4. Estudo de Caso

Test-Driven Maintenance não poderia ser considerada uma técnica completa sem que conduzíssemos um estudo de caso que verificasse os conceitos propostos e comprovasse sua aplicabilidade em um cenário real.

Acreditamos que uma técnica de desenvolvimento apresenta resultados concretos somente quando ultrapassa a esfera teórica e consegue evidenciar sua utilidade no cotidiano de equipes de desenvolvimento do mercado.

O objetivo deste capítulo é documentar, de forma objetiva e imparcial, as lições aprendidas e os resultados obtidos na utilização de *Test-Driven Maintenance* na manutenção de um sistema real, deixando a avaliação crítica dos resultados apresentados para o próximo capítulo.

Estudos de caso produzem uma análise baseada na percepção de fenômenos observados em um contexto específico e, portanto, não permitem que as conclusões obtidas sejam diretamente generalizadas para outras circunstâncias, restringindo sua validade para o cenário onde foi conduzido.

O acompanhamento de estudos de caso em cenários reais também não permite o controle completo das variáveis estudadas, como em experimentos científicos, especialmente porque neste contexto estão envolvidas pessoas, projetos, prazos e clientes reais.

Realizar estudos de caso continuamente, embora não produza uma análise determinística sobre a utilização de uma técnica, pode evidenciar indicadores suficientemente fortes para justificar a sua adoção em contextos semelhantes.

Descrevemos, nas próximas seções, um estudo de caso que busca apresentar uma visão realista sobre a aplicabilidade de *Test-Driven Maintenance* em um cenário real e, assim, pretendemos incentivar a realização de muitos outros para consolidar os resultados apresentados.

4.1 Cenário

O estudo de caso foi conduzido em uma empresa privada do Rio de Janeiro, especializada em Gestão do Capital Humano, que conta, no momento de produção deste trabalho, com cerca de duzentos colaboradores.

A equipe de desenvolvimento onde o estudo foi realizado era composta por cinco integrantes graduados na área, incluindo um mestre em Engenharia de Software, e adotava um processo de desenvolvimento baseado no *Scrum* para a produção do *software* estudado.

A equipe adotava a prática de *code reviews*, utilizava o *Jira* como ferramenta de controle de tarefas, o *SVN* como sistema de controle de versionamento, o *Confluence* como ferramenta para compartilhamento de informações, o *Eclipse* como *IDE* e a linguagem de programação *Java*.

O sistema estudado, que atualmente possui mais de 500 mil linhas de código, foi concebido em 2003 e, desde então, já atendeu cerca de 200 mil usuários, em mais de 20 clientes. Buscando atender as demandas apresentadas pelos clientes ao longo do tempo, o sistema sofreu intensas evoluções que lhe renderam características clássicas de um sistema legado.

Os membros responsáveis por projetar e construir as primeiras versões do sistema já não participavam do seu desenvolvimento, as responsabilidades dos módulos e camadas se tornaram obscuras, a estrutura interna confusa e a capacidade de manutenção extremamente baixa.

Todos esses fatores contribuíram para que a grande quantidade de defeitos comprometesse a confiabilidade do sistema, a introdução de novos defeitos se tornasse comum e a produtividade da equipe durante a realização das constantes manutenções atingisse níveis insatisfatórios.

4.2 Processo de Introdução da Técnica

Diante do cenário apresentado, ficou evidente, para a equipe de desenvolvimento, a necessidade da realização de mudanças substanciais que alterassem o rumo que o sistema tinha tomado, caso contrário, a sua manutenção se tornaria insustentável em um curto prazo.

Os objetivos buscados eram o aumento da manutenibilidade do sistema e a redução da quantidade e da introdução de defeitos. As possíveis soluções eram restritas pela impossibilidade de paralisarmos o desenvolvimento de novas funcionalidades para um longo período de melhorias estruturais no legado.

A adoção de testes automatizados surgiu naturalmente como solução para as questões abordadas e, quando confrontamos a criação de testes depois do desenvolvimento com *Test-Driven Development*, as vantagens do desenvolvimento orientado a testes ficaram bastante claras para toda a equipe e a adoção da técnica foi consenso entre todos os membros.

O conhecimento da equipe em relação a testes unitários ou *Test-Driven Development* era apenas superficial, sem que houvesse nenhuma experiência prática. Após a realização de estudos, pesquisas e algumas reuniões internas, a equipe decidiu prosseguir com a idéia e buscar respaldo da organização para a adoção da técnica.

A equipe de desenvolvimento agendou uma reunião junto aos gerentes e diretores envolvidos para que as circunstâncias aqui descritas fossem apresentadas, as vantagens e desvantagens da solução fossem detalhadas e os riscos existentes evidenciados. Nesta reunião, todos os envolvidos, decidiram, em conjunto, adotar a solução apresentada, bem como arcar com os conseqüentes riscos.

O planejamento para implantação da técnica se baseava na criação de uma equipe responsável pela sua introdução, pela execução de um projeto piloto, pela realização de um treinamento da equipe de desenvolvimento e pela refatoração de um módulo crítico do sistema.

A criação de uma equipe responsável pelo processo de introdução da técnica foi determinante para que todas as etapas fossem realizadas com os cuidados necessários e para que o desenvolvimento de novas funcionalidades que agregassem valor para a organização pudesse continuar.

O objetivo da realização do projeto piloto era validar os conceitos apresentados, efetuar eventuais adaptações na técnica original, estabelecer os padrões necessários, realizar e documentar as modificações convenientes no ambiente de desenvolvimento e configurar as ferramentas facilitadoras para a adoção da técnica.

Os estudos realizados nesta etapa foram determinantes para embasar o processo de adaptação de *Test-Driven Development* que, iniciado na realização do projeto piloto, culminou na técnica apresentada no Capítulo 2.

A identificação de ferramentas adequadas de apoio ao desenvolvimento proporcionou a sustentação necessária para produtividade da equipe e viabilizou a adoção da técnica em um cenário real.

A ferramenta básica adotada foi um *framework*, amplamente utilizado no mercado, chamado *JUnit*, que permite a automatização da verificação dos resultados dos testes e a sua execução em conjuntos.

Adotamos, também, o *DBUnit*, uma ferramenta complementar ao *JUnit* que gerencia a integração dos testes automatizados com o sistema de gerenciamento do banco de dados, controla o estado da base de dados antes e depois da execução dos testes e administra eventuais repositórios externos de dados para os testes.

Mesmo combinando as duas ferramentas apresentadas, freqüentemente observávamos a realização de *overhead* para a localização dos repositórios de dados de cada unidade de testes, para a configuração de aspectos de infra-estrutura do sistema, para a importação de bibliotecas complementares e para o controle transacional durante a execução dos testes de integração com a base de dados.

Desenvolvemos, então, um *framework* de testes, cliente do *JUnit* e do *DBUnit*, que centralizava todo o *overhead* existente, estabelecendo uma interface customizada que se adequava com mais precisão às circunstâncias específicas do projeto e tornava a criação dos testes automatizados mais simples e produtiva.

A incorporação da cultura de testes, por parte da equipe de desenvolvimento, é fundamental para que a técnica seja bem sucedida. Adotar um servidor de integração contínua acelera o processo de incorporação desta cultura, explicitando, em tempo real, o impacto de cada manutenção realizada no sistema.

Buscando inserir essa cultura no cotidiano da equipe de desenvolvimento adotamos um servidor de integração contínua, largamente utilizado no mercado, chamado *Bamboo*, e o configuramos para executar a suíte de testes automatizados do sistema toda vez que uma modificação fosse integrada ao sistema de controle de versionamento.

