# Um Método para Suporte à Evolução de Métricas de PDS

**Leandro P. Bogoni[1], Duncan D. A. Ruiz[1]**

[1] Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS – Porto Alegre – RS – Brasil

bogoni@gmail.com,duncan@inf.ufrgs.br

***Abstract.*** *This work presents a method to extract, organize and present Software-Development-Process (SDP) metrics, taking into account the evolution of SDP scheme and corresponding metrics program. Based on a Data Warehousing environment approach, the proposed solution targets the recall of metrics previously captured, considering different SDP models and metrics programs. To achieve such metrics recall, a set of procedures on how to properly handle metrics inserting, updating and deleting are proposed. The goal is to build a solid information-base of SDP metrics. The requirements for the method were identified in an Information Technology organization, CMMI2-certified, which the majority of software projects are business process-oriented. The main contribution of this work is the ability to properly store and handle current and past metrics into a unique metrics repository. Besides, older metrics remain comparable to current ones, focusing on a better control of software projects and higher quality of corresponding software products.*

***Resumo.*** *Este trabalho apresenta um método para extração, organização e apresentação de métricas para Processo de Desenvolvimento de Software (PDS), levando em consideração a evolução do próprio PDS e do conjunto de métricas correspondente. A solução, baseada em um ambiente de Data Warehousing, tem a finalidade de resgatar medições feitas em projetos passados, sob diferentes modelos de PDS e programas de métricas e formar uma base sólida de informações desses projetos. Para viabilizar o resgate destas medições, são propostos procedimentos para tratar adequadamente a criação, alteração e exclusão de métricas. Tais requisitos foram identificados em uma empresa de Tecnologia de Informação, certificada CMMI nível 2, cuja principal característica é que a maioria das aplicações desenvolvidas é voltada à automação de Processos de Negócio. A principal contribuição deste trabalho é a de permitir que medições presentes e passadas possam ser mantidas em um repositório único de métricas da organização e que as mesmas sejam comparáveis, viabilizando um melhor controle dos projetos de software e qualidade de seus produtos.*

## 1. Introdução

Na busca por qualidade, as organizações promovem o aperfeiçoamento de seu Processo de Desenvolvimento de Software (PDS) com o intuito de prever, com melhor precisão, os prazos e custos, aumentando assim a competitividade. Seguindo a abordagem

encontrada em Harrington *apud* List (2004) onde, para melhorar é preciso controlar, para controlar é preciso medir, entende-se que coletar métricas é uma atividade importante para contribuir no aperfeiçoamento de PDS. As métricas coletadas devem prover informações para o auxílio na tomada de decisões de acordo com os objetivos e estratégias da organização. Segundo Gopal *et al* (2005), as atividades necessárias para a coleta, organização e análise das métricas devem ser guiadas por um Programa de Métricas. Em decorrência do aperfeiçoamento dos PDSs, ocorre uma evolução dos Programas de Métricas.

O enfoque principal deste trabalho está na definição de um método para a extração, organização e apresentação de métricas para PDSs, levando em consideração a evolução desses PDSs e do conjunto de métricas correspondente. A principal contribuição refere-se ao auxílio na aplicação de Programas de Métricas em empresas de Tecnologia de Informação (TI) que buscam o aperfeiçoamento contínuo de seu PDS, permitindo que medições presentes e passadas possam ser mantidas em um repositório único de métricas da organização e que as mesmas sejam comparáveis, viabilizando um melhor controle dos projetos de *software* e qualidade de seus produtos.

## 2. Fundamentação Teórica

### 2.1 Qualidade no Processo de Desenvolvimento de Software

A busca por soluções para melhorar a qualidade de software vem sendo feita há muito tempo. Hoje, com a rápida inovação tecnológica, a chave para a sobrevivência de empresas de software é contar com uma melhoria contínua de seu processo (Niazi *et al* 2005). Esta melhoria contínua traduz-se na evolução do PDS, tratada na literatura como a aplicação de normas e modelos de *software process improvement* (SPI). Um dos principais modelos de SPI é o *Capability Maturity Model Integration* (CMMI).

### 2.2 Programa de Métricas

No contexto de PDS, Medição e Análise é uma das principais áreas de processo e tem um impacto direto em todas as outras áreas do CMMI. O propósito da Medição e Análise é desenvolver e sustentar uma capacidade de medição que é usada para dar suporte ao gerenciamento de informações (Palza *et al*, 2003).

