4
Trabalhos relacionados

O intuito deste capítulo é analisar trabalhos cujos objetivos estejam relacionados ao problema principal do sistema Clairvoyant: providenciar uma capacidade de evolução transparente do modelo de medição em repositórios de medição de software. Isso é feito para situar o leitor deste trabalho quanto a suas peculiaridades em relação ao que já existe na literatura e também para apontar possíveis vantagens e desvantagens comparativas.

O primeiro trabalho a ser analisado nesse capítulo é aquele que inspirou este trabalho e que consideramos o mais próximo do sistema Clairvoyant em propósito: (Harrison, 2004), que apresenta um modelo evolutivo e genérico de medição denominado Repositório Universal de Métricas. Ele tem a peculiaridade de basear-se num modelo *transformacional* de desenvolvimento de software.

O próximo trabalho a ser analisado foi (Palza et al., 2003), que também apresenta um modelo evolutivo e genérico de medição de software denominado, por sua vez, Modelo Multidimensional de Medição. Sua peculiaridade é armazenar as medições segundo um esquema multidimensional típico de *data warehouses* (armazéns de dados) facilitando a sua consulta por ferramentas OLAP (*on-line analytical processing*).

Foi analisado, em seguida, (Franca et al., 1999), um trabalho que apresenta um modelo de medição genérico voltado para a geração de programas de medição para pequenas organizações. Ele embute uma certa flexibilidade no modelo de medição ao originá-lo como uma instanciação de programa de medição. Essa abordagem também o torna evolutivo, mas com uma granularidade mais grossa que a do sistema Clairvoyant.

Prosseguimos à análise de trabalhos relacionados à evolução de esquema em bancos de dados. Começamos analisando o sistema Encore (Skarra e Zdonik, 1986), um trabalho pioneiro e freqüentemente referenciado na área de evolução de esquemas. Sua abordagem de evolução ainda não pode ser considerada transparente, mas já apresenta idéias que serão úteis para o sistema Clairvoyant.

Em seguida, analisamos o sistema TSE (*Transparent Schema Evolution*) (Ra e Rundensteiner, 1997), apresentado num trabalho que foca na evolutibilidade transparente num contexto mais amplo que o de modelos de medição de software: o domínio de bancos de dados orientados a objeto. Assim sendo, os problemas que ele visa a resolver constituem um conjunto de onde o mecanismo de evolução transparente do sistema Clairvoyant extrai um pequeno subconjunto para solucionar.

Por fim, analisamos a metodologia PISE (*Program Independency Schema Evolution*) (Ra, 2004) que apresenta uma abordagem simples porém muito útil para a evolução de esquema em bancos de dados relacionais baseada em visões.

4.1. Repositórios de medições

4.1.1. Modelo de medição universal

(Harrison, 2004) nos apresenta um meta-modelo simples de medição respondendo, principalmente, aos anseios do desenvolvimento iterativo. O autor denomina o seu modelo de modelo universal justamente por não restringir o tipo de métrica representada no sistema.

O trabalho inicia explicitando a visão de seu autor sobre o processo de coleta de dados de medição de software. Ele considera que o conjunto de medições coletadas num ambiente de engenharia de software muda com o passar do tempo. Isso ocorreria em virtude da iteratividade do processo de coleta de dados de medição: quanto mais se aprende, mais se descobre quais outros dados são necessários e como melhor coletá-los. É importante notar que essa visão apresentada está em plena sintonia com os princípios que levaram ao desenvolvimento do sistema Clairvoyant.

Em seguida, essa visão é contrastada com as limitações de repositórios de medição pesquisados pelo autor. Como primeira limitação, ele cita a obsolescência: se uma métrica cair em desuso, isso acarretaria uma mudança de esquema complicada e custosa no banco de dados. Em seguida ele discorre sobre a percepção de ambigüidade do significado de medições – isso ocorreria pelo fato das descrições desses significados estarem armazenadas, via de regra, em arquivos

avulsos e não no banco de dados. No mais, ele aponta problemas relacionados ao aumento do conjunto de medições coletadas, como um possível particionamento das informações, que viria como forma de evitar de mudar o esquema existente do repositório de medição. Então, ele critica o foco adotado pelos repositórios: eles seriam excessivamente centrados em produtos, e isso os torna inapropriados para representar informações sobre processos e eventos. O autor então apresenta uma solução para essas limitações: o repositório de métricas universal, com um modelo de medição calcado sobre duas características: uma visão transformacional do desenvolvimento de software e um esquema de banco de dados baseado em meta-dados.

