# 4
# Aplicações do Modelo de Proveniência

Para a análise do modelo conceitual de proveniência, também referenciado como modelo de proveniência, avaliamos quatro diferentes domínios de aplicação. Inicialmente, as preliminares apresentam a importância de uma API de serviços para proveniência e como consultas a proveniência poderiam ser estruturadas (seção 4.1). Em seguida, utilizamos dois domínios como motivação: o primeiro domínio analisa a aplicação do modelo sob a perspectiva de aplicações semânticas para desktops (seção 4.2) e o segundo domínio propõe a discussão do modelo como uma possível generalização a ser considerada para *Design Rationale* (seção 4.3). Em seguida, no terceiro domínio realizamos um estudo preliminar (seção 4.4) que sugere a adoção do modelo conceitual de proveniência para o desenvolvimento de um centro de informações que tem atribuições de registrar a história de um empreendimento. Por fim, concluímos nossa avaliação com o quarto domínio (seção 4.5), que utiliza dados reais de um projeto de catálogo, a partir do mapeamento da ferramenta adotada pela equipe de desenvolvimento para o gerenciamento de configuração de software.



## 4.1.
## Preliminares

Relembramos aqui que Simmhan et al. (2005) conclui que o projeto PASOA (Moreau & Ibbotson, 2006) está na direção certa ao buscar a definição de uma API para proveniência, mas ressalta que precisa de maiores refinamentos para elucidar como a proveniência é representada e recuperada. Acrescenta ainda que qualquer padrão proposto para proveniência deve satisfazer as necessidades de múltiplos domínios, e tais requisitos necessitam ser identificados. Conclui, apontando que um deles é seguramente a padronização da semântica de termos.

Bose & Frew (2005) esclarecem algumas questões gerais que podem servir de guias para métodos de recuperação de linhagem (*lineage*) utilizados para sistemas protótipos: como alguém pode navegar pela linhagem de um item em um sistema? Quais métodos são utilizados para a recuperação da linhagem? Quais consultas relativas à linhagem podem ser feitas a um sistema? Como são os resultados dessas consultas? Bose & Frew (2005) respondem que qualquer

navegador deveria poder acessar um objeto de linhagem, onde o usuário fornece um identificador de um item de interesse e especifica a quantidade de níveis de linhagem para trás ou para frente a serem recuperadas. Descreve ainda que, no caso do sistema ESSW (Bose & Frew, 2005), a aplicação utiliza consultas recursivas em SQL retorna os resultados como um grafo no formato GraphViz[50] de objetos científicos. Conclui que, clicando em qualquer objeto do grafo, o respectivo metadado pode ser visualizado.

## 4.2. Aplicações de Desktop Semântico

A modelagem de proveniência pode ser explorada para apoiar a recuperação e análise de dados baseados em contexto, por exemplo, para a construção de aplicações para *desktop* semântico (Marins et al., 2007).

Com o atual volume de informação disponível para consulta através da Web, validade e propriedade intelectual tornam-se, mais do que nunca, essenciais. Ambos estão listados entre os problemas a serem atacados pela comunidade Científica de Computação, ao final da seção 2 do documento (Carvalho et al., 2006). Permitem questionamentos relacionados à proveniência: "É possível confiar na validade de uma dada informação? De onde veio?". Ou simplesmente, qual a sua origem?

Em setembro de 2006, o mecanismo de indexação do Google processou 850 TB de dados brutos coletados da Web. Considerando que o *crawler* tem uma capacidade de compressão de 11%, estamos falando de 8.5 Peta Bytes de dados disponível *on line* (Chang et al., 2006). Sem contar com transmissões de rádio e TV, que ainda não estão totalmente digitalizadas ainda, uma pessoa pode encontrar (e comprar) praticamente qualquer informação em formato digital. As listas de divulgação comercial escalaram com a venda de listas de usuários de cartões de crédito (acima de três bilhões de números), arquivos criminais (100 milhões), o nome e o endereço de cada cidadão Mexicano, identidade e telefone de cada Argentino, como anunciado pelas companhias LexisNexis e Choice Point (Gaspari, 2005).

[50] http://www.graphviz.org/

Em virtude do crescimento exponencial dos dados digitais, é virtualmente impossível para seres humanos, gerenciar a complexidade e volume de informações disponíveis. O perigo é a criação de banco de dados somente de leitura ("*write only*" *databases*), nos quais a informação é constantemente armazenada, mas impossível de ser minerada para propósitos significativos. Este fenômeno levanta uma ameaça à usabilidade da Web atualmente.

Novas abstrações são necessárias para ajudar a modelagem e sumário de volumes massivos de dados para uma dimensão processável pelas pessoas. Pesquisadores da indústria, governo e academia estão agora explorando a possibilidade de criar uma Web com mais significado (semântica) na qual o conhecimento é colocado de forma mais explícita, permitindo que máquinas, em oposição a humanos, processem e integrem recursos de forma inteligente. (Breitman et al., 2007).

As aplicações para desktop semântico exploram tecnologias inovadoras da Web Semântica para oferecer uma solução computacional para o que é conhecido como "problema da sobrecarga de informação" (*information overload problem*). Em particular, o uso de metadados de proveniência pode ser aplicado para melhorar a recuperação e análise de dados baseado em contexto no ambiente da computação pessoal. Aplicações para desktop semântico, em geral, se beneficiariam de um modelo de proveniência, tornando-se então, aplicações aptas a recuperar proveniência (*provenance-ready applications*).

A metáfora de área de trabalho e interface de trabalho (*desktop interface*) ajudam indivíduos a gerenciar dados armazenados em seus computadores pessoais e representam um ponto de entrada único para as aplicações disponíveis para produzir, armazenar, acessar e gerenciar objetos digitais pessoais ou corporativos. O conjunto de métodos, estrutura de dados e ferramentas que estendem a área de trabalho e atribuem um significado bem definido representa o que se convencionou chamar de área de trabalho semântica (*semantic desktop*). Se a palavra "social" (Chernov et al., 2006) é adicionada como sufixo, então significa que a área de trabalho está habilitada a trocar dados além das fronteiras individuais e com isso melhorar a colaboração *on line* e ajudar na organização dos dados criados por um grupo de usuários.

Rastrear a origem do dado, através da utilização de metadados de proveniência, é uma forma interessante em atribuir mais significado ao dado. Metadados de proveniência incluem relacionamentos relevantes entre fatos, pessoas, publicações e qualquer outro objeto digital, bem como a resenha histórica entre essas entidades.

Aplicações de desktop oferecem inúmeras oportunidades de captura de metadados de proveniência. Por exemplo, quando um anexo de uma mensagem de correio eletrônico é salvo no disco rígido, as informações sobre a conexão deste arquivo e quem o enviou podem ser armazenadas com metadados de proveniência. Quando se abre o navegador a partir de um link no corpo de uma mensagem e se faz o download de um arquivo daquela página Web, também oferece um rico conjunto de metadados de proveniência para este arquivo. Estes dois exemplos ilustram porque aplicações de desktop devem ser instrumentalizadas para salvar e armazenar metadados de proveniência, que serão utilizados mais tarde para a recuperação dos dados.

Idealmente, os metadados de proveniência devem ser coletados de todas as aplicações de desktop para permitir que mais significado seja armazenado e não apenas o conteúdo. A ferramenta de indexação e recuperação de conteúdo no desktop é uma tecnologia base para explorar diferentes tipos e formatos de conteúdos locais. A Figura 38 apresenta um *framework* simplificado para aplicações de desktop semântico, adaptado de (Sauermann, 2005).

Várias soluções de buscadores para desktop (d*esktop search*), como o Google Desktop, Microsoft Desktop, Copérnico Desktop e Beagle, recentemente surgiram para suprir a dificuldade de recuperação de informações armazenadas nos computadores pessoais. Boa parte delas ainda não é compatível com proveniência.

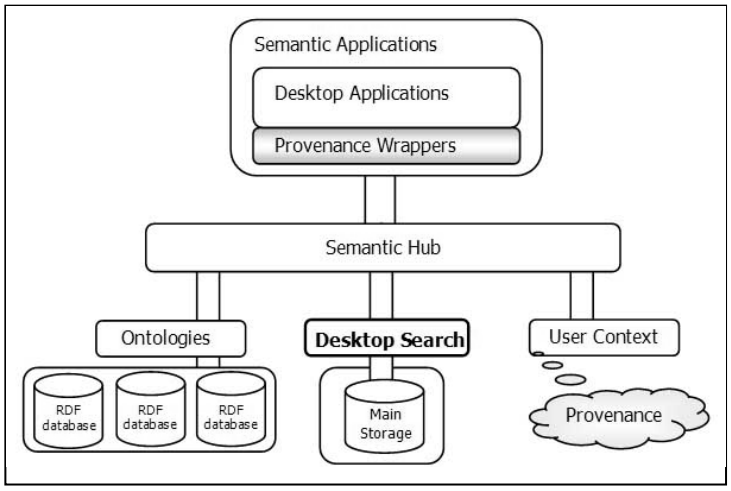


Figura 38: Arquitetura Simplificada para Desktop Semântico

O gerenciador semântico (*Semantic Manager*) interfaceia entre empacotadores (*provenance wrappers*) de proveniência, contexto do usuário e

ontologias de domínio disponíveis, para coletar os dados de proveniência. Na camada final da arquitetura, no meio da Figura 38, temos o componente de busca do desktop que idealmente deve estar disponível e integrado a todos os aplicativos.

Esse componente de busca no desktop (*desktop search*) indexa, armazena e consolida localmente as informações pessoais. Também habilita a consulta ao conteúdo, ordenando e classificando os resultados a partir dos metadados de proveniência. Aplicações semânticas seriam, entre outras, aplicações de desktop enriquecidas com a capacidade de capturar e recuperar a proveniência.

Assim, essa instanciação sugere que o armazenador de metadados tradicional deve ser repensado como um componente dedicado a armazenar os metadados de proveniência (*provenance storage*). Um possível caminho para construção desse componente apresentado na Figura 39, é a aplicação direta do modelo de proveniência (seção 3.5). Nessa figura, os índices de conteúdo e de metadados de proveniência estão fisicamente separados. Os metadados têm demandas de gerenciamento distintas das de conteúdo. Por exemplo, requer políticas e procedimentos de arquivamento e descarte específicos.

O lado esquerdo da Figura 39, ilustra que sempre que um evento ocorre no sistema operacional, por exemplo, o salvamento de um anexo de uma mensagem eletrônica, os empacotadores de conteúdo e proveniência são acionados e responsáveis por suas respectivas indexações criando um fluxo unidirecional do evento até os índices.

