturalmente, quando necessário, o gerente pode intervir para que estórias não fiquem "sem dono". Entretanto, a abordagem impositiva não é aconselhada por razões motivacionais.

Deve-se procurar também fazer com que os desenvolvedores não busquem apenas tarefas com as quais se sentem mais confortáveis, por exemplo, por causa das suas especialidades. A estimativa das tarefas e a responsabilidade sobre elas são individuais, mas o desenvolvimento é feito em pares, nesse momento haverá espaço para os especialistas atuarem em seus respectivos campos. Distribuir de forma homogênea as tarefas contribui para o nivelamento da equipe. (BECK; ANDRES, 2004).

3.4.8 Ciclo trimestral

Ciclos trimestrais também devem demarcar o planejamento. Nesse período, deve se elaborar um plano mais abrangente, que trata dos temas a serem desenvolvidos nos três meses que se seguirão. Também é um ótimo momento para se realizar retrospectivas e reflexões sobre os problemas no processo e no projeto, e encontrar maneiras de solucioná-los (COCKBURN, 2002; TELES, 2004).

Novamente, o papel do gerente é mediar interações entre os clientes e a equipe de desenvolvimento. Ele também deve se preocupar com os fatores externos à equipe e solucionar quaisquer problemas.

Os temas definidos para o trimestre serão especificados em forma de estórias, que virão novamente à tona em cada ciclo semanal de desenvolvimento.

3.4.9 Folga (*slack*)

A prática da folga (*slack*) implica incluir sempre nos planos uma margem de segurança de estórias que podem ser deixadas de lado, caso o cronograma não possa ser cumprido. Ela envolve também a solidificação da relação de confiança entre os membros da equipe, o gerente e o cliente.

Com o decorrer do projeto, as estimativas dos programadores tornar-se-ão mais precisas, mas ter uma folga é importante para contornar situações problemáticas e riscos que não forem evidenciados anteriormente. XP não tem práticas que tratem diretamente do gerenciamento e monitoramento de riscos. Mesmo assim, diversas práticas contribuem para a redução do impacto dos principais riscos nos projetos de software, como o turnover. O gerente deve evidenciar os riscos aos integrantes da equipe, para que eles possam prevenir-se ou combater os impactos negativos, caso os riscos venham a se tornar realidade.

3.4.10 *Build* em dez minutos

O *Build* do sistema completo, juntamente com a execução de todos os testes não deve exceder dez minutos. O primeiro passo é automatizar o processo de compilação e testes do sistema. Quando o processo exige intervenção manual ele tende a ser executado menos frequentemente. (BECK; ANDRES, 2004).

O fundamento desta prática é o valor do feedback. Caso a compilação e testes demorem muito tempo, haverá menos feedback sobre o trabalho realizado, e a quantidade de erros e stress aumentará.

3.4.11 Integração contínua

A prática da integração contínua é uma das mais valiosas da metodologia. Ela transforma o processo de integração do projeto, normalmente caótico e imprevisível, num processo contínuo, onde as novas partes são integradas ao corpo principal assim que são feitas, várias vezes ao dia. Dessa forma, erros introduzidos pelas novas incorporações são rapidamente percebidos e corrigidos.

A prática da integração contínua envolve diversas outras técnicas e boas práticas de desenvolvimento, entre elas o *build* rápido, discutido anteriormente. Deve-se (FOWLER, 2006):

- Manter um repositório único de código;
- Automatizar o *build* do projeto e torná-lo auto-testável;
- Introduzir código no repositório com regularidade, pelo menos diária, e por todos os integrantes;
- Toda mudança no repositório deve provocar um *build*, que deve ser executado rapidamente;
- Testar numa cópia do ambiente de produção;
- Tornar de fácil acesso todas as informações e artefatos acerca dos testes e da integração;
- Automatizar o processo de instalação (*deployment*).

A introdução da integração contínua no ambiente de desenvolvimento promove diversos benefícios, quais sejam:

- Defeitos introduzidos são evidenciados pelo processo de *build* automatizado que ocorre sempre que algum código novo entra no repositório.
- O fluxo no desenvolvimento, devido ao processo de integração contínua eleva a qualidade do software desenvolvido, e exige dos desenvolvedores mais comunicação, à medida que ocorre qualquer problema para a continuidade do processo.

A prática também é o primeiro passo para aumentar a freqüência nas entregas e instalações nos clientes.

> Instalar [*deploy*] freqüentemente é valoroso porque permite aos seus usuários adquirir as funcionalidades mais rapidamente, dar feedback sobre estas funcionalidades, e tornar o ciclo de desenvolvimento mais colaborativo. Isso ajuda a quebrar as barreiras entre os clientes e os desenvolvedores – barreiras que eu acredito serem as maiores barreiras para o sucesso no desenvolvimento de software. (FOWLER, 2006, tradução nossa).

3.4.12 Programação orientada a testes

A programação orientada a testes, além de uma simples disciplina de testes, ajuda a (BECK; ANDRES 2004):

- restringir o escopo;
- diminuir o acoplamento e aumentar a coesão da arquitetura;
- aumentar a confiança entre os desenvolvedores; e
- estabelecer um ritmo sustentável de desenvolvimento.

Tudo isso é alcançado produzindo código de testes antes de escrever o programa propriamente dito, não se pode fazer nenhuma modificação sem antes escrever um teste que evidencie o novo comportamento.

O escopo é reduzido à medida que só se escreve código suficiente para passar nos testes. Devem-se evitar os desperdícios e os casos que não são importantes para aquela funcionalidade.

De acordo com Beck e Andres (2004, p. 50, tradução nossa): “se está difícil de escrever um teste, é sinal que você tem um problema no design, não um problema nos testes. Código fracamente acoplado e altamente coeso é fácil de testar”.