A realização do projeto piloto, que durou cerca de um mês, confirmou a possibilidade de utilização da técnica, com as devidas adaptações, no contexto de legados e, então, a equipe responsável pela implementação da técnica foi designada para realizar *coaching* dos membros da equipe de desenvolvimento.

O objetivo do treinamento realizado era desenvolver as competências relacionadas com a técnica de modo a estabilizar o nível de conhecimento da equipe em um patamar que permitisse que ela pudesse dar continuidade à evolução da técnica de acordo com suas próprias iniciativas e percepções.

O *coaching* foi realizado em uma etapa inicial de reuniões para apresentação dos conceitos, ferramentas e padrões estabelecidos, seguida por uma etapa de *pair programming*, sempre com membros das duas equipes, que buscava exemplificar a aplicação da nova abordagem de desenvolvimento nas tarefas do dia a dia.

A realização da programação em pares apresentou aspectos extremamente positivos para a disseminação do conhecimento entre os membros da equipe, fomentando discussões extremamente ricas sobre os conceitos adotados, potencializando o engajamento da equipe e impulsionando a adaptação e evolução da técnica.

Buscando alcançar maiores benefícios e, assim, maximizar o retorno sobre o investimento realizado, identificamos o módulo que concentrava, segundo a percepção da equipe, a maior quantidade de defeitos, a maior propensão a introdução de erros e a menor manutenibilidade do sistema para que ele pudesse ser refatorado e, naturalmente, coberto pelos respectivos testes automatizados.

A refatoração realizada seguiu a técnica proposta no Capítulo 2 e os seus resultados, apresentados na próxima seção, foram avaliados segundo o modelo descrito no Capítulo 3.

4.3 Resultados Obtidos

O processo de refatoração do módulo foi realizado pela equipe responsável pela introdução da técnica e, conforme proposto no modelo apresentado no Capítulo 3, diversas informações foram coletadas, do ambiente de desenvolvimento e da equipe envolvida, com o objetivo de fundamentar a análise sobre trabalho realizado.

As informações coletadas podem ser divididas em três categorias: informações provenientes da mineração de dados do ambiente de desenvolvimento, informações coletadas através da realização de pesquisas de opinião e, por fim, informações oriundas das entrevistas individuais realizadas com os membros da equipe de desenvolvimento. Apresentamos cada uma dessas categorias nas seções a seguir.

4.3.1 Resultados obtidos através da mineração de dados

A mineração de dados do ambiente de desenvolvimento envolveu a extração, através da ferramenta apresentada na Seção “3.2 Ferramenta de Apoio a Análise”, das informações referentes aos indicadores de quantidade de defeitos existentes no módulo refatorado, de

criticidade destes defeitos e de quantidade de testes automatizados que exercitavam o módulo em questão.

Apresentamos, na tabela abaixo, os dados minerados referentes aos indicadores da quantidade de defeitos existentes e da quantidade de testes automatizados do módulo refatorado.

Mês	Quantidade de Defeitos	Quantidade de Testes
Fevereiro/2008	0	0
Março/2008	2	0
Abril/2008	0	0
Maio/2008	1	0
Junho/2008	0	0
Julho/2008	2	0
Agosto/2008	1	0
Setembro/2008	1	0
Outubro/2008	1	0
Novembro/2008	0	0
Dezembro/2008	2	0
Janeiro/2009	2	0
Fevereiro/2009	2	0
Março/2009	4	0
Abril/2009	3	0
Maio/2009	2	0
Junho/2009	3	0
Julho/2009	2	0
Agosto/2009	3	0
Setembro/2009	3	0
Outubro/2009	1	0
Novembro/2009	2	32
Dezembro/2009	3	30
Janeiro/2010	0	58
Fevereiro/2010	0	358
Março/2010	0	558
Abril/2010	0	569
Maio/2010	0	594
Junho/2010	1	614

Julho/2010	0	620
Agosto/2010	0	620
Setembro/2010	1	622
Outubro/2010	1	628
Novembro/2010	0	628
Dezembro/2010	0	644

Tabela 1: Histórico da quantidade de defeitos encontrados e da quantidade de testes existentes

Os dados apresentados na tabela serviram como base para a construção de uma série de relatórios, baseados nos modelos propostos na Seção "3.1 Processo de Coleta de Dados", com o objetivo retratar a variação dos indicadores coletados durante a adoção da técnica.

Os dois primeiros relatórios que construímos apresentam uma representação gráfica para a evolução, ao longo do tempo, dos indicadores coletados. Vale notar que, em ambos os relatórios, os dados foram agrupados em bimestres com o objetivo de dispersar eventuais distorções pontuais e apresentar uma visão normalizada dos resultados.

Apresentamos, então, a seguir, os relatórios de acompanhamento da quantidade de defeitos existentes e da quantidade de testes automatizados referentes ao módulo em questão.