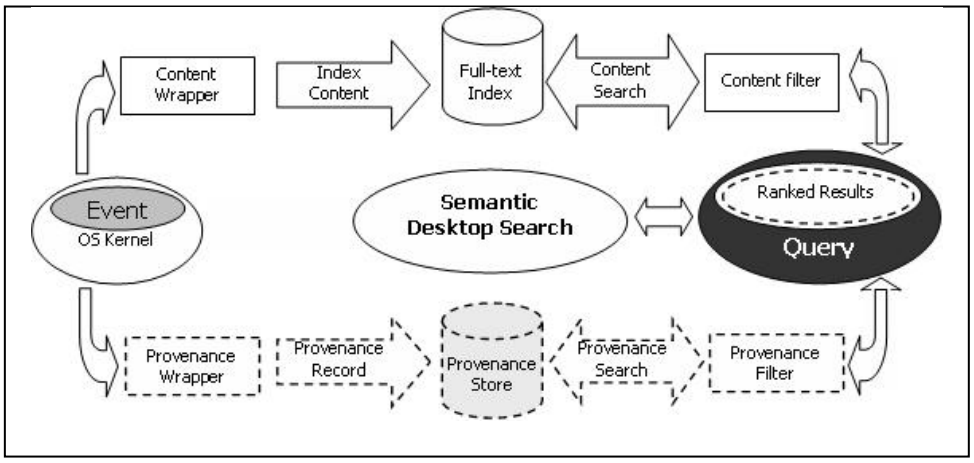


Figura 39: Arquitetura de Busca em Desktop Semântico baseada no Beagle$^{++}$

Por outro lado, à direita o fluxo bi-direcional representa a consulta a esses índices. A ordenação e classificação do resultado são dadas por palavras-chave

que estejam no conteúdo (filtros de conteúdo) ou pelos metadados de proveniência (filtros de proveniência).

Questões relacionadas à granularidade, escalabilidade e segurança colocadas por (Braun et al, 2006) influenciam a modelagem do armazenador dos metadados de proveniência. Na verdade, o item segurança, é crucial para identificar entidades confiáveis, estabelecer propriedade, controlar acesso, registrar dados de auditoria e rastrear mudanças.

Recuperar a informação de proveniência é uma consulta ao modelo de proveniência a procura de um ou mais objetos. Uma consulta a proveniência tipicamente possui dois tipos de filtros (Miles, 2006):

- Manipulador do dado (*query data handle*), que identifica objetos de interesse;
- Filtro para relacionamentos (*relationship target filter*), que define o escopo da consulta, restringindo o resultado a um volume processável;

A Figura 40 ilustra um exemplo, baseado no Beagle++, com o resultado de uma consulta apresentando três itens que foram retornados a partir de uma pesquisa pelos termos "*semantic desktop*". O item retornado corresponde a um email que contém um artigo anexado. O destaque da elipse identifica a ligação desse anexo com o seu contexto.

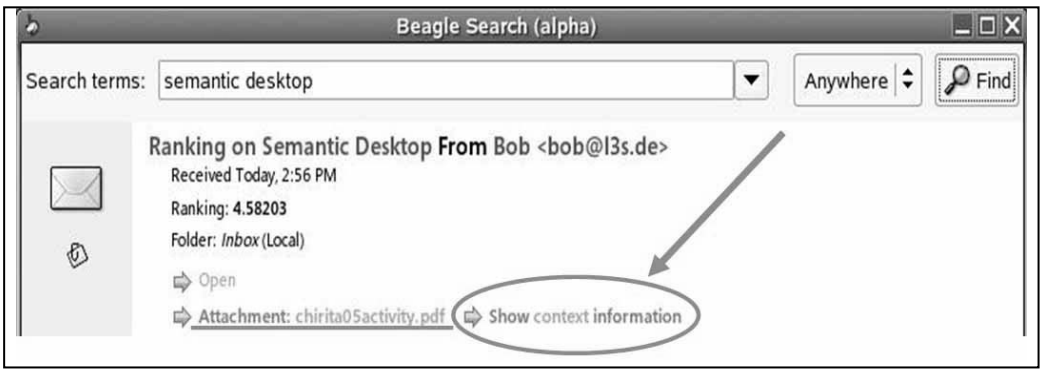


Figura 40: Resultado de uma consulta semântica

A Figura 41 é o detalhamento da elipse destacada na Figura 40, identificando com retângulos os conceitos abstratos de proveniência associados a cada parte do contexto, neste caso, referente à publicação do artigo que está em anexo ao email, e corresponde respectivamente ao item retornado, apresentado na Figura 40.

Nesse mesmo exemplo, poderíamos identificar outro evento que seria o evento relacionado ao envio da mensagem propriamente dita. Seria então, importante, identificar quem enviou a mensagem e a data de recebimento, o que respectivamente já associaria dois conceitos abstratos de proveniência.

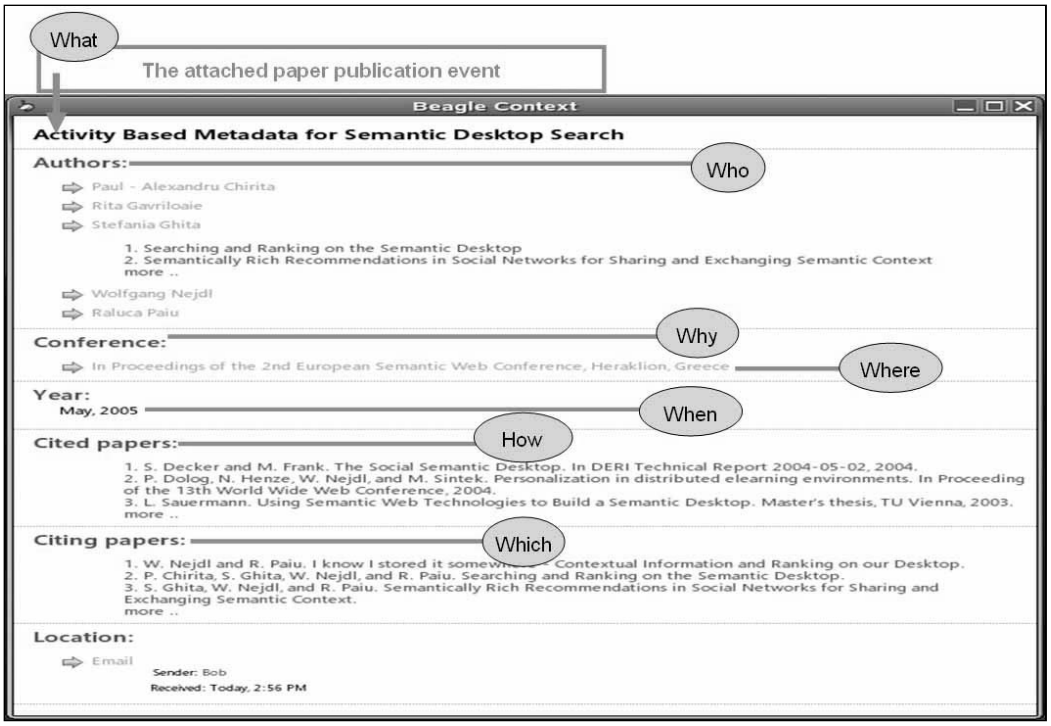


Figura 41: Conceitos de Proveniência ajudam a organizar e classificar o resultado de uma consulta semântica.

## 4.3. Generalização para Design Rationale

Outra aplicação do modelo de proveniência é referente à generalização de *design rationale*, usualmente adotado para representar esquemas baseados em argumentação porque oferece uma infra-estrutura para identificar quais decisões foram tomadas e quais os motivos relacionados. O modelo de proveniência adota o conceito de um conjunto de descrições de qualquer mudança de um objeto de informação, que registre sua história e que seja significante para resguardar autenticidade, integridade e interpretação, e pode ser visto como uma generalização para *design rationale*.

Medeiros (2006) descreve o modelo Kuaba, apresentado na Figura 42, como um conjunto de elementos (classes, propriedades, relações e restrições) que expressam o domínio de *design rationale*. Acrescenta que permite a representação explícita das decisões tomadas durante o projeto de software e

suas justificativas. Além disso, captura as relações entre os elementos de argumentação e os artefatos gerados, e inclui também, a integração da estrutura de argumentação com as descrições dos artefatos produzidos e com informações sobre a história de projeto do artefato: quando as decisões foram tomadas, quem tomou as decisões, que método de design foi utilizado etc. Essas são perguntas que podemos interpretar como consultas a proveniência das decisões. Há claramente a preocupação com a rastreabilidade do raciocínio utilizado ao longo do ciclo de desenvolvimento de um software, ratificado pela necessidade de registro da história.

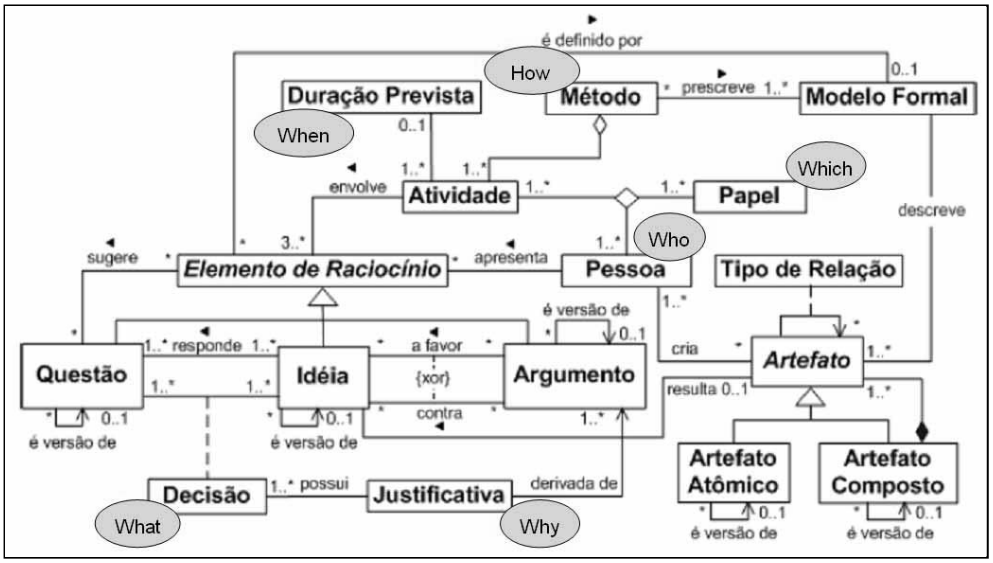


Figura 42: Modelo Kuaba e conceitos abstratos adaptado de (Medeiros, 2006)

Focaremos nossa análise no mapeamento entre elementos do modelo Kuaba e classes do modelo de proveniência. Breitman et al. (2007) argumenta que mapas de analogia facilitam a modelagem conceitual permitindo que o projetista reinterprete fragmentos de modelos conceituais conhecidos em outros contextos. Analogamente à construção do modelo, iniciamos identificando qual classe representa o conceito de evento no modelo Kuaba. O principal propósito do modelo é **"identificar quais decisões foram tomadas e quais os motivos relacionados."**. As decisões são então os eventos que estamos interessados. A Tabela 22 propõe um mapeamento entre elementos do modelo Kuaba e conceitos do modelo de proveniência.

