2
Visão e escopo

Para especificar e construir um repositório de medições de software dotado de evolução transparente, necessitamos explicitar o contexto no qual operamos. Precisamos conceituar medição de software, repositório de medição de software, evolução e evolução transparente num tal repositório.

Visamos com isso a explicar qual o ponto de vista adotado deste trabalho sobre o tema proposto e quais as questões que ele pretende abordar para resolver os problemas. Dito de outra maneira, pretendemos apresentar sua visão e delimitar o seu escopo. Além disso, temos por meta, neste capítulo apresentar uma visão panorâmica do tema para deixar bem claro o anti-escopo e situar a proposta num contexto mais amplo.

A abordagem adotada neste capítulo consiste em partir de considerações mais gerais em direção a temas mais específicos. Primeiramente, caracterizaremos medição e análise. Discorreremos, em seguida, sobre o armazenamento de informações coletadas de medição e análise, dando ênfase em estrutura e funcionalidades de sistemas de armazenamento de medições de software. Por fim, conceituamos evolução e evolução transparente em repositórios de medição, para delimitarmos quais definições desses conceitos adotaremos neste trabalho.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.1.
Medição e análise de software

Esta seção pretende apresentar o contexto no qual se situa este trabalho em relação ao tema proposto, referenciando modelos conhecidos de medição e análise para construir uma visão de medição de software e do processo de medição e análise de software.

2.1.1. Medição de software

No âmbito deste trabalho, adotamos a definição de medição apresentada em (Fenton e Pfleeger, 1997): uma medição é um mapeamento entre um mundo empírico e um mundo formal. Logo, uma medição é o número ou símbolo atribuído a uma entidade por esse mapeamento para caracterizar um atributo.

Por analogia, a medição de software é a atribuição de um valor a uma propriedade de medida de um sistema de software ou do seu processo de engenharia (desenvolvimento ou manutenção), incluindo propriedades do ambiente de engenharia.

No contexto da medição de software, podemos dividir as medições em medições-base e medições derivadas. Medição-base é aquela coletada diretamente (por exemplo, número de linhas de código de um módulo). Já a medição derivada é aquela que não é coletada diretamente, mas sim calculada, a partir de outras medições básicas ou derivadas (por exemplo, densidade de defeito definida como numero de defeitos sobre tamanho).

Outra maneira de caracterizar as medições de software é pelo tipo de escala (Fenton e Pfleeger, 1997). Isso influencia o modo como podemos representá-la e as operações às quais podemos submetê-las:

- **Nominal**: um atributo nominal é um atributo classificado em categorias. Temos como requisito para esse tipo de atributo a *exclusão mútua* e a *exaustividade conjunta*, ou seja, cada elemento só pode pertencer a uma categoria e o conjunto de categorias deve abranger todos os possíveis elementos. Se houver categorias existentes que não nos interessem, devemos criar uma categoria *outros*. Não há relação de ordem entre itens de uma categoria, logo não há sentido comparações do tipo “maior que” ou “menor que”. Tampouco há sentido em efetuar operações aritméticas sobre atributos nominais. As únicas operações que fazem sentido em atributos desse tipo de escala são operações de pertinência. Como exemplos de atributos de tipo de escala nominal, podemos citar a linguagem de programação de um módulo, ou o tipo de disciplina associada a uma atividade executada (requisitos, análise, teste, etc.).

PUC-Rio - Certificação Digital Nº 0510997/CA

- **Ordinal**: um atributo ordinal é uma classificação em categorias dotadas de uma relação de ordem. Para esse tipo de escala, fazem sentido comparações baseadas na ordem, que são transitivas. Além disso, as relações de pertinência mencionadas anteriormente. Não fazem sentido efetuar sobre elas operações aritméticas de nenhuma espécie. Isso ocorre porque não há informação de distância contida entre itens categóricos ordinais. Um erro comum na interpretação de dados ordinais é partir do princípio que a distância entre as categorias é constante. Outro erro comum é interpretá-lo como um atributo numérico quando o rótulo da categoria é numérico (por exemplo, tentar exprimir um nível médio de severidade de defeitos quando estes têm níveis rotulados como “1”, “2” e “3”). Como exemplo de exemplos de atributos ordinais, podemos citar o nível de maturidade de processo de uma organização ou a severidade de um defeito.
- **Intervalo**: um atributo de tipo de escala *intervalo* é um atributo numérico cujo valor zero não tenha um significado de ausência da propriedade. Costuma-se dizer que elas têm um “zero arbitrário”. Os exemplos mais comuns na física são datas e temperatura nas escalas Celsius ou Fahrenheit (notem que, por definição, a “temperatura zero” da escala Kelvin corresponde à ausência de agitação térmica, logo ela não é uma medição de tipo de escala intervalo por não ter um “zero arbitrário”). Por esse motivo, as operações de soma e subtração estão definidas para esse tipo de escala, mas não as de multiplicação e divisão. Como exemplo de medições de software nessa escala, podemos citar quaisquer medições envolvendo datas (prazos, datas de fim, de início, etc.).
- **Razão**: um atributo de tipo de escala *razão* é um atributo numérico cujo valor zero tenha um significado de ausência da propriedade, ao contrário dos atributos de tipo de escala *intervalo*. Por esse motivo, todas as operações aritméticas são permitidas neles. Como exemplo, podemos citar medições de número de linhas de código de módulos e de duração de tarefas. É importante notar que a operação de subtração entre atributos de tipo intervalo pode nos dar um atributo

PUC-Rio - Certificação Digital Nº 0510997/CA

do tipo razão (por exemplo, a diferença entre duas datas tendo como resultado uma duração).

Para melhor exemplificar possíveis medições de software, recorremos a duas compilações de medições. A primeira foi extraída de (Humphrey, 1995) e correspondente a um conjunto de medições básicas[1] recomendadas para uso individual num ambiente simples (tabela 1). Já a segunda compilação vem do modelo *Practical Software and Systems Measurement*, mais conhecido como PSM (McGarry et al., 2003). Esse documento almeja ser uma referência abrangente sobre diferentes tipos de medições em projetos de software e tem como meta ser aplicável ao maior número possível de ambientes (tabela 2).

2.1.2. Visão geral (ciclo) de medição e análise

Para construirmos um sistema de armazenamento de dados de medição de software temos que nos basear numa visão do ciclo de medição e análise, que deve ser definida e explícita. No nosso caso, escolhemos basear nossa visão em um modelo de referência: o CMMI (*Capability Maturity Model Integration*) (SEI, 2002). Ele foi escolhido por ser um modelo consagrado de melhoria de processo que evidencia as ligações da medição e análise com gerência de projeto e processo. Além disso, ele não prescreve a coleta de nenhuma medição específica (apesar de mencionar várias delas), nem as características do ambiente de engenharia de software onde deve ser aplicado. Assim sendo, ele é um modelo que se adapta a uma gama variada de cenários que podemos vir a encontrar.