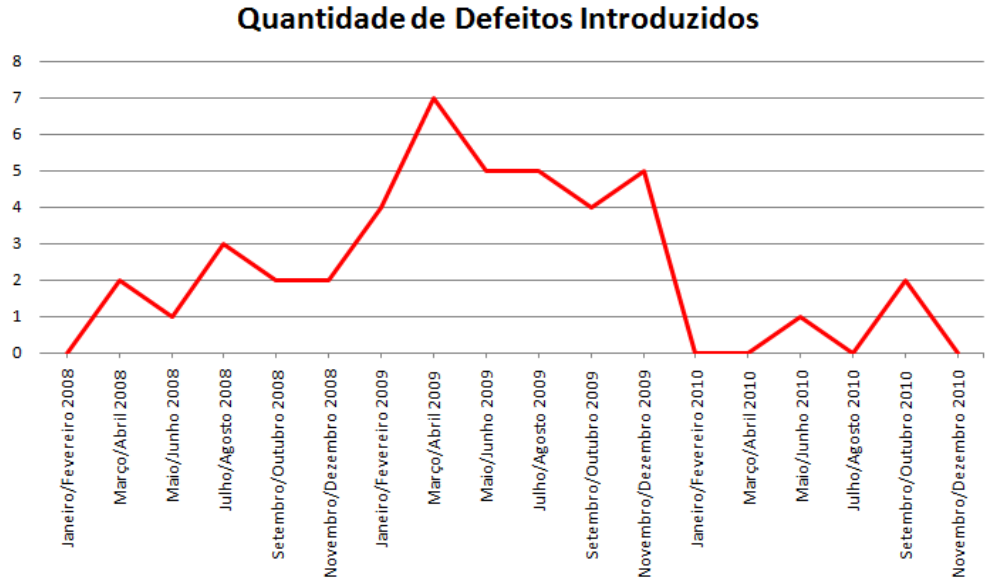

Figura 35: Gráfico de histórico de taxa de introdução de defeitos

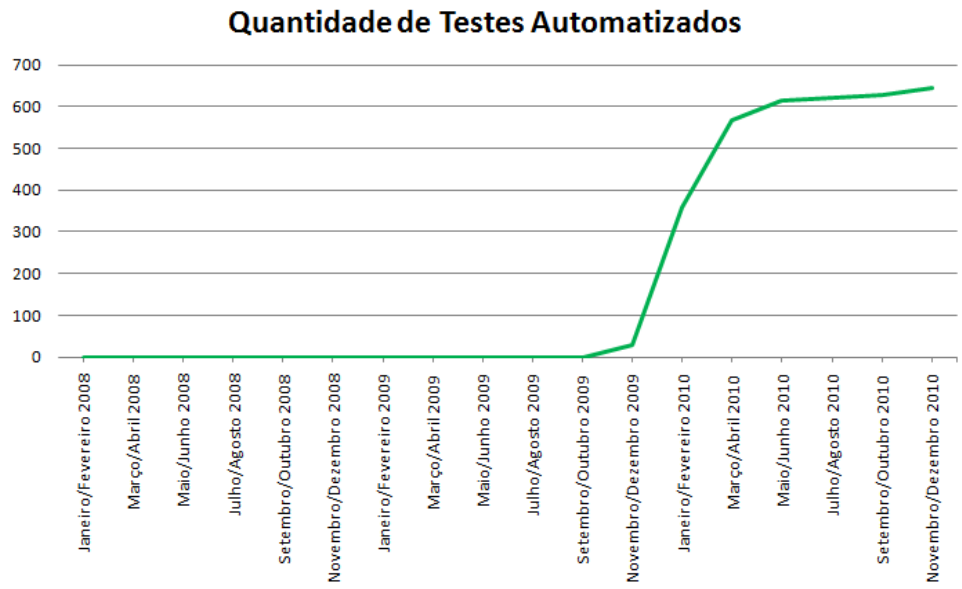

Figura 36: Gráfico de histórico da quantidade de testes automatizados

Acreditamos existir uma intensa relação entre os dois indicadores coletados, de modo que o aumento, ou redução, dos níveis observados em um deles provavelmente estará relacionado com a alteração dos níveis observados no outro.

Produzimos, então, um terceiro gráfico, apresentado abaixo, que consolida os dois primeiros e facilita a visualização deste relacionamento. Para isso, mantemos a escala utilizada no gráfico de "Quantidade de Defeitos" e passamos a apresentar a quantidade de testes em centenas.

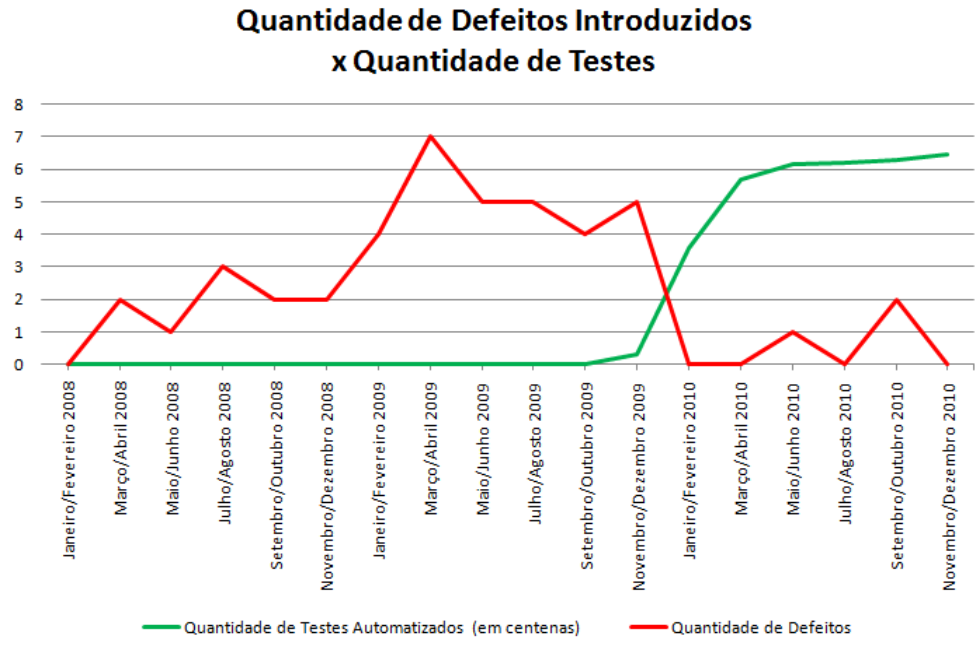

Figura 37: Gráfico de histórico de taxa de introdução de defeitos x histórico da quantidade de testes automatizados

Buscando retratar o impacto da utilização da técnica em prazos mais longos, produzimos mais dois relatórios que apresentam os valores medidos para os mesmos indicadores, mas, desta vez, agrupados em anos.

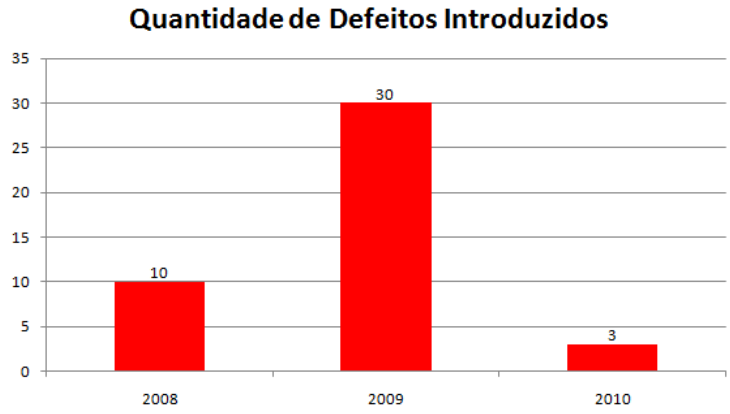

Figura 38: Gráfico de histórico de taxa anual de introdução de defeitos

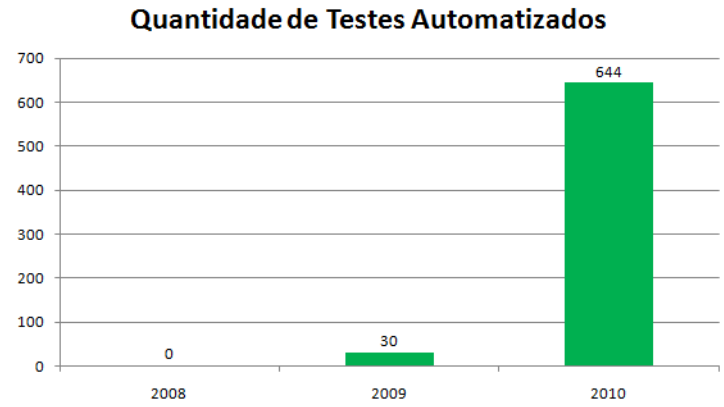

Figura 39: Gráfico de histórico anual da quantidade de testes automatizados

O modelo de avaliação proposto no Capítulo 3, além da quantidade de defeitos e da quantidade de testes, também propõe o acompanhamento da distribuição dos defeitos por nível de criticidade como indicador de qualidade.

Buscando documentar o histórico da distribuição dos defeitos no módulo refatorado, extraímos, também através da ferramenta desenvolvida e de acordo com os níveis de criticidade definidos no projeto, as informações relativas a este indicador.

O projeto estudado classificava os defeitos em cinco níveis de gravidade. Os critérios para classificação dos defeitos nos níveis de criticidade foram definidos pela organização e variavam de acordo com as conseqüências causadas e com os prazos estipulados para a sua resolução.