Tabela 22: Analogia entre Design Rationale e Proveniência

| **Elemento Kuaba** | **Conceito de Proveniência** | **Conceito Abstrato** |
|---|---|---|
| Decisão | E5 Event | *What* |
| Duração Prevista | E52 Time-Span | *When* |
| Atividade | E7 Activity | - |
| Método | CID5 Means | *How* |
| Pessoa | CID2 Participant | *Who* |
| Justificativa | CID4 Reason | *Why* |
| Papel | CID3 Functional Role | *Which* |
| Artefato<br>Artefato Atômico<br>Artefato Complexo | CID2 Participant | *Who* |
| Tipo da Relação | E55 Type | *?* |
| Elemento de Raciocínio<br>Modelo Formal<br>Questão<br>Idéia<br>Argumento | E28 Conceptual Object | *?* |
| *?* | E53 Place | *Where* |

Os pontos de interrogação “?” da Tabela 22 representam questionamentos não resolvidos neste exercício e demandam maior aprofundamento que não exploraremos nesta dissertação. Todas as propriedades existentes no modelo Kuaba devem ser analisadas com maior profundidade para avaliar quais propriedades do modelo de proveniência podem ser reusadas e quais precisam ser especializadas ou compostas. Uma possibilidade seria utilizar a expansão DISP (seção 3.5.2.2.2) a partir da importação das classes *Determinant, Immanent, Source e Product* da ontologia de topo de Sowa (2001b). Os artefatos – atômicos e complexos – poderiam ser então instâncias de *Source* ou Product, ou de suas especializações. No modelo Kuaba há um relacionamento ternário entre Atividade, Papel e Pessoa. Com isso, outra importação seria necessária: a classe Functional Role da DOLCE, por exemplo, seria importada para o modelo como a classe CID3 Functional Role, para possibilitar a reificação da propriedade “P14.1 in the role of” presente no modelo conceitual. Neste caso, também seriam necessários a criação de dois pares de relacionamentos (com inversa) que ligariam: a classe CID3 Functional Role à classe CID2 Participant (ou alguma de suas especializações) e à classe E7 Acitivity. Por hora, esta dissertação não avançará a análise, deixando-a como proposta para ser estudada como alternativa para o projetista do domínio de *Design Rationale*.

Avançaremos agora para um exemplo mais abrangente, acrescentando um estudo preliminar de um centro de informações que adota o modelo de proveniência para a preservação digital (seção 4.4).

## 4.4. Centro de Informações

Ao longo desta seção realizaremos um estudo preliminar da aplicação do modelo de proveniência para um centro de informações (CI) de um empreendimento. Um CI envolve uma grande variedade de elementos - textos, planilhas, apresentações, dados geográficos (mapas), mídia contínua (áudio e vídeo), imagens (fotos), entre outros - que referenciaremos aqui como documentos. A principal atribuição de um centro de informações é a organização destes documentos com o objetivo de atender a preservação digital e registrar a história do empreendimento.

Um centro de informações consolida o conteúdo indicado por entidades produtoras, garante sua preservação e fornece acesso para que consumidores o consultem. A Figura 43 representa o modelo funcional adaptado da ISO 14721:2003 (seção 2.4.2.1), para sistemas abertos de arquivamento de informação, que pode ser adotado também como ilustração para as macro funções de um centro de informações.

Descrevemos agora sucintamente as macro funções da Figura 43. A Admissão: oferece serviços e processos necessários para aceitar os conteúdos indicados pelos produtores. O Arquivamento (*archival*) é responsável por arquivar, manter e recuperar os conteúdos que passaram pela Admissão. O Gerenciamento de dados (*storage*) suporta a inclusão, manutenção e acesso de informações descritivas, que identificam e documentam os conteúdos, bem como os dados propriamente ditos. A Administração do sistema opera o sistema como um todo. O Planejamento da preservação gerencia e monitora o ambiente, registrando as mudanças, com objetivo de assegurar que a história seja preservada no longo prazo. Por fim, a função de Acesso apóia os consumidores na determinação da existência, descrição e localização dos dados armazenados.

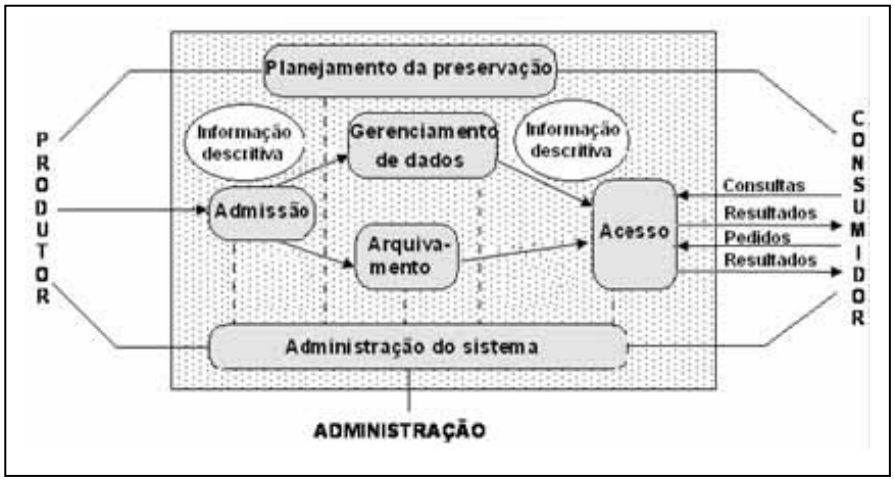


Figura 43: Modelo Funcional de um Centro de Informações adaptado da ISO 14721:2003

O foco de nosso estudo está na função de preservação mais especificamente, na proveniência, que a ISO 14721:2003 (seção 2.4.2.1) considera como parte da informação de descrição de preservação. A informação de descrição de preservação é composta por informação de referência, de proveniência, de contexto e de fixidez (*fixidity*) (veja detalhes na seção 2.4.2.1).

A proveniência definida pela informação de descrição de preservação da ISO 14721:2003 é a informação que documenta o histórico de uma informação de conteúdo: relato da origem ou da fonte da informação de conteúdo e mudanças de custódia desde a sua produção. Lembramos que a ISO 14721:2003, em muitos casos, utiliza o termo “informação de proveniência” quando o termo que consideramos correto seria “dado de proveniência”.

A proveniência a que se refere o modelo conceitual de proveniência (seção 3.5) é a de um conjunto de descrições de qualquer mudança de um objeto de informação que registre sua história e que seja significante para resguardar autenticidade, integridade e interpretação.

A definição de proveniência do modelo de proveniência é mais ampla porque não se restringe apenas à custódia. Todos os fatos geradores que representam a história como um conjunto de mudanças e suas respectivas descrições são potenciais eventos a serem representados por um CI. Assim, prosseguimos como nas outras aplicações do modelo conceitual de proveniência descritas ao longo deste capítulo e, identificamos os eventos de interesse de um centro de informações.

O centro de informações de um empreendimento é construído em paralelo a diversos eventos que devem ser registrados. Elegemos como exemplo, os

eventos que simbolizam a história da construção civil do próprio centro de informações. Considere também que, a cada evento descrito, há diferentes documentos associados e que a produção de cada documento associado ao evento pode ser interpretada como uma atividade. No exemplo que descreveremos a seguir, não estamos preocupados em definir como será o acesso a estes metadados.

Começamos pelo mapeamento de conceitos para descrever eventos e documentos. As instâncias da classe E5 Event seriam os eventos e as instâncias da classe E7 Activity as atividades que produziram algum documento.

Imagine que no dia 14 de junho de 2007 (E52 Time-Span), os moradores da região (CID2 Participant) como expectadores (P14.1 in the role of) participaram juntamente com o “Sr. Fulano de Tal” (CID2 Participant) presidente do centro de informações (P14.1 in the role of) do “Lançamento da Pedra Fundamental do CI” (E5 Event).

Continuando a construção de nosso exemplo, o projetista deste domínio poderia julgar necessária a representação da herança cultural. Neste caso, sugerimos a expansão CH (*Cultural Heritage*) que oferece classes que capturam conceitos importantes para representar a história de uma obra, expressão, manifestação e veículos de persistência. Considere agora três atividades (E7 Activity) que produziram cada uma um documento (E84 Information Carrier) diferente: “foto.jpg”, “recibo de entrega da lápide.doc” e “nota de divulgação para imprensa.pdf.” Essas três atividades, em um arranjo (CID5 Means), poderiam ser uma descrição, preliminar, de como o evento “Lançamento da Pedra Fundamental” (E5 Event) ocorreu, complementar às informações sobre data, local e participantes. A lápide (F5 Item) de polipropileno (F3 Manifestation Product Type) poderia conter um trecho em português (F2 Expression) da obra (F1 Work) de Guimarães Rosa.

Cada atividade, da mesma forma que o evento “Lançamento da Pedra Fundamental”, poderia ter um detalhamento, indicando entre outros, onde, quando e quem. Por exemplo, a foto foi tirada pelo satélite brasileiro “ABC”[51] (CID2 Participant), nas coordenadas geográficas “XYZ” (E53 Place) às 13h em 01/01/2007 (E52 Time-Span).

[51] ABC é uma instância por exemplo, de E41 Appellation.

Por fim, imagine outros dois documentos (E84 Information Carrier), o vídeo de lançamento do empreendimento e a mapa do terreno adquirido, que poderiam ser respectivamente a motivação (P17 was motivated by) e influência (P15 was influenced by) para o evento de Lançamento da Pedra Fundamental.

Considere agora que tenhamos também descrições semelhantes para o evento "Início da Construção do Centro de Informações". Entre este último e o evento de "Lançamento da Pedra Fundamental" ocorreram outros eventos que, se devidamente registrados e relacionados, formariam uma rede de conhecimento que descreve o empreendimento que poderia ser representada a partir do modelo conceitual de proveniência. Para isso, recomendamos ao projetista que avalie se as expansões PO (seção 3.5.2.1) ou RQR (seção 3.5.2.2.1) atendem completamente a representação de uma rede de conhecimento. A capacidade de realizar consultas a essa rede e recuperar a proveniência dos eventos nos permitiria avaliar sua história.

Podemos imaginar que a interface de um centro de informações com os consumidores seja uma ferramenta de portal, por exemplo, o Liferay[52]. O serviço de proveniência do portal poderia ser dividido entre dois *portlets*[53]: *portlet* de coleta e *portlet* de consulta, que desempenhariam respectivamente funções análogas ao *provenance wrapper* e *provenance filter* da Figura 39 (seção 4.2). A partir da especificação JSR 286[54] que define, entre outras regras, as de comunicação entre *portlets* e está disponível como minuta, desde 13/12/2007. Qualquer *portlet* desenvolvido de acordo com essa especificação poderia então se registrar junto aos *portlets* de proveniência para a troca de dados (coleta e consulta de proveniência). Por exemplo, um *portlet* de *upload* de documentos poderia repassar os valores manualmente coletados no formulário, para que o *portlet* de proveniência registrasse esses dados como respectivas instâncias das classes do modelo, sendo o evento de criação de documento no portal uma instância da classe E7 Activity. Por outro lado, um *portlet* de consulta poderia utilizar a API Lucene[55] para recuperar dados dos índices de conteúdo e metadados e apresentar os resultados consolidados no *portlet* de consulta.

[52] http://www.liferay.com

[53] Um Portlet pode ser entendido como um "Servlet Visual" independente que pode ser utilizado para disponibilizar informações dentro de uma página Web.

[54] http://jcp.org/en/jsr/detail?id=286

[55] http://lucene.apache.org/

Concluímos ressaltando então que, do ponto de vista de preservação digital, um centro de informações poderia adotar o padrão ISO 14721:2003 como especificação de serviços e processos cobrindo as suas principais funcionalidades. De forma complementar, do ponto de vista de metadados, poderia adotar o padrão de projeto para proveniência que está essencialmente fundamentado pela ISO 21127:2006.

Conceitualmente, a adoção de um modelo de proveniência permite que o centro de informação seja estruturado para oferecer uma navegação multifacetada: eventos, agentes, conceitos etc. Imagine um centro de informações que possui uma galeria, onde os quadros são monitores de alta definição. Considere também que a definição da história a ser contada para os visitantes dessa galeria fosse guiada, não necessariamente associada a uma ordem cronológica, e visualizada através destes “quadros”. A história poderia ser contada de diferentes formas, dependendo do arranjo (*partial order*) “estipulado” previamente ao início do *tour*. Os eventos estariam associados de forma a satisfazer as expectativas do público visitante.