Segundo Gopal *et al* (2005), um Programa de Métricas de Software deve compreender um conjunto de atividades para definir, implementar, operar e manter um sistema de informações de coleta, análise e apresentação de métricas de PDS, produtos e serviços. Um Programa de Métricas necessita da combinação de padrões, procedimentos, técnicas e ferramentas de suporte.

Para auxiliar na gestão de PDS, os valores das métricas coletadas devem estar acessíveis, consistentes e organizados a fim de facilitar o seu acesso e utilização pelos responsáveis. Segundo Palza *et al* (2003), o CMMI sugere que os valores das métricas sejam armazenados em um Repositório de Métricas da Organização. Este repositório é usado para manter dados disponíveis do processo e dos produtos, contendo dados de métricas e informações relacionadas, necessárias para entender e analisar estas métricas.

## 3. Trabalhos Relacionados

### 3.1. SPDW: a Software Development Process Performance Data Warehousing Environment

Em Becker *et al* (2006), é proposto um ambiente de *Data Warehousing* para dar suporte a programas de métricas para PDS, permitindo o armazenamento dos dados de medição de diferentes projetos em uma base de dados unificada e centralizada. Este ambiente permite que o acesso aos dados seja realizado por meio de interfaces que possibilitem consultas e análises de forma simplificada sobre a perspectiva das métricas, trazendo uma visão unificada dos diferentes projetos da organização.

A arquitetura de três camadas proposta nesse trabalho endereça os problemas advindos da heterogeneidade, dos projetos e de ferramentas de captura, e armazenamento dos dados próprios a cada projeto. A camada de integração de aplicações é responsável pela extração de dados brutos dos projetos provenientes das diversas ferramentas e pela carga no *Data Staging Area* (DSA). Cada projeto é descrito por metadados, expressos em XML *Schema*, que parametrizam a implementação dos *wrappers*. Os metadados de projeto definem as ferramentas adotadas e como os dados requeridos são armazenados nessas ferramentas. Na camada de integração de dados, os dados extraídos das ferramentas do ambiente transacional são limpos e transformados no DSA, por meio de rotinas de limpeza e transformação, auxiliadas pelos metadados organizacionais. Após todo esse processo, os dados consolidados são carregados no *Data Warehouse* (DW). O armazenamento de dados dos PDSs, do ponto de vista organizacional, é realizado por meio de um modelo analítico multidimensional voltado ao acompanhamento das métricas da organização. A camada de apresentação do ambiente citado possibilita o acesso diferenciado para cada perfil de usuário, por meio de permissões. Essa proposta provê, ainda, a realização de consultas OLAP sobre os dados relativos a estas métricas, e a visualização destes por meio de interfaces que provêm formas de análise simplificadas sobre os dados gerados.

### 3.2. Multidimensional Measurement Repository (MMR)

Em Palza *et al* (2003) é proposto o *Multidimensional Measurement Repository* (MMR), com o intuito de coletar, armazenar, analisar e reportar os dados de medições, baseado nos requisitos do CMMI. O MMR visa organizar e armazenar dados de medidas históricas de projetos para auxiliar no seu acompanhamento e na análise de tendências, provendo assim meios para aumentar a qualidade dos produtos. O MMR baseia-se em *Data Warehousing* de métricas e disponibiliza consultas sobre os dados por meio de técnicas OLAP. Este repositório é extensível, permitindo a adoção de diferentes medidas, dando suporte à organização ao longo de sua evolução de maturidade.

O MMR utiliza-se de metadados para permitir a evolução do repositório de métricas da organização, atendendo às Áreas de Processo do CMMI. As versões do repositório são mantidas, e as informações de cada projeto são armazenadas na versão do repositório, referente à versão do PDS em que cada projeto foi concebido.

### 3.3. Versionamento e Evolução de Esquemas

Em Galante *et al* (2002) e Jensen and Dyreson (1998), são apresentadas duas técnicas que permitem modificações no esquema de banco de dados, mantendo a consistência entre o esquema e a extensão de dados. Estas técnicas são: a evolução de esquemas e o versionamento de esquemas. Basicamente, a diferença entre elas é que a evolução de esquemas mantém somente o esquema mais recente, juntamente com sua extensão de dados, enquanto que o versionamento de esquemas preserva todas as versões de esquema e suas extensões de dados durante a evolução.