- Trivial: “Entendemos como triviais aquelas que podem aguardar o final do *Sprint* corrente para repriorização ou que demandem pequeno esforço.”
- Pequeno: “Entendemos como Pequenas pendências que devem ser resolvidas até o final do *Sprint* corrente. São exemplos: Comportamentos equivocados do sistema, desde que não classificados na prioridade anterior. Solicitações de clientes estratégicos que não impeçam o funcionamento do sistema, mas que significativamente representem uma melhoria no funcionamento, desde que dentro do contrato existente (caso contrário demandarão geração de OS); Configurações que não travem a continuidade do processo do cliente, mas que prejudiquem seu uso; Carga de dados que não travem a continuidade do processo do cliente, mas que prejudiquem seu uso; Pendências solicitadas por parceiros que estejam em processo corrente de venda.”
- Grande: “Entendemos como Grandes pendências que devem ser resolvidas em até 3 dias. São exemplos: Comportamentos equivocados do sistema, desde que não classificados nas duas prioridades anteriores; Solicitações de clientes estratégicos que não impeçam o funcionamento do sistema, mas que significativamente representem uma melhoria no funcionamento, desde que dentro do contrato existente (caso contrário demandarão geração de OS); Configurações que travem a continuidade do processo do cliente; Carga de dados que travem a continuidade do processo do cliente; Pendências solicitadas por parceiros que estejam em processo corrente de venda e que impeçam uma demonstração considerada crítica (que essencialmente deve ser validada pelo nosso Diretor Comercial).”
- Crítico: “Entendemos como Críticas aquelas pendências que podem aguardar até 24 horas para sua solução. São exemplos: Sistema fora do ar em períodos sem avaliação; Erros (mensagens de erro, não necessariamente comportamentos inesperados) que impedem o uso do sistema em períodos sem avaliação; Problemas que possam representar perdas de dados críticos, porém fora do período de avaliação; Pendências que impeçam que projetos entrem em produção nas datas combinadas; Atrasos superiores a 7 dias em qualquer entrega prometida para clientes com contratos ativos ou em períodos de implantação; Comportamentos equivocados do sistema (não confundir com desejos de comportamento do cliente); Atrasos superiores a 3 dias em clientes considerados

estratégicos; Solicitação expressa do *Product Owner*, desde que este classifique também a pendência como Crítica; Solicitação expressa de membros da Diretoria desde que validadas com o *Product Owner* frente ao conjunto total de demandas da equipe; Solicitação expressa de gerentes de projeto nomeados, desde que validadas com o *Product Owner* frente ao conjunto total de demandas da equipe."

- Severo: "Entendemos como Severas aquelas pendências que demandam parada total e solução imediata do problema. São exemplos: Sistema fora do ar durante processo de avaliação; Erros (mensagens de erro) que impedem o uso do sistema durante processo de avaliação; Problemas que possam representar perdas de dados críticos durante processos de avaliação; Comportamentos equivocados do sistema durante o período de avaliação; Atrasos superiores a 15 dias em qualquer entrega prometida para clientes com contratos ativos ou em períodos de implantação; Atrasos superiores a 5 dias em clientes considerados estratégicos; Solicitação expressa do *Product Owner*, desde que este classifique também a pendência como Severa; Solicitação expressa de membros da Diretoria desde que validadas com o *Product Owner* frente ao conjunto total de demandas da equipe."

Os dados extraídos foram organizados na tabela abaixo e utilizados como base para a construção de uma série de relatórios, descritos a mais adiante, que buscam, também, oferecer uma representação gráfica para a evolução da distribuição dos defeitos por nível de criticidade no módulo refatorado.

	Trivial	**Pequeno**	**Grande**	**Crítico**	**Severo**
Fevereiro/2008	0	0	0	0	0
Março/2008	0	1	1	0	0
Abril/2008	0	0	0	0	0
Maio/2008	0	0	1	0	0
Junho/2008	0	0	0	0	0
Julho/2008	0	0	0	1	1
Agosto/2008	0	0	1	0	0
Setembro/2008	0	0	1	0	0
Outubro/2008	0	0	1	0	0
Novembro/2008	0	0	0	0	0
Dezembro/2008	0	0	1	1	0

Janeiro/2009	1	0	1	0	0
Fevereiro/2009	0	1	1	0	0
Março/2009	0	2	2	0	0
Abril/2009	0	0	1	2	0
Maio/2009	0	0	2	0	0
Junho/2009	1	0	2	0	0
Julho/2009	0	0	2	0	0
Agosto/2009	1	1	1	0	0
Setembro/2009	0	0	2	1	0
Outubro/2009	0	0	0	1	0
Novembro/2009	0	0	1	1	0
Dezembro/2009	0	2	1	0	0
Janeiro/2010	0	0	0	0	0
Fevereiro/2010	0	0	0	0	0
Março/2010	0	0	0	0	0
Abril/2010	0	0	0	0	0
Maio/2010	0	0	0	0	0
Junho/2010	0	0	1	0	0
Julho/2010	0	0	0	0	0
Agosto/2010	0	0	0	0	0
Setembro/2010	0	0	1	0	0
Outubro/2010	0	0	1	0	0
Novembro/2010	0	0	0	0	0
Dezembro/2010	0	0	0	0	0

Tabela 2: Histórico da distribuição dos defeitos por nível de criticidade

Analogamente aos relatórios de quantidade de defeitos e quantidade de testes, construímos duas versões, apresentadas a seguir, para os relatórios de acompanhamento da distribuição de defeitos por criticidade para o módulo refatorado.

A primeira versão agrupa os dados por bimestre, oferecendo uma visão mais detalhista do histórico de evolução deste indicador. A segunda tem o intuito de descrever o impacto da utilização da técnica de maneira mais abrangente e, portanto, agrupa os dados em anos.

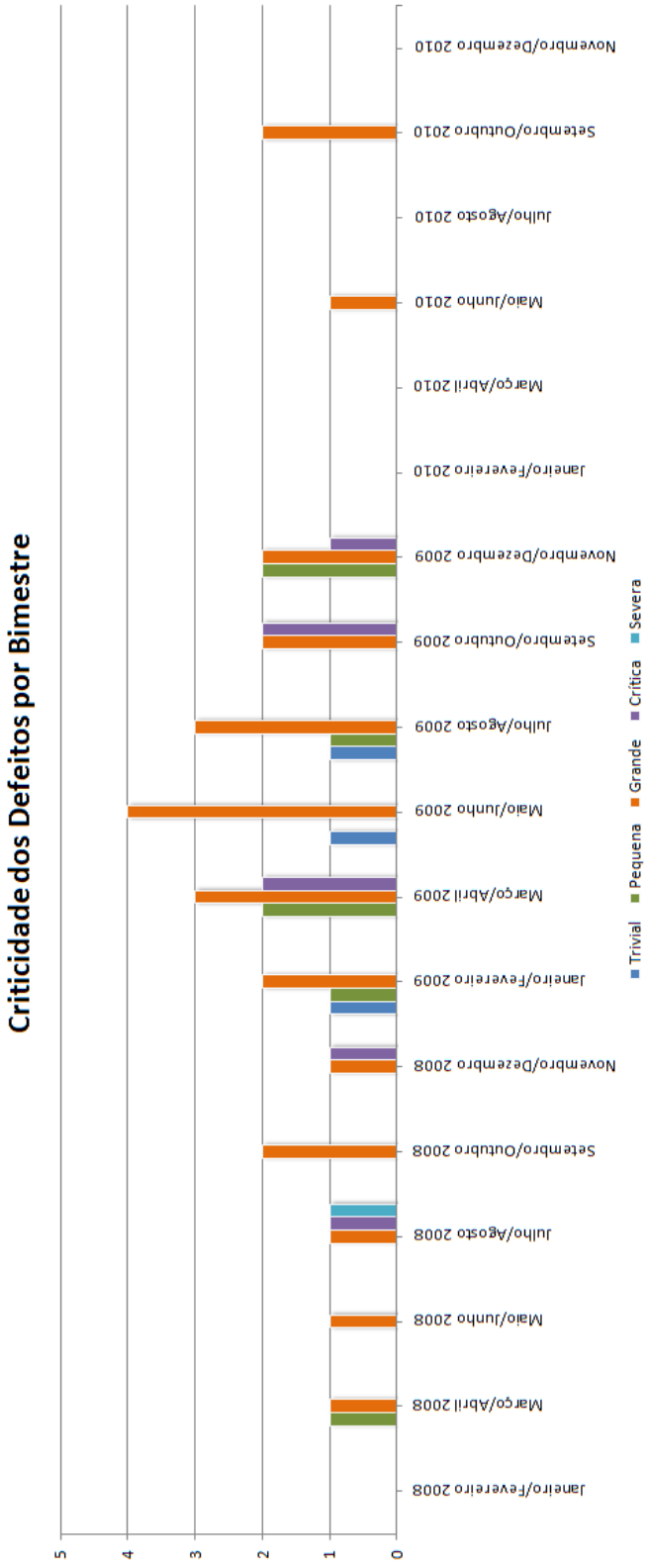

Figura 40: Gráfico de distribuição bimestral de defeitos por nível de criticidade