Por fim, sugerimos o modelo conceitual de proveniência como a base para esta visão porque está centrado em eventos e relacionamentos e tem recursos associados a eventos. Avançaremos adiante e faremos um estudo mais amplo do modelo de proveniência, utilizando uma ferramenta do domínio de gerenciamento de configuração de software (seca 4.5).

## 4.5. Serviço de Proveniência na Web para ferramenta de Gerenciamento de Configuração de Software

Descrevemos aqui a aplicação do padrão de projeto de proveniência, utilizando como motivação um serviço de proveniência na Web (*Web Provenance Service* ou abreviadamente WPS). O objetivo deste serviço é disponibilizar via Web os dados de proveniência de um banco de dados modelado a partir do padrão de projeto para proveniência para o domínio de gerenciamento de configuração de software. Iniciamos, descrevendo a especificação (seção 4.5.1), em seguida, detalhamos o ambiente (seção 4.5.2) utilizado para o desenvolvimento do protótipo, e por fim apresentamos o estudo da ferramenta Trac (seção 4.5.3) que inclui a instanciação do modelo de proveniência.

### 4.5.1. Especificação do Serviço

Por limitações de tempo, o serviço de proveniência na Web contempla apenas a recuperação da proveniência, não tratando de sua coleta automática ou manual. O objetivo de um WPS (*Web Provenance Service*) é fornecer aos usuários a possibilidade de disponibilizar acesso aos dados de proveniência armazenadas em um determinado servidor de banco de dados relacional através de requisições HTTP a um servidor de aplicações. Através deste serviço os dados de uma determinada aplicação são disponibilizados através da Web sem a necessidade do cliente conhecer os detalhes de armazenamento do banco de dados acessado.

Os usuários de um serviço WPS são seres humanos, sistemas, agentes de software e outros clientes que acessem o serviço utilizando requisições HTTP. O WPS faz interação com os sistemas clientes (usuários) do serviço de proveniência.



#### 4.5.1.1. Requisitos

Um WPS possui os requisitos:

1. Atender requisições HTTP.
2. Acessar o banco de dados através dos parâmetros em um arquivo de configuração.
3. A tela inicial com as operações do WPS corresponde ao resultado da consulta a capacidade do serviço WPS e deve estar em formato XML.
4. Permitir a iteração com usuário através da URL.
5. Ler as consultas realizadas as fontes de dados a partir de arquivos de consultas. Cada tipo de consulta tem seu respectivo arquivo ou conjunto de arquivos.
6. Converter resultados para XML utilizando extensão da linguagem SQL, minimizar o acoplamento entre o código de conversão do *resultSet* e o código do sistema.
7. Disponibilizar o Serviço de Proveniência na Web em um servidor de aplicações.
8. Listar e descrever quais operações e respectivos parâmetros o serviço disponibiliza, apresentando o resultado em formato XML.

9. Retornar os dados de proveniência em formato XML a partir da identificação global de uma entidade. A identificação global deve ser única.
10. Servir uma requisição retornando as entidades correspondentes de acordo com o tipo da entidade, tipo de relacionamento e palavra de referência requisitada, limitando o resultado formatado em XML a um número máximo de objetos, identificado também como parte da requisição.

#### 4.5.1.2. Descrição das Operações

Um WPS oferece três operações básicas para atender aos requisitos (seção 4.5.1.1). As operações de um WPS são um subconjunto das operações disponíveis em um Web Object Service (WOS) (Vretanos, 2003). Neste projeto não estaremos oferecendo operações para transação.

As operações são consultas Web que o serviço WPS é capaz de responder. Uma consulta Web (requisição) utiliza o protocolo HTTP para atender as requisições. O WPS aceita requisições através dos métodos GET e POST. O prefixo da requisição é definido como uma string incluindo o protocolo, nome do servidor, número da porta, caminho, um ponto de interrogação '?', e, opcionalmente, um ou mais parâmetros específicos do serviço concatenados com o símbolo '&'.

A sintaxe da consulta Web é uma URL genérica que representa uma requisição HTTP. Neste projeto essa requisição é feita ao WPS.

http://servidor[:porta]/caminho?{nome[=valor]&}[56]

O cliente adiciona os parâmetros de requisição necessários, respeitando a multiplicidade, e para cada novo parâmetro que deseja incluir na consulta utiliza a sintaxe "nome=valor&", onde nome representa o nome do parâmetro e valor o seu valor propriamente dito.

[56] colchetes [ ] denotam zero ou um ocorrência e chaves { } denotam zero ou mais ocorrências.

O WPS aceita requisições com parâmetros específicos a cada operação. O único parâmetro comum a todas as operações é o parâmetro REQUEST que pode assumir um dos seguintes valores: *getCapabilities*, *getObjectById* ou *getObject* que equivalem exatamente às operações que a OGC[57] (*Open Geospatial Consortium*) implementa para vários dos seus serviços. Este parâmetro é *case sensitive* (maiúsculas e minúsculas são interpretadas de forma diferente). O serviço WPS retorna as respostas às requisições HTTP com o tipo MIME (*Multipurpose Internet Mail Extensions*) definido como text/xml. As descrições de cada operação de REQUEST são:

- ***getCapabilities*:** descrever as suas capacidades. Especificamente, ele deve ser capaz de indicar os tipos de objetos que pode servir e quais operações e respectivos parâmetros são suportados. Esta operação é responsável por servir uma requisição HTTP retornando as capacidades do serviço e apresentando o resultado no formato XML;
- ***getObjectById*:** retornar uma única instância de um objeto da fonte dados a partir de um identificador global. Esta operação é responsável por servir uma requisição HTTP retornando a proveniência do objeto identificado, apresentando o resultado no formato XML;
- ***getObject*:** suportar consultas mais complicadas que envolvam o uso de parâmetros como *typeName*, *relationToType* e *showReference*. Esta operação é responsável por servir uma requisição HTTP retornando as instâncias correspondentes aos valores dos parâmetros requisitados, apresentando o resultado no formato XML.



#### 4.5.1.3. Arquitetura

A arquitetura do sistema está divida em três camadas horizontais identificadas pelas linhas tracejadas da Figura 44. A camada superior representa a Web, a camada intermediária o serviço propriamente dito e por fim a camada inferior simboliza a fonte de dados que desejamos compartilhar na Web.

[57] http://www.opengeospatial.org

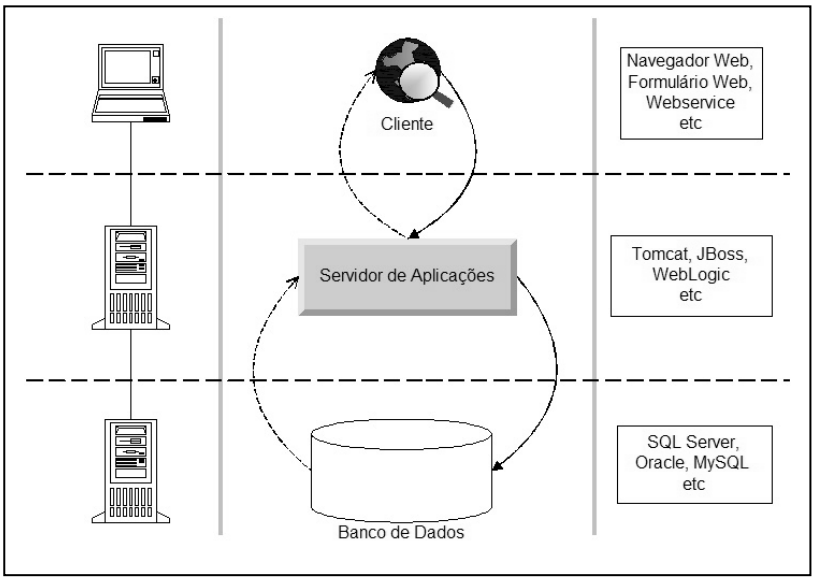


Figura 44: Arquitetura do *Web Provenance Service* (WPS)

Entre as duas linhas sólidas verticais, temos a arquitetura propriamente dita onde as setas sólidas representam requisições (*request*) e as setas pontilhadas as respostas (*response*). Os ícones à esquerda da arquitetura são exemplos de instanciação de hardware e à direita listamos alguns exemplos de instanciação de software.

### 4.5.2. Ambiente de Desenvolvimento

O ambiente de desenvolvimento está consolidado em uma única máquina virtual. Utilizamos o VMWare para a configuração de um servidor com sistema operacional Windows 2003 Server, interface de programação MyEclipse 5.5 e interface para abstração de consultas SQL Hummingbird BI v8.1. Para prototipar o serviço de proveniência, temos ainda as seguintes instanciações das tecnologias da arquitetura: Navegador Web - Internet Explorer 7.0, Servidor de Aplicações - Tomcat 5.5.23 e Servidor de Banco de Dados - MS SQL Server 2000.

#### 4.5.2.1. Catálogo TDK

Um catálogo é uma coleção de dados descritivos (metadados) a respeito dos dados (objetos) armazenados em um banco de dados. Os metadados atuam como propriedades que podem ser consultadas e requisitadas através do serviço

de catálogo para avaliação dos recursos fornecidos por um banco de dados ou serviço.

O Catálogo TDK oferece serviços de descoberta, acesso e gerenciamento que permitem ao cliente respectivamente: localizar os metadados que descrevem os dados, acessar métodos para requisição de serviços sobre os dados e acessar métodos para alterações dos metadados do catálogo.

Elegemos o esquema do Catálogo TDK como base para a construção do modelo físico. A escolha foi importante porque permitiu uma abstração ainda maior para o modelo de proveniência porque possibilita que o modelo de proveniência seja composto de diferentes *views* criadas a partir do modelo adaptado do Catálogo TDK.

Como o esquema original do Catálogo TDK não possui em suas tabelas a possibilidade de representação de relacionamentos entre entidades, algumas adaptações foram realizadas (seção 4.5.2.2). Além disso, há várias tabelas para controle de acesso e outras funções que não foram relevantes para este projeto e que não estão descritas neste documento. Selecionamos um conjunto reduzido de tabelas do esquema original, definindo-o como esquema mínimo do Catálogo TDK (seção 4.5.2.2).



### 4.5.2.2. Modelo Físico construído por Reuso e Adaptação

Nomeamos o modelo físico de proveniência de W6H2. (W6H correspondem aos conceitos abstratos utilizados para a construção do modelo conceitual e o número 2, lemos como “*to*”, simbolizando a importância dos relacionamentos entre eventos, entre recursos e entre ambos).

Como em nosso modelo de proveniência (mínimo) (seção 3.5) só possuímos relacionamentos binários, é possível utilizar um esquema reduzido do Catálogo TDK desde que sejam criadas as respectivas tabelas para representar o auto-relacionamento entre as diferentes entidades do modelo. Este auto-relacionamento permite a construção de entidades complexas e bem como a qualificação de outros relacionamentos.

Assim, o esquema do banco de dados implementado neste projeto corresponde ao da Figura 45, que tem circulado com elipses as tabelas adicionais acrescentadas ao esquema do Catálogo TDK, de forma a comportar os relacionamentos entre entidades.

As tabelas que de fato foram utilizadas para popular o banco de dados foram:

- TDK_CAT_ENTITY: armazena os atributos fixos de uma entidade.
- TDK_CAT_TYPE: armazena o tipo de uma entidade.