Na evolução de esquemas, quando ocorre uma alteração, o novo esquema torna-se o esquema corrente e o esquema anterior é descartado. Os dados antigos, baseados no esquema antigo são convertidos para o novo esquema. No versionamento de esquemas, quando um novo esquema é criado, os dados do esquema anterior são adaptados para este novo esquema. Porém, os esquemas anteriores e suas respectivas extensões de dados são preservados e permanecem armazenados no sistema de banco de dados como versões antigas.

## 4. Caracterização do Problema

As melhorias contínuas aplicadas aos PDS's das Organizações promovem adaptações em suas atividades e artefatos, impactando diretamente no Programa de Métricas dessas organizações. Quando artefatos são incluídos, excluídos ou modificados, as métricas que tinham como entrada dados desses artefatos também sofrem alterações. Conforme o processo vai evoluindo, seja naturalmente ou por demanda de certificação, surgem novas necessidades referentes ao seu gerenciamento. Desse modo, novas métricas podem ser necessárias para responder às novas questões que surgem, algumas métricas podem ser adaptadas (modificadas) e outras métricas podem ter suas coletas abandonadas por julgar-se não serem mais úteis. Neste contexto, ocorre uma evolução do Conjunto de Métricas da Organização.

Existe também uma preocupação quanto ao armazenamento das informações coletadas junto aos PDSs. A utilização de um Repositório Organizacional de Métricas facilita o acesso a essas informações, propiciando um melhor gerenciamento delas. Com a aplicação de um processo de *Data Warehousing,* a extração das métricas, o cálculo das métricas derivadas e a carga no repositório, estão contemplados nos conceitos de extração, transformação e carga (ETL) definidos por Kimball (1998), porém, problemas advindos da heterogeneidade de projetos e ferramentas utilizadas no armazenamento dos dados próprios a cada projeto, acabam exigindo uma solução mais flexível, para dar suporte adequado à análise dessas métricas.O uso de informações de projetos passados para o planejamento, a elaboração de estimativas e o controle de projetos em andamento depende de uma base sólida de métricas da organização. A utilização dessas informações é dificultada pela evolução do PDS e do programa de métricas. Devido às alterações no conjunto de métricas, os dados coletados em projetos que seguiram versões diferentes do PDS podem não ser passíveis de comparação. O número de projetos de software, mesmo em empresas de TI de grande porte, é considerado pequeno se comparado, por exemplo, com números típicos de execução de processos de negócio em organizações de médio/grande porte. Assim, utilizar apenas as métricas de projetos que estão na mesma versão de PDS pode não ser suficiente para prover recursos aos

gestores, e contribuir no aumento da qualidade do planejamento e execução de projetos de software.

Existem na literatura alguns trabalhos que abordam questões relacionadas à implantação de Programas de Métricas. Porém, os trabalhos encontrados não suprem todas as necessidades descritas no cenário desta pesquisa, pois não garantem que os dados de projetos concebidos em versões diferentes de PDS sejam comparáveis entre si.

## 5. Solução Proposta

O princípio básico desta solução está fundamentado no trabalho de Becker *et al* (2006), que foi estendido para dar suporte às questões relacionadas à evolução do PDS e do Programa de Métricas, as quais não eram tratadas na proposta original. São propostas alterações na arquitetura do trabalho citado, nas camadas de integração de aplicações e integração de dados.

A arquitetura proposta (Fig. 1), composta de três camadas é orientada a serviços, utilizando a facilidade desta abordagem para o acesso aos dados de projetos nas ferramentas de apoio. Faz-se necessário o uso de metadados, os quais servem de configuração para a captura de cada métrica, para cada projeto. Quando ocorre uma evolução do PDS, e, consequentemente do Conjunto de Métricas, esses metadados devem ser alterados para dar suporte à coleta das novas métricas, ou a alteração das métricas existentes.

