Medição de Software para Pequenas Empresas: Uma Solução Baseada na Web

Luiz Paulo Alves Franca[1]
e-mail: franca @inf.puc-rio.br

Arndt von Staa[2]
e-mail: arndt @inf.puc-rio.br

Carlos José Pereira de Lucena
e-mail: lucena @inf.puc-rio.br

PUC-RioInf.MCC32/98

Abstract: The use of metrics is directly related to software process improvement. Measurement programs should be flexible and easily adaptable to the features of each organization. Moreover, in the context of small software business, measurement should be feasible in spite of the lack of resources. In order to produce measurement systems that face the needs of the dynamic reality of small business and support an incremental evolution, we have developed a meta-system that is capable of instantiating suitable measurement systems to each organization. The system generation is based on the generic measurement model stored in software engineering meta-environment. The measurement systems generated are Web-based applications.

Keywords: Software Engineering Environment, Application Generators, Software Metrics, Small Software Organizations, Software Quality, WEB-based software.

Resumo: O uso de métricas está diretamente ligada à melhoria do processo de produção de software. Sistemas de medição devem ser flexíveis e facilmente adaptáveis às características de cada organização. Adicionalmente, no contexto de pequenas empresas de software, a medição tem que adaptar-se à escassez de recursos. Com o objetivo de produzir sistemas de medição evolutíveis e que atendam à realidade dinâmica de pequenas empresas, desenvolvemos um meta-sistema capaz de instanciar sistemas de medição adequados a cada organização. A geração dos sistemas é feita a partir do modelo genérico de medição armazenado num meta-ambiente de Engenharia de Software. Os sistemas de medição instanciados são aplicações baseadas na Web.

Palavras-chave: Ambientes de Engenharia de Software, Geradores de Aplicação, Métricas de Software, Pequenas Organizações de Software, Qualidade de Software, Software Baseado na Web.

[1] Trabalho apoiado por: CNPq, Bolsa de Doutorado

[2] Trabalho apoiado por: CNPq, Bolsa de Pesquisador 300029/92-6

1. Introdução

Em muitas empresas, os projetos de desenvolvimento transcorrem como uma verdadeira caixa-preta, o usuário não tem muita certeza do que irá receber e o gerente do projeto apenas aguarda pelo fim do projeto. Como conseguir uma maior visibilidade de um projeto em andamento ?

A cada ano, a área de desenvolvimento de software, é apresentada a novas ferramentas e técnicas que são vendidas como verdadeiras campeãs de produtividade. Como comprovar os benefícios esperados [15]?

A resposta para ambas as perguntas reside na medição de software. A medição de software está diretamente ligada à melhoria do processo de produção de software [39,40]. Através da medição, o processo de produção de software pode ser monitorado, proporcionando uma avaliação constante do processo e a possibilidade de ajustes em função de tendências detectadas. Nos modelos de maturidade para a melhoria do processo de software, a medição tem um papel destacado. O CMM [37] (*Capability Maturity Level*) da SEI, enfatiza o controle quantitativo do processo, e nos níveis mais elevados (4 e 5), a medição é uma KPA (*key process area*). No SPICE [13] (*Software Improvement and Capability dEtermination*), a medição é considerada uma prática genérica aplicada em todos os processos. Nas séries ISO-9000, a medição é a base para o controle de qualidade.

Dentro de uma organização, a iniciativa de implementar um programa de medição eficiente e eficaz, depara-se muitas vezes com problemas do tipo:

- Da extensa lista de métricas, quais são as adequadas para o meu caso?
- Quais são as ferramentas e os processos necessários para realizar as medições desejadas?
- Quais são as ferramentas e os processos disponíveis?
- Quais são os recursos financeiros e humanos necessários para implementar e manter o programa de medição?
- Como analisar os dados coletados?
- Como apresentar as informações relacionadas aos dados coletados?
- Como atuar sobre ferramentas, métodos e processos no sentido de automatizar a medição?

Normalmente os programas de medição são abordados no contexto de grandes organizações [17, 18, 38], porém o setor de software é caracterizado por uma maciça participação de pequenas empresas. Segundo dados do Ministério da Ciência e Tecnologia, no Brasil, 90% das empresas de software são pequenas empresas com menos de 50 empregados e faturamento inferior a U$ 4 milhões [5]. Nessas empresas, devido ao porte, os recursos para investimentos, quando existem, se concentram nas atividades fim [1,7]. É muito difícil introduzir nessas empresas, um sistema de medição, que além do custo de software, também adiciona custos de pessoal.

Um programa de medição envolve a geração de um grande volume de dados, que somente pode ser manipulado e analisado de forma eficaz, se devidamente automatizado. Caso o programa de medição envolva várias facetas do processo de desenvolvimento, ele precisará ser apoiado por uma coletânea de ferramentas de medição [10, 11, 14, 19, 20, 21], bem como por bases de dados capazes de prover

tratamento estatístico dos dados, facilitando a exploração das informações que podem ser extraídas (*data mining*) das bases de dados de medição.

No sentido de suplantar as dificuldades para a introdução de um programa de medição e adequar a sua aplicação à realidade de poucos recursos de uma pequena empresa de software, desenvolvemos um meta-sistema que facilita a configuração de diversos programas de medição.

Este artigo apresenta o meta-sistema de medição de software da seguinte forma: a seção 2 apresenta os principais problemas da área de medição de software; a seção 3 apresenta a medição de software no contexto da pequena empresa; a seção 4 apresenta os requisitos básicos de um sistema de medição; a seção 5 apresenta a arquitetura do sistema de medição; a seção 6 descreve o meta-sistema de medição; a seção 7 apresenta uma comparação com trabalhos relacionados e análise de algumas ferramentas de medição. Na seção 8 é apresentada a situação atual e futura deste trabalho. As conclusões são apresentadas na seção 9.