Todas as classes do modelo de proveniência estão mapeadas na tabela TDK_CAT_ENTITY, sendo que seus respectivos tipos (E5 Event, E7 Activity, CID2 Participant etc) estão representados pela tabela TDK_CAT_TYPE. Por fim, a tabela TDK_CAT_FILE referencia os itens físicos propriamente ditos de uma instância de E84 Information Carrier (documentos digitais) que por sua vez estão relacionados diretamente a uma entidade de proveniência.

Os relacionamentos existentes entre as classes do modelo conceitual de proveniência serão representados utilizando as tabelas que foram adicionadas ao esquema do Catálogo TDK:

- W6H2_TDK_CAT_RELATIONS: armazena os relacionamentos entre entidades.
- W6H2_TDK_CAT_RELATION_TYPE: armazena o tipo de relacionamento entre entidades.

Essas tabelas estão identificadas com elipses na Figura 45 que foi gerada pelo diagrama de esquema do banco de dados TDKSimple, utilizando a ferramenta *Enterprise Manager* do Microsoft SQL Server 2000.

Todas as propriedades das classes do modelo conceitual de proveniência foram completamente mapeadas nos atributos disponíveis nas classes descritas nesta seção. Em nossa instanciação, por simplificação, algumas das tabelas do esquema mínimo do Catálogo TDK não foram necessárias, por exemplo, a representação da autoridade criadora de cada entidade (tabelas TDK_CAT_ENTITY_AUTHOR). Existem ainda outras tabelas presentes no esquema original do banco de dados do Catálogo TDK que também não foram utilizadas, mas são essenciais ao modelo físico do TDK porque flexibilizam a extensão do modelo físico a partir da adição de novos atributos a qualquer entidade:

- TDK_CAT_ATTRIBUTES: armazena os atributos estendidos de uma entidade.
- TDK_CAT_ENTITY_ATTRIBUTES: associa entidades a atributos estendidos.

- TDK_CAT_ATTRIBUTE_TYPE: armazena o tipo de atributo
- TDK_CAT_ATTRIBUTE_CATEGORY: define categorias que têm como propósito agrupar os atributos de entidades.
- TDK_CAT_AUTHOR: armazena os autores da entidade
- TDK_CAT_ENTITY_AUTHOR: associa entidades a autores

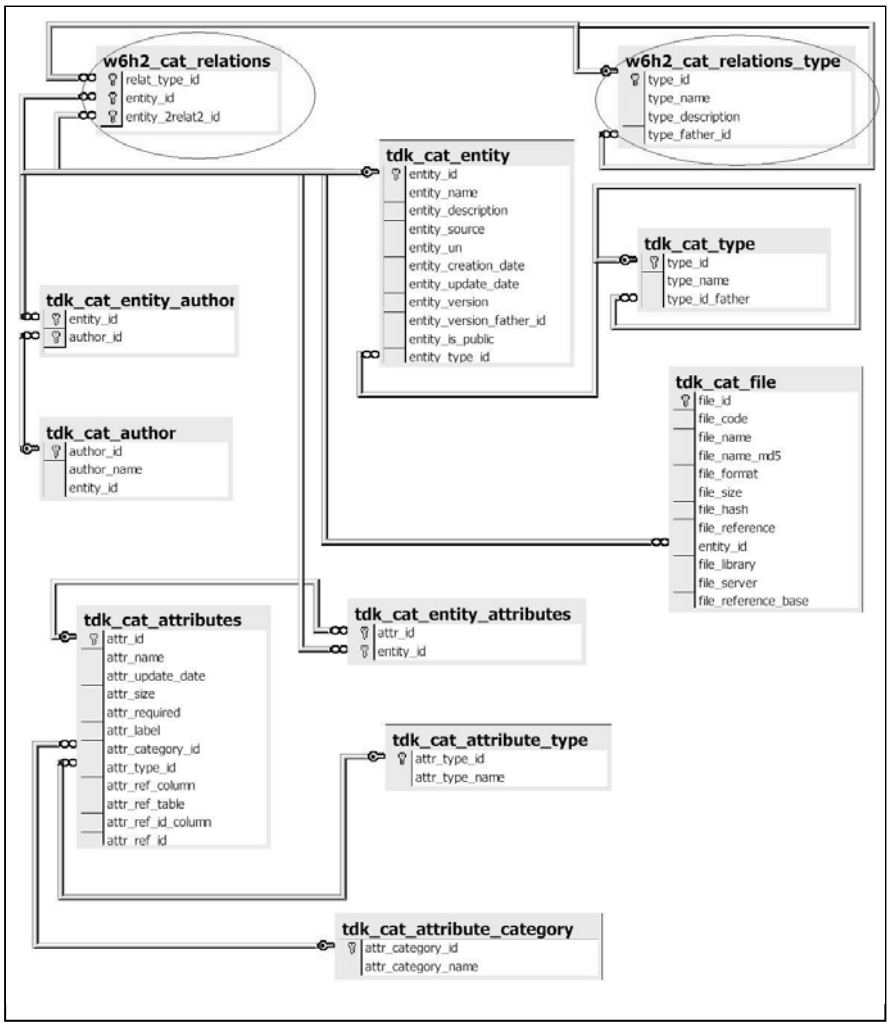


Figura 45: Principais Tabelas do Catálogo TDK Adaptado

Há apenas uma tabela existente originalmente no esquema mínimo do Catálogo TDK que optamos por utilizar em nossa aplicação, como forma de representar documentos (*files*) que podem estar associados a qualquer entidade do modelo. A tabela TDK_CAT_FILE armazena os documentos associados a uma determinada entidade. Poderíamos realizar uma engenharia reversa e, imaginar que esta tabela do modelo físico seria equivalente no modelo conceitual a uma expansão que importa a classe E84 Information Carrier da ISO 21127

(2006) como especialização indireta de CID2 Participant. Por fim, esta tabela está associada à tabela TDK_CAT_ENTITY o que implicaria acrescentar a esta expansão uma propriedade a classe CID1 Provenance Entity que poderia ser "PID14 has file reference (is file reference to)" que tem classe-imagem E84 Information Carrier.

#### 4.5.2.3. Abstração para o Modelo Físico - Ferramenta BI

A ferramenta Hummingbird[58] BI (*Business Intelligence*) também foi utilizada para a construção e validação das consultas SQL. Incluímos na abstração da Figura 46 apenas as tabelas que serão utilizadas por este projeto. No projeto foi implantada a opção de consulta apenas em uma direção no relacionamento entre entidades, mas a representação da inversa está disponível no esquema adaptado para ser explorada em trabalhos futuros. No caso da recuperação também da inversa, são necessárias duas consultas a fonte de dados para recuperar o relacionamento em ambas as direções.

O estudo da ferramenta Trac (seção 4.5.3) também fez uso da modelagem BI para a compreensão de seu banco de dados (seção 4.5.3.1), a partir da análise de consultas SQL.



[58] http://www.hummingbird.com

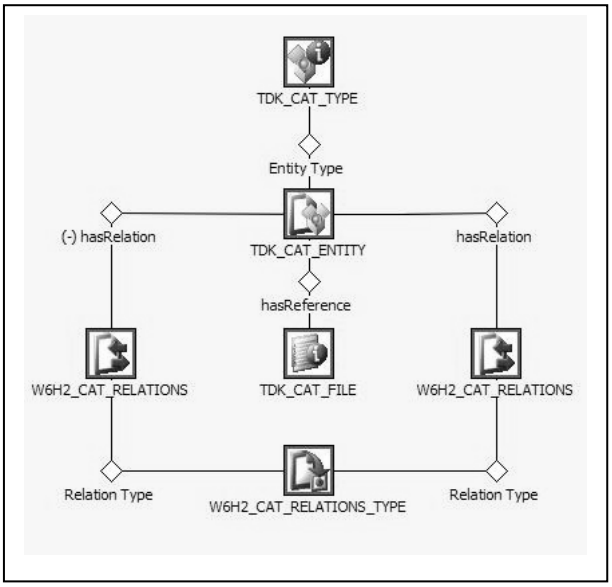


Figura 46: Abstração BI para o modelo físico



## 4.5.3.
## Estudo da Ferramenta Trac

### 4.5.3.1.
### Preliminares

A instanciação de nosso modelo conceitual foi feita para o domínio GCS (Gerenciamento de Configuração de Software). Escolhemos o domínio GCS ou do inglês SCM (*Software Configuration Management*) porque em nosso levantamento de oportunidades de acesso a dados reais, teríamos contato com a equipe do laboratório TecGraf da PUC-Rio responsável pelo Projeto TDK Catalog[59].

Este projeto tem sua gestão baseada na ferramenta Trac[60], que é um Wiki estendido que oferece um sistema de controle de ocorrências e também provê uma interface de consulta ao Subversion[61]. O Trac está integrado ao sistema de controle de versões, possibilitando a criação de links entre defeitos, tarefas, *Changesets*, anexos (*attachments*) e páginas Wiki.

[59] http://www.tecgraf.puc-rio.br/tdk/

[60] http://trac.edgewall.org/

[61] http://subversion.tigris.org/

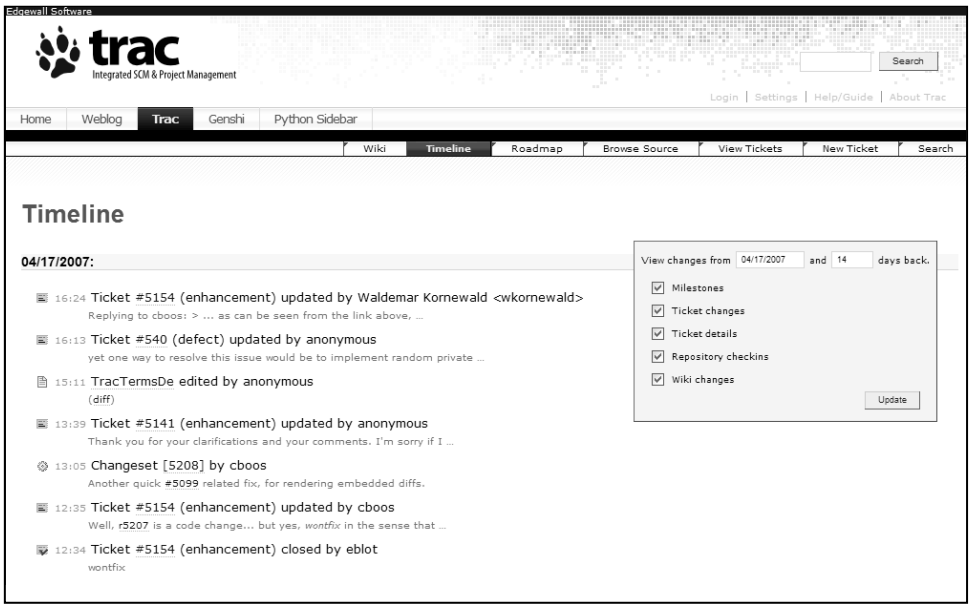


Figura 47: Tela de Entrada do Trac

O Trac possui a noção de *Timeline* (Figura 47) onde estão listados vários tipos de eventos em ordem cronológica facilitando uma visão geral do projeto. Para que o mapeamento do domínio dessa ferramenta fosse possível precisávamos entender quais dados estavam disponíveis, como estavam organizados, como acessá-los e de que forma utilizá-los. Aprofundando essa visão geral, estudamos o banco de dados original utilizado pelo Trac (SQLite[62]). Neste projeto foi necessário realizar uma transformação dos dados originais (do inglês DTS - *Data Transformation Task*) para que todos os dados do banco original estivessem disponíveis em um servidor com banco de dados SQL Server 2000.