A bateria de testes automatizados também serve para documentar o sistema. Além do próprio programa em funcionamento, os testes são excelentes para descrever o comportamento dos componentes do software. Ao trabalhar sobre um código testado, os integrantes da equipe não se preocupam com a autoria do código. Não se trata de desprezar a rastreabilidade dos artefatos e a troca de informações entre os participantes do projeto, ao contrário, XP preza pela comunicação acima de tudo. Trata-se de confiança no trabalho dos colegas, cujo código é testado antes mesmo de ser produzido.

A disciplina exigida pela prática da programação orientada a testes traz à atividade de desenvolvimento uma cadência característica: escrever um teste que falha, fazê-lo passar,

refatorar o código. Dessa forma, o desenvolvedor sempre sabe o que fazer depois que acaba uma tarefa (BECK; ANDRES, 2004).

3.4.13 Design incremental

Baseada no valor da simplicidade, XP defende que investimentos no design não devem ser feitos muito cedo no desenvolvimento do projeto. De forma diferente, o design deve ser realizado em passos pequenos, deve evoluir junto com a compreensão da equipe e do cliente sobre o problema tratado pelo software em desenvolvimento.

> O conselho aos times XP não é para minimizar o investimento em design em curto prazo, mas para manter o investimento em design na proporção das necessidades do sistema até aquele momento. [...] Design Incremental sugere que a hora mais efetiva para realizar o design é aquela indicada pela "luz da experiência". (BECK; ANDRES, 2004, p.52, tradução nossa).

Deve-se desenvolver a arquitetura apenas o suficiente para acomodar as funcionalidades previstas para o curto prazo. Grandes antecipações apenas introduzem complexidade desnecessária à arquitetura. Código complexo é mais difícil de manter do que código simples. Portanto, antecipar o desenvolvimento e as decisões arquiteturais para acomodar funcionalidades as quais não serão desenvolvidas em curto prazo só aumenta os custos de manutenção da base de código em desenvolvimento.

Diversas práticas auxiliam o Design Incremental: a integração contínua, as práticas de testes e a refatoração. Essas práticas possibilitam a chamada curva de custo de mudança constante, defendida pela metodologia. Ou, segundo Ambler (2006c, tradução nossa), elas "focam o lado esquerdo, virtuoso da curva tradicional [exponencial]".

Os testes automatizados e a disciplina de testar antes de implementar a solução promovem a segurança necessária para evoluir a arquitetura do software mais tarde no desenvolvimento. A Integração Contínua possibilita a sincronização de todo o time de desenvolvimento, e gera menos preocupações acerca da integração dos componentes. A refatoração do código funciona mantendo o código simples e fácil de entender, de forma que o próximo a trabalhar sobre aquele código possa ser mais eficiente e eficaz. (FOWLER, 2004).

3.4.14 Práticas do corolário

As práticas do corolário são mais específicas e podem ser implementadas independentemente. Elas também são mais avançadas e devem ser utilizadas com cautela, de preferência quando as práticas primárias já constituírem o dia-a-dia da equipe. É importante observar os efeitos causados pela adoção de uma prática. Por exemplo, não faz sentido tentar gerar versões do software diariamente se a quantidade de defeitos não está num patamar baixo, isso só traria mais caos ao projeto. Alguns exemplos de práticas do corolário são descritos por Beck e Andres (2004) e tratam:

- do envolvimento do cliente com a equipe;
- da instalação gradual da aplicação;
- da continuidade e do desenvolvimento das equipes;
- da resolução de problemas utilizando *Root-Cause Analysis*;
- do gerenciamento do código;
- de contratos de escopo negociável; e
- de questões financeiras do projeto de desenvolvimento de software.

As práticas do corolário não serão descritas, pois fogem ao foco principal do trabalho que é a comparação entre os princípios regentes do gerenciamento na Toyota e a abordagem XP para o desenvolvimento de software.

3.5 CONSIDERAÇÕES FINAIS

Partindo do abstrato para o concreto, foram descritos os valores, princípios e práticas que compõem XP. Apesar de observarmos um grande foco nas práticas de desenvolvimento e gestão, são de extrema importância os valores que motivam essas práticas e os princípios que auxiliam a entendê-las. Nem sempre haverá uma prática específica para tratar de problemas que possam advir durante um projeto de desenvolvimento de software. Nessa hora, é preciso manter-se fiel aos valores regentes para buscar soluções.

Também é importante salientar o fato de que XP não aborda diretamente diversos aspectos do processo de desenvolvimento de software. Surge daí uma das discussões na comunidade XP mundial: deve-se ou não chamar XP de metodologia. Normalmente, a discussão encerra-se ao trazer à tona a necessidade apresentada pelo mercado de melhorar o

processo de desenvolvimento de software, e como XP tem ajudado nessa melhoria. Ser ou não uma metodologia não viria então ao caso.

É importante perceber a concordância da abordagem XP com o Manifesto Ágil (BECK et al., 2001, tradução nossa), que enuncia:

> [...] valorizar mais:
> as pessoas e a interação entre elas, que os processos e ferramentas,
> software em funcionamento, que a documentação extensiva,
> a colaboração do cliente, que a negociação de contratos,
> responder à mudança, que seguir um plano.

A valorização das pessoas e o foco no cliente e sua satisfação são as principais semelhanças entre XP e a maneira Toyota de produzir, regida pelos princípios já descritos. No capítulo seguinte, serão evidenciados outros pontos de concordância, mostrando como XP já se apresenta bastante adequado aos princípios de gerenciamentos descritos por Liker (2003). Será explicitado como as práticas da abordagem ágil de desenvolvimento de software podem representar implementações dos princípios do *Toyota Way*. Alguns dos princípios, no entanto, não têm relação clara e direta com práticas de XP. Na medida do possível, serão sugeridas outras práticas ou adaptações de práticas existentes para serem inclusas no dia-a-dia das equipes.

