3rd International Conference on Information Systems and Technology Management
3º Congresso Internacional de Gestão da Tecnologia e Sistemas de Informação
11th World Continuous Auditing Conference
De 31 de Maio a 02 de Junho de 2006 – São Paulo/SP – Brasil

MELHORIA CONTÍNUA DIRECIONADA POR INDICADORES DE QUALIDADE E COM FOCO NA ARQUITETURA DO SISTEMA

Dimas Ribeiro de Magalhães
IPT – Instituto de Pesquisas Tecnológicas do Estado de São Paulo
Mestrado Profissional em Engenharia de Computação – São Paulo - Brasil
dimas_magalhaes@yahoo.com.br

ABSTRACT

An application software system has its level of quality decreasing during the course of its economic life. This fact occurs due to the constant changes determined by corrective maintenance, the necessity to fit to the changes in the business process, to meet the law requirements and technological update. These changes, after some time, make the system deviate from its original architecture and quality requirements from the time they were built. As the level of quality decreases, the services provided are no longer as efficient as they used to be, not to mention the increase in costs and time for maintenance it causes.

This paper presents a process of continual improvement of runtime software systems that proposes to change this common situation, in which the intensive maintenance decreases the quality of software systems. The process was defined from an incremental evolution methodology that is based on software quality standards that supply the mechanisms of sustentation for the improvements and it makes use of the software architecture documentation as a way to provide a clear communication among stakeholders. An example of application is presented to illustrate the results achieved from the practical application of the process.

Key words: software product improvement, software architecture, software quality, software quality standards, software maintenance.

RESUMO

Um sistema de software de aplicação tem seu nível de qualidade decrescente durante a sua vida útil, fato este que decorre das constantes alterações determinadas por manutenções corretivas, necessidade de adequação às mudanças nos processos de negócio, atendimento à legislação e por atualizações tecnológicas. Estas mudanças, com o tempo, levam o sistema a desviar-se de sua arquitetura e de seus requisitos de qualidade originais de quando foi construído. Com o nível decrescente de qualidade, os serviços prestados passam a não ter a mesma eficiência, além de causar aumento nos custos e nos prazos de manutenção.

Este artigo apresenta um processo de melhoria contínua de sistemas de software em produção que se propõe a reverter esta situação comum, de que manutenções constantes obrigatoriamente diminuem a qualidade dos sistemas. O processo foi definido a partir de uma metodologia de evolução incremental de sistemas que baseia-se em normas de qualidade, que fornecem os mecanismos de sustentação para as melhorias, e utiliza-se da documentação da arquitetura de software como linguagem para a comunicação entre os diversos stakeholders. Um exemplo de aplicação é apresentado para ilustrar os resultados obtidos com a utilização prática do processo.

Palavras-chave: melhoria do produto de software; arquitetura de software; qualidade de software; normas de qualidade de software; manutenção de software.

1 Introdução

Os sistemas de apoio a operações de negócios, chamados de sistemas aplicativos, tem um ciclo de vida estritamente ligado às operações do negócio que apóiam. Enquanto durarem estas operações, o sistema que as apóia será necessário. Esta forte vinculação faz com que todas as demandas relacionadas a requisições de mercado e a mudanças em legislação devam rapidamente ser atendidas pelo sistema. Além disso, as atualizações tecnológicas, de hardware e de software, também precisam ser aplicadas sobre o sistema para que os produtos da empresa não fiquem em condições desfavoráveis perante a concorrência. Também a manutenção corretiva demanda parte do tempo e dos recursos da equipe de manutenção. Neste contexto, o processo de manutenção sofre constante pressão para trabalhar com redução de recursos e de tempo para apresentar as melhores soluções. Desta forma, tentando atender a todas as demandas de manutenção necessárias e respeitando as restrições de recursos, é comum que a equipe de manutenção, com a aprovação dos usuários, dê soluções superficiais que resolvem o problema e as suas conseqüências imediatas, sem se aprofundar na análise de implicações em outros aspectos do sistema.

As soluções que potencialmente podem minimizar os problemas inerentes à manutenção estão relacionadas a diversos fatores: mudança cultural, para que se possa parar de privilegiar soluções parciais e imediatas que podem não ser as melhores; conhecimento da arquitetura do sistema atual; envolvimento dos diversos *stakeholders* no desenho das soluções e entendimento de que a arquitetura existente é uma restrição a ser respeitada com a vantagem de que funciona efetivamente e tem muito conhecimento acumulado. Estes são fatores que devem ser levados em consideração em qualquer processo de manutenção que tenha como objetivo prolongar a vida útil dos sistemas e otimizar os investimentos já feitos pelas organizações.

O presente artigo apresenta uma forma de se fazer manutenção de sistemas incluindo-se, como parte do processo, procedimentos que ajudem a melhorar o entendimento do problema, através da visão da estrutura do software e da solução proposta, além de criar mecanismos que possibilitem a quantificação da solução e a medição da evolução do sistema. Aplicando-se estes procedimentos de maneira cíclica e contínua pode-se obter a melhoria contínua do sistema prolongando-se assim a sua vida útil.

O processo de melhoria contínua (PMC), foi desenvolvido a partir da observação e do estudo da manutenção de sistemas de aplicação comercial. Sua aplicação foi testada neste contexto, mas não há impedimento para a aplicação em outros tipos de sistemas de processamento de dados. O nível de abstração em que ele está sendo apresentado permite que outros sistemas de software possam utilizá-lo. A proposta de processo refere-se a um modelo base que deve ser adaptado para utilização em outros sistemas. Os documentos arquiteturais e as métricas a serem utilizados dependem de cada caso em estudo e de cada sistema.

No desenvolvimento deste texto descreve-se o contexto conceitual do processo de melhoria, onde apresenta-se os principais conceitos sobre manutenção de software, qualidade de software, normas relacionadas, arquitetura de software e sua documentação. Na seqüência se apresenta o processo de melhoria contínua com suas atividades e seus artefatos. Um exemplo de aplicação é demonstrado para a melhoria de um sistema de seguros de uma empresa real. Finalmente são apresentados os resultados e as conclusões.

2 Trabalhos relacionados

Alguns autores têm tratado do assunto evolução de sistemas. Khan e Zheng (2005) citam que, dos vários modos de se gerenciar sistemas legados, entre não se fazer nada, reescrever ou substituir o sistema por outro, os autores recomendam a evolução incremental, pois ela foca em problemas que são mais visíveis aos usuários, resolvendo-os um de cada vez. Citam ainda que esta é a única maneira de muitas organizações manterem seus sistemas competitivos e que, a falta de consideração dos problemas apontados pelos usuários no processo de evolução incremental pode levar a insatisfação destes quando não vêem na evolução os benefícios que esperam.

Seacord, Plakosh e Lewis (2003), citam que a evolução do software pode ocorrer por manutenção, modernização e substituição, conforme figura 1. A manutenção ocorre enquanto as alterações conseguem acompanhar as necessidades dos usuários. Quando há um distanciamento entre o que se espera do sistema e o que ele efetivamente faz, é necessário que ocorra uma modernização. Esta é, normalmente, uma manutenção de grande monta e, às vezes, estrutural. A substituição é inevitável quando o sistema não mais atende plenamente às funcionalidades requeridas e os custos de manutenção ou modernização ficam proibitivos.

Neste trabalho tem-se o intuito de prolongar a vida útil do sistema fazendo com que ele esteja constantemente alinhado com as funcionalidades requeridas pelos usuários.