A visão transformacional pode ser resumida como modelar o desenvolvimento de software como uma sucessão de transformações de artefatos, que, por sua vez, consomem recursos, têm características e estão ligadas a eventos. Já o meta-modelo de medições, que tem um protótipo de prova de conceito implementado num banco de dados relacional, é constituído por entidades e relações (ver figura 5).

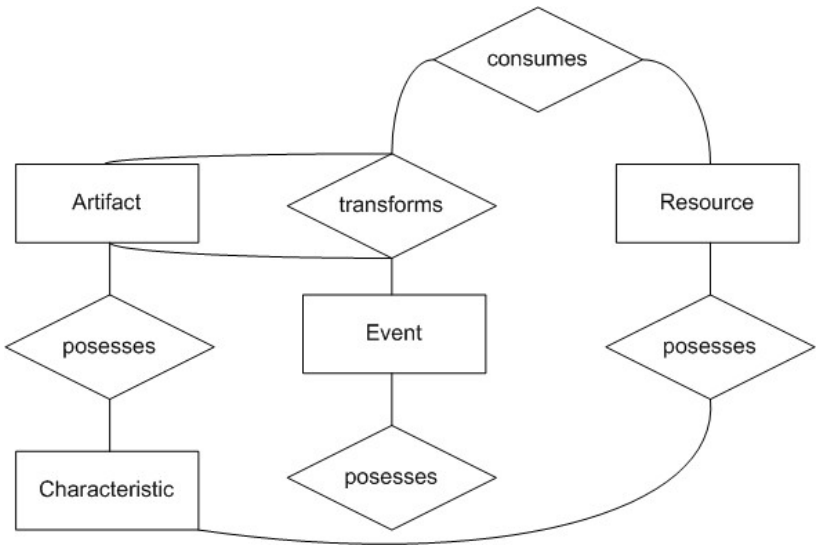

Figura 5 – representação do meta-modelo de medição implementado no Repositório Universal de Métricas (adaptado de Harrison, 2004)

As entidades do modelo são:

- *Artifact* (*id*, *name*)
- *Event* (*id*, *name*)
- *Resource* (*id*, *name*)
- *Characteristic* (*id*, *name*,*type*,*description*)

Já as suas relações são:

- *transforms* (*id*, *Artifact.id*, *Event.id*, *End-Date*)
- *possesses* ({*Artifact*, *Resource*, *Event*}*.id*, *Characteristic.id*, *qty*)
- *consumes* (*transforms.id*, *Resource.id*, *quantity*)

Analisando esse modelo, podemos traçar vários paralelos com o sistema Clairvoyant. Eles alcançam os mesmos objetivos de evolução e generalidade, e o fazem de maneira parecida por se tratarem de meta-modelos. Percebemos que a entidade Artifact corresponde aproximadamente ao conceito de Entidade do sistema Clairvoyant, assim como Characteristic corresponde aproximadamente ao conceito de atributo. No mais, no repositório universal não há um conjunto de características obrigatórias para um artefato, assim como no sistema Clairvoyant não há atributos obrigatórios para uma entidade. A diferença mais significativa entre os modelos é que o sistema Clairvoyant não adota a idéia da visão transformacional de desenvolvimento de software.

Em seguida, o autor trata da execução de consultas em seu repositório, e as diferenças entre os dois trabalhos ficam claras. Ao contrário do sistema Clairvoyant, o repositório universal não é provido de um modelo de consulta. Logo o usuário final tem que executar as consultas diretamente no banco de dados e fica exposto aos vários níveis de indireção do sistema. O autor considera que é um pequeno inconveniente diante do ganho na capacidade de evolução – ou seja, não se preocupa com a transparência na evolução do modelo de medição. Nesse ponto há uma divergência em relação ao sistema Clairvoyant, que considera essa exposição inaceitável, e que, por isso, fez de seu objetivo principal prover uma funcionalidade de evolução de modelo de medição transparente.