Capítulo 4
EXTREME PROGRAMMING À LUZ DO *TOYOTA WAY*

Do you value the practices, or do you practice the values?
WILLEM VAN DEN ENDE, agile software developer and COACH

Este capítulo procura estabelecer uma comparação entre os princípios de gerenciamento da Toyota e a abordagem XP para o desenvolvimento de software, de forma a demonstrar a aplicabilidade dos citados princípios no desenvolvimento de software. Pode ser percebida a influência direta da Toyota e seu sistema de produção também no XP, na reedição da publicação que deu origem à metodologia, Beck e Andres (2004) incluem referências ao TPS. O *Toyota Way* servirá como ponto inicial para a comparação; para cada princípio serão mostradas práticas do desenvolvimento ágil de software utilizadas em XP, que são alinhadas aos princípios, bem como características de XP nas quais é possível observar o alinhamento. Caso não seja possível o relacionamento direto, serão buscadas possíveis adaptações ou práticas ágeis que possam ser incluídas no dia-a-dia da equipe de desenvolvimento e mantenham-se alinhadas a XP.

4.1 FILOSOFIA

De forma geral, as abordagens ágeis de desenvolvimento e gerência são baseadas em fundamentos ideológicos, valores que regem a atuação da equipe durante o desenvolvimento de software. Não há apenas fundamentos filosóficos: a exemplo do manifesto ágil, a abordagem XP compartilha de valores semelhantes aos da gigante japonesa. *Extreme Programming* é orientada pela necessidade apresentada pelo cliente e busca sempre formas de relacionar a produção da equipe com o valor gerado.

Apesar de ser sempre exigidos, do integrante da equipe, perícia e dedicação na realização das práticas de desenvolvimento, sempre há a preocupação de "praticar os valores" vigentes mais do que "valorizar as práticas" de desenvolvimento. Em sua totalidade, as práticas procuram refletir a adoção dos valores, mesmo porque não são representações absolutas e imutáveis, e devem ser moldadas e adaptadas para a realidade de cada equipe, de cada projeto. Haverá momentos em que utilizar uma prática pode ser prejudicial ao

desenvolvimento do projeto e ao aumento da produtividade, cabe ao gerente apontar essa divergência e, com o auxilio da equipe, contornar essa situação buscando formas de seguir melhor os valores e princípios vigentes, formas de ser mais ágil.

4.2 PROCESSO

Com o grupo de princípios que tratam do processo produtivo em si, o qual concentra a maioria dos princípios de gerenciamento, XP se alinha totalmente.

4.2.1 Fluxo Contínuo

Extreme Programming compartilha do mesmo princípio de estabelecer um fluxo contínuo de produção. No caso do desenvolvimento de software, este fluxo traduz-se em entregas regulares e no desenvolvimento em pequenos passos.

A prática da Integração Contínua é o maior exemplo de estabelecimento do fluxo no desenvolvimento de software. Ela mantém o foco na produção em pequenos passos e na realização freqüente de testes completos no sistema, sempre que há alguma mudança no repositório de código. A prática exige disciplina da equipe e funciona exatamente como sugere o *Toyota Way*: estabelecendo um processo contínuo que traz à superfície os problemas. No caso da produção de software, a integração freqüente das partes evidencia os problemas introduzidos na aplicação em desenvolvimento assim que eles entram no código.

O uso de entregas freqüentes e iterações curtas também exemplificam o estabelecimento da regularidade no desenvolvimento de software. No desenvolvimento ágil de software, o estabelecimento do fluxo contínuo traz benefícios e ganhos na flexibilidade e produtividade da equipe, bem como a possibilidade de embutir o controle de qualidade no processo produtivo, no caso de XP, a introdução de testes automatizados, de preferência criados antes do código de produção.

4.2.2 Sistemas *Pull*

A utilização de Sistemas *Pull* também está presente no desenvolvimento ágil de software com XP. Duas práticas principais levam a produção de software a funcionar de acordo com a demanda: o uso de estórias de usuário e a programação orientada a testes.

Durante o planejamento da iteração, os desenvolvedores estimam e assumem a responsabilidade pelas estórias. Durante o desenvolvimento, elas são realizadas na ordem de prioridade definida pelo cliente, procurando obter maior e mais rápido retorno sobre o investimento realizado. A dinâmica do uso das estórias assemelha-se bastante com o sistema de *kanban* utilizado na Toyota: eles indicam à equipe o que está por fazer.

Num nível mais específico, o desenvolvimento orientado a testes estabelece a demanda através do desenvolvimento dos testes automatizados, que deve ser suprida com a produção do código que atenda às necessidades impostas pela suíte de testes.

Os dois mecanismos demonstram como evitar a superprodução, aqui representada pela introdução de complexidade adicional no software, e como estabelecer uma ligação direta da produção com a demanda.

4.2.3 Ritmo Sustentável (*heijunka*)

O estabelecimento do fluxo regular de trabalho auxilia a evidenciar problemas no processo produtivo. Além disso, estabelecer um ritmo sustentável de produção de software também é importante para não sobrecarregar os desenvolvedores.

A prática do trabalho energizado, a qual já foi chamada de Ritmo Sustentável (TELES, 2004) e *40-hours week* (BECK, 2001), demonstra como XP procura regularizar a carga de trabalho. As necessidades do cliente são avaliadas e as estórias de usuário são revistas e estimadas novamente sempre que é possível fazê-lo de forma mais aprimorada. Com o tempo, quando as estimativas dos desenvolvedores ficam mais acuradas, é possível se beneficiar do trabalho regular da mesma forma que as montadoras japonesas.