Adicionalmente, adotamos a ferramenta BI Query da Hummingbird para aprendizado do conteúdo do banco de dados. A Figura 48 ilustra o modelo BI construído com essa ferramenta para o banco de dados do Trac já convertido para o servidor SQL.

[62] http://www.sqlite.org/

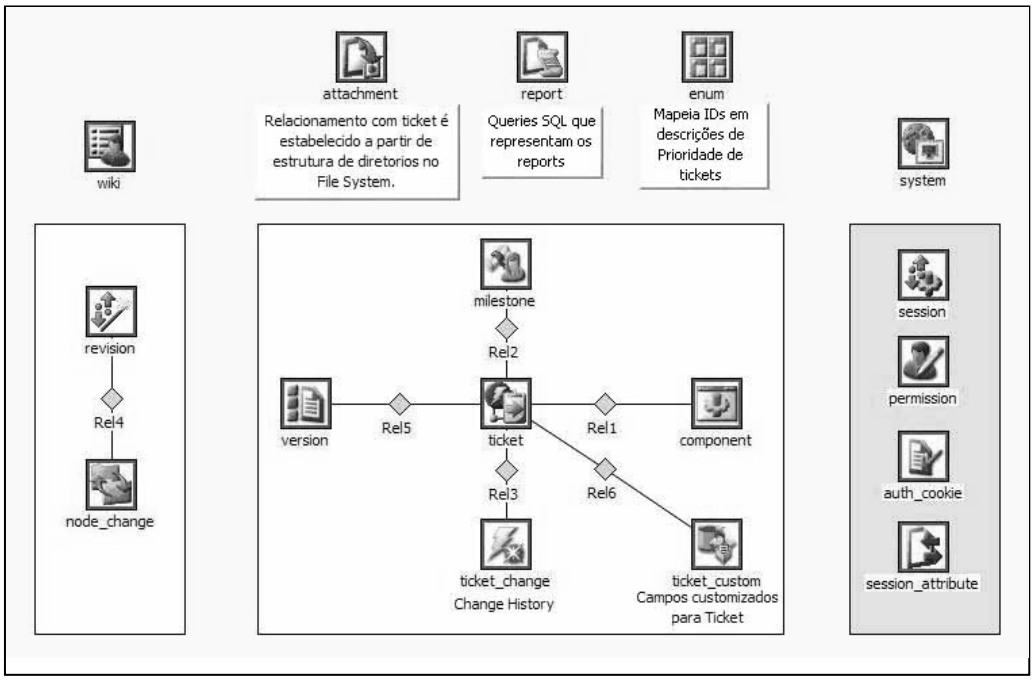


Figura 48: Abstração BI para o esquema do banco de dados do Trac

As tabelas REVISION e NODE_CHANGE contêm dados que vêm diretamente do *Subversion* (por exemplo, artefatos alterados em um determinado *ChangeSet*), enquanto que as tabelas que tem relacionamento com a tabela TICKET e as tabelas WIKI, ATTACHMENT, REPORT e ENUM representam os dados relativos à ferramenta Trac propriamente dita.

Após o estudo do banco, procuramos identificar características específicas que demonstram a necessidade de rastreabilidade dos dados atendida parcialmente pela ferramenta Trac. Destacamos que a ferramenta Trac oferece uma forma de relacionamentos entre *ChangeSets* que pode ser entendida através da Figura 49 que ilustra o *plugin* RevTree. Este *plugin* permite uma visualização gráfica dos relacionamentos. Nessa figura temos diferentes ramos de versionamento armazenados no Subversion que são identificados ao topo da figura. As setas ligam um *ChangeSet* a outro verticalmente indicando de onde foi originado, e horizontalmente indicando uma eventual mudança de ramo.

Além dessas informações disponibilizadas pelo *plugin* RevTree e do *Timeline*, a ferramenta Trac oferece alguns relatórios que permitem a

identificação de estado das ocorrências (*tickets*) e algumas outras consultas para acompanhamento do projeto, mas não uma visão gerencial consolidada do projeto. Durante o estudo do esquema do banco de dados do Trac, também foi consultado o projeto DrProject[63], que na verdade é uma extensão da ferramenta Trac habilitando a capacidade de multi-projetos e também oferecendo suporte a lista de discussões.

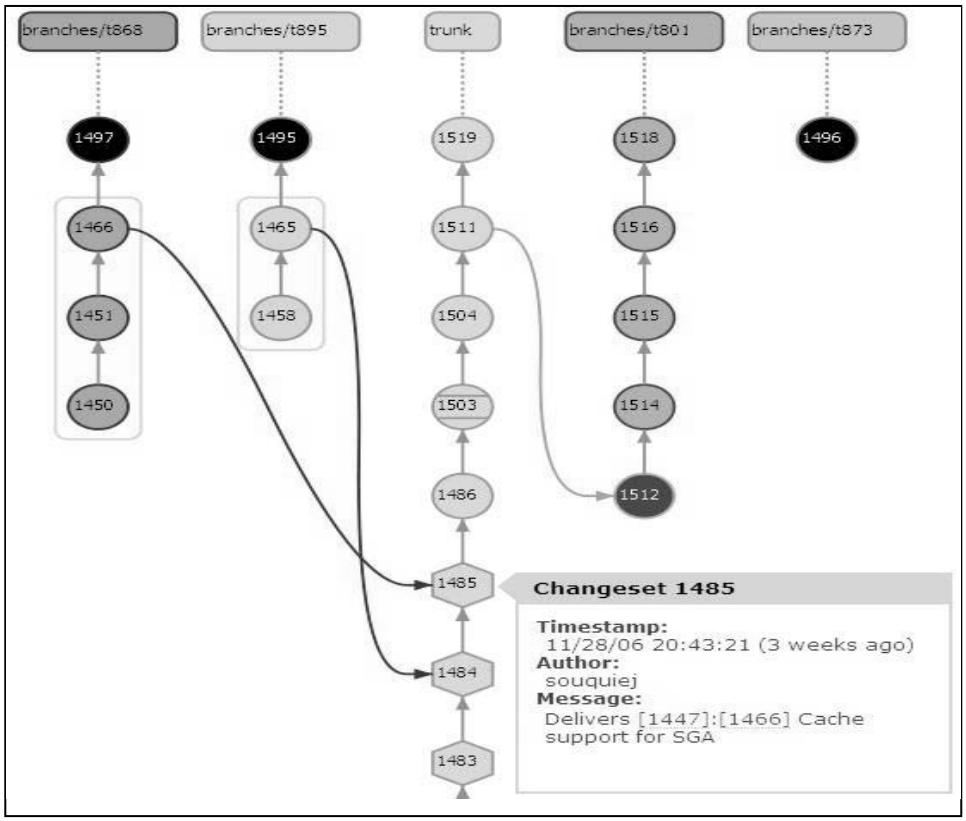


Figura 49: Rastreabilidade de *ChangeSets*

A Figura 50 é um exemplo de dados sintéticos acrescentados a base do Trac. Ao topo há o *ticket* #20 com status "*closed*" e ao final dessa figura há uma referência ao *Changeset* "1129" correspondente a sua conclusão.

[63] Projeto da Universidade de Toronto: https://www.drproject.org/

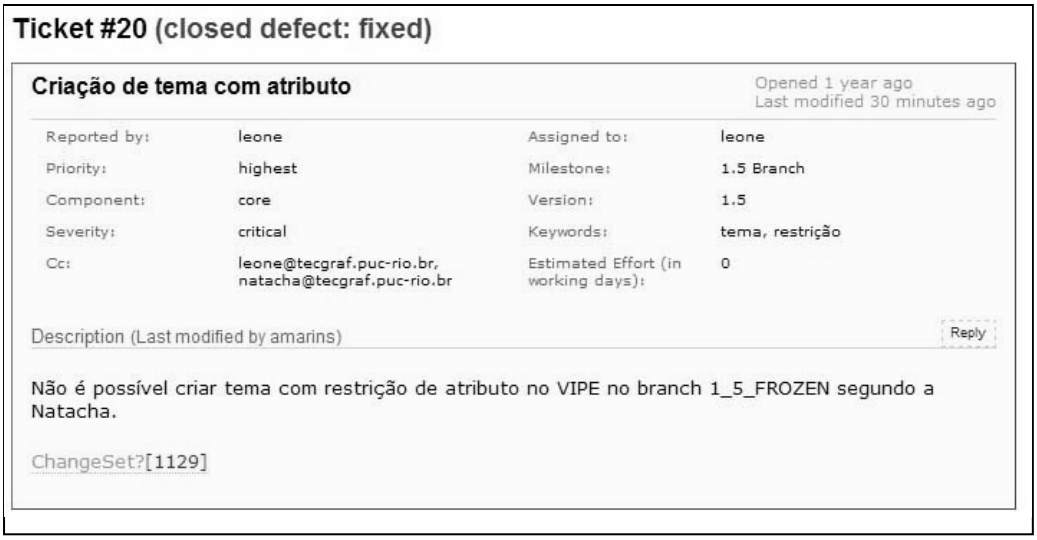


Figura 50: Referências cruzadas entre *Ticket* e *ChangeSet* apresentada no *Ticket*

A Figura 51 destaca da lista de eventos registrados no *Timeline*, o evento que corresponde ao *Changeset* "1129" que foi responsável pela conclusão do *ticket* #20. A identificação que o ticket está concluído é representada pelos caracteres tachados "~~#20".~~


Figura 51: Referências cruzadas entre ChangeSet e Ticket apresentada no Timeline

O *ChangeSet* é o identificador de um evento. A ligação entre *ChangeSets* simboliza o encadeamento de eventos. Interessa-nos validar este encadeamento para os *milestones*, que também são eventos, de forma que aos dados de proveniência dos *milestones* agreguem valor à gestão do projeto.

Os dados do Trac utilizados para o Serviço de Proveniência na Web correspondem a um subconjunto dos dados reais do projeto TDK Catalog desenvolvido pelo laboratório Tecgraf. Algumas conversões e ajustes foram feitos inserindo alguns dados sintéticos de forma a completar os dados reais uma vez que não havia referências organizadas entre *Changesets* que concluíam *tickets* em aberto.

Agora, apresentamos o mapeamento dos conceitos do Trac em conceitos do modelo de proveniência (seção 4.5.3.2).

### 4.5.3.2. Mapeamento para o Modelo de Proveniência

Como o modelo de proveniência é centrado em eventos, era necessário identificá-los, bem como seus níveis de granularidade. Identificamos dois importantes tipos de eventos: *Milestone e Commit*, sendo o primeiro um tipo de evento do Trac e o segundo um tipo de evento do Subversion. O conceito de *Milestone* tipicamente representa um evento, de duração zero, importante em um projeto, como a conclusão de um componente, uma fase ou a criação de um protótipo entre outros. Já o conceito *Commit* é representado por um número: *ChangeSet,* e simboliza um ou mais artefatos criados, alterados ou excluídos.

Como Trac e Subversion trabalham de forma integrada, inicialmente, analisamos o mapeamento desses dois conceitos de eventos, com o objetivo de explorar os conceitos relevantes para validar a cobertura do modelo de proveniência. Essa integração permite o desenvolvimento de *hooks* que podem ser ativados antes ou após a execução de um evento de *Commit* do Subversion. Um *hook* pode ser entendido como um *script* que poderia, entre outras ações de controle, por exemplo, concluir um *ticket* atualizando seu status para fixado. Complementando a direção inversa, um usuário, ao fechar um *ticket* pela interface do Trac, poderia referenciar o *Changeset* que o motivou a concluí-lo. Os comentários dos *Commits*, as descrições de *tickets* relacionadas à troca de status, por exemplo, de “novo ticket” para “ticket concluído”, ou até textos do Wiki do Trac referentes a algum *ticket,* poderiam definir a necessidade da criação de especializações para a classe E38 Conceptual Object, de forma que esses dados fossem representados estruturadamente.