PUC-Rio - Certificação Digital Nº 0510997/CA

[1] Coincidentemente, todas as medições da tabela correspondem ao tipo de escala *razão*

Medição	Definição
Tempo	O tempo total gasto no projeto até o momento
Contagem de defeitos	A contagem total acumulada de defeitos no projeto até o momento
Tamanho em LOC	Tamanho de um programa em linhas de código contadas, divididas entre: LOC base, LOC excluídas, LOC modificadas, LOC adicionadas, LOC reutilizadas, LOC total
LOC/Hora	Total de LOC novas e modificadas até o momento, divido pelas horas necessárias para desenvolvê-las; LOC/Hora (fase) = 60*(LOC novas e modificadas na fase) / (Tempo em fase atual, em minutos) LOC/Hora (fase) = 60*(LOC novas e modificadas totais) / (Tempo atual total, em minutos)
Índice de Desempenho do Custos (CPI)	CPI = tempo de desenvolvimento planejado total até o momento dividido pelo tempo total em fase até o momento
Valor realizado[2]	O valor planejado para uma tarefa é contabilizado quando ela é completada. Não há crédito parcial para tarefas parcialmente completadas.
Defeitos de Testes /KLOC	Defeitos de teste removidos por KLOC novas e modificadas: Defeitos/KLOC = 1000*(defeitos de teste)/N(realizado)
Defeitos Totais /KLOC	Defeitos totais removidos por KLOC novas e modificadas: Defeitos/KLOC = 1000*(defeitos totais)/N(realizado)
Rendimento	O percentual de defeitos totais removidos antes da primeira compilação (por procedimentos de inspeção e revisão)
Tempo de avaliação	Tempo total gasto na revisão do projeto e do código
% COQ de avaliação	O percentual de tempo total gasto em avaliação: % COQ de avaliação = 100 * (tempo de avaliação) / (tempo total de desenvolvimento)
Tempo de falha	O tempo total gasto em compilação e teste
% COQ de falha	O percentual de tempo total gasto em recuperação de falhas: % COQ de falha = 100 * (tempo de falha) / (tempo total de desenvolvimento)
COQ	Custo da qualidade: COQ = % COQ de avaliação + % COQ de falha

Tabela 1 – Medições recomendadas pelo PSP (Humphrey, 1995)

PUC-Rio - Certificação Digital Nº 0510997/CA

Mapeamento Questão – Categoria – Medição		
Questão	Categoria	Medição
Prazo e progresso	Desempenho de marcos	Datas de marcos Desempenho do caminho crítico
	Progresso de unidades de trabalho	Situação dos requisitos Situação dos relatórios de problemas Situação da revisão Situação das requisições de mudança Situação dos componentes Situação dos testes Situação dos itens de ação
	Capacidade incremental	Conteúdo do incremento – componentes Conteúdo do incremento – funcional
Recursos	Pessoal	Esforço Experiência do pessoal Rotatividade de pessoal
	Desempenho financeiro	Valor realizado[3] Custo
	Recursos de ambiente e apoio	Disponibilidade de recurso Utilização de recurso
Tamanho e estabilidade de produto	Tamanho e estabilidade físicos	Tamanho do banco de dados Componentes Interfaces Linhas de código (LOC) Dimensões físicas
	Tamanho e estabilidade funcionais	Requisitos Carga de mudança funcional Pontos de função
Qualidade de produto	Corretude funcional	Defeitos Desempenho Técnico

[2] Soma dos valores planejados das tarefas já completadas

[3] Produto do valor total orçado pelo projeto e a proporção entre o esforço planejado total das tarefas já realizadas e o esforço planejado total do projeto

PUC-Rio - Certificação Digital Nº 0510997/CA

Mapeamento Questão – Categoria – Medição		
Questão	Categoria	Medição
Qualidade de produto	Suporte e disponibilidade	Tempo de recuperação Complexidade ciclomática Ações de manutenção
	Eficiência	Utilização Taxa de transferência Tempo de resposta
	Portatilidade	Conformidade com padrões
	Usabilidade	Erros de operador
	Confiabilidade	Falhas Tolerância a falhas
Desempenho do processo	Conformidade com o processo	Grau no modelo referência Resultados da auditoria de processo
	Eficiência do processo	Produtividade Tempo de ciclo
	Efetividade do processo	Contenção de defeitos Volume de retrabalho
Efetividade da tecnologia	Adequação da tecnologia	Cobertura de requisitos
	Impacto	Impacto da tecnologia
	Volatilidade da tecnologia	Mudanças na linha de base
Satisfação do cliente	Apreciação do cliente	Resultado de pesquisa Grau de desempenho
	Apoio ao cliente	Pedidos de suporte Tempo de suporte

Tabela 2 – Exemplos de medições apresentadas pelo PSM (McGarry et al., 2003)

PUC-Rio - Certificação Digital Nº 0510997/CA

2.1.2.1. Capability Maturity Model Integration (CMMI)

O *Capability Maturity Model Integration* (CMMI) (SEI, 2002) é um modelo de referência de melhoria contínua de processos de desenvolvimento e manutenção de software. Ele surgiu no fim dos anos 90 como uma evolução do modelo CMM (*Capability Maturity Model*) (Paulk et al, 1993) que, por sua vez, serve ao mesmo propósito e foi encomendado pelo Departamento de Defesa norte-

americano ao Instituto de Engenharia de Software (SEI – *Software Engineering Institute*) da Universidade Carnegie Mellon no fim dos anos 80. A inspiração original do CMM (e, indiretamente, do CMMI) foi o conjunto de metodologias de gerência de desenvolvidas para processo industriais como, por exemplo, as metodologias de controle de qualidade (Juran et al., 1974) (Shewhart e Deming, 1986). Estas abordagens são partidárias de um alto grau de planejamento, do controle estatístico dos processos e da sua melhoria contínua.

O CMMI é composto por um conjunto de práticas e critérios de avaliação para o desenvolvimento e manutenção de software. As práticas são reconhecidas na indústria como exemplares, e estão agrupadas de maneira a expor suas relações. O CMMI está entre os modelos usados com mais freqüência para avaliar o processo de desenvolvimento de software em organizações. Isso ocorre atribuindo-se níveis de maturidade ao processo (ou partes dele), de acordo com a execução ou não destas práticas. São cinco níveis em ordem crescente:

1. **Inicial:** os processos são normalmente improvisados ou caóticos. A instituição não costuma prover um ambiente estável.
2. **Gerenciado:** os requisitos são gerenciados e os processos são planejados, executados, medidos e controlados.
3. **Definido:** os processos estão bem caracterizados e compreendidos, e estão descritos em padrões, procedimentos, ferramentas e métodos.
4. **Quantitativamente Gerenciado:** são selecionados sub-processos que contribuem significativamente à performance do processo como um todo. Estes sub-processos selecionados são controlados usando técnicas estatísticas e outras técnicas quantitativas.
5. **Otimizado:** os processos são melhorados continuamente baseados numa compreensão quantitativa das causas comuns de variação inerentes aos processos.