4.2.4 Qualidade embutida (*jidoka*)

São os testes automatizados os maiores representantes do controle de qualidade embutido no processo de desenvolvimento de software, juntamente com a Integração Contínua, que recomenda automatizar o processo de *build* da aplicação em desenvolvimento, e torná-lo auto-testável.

Ao invés de estabelecer uma única equipe responsável pela garantia de qualidade, toda a equipe de desenvolvimento faz o papel de fiscal da qualidade. Cada desenvolvedor escreve testes para o código que ele mesmo produz.

A integração contínua complementa a aderência ao princípio da Toyota. Nela está o requisito de parar a produção para corrigir qualquer problema que não exceda a margem de tolerância. As práticas mais avançadas da abordagem XP só podem ser introduzidas depois que a incidência de erros for baixíssima. Uma prática que exemplifica esse rigor é a Instalação Diária, prática do corolário que recomenda colocar o sistema em produção diariamente, para diminuir a lacuna entre o que está em desenvolvimento e o que o cliente está usando. Gerar uma nova versão do sistema diariamente, com novas funcionalidades, pode ser catastrófico caso a taxa de erros não esteja num nível baixo o suficiente.

4.2.5 Atividades Padronizadas

O desenvolvimento de software ágil é relacionado ao princípio da humanidade, descrito por Beck e Andres (2004). Este, por sua vez, está intimamente ligado ao desenvolvimento pessoal dos integrantes da equipe. As práticas de desenvolvimento com XP são bem descritas e refletem experiências reais. Elas foram criadas e melhoradas em processos altamente participativos. A própria metodologia sofre bastante influência da comunidade em seu redor.

Práticas a exemplo da programação orientada a testes demonstram como se deve realizar o trabalho, definindo claramente, para todo o desenrolar do projeto, o que o programador deve fazer quando acabar cada etapa (BECK; ANDRES, 2004).

Outras práticas, como a reflexão, durante as reuniões dos ciclos semanais e trimestrais, contribuem para o alinhamento da equipe, em busca das melhores formas do desenvolvimento e do aprimoramento das práticas atuais.

Algumas práticas do corolário tratam exatamente do gerenciamento de código, de como manter uma base única e compartilhada de código, utilizar padrões de codificação, etc. sempre considerando que peculiaridades são mais difíceis de gerenciar do que comportamentos e atividades padronizadas.

4.2.6 Controle Visual

Como descrito por Jeffries (2004) e Cockburn (2002), as metodologias ágeis utilizam cartazes e quadros nas paredes do ambiente de trabalho para manter todos os interessados

informados. A comunicação transparente e direta é fortemente valorizada e incentivada quando se trabalha com XP.

A principal prática que se relaciona com o princípio de utilizar o controle visual para que os erros não fiquem escondidos é a prática do ambiente de trabalho informativo. De forma semelhante à Toyota, os adeptos de XP confiam na eficácia de painéis em detrimento de relatórios na intranet da empresa. O argumento é o mesmo: certas informações não devem esperar que alguém vá em sua busca.

A utilização de estórias de usuários em cartões também favorece o controle visual. Como descrito por Poppendieck & Poppendieck (2003), os cartões servem ainda como informativos do progresso do projeto.

4.2.7 Tecnologia a serviço das pessoas

Extreme Programming tem como base o uso de tecnologias comprovadamente eficazes e eficientes. Faz parte da cultura ágil a busca por ferramentas simples, adequadas às necessidades da equipe. O uso de cartões de estórias, sem dúvida, é o maior exemplo disso. Ao invés de utilizar softwares de planejamento pesados, gráficos de Gantt, que não mostram a realidade, os gerentes ágeis preferem planejar de forma evolutiva utilizando cartões. Além de serem utilizados posteriormente para verificar o progresso do projeto, eles servem como uma mídia comum para desenvolvedores e clientes, que podem se comunicar de forma clara e eficiente.

Quando se trata da inclusão de uma nova tecnologia ao software em desenvolvimento, XP também lida com cautela. A adoção de uma nova idéia ao projeto passa por experimentos chamados *Spike Solutions* (JEFFRIES; ANDERSON; HENRICKSON, 2001), que auxiliam os integrantes da equipe a ponderar e realizar estimativas sobre o uso da novidade.

4.3 PESSOAS E PARCEIROS

Será demonstrado neste item como o desenvolvimento de software com XP se alinha com o grupo de princípios que diz respeito ao trato com pessoas e parceiros.

4.3.1 Líderes

A forma como a Toyota cultiva seus líderes e o que deles espera, se assemelha bastante à abordagem XP. O *Coach* na equipe de desenvolvimento de software deve ter exatamente as mesmas funções que um líder de equipe na gigante automobilística. A forma de atuar, o elevado conhecimento técnico e dos processos envolvidos na atividade são pontos fortes de congruência.

O *Coach* não é a única figura gerencial da equipe XP. Sempre deve haver um gerente propriamente dito, que viabiliza o trabalho dos desenvolvedores. Se por um lado, a postura do gerente tende para o lado de facilitador de grupo, se afastando da maneira Toyota; por outro a observação e o entendimento da situação fazem parte do seu dia-a-dia.

Considerando que a figura do líder dentro da equipe XP está dividia entre o *coach* e o gerente, pode-se afirmar que o cultivo das lideranças e a postura esperada desses líderes são muito similares ao que defende o nono princípio de gerenciamento descrito por Liker (2003).

4.3.2 Pessoas e Times

Quando se trata da formação das pessoas e dos times, há uma forte ligação entre os princípios difundidos na Toyota e a *Extreme Programming*. A prática da programação em pares reflete a preocupação com o nivelamento dos integrantes da equipe. O objetivo da rotação de pares é sempre elevar o nível de conhecimento da equipe. Dessa forma, são desenvolvidos profissionais com elevado conhecimento técnico.