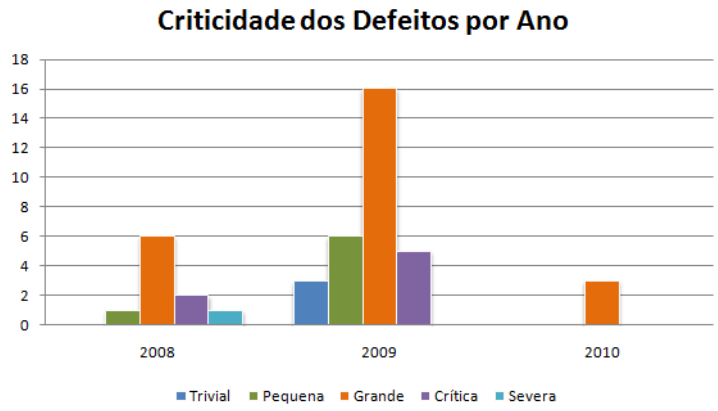

Figura 41: Gráfico de distribuição anual de defeitos por nível de criticidade

Embora os dados minerados proporcionem fundamentos estatísticos para análise do impacto da adoção de *Test-Driven Maintenance* no sistema estudado, a percepção da equipe de desenvolvimento sobre os resultados da sua utilização pode, eventualmente, divergir dos resultados apresentados.

Realizamos, então, uma pesquisa de opinião, detalhada na próxima seção, para identificar eventuais divergências entre os resultados aqui apresentados e a visão da equipe de desenvolvimento.

4.3.2 Resultados obtidos através de pesquisas de opinião

A pesquisa de opinião realizada na equipe de desenvolvimento também era aderente ao modelo de avaliação proposto no Capítulo 3 e, portanto, foi feita através da aplicação de questionários com questões de múltipla escolha.

A estrutura dos questionários foi definida com dois fatores principais que, por sua vez, eram compostos por quatro questões cada. Os fatores principais avaliavam a introdução da técnica na equipe e os resultados obtidos na refatoração do módulo crítico selecionado, respectivamente. Detalhamos, abaixo, a estrutura do questionário aplicado.

- **Fator 1**: Avaliação da introdução de *Test-Driven Maintenance* na equipe.
  - **Questão 1.1**: “De que maneira você avalia o impacto na produtividade da equipe, a curto prazo, da adoção de *Test-Driven Maintenance* na manutenção do sistema?”.
  - **Questão 1.2**: “De que maneira você avalia o impacto na produtividade da equipe, a longo prazo, da adoção de *Test-Driven Maintenance* na manutenção do sistema?”.

- **Questão 1.3**: “De que maneira você avalia o impacto na qualidade do sistema, a curto prazo, da adoção de *Test-Driven Maintenance* na manutenção do sistema?”.
- **Questão 1.4**: “De que maneira você avalia o impacto na qualidade do sistema, a longo prazo, da adoção de *Test-Driven Maintenance* na manutenção do sistema?”.

- **Fator 2**: Avaliação do resultado da refatoração do módulo crítico selecionado.
  - **Questão 2.1**: “De que maneira você avalia o impacto na produtividade da equipe, a curto prazo, da refatoração realizada no sistema?”.
  - **Questão 2.2**: “De que maneira você avalia o impacto na produtividade da equipe, a longo prazo, da refatoração realizada no sistema?”.
  - **Questão 2.3**: “De que maneira você avalia o impacto na qualidade do sistema, a curto prazo, da refatoração realizada?”.
  - **Questão 2.4**: “De que maneira você avalia o impacto na qualidade do sistema, a longo prazo, da refatoração realizada?”.

O questionário definia um conjunto único composto por cinco alternativas de resposta para todas as questões de múltipla escolha da pesquisa de opinião. As alternativas disponíveis estão listadas abaixo.

- Afeta Muito Negativamente
- Afeta Negativamente
- Não Afeta Significativamente
- Afeta Positivamente
- Afeta Muito Positivamente

Aplicamos, então, os questionários com a estrutura descrita anteriormente e consolidamos o resultado da pesquisa de opinião em uma série de histogramas, apresentados abaixo, detalhando a distribuição das respostas da equipe, questão a questão.

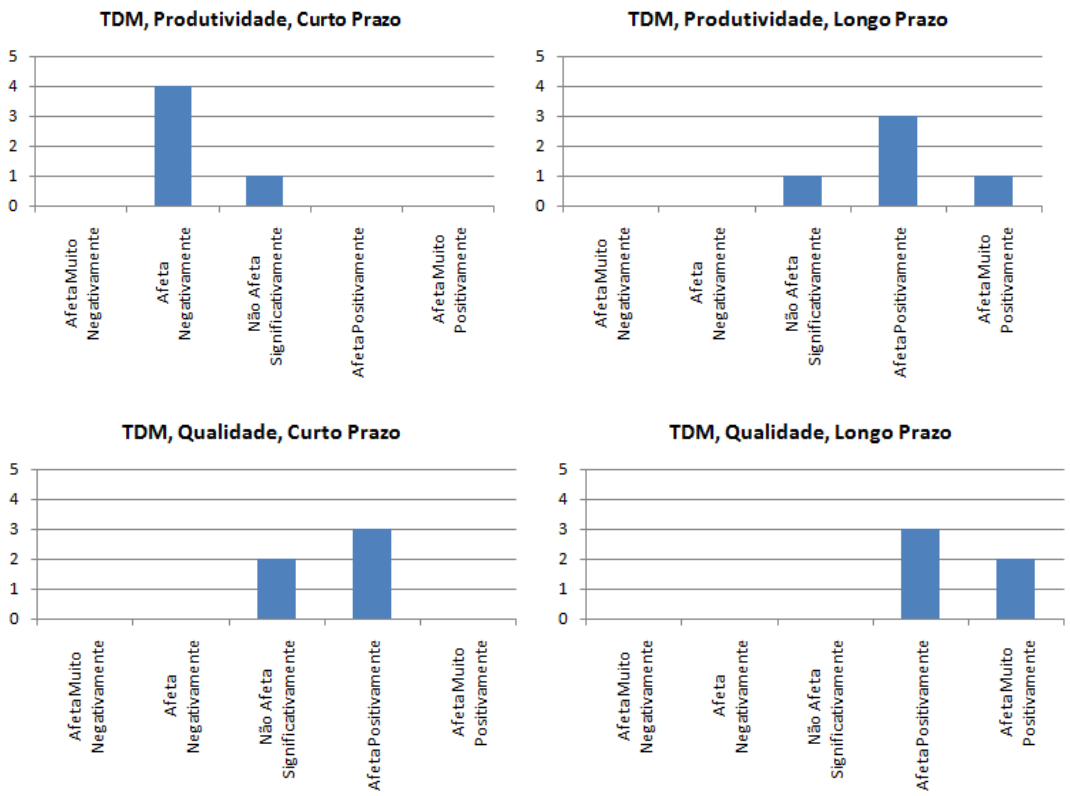

Figura 42: Histogramas por questão de distribuição de respostas sobre Test-Driven Maintenance