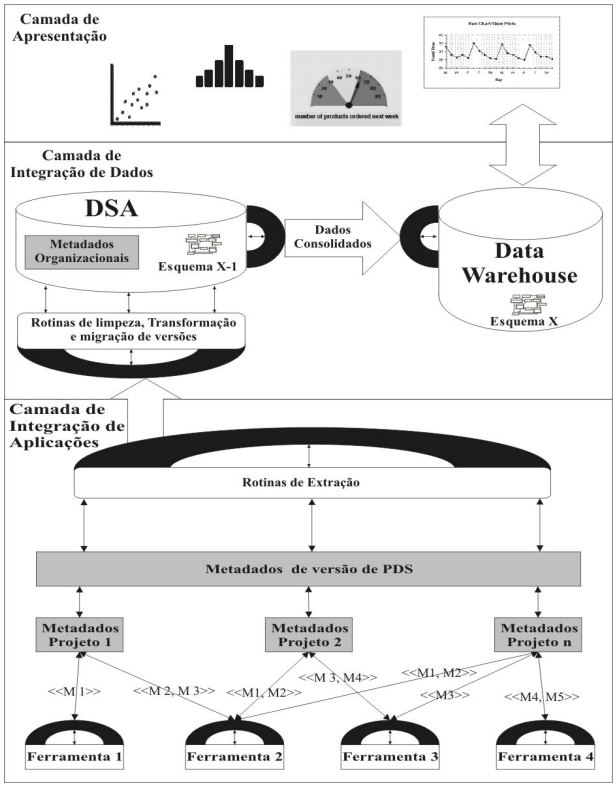


**Fig. 1. Arquitetura de Data Warehousing para PDS em evolução**

### 5.1. Camada de Integração de Aplicações

Na Camada de Integração de Aplicações, em Becker *et al* (2006), os metadados são utilizados em nível de projeto, com a configuração da localização, forma de captura e cálculo de cada métrica. Com as características de evolução, surge a necessidade de metadados em nível de versão de PDS, os quais são utilizados com o intuito de adaptar a coleta de cada projeto em sua versão original, para a versão atual do PDS. Para ilustrar a utilização dos metadados, pode-se tomar um PDS Exemplo, que esteja na versão X, e seu respectivo Conjunto de Métricas, como exposto na Fig. 2 (a). Supondo que houvesse uma evolução neste PDS Exemplo para a versão X+1, e que o Programa de Métricas tivesse a criação das métricas de Tamanho, e que as métricas de Custo fossem analisadas também pelo ponto de vista do custo Sub-contratado, o Conjunto de Métricas resultante poderia ser representado como na Fig. 2 (b).

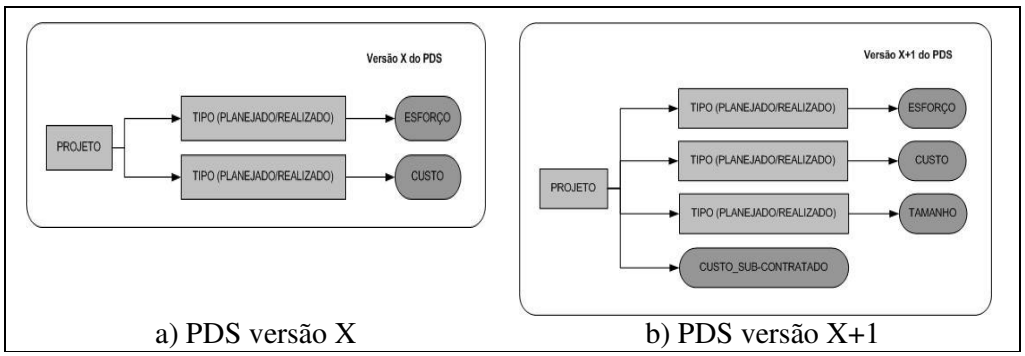


**Fig. 2 – Conjunto de Métricas para o PDS Exemplo**

Para tratar a evolução do PDS e do Programa de Métricas, é preciso analisar o Conjunto de Métricas das versões anterior e atual do PDS, identificando as alterações do tipo:

- *Criação de métrica: P*ara o caso da criação de uma métrica, são propostas quatro soluções: (1) capturá-la, caso seja possível, nas ferramentas de apoio, (2) derivar de outras métricas, (3) adotar valores aproximados ou (4) indicar como "Valor não disponível", caso não seja possível a adoção de nenhum valor.

- *Exclusão de métrica:* Para uma métrica excluída, propõe-se três soluções: (1) continuar coletando, mas sem a necessidade da análise desta métrica, (2), a atribuição de algum valor aproximado, por exemplo, a média dos valores coletados anteriormente ou (3) a indicação de "Valor não disponível".

- *Modificação de métrica existente*:

    - *Inclusão de uma dimensão:* No caso de modificação em uma métrica existente, em que ocorre a inclusão de uma dimensão, são propostas duas soluções: (1) coletar essa informação, diretamente nas ferramentas de apoio, (2) utilizar heurísticas para distribuir os valores dessa métrica entre os registros da dimensão criada.

- *Exclusão de uma dimensão:* Outra possibilidade, quanto à modificação de uma métrica existente, é a exclusão de uma dimensão. Neste caso, são propostas 2 soluções: (1) continuar a coleta dos valores relacionando-os a esta dimensão, mas não levar em consideração esta dimensão no momento da visualização, ou (2) excluir a dimensão e agrupar os valores.