Do mesmo modo, outros mecanismos permitem a responsabilização e a contribuição individual de cada componente da equipe ao projeto. As tarefas definidas para se implementar uma estória de usuário são sempre assumidas por um integrante, mesmo que ele trabalhe em par. A dinâmica da programação em pares e a sua rotação também possibilitam que cada desenvolvedor colabore com sua área de especialidade.

A adesão de cada integrante da equipe à abordagem de desenvolvimento é naturalmente essencial. É papel do gerente e do *coach* mostrarem à equipe como trabalhar e alinhar os objetivos de cada integrante ao objetivo do projeto.

4.3.3 Respeito aos parceiros e fornecedores

Não há, em *Extreme Programming*, indícios de semelhança com a abordagem da Toyota para lidar com empresas e colaboradores externos. XP não é propriamente uma metodologia, e não há a preocupação de tratar de todas as nuances do desenvolvimento de software em detalhes. Uma das lacunas na abordagem é o detalhamento dos processos e práticas de parceria externa. XP é mais endógena.

O aspecto dessa abordagem ágil de desenvolvimento de software que trata da necessidade de aportes externos, principalmente de conhecimento e capital humano, é refletido na prática do time completo (BECK; ANDRES, 2004), e segue o mote de "não dar o peixe, mas ensinar a pescar". A razão disso é para que o conhecimento necessário pela equipe possa permanecer no projeto mesmo que o especialista tenha que se afastar.

Esta ótica se assemelha à abordagem da Toyota. Quando firma parcerias, a Toyota procura não depender completamente dos fornecedores, apesar de sempre manter relações estreitas e confiáveis. Dois pontos de concordância podem ser observados: a simbiose cultivada, quase que exigida, pela Toyota e a necessidade de reter o conhecimento desenvolvido. Por outro lado, não são percebidos, na abordagem fundada por Beck (1999), os desafios que a Toyota apresenta aos seus parceiros e fornecedores.

Para a congruência completa seria necessária a ampliação da abordagem XP para tratar de aquisições e contratações. Sem dúvida, este tratamento se daria na mesma linha já observada quando há necessidade de obtenção de conhecimento e aprimoramento técnico da equipe, buscando reter conhecimento mesmo depois de o especialista ter se afastado.

4.4 RESOLUÇÃO DE PROBLEMAS

Veremos, a seguir, sob quais aspectos XP assemelha-se aos três últimos princípios de gerenciamento. Eles representam o topo da pirâmide de diretrizes que levaram a Toyota ao estágio de empresa que aprende (*learning enterprise*), e tratam da resolução de problemas e da melhoria contínua dos processos da empresa.

4.4.1 Veja com seus próprios olhos (*genchi genbutsu*)

Diversos aspectos do *genchi genbutsu* podem ser percebidos em XP. A presença do *coach* e do gerente, as reuniões em pé, o ambiente de trabalho informativo e a obsessão por testes são exemplos que compõem o que poderia ser a visão ágil da prática japonesa.

A prática de testes sempre foi um dos fundamentos de XP. Há defensores mais radicais da metodologia que defendem apenas confiar que determinado código funciona quando houver testes para ele (JEFFRIES; ANDERSON; HENDRICKSON, 2001). Esta obsessão por testes, já foi difundida por equipes XP, (JUNIT, 2006), está muito ligada ao aspecto do *genchi genbutsu* de ser responsável pela informação que se passa adiante.

O ambiente de trabalho informativo auxilia o gerente a diagnosticar quaisquer problemas com a equipe. O processo de elaboração e análise de métricas, mesmo de métricas simples, é usado extensivamente na tentativa de medir ganhos na produção e eventuais necessidades de intervenção devido a problemas.

A presença do *coach* e do gerente, e sua cobrança regular, procura não atrapalhar o desenvolvimento do projeto, mas manter a todos informados. Outras práticas já difundidas, como a reunião em pé, também utilizada no SCRUM, têm como fundamento a comunicação e o diagnóstico de problemas. As soluções para problemas encontrados nas reuniões diárias não devem ser resolvidos durante a reunião, que deve ser curta, mas em outra ocasião.

4.4.2 Tomada de decisão (*nemawashi*)

O processo de tomada de decisão utilizado na Toyota é baseado em decidir por consenso, fundamentado por muita informação e protótipos, para depois por em prática de forma rápida. Pode-se encontrar pontos em comum e pontos de discordância do XP, ao observar os cinco passos que compõem o *nemawashi*, descritos na seção 2.4.2.

A parte que define como encontrar o problema e como entender as causas que explicam o seu aparecimento é semelhante. A *Root Cause Analysis* (RCA) também é descrita por Beck e Andres (2004) dentre as práticas do corolário.

No que tange à consideração abrangente de alternativas, XP se distancia da abordagem recomendada pela Toyota. É possível agregar a XP práticas para o desenvolvimento paralelo, e prototipação extensiva até que se encontre a melhor forma de tratar uma questão do projeto. Entretanto, a metodologia prioriza a simplicidade e o design incremental, que procuram uma

solução simples para depois aprimorá-la. Como XP foi criada para tratar especialmente de requisitos vagos, o que é comum no desenvolvimento de software, não é viável conceber planos de longo prazo para a arquitetura do software, ela deve sempre evoluir.

A busca pelo consenso também está presente na metodologia ágil, que prima pela comunicação e o feedback. Práticas como o sentando juntos e o time completo procuram envolver todos os interessados no projeto para que seu desenvolvimento seja consensual. Sob essa ótica podemos afirmar a semelhança entre as duas abordagens.