A vantagem mencionada possibilitada pela visão transformacional é o *suporte* a vários modelos de ciclo de vida de desenvolvimento de software. Essa afirmação é justificada por que o modelo transformacional tornaria mais fácil lidar com artefatos que aparecem e desaparecem antes da completação de um projeto, pois todos esses eventos estariam persistidos no repositório. Esse poderia ser o caso ao lidarmos com modelos de ciclo de vida com características incrementais ou iterativas.

Acreditamos que a visão transformacional é útil pelos motivos acima mencionados. Entretanto, ela pode ser implementada por outros meta-modelos de medição. Por exemplo, se o meta-modelo tiver elementos correspondente a

entidades e *relacionamentos*, basta instanciá-lo com um modelo com uma entidade representado o elemento *Event* e um relacionamento representando o elemento *transforms*.

4.1.2. Modelo de medição multidimensional

(Palza et al., 2003) apresenta um modelo de medição multidimensional. Os anseios de análise que motivaram a concepção e implementação de um modelo dimensional são claramente apresentados. Esse modelo dimensional baseia-se em um meta-modelo de medição, assim como o sistema Clairvoyant, e também é flexível e evolutivo.

Um modelo multidimensional de medições é um modelo de medições baseado no modelo dimensional de armazenamento usado em armazéns de dados (Inmon, 2005), onde certos dados são lançados como indicadores de um elemento chamado fato e outros são lançados como dimensões; os primeiros desempenhariam o papel de variável agregada e os últimos os fatores de agregação nas consultas. Essa dimensionalidade dos dados facilita a análise dos indicadores em diferentes níveis de granularidade. Em especial, são facilitadas consultas do tipo *roll-up* e *drill-down*, e outras típicas de aplicações OLAP (*On-Line Analytical Processing*) (Codd, 1993).

Estruturalmente, o repositório multidimensional está dividido em 4 níveis:

1. **Coleta**
    - **Manual:** via formulários web.
    - **Automática:** via uma ferramenta ETL (*Extraction, Transformation and Loading*).
2. **Repositório de medição:** provê o armazenamento dos dados de medição segundo o modelo da figura 6.
3. **Processador analítico (OLAP):** provê a capacidade de computar medições derivadas e agregá-las por múltiplas dimensões.
4. **Interface com o usuário**
    - **Indicadores e tendências:** relatórios com dados nos níveis mais altos de agregação.

- **Funcionalidades analíticas, *roll-up* e *drill-down*:** relatórios com dados num nível intermediário de agregação.
- **Administrativo e controle da qualidade:** definição de medições, controle dos acessos e auditoria dos dados.

Podemos notar algumas semelhanças entre o meta-modelo desse trabalho (figura 6) e o meta-modelo de medição do sistema Clairvoyant, como a presença dos conceitos de entidade, relacionamento e atributo. É importante notar que o trabalho não menciona como é feito o mapeamento desse modelo conceitual orientado a objeto em possíveis modelos dimensionais.