As práticas (e suas metas associadas) são agrupadas em *áreas de processo*. As áreas de processo, por sua vez, são agrupadas em quatro categorias:

- **Gerência de processo:** contém as atividades inter-projeto relacionadas à definição, planejamento, viabilização (fornecimento de recursos), implantação, implementação, monitoramento, controle, avaliação, medição e melhoria de processos;

PUC-Rio - Certificação Digital Nº 0510997/CA

- **Gerência de projeto:** cobre as atividades relacionadas ao planejamento, monitoramento, e controle do projeto;
- **Engenharia:** cobre as atividades de manutenção e desenvolvimento que são compartilhadas através das disciplinas de engenharia;
- **Apoio:** cobre processos que são usados no contexto da execução de outros processos.

No CMMI, temos as seguintes áreas de processo, agrupadas por categoria e nível de maturidade onde se tornam obrigatórias (tabela 3):

	Engenharia	Apoio	Projeto	Processo
Nível 2	•Gerência de Requisitos	•Medição e Análise •Garantia de Qualidade do Processo e do Produto •Gerência de Configuração	•Planejamento de Projeto •Controle e Monitoramento de Projeto •Gerência de Acordos com Fornecedores	
Nível 3	•Desenvolvimento de Requisitos •Solução Técnica •Integração do Produto •Verificação •Validação	•Análise de Decisão e Resolução	•Gerência do Risco •Gerência Integrada de Fornecedor •Gerência de Projeto Integrada	•Foco no Processo Institucional •Definição do Processo Institucional •Treinamento Institucional
Nível 4			•Gerência Quantitativa de Projeto	•Desempenho do Processo Institucional
Nível 5		• Análise Causal e Resolução		•Inovação Institucional e Implantação

Tabela 3 – Áreas de processo do modelo CMMI discriminadas por nível de maturidade e categoria (SEI, 2002)

No CMMI, a implantação de uma infra-estrutura física e processual para efetuar medições é um requisito desde o nível 2, por estar explícita nas práticas da área de processo *Medição e Análise.* Já no nível 3, surge a demanda de um

PUC-Rio - Certificação Digital Nº 0510997/CA

repositório institucional de medições, na área de processo *Definição do Processo Institucional*. Até então, as medições poderiam estar separadas em repositórios específicos de projeto. No nível 4, as áreas de processo *Desempenho do Processo Institucional* e *Gerência Quantitativa de Projetos* fazem uso intensivo das medições coletadas graça à estrutura estabelecida pela área de medição e análise para gerenciar processos e projetos, respectivamente.

Como a área de processo que efetivamente estabelece o processo de medição é *Medição e Análise*, é ela que nos indicará um ciclo de medição. Ela está divida em 2 metas principais: o alinhamento de atividades de medição e análise com as necessidades institucionais de informação e o fornecimento de resultados de medição, que corresponde à execução da medição e análise propriamente ditas, composta pelas seguintes metas e práticas associadas:

1. **Alinhar atividades de medição e análise**
   - 1.1 **Estabelecer objetivos de medição:** trata do estabelecimento, manutenção, documentação, revisão e atualização de objetivos de medição baseados em necessidades de informação gerenciais, técnicas, de produto, projeto ou processo.
   - 1.2 **Especificar medições:** trata do refinamento de objetivos de medição em medições precisas e quantificáveis, gerando suas definições operacionais. Elas devem ser periodicamente priorizadas, revistas e atualizadas.
   - 1.3 **Especificar procedimentos de coleta e armazenamento de dados:** trata da definição de como coletar e armazenar as medições, especificando procedimentos e ferramentas.
   - 1.4 **Especificar procedimentos de análise:** trata da seleção, especificação e priorização dos métodos e ferramentas de análise usados com os dados de medição.
2. **Fornecer resultados de medição**
   - 2.1 **Coletar dados de medição:** abrange a obtenção das medições-base, o cálculo das medições derivadas, e a verificação da integridade dos dados.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.2 Analisar dados de medição: abrange a execução de análises iniciais, eventuais análises suplementares necessárias e a revisão dos resultados de análise com as partes interessadas.

2.3 Armazenar dados e resultados: trata do armazenamento de dados de medição, especificações de medições e resultados de análise.

2.4 Comunicar resultados: trata da divulgação dos resultados das atividades de medição e análise às partes interessadas, e um eventual auxílio à interpretação subseqüente.

2.1.2.2. Visão de ciclo de medição adotada neste trabalho

Baseado nessa análise dos modelos de referência, derivamos o nosso modelo de ciclo de medição e análise valendo-nos das seguintes etapas (figura 1):

- **Definição:** caracterização das outras atividades do ciclo.
- **Coleta:** ação de obtenção de um dado a respeito de algum ente que queiramos caracterizar.
- **Armazenamento:** registro dos dados coletados em algum tipo de memória persistente.
- **Análise:** processamento e apreciação dos dados visando a aumentar o entendimento sobre algum fenômeno relacionado aos dados coletados.
- **Ação:** uso dos dados de medição analisados para influenciar o ambiente que serve como fonte dos dados.