2. Problemas na Área de Medição de Software

Dentro da Engenharia de Software, a medição de software é campo de pesquisa relevante. Segundo Dentro da Engenharia de Software, a medição de software é campo de pesquisa relevante. Segundo Pfleeger [40], "*em qualquer campo científico, as medições fornecem descrições quantitativas dos processos e dos produtos, possibilitando a compreensão de comportamentos e de resultados. Este aumento de entendimento permite a melhor seleção de técnicas e ferramentas para controlar e melhorar os processos, produtos e recursos. Como a Engenharia envolve a análise de medições, a Engenharia de Software somente será uma verdadeira engenharia, quando estiver sedimentada numa sólida fundação de teorias de medição.*" [6,29]

Basili, Fenton, Pfleeger e Glass [2, 4, 15] entre muitos outros têm chamado a atenção para a carência de evidências experimentais que confirmem os benefícios das novas propostas de métodos, técnicas, ferramentas e processos apresentadas pelos diversos grupos de pesquisa e desenvolvimento, e pelas empresas que comercializam ferramentas ou treinamento. Torna-se necessário não só propor, como também validar experimentalmente as propostas. Dentro do contexto da Engenharia de Software Experimental (ESE), a medição de software é um importante instrumento de verificação. Através da medição, os laboratórios de Engenharia de Software, poderão obter dados quantitativos, para contrapor ou confirmar os supostos benefícios advogados sem a devida comprovação experimental

A medição de software está diretamente ligada à garantia de qualidade de software. Ao longo do desenvolvimento do software, três pontos devem ser foco da medição: produto, processo e recurso. Na prática, observa-se que o foco principal da medição tem sido o produto final, ajudando a verificar a robustez do produto e sua aderência em relação à especificação. Na literatura são encontrados inúmeros artigos sobre medição associada a testes, identificação de tendências de erros, previsibilidade de falhas. Conforme é lamentado por Grady [22], encontra-se ainda pouco material sobre medição de software relacionada a artefatos intermediários tais como especificação de Requisitos, análise e design. Entre os artigos que abordam este assunto, encontram-se [8, 9, 12, 24, 25, 36, 41, 42, 46], relatando experiências feitas, e os benefícios alcançados, ao aplicar medição aos artefatos produzidos ao longo do processo de desenvolvimento.

Kitchenham et al [29]. e Briand et al. [6] mencionam a necessidade da criação de um framework para a validação das métricas de software. Empiricamente observa-se que o princípio de Pareto se aplica ao conjunto de medições, ou seja, uma parte significativa das métricas propostas pouco contribui para levar ao aprimoramento efetivo de produtos e processos de desenvolvimento de software. Por outro lado todas estas medições induzem custos. Torna-se necessário então poder identificar quais as métricas que melhor explicam propriedades relevantes do software e dos processos utilizados.

A literatura corrente enumera uma extensa lista de métricas que podem ser utilizadas na medição do processo de software. Essas métricas podem ser classificadas quanto às propriedades ou artefatos que estão medindo (produto, processo, recurso), ou quanto à forma da medição (direta ou indireta). Esta grande variedade de métricas, se de um lado permite que vários aspectos do desenvolvimento de software sejam mensurados, por outro pode gerar a falta de objetividade e, por conseguinte, o fracasso do programa de medição dentro de uma empresa [3, 35].

3. Programas de Medição em Pequenas Empresas

Na literatura a medição integrada ao processo de desenvolvimento, é geralmente retratada no contexto de grandes empresas, com grandes equipes de desenvolvimento [17, 26, 36, 38]. No entanto, sabemos que grande parte da produção de software é realizada por pequenas empresas. Nessas empresas, devido ao reduzido porte, os recursos de investimentos, quando existem, se concentram nas atividades fins, sendo gerados poucos excedentes que poderiam ser utilizados por um programa de medição abrangente e detalhado. É muito difícil introduzir um sistema de medição nessas empresas, pois, além do custo de disponibilização (*deployment*), também gera novos custos de pessoal. Este problema é agravado quando se trata de pequenas empresas no início de sua existência.

O INFOGENE é uma incubadora de empresas de software, criada em 1997, mantida pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). As empresas incubadas são formadas por recém-formados. As empresas são admitidas com base em um plano de negócios avaliado pela comissão gestora da incubadora. O período de incubação é de 2 anos.

A incubadora fortalece novas empresas, ajudando-as a se estabelecerem e crescerem durante o período de incubação. A incubadora fornece assistência administrativa, apoio comercial, acessos a financiamentos, e suporte técnico (Figura 1). Ela também oferece serviços de escritório, estrutura de informática e espaço físico para a instalação das empresas. O principal objetivo da incubadora é produzir empresas que sejam viáveis após o período de incubação.

O suporte técnico é um dos serviços prestados pelo INFOGENE. Através deste serviço é esperado que as empresas incorporem técnicas de engenharia de software ao seus processos de produção. No início da incubação, a jovem empresa apresenta pouca maturidade nos seus processos, durante a incubação, a empresa vai sendo exposta a técnicas e ferramentas de engenharia de software. No final da incubação, espera-se que as empresas estejam aptas a alcançar o nível 3 do CMM. Cabe salientar que o serviço de suporte técnico pode ser estendido a pequenas empresas que não estejam vinculadas à incubadora, ou que já tenham concluído o seu período de incubação.

Uma das formas de monitorar a evolução dos processos de produção das empresas, é a adoção de um programa de medição. O suporte técnico pretende fornecer a cada empresa que utilize seus serviços, um sistema de medição de software. Este sistema de medição deve adaptar-se a realidade de cada empresa. No início deverão ser coletadas somente medições rudimentares, porém à medida que a empresa evolua, o sistema de medição deve estar apto para coletar, registrar e manipular medições bastante sofisticadas.

O banco de dados contendo as medições, será uma base de experiência a ser compartilhada entre todas as empresas, sendo uma ferramenta importante para homogeneização da qualidade das empresas. O banco de medições será utilizado também para avaliar e melhorar o próprio processo de incubação adotado pelo INFOGENE.