A configuração dos metadados de versão de PDS, baseados na linguagem XML, referentes ao Programa de Métricas para o PDS exemplo na versão X, é exemplificada pela Fig. 3. Ainda, para demonstrar a captura de métricas para dois projetos concebidos na versão X do PDS, são descritos os metadados de projeto, na Fig. 4.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Versao_Captura" value="X">
        <xs:element name="Versao_Projeto" value="X">
                <xs:element name="Esforco_Planejado">
                        <xs:element name="Capturar" value="Esforco_Planejado"/>
                </xs:element>
                <xs:element name="Esforco_Realizado">
                        <xs:element name="Capturar" value="Esforco_Realizado"/>
                </xs:element>
                <xs:element name="Custo_Planejado">
                        <xs:element name="Capturar" value ="Custo_Planejado"/>
                </xs:element>
                <xs:element name="Custo_Realizado">
                        <xs:element name="Capturar" value ="Custo_Realizado"/>
                </xs:element>
        </xs:element>
</xs:element>
</xs:schema>
```

**Fig. 3 – Metadados para a Versão X do PDS exemplo**

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Projeto1">
        <xs:element name="Versao_Projeto" value="X"/>
        <xs:element name="Base1">
                <xs:element name="Esforco_Planejado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Esforco_Planejado" />
                </xs:element>
                <xs:element name="Esforco_Realizado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Esforco_Realizado" />
                </xs:element>
                <xs:element name="Custo_Planejado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Custo_Planejado" />
                </xs:element>
                <xs:element name="Custo_Realizado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Custo_Realizado" />
                </xs:element>
        </xs:element>
</xs:element>
<xs:element name="Projeto2">
        <xs:element name="Versao_Projeto" value="X"/>
        …
        </xs:element>
</xs:element>
</xs:schema>
```