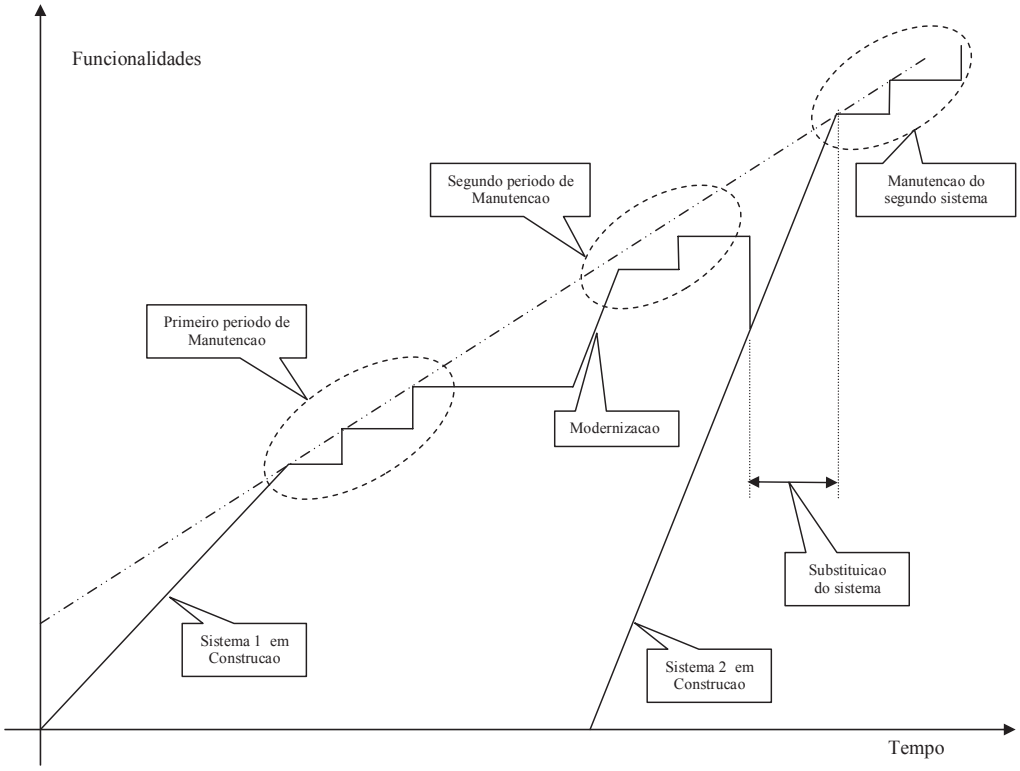

Figura 1 – Ciclo de vida de um sistema de informações

3 Contexto conceitual do processo de melhoria

Para entendimento do processo de melhoria apresentado, alguns tópicos devem ser conhecidos. Manutenção de sistemas e as leis de Lehman (1997), arquitetura de software e geração de sua documentação e qualidade de software com suas normas e critérios de avaliação foram utilizados na concepção do processo de melhoria.

3.1 Manutenção de software

Manutenção de sistemas de software conforme definida por Pigoski (1996), é a totalidade de atividades requeridas para prover a custo efetivo o suporte a um sistema de software. A manutenção dos sistemas existentes pode responder por 60% de todo o esforço da área de desenvolvimento de sistemas (Pressman, 2001). Pode-se atestar que algumas questões comuns na manutenção de sistemas são: urgência para solução do problema, necessidade de priorização e limitação de recursos. A realização da manutenção sob estas condições faz com que, no decorrer do tempo, os sistemas de aplicação fiquem desestruturados, com processos complexos, módulos executando sem necessidade, bases de dados desbalanceadas, problemas de performance, desconhecidos pelos desenvolvedores que têm a manutenção sob sua responsabilidade.

As leis da evolução dos sistemas em manutenção de Lehman (citado em Pfleeger, 2004), já citavam estas características da manutenção e determinavam que, somente com uma ação efetiva e contínua de melhoria é que se poderia manter os sistemas de aplicação úteis e com níveis satisfatórios de qualidade. Dentre elas, as três primeiras leis, são citadas abaixo:

- Mudança contínua: os sistemas nunca estão completos e devem passar por continuas mudanças para continuarem úteis;
- Aumento da complexidade: com a mudança os sistemas tornam-se mais complexos a menos que se faça um trabalho para manter-se ou reduzir-se a complexidade;
- Lei fundamental da evolução do programa: os sistemas de software apresentam comportamentos e tendências regulares que podem ser medidas e previstas;

3.2 Qualidade de software

A norma NBR ISO/IEC 9126 – Qualidade de Produto de software foi desenvolvida num esforço para se identificar atributos-chave de qualidade de software (Pressman 2001). A relevância no contexto do PMC é a orientação quanto a definição das características ou subcaracterísticas a avaliar.

Quadro 1 - Características e subcaracterísticas da NBR ISO/IEC 9126

NBR ISO/IEC 9126 – Características e subcaracterísticas	
Característica	**Subcaracterísticas**
Funcionalidade	Adequação – Acurácia – Interoperabilidade – Segurança de acesso
Confiabilidade	Maturidade – Tolerância a falhas – Recuperabilidade
Usabilidade	Inteligibilidade – Apreensibilidade – Atratividade – Operabilidade
Eficiência	Comportamento em relação ao tempo – Comportamento em relação aos recursos
Portabilidade	Adaptabilidade – Capacidade para ser instalado – Coexistência – Capacidade para substituir
Manutenibilidade	Analisabilidade – Modificabilidade – Estabilidade – Testabilidade

A norma NBR ISO/IEC 9126 (quadro 1) define seis características. Estas têm um mínimo de sobreposição, e se subdividem, de forma hierárquica, em subcaracterísticas que podem ser utilizadas para se avaliar a qualidade do produto de software. Fica a cargo dos usuários da norma estabelecerem as métricas de avaliação de cada característica ou subcaracterística.

3.3 Arquitetura de software

Uma conseqüência natural da evolução dos negócios e do ambiente operacional é que os sistemas de software fiquem cada vez maiores e mais complexos. Neste contexto, desenvolver e manter sistemas fica cada dia mais difícil. Para ajudar a resolver esta situação surge a arquitetura de software como um novo nível de abstração que trata as questões relacionadas a entendimento, controle, manutenção, evolução e reuso de componentes ou partes do software. Antonio Mendes (2002) cita que à medida que o tamanho e a complexidade dos sistemas de software aumentam, o problema do projeto extrapola as estruturas de dados e os algoritmos da computação demandando assim uma nova forma de abstração que é a arquitetura de software. Para Bass, Clements e Kazman (2003) a arquitetura de software de um programa ou de um sistema de computador é a estrutura ou estruturas do sistema, as quais compõem-se de elementos de software, as propriedades visíveis destes elementos e o relacionamento entre eles. Já, segundo Shaw e Garlan (1999), arquitetura é o nível de projeto que envolve a descrição de elementos dos quais o sistema é construído, a interação entre estes elementos, os padrões que guiam sua composição e as restrições e regras destes padrões.

O conhecimento da arquitetura de um sistema é importante não somente para facilitar a implementação e os testes do sistema, mas também para proporcionar rapidez e eficiência com que se poderão implementar alterações e manutenções (Pfleeger 2004).