O uso de veículos de comunicação eficientes é um ponto de congruência entre as duas abordagens, naturalmente. *Extreme Programming* tem como um de seus pilares o valor da comunicação. Beck e Andres (2004) acreditam que a grande maioria dos problemas surgidos nos projetos de software é relacionada com a defasagem na maneira e freqüência de comunicação. Dificilmente esses problemas serão devidos a carências de conhecimento técnico.

O desenvolvimento paralelo de alternativas, descrito por Ward (1995), é um ponto com o qual a abordagem XP não se assemelha. O uso de *spike solutions*, por outro lado, poderia ser ampliado e utilizado com outra ênfase: não apenas para auxiliar o julgamento e a estimativa sobre o impacto de uma decisão a ser provavelmente tomada, mas para a investigação de mais possibilidades.

4.4.3 Organização que aprende (*hansei*) e melhora continuamente (*kaizen*)

Principalmente a partir da segunda edição do livro que iniciou a divulgação de *Extreme Programming*, percebe-se o foco dos autores no aprimoramento das pessoas e no ganho de produtividade contínuos. XP utiliza ferramentas semelhantes às da Toyota para possibilitar a melhoria dos processos.

O *hansei* japonês pode ser comparado com as retrospectivas descritas por Teles (2004) e Cockburn (2002), utilizadas nos marcos importantes do projeto para averiguar a eficácia de decisões tomadas anteriormente e evidenciar problemas no processo de desenvolvimento. Para auxiliar a investigação de problemas, a *Root Cause Analysis* também pode ser utilizada no desenvolvimento de software, como descrevem Beck e Andres (2004).

É descrita por Beck e Andres (2004), também, a necessidade de entrosamento de todos os níveis hierárquicos da empresa para o sucesso do projeto. Os executivos devem prover

confiança e motivação, ao passo que é legítimo fornecer, em retorno, formas de relacionar o investimento no projeto e a produção da equipe.

Além disso, a aplicação de Teoria das Restrições também é reportada por Beck e Andres (2004) como uma ferramenta para encontrar gargalos no desenvolvimento do software. Há de chegar um momento na busca por possibilidades de aprimoramento no processo, a partir do qual os gargalos não serão mais encontrados na equipe de desenvolvimento, mas em outro setor da empresa. Como tudo o que ocorre dentro da empresa, é extremamente necessário que, ao optar por utilizar XP, haja não só concordância, mas também o apoio da alta gerência, já que a sua adoção pode representar uma ruptura no paradigma atual.

Mesmo sem formalizar a aplicação do ciclo PDCA, XP tem como princípio o aprimoramento, defende que a busca pela melhoria deve ser diária. Claramente, os praticantes de *Extreme Programming* devem buscar aprender e melhorar continuamente, promovendo mudanças quando necessário, envolvendo inclusive os mais altos executivos da empresa.

4.5 CONSIDERAÇÕES FINAIS

Este capítulo procurou mostrar como o *Extreme Programming* é semelhante aos princípios de gerenciamento utilizados pela Toyota. Em outras palavras, como as práticas e nuances da metodologia ágil podem ser adequadas para trabalhar em acordo com o *Lean*.

Uma das distinções está na ausência do tratamento de aquisições na abordagem ágil. XP não descreve práticas específicas para lidar com terceirização e outras maneiras de suprir deficiências da equipe.

O outro aspecto sobre o qual XP não segue os princípios do *Toyota Way* é a consideração de diversas alternativas, e o desenvolvimento dessas possibilidades em paralelo até que se possa tomar uma decisão melhor abalizada. O planejamento longo e a execução rápida utilizados pela Toyota, à primeira vista, parecem se distanciar da abordagem XP, que utiliza planejamento evolutivo e constante. Entretanto, o planejamento extenso da Toyota é justificado pela necessidade de angariar informações acerca das alternativas de implementação. De forma semelhante, o desenvolvimento de software com XP busca o aprendizado sobre o problema em questão, ao passo que vai implementando o projeto. Apesar de distintas, as duas abordagens têm justificativas e objetivos semelhantes.

Com os demais princípios, XP se alinha e sempre é possível encontrar uma prática ou aspecto da abordagem ágil que demonstre a aplicação do princípio em evidência.

A Tabela 4-1 ilustra a os princípios do *Toyota Way* e os aspectos de *Extreme Programming* que demonstram o alinhamento entre a abordagem ágil e as diretrizes da gigante japonesa. Quando não há total concordância com os princípios, possíveis adaptações a XP são brevemente descritas.

Toyota Way	***Extreme Programming***	**Adaptação necessária**
Filosofia de Longo Prazo	Valores e princípios que fundamentam as práticas	
Fluxo Contínuo	Integração contínua, entregas freqüentes	
Sistemas *Pull*	Estórias de usuário, programação orientada a testes	
Ritmo Sustentável	Trabalho energizado	
Qualidade embutida	Testes automatizados, integração contínua	
Atividades Padronizadas	Programação orientada a testes, padrões de codificação	
Controle Visual	Ambiente de trabalho informativo	
Tecnologia a serviço das pessoas	Ferramentas simples, estórias de usuário, *spike solutions*	
Líderes	O papel do *Coach* e do gerente	
Pessoas e Times	Programação em pares	
Respeito aos parceiros e fornecedores	Time completo	Introduzir o tratamento de aquisições
Veja com seus próprios olhos	Ambiente de trabalho informativo, reuniões em pé, obsessão por testes	
Tomada de decisão	*Root Cause Analysis*, valorização da comunicação e feedback, time completo	Adaptação das *spike solutions*
Organização que aprende e evolui	Retrospectivas, ciclo semanal e trimestral, teoria das restrições	

Tabela 4-1 Relação entre o *Toyota Way* e *Extreme Programming*

Capítulo 5
CONCLUSÕES E TRABALHOS FUTUROS

Parallels with other disciplines have to be applied with care. What works on the manufacturing floor may not work in the programming room and vice versa. However, other disciplines have been working on difficult people problems, like encouraging organizational change, for decades. Understanding the lessons they have learned can help accelerate application of XP.
KENT BECK

Este trabalho analisou a possível aderência do *Extreme Programming* aos princípios de gerenciamentos do *Toyota Way*. Após a descrição dos princípios utilizados pela gigante japonesa e da abordagem ágil para o desenvolvimento de software, procurou-se mostrar sob quais aspectos a metodologia ágil demonstrava aplicar as diretrizes de gerenciamento. Facetas de práticas utilizadas no dia-a-dia da equipe XP foram ressaltadas para ratificar essa concordância.