**Fig. 4 – Metadados para os projetos na versão X do PDS exemplo**

Após a evolução do PDS exemplo para a versão X+1, devido às alterações no programa de métricas, surgem também alterações nos metadados. Para os projetos que estão na versão X+1, apenas são adicionadas as opções de captura, de acordo com o programa de métricas desta versão, enquanto que os projetos da versão anterior do PDS têm seus metadados adaptados para a versão X+1. Para a métrica de tamanho planejado, supondo-se que essa informação fosse armazenada nas ferramentas de apoio, mesmo antes da evolução do PDS, apenas adicionam-se as linhas referentes à captura desta métrica. Quanto ao tamanho realizado, supondo-se que seu armazenamento não era feito na versão anterior, e que não há possibilidades de derivar este valor de outras métricas, atribui-se a esta métrica a expressão "Valor não disponível". O custo sub-contratado, pode ter um valor aproximado, como por exemplo, 25 % do valor total do custo realizado. Estes metadados, de versão de PDS e de projeto, baseados versão X+1 do PDS exemplo, podem ser vistos nas figuras Fig. 5 e Fig. 6.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Versao_Captura" value="X+1">
        <xs:element name="Versao_Projeto" value="X">
                <xs:element name="Esforco_Planejado">
                        <xs:element name="Capturar" value="Esforco_Planejado"/>
                </xs:element>
                <xs:element name="Esforco_Realizado">
                        <xs:element name="Capturar" value ="Esforco_Realizado"/>
                </xs:element>
                <xs:element name="Custo_Planejado">
                        <xs:element name="Capturar" value ="Custo_Planejado"/>
                </xs:element>
                <xs:element name="Custo_Realizado">
                        <xs:element name="Capturar" value ="Custo_Realizado"/>
                </xs:element>
                <xs:element name="Tamanho_Planejado">
                        <xs:element name="Capturar" value ="Tamanho_Planejado"/>
                </xs:element>
                <xs:element name="Tamanho_Realizado">
                        <xs:element name="Atribuir" value ="Valor Não Disponível">
                </xs:element>
                <xs:element name="Custo_Sub-Contratado">
                        <xs:element name="Atribuir" value ="Custo_Realizado*0.25">
                </xs:element>
        </xs:element>
        <xs:element name="Versao_Projeto" value="X+1">
                <xs:element name="Esforco_Planejado">
                        <xs:element name="Capturar" value="Esforco_Planejado"/>
                </xs:element>
                <xs:element name="Esforco_Realizado">
                        <xs:element name="Capturar" value ="Esforco_Realizado"/>
                </xs:element>
                <xs:element name="Custo_Planejado">
                        <xs:element name="Capturar" value ="Custo_Planejado"/>
                </xs:element>
                <xs:element name="Custo_Realizado">
                        <xs:element name="Capturar" value ="Custo_Realizado"/>
                </xs:element>
                <xs:element name="Tamanho_Planejado">
                        <xs:element name="Capturar" value ="Tamanho_Planejado"/>
                </xs:element>
                <xs:element name="Tamanho_Realizado">
                        <xs:element name="Capturar" value ="Tamanho_Realizado">
                </xs:element>
                <xs:element name="Custo_Sub-Contratado">
                        <xs:element name="Capturar" value ="Custo_Sub-Contratado">
                </xs:element>
        </xs:element>
</xs:element>
</xs:schema>
```

**Fig. 5 – Metadados para a Versão X+1 do PDS exemplo**

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Projeto1">
        <xs:element name="Versao_Projeto" value="X"/>
        <xs:element name="Base1">
                <xs:element name="Esforco_Planejado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Esforco_Planejado" />
                </xs:element>
                <xs:element name="Esforco_Realizado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Esforco_Realizado" />
                </xs:element>
                <xs:element name="Custo_Planejado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Custo_Planejado" />
                </xs:element>
                <xs:element name="Custo_Realizado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Custo_Realizado" />
                </xs:element>
        </xs:element>
        <xs:element name="Base2">
                <xs:element name="Tamanho_Planejado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Tamanho_Planejado" />
                </xs:element>
        </xs:element>
</xs:element>
<xs:element name="Projeto2">
        …
</xs:element>
<xs:element name="Projeto3">
        <xs:element name="Versao_Projeto" value="X+1"/>
        <xs:element name="Base1">
                <xs:element name="Esforco_Planejado">…</xs:element>
                <xs:element name="Esforco_Realizado">...</xs:element>
                <xs:element name="Custo_Planejado">...</xs:element>
                <xs:element name="Custo_Realizado">...</xs:element>
                <xs:element name="Custo_Sub-Contratado">
                        <xs:element name="Tabela" value="Tabela1" />
                        <xs:element name="Campo" value="Custo_Sub-Contratado" />
                </xs:element>
        </xs:element>
        <xs:element name="Base2">
                <xs:element name="Tamanho_Planejado">
                        <xs:element name="Tabela" value="Tabela2" />
                        <xs:element name="Campo" value="Tamanho_Planejado" />
                </xs:element>
                <xs:element name="Tamanho_Realizado">
                        <xs:element name="Tabela" value="Tabela2" />
                        <xs:element name="Campo" value="Tamanho_Realizado" />
                </xs:element>
        </xs:element>
</xs:element>
</xs:schema>
```

**Fig. 6 – Metadados para os projetos da versão X+1 do PDS exemplo**

### 5.2. Camada de Integração de Dados

No trabalho de Becker *et al* (2006), a camada de integração é responsável por carregar os Dados recebidos da Camada de Integração de Aplicações no DSA, por meio de rotinas de limpeza e transformação, auxiliadas pelos metadados de versão de PDS. Após todo esse processo, os dados consolidados são carregados no DW. Para este trabalho, a Camada de Integração de Dados sofre alterações para comportar as rotinas de migração de dados entre as versões. A cada evolução, o DSA passa a englobar a extensão de dados contida no DW da versão anterior. O modelo de dados do DW é alterado sempre

que ocorre modificação no conjunto de métricas. As métricas que não sofrem alterações, simplesmente são carregadas no novo modelo à partir do modelo anterior, e as métricas que foram modificadas, são capturadas novamente das ferramentas de apoio ou adaptadas.

A evolução do DW foi tratada como uma evolução de esquemas, pela necessidade de manter uma visão unificada das métricas. Mantém-se apenas a última versão do modelo analítico, e a extensão do modelo anterior é migrada para o novo modelo. O DSA recebe, da Camada de Integração de Aplicações, os dados disponíveis nas ferramentas de apoio, já no formato do novo Programa de Métricas. Para os dados que não estão disponíveis nas ferramentas de apoio, são buscadas alternativas, tais como a derivação de outras métricas já capturadas, existentes na extensão de dados da versão anterior, ou heurísticas para adoção de valores aproximados, quando possível.