PUC-Rio - Certificação Digital Nº 0510997/CA

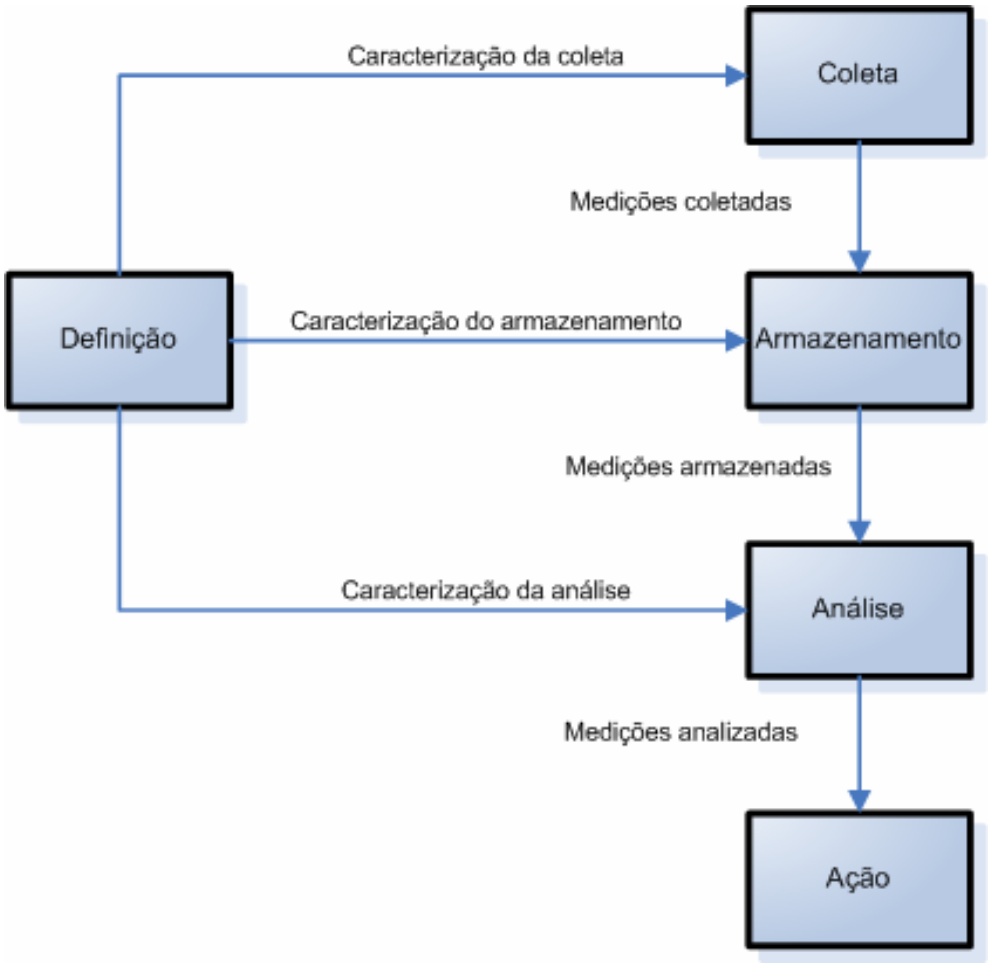

Figura 1 – Visão de ciclo de medição e análise inspirada no CMMI (SEI, 2002) e adotada pelo sistema Clairvoyant

2.1.3. Abordagem do trabalho

Neste trabalho, o escopo se restringirá a uma atividade do ciclo: o armazenamento. Além de estudarmos a etapa de armazenamento, estudaremos também sua interação com as outras etapas do ciclo, ou seja, as influências que elas exercem sobre o armazenamento e vice-versa:

- **Entre armazenamento e definição:** Como é na parte da definição do processo de medição e análise em que se define a coleta de dados, essa definição tem que estar de acordo com as possibilidades oferecidas pela infra-estrutura de armazenamento. É importante ressaltar que idealmente temos as definições de medições coletadas sendo influenciadas somente pelas necessidades de informação da organização ou do projeto. Elas não devem sofrer com restrições relacionadas à infra-estrutura de armazenamento. Assim sendo, a

PUC-Rio - Certificação Digital Nº 0510997/CA

generalidade de uma infra-estrutura de armazenamento de medições é um atributo desejável.

- **Entre armazenamento e coleta:** a interação entre os processos de coleta e de armazenamento se dá no momento em que, já dispondo dos dados coletados, desejamos armazená-los. Para isso, deve-se assegurar compatibilidade entre os dados coletados e a estrutura de armazenamento destinada a eles. Se o formato dos dados coletados diferir do formato esperado pela estrutura de armazenamento, poderemos compatibilizá-los através de processamento e transformação dos dados entre a coleta e o armazenamento (por exemplo, normalização ou conversão de unidades).
- **Entre armazenamento e análise:** A interface entre as etapas de armazenamento e análise se situa no processo de resgate de dados do sistema. Independentemente de desejarmos executar uma análise simples baseada na inspeção dos dados de medição ou uma análise mais sofisticada por meio de uma ferramenta externa, a análise impõe demandas ao processo de consulta. No primeiro caso, o mecanismo de consulta deve ser flexível o suficiente para que o usuário possa expressar a pergunta cuja resposta ele deseja obter nos dados. Já no segundo caso, o mecanismo deve assegurar compatibilidade entre o formato resgatado dos dados e o formato dos dados esperado pelas ferramentas externas de análise. É importante notar que, analogamente ao registro de dados no sistema, o resgate também pode se beneficiar de processamento auxiliar.
- **Entre armazenamento e ação:** os processos de armazenamento exercem influência sobre as ações, mas de maneira indireta – através dos processos de análise. A influência direta do armazenamento sobre o uso das medições é insignificante, assim como a influência direta do uso sobre os processos de armazenamento.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.2.
O armazenamento dos dados de medição de software

Para efetuar o armazenamento de dados de medição de software, necessitaremos de infra-estrutura física e lógica destinada a esse fim. Chamaremos a parte lógica dessa infra-estrutura de *sistema de armazenamento de medições de software*. As responsabilidades atribuídas a tal sistema correspondem à etapa de armazenamento do ciclo de medição e análise e à sua interação com as outras etapas.

O tema deste trabalho é o estudo desses sistemas: como melhor estruturá-los para atender a seus requisitos. Para melhor ilustrar os possíveis requisitos de um sistema de armazenamento de medições de software, fornecemos o seguinte exemplo, presente em (SPC, 2004):

- ser fácil de usar para que seus usuários possam atualizar e divulgar dados com o menor esforço possível;
- ser flexível para poder:
  - evoluir sua estrutura à medida que o programa de medição subjacente evolui e novos dados são coletados;
  - realizar consultas e criar relatórios caso a caso nos dados subjacentes;
  - usar os dados para criar métricas que não haviam sido identificadas anteriormente.
- comunicar-se com outras ferramentas, como gerência de configuração, acompanhamento de defeitos, e sistemas de gestão de projetos. Isso poderia simplificar significativamente a coleta de dados e reduzir a repetição de dados numa instituição;
- ser grande o suficiente para conter informações históricas;
- evitar repetição dos dados;
- fornecer a segurança de acesso necessária, caso a segurança dos dados seja uma questão relevante. Isso inclui a segurança para proteção contra uso não-autorizado assim como corrupção de dados.

Dado que queremos abordar sistemas de armazenamento de dados de medição de software, precisamos determinar o enfoque sob o qual este trabalho os estuda. Esse enfoque será dividido em duas partes: o seu funcionamento interno e

PUC-Rio - Certificação Digital Nº 0510997/CA

a interação do sistema com o mundo exterior. A parte interna corresponde o modelo lógico do sistema. A parte externa, por sua vez, corresponde ao estudo dos macro-processos do sistema.