A arquitetura pode orientar a manutenção de sistemas nos casos de:

- Servir como base de conhecimento, pois sistemas grandes têm muitos envolvidos que o enxergam cada um a sua maneira e os modelos de representação arquitetural geram as visões necessárias para a correta negociação entre estes grupos;
- Proporcionar o correto crescimento ou evolução do sistema, pois numa arquitetura bem definida, pode-se trocar, corrigir, evoluir ou integrar partes do sistema sem afetar o restante do mesmo, ou afetando de maneira prevista e controlada;
- Integração com outros sistemas, deixando claro os pontos e as formas de interface;
- Otimização para se manter o sistema enxuto e eficiente, ou seja, sem funcionalidades repetidas, mal implementadas ou não utilizadas;
- Correção, para ajudar a localizar mais facilmente o ponto errado e entender as várias conseqüências de um erro e de sua correção.

A documentação da arquitetura de software é importante porque, na fase de projeto, antecipa discussões e decisões que vão direcionar o desenvolvimento e a implantação do sistema. Na fase de manutenção ajuda a entender o sistema e minimizar o esforço para mantê-lo. Além disso, supre as necessidades de entendimento e comunicação que variam de acordo com os envolvidos e com a fase do ciclo de vida do sistema.

Para Clements et al. (2003) a documentação de arquitetura está baseada em tipos de visão. Tipos estes entendidos como a especificação do tipo de informação a ser provida por uma visão. Existem três tipos de visão e para cada tipo de visão existe um certo número de estilos arquiteturais. Estilos arquiteturais são especializações dos tipos de visão. O quadro 2 mostra a os tipos de visão e exemplos de seus estilos, além de uma breve descrição de cada estilo arquitetural.

Tipo de visão	Estilo Arquitetural	Descrição
Módulos Mostra a estrutura do sistema como um conjunto de unidades de implementação, cada uma com um conjunto de responsabilidades. Tem por finalidade: comunicação entre stakeholders e base para análise	• Decomposição	Foco: Explora relação: "é parte de"
	• Uso	Destaca a dependência funcional, explorando a relação "depende".
	• Generalização	Mostra o relacionamento de especialização entre os módulos pela relação de hierarquia "is a "
	• Camada	Mostra as unidades de implementação organizadas em camadas
Componentes e Conectores Exprime o comportamento do sistema em execução	• Canos e Filtros	Mostra as transformações sucessivas em dados processados
	• Dados compartilhados	Compartilha dados entre componentes
	• Cliente servidor	Componentes interagem requisitando serviços de outros componentes
Alocação Mapeia os elementos de software nos ambientes	• Distribuição	Mapeia os elementos de software no ambiente de hardware
	• Implementação	Mapeia os elementos nos sistema de arquivos e estrutura de desenvolvimento
	• Atribuição do trabalho	Mapeia os elementos de software nos times de desenvolvimento.

Quadro 2 – Tipos de visão e exemplos de seus estilos

4 Processo de melhoria contínua (PMC)

O processo de melhoria contínua (PMC) pode ser entendido como uma forma de se conduzir a manutenção de sistemas de software para a obtenção da melhoria contínua e pode ser considerado uma variante do modelo evolucionário incremental do ciclo de vida de sistemas, já que parte de uma situação em funcionamento e, como resultado de uma nova implementação disponibiliza uma nova versão evoluindo, desta forma, o sistema.

A figura 2 mostra uma visão geral do processo. Tendo-se como ponto de partida o sistema em produção, documenta-se, ou se complementa os documentos de arquitetura existentes, avalia-se o sistema para priorizar as ações de melhoria, emite-se um diagnóstico, faz-se a proposição da melhoria. Por fim, se implementa a melhoria consolidando-se desta forma uma nova versão do sistema em produção. Esta nova versão corresponde ao resultado de uma melhoria implementada, ou seja, a cada finalização do ciclo de melhoria, realizada de acordo com o PMC, temos o sistema em um novo patamar de qualidade. A repetição contínua do processo tende a elevar a qualidade do sistema para patamares cada vez mais altos.

Sistemas de aplicação suportam operações de negócio e portanto existem enquanto a operação do negócio existir. Desta mesma forma o processo de melhoria contínua é persistente durante toda a vida do sistema.

O sistema em produção, com a versão atualizada e, portanto em um nível de qualidade melhor, deve ser reavaliado para se verificar seu novo ponto critico. É certo que o ponto deslocou-se para uma nova necessidade. O deslocamento comum é a saída do tratamento de problemas mais básicos e de solução mais simples, como quantidade de paradas anormais em produção ou erros de consistência, para problemas com outro nível de dificuldade e de solução mais complexa. Por exemplo: usabilidade e desempenho.

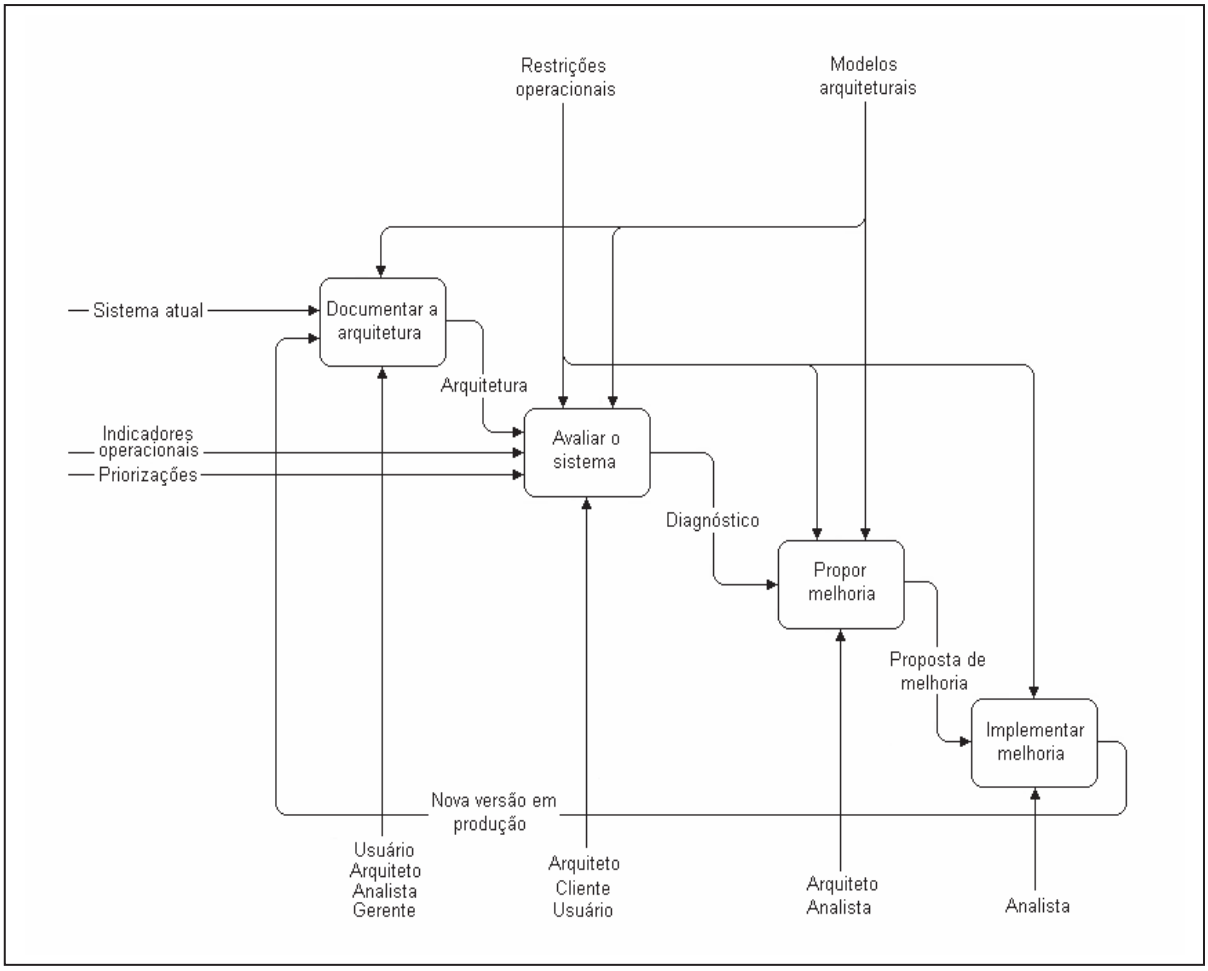

Figura 2 – PMC visão geral do processo

Indicadores operacionais são entendidos como indicadores do nível de serviço do sistema. O nível de serviço pode ser observado de diversas formas: satisfação dos usuários, atendimento aos níveis de desempenho definido, eficiência na utilização de recursos e facilidade de manutenção, por exemplo. Sendo este um processo baseado em representação arquitetural, tem a vantagem de poder proporcionar uma visão global do sistema e da interação entre as suas partes constituintes e, portanto, ajudar a avaliar situações de difícil simulação.