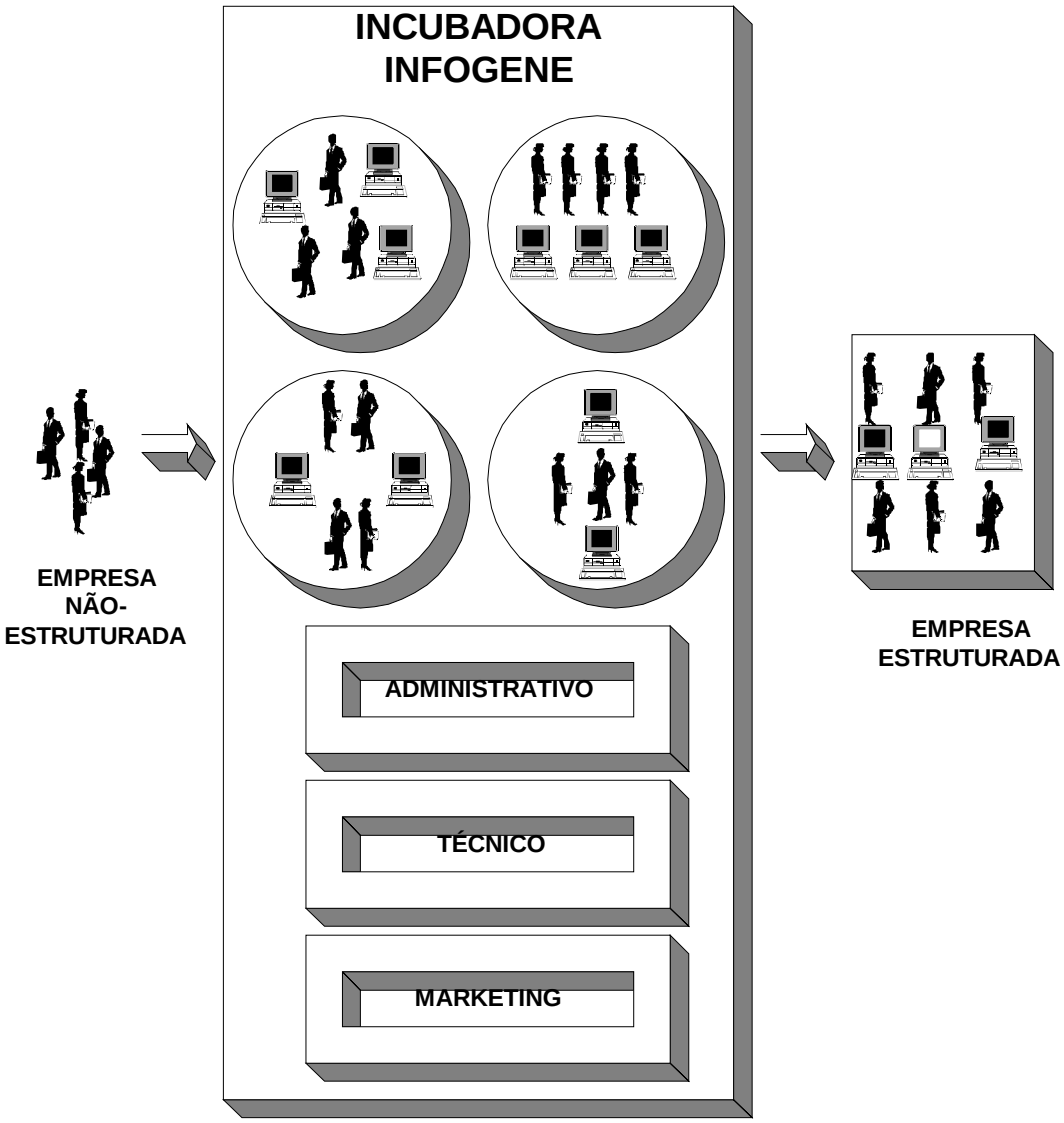

Figura 1. Visão Esquemática da Incubadora INFOGENE

4. Requisitos Básicos do Sistema de Medição de Software

Um sistema automatizado de suporte ao programa de medição de software deve adaptar-se à seguinte realidade:

- As empresas apresentam diferentes níveis de maturidade

O sistema de medição deve se adequar tanto a uma organização (empresa ou divisão) no nível 1 do CMM como às que estejam em níveis superiores. Numa organização no nível 1, onde os processos de software não são definidos, é pré-requisito para a melhoria da maturidade da organização, os seus projetos de software passarem a ser controlados. Nesta situação, o sistema de medição deve dar apoio a medições básicas de controle de projeto. Caso a organização já possua alguma maturidade, mesmo que ainda no nível 1, o sistema de medição estará mais alinhado aos processos da organização, podendo manipular medições mais sofisticadas e convivendo com uma maior automação do processo de software.

- As medições variam conforme a necessidade

As medições são realizadas no sentido de suprir diferentes necessidades de informação de uma organização. Mesmo dentro de uma mesma organização a necessidade de informação varia por usuário. Enquanto o gerente de projeto está interessado em medições que sinalizem a evolução do projeto, o analista de sistema pode estar interessado na natureza e taxa de erros dos programas codificados.

- Criação de nova métricas

Novas métricas podem ser criadas no sentido de atender novas demandas de informação. A necessidade de obter informações mais detalhadas envolvendo as aplicações desenvolvidas, faz com que métricas relacionadas a diferentes tipos de aplicações sejam criadas (Ex.: automação comercial, computação gráfica, etc.).

- Medição de Software não é padronizada

Ainda não existem normas de ampla aceitação para a medição de software. Mesmo entre as medições mais conhecidas existem variações quanto à sua aplicação. Por exemplo, existem diferentes maneiras para contabilizar linhas de código. Mesmo para a medição de tamanho do produto a literatura sugere a utilização de linhas de código ou de pontos de função, com um intenso debate entre os adeptos de cada uma das formas de medir [18, 28].

- Medição Distribuída

O sistema de medição deve funcionar tanto para equipes de trabalho situadas no mesmo local como para equipes geograficamente distribuídas. A subcontratação é um exemplo de distribuição de equipes de trabalho. As organizações podem realizar a produção do software tanto em seu próprio espaço físico, como em laboratórios, ou mesmo no cliente.

As seguintes facilidades são esperadas de um sistema de medição [3,16]:

- Coleta de Dados não intrusiva

As medições devem ser realizadas de forma a não interferir no processo de produção. Sempre que possível a coleta de dados deve ser feita de forma automática.

- Histórico das Medições

As medições devem ser registradas em um banco de dados. Este banco de dados serve tanto para a modelagem do processo de software como para construção de modelos de predição aplicados à realidade da organização.

- Divulgação das Medições

O sistema deve prover mecanismo de divulgação das medições realizadas, permitindo que várias pessoas de diferentes lugares possam ter pronto acesso às informações mais atuais sempre que necessário. As análises realizadas devem ser apresentadas através de gráficos e estatísticas. O sistema deve permitir diferentes agregações de dados e possibilitar consultas ad hoc.

5. Arquitetura do Sistema de Medição de Software

Foi desenvolvido um modelo genérico de dados para sistemas de medição. Este modelo foi baseado nas proposta de modelo de validação de medição proposto por Fenton em [29], acrescentando a modelagem do processo de trabalho (ex.: alocação da equipe, habilitação da equipe por tipo de métrica, etc.). Neste modelo as medições são modeladas como atributos dos produtos que são gerados ao longo do desenvolvimento do software. Esta abordagem permite que o modelo de dados de um sistema de medição específico (ex.: sistemas de medição de tamanho de produtos) já esteja mapeado no modelo genérico de medição. Analisamos algumas ferramentas de medição (vide seção 5) no sentido de ajustar e validar a proposta de modelo genérico de dados para sistemas de medição.

A figura 2 descreve a estrutura do banco de dados de um sistema genérico de medição.