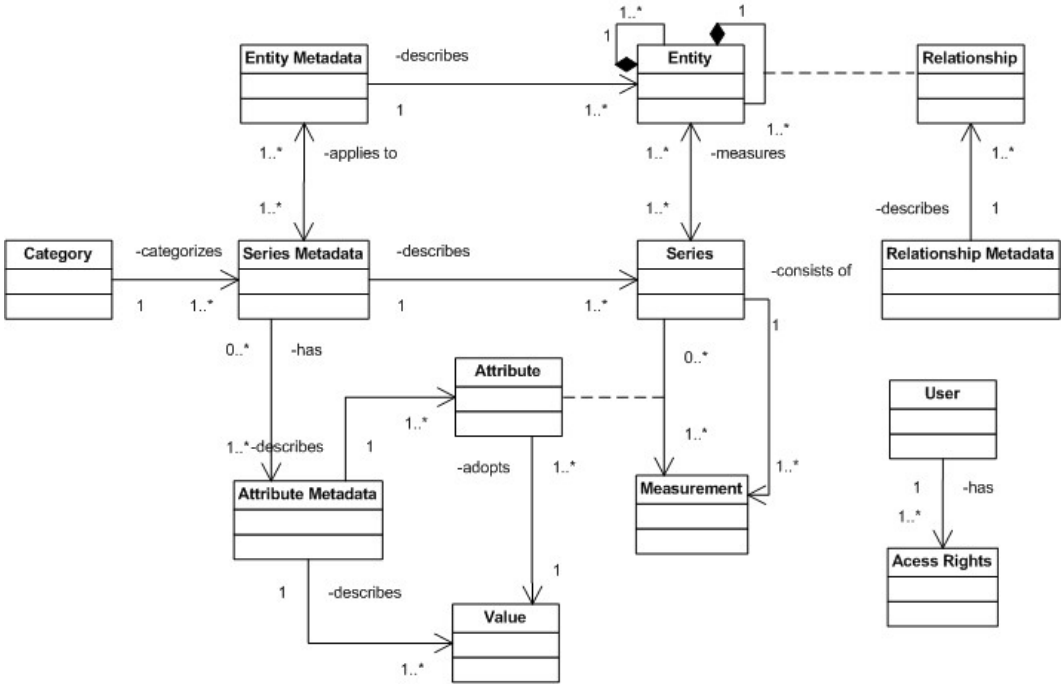

Figura 6: representação do meta-modelo do repositório do Modelo Multidimensional de Medição (adaptado de Palza et al. , 2003)

No decorrer desse trabalho, é feita alusão à capacidade evolutiva desse repositório, mas os autores não explicam como essa evolução ocorre. Eles somente mencionam que ela é possibilitada pelo meta-modelo. A possível transparência na evolução não é um tema abordado, mas, no decorrer do trabalho, os autores mencionam que a definição de novas medições é atribuição do administrador. Sendo assim, somos levados a crer que isso envolveria a interação direta com o esquema do banco de dados subjacente, violando os preceitos da evolução transparente.

Para tornar mais clara a utilidade de um modelo dimensional, recomendamos a análise de exemplos de possíveis modelos dimensionais para medição de software, como (Ruiz et al., 2005). Entretanto, esse modelo de medição apresentado não é flexível o suficiente para se adaptar a mudanças nas métricas coletadas, logo não é evolutivo.

Como observação final, notemos que a aplicabilidade desse trabalho é bem mais restrita que a do sistema Clairvoyant, pois ele depende do armazenamento baseado em armazéns de dados (Inmon, 2005). Estes costumam ser sistemas proprietários caros, e esse custo é agravado pela demanda associada por hardware de alto desempenho.

4.1.3. Gerador de programas de métricas para pequenas empresas

(Franca et al., 1999) apresenta um modelo de classes de um gerador de programas de métricas voltado para pequenas empresas. Esse trabalho aponta a necessidade de medições diferentes em diferentes organizações, apontando um problema freqüente, que é a adaptação das iniciativas de medição em organizações às ferramentas utilizadas ao invés de às suas necessidades de informação. A solução apresentada é um meta-ambiente de medição de software denominado *COMPASSO*, que instancia programas adaptados às necessidades de medição de cada organização ou até a projetos específicos de uma organização.

Funcionalmente, o sistema COMPASSO é dividido em três módulos (figura 7): personalizador, medidor e analisador. O personalizador executa as funções de definição e personalização dos programas de medição. Já o medidor é responsável pela coleta dos dados (variando com as medições selecionadas pelo módulo de personalização), tanto por meio de preenchimento de formulários como de processamento de arquivos texto. Por fim, o apresentador seleciona e agrega os dados para mostrá-los em vários formatos. No mais, os dados dos diferentes programas de medição podem ser integrados por estarem armazenados num banco de dados comum segundo o mesmo meta-modelo.