O quadro 3 apresenta o detalhamento do processo de melhoria contínua em seus sub-processos. Neste nível já é possível mostrar as entradas e saídas de cada atividade, além de se identificar os responsáveis e os mecanismos que podem ser utilizados para se completar esta atividade.

Quadro 3 – Sub-processos do "Processo de Melhoria Contínua"

Sub-processos	Elementos
• Documentar a arquitetura Visa obter documentos que representem os sistemas e o ambiente onde ele é utilizado. Os documentos devem permitir o entendimento do sistema e a sua avaliação. Para isso são utilizados os conceitos e documentos da arquitetura de software.	E: Sistema atual
	S: Documentos da arquitetura do sistema
	M: Reuniões / Levantamentos Conceitos da arquitetura de software
	R: Usuário Final / Arquiteto Analista de produção Desenvolvedor / Gerente
• Avaliar o sistema Tem como ponto de partida os documentos da arquitetura e os indicadores operacionais e tem como objetivo gerar um diagnóstico. Este diagnóstico deve conter: • Priorização dos problemas • Possíveis causas • Identificação dos envolvidos • Documentos arquiteturais que auxiliem o estudo da solução	E: Documentos da arquitetura Indicadores operacionais Priorizações e restrições operacionais
	S: Diagnóstico
	M: documentos da arquitetura do software ISO 9126 e 14598
	R: Arquiteto Cliente / Usuário final
• Propor melhoria Deve gerar uma proposta de melhoria com os seguintes itens: • Identificação dos itens de melhoria • Alteração da arquitetura para solução do problema • Indicadores objetivos para as métricas escolhidas	E: Diagnóstico
	S: Proposta de melhoria
	M: Documentos de arquitetura ISO 9126 e 14598
	R: Arquiteto / Desenvolvedor
• Implementar melhoria É o desenvolvimento e implementação das alterações especificadas na proposta de melhoria. Pode contemplar alterações em software, hardware, documentos e processos.	E: Proposta de melhoria
	S: Nova versão do sistema
	M: Processo de desenvolvimento
	R: Desenvolvedor Analista de produção
Legenda: E: insumos de entrada / S: produtos de saída M: mecanismo, técnica ou ferramenta utilizada para se realizar a atividade R: responsáveis pela realização	

4.1 Documentar a arquitetura

O sub-processo "Documentar a arquitetura" visa obter documentos que representem os sistemas e o ambiente onde ele é utilizado. Estes devem ainda permitir o entendimento do sistema e a sua avaliação por parte dos stakeholders. Para isso, neste trabalho, são utilizados os conceitos e documentos da arquitetura de software.

A documentação de sistemas legados, já em processo de manutenção, não é uma atividade trivial, mas partindo-se da documentação existente e focando-se na construção das visões de módulos o trabalho se torna possível. Se não houver documentação existente no que se refere a componentes e conectores e alocação, pode ser necessário recorrer-se a mecanismos de engenharia reversa.

A documentação deve ser direcionada aos envolvidos, por isso antes de se definir quais os documentos importantes deve-se identificar estes envolvidos.

Este sub-processo tem as seguintes atividades:

4.1.1 Identificar os envolvidos.

É a atividade de mapeamento das pessoas que tem relacionamento com o sistema.

O arquiteto, ou o desenvolvedor, usa as técnicas de entrevistas, levantamentos e questionários para, a partir dos sistemas e dos processos atuais, identificar os envolvidos.

4.1.2 Definir visões de interesse de cada envolvido

Atividade de se identificar qual a necessidade de visualização e grau de profundidade necessário para cada envolvido.
O arquiteto, ou o desenvolvedor, usa as técnicas de entrevistas, levantamentos e questionários, além dos conceitos de documentação da arquitetura de software para, a partir dos sistemas e os processos atuais, identificar a relação de documentos de arquitetura e a profundidade necessária para cada envolvido.
O quadro 4, baseado nas indicações de Clements et al. (2003), mostra um exemplo de um possível resultado desta atividade.

Quadro 4 – Exemplo de indicação de necessidade de documentação. Fonte Clements et al, 2003

	Tipos de visão					
	Módulos		Componentes e conectores		Alocação	
Stakeholder	Decomposição	Uso	Dados Compartilhados	Cliente servidor	Distribuição	Atribuição de trabalho
Arquiteto	d	m	d	m	d	-
Analista de sistemas	d	m	d	m	m	g
Testador	d	g	d	m	d	g
Usuário Final	g	-	g	-	g	-
Gerente	m	g	g	g	g	d
Legenda: d: informação detalhada – g: informação geral – m: grau médio de detalhamento da visão						

4.1.3 Documentar módulos

É a atividade de mapeamento do sistema como um conjunto de módulos, destacando como o sistema é decomposto e como os módulos se relacionam. O tipo de visão Módulos, englobando os estilos Decomposição, Uso, Generalização e Camada, preocupa-se em mostrar a estrutura do sistema como um conjunto de unidades de implementação. Cada uma com um conjunto de responsabilidades. No processo apresentado, tem por finalidade servir como base para análise e proporcionar uma fácil comunicação entre os *stakeholders*.
Pode ser realizada pelo desenvolvedor que, a partir dos sistemas atuais a da documentação de processos e sistemas existentes, usando conceitos da UML e da arquitetura de software, gera as várias visões de módulos do sistema.

4.1.4 Documentar componentes e conectores

Consiste de gerar os documentos que representem o comportamento do sistema em execução. O desenvolvedor ou arquiteto usando a documentação existente dos sistemas atuais e considerando os conceitos de arquitetura de software deve gerar as visões de componentes e conectores (por exemplo, cliente-servidor ou dados compartilhados) necessárias para o entendimento e a análise do sistema.

4.1.5 Documentar alocação

Nesta atividade deve-se gerar as visões que representem a alocação dos elementos do sistema de software no ambiente de hardware e a sua distribuição nos times de desenvolvimento e no sistema de arquivos. Pode-se usar a UML como instrumento de representação. De acordo com o foco sendo dado, estas visões são construídas pelo gerente, desenvolvedor ou pelo responsável pela produção.

4.1.6 Consolidar documentos da arquitetura

Atividade de se verificar a consistência e integridade dos diversos documentos gerados, padronizar a linguagem utilizada e manter a unicidade de metodologia, usando UML por exemplo.
Neste ponto o arquiteto deverá gerar um documento de arquitetura que deve ter um corpo único e padronizado sobre as visões arquiteturais, além de descrever o relacionamento entre as diversas visões.