Tomando-se como exemplo o PDS na versão X, citado anteriormente, desenvolve-se o modelo analítico, para comportar estas métricas, como apresentado na Fig 7. a), onde utiliza-se uma dimensão tipo, para diferenciar o tipo de esforço e custo (Planejado ou Realizado), cria-se uma dimensão Projeto, para diferenciar as métricas de cada projeto e cria-se também uma dimensão de data, para incluir a data de coleta de cada métrica.

Com a evolução do PDS utilizado como exemplo, desenvolve-se um novo modelo analítico, com base nas métricas da versão X+1 do PDS, tal como demonstrado na Fig. 7 b). As métricas de Custo, Tamanho e Esforço, por possuírem as mesmas dimensões, são armazenadas na mesma tabela fato, enquanto que a métrica de custo sub-contratado, por não possuir a dimensão "Tipo", é acondicionada em uma outra tabela fato.

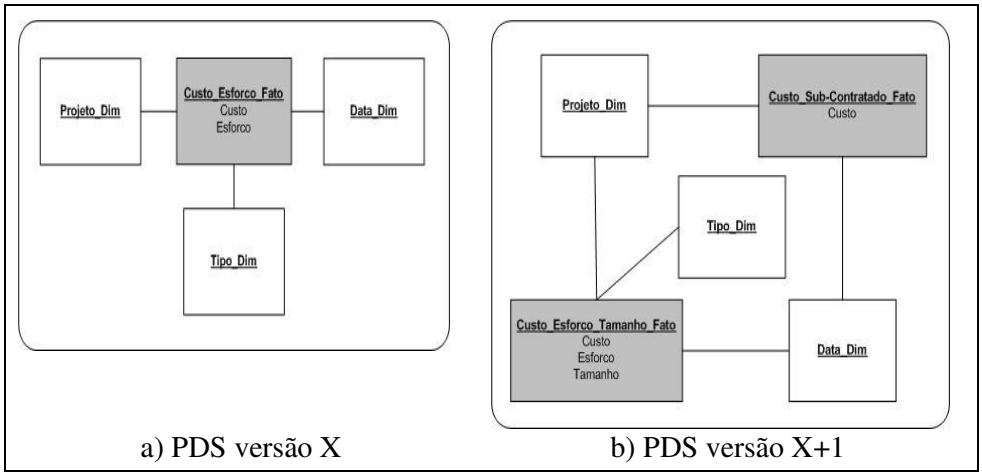


**Fig. 7 – Modelo analítico para o PDS Exemplo**

### 5.3. Camada de Apresentação

A camada de apresentação não sofre alterações porque a solução visa encapsular as peculiaridades provenientes da evolução de métricas, mantendo os dados do DW na última versão do processo, fazendo com que esta camada receba os dados do DW, e ignorando os procedimentos que foram executados para tratar a evolução.

## 6. Experimentação da Solução

A solução foi testada em uma empresa de Tecnologia de Informação, com certificação CMMI nível 2, que vem aperfeiçoando continuamente seu processo, para melhorar a qualidade de seus produtos e buscar o avanço nos níveis de certificação do CMMI. Porém, a realidade encontrada é simplificada pela homogeneidade dos projetos e aplicações utilizadas. Desta forma, não é necessária a utilização de *wrappers*, como sugerido na arquitetura, pois a empresa segue um padrão no uso das ferramentas, armazenando suas informações num mesmo SGBD (MS SQL Server). Assim, todo o processo de extração, transformação e carga foi suprido pelo desenvolvimento de procedimentos em T-SQL.

A versão atual do PDS utilizado pela empresa é a 5.0. Contudo, ainda estão ativas as versões anteriores, 4.2 e 4.1. Os dados utilizados para testes estão relacionados a defeitos, requisitos, esforço e tamanho do projeto, onde puderam ser tratadas algumas características da evolução.

Na versão 4.2 do PDS foi criada a métrica "Tamanho de Cada Requisito", que não existia na versão 4.1. A alternativa para tratar esta criação de métrica foi derivar de outras métricas, onde chegou-se a um valor aproximado, com a divisão do tamanho total do projeto pelo número de requisitos. Ainda na evolução da versão 4.1 para 4.2, houve a alteração das métricas de esforço, com a criação de uma nova dimensão. Desta forma, utilizou-se a adoção de valores aproximados, destinando-se 40% das horas de esforço para o esforço de Gestão e 60% para esforço de engenharia. Ao lançamento da versão 5.0 do PDS, ocorreu a inserção de uma métrica para controle de defeitos, levando em consideração o ambiente onde o defeito foi encontrado (Desenvolvimento/Produção).