2.2.1. Modelo conceitual para armazenamento de dados de medição

Para estabelecermos um sistema de armazenamento de dados de medição, é essencial respondermos as seguintes perguntas:

1. Como poderemos resgatar os dados armazenados no sistema?
2. Como serão estruturados os dados coletados quando armazenados?

Como esse sistema de armazenamento pode tomar várias formas, o seu funcionamento interno será abordado de uma maneira abstrata. Os dois elementos abstratos do sistema a serem estudados serão o seu modelo de medição (a estrutura utilizada pelo sistema para representar os dados armazenados) e o seu modelo de consulta (a representação das possibilidades de resgate de dados armazenados no sistema).

2.2.1.1. Modelo de consulta

Chamamos de *consulta* num repositório de medição um procedimento de extração nele feito cuja saída será diretamente interpretada por um ser humano. À representação (*abstração*) das diferentes possibilidades de diferentes consultas possíveis num sistema de armazenamento de medição de software damos o nome de *modelo de consulta*.

Tipicamente, a principal inspiração para consultas em sistemas de medição são as linguagens e modelos de consultas em bancos de dados (e.g., SQL e álgebra relacional) (Silberschatz et al, 1999). Assim sendo, são esperadas num modelo de consulta de um sistema de armazenamento de medição operações como seleção, projeção e agrupamento.

Como se espera que grande parte dos dados de medição armazenados no sistema passe por procedimentos de análise, a maioria dos quais estatísticos, é muito útil ao modelo de medição do sistema a incorporação de operações estatísticas ou que facilitem análises estatísticas. Como exemplo, podemos citar

PUC-Rio - Certificação Digital Nº 0510997/CA

operadores para obter a média ou desvio padrão de um conjunto de valores, ou a capacidade de apresentar uma distribuição de freqüência de dados agrupados.

Por fim, alguns dos dados contidos no sistema são dados temporais como, por exemplo, o tempo médio entre falhas. Logo, a presença no modelo de consultas de operações típicas de linguagens de consultas de bancos de dados temporais, como TSQL (Snodgrass, 1995) seria muito útil.

2.2.1.2. Modelo de medição

Um *modelo de medição* de uma estrutura de armazenamento de medições de software é uma representação abstrata da parcela do ambiente de engenharia de software estudada pelo processo de medição e análise. Esse modelo deve especificar *quais propriedades* estão sendo medidas a respeito de *que objetos de medição*, além de representar como esses atributos se relacionam. Neste trabalho, chamaremos de *entidade* os objetos de medição e as suas propriedades medidas de *atributos*. Existem também os *relacionamentos* entre as entidades, que modelam os mais variados tipos de relacionamentos e associações possíveis entre entidades.

entidade:	produto, processo, projeto, pessoa, organização
atributo:	duração, tamanho físico, tamanho funcional, percentual de progresso, severidade
relacionamento:	“membro” (pessoa/organização), “trabalha” (pessoa/projeto)

Tabela 4 - Exemplos de entidades, atributos e relacionamentos

Os principais requisitos do modelo de medição são a fidedignidade da representação do ambiente de engenharia de software e a facilidade de manipulação para que a inclusão, a exclusão e o resgate de informações do sistema seja o mais simples possível, diminuindo o custo das operações e sua propensão ao erro.

Em certos casos, o repositório pode estar baseado sobre um *meta-modelo de dados*: um modelo de modelos de medição, do qual o modelo de medição em uso para armazenamento num dado momento seria uma instância. Nesses casos,

PUC-Rio - Certificação Digital Nº 0510997/CA

costuma existir algum grau de indireção entre as entidades e os atributos, o que leva a uma maior complexidade na extração de dados por consultas e no registro de informações no sistema.

2.2.2. Macro-processos

Chamamos de macro-processos os conjuntos de operações correlatas que implementam as funcionalidades oferecidas pelo sistema de armazenamento de medições. Os macro-processos previstos no âmbito do armazenamento de medições de software são:

- **Manutenção do modelo de dados:** abrange as primitivas relacionadas à definição e atualização do modelo de medição do sistema do sistema;
- **Importação de dados:** abrange as primitivas de pré-processamento de dados de medição, o mapeamento de tipos de dados de importação com operações de armazenamento e o armazenamento propriamente dito de dados de medição no sistema;
- **Consulta aos dados:** abrange as primitivas de resgate de informações para uso direto humano;
- **Exportação de dados:** abrange as primitivas de resgate e pós-processamento dos dados do sistema para uso em ferramentas externas.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.2.2.1. Manutenção do modelo de dados do sistema

Chamamos de manutenção do modelo de dados do sistema as funcionalidades que permitem definir e alterar o seu modelo de medição. Elas estão intimamente ligadas ao modelo de consulta e mais ainda ao modelo de dados de medição. Podemos posicioná-las em relação ao sistema de armazenamento de medições como tendo uma utilidade análoga à linguagem de definição de dados de um sistema gestor de banco de dados.

Os exemplos típicos dessas funcionalidades são a inclusão de nova entidade no modelo de medição, inclusão de novo atributo, associação entre entidades e

atributos, inclusão de novo tipo de ligação entre entidades e as operações inversas, as de remoção.

2.2.2.2. Importação de dados de medição

Enquanto o modelo de medição do sistema de armazenamento deve garantir a representação mais fiel possível das medições e dos entes medidos em relação ao ambiente de engenharia de software do nosso interesse, o processo de importação de dados deve assegurar que os dados venham desse ambiente e sejam incorporados ao repositório integramente e facilmente.

O processo de importação de dados se divide em três partes: o desdobramento de uma entrada de dados de medição operações de registro de dados (através de um mapeamento construído anteriormente), o pré-processamento dos dados (conversões e transformações dos dados, como por exemplo, normalização e mudança de escala) e o seu efetivo registro no sistema por meio de suas primitivas de armazenamento.

As primitivas de importação de dados de medição correspondem a comandos de importação. Como exemplo de primitivas de importação podemos citar a atribuição de valor a um atributo de uma entidade, a instanciação de entidade, entre outras.