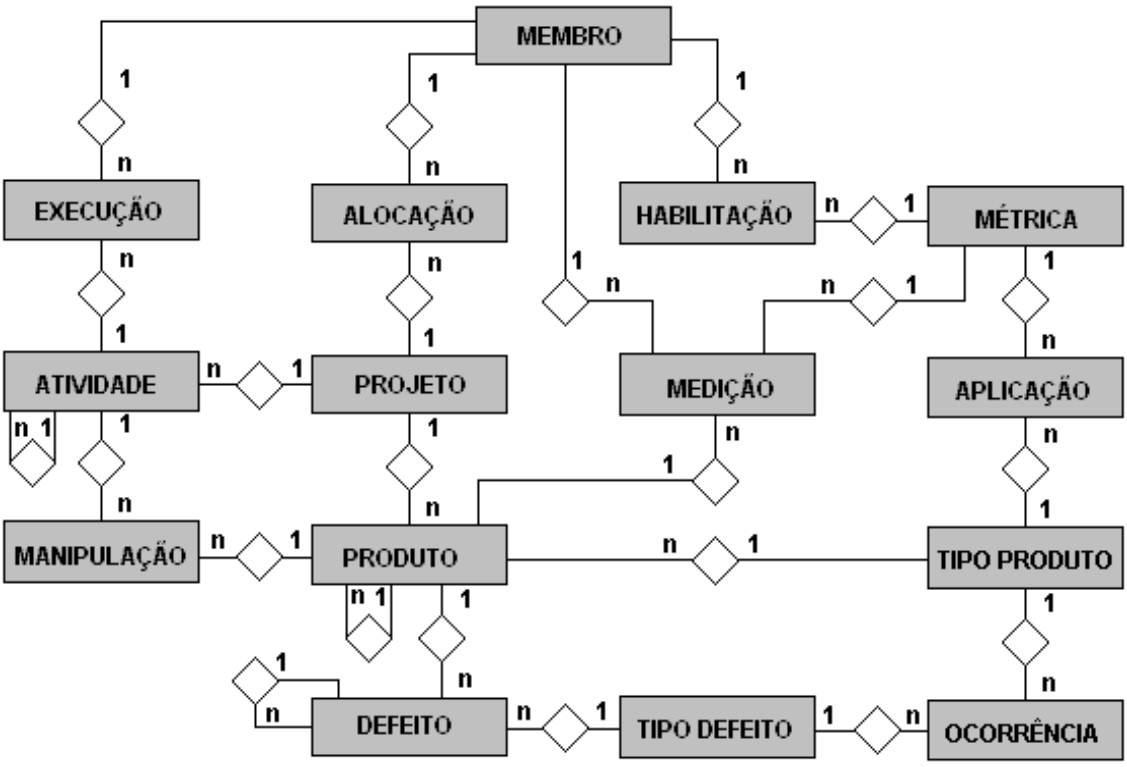

Figura 2. Esquema de dados do sistema genérico de medição

As principais entidades do modelo são: Atividade, Artefato[3], Defeito, e Métrica. Estas entidades podem apresentar algum tipo de classificação (ex.: atividade de análise, atividade de *design*, etc.). As entidades Artefato, Atividade, e Defeito apresentam um auto-relacionamento. No caso do Artefato e da Atividade, o auto-relacionamento representa a relação de decomposição; já no caso do Defeito, representa a relação de causalidade entre defeitos. O processo de trabalho está representado no modelo através da entidade Membro e dos seus relacionamentos. Um membro da equipe pode estar alocado a mais de um projeto. Um atividade pode envolver a participação de um ou mais membros. Mais de uma pessoa da equipe pode estar habilitada na aplicação de uma determinada métrica. Em relação a um artefato, durante o seu desenvolvimento e utilização, várias medições são realizadas. A execução de uma atividade envolve o registro de tempo de sua realização.

No sentido de atender os requisitos descritos na seção 2, adotamos a solução Web como arquitetura do nosso sistema (Figura 3). A heterogeneidade e a facilidade de operação foram pontos importantes para seleção de uma arquitetura web. Os diferentes usuários do sistema de medição possuem diversas plataformas operacionais (PC, Sun, Mac, Unix, Windows, etc.), a solução Web garante um sistema de medição para múltiplas plataformas. A utilização do sistema de medição através de *browsers*, facilita o acesso e operação do sistema pelo usuário.

O sistema de medição poderá ser distribuído, correspondendo, por exemplo, a uma hierarquia de sistemas de medição, na qual os sistemas de um dado nível correspondem à agregação das medições dos sistemas dos níveis hierárquicos inferiores. Uma outra arquitetura, é um sistema de medição corporativo que armazena as medições realizadas por sistemas de medições remotos.

O repositório armazena dados sobre projetos de desenvolvimento e manutenção de software. Os dados registrados abrangem medições dos projetos, dos produtos, dos processos de desenvolvimento, e dos produtos finais.

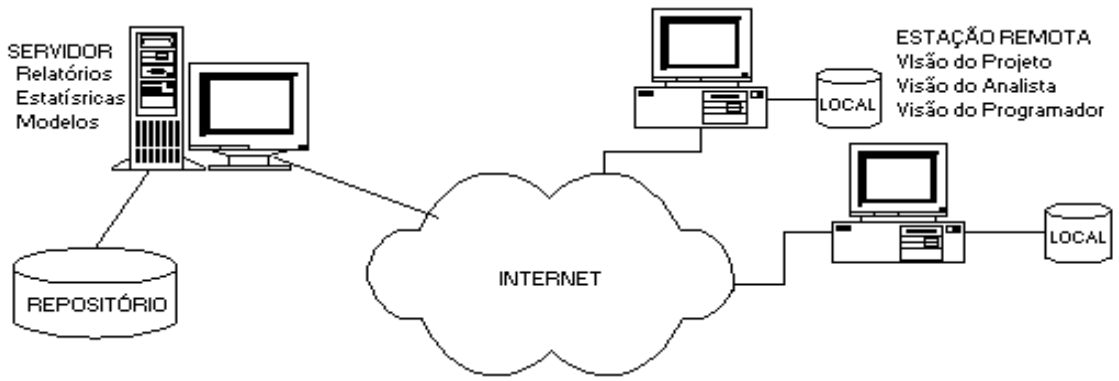

Figura 3. Arquitetura do Sistema de Medição

[3] *Artefato* é qualquer resultado controlado gerado durante o processo de desenvolvimento (work product) ou por ele necessitado. Exemplos de artefatos: especificação de requisitos, planos, módulos de programa, os programas entregues ao usuário, manuais, compiladores, ferramentas. Podem ser externos (produtos) quando entregues ao usuário final, ou internos.

A arquitetura proposta permite que o repositório central seja diretamente preenchido pelo desenvolvedor ou através da importação de dados de repositórios remotos. Caso o usuário não esteja conectado ao banco de dados central, as medições poderão ser registradas localmente, para posterior *upload* dos dados, assim que a conexão via Internet for estabelecida.

As medições a serem coletadas são definidas a partir das necessidades de informação da organização. O GQM (*Goal-Question-Metric Paradigm*) [3] e sua extensão M^3P (*Model, Measure, Manage Paradigm*) [34, 35] são métodos sistemáticos para determinar as métricas que realmente dão apoio à tomada de decisão do usuário deste tipo de informação (tais como o gerente de projeto, analista de sistemas, grupo de controle de qualidade, etc.).