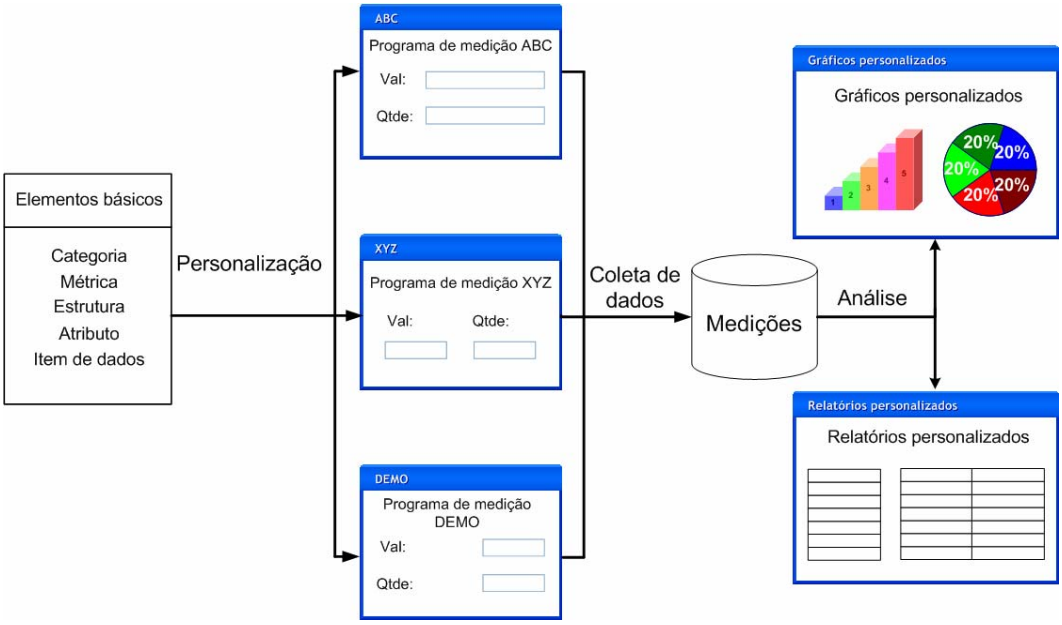

Figura 7: Os três módulos do sistema COMPASSO – personalizador, medidor e analisador (adaptado de Franca et al., 1999)

Esses módulos se baseiam num modelo de classes com quatro grandes grupos:

- **Elementos básicos:** apresenta um meta-modelo de medição, cujas instâncias são os elementos que podem participar de um programa de medição. Podemos citar como exemplos de elementos Categoria, Métrica, Estrutura, Item de Dados, entre outros.
- **Programa de medição:** apresenta classes que servem para indicar quais elementos básicos participam de cada programa de medição instanciado.
- **Medição:** apresenta classes usadas para o armazenamento de dados de medição. Suas classes correspondem basicamente a instâncias dos elementos básicos.
- **Elementos administrativos:** apresenta classes utilizadas para armazenar informações administrativas, como qual pessoa está alocada a qual projeto e por qual métrica ela é responsável, por exemplo.

É interessante notar que o meta-modelo básico utilizada pelo sistema COMPASSO para representar os dados de medição tem pontos em comum com o sistema Clairvoyant. Seus elementos básicos são: estrutura, que é análogo a

entidade no sistema Clairvoyant, atributo, que corresponderia aos nossos atributos categóricos, e item de dados, correspondente aos nossos atributos numéricos. Esse modelo do sistema COMPASSO baseia-se no modelo PSM (McGarry et al., 2003).

Apesar da evolutibilidade não ter sido explicitamente citada como preocupação neste trabalho, podemos dizer que o sistema COMPASSO é, de uma certa maneira, evolutivo. Afinal, ele permite acomodar, por meio de seu mecanismo de instanciação, uma mudança no modelo de medição em uso numa instituição. Apesar disso, a evolução não é transparente, pois não há como fazê-lo sem ser exposto ao meta-modelo (devido ao processo de instanciação do meta-ambiente). Caso a evolução seja freqüente, esse processo será demasiadamente custoso.

4.2. Evolução de esquema em bancos de dados

4.2.1. O sistema Encore

O sistema gestor de bancos de dados Encore (Skarra e Zdonik, 1986) é freqüentemente referenciado em outros trabalhos de evolução de esquema de bancos de dados, sendo citado como um dos trabalhos pioneiros desse campo. Ele apresenta um mecanismo para a evolução de esquema em bancos de dados orientados a objetos que preserva a interoperabilidade entre diferentes versões dos tipos de dados sujeitos à evolução.