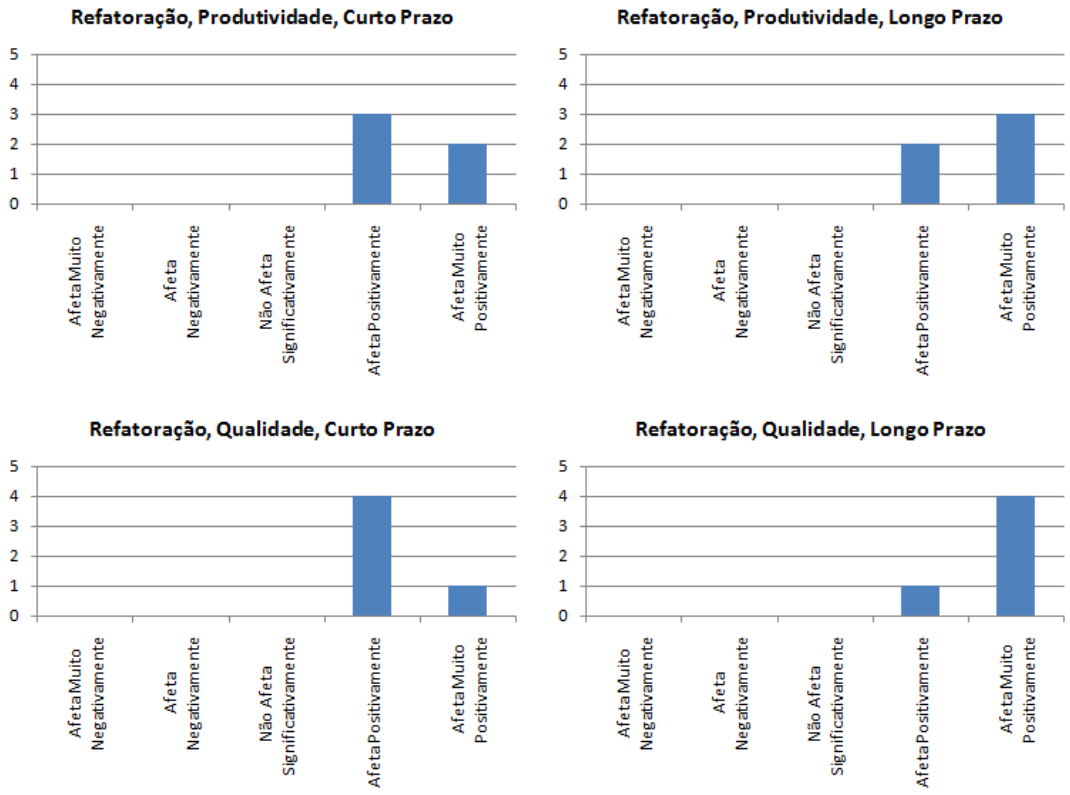

Figura 43: Histogramas por questão de distribuição de respostas sobre o módulo refatorado

Os resultados apresentados podem, ainda, ser consolidados para cada um dos fatores principais e, assim, proporcionar uma visão mais abrangente da opinião da equipe. A consolidação dos resultados por fator pode ser realizada através do acúmulo das respostas de suas respectivas questões, conforme apresentamos nos histogramas abaixo.

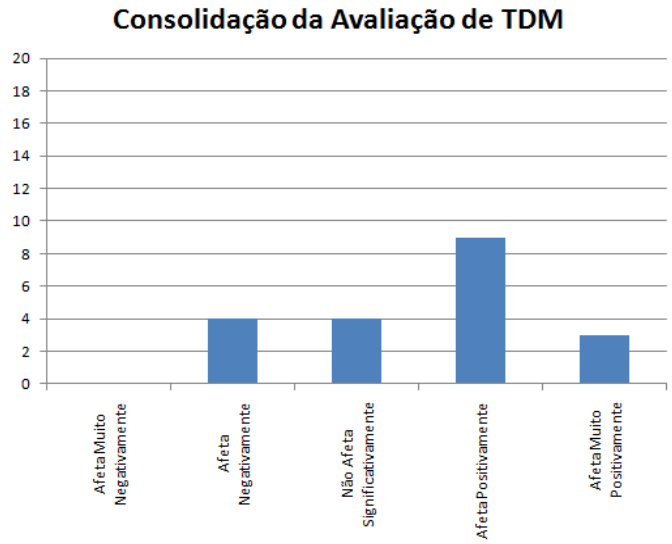

Figura 44: Histograma da distribuição das respostas sobre Test-Driven Maintenance

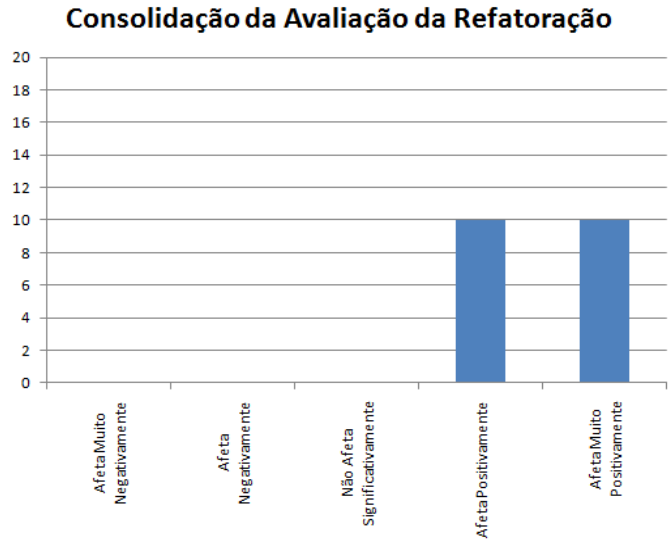

Figura 45: Histograma da distribuição das respostas sobre a refatoração do módulo crítico

Os histogramas que apresentam os resultados por fatores podem, por sua vez, ser combinados em um único histograma geral que consolida os resultados de toda a pesquisa de opinião realizada, como podemos visualizar a seguir.

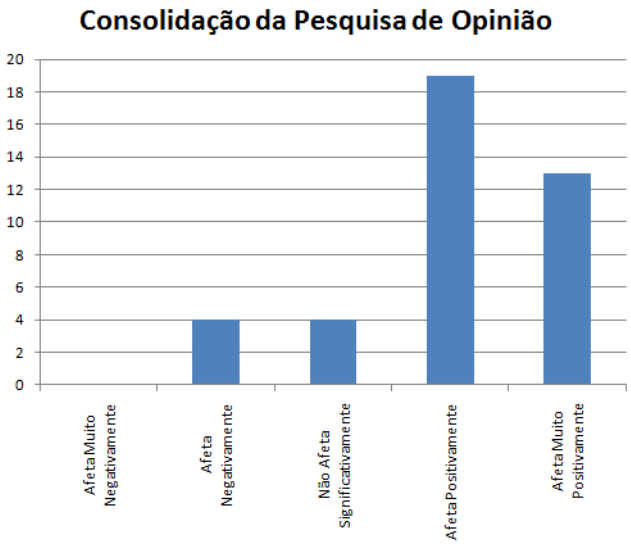

Figura 46: Histograma da distribuição das respostas de toda a pesquisa de opinião

A pesquisa de opinião descreve claramente a distribuição entre as opiniões dos membros da equipe, mas, por outro lado, ignora detalhes do ponto de vista de cada um deles.