Os metadados relativos à versão de PDS e aos projetos, com a coleta baseada nas versões 4.1, 4.2 e 5.0 do PDS foram desenvolvidos para configurar a captura das métricas acompanhando a evolução do PDS e do Programa de Métricas correspondente. Após capturadas das ferramentas de apoio, as métricas foram acondicionadas no DW. Para isto, desenvolveu-se o modelo analítico para cada versão de PDS. As métricas que não sofreram modificações foram migradas diretamente para os novos modelos, enquanto que as métricas que foram criadas ou modificadas foram tratadas e organizadas corretamente no novo modelo de dados.

## 7. Conclusões

Este trabalho propôs uma solução para tratar o processo completo de extração, organização e apresentação das métricas de PDS, levando em consideração a evolução do próprio PDS e do conjunto de métricas correspondente. A solução foi testada de uma forma simplificada em uma empresa de Tecnologia de Informação, com certificação CMMI nível 2, que vem aperfeiçoando continuamente seu processo, para melhorar a qualidade de seus produtos e buscar o avanço nos níveis de certificação do CMMI. Os aspectos relacionados a arquitetura orientada a serviços não foram testados pois, nesta empresa, há homogeneidade nos projetos e nas aplicações utilizadas. Assim, todo o processo de extração, transformação e carga foi suprido pelo desenvolvimento de procedimentos em T-SQL. Mesmo assim, o experimento permitiu explorar um modelo de PDS em evolução e as atualizações nos respectivos Programas de Métricas. Como resultado, o experimento demonstrou que é possível formar uma base sólida, com

informações de projetos em diferentes versões de PDS e que estas informações podem ser comparáveis entre si, em grande parte dos casos.

Como trabalhos futuros, sugere-se estender a presente proposta nos seguintes aspectos. (1) Aplicar técnicas de Descoberta de Conhecimento sobre o Repositório de Métricas da Organização, propiciando um maior aproveitamento das informações nele armazenadas. (2) Explorar, na Camada de Apresentação, formas de visualizar e tratar métricas com diferentes níveis de confiabilidade, ou seja, diferenciar métricas efetivamente obtidas da execução dos projetos, daquelas produzidas por heurísticas.

## Agradecimentos

Este trabalho está inserido no projeto "ALTAR – Aceleração do Levantamento, Desenvolvimento e Testes de Projetos de Software", financiado pela parceria entre a empresa TLANTIC Sistemas de Informação e a Pontifícia Universidade Católica do Rio Grande do Sul.

## Referências

BECKER, K.; RUIZ, D. D. A.; CUNHA, V. S. da; NOVELLO, T. C. ; SOUZA, F. V. SPDW: a Software Development Process Performance Data Warehousing Environment. In: 30th Annual IEEE/NASA Software Engineering Workshop (SEW-06), Columbia, MD, EUA, 2006. p. 1-10

GALANTE, R. M.; ROMA, A. B. S.; JANTSCH, A.; EDELWEISS, N. e SANTOS, C. S. Dynamic Schema Evolution Management using Version in Temporal Object-Oriented Databases. In: DEXA - 13th International Conference on Database and Expert Systems Applications, Aix en Provence, France, September 2002. p 67-175.

GOPAL, A.; MUKHOPADHYAY, T.; KRISHNAN, M.S. The impact of institutional forces on software metrics programs. In: Software Engineering, IEEE Transactions on Volume 31, Issue 8, Aug. 2005. p. 679 - 694.

JENSEN, C. S.; DYRESON, C.E. Temporal Databases: Research and Practice. Heidelberg : Springer Verlag, 1998. p. 367- 405.

KIMBALL, R.; REEVES, L; ROSS, M.; THORNTHWAITE, W. The Data Warehouse Lifecycle Toolkit: expert methods for designing, developing, and deploying data warehouses. New York: Johs Wiley e Sons, 1998. 800 p.

LIST, B. and MACHACZEK, K. Towards a Corporate Performance Measurement System. In: 2004 ACM symposium on Applied computing. Nicosia, Cyprus, 2004, p. 1344-1350.

NIAZI, M.; WILSON, D.; ZOWGHI, D. A framework for assisting the design of effective software process improvement implementation strategies. Journal of Systems and Software, Volume 78, Issue 2, November 2005, Pages 204-222

PALZA, E.; FUHRMAN, C.; ABRAN A. Establishing a Generic and Multidimensional Measurement Repository in CMMI context. In: 28th Annual NASA Soft. Eng. Workshop (SEW'03), 2003. p. 12-20.