Sua abordagem para a evolução de esquema passa pelo uso de uma interface para os conjuntos de versões (*version set interface*) para garantir uma interface que acomode todas as versões dos tipos e tratadores de erro para resolver incompatibilidades entre versões. A grosso modo, podemos dizer que são mecanismos para preservar a interoperabilidade sintática e semântica, respectivamente.

A interface para o conjunto de versões – uma das inovações do sistema Encore – nada mais é que a união de todas as propriedades e operações em algum momento especificadas para um dado tipo. Ela estabelece uma interface padrão para todas as instâncias de um tipo, independentemente de sua versão. Desta

maneira, o sistema Encore elimina a possibilidade de erro de acesso ao tentarmos acessar uma operação ou propriedade que deixou de ser referenciada por uma nova versão de um tipo.

Entretanto, a possibilidade de uma instância de um tipo acessar uma propriedade ou operação não garante que ele estará definido. Se houver uma tentativa de acesso a uma propriedade ou operação não definida para uma instância, será acionado um tratador de erro – outra característica marcante do sistema Encore. Além disso, tratadores de erros podem também ser acionados em tentativas de escrita ou leitura de valores de propriedades incompatíveis com o domínio da mesma no momento da operação.

O sistema Encore é interessante por evidenciar claramente os principais problemas associados à interoperabilidade na evolução de esquemas de bancos de dados. Entretanto, a elevada complicação conceitual de conseguir criar um tratador de erros semanticamente relevante diminui a usabilidade do sistema. No mais, o trabalho adicional que temos com os tratadores de erro significa que não podemos, pelos nossos critérios, considerar o mecanismo de evolução do sistema Encore como transparente.

4.2.2. Evolução transparente de esquema com visões orientadas a objeto

Dada a necessidade de evolução transparente do modelo de medição no sistema Clairvoyant, era natural que se pesquisasse evolução transparente em bancos de dados em geral. Uma solução que consideramos interessante foi a apresentada por (Ra e Rundensteiner, 1997).

Esse trabalho apresenta um mecanismo de versionamento de esquemas para bancos de dados orientados a objetos, chamado de sistema *transparent schema-evolution* (TSE). O sistema TSE é baseado em visões de bancos de dados, de tal maneira que a todo aplicativo corresponde uma diferente visão do esquema físico subjacente representando um “esquema virtual” do banco de dados visível para esse aplicativo. Quando uma mudança no esquema do banco é requisitada, acontecem os seguintes fatos:

- o esquema físico do banco é alterado;

- a visão correspondente ao aplicativo solicitante é recalculada para que seu esquema virtual correspondente seja afetado pela alteração solicitada;
- as demais visões são recalculadas para manterem seus esquemas virtuais correspondentes inalterados.

Uma importante característica desse mecanismo é que as instâncias de objetos são compartilhadas por várias visões (*compartilhamento de instância*). Assim, não há necessidade de replicar dados para efetuar o versionamento, ao contrário do que é feito na maioria de sistemas de versionamento de esquema, que é o versionamento por cópias de instâncias de objeto. Com isso, o sistema TSE ganha na facilidade de manter a integridade dos dados e diminui o espaço de armazenamento para os dados.

O sistema TSE implementa essa funcionalidade por meio de um paradigma chamado *object-slicing* (fatiamento de objeto). Resumidamente, cada vez que a representação de um objeto mudar, os campos que forem adicionados constituirão uma nova classe no modelo subjacente, e as instâncias dessa nova versão serão representadas por visões que apontam para as partes antiga e nova do objeto, fazendo a junção dos dados de maneira transparente ao usuário.

Podemos dizer que a abordagem de evolução do sistema Clairvoyant tem algumas semelhanças com o *object-slicing*, pois os dados de cada atributo são armazenados em tabelas diferentes, e a visão referente a entidade faz a junção dos dados da mesma de maneira transparente ao usuário. Entretanto, há uma diferença fundamental: graças ao meta-modelo de medição, o sistema Clairvoyant não precisa mudar o esquema físico subjacente de seu banco de dados.

No mais, é importante notar uma semelhança entre os dois sistemas, que é a manutenção dos dados de esquemas anteriores. Desta maneira, é possível fazer análises histórica e auditoria dos dados.