Procurando entender mais detalhadamente a percepção da equipe sobre a introdução de *Test-Driven Maintenance*, realizamos, conforme proposto na Seção "3.1.3 Entrevistas Individuais", uma série de entrevistas com cada um dos integrantes da equipe e descrevemos, na próxima seção, os resultados obtidos.

4.3.3 Resultados obtidos através de entrevistas

As entrevistas realizadas revelaram questões extremamente interessantes, e de naturezas distintas, sobre o processo de introdução de *Test-Driven Maintenance* na equipe de desenvolvimento acompanhada.

O primeiro aspecto que destacamos está diretamente relacionado com a necessidade de incorporação da cultura de testes por parte dos desenvolvedores envolvidos. Parte dos entrevistados relatou observar certa "resistência a uma nova filosofia de trabalho" entre os membros da equipe e outra parte, confirmando indiretamente essa observação, argumentou considerar "o novo fluxo de trabalho antinatural".

A maioria dos entrevistados apontou a utilização de um servidor de integração contínua como um dos fatores mais importantes para a adoção da técnica, demonstrando, até, uma excessiva dependência em relação à ferramenta.

Os principais argumentos apresentados foram de que a sua utilização facilitava a incorporação da cultura de testes no cotidiano da equipe, forçava a execução freqüente da suíte e tornava simples a identificação da modificação responsável por “quebrá-la”.

Alguns entrevistados, inclusive, justificaram os períodos em que a suíte de testes não estava sendo aprovada argumentando que “o *Bamboo* estava fora do ar” e que, desta forma, não era possível ter a garantia de que as modificações realizadas por eles eram responsáveis por “quebrar” os testes.

Os entrevistados afirmaram acreditar que a construção dos testes antes da implementação evita algumas conseqüências indesejadas que ocorrem no fluxo natural. A construção dos testes após a implementação, segundo os colaboradores, “torna a seleção de casos de testes polarizada”, faz com o que os módulos acumulem muitas responsabilidades e os testes sejam extremamente grandes, tornando o processo de criação e manutenção de testes mais lento e menos produtivo.

O argumento utilizado para justificar a “seleção polarizada dos casos de testes” foi que, ao selecionar os casos após a implementação, os casos de testes tendem a testar somente o que está implementado e não o que deveria estar, reduzindo, assim, a quantidade de defeitos encontrados.

O acúmulo de responsabilidades, ainda segundo os entrevistados, era responsável pela construção de testes maiores, já que “o número de variáveis controladas era maior”, e pela criação de uma quantidade maior de testes, já que “a quantidade de casos de testes cresce exponencialmente em relação ao crescimento do número de variáveis controladas”.

Mesmo diante das conseqüências indesejadas observadas na construção de testes após a implementação, alguns membros da equipe afirmaram realizar este fluxo de trabalho “quando julgavam as tarefas muito simples” e os problemas descritos eram, eventualmente, encontrados na manutenção dos testes.

Os entrevistados responsáveis pela realização de *code reviews*, prática já adotada pela equipe antes da introdução da técnica, relataram que, durante a etapa de revisão, os casos descritos eram facilmente identificados e, normalmente, as extensas correções eram realizadas. Infelizmente, em alguns casos, o desenvolvedor que tinha o código revisado oferecia algum tipo de resistência, afinal o código já estava “pronto e funcionando”, e as modificações ficavam para um “momento posterior”.

O problema descrito foi fonte para uma série de discussões que buscavam tornar as revisões de código mais rigorosas e que resultaram na prática de planejamento em pares, proposta na Seção “2.3.1 Micro planejamento em pares e revisão de código”. Os entrevistados relataram que, após a adoção dessa prática, a ocorrência desses problemas foi expressivamente minimizada.

A maior parte do sistema, especialmente por se tratar de um legado, não estava coberta por testes, mesmo depois de mais de um ano de utilização da técnica, e o impacto dos testes era tão grande que alguns entrevistados afirmaram que “a única parte confiável do é a coberta por testes”.

A equipe de desenvolvimento observou, também, que, embora aplicasse a técnica proposta, a realização de testes de regressão na interface da aplicação continuava necessária, o que é natural, já que ambas as categorias de testes são complementares.

Os entrevistados destacaram que o apoio da organização, especialmente quando a equipe está pressionada por prazos, é extremamente importante para que a técnica seja aplicada corretamente e, de fato, evite que mais problemas sejam criados nos momentos críticos.

As entrevistas revelaram que muitos dos colaboradores acreditam que transparecer periodicamente os resultados obtidos para a organização seria uma prática que conquistaria um apoio maior, embora não tenha sido realizada ou sugerida durante o projeto.

Algumas das vantagens da utilização da técnica ainda não estavam claras para todos os entrevistados. Parte deles, inclusive, destacou a importância de que todos compreendessem os conceitos envolvidos para que pudessem aplicá-los com mais acurácia e entender os objetivos buscados, especialmente a longo prazo.

Embora nem todas as vantagens estivessem claras para os entrevistados, todos acreditam que a aplicação da técnica nos leva a “um design ótimo”, nos “força a realizar uma etapa de análise” e serve como “uma documentação fortemente estruturada” que simplifica a compreensão das unidades testadas.

Uma informação que retrata bem a receptividade da equipe em relação à técnica é que todos os entrevistados, quando perguntados se adotariam *Test-Driven Maintenance* em futuros projetos, responderam positivamente.

Os pontos de vista apresentados nesta seção foram extremamente relevantes para as adaptações realizadas na técnica e muitas das idéias identificadas durante este processo foram convertidas em lições aprendidas, que descreveremos na próxima seção.

4.4 Lições Aprendidas

A realização do estudo de caso apresentado neste capítulo produziu, especialmente por retratar a primeira aplicação de *Test-Driven Maintenance* em um cenário real, um conjunto bastante rico de lições aprendidas.

A documentação das lições aprendidas é um processo extremamente importante para o compartilhamento do conhecimento adquirido e para a potencialização dos resultados alcançados em futuras implantações da técnica, conforme argumentamos na Seção “3.3 Documentação das Lições Aprendidas”.

As lições aprendidas foram agrupadas em duas categorias que reúnem as experiências adquiridas em diferentes etapas do estudo de caso.

4.4.1 Projeto Piloto

A realização do projeto piloto foi essencial para a identificação, e efetuação, de muitas das adaptações realizadas na técnica original. Alguns dos conceitos propostos no Capítulo 2 são, inclusive, produto das experiências adquiridas durante o piloto, como descreveremos nesta seção.

A primeira questão com que nos deparamos foi a dificuldade de estabelecer um único algoritmo que descrevesse o modo adequado de efetuar os diversos tipos de manutenção no sistema legado.

As experiências adquiridas durante as sucessivas manutenções efetuadas no projeto piloto contribuíram para a identificação das diferentes estratégias propostas no ciclo de desenvolvimento de *Test-Driven Maintenance*, conforme descrevemos na Seção “2.2 Ciclo de Desenvolvimento”.

A construção de testes automatizados que, de fato, garantam que o comportamento desejado seja realizado, é fundamental para que a técnica seja efetiva. A seleção *ad hoc* de casos de testes e a ausência de uma estrutura bem definida para as assertivas de entrada e saída dos testes contribuíam para que os testes fossem eventualmente falhos.