Aprofundando, um *hook* poderia ser definido para a captura manual da proveniência, a partir de propriedades obrigatórias a serem preenchidas durante um evento de *Commit*. Outra informação interessante que um *hook* deste tipo poderia capturar seria o nome do *desktop* do responsável pelo *Commit,* permitindo com isto a coleta automática da proveniência de onde (o nome do *desktop* que possui o espelhamento do respectivo ramo do Subversion) foi realizado o *Commit*, registrando o dado de proveniência como instância da classe E53 Place. Isto poderia ajudar no acompanhamento da distribuição dos artefatos fora do repositório central.

Neste momento, interessa-nos explorar a aplicação do modelo de proveniência a partir de sua especialização. Com isso, utilizamos uma analogia com os principais conceitos do domínio do Trac produzindo o mapeamento da Tabela 23. Ressaltamos que o uso dos conceitos abstratos (W*h-questions*), adotados pela metodologia de construção do modelo, é também adotado na especialização do modelo para este domínio.

Nossa instanciação foca o conceito de *Milestone* como os eventos que nos interessam. No caso do Trac, um *milestone*[64] é um conjunto de *tickets*. Os *milestones* não estão relacionados uns com os outros explicitamente e tem apenas uma descrição e uma data de expiração (*Due Date*). No modelo mínimo, instâncias da classe CID4 Reason poderiam ter o significado de um projeto como sendo a motivação mais geral para o porquê (*Why)* dos respectivos *milestones.* O Trac, a partir da adoção da expansão TR (*Traceability*) poderia oferecer a rastreabilidade de *tickets*. Outra alternativa possível seria a adaptação do banco de dados da ferramenta, incluindo outras tabelas relacionadas à tabela *Ticket* para a representação de auto-relacionamentos qualificados.

Tabela 23: Mapeamento entre conceitos do Trac em classes do modelo de proveniência

| **Conceito do Trac** | **Classe do Modelo de Proveniência** | **Conceito Abstrato** |
|---|---|---|
| *Milestone* | E5 Event | *What* |
| - | E53 Place | *Where* |
| *Milestone Due Date* | E52 Time-Span | *When* |
| *Ticket* | E7 Activity | - |
| *Ticket Owner* | CID2 Participant | *Who* |
| Comentários de *Commits* do SVN ou justificativas de fechamento de tickets | E28 Conceptual Object | - |

A ferramenta Trac poderia também representar de forma explícita, a noção de explicação (CID5 Means) como uma ordem parcial de *tickets* (E7 Activity) que permite compreender como o *milestone* (E5 Event) foi alcançado. Essa

[64] Um *milestone* é um evento que não tem duração e simboliza um marco importante no curso de um projeto.

representação poderia se capturada pela expansão PO (seção 3.5.2.1) através da classe CID6 PO Event.

Outra perspectiva é que, em sua configuração padrão, a ferramenta Trac não permite limitar o papel a ser executado dentro de um *ticket*. Essa funcionalidade é interessante do ponto de vista gerencial, para evitar que um recurso no papel de desenvolvedor, altere inadvertidamente um *ticket* de relatório gerencial onde apenas uma pessoa no papel de coordenador de projeto poderia realizar tal atividade. A ferramenta Trac não possui o conceito de papel (*Role*). Por isso, poderia ser aprimorada para oferecer a representação de papéis desempenhados por seus usuários (CID2 Participant). O conceito *Role* (papel) poderia ser tão simples quanto um papel estático que um participante rotineiramente desempenha (*User Description*) ou mais sofisticado se o mesmo participante desempenhar papéis dinâmicos dentro de um mesmo *milestone*.

Por fim, acrescentamos que outros conceitos do Trac como *Component, Priorities, Severities* e *Versions* poderiam ser mapeados em instâncias ou especializações da classe E28 Conceptual Object. O conceito de tipos de *tickets* (*Ticket Types*) pode ser mapeado para a classe E55 Type.

Realizamos então alguns exemplos motivados pelos mapeamentos indicados na Tabela 23. Cadastramos manualmente alguns valores, do projeto TDK, gerenciado pela ferramenta Trac, no banco de dados criado a partir do modelo conceitual de proveniência. Em seguida, capturamos e apresentamos as respostas às requisições de operações (seção 4.5.1.2) do serviço de proveniência na Web.



#### 4.5.3.3 Respostas do Serviço de Proveniência na Web

Por limitações de tempo, não exploramos o serviço de proveniência na Web como um *plugin* para o Trac ou Subversion. Seria necessário primeiro desenvolver uma coleta automática da proveniência, que provavelmente também seria outro *plugin* para o Trac ou *hook* para o Subversion. O serviço de proveniência na Web foi alimentado manualmente com os dados do projeto TDK extraídos do Trac. O estudo desenvolvido teve como objetivo uma validação inicial do modelo aplicado ao domínio de gerenciamento de configuração de software. Apresentamos aqui apenas as respostas à operação *getObject,* que oferece as consultas mais complexas que o serviço disponibiliza. As demais operações – *getCapabilities* e *getObjectById* – fazem parte do serviço, mas seus

respectivos resultados não serão apresentados aqui porque consideramos que as respostas das requisições a operação *getObject* são suficientes para a didática que necessitamos desenvolver durante nossa explanação.

Os resultados apresentados são conversões diretas[65] das saídas de consultas (*prepared queries*) SQL, que correspondem às chamadas das operações do Serviço para Proveniência na Web, a partir da cláusula *FOR XML AUTO, ELEMENTS*.

Inicialmente ilustramos três *milestones* que existem no projeto do Catálogo TDK na Figura 52 a partir do acesso a página *Roadmap* da ferramenta Trac deste projeto.

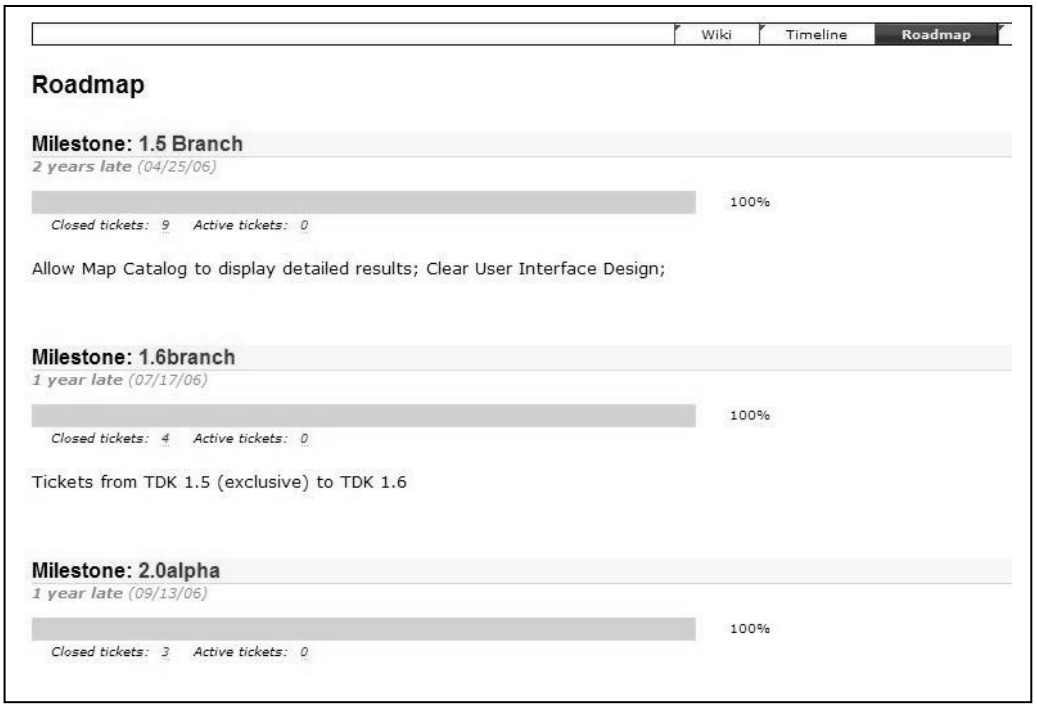


Figura 52: Página de *Roadmap* da ferramenta Trac que apresenta os *milestones* existentes no projeto

A requisição a operação *getObject* para recuperar os *milestones* é possível através do parâmetro que especifica o tipo do objeto que se deseja recuperar. A consulta é formada pela URL base “http://localhost:8080/ServletWPS/WPS?”

[65] Por limitações de tempo não disponibilizamos nesta versão do serviço o respectivo esquema XML.

acrescida de “REQUEST=getObject&maxRows=20&typeName=E5 Event”. Esta consulta produz a resposta apresentada na Figura 53 que é o resultado da operação *getObject* retornando todas as entidades que são do tipo “E5 Event” presentes no banco de dados de proveniência. O atributo “*entity_source*” representa a proveniência do conceito da ferramenta da respectiva instância do objeto retornado.


Figura 53: Resultado com todos os eventos no banco de dados de proveniência.

A Tabela 24 contém a consulta SQL (*prepared query*) correspondente ao resultado da Figura 53. Esta consulta recebe apenas um parâmetro - representado pelo caracter de interrogação "?" da linha destacada em ***negrito itálico*** - que corresponde ao tipo do objeto que se deseja recuperar, por exemplo, os nomes (completos que incluem o respectivo identificador) das classes do modelo mínimo: E5 Event, E7 Activity, CID2 Participant etc.

Tabela 24: Consulta SQL (*prepared query*) para a operação *getObject* que especifica o tipo do objeto que deve ser retornado

```
use TDKSimple

select
        entity.entity_id,
        entity.entity_name,
        entity.entity_description,
        entity.entity_source,
        relation_type.type_name,
        relation_type.type_description
from
        w6h2_cat_relations_type AS relation_type,
        w6h2_cat_relations AS relation,
        tdk_cat_entity AS entity
where
        entity.entity_id = relation.entity_id
        and
        relation.relat_type_id = relation_type.type_id
        and
        relation_type.type_name = ?
for XML AUTO, ELEMENTS
```

A operação *getObject* pode filtrar outros resultados através dos parâmetros relationToType e showReference. O primeiro permite especificar o tipo da propriedade de um objeto, por exemplo, “P11 has participant”. Lembramos que o Serviço de Proveniência na Web não implementa, na versão atual, a consulta automática à inversa de “P11 has participant”. Porém, em versões futuras, poderia realizar duas consultas SQL e então apresentar a propriedade consultada e sua inversa (se existente) de forma transparente para o usuário do serviço. O segundo parâmetro permite especificar um nome parcial de qualquer artefato (E84 Information Carrier), por exemplo, “.cpp” ou “Catalog”. Neste caso, assumimos que o modelo mínimo foi expandido com esta classe. O artefato pode estar associado a uma instância de E5 Event que represente, por exemplo,

o evento de *Commit* do Subversion que através de um *hook* concluiu *tickets* pendentes. A Figura 54 ilustra três destes eventos que correspondem aos *ChangeSets* 1127, 1128 e 1129 e as respectivas instâncias de E84 Information Carrier associadas. A Figura 54 corresponde a URL base "http://localhost:8080/ServletWPS/WPS?" acrescida de:

"REQUEST=getObject&maxRows=20&typeName=E5 Event&showReference".