O sucesso da Toyota na indústria automobilística e o êxito demonstrado pelas abordagens ágeis de desenvolvimento foram a principal razão para a comparação. Por ser a mais famosa dentre as abordagens ágeis, XP foi escolhida para a confrontação, cujo objetivo é auxiliar o desenvolvimento e a aplicação das metodologias ágeis.

Sem dúvida, é possível alcançar melhorias espelhando-se no exemplo da Toyota. Os princípios de gerenciamento levaram a empresa ao nível de excelência atual. Eles são aplicados em todos os campos de atuação do grupo, e em todos os setores das empresas, da manufatura ao ramo de serviços.

Dentre os pontos similares encontrados na comparação, é importante destacar a semelhança com o grupo de princípios referente aos processos. Esta semelhança, no entanto, deve ser vista com cautela. A maioria das empresas que falham ao ingressar no mundo *Lean* fracassa por acreditar que apenas ferramentas e processos podem levar ao sucesso. Mais importante, porém é a aplicação de uma filosofia de longo prazo, e a adoção desta filosofia por todos os integrantes da empresa.

Da mesma forma são os métodos ágeis. É bastante provável que se consigam ganhos de produtividade e qualidade aplicando práticas ágeis consagradas como a programação em pares e a integração contínua. Entretanto, para alcançar o nível de excelência referido pelos gurus do movimento ágil, é preciso por em prática os valores que fundamentam suas

abordagens. Esta, sem dúvida, é a maior semelhança entre a abordagem Toyota e a do movimento ágil.

5.1 TRABALHOS FUTUROS

Quanto aos pontos onde não há aderência das práticas de XP aos princípios da Toyota, cabe estudar formas de preencher essas lacunas. A ausência de práticas para tratar de aquisições em XP sem dúvida pode ser objeto de estudo em trabalhos futuros.

A possibilidade de comparar criticamente quais características de *Extreme Programming* **não** aderem aos princípios da Toyota, demonstrando a importância dessa diferença, poderia complementar este trabalho.

A ausência de evidências práticas também pode ser salientada como uma dificuldade durante a pesquisa. Apesar da extensa literatura, seria importante a realização de experimentos para validar esta comparação teórica, sempre no sentido de contribuir com o aprimoramento das abordagens de desenvolvimento de software.

REFERÊNCIAS

AGILE ALLIANCE. **What Is Agile Software Development?**. Última modificação em 7 set. 2005. Disponível em <http://www.agilealliance.org/intro>. Acesso em: 15 maio 2006.

AMBLER, Scott. Crossing the Chasm. In: **Dr. Dobb's Journal ERCB**. San Francisco: Miller Freeman, May 2006. ISSN 1044-789X. Disponível em: <http://www.ddj.com/dept/architect/187200223>.

AMBLER, Scott. **Examining the Agile Cost of Change Curve**. 2006. Última modificação em 9 maio 2006. Disponível em: <http://www.agilemodeling.com/essays/costOfChange.htm>. Acesso em: 04 ago. 2006.

AMBLER, Scott. **Why Agile Software Development Techniques Work: Improved Feedback**. 2006. Última Modificação em 10 maio 2006. Disponível em: <http://www.ambysoft.com/essays/whyAgileWorksFeedback.html>. Acesso em: 04 ago. 2006.

BECK, Kent. **Extreme Programming Explained**: Embrace Change. Boston: Addison-Wesley, 1999. 224p. ISBN 201-61641-6.

BECK, Kent et al. **Manifesto for agile software development**. 2001. Disponível em: <http://agilemanifesto.org>. Acesso em: 1 ago. 2006.

BECK, Kent; ANDRES, Cynthia. **Extreme Programming Explained**: Embrace Change. 2nd. ed. Boston: Addison-Wesley, 2004. 224 p., ISBN 0-321-27865-8.

BECK, Kent; CUNNINGHAM, Ward. A laboratory for teaching object oriented thinking. In: ACM CONFERENCE ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES AND APPLICATIONS, 4., 1989, New Orleans. **Conference Proceedings**. New York: ACM, 1989.

BECK, Kent; FOWLER, Martin. **Planning Extreme Programming**. Boston: Addison-Wesley, 2001. 139 p.

COCKBURN, Alistair. **Agile software development**. Boston: Addison-Wesley, 2002. 278 p.

COCKBURN, Alistair; WILLIAMS, Laurie. The Costs and Benefits of Pair Programming. In: SUCCI, Giancarlo; MARCHESI, Michele. **Extreme Programming Examined**. Boston: Addison-Wesley, 2001. Como Conference proceedings da XP'2001, Villasimius, Sardinia, Itália.

CUNNINGHAM, Ward. **WikiWikiWeb**. 1995- . Disponível em: <http://c2.com/cgi/wiki>. Acesso em: jul. 2006.

DEMARCO, Tom. **Slack**: getting past burnout, busywork, and the myth of total efficiency. New York: Broadway Books, 2001. 227 p.

ELSSAMADISY, Amr. XP On A Large Project: A Developer's View. In: XP UNIVERSE, 1., 2001, Raleigh. [**Trabalhos Apresentados**]. 2001. Disponível em: <http://www.xpuniverse.com/2001/pdfs/EP202.pdf>.