As medições podem ser por exemplo:

- cronogramas de projeto;
- esforço para execução do projeto;
- características do projeto (tamanho, complexidade, natureza, custo, etc.);
- características do produto (tamanho, complexidade, natureza, etc.);
- medições variadas, selecionadas ou definidas pelos gerentes de software;
- falhas, defeitos e faltas encontradas, catalogadas por sua natureza.

É importante ressaltar que a lista de métricas disponibilizada pelo sistema é configurável, variando de organização para organização dependendo da amplitude e abrangência do programa de medição de cada empresa.

Na medida em que o repositório for sendo povoado com as diversas medições, ele será fonte importante para:

- Entendimento dos processos de desenvolvimentos adotados;
- Suporte à gerência de projeto;
- Avaliação de melhorias do processo de desenvolvimento;
- Identificação dos problemas sistemáticos encontrados nos produtos e nos processos.

6. Meta-sistema de Medição de Software

6.1 Processo de construção do Meta-sistema de Medição de Software

A implementação da arquitetura apresentada na seção 3, utilizando técnicas convencionais de produção de software é economicamente inviável, pois teríamos que desenvolver sistemas específicos para conjuntos de métricas ou por tipos de organização. Para viabilizarmos a nossa proposta de sistemas de medição flexíveis, adotamos um processo de construção baseado em um gerador de aplicação a partir de especificação de elevado nível de abstração (Figura 4). Esta técnica de construção de geradores de aplicação orientados por domínio tem alcançado bons resultados [31, 32, 33].

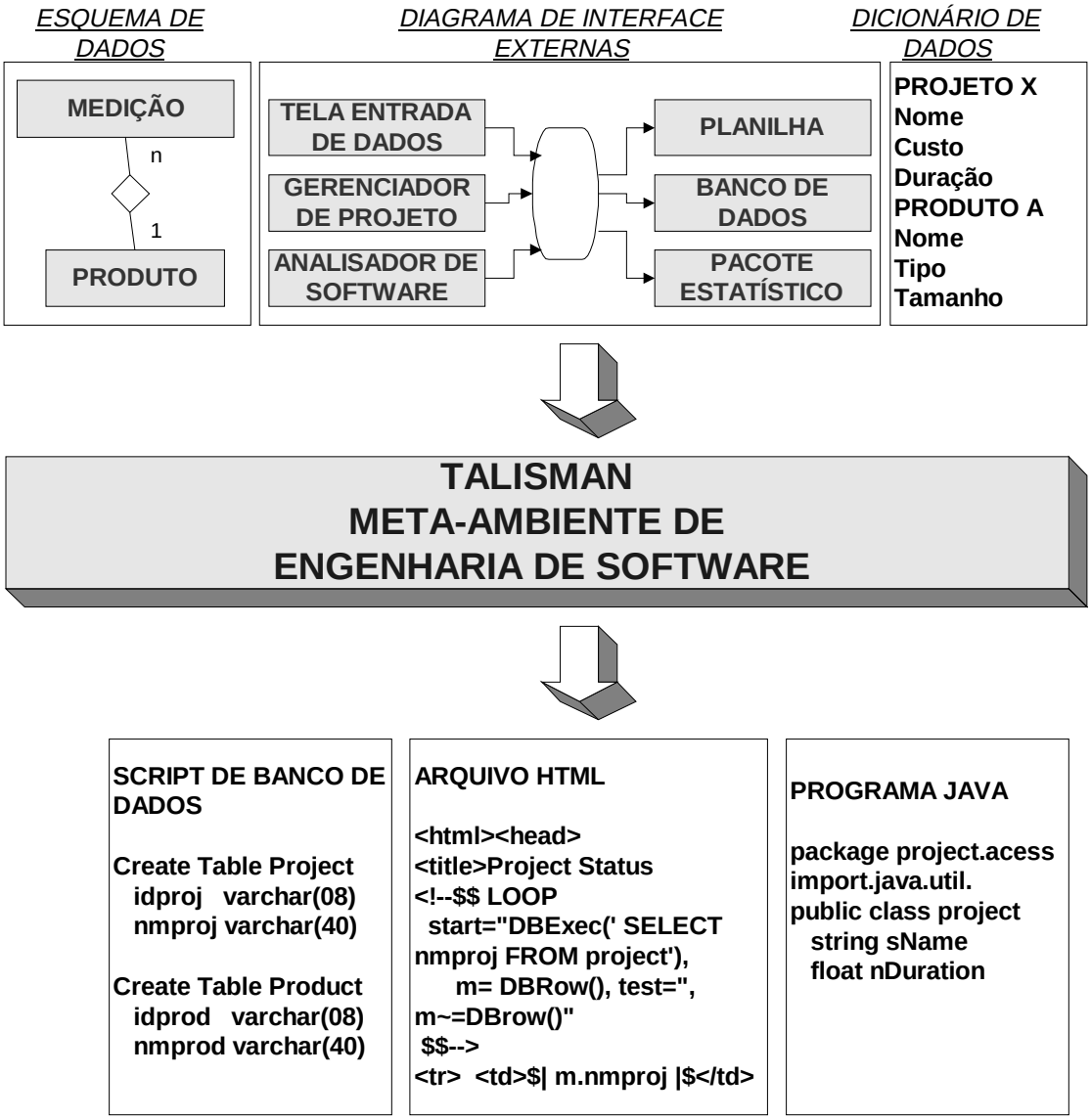

Figura 4. Visão geral do meta-sistema de medição de software

Em nosso trabalho, o sistema-alvo é um aplicação para ser executada através de *browsers* da Internet, contendo partes escritas em HTML, Java e LUA [23]. O HTML e o Java são utilizados para codificar os programas de interface com usuário. Já as funções escritas em LUA são utilizadas para fazer a interação da aplicação com o banco de dados. Para geração deste tipo de aplicação, utilizamos o meta-ambiente de Engenharia de Software Talisman [45]. Tanto o acesso como a apresentação do dicionário de dados do meta-ambiente utilizado podem ser configurados.