4.2 Avaliar o sistema

Este sub-processo consiste de levantar os pontos pendentes de manutenção e obter junto aos interessados a priorização destes pontos, desta forma definindo o primeiro item da lista a ser resolvido. Também trata de verificar se a documentação disponível da arquitetura atende às necessidades de análise. Se não atender, um complemento da documentação deve ser feito. Por fim, trata da escolha de métricas para avaliação inicial do problema e da solução final. Como produto gera um documento que representa o diagnóstico do problema critico apontado como prioritário para ser solucionado. De forma mais detalhada tem as seguintes atividades:

4.2.1 Identificar os pontos críticos

Usuários, desenvolvedores e gerente responsável, através de reuniões, análise da arquitetura e de outros documentos, também considerando listas de priorizações, restrições da instalação e indicadores operacionais (avaliações sobre o nível de serviço do sistema), identificam os pontos do sistema que necessitam algum tipo de ação de manutenção.
Como nem toda necessidade de manutenção afeta a arquitetura, apesar de auxiliar em qualquer caso, o processo de melhoria contínua somente é efetivo em sua totalidade para os pontos que determinam alteração na organização ou estrutura do sistema.
Identificados estes pontos de manutenção, o consenso dos envolvidos deve gerar uma ordem de prioridade para o atendimento, determinando assim o ponto crítico a ser resolvido primeiramente. A definição desta ordem é importante para que se possa iniciar o ciclo do processo de melhoria contínua. Um exemplo do resultado desta atividade está indicado no quadro 5.

Quadro 5 - Exemplo de lista de prioridades

Priorização	Necessidade	Motivo
1	Diminuir o tempo de testes	Atualmente 40% do tempo de testes representa os testes de novas funcionalidades e 60% são testes regressivos.
2	Diminuir a interferência manual nos processos do sistemas	O processo atual tem etapas manuais e automáticas intercaladas e sem controle efetivo.
3	Unificar sistemas integrados em plataforma única	Falta domínio da tecnologia e os tempos de processamento muito altos.

4.2.2 Revalidar visões arquiteturais

De acordo com o ponto critico apontado como de maior prioridade, deve-se indicar quais visões arquiteturais vão orientar a solução. Nesta atividade deve-se avaliar se as visões disponíveis são suficientes para a análise e proposição de solução para o problema. Caso não sejam, novas visões devem ser criadas. Como ponto de partida tem-se a documentação da arquitetura e uma lista de prioridades. Como produto da atividade têm-se as representações arquiteturais revisadas ou complementadas. Os mecanismos usados pelo arquiteto ou pelo desenvolvedor, para apoio a esta atividade, são a UML como instrumento de documentação e os conceitos da arquitetura de software como instrumento de estruturação e organização do sistema.
Outro instrumento de apoio a esta atividade é o quadro 6 aonde se mostra exemplos de relacionamento entre ponto crítico, sua necessidade de visualização e possíveis artefatos arquiteturais que permitam seu estudo. Este quadro exemplo não é exaustivo, mas mostra a forma de se chegar aos artefatos arquiteturais necessários. Para cada ponto crítico apontado deve-se construir um quadro semelhante ao quadro 6. O resultado não é único e depende da maturidade e experiência do arquiteto e dos outros stakeholders.

Quadro 6 - Exemplos de relacionamento problema crítico versus artefato arquitetural

Ponto crítico	Necessidade de visualização	Artefato arquitetural sugerido
Alterar módulo, substituir ou criar novo	Integração de módulos Interfaces entre módulos	Visão da decomposição dos módulos Visão da distribuição dos módulos
Falha de segurança ou acesso indevido a dados	Estrutura e pontos de controle	Visão da distribuição dos módulos Visão do estilo em Camadas
Demora em fazer alterações, acertos, testes, etc	Estruturação dos sistemas e das interfaces Organização do desenvolvimento Nível de acoplamento	Visão da atribuição do trabalho Visão de uso Visão da decomposição dos módulos

4.2.3 Definir métricas

A escolha de métricas deve necessariamente levar em consideração o objetivo a se atingir. Não pode ser genérica, mas sim específica e se relacionar diretamente com o problema que está sendo tratado. O desenvolvedor, neste caso, parte do ponto critico escolhido e, direcionado pela norma ISO 9126, define as métricas e indicadores para se avaliar o estágio atual e o desejado.

Como instrumentos de apoio e auxílio à realização desta atividade tem-se o quadro 7 que mostra exemplos de classificação dos problemas críticos segundo ISO 9126 e o quadro 8 que dá exemplos de métricas sugeridas por característica de qualidade. Estes quadros não esgotam as possibilidades mas mostram um caminho a ser explorado na determinação das métricas.

As características e subcaracterísticas a serem observadas a partir do ponto crítico apontado são direcionadas pelo ponto que mais preocupa os usuários, pois este é que vão determinar as métricas pelas quais a solução do problema será avaliada. Deve-se ter em mente que um ponto critico pode afetar outras características além da principal apontada pelo usuário.

Quadro 7 - Classificação dos problemas críticos segundo ISO 9126

Ponto crítico / Necessidade	Classificação ISO 9126	
	Característica	Subcaracterística
Substituir sistema Alterar módulo, substituir ou criar novo Aceitar a alteração em sistema externo integrado	Funcionalidade Manutenibilidade	Adequação Modificabilidade Interoperabilidade
Demora para fazer alterações, acertos, testes, etc	Manutenibilidade	Modificabilidade Analisibilidade
Falha de segurança ou acesso indevido a dados	Funcionalidade	Segurança de acesso
Mudança de plataforma, evolução do ambiente operacional	Portabilidade	Adaptabilidade

Para construção do quadro 8 foram usados direcionamentos e exemplos da norma NBR ISO/IEC 9126, mas não literalmente as métricas que lá constam. A obtenção de novas métricas quando da aplicação do PMC deve ser feita pelo arquiteto levando em consideração o conhecimento e a capacidade dos stakeholders de visualizar a medida do problema por eles apontados.

Quadro 8 - Exemplos de métricas sugeridas por característica de qualidade

Classificação ISO 9126		Métrica
Característica	Subcaracterística	
Confiabilidade	Maturidade	Tempo médio entre falhas
	Tolerância a falhas	Falhas evitadas / Total de casos de teste
	Recuperabilidade	Tempo de recuperação pós-falha
	Conformidade	Atendimento aos padrões e normas
Manutenibilidade	Analisabilidade	Tempo entre a informação de erro e a descoberta do motivo do erro
	Modificabilidade	Tempo de alteração / tempo total da tarefa
	Estabilidade	Erros pós-implantação / número de implantações
	Testabilidade	Tempo de testes / tempo total da tarefa
	Conformidade	Projetos com roteiro de testes / total de projetos

4.2.4 Montar diagnóstico

Atividade de consolidação da análise a da documentação gerada de modo a se manter a consistência e organização do processo. Aqui o desenvolvedor ou arquiteto, partindo da documentação da arquitetura, dos indicadores e métricas definidas, gera um documento formal de diagnóstico usando, como sugestão, a UML para as representações arquiteturais.

4.3 Propor soluções e elaborar proposta de melhoria

A atividade de proposição de melhoria pode ser bastante abrangente chegando no limite a propor a reengenharia completa do sistema ou mesmo a sua substituição. As soluções podem ser as mais variadas e sempre vai depender do problema e do ambiente existente em cada caso Para os termos deste estudo o assunto não é detalhado por ser muito extenso.