Já como exemplo de transformações de dados podemos citar a normalização de dados, troca de escala (e.g., de “milhares de linhas de código” para “linhas de código”, re-mapeamento – “bom, médio ou ruim” para “1, 2 e 3”)

Finalmente, o processo de desdobramento consiste em transformar uma entrada de um determinado tipo e um mapeamento entre o tipo de entrada e o modelo de medição numa série de operações de registro e transformações. Como exemplos de tipos de dado e ferramentas associadas, podemos citar contadores de linhas de código, sistemas de gerência de incidente, ferramentas de cobertura de testes, entre outros.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.2.2.3. Consulta aos dados do sistema

Já que o modelo de consulta do sistema de armazenamento representa as possibilidades para o "produto" consulta, o macro-processo de consulta rege a funcionalidade que especifica o processo de criação dessa requisição. Esse processo abrange as etapas para se construir uma consulta e as possíveis interfaces para fazer uma consulta (por meio de uma interface visual, ou linguagens de consulta, escrita, por exemplo). Evidentemente, um modelo de consultas bem escolhido possibilita a construção de primitivas mais fáceis de usar e poderosas para o macro-processo de consulta aos dados do sistema. Alguns exemplos de primitivas de consulta são a delimitação do domínio da consulta, a adição de cláusulas de restrição à consulta e a seleção de atributos para a projeção do seu resultado.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.2.2.4. Exportação de dados de medição

A análise dos dados de medição é a razão de ser da própria medição, pois o valor agregado da medição de software se realiza somente quando os dados são utilizados de alguma maneira no ambiente de engenharia de software. Esse uso dos dados de medição freqüentemente incorre na utilização de algum outro sistema – normalmente, para análise ou divulgação dos dados. Assim, os dados são usados, direta ou indiretamente, como insumos para tomada de decisão nesse ambiente. Como exemplos de destinos comuns para os dados, podemos citar planilhas eletrônicas, sistemas estatísticos (e.g., SAS, SPSS), serviços de geração de relatórios de SGBDs e armazéns de dados.

Ao desejarmos efetuar um procedimento de análise com os dados, é necessário que os repassemos à ferramenta que efetuará a análise no formato em que ela os esteja esperando. Por exemplo, se quisermos representar um conjunto de dados como um gráfico de uma nuvem de dispersão, precisaremos exportá-los como duplas de valores, ao passo que se quisermos exportar como histograma, devemos exportar os dados como um rótulo de valor e a freqüência relativa desse valor. Para assegurar essa compatibilidade entre o formato dos dados exportados e o esperado pela ferramenta, pode ser necessário recorrer a pós-processamentos

como transformações (e.g., transformar freqüências relativas em acumuladas), agrupamento de dados ou outros.

No mais, a exportação pode, freqüentemente, recorrer às funcionalidades de consulta do sistema ou ao seu modelo de consulta. Desta maneira, poderíamos descrever uma saída de dados exportados em termos do resultado de uma consulta feita ao sistema. Por exemplo, podemos querer executar uma análise de regressão linear entre os tamanhos físico e funcional de sistemas. Para isso, forneceríamos à ferramenta encarregada dessa análise o resultado de uma consulta que recuperasse os tamanhos físicos e funcionais de cada sistema estudado.

Por fim, é possível que a análise seja precedida de uma etapa de pós-processamento dos dados armazenados. Por exemplo, podemos querer descartar os valores extremos do conjunto de dados efetivamente exportado pelo sistema ao efetuarmos procedimento de exportação.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.2.3. Escopo do trabalho

Como julgamos que não seria possível desenvolver um repositório de medições completo no âmbito de uma dissertação de mestrado, algumas partes das primitivas acima foram excluídas do escopo deste trabalho. Procuramos manter no escopo um conjunto coerente de funcionalidades que tivesse utilidade a um possível usuário final e que abordasse problemas interessantes do ponto de vista teórico. Logo, mantivemos as seguintes funcionalidades em cada macro-processo*:*

- **Manutenção dos dados de medição:** mantida por completo. Estudaremos como o meta-modelo de medição afeta as suas primitivas e qual é o desdobramento delas no contexto desse modelo.
- **Primitivas de armazenamento:** mantivemos somente a parte da importação após pré-processamento e transformações. Estudaremos qual é o desdobramento dessas primitivas no contexto do meta-modelo de medição.
- **Consulta:** mantida por inteiro. Estudaremos o desdobramento das primitivas de suas primitivas, baseado no modelo subjacente de consulta e no meta-modelo de medição.

- **Exportação:** a composição de primitivas simples de exportação baseadas nas funcionalidades de consulta.

Por conseguinte, retiramos do escopo do trabalho as seguintes funcionalidades:

- **Na importação:** estudo e elaboração de mecanismos de importação abrangendo transformações e pré-processamento (normalização, verificação de integridade, etc.).
- **Na exportação:** o pós-processamento dos dados armazenados.

2.3. Alternativas para armazenamento

A necessidade de armazenamento dos dados de medição decorre da necessidade do seu uso posterior à sua coleta. Como defendemos o uso de um repositório como instrumento para armazenamento das medições coletadas, temos que especificar o nosso conceito de repositório. Para isso, vamos apresentar definições de repositório encontradas na literatura. Além disso, necessitamos compará-lo com outras opções de armazenamento para justificar a nossa preferência.

2.3.1. Repositório de medição de software

Para caracterizar uma definição de repositório (não específica a repositórios de medições de software), partimos de duas definições provenientes da literatura especializada:

> A repository is a shared database of information about engineered artifacts produced or used by an enterprise.
>
> Bernstein e Dayal (1994)

> The repository is the mechanism for the definition, storage, and management of all information about an organization, its data and its software systems as well as its access.
>
> McClure (1992) apud Braungarten (2005)

PUC-Rio - Certificação Digital Nº 0510997/CA

Baseado nas definições acima, e no conceito de sistema de armazenamento de dados de medição de software, definimos como repositório de medições de software um sistema de armazenamento de medição de software, especializado para esse fim e baseado num banco de dados. Não deve haver interação direta do usuário do repositório com o banco de dados; toda e qualquer operação parte dos macro-processos de armazenamento deve ser encapsulada pelo sistema.

O sistema deve embutir conhecimento do domínio de medição e análise de software e da semântica dos dados advindos de medição de software. Esse conhecimento se manifestaria no macro-processo de importação pela adequação das primitivas oferecidas às possíveis fontes de dados de medição de software. Analogamente, as primitivas de exportação teriam de ser adequadas aos possíveis destinos dos dados de medição. Além disso, as funcionalidades de consultas devem corresponder às necessidades de informação típicas na análise de dados de medição de software. Por fim, o repositório deve ser capaz de armazenar meta-dados de medição.

Como exemplo do conhecimento da semântica dos dados de medição de software, podemos citar:

- Classificação dos dados lançados nos tipos de escala previamente apresentados;
- Proibição de certas operações para certos tipos de escala (não podemos, por exemplo, somar atributos ordinais);
- Otimização da construção de consultas relacionando cláusula da consulta com tipos de escala permitidos para uso nessa cláusula (por exemplo, só podendo agrupar por tipo de escala ordinal ou nominal)
- Facilidade para mapeamento de tipos de entrada de dados para o modelo de medição do repositório

2.3.2. Comparação com outras opções de armazenamento

Para justificar a defesa do repositório como instrumento preferido para armazenar os dados de medição de software, vamos compará-lo com duas alternativas populares: planilhas eletrônicas e bancos de dados (desprovidos de uma camada de repositório).