O diagrama de Entidade-Relacionamento do modelo genérico de medição (seção 2) foi registrado no meta-ambiente, e as telas de entrada do dicionário de dados foram customizadas para o domínio da medição de software e para a obtenção de informações da implementação. A figura 5 é um exemplo da tela customizada do dicionário de dados.

```
Nome da Entidade:  TIPO DE PROJETO
Identificador da Entidade:  TPPROJETO
Banco de Dados:   MEDICAO

Atributos definidos na entidade
   Chaves determinantes da entidade
      IDTPPROJETO
   Chaves estrangeiras da entidade
   Atributos da entidade
      NMTPPROJETO
      TXTTPPROJETO

Detalhe dos atributos
   Atributo chave: IDTPPROJETO
            Nome do Campo: IDTPPROJETO
            Tipo do atributo: Caractere X
                              Texto
                              Num inteiro
                              Decimal
                              Real
                              Data
            Tamanho do atributo : 6
            Casas decimais      :

Titulos a serem exibidos no Sistema
Gerar HTML : X
   Titulo do HTML:
      Tipos de Projeto
      Type of Project
   Titulo do Frame:
      TIPOS DE PROJETOS
      PROJECT TYPES
   Titulo de telas envolvendo a entidade:
      Tela de Inclusao:
         Entre com um novo tipo de projeto :
         New Project Type :
      Tela de Selecao:
         Selecione um tipo de projeto
         Choose Project Type
```

Figura 5. Exemplo da Tela do Dicionário de Dados de um Entidade

Na figura 5 , as palavras em negrito são informações fornecidas pelo usuário. Podemos observar que através da tela do dicionário de dados é possível obter um conjunto mínimo de informações para a definição e implementação do banco de dados e para a definição das telas do sistema de medição a ser instanciado. As informações relativas a identificação da entidade e aos tipos dos atributos são utilizados na etapa de geração do *script* de criação das tabelas do banco de dados. Já as informações referentes ao texto a ser apresentado nas telas são utilizados na etapa de geração dos programas de interface com usuário.

Informações gerais que são utilizadas em todo o processo de geração são registradas através da tela do dicionário de dados referente ao objeto Banco de Dados. Como podemos observar na Figura 6, através desta tela são informados o tipo do banco de dados, as cores da interface, e o idioma a ser utilizado na aplicação. Em relação ao tipo do banco de dados, a versão atual do gerador é capaz de gerar scripts de criação de tabelas para os bancos: MSSqlServer, SqlBase, e MSAccess. Já em relação ao idioma, o gerador é capaz de gerar aplicações de medição em português ou em inglês. Para cada texto a ser exibido, o usuário deve fornecer o texto em português e em inglês (vide Figura 5).

```
Configuracao da geracao  do  HTML-LUA

   Idioma: PORT
   Tipo de Banco de Dados: MSSqlServer
   Titulos a serem exibidos pelo sistema:
      Opcao do Menu Principal:
         Menu Principal
         Main Menu
      Mensagem de Erro na Conexão do Banco de Dados:
         Nao foi possivel conectar ao Banco de Dados
         Cannot open database
   Cores do sistema:
      Cor do titulo do frame :ffffff
      Cor da opcao selecionada :ffff00
      Cor do corpo do Html :ffffff
      Cores do corpo do Frame:
         Bgcolor :006666 Link :000000 Vlink :00000
      Cores da Mensagem de Erro:
         Bgcolor :ffffff  Color   :
```

Figura 6. Tela do Dicionário de Dados: Informações Gerais

Uma coletânea de programas transformadores foram escritos utilizando a linguagem de manipulação do meta-ambiente. Estes programas geram código a partir das informações extraídas dos diagramas e do dicionário de dados do ambiente. Além dos arquivos HTML, Java, e Lua, os *scripts* de criação das tabelas do banco de dados do sistema de medição são gerados também através dos transformadores. A base da geração são as informações referente a cada entidade do modelo de dados. Para cada entidade, o gerador produz o *script* de criação da tabela da entidade e os programas de manipulação da entidade. Funções que são de uso genérico dentro da aplicação, i.e., não estão associadas a uma entidade específica, são codificadas dentro do programa gerador, e são copiadas para os arquivos gerados quando necessário. A figura 7, é um exemplo de um trecho do programa de transformação, as palavras em negrito são informações provenientes do dicionário de dados. A figura 8 é um exemplo do arquivo gerado, as palavras em negrito são informações que foram fornecidas pelo usuário. No apêndice I são apresentadas 2 telas de um sistema de medição instanciado.

```
InicFrm "Script HtmlMonta"
        Sequencia sAUX;
        FimDecl

        Titulo "##";
        NaoAvLin;
        Alias AliasIdent;
        NaoAvLin;
        Titulo "_monta.html";
        Titulo " <html>";
        Titulo " <head><title>";
        NaoAvLin;
        sAUX = LinhaTexto([Texto TxtHtml],GBL_nIdioma);
        Titulo sAUX;
        NaoAvLin;
        Titulo "</title></head>";
        Titulo " ";
        Titulo " <!--$$";
        Titulo " ";
        Titulo " dofile( \"";
        NaoAvLin;
        Alias AliasIdent;
        NaoAvLin;
        Titulo ".lua\" ) ";
        Titulo " ";
        Titulo " if not cgi.selecao then cgi.selecao = 1
                 end";
        Titulo " nome = opcoes[tonumber(cgi.selecao)][2]";
        Titulo " ";
        Titulo " $$-->";
        Titulo " ";
        Titulo " <frameset cols=\"20%,*\">";
        Titulo "    <frame src=\"$|cgilua.mkurl(\"";
        NaoAvLin;
        Alias AliasIdent;

   FimFrm
```

Figura 7. Trecho do fonte do Gerador

```
##TPPROJETO_monta.html
<html>
<head><title>Tipos de Projeto</title></head>
<!--$$
dofile("TPPROJETO.lua")
if not cgi.selecao then cgi.selecao = 1 end
nome = opcoes[tonumber(cgi.selecao)][2]
$$-->
<frameset cols = "20%,*">
  <frame src="$|cgilua.mkurl("TPPROJETO_barra.html",
        {selecao = cgi.selecao}|$" name = "esquerdo">
  <frame src="$|cgilua.mkurl(nome,
        {selecao = cgi.selecao}|$" name = "direito">
</frameset>
</html>
```

Figura 8. Trecho do arquivo HTML gerado

6.2 Utilização do Meta-sistema de Medição de Software

O processo de geração do sistema de medição envolve a seleção do esquema de dados mais adequado, do Banco de Dados, do banco de medições, e das interfaces com as ferramentas externas de entrada e de saída de dados. Este processo possibilita a construção de sistemas de medição, utilizando diferentes bancos de dados e diferentes interfaces externas. A evolução dos sistemas de medição gerados, pode ser realizada de diversas maneiras. Por exemplo, através da criação de novos esquemas de dados do banco de medição, da customização dos esquemas de dados existentes, da criação de geradores de *scripts* para novos bancos de dados, ou através da criação de geradores para novas interfaces externas.

Conforme pode ser visualizado na Figura 9, a configuração do sistema de medição dependerá das suas interfaces externas. A seleção das interfaces do sistema de medição, será em função das medições a serem realizadas. O usuário interage com o sistema de medição, definindo o esquema de dados mais adequado para o programa de medição que está sendo adotado. A definição do esquema de dados serve como um filtro na configuração das interfaces do sistema.