4.3.1 Estudar alternativas de melhoria e escolher a mais adequada

É a atividade onde usuários, arquiteto e desenvolvedor, tendo o diagnóstico em mãos, discutem as possibilidades de solução do problema apontado com crítico e, em consenso, escolhem uma alternativa a ser implementada.

4.3.2 Detalhar a alteração da arquitetura do sistema

Partindo da alternativa de solução escolhida, o arquiteto detalha as alterações necessárias na arquitetura do sistema. Uma nova arquitetura resulta da implementação da alteração proposta sobre a arquitetura atual. Para realizar o seu trabalho, o arquiteto deve levar em consideração: aderência ao ambiente e arquitetura atuais, restrições de recursos e conformidade com padrões e regras existentes.

4.3.3 Estabelecer objetivos de métricas

Na fase de diagnóstico foram definidas as métricas para se avaliar o status do sistema antes da alteração e após a implementação das alterações na arquitetura. O estabelecimento de métricas de avaliação da solução proposta é fundamental para ter-se a medida do sucesso da implementação. Nesta atividade definem-se os objetivos das métricas escolhidas na fase de diagnóstico.

4.3.4 Elaborar a proposta

Consolidar as definições anteriores gerando uma proposta formal de melhoria da arquitetura do sistema. A importância da geração de um documento com formalizações e aprovações cria compromissos, padroniza as visões e permite o gerenciamento efetivo das atividades da manutenção.

4.4 Implementar a melhoria

O desenvolvimento e a implantação da alteração, apesar de fazer parte do processo de melhoria, não foi o foco principal deste estudo, por isso este sub-processo está descrito de maneira sucinta.

4.4.1 Implementar a alteração

Corresponde a executar as tarefas necessárias para implementar a alteração definida. Estas alterações podem incluir: desenvolvimento ou alteração de software, compra e integração de pacotes, desativação de sistemas, compra e instalação de hardware.

4.4.2 Testar a alteração

Validar as alterações realizadas. Esta atividade corresponde a verificação de conformidade com normas e padrões, testes individuais e testes de homologação com os usuários.

4.4.3 Implantar as alterações

Uma vez completados os testes e homologadas as alterações, é a atividade de se transferir efetivamente, estas alterações, para o ambiente de produção.

4.4.4 Avaliar segundo as métricas definidas

Deve ser feita a avaliação da alteração seguindo-se os critérios do que foi determinado na proposta de melhoria. A comparação dos resultados da situação anterior e posterior a alteração da arquitetura é fundamental para se conduzir o processo de melhoria de maneira adequada.

5 Exemplo de utilização do PMC

O contexto no qual se desenvolve este exemplo de aplicação é o de uma empresa do mercado de seguros de varejo, mais precisamente seguros de automóvel, vida e multirriscos. Os sistemas têm processamento prioritariamente centralizado e estão residentes em computador de grande porte. O software em operação tem mais de vinte anos de construção e tem atendido satisfatoriamente as áreas usuárias. Este software, no decorrer do tempo vem sofrendo constantes manutenções. A experiência relatada decorreu da somatória de problemas acumulados que começaram a interferir no trabalho cotidiano das áreas usuárias. Estes problemas foram identificados como críticos e inseridos no processo de melhoria.

5.1 Documentação da arquitetura

A aplicação do PMC inicia-se com a geração da documentação da arquitetura do sistema. A figura 3 mostra um esquema com exemplos das várias visualizações possíveis para documentar a arquitetura do sistema.

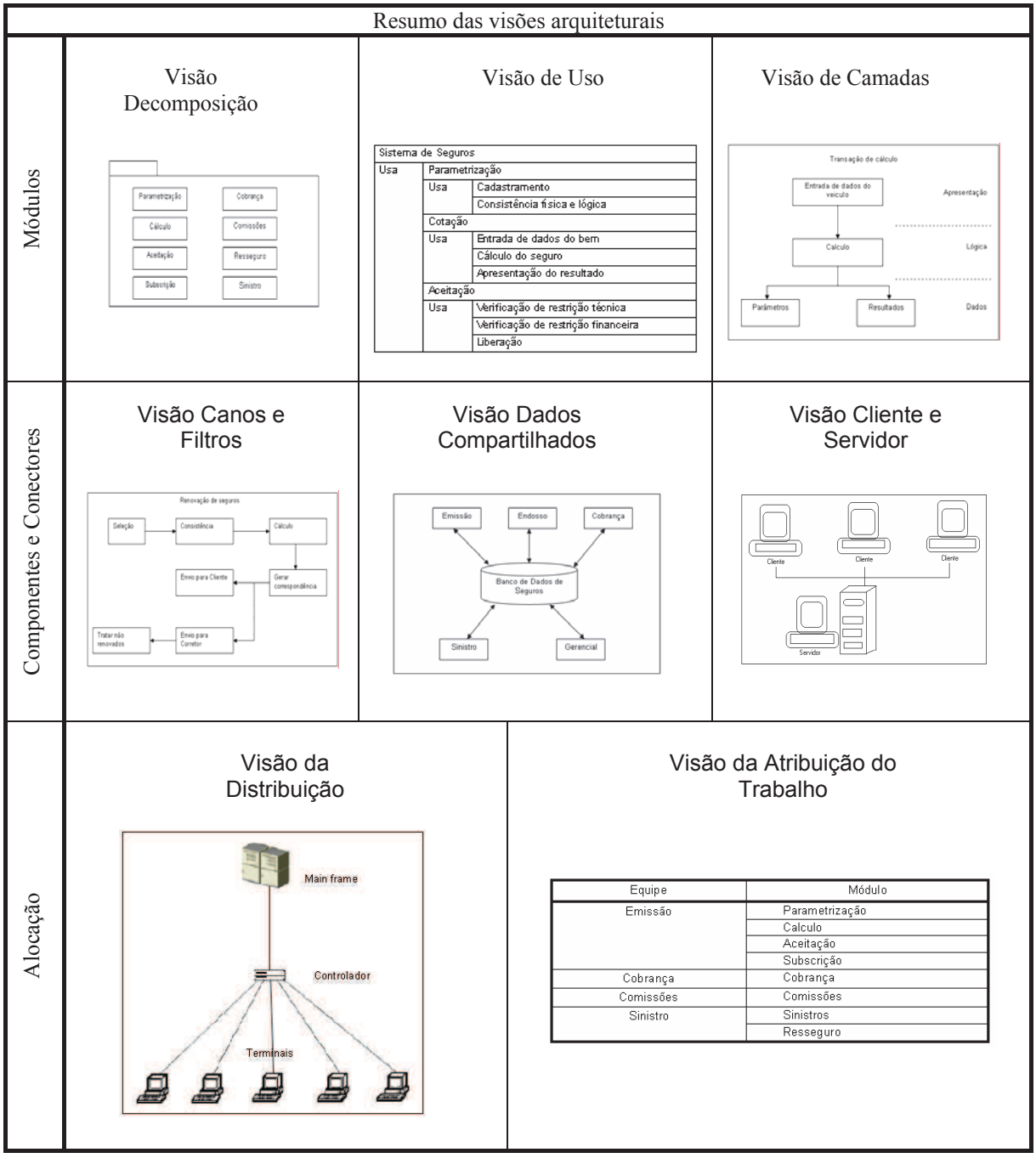

Figura 3 – Resumo das visões arquiteturais

5.2 Avaliação do sistema

Após a realização de reuniões e discussões sobre as características gerais e sobre os problemas do sistema, atendendo aos sub-processos "1- Documentar a arquitetura" e "2 – Avaliar o sistema", houve a identificação, sob o ponto de vista dos usuários e dos desenvolvedores, de três pontos críticos. A lista de pontos críticos considerou somente aqueles itens que requeiram alterações estruturais no sistema, casos de manutenção pontuais não foram considerados. Estes três pontos críticos, conforme quadro 9, foram priorizados na ordem mostrada. Também foram listados os possíveis motivos para a ocorrência de cada um destes problemas.