Podemos notar que o problema abordado pelo mecanismo de evolução de esquema do sistema TSE é bem mais geral que o problema específico de evolução do modelo de medição do sistema Clairvoyant. Além disso, a aplicabilidade dos dois mecanismos é diferente: o sistema TSE atua sobre bancos de dados orientados a objeto, enquanto o sistema Clairvoyant atualmente se baseia em bancos de dado relacionais. Assim sendo, o sistema TSE aborda vários problemas com os quais o sistema Clairvoyant não tem que se preocupar, como, por

exemplo, a propagação da evolução em uma hierarquia de classes ou a evolução em sub-esquemas.

4.2.3. A metodologia PISE

A metodologia PISE (*Program Independency Schema Evolution*) (Ra, 2004) foi concebida com o objetivo de solucionar o problema do impacto causado por evoluções no esquema de um banco de dados sobre aplicativos que dele dependem. Deve ser preservada a capacidade dos programas de consultar e atualizar os dados. Essa propriedade é denominada no trabalho como independência dos programas na evolução de esquema.

O mecanismo de evolução da metodologia PISE é baseado em visões, supondo que cada programa acessa o banco de dados através de uma visão de esquema a ele atribuída. São previstos dois tipos de evolução de esquema: com aumento de capacidade (e.g., adição de atributo) ou sem aumento de capacidade. No caso de evolução com aumento de capacidade, temos:

1. modificação do esquema para prover a capacidade adicional demandada pela operação;
2. reconstrução do esquema original como visão sobre o esquema modificado (1);
3. o esquema final pretendido é gerado como uma visão sobre o esquema modificado (1).

Já no caso específico de modificação de esquema sem aumento de capacidade, somente a operação 3 é necessária. Podemos assim perceber que a metodologia PISE funciona supondo que todos os esquemas utilizados por usuários humanos ou aplicativos são visões derivadas de um mesmo esquema subjacente.

Em termos de aplicabilidade, a principal diferença da metodologia PISE em relação aos outros trabalhos analisados é a sua aplicabilidade a SGDS relacionais. Apesar de sua abordagem ser aplicável a bancos de dados orientados a objeto, ela não seria suficiente para atingir seus objetivos propostos nesse contexto, pois ela não aborda operações de evolução específicas a bancos de dados orientadas a objeto. Como exemplo, podemos citar as operações relacionadas à hierarquia de

herança de uma classe. Vale a pena lembra que essas operações não são relevantes do ponto de vista do sistema Clairvoyant.

Dada a quantidade mais restrita de operações com as quais o sistema PISE tem que lidar em relação aos outros trabalhos, a sua abordagem é mais simples que as demais. Consideramos justamente essa simplicidade como seu principal ponto positivo. Como os programas acessam o banco de dados através de visões, é indiferente para eles se a mudança no esquema é real ou virtual. Assim sendo, podemos, pelos nossos critérios, considerar o mecanismo de evolução da metodologia PISE como transparente

4.3. Conclusão

Um resultado importante deste estudo de confronto com a literatura é o respaldo dado ao uso de meta-dados e de indireção como mecanismos para possibilitar a evolução do modelo de medição. Esse respaldo é dado pelos trabalhos analisados (Harrison, 2004), (Palza et al., 2003) e (Franca et al. 1999), que utilizam meta-modelos de medição para tornar os seus modelos de medição genéricos e evolutivos.

Foi dado respaldo também ao mecanismo de visões e versionamento que possibilita a evolução transparente no sistema Clairvoyant. Isso decorre do resultado da nossa análise dos trabalhos escolhidos sobre evolução transparente de esquema de banco de dados (Ra e Rundensteiner, 1997) (Ra, 2004). Percebemos que, abstraindo as diferenças nas soluções provenientes das diferenças entre os escopos dos respectivos problemas, que a filosofia das respectivas soluções aos problemas de evolução transparente é a mesma.

Pudemos também notar que o sistema Clairvoyant precisa de um melhor acabamento para uso prático em ambientes reais de engenharia de software. Uma evidência disso é a sua falta de informações administrativas, associando usuários do sistema a medições lançadas. Essa informação poderia trazer benefícios como a produção de um histórico de operações executadas por um indivíduo, para fins de auditoria, ou controle de acesso aos dados e de suas consultas relacionadas.