Estabelecemos, então, uma etapa de planejamento de testes, conforme argumentamos na Seção “2.2 Ciclo de Desenvolvimento”, baseada nos conceitos apresentados por [Hunt, 2003], que buscava definir critérios sistêmicos para a sua seleção, bem como determinar as assertivas necessárias.

Complementando a etapa de planejamento, definimos uma estrutura que estabelecia o padrão de implementação dos testes. O padrão definido auxiliava o mapeamento dos casos de testes selecionados na sua implementação, simplificava sua compreensão e, conseqüentemente, facilitava sua construção e manutenção.

A estrutura dos testes estabelecia a verificação de pré-condições, apresentadas na Seção “2.3.3 Verificação de pré-condições”, e a criação de repositórios de dados externos às unidades de testes, apresentadas na Seção “2.3.2 Repositório de dados externo à unidade de testes”.

Durante o planejamento dos testes, encontramos freqüentes dificuldades para tratar a realização do redesenho de interfaces e, à medida que as alternativas para o problema surgiam, eram avaliadas e os aspectos que as influenciavam eram absorvidos e utilizados para a identificação de novas soluções.

Inicialmente adotamos a estratégia de criar os testes para a interface sendo redesenhada, explicitando a necessidade de garantir a manutenção do seu comportamento, para depois tentar migrá-los para a nova modelagem. Constatamos o baixo valor agregado por essa estratégia e o alto custo para a sua realização.

Diante da alternativa descartada, adotamos a estratégia de minimizar, o tanto quanto possível, a realização de redesenho, sempre mantendo as interfaces e responsabilidades intactas, limitando-se a refatorações internas aos módulos. Percebemos a dificuldade de criar testes para essas interfaces e a limitação que estávamos impondo contra a evolução do design do sistema.

Chegamos, então, a solução adotada, proposta na Seção “2.2.3 Refatorações”, e diante da explícita necessidade de, para casos críticos, maior segurança para a realização das manutenções, identificamos na criação de testes mais abrangentes, conforme descrevemos na Seção “2.3.4 Testes abrangentes e especialização de testes”, uma estratégia complementar adequada para essa questão.

Observamos, também, que as unidades de testes freqüentemente possuíam uma grande quantidade de código de configuração da sua infra-estrutura, como localização de

repositórios de dados, controle transacional e importação de bibliotecas comuns. A configuração da infra-estrutura demandava um esforço repetitivo e propenso a erros, aumentando o trabalho realizado na construção dos testes.

Sabendo que o esforço necessário para criação e manutenção dos testes é determinante para que o retorno sobre o investimento realizado na adoção da técnica seja maximizado, decidimos construir um *framework*, conforme discutimos na Seção "4.2 Processo de Introdução da Técnica", que realizava todo tipo de configuração necessária.

O *framework* construído tornou os testes mais "limpos" e reduziu o esforço necessário para adoção da técnica, tornando a configuração da infra-estrutura declarativa e deixando as unidades de testes com a responsabilidade exclusiva de, naturalmente, testar os módulos do sistema.

As experiências adquiridas durante o projeto piloto foram fundamentais para a consolidação da técnica e contribuíram fortemente para o processo de introdução na equipe de desenvolvimento. Apresentamos, na próxima seção, as lições aprendidas durante a adoção da técnica, após a realização do projeto piloto, na equipe de desenvolvimento.

4.4.2 Aplicação da Técnica

A aplicação da técnica foi bastante facilitada pelas experiências adquiridas durante a realização do projeto piloto, mas, como era de se esperar, ainda assim, identificamos questões complementares extremamente importantes para a adoção de *Test-Driven Maintenance* em uma equipe de desenvolvimento real.

Observamos que gerenciar o processo de introdução da técnica no cotidiano da equipe de desenvolvimento de maneira adequada é determinante para que os resultados desejados sejam alcançados.

A introdução de *Test-Driven Maintenance* é especialmente crítica por modificar o fluxo de trabalho natural adotado pelos desenvolvedores. As vantagens apresentadas são atingidas somente se a técnica for aplicada com rigidez e a cultura da criação de testes automatizados for incorporada pela equipe.

Acreditamos que a adoção de ferramentas apropriadas, embora não seja determinante para a aplicação bem sucedida da técnica, pode acelerar o processo de incorporação da

cultura de testes, desde que os desenvolvedores entendam o papel das ferramentas e os conceitos existentes por trás delas.

Oferecer o treinamento adequado para a equipe de desenvolvimento é essencial para que os conceitos que fundamentam a técnica possam ser plenamente compreendidos e traduzidos corretamente na prática.

Entender a teoria que sustenta a técnica é fundamental para que a equipe esteja convencida das vantagens de sua utilização, condição básica para que a técnica seja efetivamente aplicada.

Transcender a teoria e aplicar os conceitos em tarefas do cotidiano pode, ainda, ser um desafio para os desenvolvedores. A prática de programação em pares se mostrou extremamente eficaz nestas circunstâncias, como argumentamos na Seção “4.2 Processo de Introdução da Técnica”, e é fortemente recomendada.

Realizar fóruns de discussão, conforme proposto na Seção “3.3 Documentação das Lições Aprendidas”, é uma excelente alternativa para compartilhar os problemas encontrados no dia a dia, bem como as respectivas soluções. Os fóruns de discussão criam um ambiente mais colaborativo para os desenvolvedores, aumentando o seu engajamento no processo de introdução da técnica e os motivando a identificar a pontos passíveis de melhoria.

As dificuldades encontradas, durante o projeto piloto, para a criação de testes que, de fato, garantissem que o comportamento desejado fosse realizado, foram reincidentes neste novo contexto.

A formalização de uma etapa de planejamento de testes amenizou o problema, mas, freqüentemente, as revisões de código ainda demandavam retrabalho para a correção de eventuais falhas. Ora testes não verificavam as condições necessárias, ora faltavam, ora sobravam, ora o design adotado não era adequado.

Percebemos, então, que somente a realização de um processo essencialmente corretivo de revisão não se adequava ao contexto da equipe. Decidimos adotar uma abordagem preventiva, apresentada na Seção “2.3.1 Micro planejamento em pares e revisão de Código”, que apoiava, também, o planejamento das tarefas.

Os resultados desta prática foram melhores do que o esperado e, além de observarmos uma redução na realização de retrabalho, percebemos que a sua adoção atuava na

disseminação de *Test-Driven Maintenance*, equilibrava o conhecimento entre os membros do time e contribuía para a difusão da cultura de testes dentro da equipe.

O respaldo da organização para aplicação da técnica é fundamental para que, em momentos de intensas pressões externas, a equipe de desenvolvimento possa aplicá-la da maneira adequada, evitando que circunstâncias críticas tornem-se mais complicadas ainda e obtendo, assim, os resultados desejados.

Acreditamos que a estratégia mais apropriada para conquistar o apoio da organização é apresentar, periodicamente e de maneira proativa, resultados concretos, incluindo indicadores de qualidade e produtividade, que descrevam o impacto da adoção da técnica e facilitem a análise do retorno sobre o investimento realizado.

As experiências adquiridas durante todo o processo de adaptação e introdução da técnica foram fundamentais para que os resultados obtidos fossem atingidos. Determinar se os resultados apresentados justificam a aplicação de *Test-Driven Maintenance* é crucial para que a técnica seja novamente aplicada. Sendo assim, apresentamos, no próximo capítulo, uma análise dos resultados descritos até aqui.