Neste artigo somente a primeira iteração, correspondente a solução do primeiro item priorizado, do PMC é demonstrada. Para a conclusão da seqüência outros dois ciclos foram realizados.

O processo, sendo contínuo, determina que após a execução de uma iteração o sistema deve ser reavaliado para a definição do novo ponto crítico a ser tratado de forma cíclica.

Quadro 9 – Priorização dos problemas críticos

Priorização	Problema / Necessidade	Motivo
01	Alteração nas rotinas de cotação de seguros para um usuário gera testes para todas as áreas. O processo como um todo é demorado;	• Existe um mesmo conjunto de programas para atender a diversas áreas usuárias; • O controle das variáveis de parametrização é deficiente; • O ambiente de desenvolvimento não permite manutenções paralelas nos mesmos objetos.
02	Processo de renovação contém interferência manual	• Diversos (não detalhado neste artigo)
03	A geração e a administração de taxas não é eficiente.	• Diversos (não detalhado neste artigo)

Dentro do desenvolvimento do sub-processo "2 - Avaliar o sistema", para possibilitar o entendimento e a discussão sobre os problemas apontados, usando o exemplo de Clements et al. (2003), foram mapeadas as necessidades de documentos conforme o quadro 10. Ele mostra as principais visões arquiteturais indicadas como importantes para o entendimento geral do sistema e para o estudo de soluções do problema apontado. A montagem deste quadro faz parte do trabalho do arquiteto que, para realizá-lo, leva em consideração as características do problema e os interesses dos outros stakeholders envolvidos.

Quadro 10 – Visões de interesse

	Tipos de visão				
	Módulos		Componentes e conectores	Alocação	
Stakeholder	Decomposição	Uso	Dados Compartilhados	Distribuição	Atribuição de trabalho
Gerente	m	g	g	g	d
Arquiteto	d	d	d	-	g
Desenvolvedor	d	d	d	-	-
Testador	d	m	d	-	-
Usuário Final	g	g	-	-	-
Legenda: d: informação detalhada – g: informação geral – m: grau médio de detalhamento da visão					

O quadro 11 é um extrato do sub-processo "2 - Avaliar o sistema" e resume as informações do diagnóstico para solução do problema critico apontado: ineficiência das rotinas que envolvem o cálculo dos seguros.

Para a solução do problema em questão foram identificados e listados os fatores que afetam o usuário de maneira negativa. Do relacionamento destes fatores com as características e subcaracterísticas de qualidade da norma NBR ISO/IEC 9126 e, como resultado do consenso entre os envolvidos, definiram-se as métricas que deveriam ser utilizadas para avaliar o resultado da melhoria. As métricas são: "Efetividade dos testes", "Fator de análise" e "Efetividade da divisão das equipes". Para cada uma delas foram medidos os valores iniciais, antes de se alterar o sistema, e definidos os novos valores a serem atingidos após a implementação da solução proposta.

A métrica "Efetividade dos testes" demonstra o quanto do tempo total dos testes foi realmente aplicado em testar as novas funções. Na situação inicial somente 40% do tempo total dos testes foi aplicado nas novas funções e 60% para validar a continuidade de funções existentes. Para esta métrica estabeleceu-se o objetivo de se chegar a 90% do tempo total de testes para as novas funções.

Para "Fator de análise", que significa o quanto do tempo total para resolução de um problema se gasta em análise, a situação inicial determinava que 60% do tempo total de alteração gastava-se am análise. Para melhoria foi determinado o objetivo de se reduzir a 30% o tempo necessário para se localizar a solução de um problema.

Finalmente, na métrica "Efetividade da divisão das equipes", que está relacionada à espera que cada solicitação de serviço tem para ser atendida, somente 33% dos pedidos eram atendidos e 67% ficavam pendentes aguardando a liberação dos recursos por outra equipe. Determinou-se o objetivo de se atingir o parâmetro de 90% dos itens sem aguardar liberação de recursos de outra equipe.

Quadro 11 – Alteração do cálculo - Métricas de qualidade

<table>
<tr><td colspan="2">Problema / Necessidade:</td><td colspan="3"><u>Alterar cálculo de seguros</u></td></tr>
<tr><td rowspan="2">Fatores que influenciam</td><td rowspan="2">Característica</td><td rowspan="2">Sub-característica</td><td colspan="2">Métrica</td></tr>
<tr><td>Medida</td><td>Atributo</td></tr>
<tr><td rowspan="2">1. Alteração para um usuário afeta todos os outros. Área que não solicitou alteração também tem que testar;</td><td>Manutenibilidade</td><td>Testabilidade</td><td rowspan="2">TT / TTT
TT=Tempo de Testes de novas funções
TTT= Tempo Total de Testes</td><td rowspan="2">Inicial: 0,4
Objetivo: 0,9</td></tr>
<tr><td colspan="2">Métrica: Efetividade dos testes</td></tr>
<tr><td rowspan="2">2. Tempo grande demandado para identificar ponto de problema e realizar alterações;</td><td>Manutenibilidade</td><td>Analisibilidade</td><td rowspan="2">TA / TTA
TA = Tempo de Análise
TTA= Tempo total da Alteração</td><td rowspan="2">Inicial: 0,6

Objetivo 0,3</td></tr>
<tr><td colspan="2">Métrica: Fator de análise</td></tr>
<tr><td rowspan="2">3. Mudanças são seqüenciais e não paralelas. Não adianta ter equipes especificas.</td><td>Manutenibilidade</td><td>Modificabilidade</td><td rowspan="2">1 - SE / TS
SE = Solicitações de manutenção do módulo em espera
TS = Total de solicitações de manutenção do módulo</td><td rowspan="2">Inicial: 0,33

Objetivo 0,90</td></tr>
<tr><td colspan="2">Métrica: Efetividade da divisão das equipes</td></tr>
</table>

5.3 Proposição da melhoria

Para o problema em estudo existiam diversas alternativas de solução. Dentre elas: rever e alterar o processo de parametrização, viabilizar ferramenta de controle de versões e especializar os módulos de cálculo. Apesar de inicialmente parecer a menos eficiente por aumentar o número de programas, escolheu-se a alternativa de especializar os módulos de cálculo (figura 4), criando um módulo para cada área. Esta especialização tem vantagens, pois, cada área realiza somente os testes das alterações que solicitou e a empresa pode efetivamente usar equipes de desenvolvimento específicas para o atendimento de cada área usuária, desta forma focando o atendimento e diminuindo o tempo das manutenções.