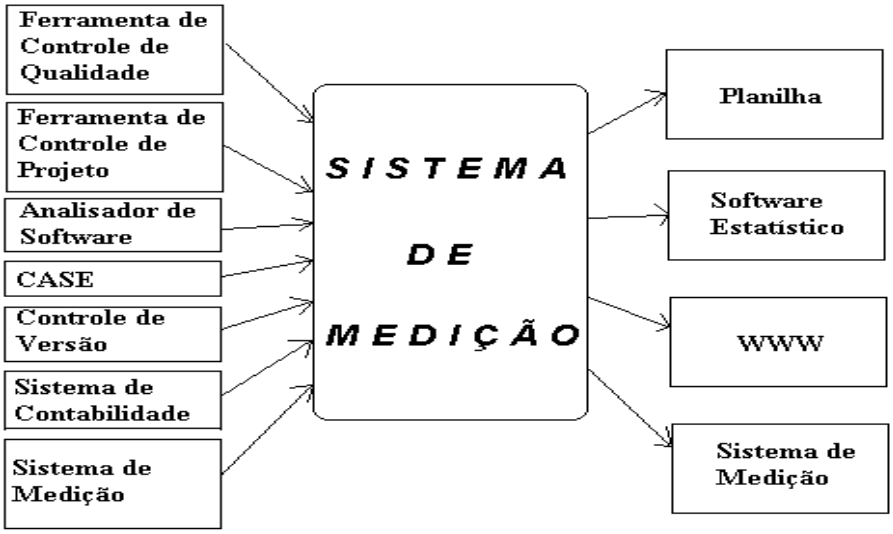

Figura 9. Interfaces Externas do Sistema de Medição

As interfaces tanto de entrada como de saída do sistema de medição variam conforme o grau de automação do ambiente externo do sistema de medição. Num ambiente externo completamente automatizado, as interfaces são implementadas através de *drivers* que interagem com as ferramentas da entrada e da saída, por exemplo, o *driver* de importação da ferramenta de controle de projeto MS Project. Já no caso do ambiente externo pouco automatizado, a entrada de dados é feita através de formulários *data-entry*. Nesta situação de pouca automação, o sistema de medição deve fornecer ferramentas básicas que apresentem funcionalidades mínimas das ferramentas de Controle de Projeto, Controle de Qualidade, Analisador de Software, Planilha, e Software Estatístico.

Quando da implantação do sistema de medição, as interfaces do sistema são avaliadas, sendo que a configuração final do sistema, poderá ser uma combinação de *drivers* e de formulários *data-entry*. Nesta etapa poderá ser identificada a necessidade de criação de novos *drivers* para interface.

7. Trabalhos Relacionados

No sentido de superar os problemas presentes na introdução de um programa de medição, e que são inerentes à pouca maturidade e difusão da área de medição de software, alguns projetos de pesquisa estão buscando o desenvolvimento de ferramentas de medição, que podem ser configuradas em razão da necessidade de medição.

O projeto coordenado por Kitchenham [40] desenvolveu uma ferramenta que permite a seleção de métricas utilizadas para povoar um repositório com os dados coletados.

O Amadeus [44], um projeto patrocinado pela ARPA (*Advanced Research Projects Agency*), monitora o processo de desenvolvimento desde as etapas iniciais e armazena os dados para uma análise futura. O Amadeus destaca-se pelas facilidades de customização da coleta de dados, e pela integração com algumas ferramentas CASE. Possui um conjunto de analisadores de linguagem e ferramentas de predição. No Amadeus, a coleta de dados é coordenada através de *scripts* que monitoram os eventos que ocorrem no ambiente de desenvolvimento (ex.: fim de uma fase do projeto). Através destes *scripts* as ferramentas são ativadas para a execução de coleta de dados específicas. Para implementar esta operação, o Amadeus possui uma forte integração com o sistema operacional (serviços para: execução de programas, comunicação entre processos, e calendário) e com o banco de dados do ambiente de desenvolvimento (acesso e monitoramento das alterações dos objetos de medição).

Offen [34, 35] desenvolveu o ambiente *The Squatter Toolset*, que permite a captura dos modelos empíricos e numéricos implícitos no processo de medição de software, suportando também a captura de dados, análise, interpretação, e armazenamento. O Squatter permite ainda a adição de novas ferramentas e a definição de novas fontes de dados e repositórios. O Squatter pode funcionar tanto como uma ferramenta *stand-alone* como parte de uma ferramenta CASE. O repositório do Squatter é implementado no banco de dados orientados a objetos Versant. A apresentação dos resultados e o processamento das estatísticas são feitas através do pacote S-Plus. O Squatter é implementado como uma biblioteca de classes de medições escrita em C++. A inclusão de uma nova medição pode corresponder a derivação de uma classe predefinida, ou a extensão do comportamento de uma classe, ou a implementação de uma nova função na linguagem S-Plus para cálculo de medições indiretas.

A ferramenta Emerald *(Enhanced Measurement for Early Risk Assessment of Latent Defects*) [26], é um sistema de suporte à decisão projetado para melhorar a confiabilidade do software de telecomunicações. O sistema foi desenvolvido pelas empresas de telecomunicações: BNR, Nortel e Bell Canada. O Emerald integra medições de software, modelos de qualidade, e disponibiliza os resultados através de browsers WWW. O Emerald incorpora o analisador de software DATRIX para medir atributos estáticos do código fonte.

Entre diversos outros projetos podemos destacar, o projeto *TAME (Towards improvement-oriented software environments*) coordenado por Basili, o projeto da ferramenta PROMIS na área de analisadores de software [30], e o projeto de divulgação de métricas pela WWW realizado pela Lockheed Martin Tactical Defense System [43] dentro do projeto STARS.

No mercado existe uma série de ferramentas para medição de software. Em geral as ferramentas são específicas para um conjunto de métricas (Ex: tamanho, esforço, etc) [10, 11, 14, 19, 20, 21]. Procuramos concentrar nossa análise nas ferramentas que apresentam características similares ao nosso projeto. Selecionamos algumas ferramentas de medição não específicas e as primeiras ferramentas com interface WWW. Segue-se abaixo, a descrição resumida das ferramentas analisadas:

A ferramenta WISE [47] (*The Web-Integrated Software metrics Environment*) registra e acompanha problemas ocorridos durante um projeto de software. A interface com usuário é feita através de browsers WWW. A ferramenta permite acompanhar quais as questões que estão em aberto, e verificar a performance da equipe na resolução de problemas. Um série de gráficos e relatórios são disponibilizados. Os relatórios podem ser exportados para planilhas.

WebProject [48] é uma ferramenta de acompanhamento de projetos para ser utilizada através da Internet. Através da ferramenta é possível acompanhar o esforço para execução e o status do projeto. O registro de esforço é feito através da tela de cronograma do projeto. Os dados alimentados pela ferramenta podem ser exportados para outras ferramentas de projeto.