PUC-Rio - Certificação Digital Nº 0510997/CA

2.3.2.1. Planilhas eletrônicas

Uma primeira alternativa para o armazenamento de dados de medição é uma planilha eletrônica. Sua estrutura é formada por *células* referenciadas por *coordenadas*, onde o conteúdo de cada célula pode ser um valor alfanumérico ou uma *fórmula*, por meio de expressões que podem referenciar funções prontas, funções definidas pelo usuário ou outras células. Ela é certamente a alternativa mais simples, pois o registro de dados é rápido e fácil devido à interface gráfica com o usuário, e apresenta baixo custo (existem, inclusive, várias alternativas gratuitas de qualidade). Por fim, a maior parte das planilhas eletrônicas existentes são dotadas de funcionalidades significativas de análise de dados.

Uma das principais características das planilhas eletrônicas é a impossibilidade definir uma estrutura para seus dados. Isso é ao mesmo tempo uma vantagem e um problema. Isso aumenta a facilidade de adaptação do instrumento de armazenamento quando se muda o modelo de medição, mas dificulta a extração de dados por meio de consultas e a interface com outras ferramentas.

A integração de dados também é bastante problemática (por exemplo, no caso em que guardemos uma planilha de medições por projeto e queiramos fazer uma análise histórica abrangendo vários projetos). A planilha também sofre com a dificuldade de transformações levando a dificuldade de apresentar diferentes visões dos dados para diferentes usuários.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.3.2.2. Bancos de dados

Uma alternativa para persistir os dados de medição seria armazená-los em um banco de dados, no qual o modelo de medição estaria descrito em termos do esquema de dados do banco. Ao contrário da planilha eletrônica, o banco de dados nos permite a estruturação dos dados de medição, por exemplo, através de uma *Linguagem de Definição de Dados* (Silberschatz et al, 1999).

A estruturação dos dados possibilitada pelo banco de dados nos permite mais facilidade em transformar os dados, facilitando a criação de visões de

usuário e a interface com outros sistemas. Essa estruturação também aumenta a flexibilidade de construir as consultas aos dados da maneira que quisermos. No mais, ganhamos em integridade e segurança pelas funcionalidades correlatas fornecidas em sistemas gestores de bancos de dados.

Como desvantagem em relação às planilhas eletrônicas, podemos mencionar o maior custo e menor disponibilidade de sistemas gestores de bancos de dados. Eles também são mais difíceis de utilizar.

2.3.2.3. Repositório de medições

Como o repositório de medições é um banco de dados acrescido de uma camada que o encapsula, ele dispõe de todos os serviços relevantes de um banco de dados. Assim ele tem todas as vantagens de um banco de dados em relação a uma planilha eletrônica. Ele aumenta a usabilidade pela possibilidade de estruturação dos dados e por usar os serviços subjacentes do banco (controle de acesso, integridade, etc.).

A sua vantagem em relação aos bancos de dados é o conhecimento de domínio embutido. Conseqüentemente, torna-se mais fácil gerar consultas e relatórios adequados aos usos da análise de dados de medição de software. Além disso, dispomos de uma melhor integração com outras ferramentas, já que as interfaces de importação e exportação de dados são mais adequadas às devidas ferramentas emissoras e receptoras.

Como desvantagem em relação às alternativas mencionadas anteriormente, podemos citar o seu maior custo (de aquisição ou construção, dependendo do caso) e o esforço de gerenciamento que ele gera.

2.4. Evolução transparente do modelo de medição

As necessidades de informação de ambientes de engenharia de software costumam variar no tempo, por motivos que serão explicados a seguir. Temos como conseqüência as mudanças no conjunto de medições e entes medidos nesses ambientes.

PUC-Rio - Certificação Digital Nº 0510997/CA

Na literatura de medição de software, costuma-se recomendar que programas de medição comecem com poucas medições e cresçam gradativamente (McGarry et al., 2003) (Rifkin e Cox, 1991, apud Fenton e Pfleeger, 1997). Assim, a capacidade de medição de uma instituição crescerá num ritmo que pode ser efetivamente compreendida e implementada pelos participantes do seu programa de medição. Se partirmos dessa premissa, temos como conseqüência o aumento do número de medições coletadas com o passar do tempo.

Outro fator levando a mudanças no conjunto de medições coletadas nesse ambiente é a iteratividade da coleta de dados causada pelo ganho de conhecimento incremental sobre o ambiente ou, dito de outra maneira:

> The data collection process is clearly iterative. The more we learn, the more we know about what other data we need and how better to collect it.
>
> Basili (1980) apud Harisson (2004)

Isso significa que nos depararíamos com as questões do aumento do conjunto de medições coletadas e a obsolescência: o desuso de uma medição.

No mais, pode ocorrer mudança nas necessidades de informação devido a mudanças ocorridas no ambiente: mudança de ferramentas, de processo de desenvolvimento ou de paradigma de desenvolvimento. Por exemplo, pode ser que sejam coletadas métricas de código orientadas a objeto num ambiente onde esse tipo de paradigma de programação se faz presente. Quando esse paradigma pára de ser usado, as métricas se tornam obsoletas, logo deixam de ser coletadas. Em contrapartida, poderiam surgir demandas por novas medições ao adotarmos um novo paradigma de programação.

Por fim, podemos ter projetos diferentes com necessidades de informação diferentes (McGarry et al., 2003). Por exemplo, poderíamos ter um risco específico associado a um projeto levando-nos a coletar uma medição específica, como uma demanda de medir a volatilidade de requisitos num projeto onde acreditamos que a variação de requisitos será tão freqüente que terá um impacto perceptível sobre o mesmo. Assim sendo, o conjunto de métricas coletadas pode variar no tempo como função da mudança do conjunto de medições coletadas em cada projeto.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.4.1.
Caracterização da evolução

A mudança no conjunto de medições coletadas e entes medidos caracterizada acima provoca, no contexto de um repositório de medição de software, a evolução do seu modelo de medição. É esse o fenômeno ao qual nos referiremos doravante como evolução no repositório de medição. Faremos a seguir um paralelo entre a evolução do modelo de medições de um repositório com a evolução de um esquema de banco de dados, já que, no contexto deste trabalho, um repositório é um banco de dados especializado.

Para definirmos evolução de esquemas de bancos de dados, necessitamos primeiramente definir o conceito de modificação em bancos de dados. Podemos defini-lo como a mudança da definição do esquema de um banco de dados povoado. Já a definição de evolução num esquema de banco de dados corresponde a uma modificação no esquema sem a perda de dados existentes. Em especial, há o versionamento de esquema, que ocorre quando o SGBD permite visualizarmos todos os dados, retrospectivamente e prospectivamente, através de interfaces de versão definíveis pelo usuário (Roddick et al, 1993).