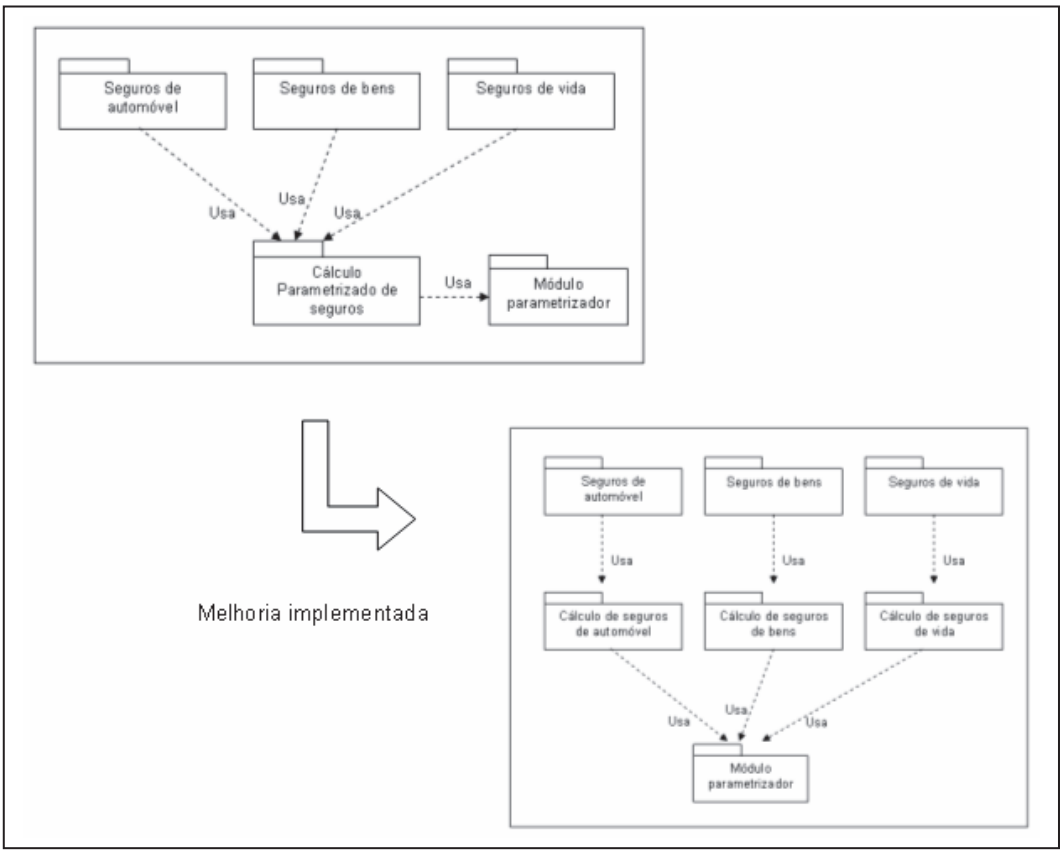

Figura 4 – Alteração do cálculo - proposta de mudança arquitetural

5.4 Implementação e resultados obtidos

A implementação foi realizada alterando-se os programas, adequando-se os processamentos e criando-se novos processos de trabalho para os usuários. Conforme o planejamento, os resultados operacionais foram os seguintes: cada área passou a realizar somente os testes das alterações que solicita, a empresa pode usufruir efetivamente da montagem de equipes de desenvolvimento específicas para o atendimento de cada área usuária. Os resultados da avaliação das métricas realizada após a implantação estão mostrados na tabela 1.

Tabela 1 – Alteração do cálculo – valores das métricas após implantação

Atingimento das métricas			
Métrica	**Valor do Atributo**		
	Anterior	**Objetivo**	**Atingido**
Efetividade dos testes	0,4	0,9	0,75
Fator de análise	0,6	0,3	0,15
Efetividade da divisão das equipes	0,33	0,90	0,85

Da análise dos resultados, considerou-se que a implementação da alteração atingiu os objetivos a contento. Os desvios apresentados, entre os valores objetivos e os valores atingidos, foram decorrentes do fato de que o conjunto de parâmetros que atendia aos diversos produtos de seguros não foi especializado, pois não se considerou vantajosa esta manutenção. Além disso, estes parâmetros têm manutenção unificada para todas as áreas usuárias.

6 Conclusões

O processo apresentado traz melhores resultados para a manutenção dos sistemas porque ao partir dos pontos críticos consensados entre técnicos e usuários define o que vai ser feito de maneira clara; ao basear-se em visões arquiteturais para conduzir o processo permite a equalização de conhecimentos, a discussão sobre o estágio atual do sistema e antecipa a discussão sobre a solução proposta e ao utilizar métricas para avaliar a situação anterior e posterior à alteração, alinha as expectativas entre os stakeholders sobre os resultados a serem obtidos.

A principal limitação à aplicação de um processo de melhoria contínua nas organizações continua sendo a pressão por soluções imediatas, mesmo que parciais ou superficiais. No caso estudado, verificou-se que somente foi aprovada a aplicação do processo nos três problemas críticos citados, lembrando que somente o primeiro deles foi descrito neste artigo. Após a realização das três iterações o processo não teve continuidade.

Sob o aspecto técnico, o tipo de visão módulo, em seus estilos: decomposição, uso e camadas, mostrou-se o mais adequado para documentar a arquitetura do sistema quando o objetivo é o entendimento e discussão de problemas e soluções entre os diversos *stakeholders*. Os tipos de visão de alocação e componentes e conectores fornecem visões complementares para a condução do processo de melhoria.

O processo descrito deu ênfase às etapas de documentação e avaliação, ressaltando a geração da documentação da arquitetura e a atribuição de métricas anteriores e posteriores a implementação das alterações. Estudos futuros poderiam aprofundar as possibilidades de se ter a melhoria contínua também se investindo em novas formas de proposição soluções e de implementação das alterações nos sistemas.

Além disso, para que o PMC possa se consolidar como mecanismo efetivo de condução da manutenção, outras linhas de estudo futuro também podem ser desenvolvidas. A investigação de um modelo econômico, para avaliação do processo como um todo, visando demonstrar os limites de investimentos que podem ser feitos para que valha a pena a implementação do PMC é uma delas. Outra possibilidade seria o estudo da introdução de uma ferramenta para registro e acompanhamento do processo que permitisse a geração de uma base de conhecimento visando realizar ajustes para a obtenção do amadurecimento do mesmo.

7 Referências bibliográficas

- Clements, Paul; Garlan, David; Little, Reed; Nord, Robert; Stafford, Judith; **Documenting Software Architectures in an Agile World**. Technical Note CMU/SEI-2003-TN-023.
- Khan, Khaled e Zheng, Yan. **Managing Corporate Information Systems Evolution and Maintenance.** Idea Group Publishing. Julho, 2005.
- L. Bass, P. Clements and R. Kazman, **Software Architecture in Practice**. Addison-Wesley, 1998.
- Lehman, M. M. "**Laws of Software Evolution Revisited**," presented at EWSPT96, 1997
- Mendes, Antonio. **Arquitetura de Software. Desenvolvimento orientado para arquitetura**. Ed. Campus. São Paulo, 2002.
- NBR ISO/IEC 14598-1, **Tecnologia de informação – Avaliação de produto de Software. Parte 1: Visão geral**, 2001.
- NBR ISO/IEC 14598-2, **Engenharia de software – Avaliação de produto. Parte 2: Planejamento e gestão**, 2003.
- NBR ISO/IEC 9126-1, **Engenharia de software – Qualidade de produto. Parte 1: Modelo de qualidade**, 2003.
- Pfleeger, S. L. **Engenharia de Software: Teoria e Prática**, Tradução: Dino Franklin. Ed. Prentice Hall. São Paulo, 2004.
- Pigoski, T. M. **Practical Software Maintenance – Best Practices for Managing your Software Investment**, John Wiley and Sons. USA, 1996.
- Pressman, R. S. **Software Engineering: A Practitioner's Approach**, McGraw-Hill, 2001.
- Seacord, R. C. Plakosh D. e Lewis, G. A. **Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices.** Addison-Wesley Professional. 2003.
- Shaw, Mary e Garlan, David. **Software Architecture. Perspectives on an emerging discipline.** ACM SIGSOFT Software Engineering Note. Outubro, 1996.