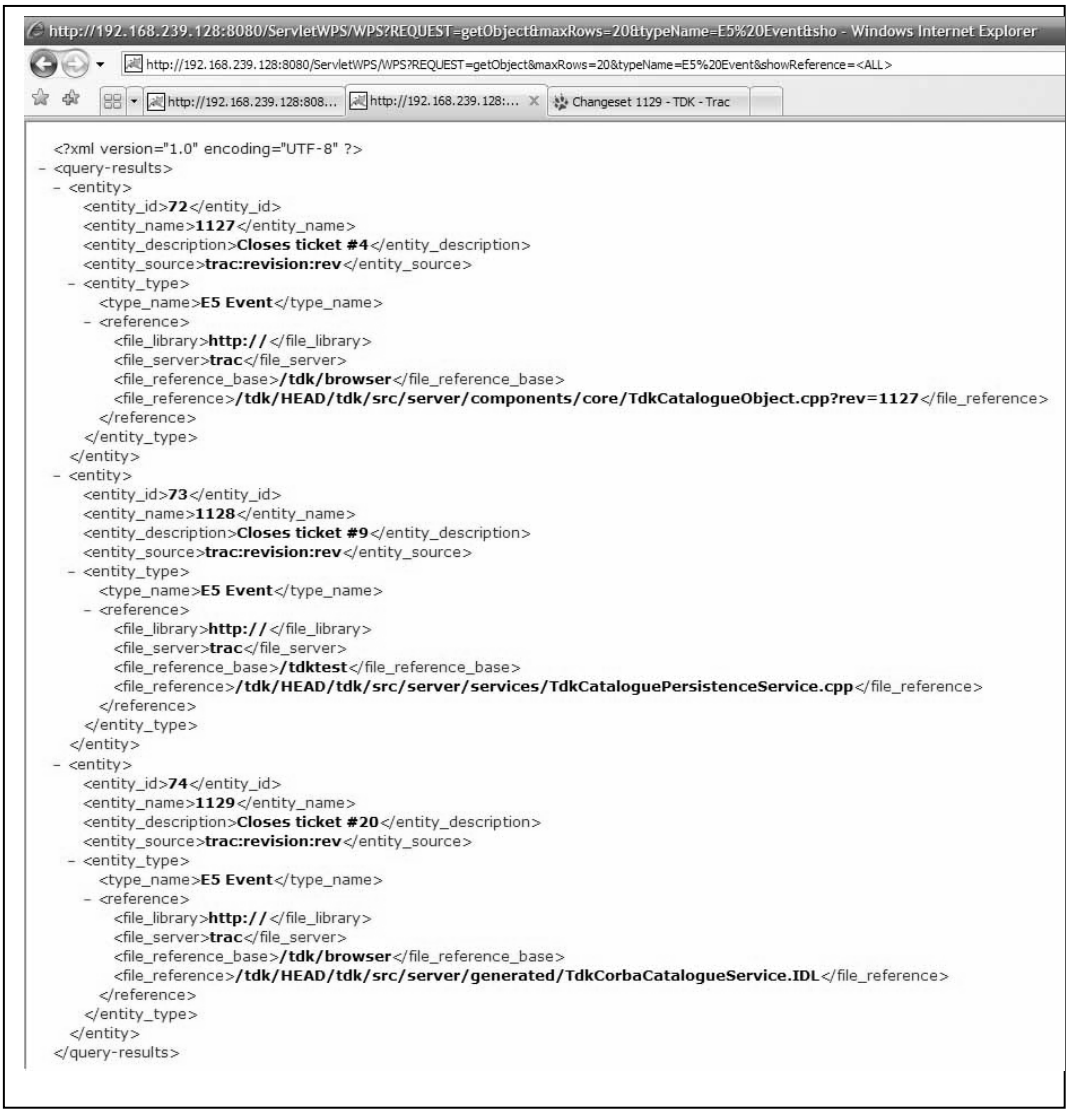


Figura 54: Eventos de *Commit* do Subversion que correspondem a *ChangeSets*

A Figura 55 ilustra o conteúdo do terceiro *ChangeSet* (E5 Event) apresentado na Figura 54.

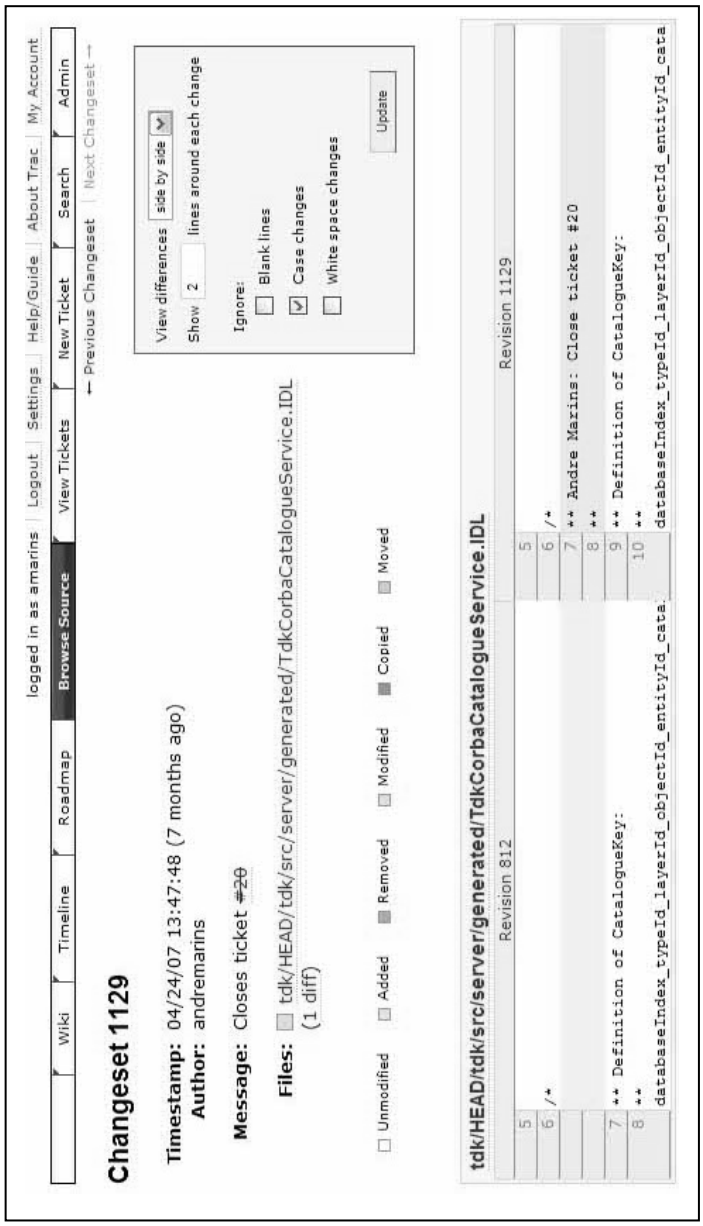


Figura 55: *ChangeSet* 1129 que concluiu o *ticket* #20.

A Figura 55 apresenta um único artefato modificado (*modified)*: “TdkCorbaCatalogueService.IDL”. Abaixo e à direita, em relação ao título

"Changeset 1129", as linhas 7 e 8 da *Revision*[66] 1129 identificam as alterações do artefato em relação ao *ChangeSet* imediatamente anterior (*Revision* 812) que também registrou alguma mudança (*Commit*) desse mesmo artefato.

Por fim, a ferramenta Trac define um *milestone* como um simples agrupamento de *tickets*. Com a incorporação do modelo de proveniência, o Trac poderia ir além das explicações dedutíveis como a linha to tempo – página *Timeline* - que simboliza a cronologia dos *ChangeSets*. Através do *plugin* RevTree (seção 4.5.3.1) seria possível "navegar" entre *ChangeSets*, por exemplo, do *ChangeSet 812* se chegar ao *ChangeSet* 1129, o que seria portanto, uma explicação dedutível análoga a linha do tempo que poderia ser construída com o evento de *Commit* (*Revision 812*) acontecendo antes do evento (*Revision 1129*). Através desses eventos até é possível se chegar aos *tickets* correspondentes, mas sem proveniência o dado fica sem "qualidade".

Os *ChangeSets* 1127, 1128 e 1129 correspondem respectivamente ao fechamento dos *tickets* 4, 9 e 20 que por sua vez estão agrupados na Figura 56 sob o *milestone* Branch 1.5 (Figura 52 e Figura 53) que possui ao todo nove *tickets* concluídos.

| Ticket | Summary | Owner | Type | Priority | Component | Version |
|---|---|---|---|---|---|---|
| *4* | *Permitir definir como estado padrão um estado que não seja de seleção.* | *leone* | *defect* | *highest* | *controle de aplicação* | *1.5* |
| *9* | *Conversão dos estilos avançados não funciona.* | *ademar* | *defect* | *highest* | *biblioteca de estilos* | *1.5* |
| *20* | *Criação de tema com atributo* | *leone* | *defect* | *highest* | *core* | *1.5* |
| 10 | GPS | ademar | task | normal | gps | before 1.5 |
| 24 | Validar os ponteiros dos canvas | leone | defect | normal | core | 1.5 |
| 26 | Inserir linha em layer vazio | natacha | defect | normal | edição | 1.5 |
| 28 | Atualização de propriedades do objeto | natacha | defect | normal | edição | 1.5 |
| 29 | Serviço createObject no JNI retorna layerId incorreto | leone | defect | normal | serviços | 1.5 |
| *21* | *GPS com frequência de 9600* | *ademar* | *task* | *low* | *gps* | *1.5* |

Figura 56: *Tickets* do *Milestone* Branch 1.5

Argumentamos que através da aplicação do modelo de proveniência seria possível associar mais significado ao dado, permitindo a representação de relacionamentos (qualificados) não dedutíveis (CID7 Event Reason) que podem capturar por que (*Why*) – causas, justificativas, motivos etc - um *ticket* está ligado a outro. Adicionalmente, um *milestone* estando associado a um arranjo -

[66] Tem o mesmo significado que *ChangeSet*

como uma ordem parcial - de *tickets* (CID6 PO Event), poder-se-ia dizer que esta ordem é uma explicação de como (*How*) o *milestone* foi alcançado.


Figura 57: *Tickets* #4, #9 e #20 e respectivos IDs de participantes

A Figura 57 ilustra uma consulta a todos os relacionamentos do tipo “P14 carried out by (performed)” para os objetos do tipo “E7 Activity” (*tickets*). Ao todo são três *tickets* tendo cada um dois participantes identificados pelos respectivos IDs.

## 4.6. Considerações Finais

O serviço de proveniência na Web é um protótipo que explorou a representação de objetos de proveniência em um banco de dados relacional e consultas, através da Web, que retornam dados de proveniência em formato XML. A avaliação do modelo de proveniência para representação de conceitos de uma ferramenta de controle de ocorrências nos permitiu uma primeira aproximação para validá-lo para o domínio de aplicações de Gerenciamento de Configuração de Software. Entretanto, é necessário que ferramentas desse e de outros domínios sejam avaliadas e, com isso, o estudo do modelo de proveniência seja aprofundado.

A seguir, apresentamos (capítulo 5) nossas conclusões, destacamos as principais contribuições deste trabalho de dissertação e apontamos algumas direções de trabalhos futuros.