FORD, Henry. (1926). **Today and Tomorrow**. Garden City, NY: Doubleday, Page & Company, 1926. Reprint Edition. Portland, OR: Productivity, 1988.

FOWLER, Martin. **Continuous Integration**. 2006. Última modificação em 1 maio 2006. Disponível em: <http://www.martinfowler.com/articles/continuousIntegration.html>. Acesso em: 3 ago. 2006.

FOWLER, Martin. Is Design Dead?. In: SUCCI, Giancarlo; MARCHESI, Michele. *Extreme* **Programming Examined**. Boston: Addison-Wesley, 2001. Como Conference proceedings da XP'2001, Villasimius, Sardinia, Itália. Última atualização do autor em maio 2004. Disponível em: <http://www.martinfowler.com/articles/designDead.html>. Acesso em: 04 ago. 2006

FOWLER, Martin. **The New Methodology**. Última modificação em 13 dez. 2005. Disponível em: <http://martinfowler.com/articles/newMethodology.html>. Acesso em: 28 mai. 2006

FOWLER, Martin; HIGHSMITH, Jim. Agile Manifesto. In: **Software Development**. San Francisco: Miller Freeman, Aug. 2001. Periódico absorvido pelo Dr. Dobb's Journal ERCB desde maio 2006. ISSN 1070-8588. Disponível em: <http://www.ddj.com/dept/architect/184414755>.

GENERAL MOTORS CORPORATION. **Mision Statement**. [mensagem possoal]. Mensagem recebida por <kvalcanti@gmail.com> em 2 jul. 2006.

JEFFRIES, Ron. **Big Visible Charts**. Março de 2004. Acessado em 24 de julho de 2006. Disponível em: <http://www.xprogramming.com/xpmag/BigVisibleCharts.htm>. Acesso em: ago. 2006.

JEFFRIES, Ron; ANDERSON, Ann; HENDRICKSON, Chet. **Extreme Programming Installed**. Boston: Addison-Wesley, 2001. 265 p., ISBN 201-70842-6

JUNIT. **JUnit Test Infected**: Programmers Love Writing Tests. 2006. Disponível em: <http://junit.sourceforge.net/doc/testinfected/testing.htm>. Acesso em: 26 set. 2006

LIKER, Jeffrey. **The Toyota Way**: 14 Management Principles from the World's Greatest Manufacturer, New York: McGraw-Hill, 2003. 352 p., ISBN 0-07-139231-9.

NAUR, Peter (Ed.); RANDELL, Brian (Ed.). **Software Engineering**: report on a conference… In: NATO CONFERENCE ON SOFTWARE ENGINEERING, Oct. 1968, Garmish, Germany. NATO Science Committee Garmish, Germany, Jan. 1969.

MORYEN, Roy. Agile Management and the Toyota Way for Software Project Management. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS, 3., 2005, Perth, Australia**. Conference Proceedings…**, Perth, 2005.

ORGANISATION INTERNATIONALE DES CONSTRUCTEURS D'AUTOMOBILES. **World motor vehicle production by manufacturer**: World ranking 2005. 2005. Disponível em: <http://www.oica.net/htdocs/statistics/tableaux2005/worldranking2005.pdf>. Acesso em 1 ago. 2006.

POPPENDIECK, Mary; POPPENDIECK Tom. **Lean Software Development**: An Agile Toolkit. Boston: Addison-Wesley, 2003. 240p., ISBN 0-321-15078-3.

STANDISH GROUP INTERNATIONAL, Inc., The, **Extreme Chaos**. Disponível em: <http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf>. Acesso em: 31 out. 2005.

TELES, Vinícius Manhães. **Extreme Programming**: aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. São Paulo: Novatec, 2004. 320 p.

TELES, Vinícius Manhães. **Um Estudo De Caso Da Adoção Das Práticas E Valores Do Extreme Programming**. Rio de Janeiro, 2005. Dissertação (Mestrado em Informática), Universidade Federal do Rio de Janeiro, 2005.180 f. Disponível em: <http://www.improveit.com.br/xp/dissertacaoXP.pdf>.

TELES, Vinícius Manhães. **Valores de XP (Coragem)**. [mensagem pessoal]. Mensagem enviada ao grupo de discussão XPRio, recebida por <kvalcanti@gmail.com> em 6 set. 2005. Disponível em: <http://groups.yahoo.com/group/xprio/message/2074>.

TIERNEY, Christine. Toyota, Honda predict gain. **The Detroit News online**. 2004. Disponível em: <http://www.automotivedigest.com/research/research_results.asp?sigstats_id=777>. Acesso em: 15 mai. 2006.

VAN CAUWENBERGHE, Pascal. **The Toyota Way of Managing Projects**. In: XPDAYS GERMANY, 2005. Karlsruhe, 2005. Disponível em: <http://xpdays.de/2005/sessions/The_Toyota_Way.html>. Acesso em: jun. 2006.

WAKE, William C.. **Extreme Programming Explored**. Boston: Addison-Wesley, 2001. 144p. ISBN 0-201-73397-8.

WARD, Allen et al. The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster. **Sloan Management Review**, Massachusetts: MIT, v. 36, n. 3, p. 43-61, Spring 1995. Periódico absorvido pelo MIT Sloan Management Review a partir de 2001.

WESTPHAL, Frank. **Testgetriebene Entwicklung mit JUnit & FIT**: Wie Software änderbar bleibt. Heidelberg, Alemanha: Dpunkt, 2005. 331 p., ISBN 38-9864-220-8

WILLIAMS, Laurie et al. Strengthening the case for pair programming. **IEEE Software**, Washington: IEEE, v. 17, n. 4, p.19-25. July-Aug. 2000. ISSN 0740-7459