Mais especificamente, podemos dividir a evolução de esquemas em evolução da estrutura do esquema (modificação ocorrida nas partes constituintes de um BD) e do domínio dos atributos, i.e., o conjunto de valores permitido para um atributo. Para melhor exemplificar a modificação da estrutura do esquema de um banco de dados, daremos um exemplo das possíveis modificações de esquema num banco de dados relacional (tabela 5):

PUC-Rio - Certificação Digital Nº 0510997/CA

Operações de evolução
Adicionar entidade
Excluir entidade
Renomear entidade
Transformar entidade em atributo de relacionamento
Transformar entidade comum em entidade fraca
Transformar entidade fraca em entidade comum
Adicionar atributo a entidade
Adicionar atributo a relacionamento
Adicionar sub-atributo a atributo
Excluir atributo de entidade
Excluir atributo de relacionamento
Excluir sub-atributo de atributo
Renomear atributo
Transformar atributo em entidade
Incluir atributo como atributo identificador
Remover atributo de conjunto identificador
Adicionar relacionamento entre duas ou mais entidades
Adicionar ligação entre relacionamento e entidade
Excluir ligação entre relacionamento e entidade
Excluir relacionamento
Renomear relacionamento

Tabela 5 – operações de modificação de estrutura de esquema em bancos de dados relacionais, segundo (Roddick et al, 1993)

Ao transpor essas operações de mudança de esquema a um contexto de modelos de medição de repositórios de medição, poderemos definir as primitivas de evolução do repositório.

Observemos que funcionalidades de migração e conversão de dados podem ser úteis durante a evolução de dados, tanto no repositório de medição quanto em esquemas de bancos de dados. Entretanto, elas não são consideradas como parte das funcionalidades de evolução, por se tratar de um problema mais amplo.

PUC-Rio - Certificação Digital Nº 0510997/CA

2.4.2.
Transparência e outros requisitos da evolução

A capacidade de um repositório de evoluir seu modelo de medição pode aumentar consideravelmente a sua utilidade. Um repositório evolutivo tornaria mais fácil a tarefa de representar a realidade mutante do ambiente de engenharia de software do qual o seu modelo de medição é uma abstração. Entretanto, a solução utilizada para provermos essa capacidade tem que ser cuidadosamente escolhida, pois algumas delas poderiam ser contraproducentes. Por exemplo, o particionamento de repositórios, criando subdivisões no repositório a cada vez que se modificasse o conjunto de medições coletadas, tornaria muito trabalhosa a tarefa de resgate dos dados para análise.

Já a adição de um mecanismo de indireção explícito ao modelo de medição, como em (Harisson, 2004), significa que o usuário teria que levar em conta entidades de ligação e operações de junção ao elaborar consultas num repositório. Desta maneira a elaboração de consultas se tornaria mais complexa – logo, mais demorada e propensa a erros. Conseqüentemente, a usabilidade do repositório seria negativamente afetada.

Este trabalho propõe como requisito para repositórios de medição a capacidade de evolução transparente, ou seja, sem que seja necessário que o usuário do repositório saiba como é feita essa evolução nem que o acréscimo dessa capacidade torne mais complexa a execução dos macro-processos do repositório (em relação a um repositório não evolutivo). Na prática, podemos precisar esse requisito como o encapsulamento da abordagem de evolução do repositório (requisito de usabilidade) e a contenção do impacto de mudanças de medições (requisito de interoperabilidade).

Queremos que o usuário possa tirar proveito da capacidade de evolução do repositório de medição sem ter que saber como seu mecanismo de evolução funciona. Para isso, suas funcionalidades devem ser acessíveis a partir de interfaces projetadas para serem, na medida do possível, tão simples quanto seriam num repositório não-evolutivo. Com isso, queremos dizer que elas não devem expor a natureza da associação das entidades e atributos. Elas não poderão expor elementos subjacentes do mecanismo de evolução em uso como, por exemplo:

PUC-Rio - Certificação Digital Nº 0510997/CA

- indireções
- junções
- versionamento de entidades

Evitar que o usuário acesse diretamente o banco de dados é imperativo para a evolução transparente pois, caso contrário, ele seria obrigado a conhecer como funciona o mecanismo de evolução para utilizar o repositório de maneira efetiva.

Outro requisito da evolução transparente é a contenção do impacto de alterações no modelo de medição. Para tal, o mecanismo de acesso a dados deve ser robusto em relação a mudanças no modelo de medição para poder preservar o ponto de vista original das operações de consulta aos dados. Mais precisamente, vinculamos esse nosso conceito de robustez das consultas à obediência de duas condições:

- **Estabilidade de interface:** nenhuma alteração do modelo de medição deve alterar os tipos de dados de medição retornados por uma consulta, ainda que essa alteração seja a exclusão de um tipo de dado de medição referenciado pela consulta do modelo de medição. Evidentemente, se uma medição referenciada por uma consulta não fizer mais parte do modelo de medição, não será mais possível lançar novos valores, mas isso não quer dizer que os seus valores antigos (lançados antes de ela deixar de fazer parte do modelo) deixarão de poderem ser resgatados. Já para as entidades posteriores à exclusão deste atributo, o valor do atributo deverá constar como nulo.
- **Independência temporal:** todos os dados de medição que satisfaçam às condições de uma consulta serão resgatados pelas mesmas, independentemente de qualquer evolução no modelo de medição posteriormente ao lançamento dos dados.

Além dos requisitos ligados à transparência, queremos que a abordagem para a evolução no repositório de medição obedeça a mais dois requisitos:

- **Compartilhamento de instância (Ra e Rundensteiner, 1997):** por motivos de desempenho e de consistência dos dados queremos quaisquer referências a uma instância de uma medição deveram apontar para uma única instância física da medição na estrutura subjacente do banco de dados. Temos como corolário que o

PUC-Rio - Certificação Digital Nº 0510997/CA

mecanismo de evolução não pode ser baseado na replicação de dados ou no particionamento do repositório.

- **Reversibilidade de operações:** todos os tipos possíveis de alterações do modelo de medição devem contar com uma operação inversa. Notadamente, para cada operação correspondente à inclusão de algum elemento deverá existir uma operação de exclusão, e para toda operação de exclusão de um elemento deverá ser possível a sua reintegração ao modelo (possivelmente com a própria operação de inclusão). Queremos, inclusive, que essa reintegração seja feita sem perda dos dados de medição anteriores à operação de exclusão revertida (conforme explicado acima). Além de facilitar a operação cotidiana do repositório, essa reversibilidade seria especialmente interessante se quisermos auditá-lo.

PUC-Rio - Certificação Digital Nº 0510997/CA