Timelog [49] é uma aplicação para registro e análise do tempo alocado a um projeto. Timelog foi projetado para dar suporte ao *Personal Software Process* (PSP) [27] desenvolvido no SEI. O Timelog é escrito em Java podendo ser executado através de browsers da internet.

A ferramenta WebTime [50] registra o tempo na execução de atividades. A interface com usuário é feita através de *browsers* WWW. Cada membro da equipe registra a alocação de seu tempo ao longo do dia. As atividades são classificadas dentro da hierarquia Projeto -> Código -> Subcódigo. A ferramenta disponibiliza uma série de relatórios estatísticos que podem estar agregados por membro, por projeto, por código, e por subcódigo.

O MetricCenter [51] possui repositório de métricas para múltiplos projetos. Repositório implementado utilizando o banco de dados Sybase. Diferentes níveis de apresentação de resultados. Customização de relatórios e gráficos. A coleta de dados pode ser integrada a outras ferramentas (Rational Requisite Pro, Microsoft Project, etc.).

O SMMIS [52] (*Software Metrics Management Information System*) é uma ferramenta desenvolvida pelo exército americano para padronização da medição de software dentro da instituição. Através da ferramenta, os diversos centros de desenvolvimento fazem suas medições e enviam estas medições para centros de desenvolvimento hierarquicamente superiores. A versão atual da ferramenta é para o sistema operacional DOS.

A ferramenta NetProblemTracker [53] registra e rastreia problemas de software pela Internet. Através desta ferramenta, o usuário pode informar um problema e acompanhar o status da solução do problema. Tanto os produtos como as características dos produtos a serem controlados podem ser customizados. O fluxo das atividades para solução de um problema também é configurável.

8. Trabalho Atual e Futuro

A versão atual do meta-sistema de medição de software é capaz de gerar: sistemas de medição através de telas de data-entry, sistema de registro de tempo baseado no PSP [27], sistema de registro de problemas. Já foram criados também geradores para os bancos de dados Access, SQLSERVER e SQLBASE. O meta-sistema permite a seleção do idioma a ser utilizado pelo sistema de medição gerado.

Estamos concluindo a criação de *drivers* e uma tela de configuração das interfaces externas. Os *drivers* serão programas escritos em Java para automatizar a coleta e a exportação de dados. A tela de configuração de interfaces externas será implementada através do Diagrama de Fluxo de Dados.

Planejamos aplicar o meta-sistema de medição nas empresas incubadas na InfoGene. Esta é uma incubadora de empresas de software, criada em 1997, e mantida pela PUC-Rio. Acreditamos que o ambiente de uma incubadora de empresas de software é rico em situações que servem para testar a flexibilidade e a adaptação de um sistema de medição. Em particular, permite avaliar de forma controlada os impactos do programa de medição sobre pequenas empresas. Os sistemas de medições customizados serão instalados nas empresas incubadas. Esta instalação permitirá avaliar a flexibilidade da abordagem proposta, o grau de utilização dos sistemas de medição, a aplicabilidade da solução no contexto de uma pequena empresa e a interferência do sistema de medição no dia à dia de cada empresa.

9. Conclusões

O aumento da atenção sobre o processo de desenvolvimento de software através da adoção de modelos de maturidade, gerou uma maior demanda de sistemas de apoio a qualidade de software. Neste trabalho, procuramos desenvolver uma solução para sistemas de medição de software caracterizada por:

- Processo de produção flexível;
- Sistemas de medição com interface amigável e integrado ao ambiente de desenvolvimento;
- Baixo custo possibilitando a adoção por pequenas empresas de software.

Na nossa proposta, um sistema de medição é construído a partir de um meta-ambiente de engenharia de software. Os meta-modelos do sistema de medição serão construídos a partir dos meta-modelos de engenharia de software, usufruindo de todo ferramental já implementado. Os sistemas de medição gerados poderão ser desde sistemas completos (captura + armazenamento + análise + divulgação de dados) até sistemas específicos para um dado conjunto de métricas, ou ambientes parciais (ex.: captura), ou partes do sistema (ex.: esquema de dados).

A arquitetura proposta permite que o repositório de medição central, seja povoado diretamente pelo desenvolvedor, ou através da importação dos dados gerados em repositórios remotos. Dessa forma, os ambientes gerados conseguem acompanhar a dinâmica de uma equipe de desenvolvimento. A solução WWW, além de possuir uma interface homem-máquina mais próxima do usuário, traz consigo os benefícios da conectividade e da facilidade de tratar heterogeneidade no contexto da Internet. Como os ambientes instanciados serão ativados por *browsers* da Internet, os resultados da medição poderão ser facilmente disponibilizados, e desta forma, tornados acessíveis pelos integrantes de diversas equipes de desenvolvimento.

Um dos problemas que observamos no nosso trabalho foi que o processo de produção através de meta-sistemas acarretou uma maior dificuldade na correção de erros do sistema de medição gerado. Na abordagem de meta-sistemas, um erro no sistema gerado pode corresponder a: erro na especificação do programa-alvo, erro no programa de geração, ou erro no ambiente de geração.

Uma boa parte do trabalho de medição de software relatado na literatura refere-se a grandes empresas, a grandes equipes de trabalhos, e a grandes sistemas. Nosso trabalho procura focar pequenas empresas, nas quais é inviável um sistema de medição de software nos moldes das grandes corporações.

10. Referências bibliográficas

[1] Alho, K.; *A Process Improveent Experience in Small PC Software Companies* , Master Thesis, Helsinki University of Technology, 1996.

[2] Basili, V.R., R.W. Selby, and D.H. Hutchens, "Experimentation in Software Engineering", *IEEE Transactions on Software Engineering*, 12(6),1986

[3] Basili, V.R. and Rombach, H.D.; "The TAME project: Towards improvement-oriented software environments", *IEEE Transactions on Software Engineering*, 14(6),1988.

[4] Basili, V.R., "The Role of Experimentation in Software Engineering: Past, Current, and Future", *Proceedings of ICSE-18*, 1996.

[5] Branco, C.E.C. e Melo, P.R.S.; "Setor de Software: Diagnóstico e Proposta de Ação do BNDES", *BNDES Setorial*, 5, Março/1997.

[6] Briand, L., Morasca, S. and Basili, V.R., "Property-Based Software Enginerring Measurement", *IEEE Transactions on Software Engineering*, 22(1),1996.

[7] Brodman, J.G. and Johnson, D.L.; "What sall business and organizations say about the CMM", *Proceedings of ICSE-16*, 1994.

[8] Bruckhaus,T., Madhavji, N., Janssen, I. and Henshaw, J., "The Impact of Tools on Software Productivity" , *IEEE Software*, 13(5), 1996.

[9] Bruegge,B., and Dutoit, A., "Communication Metrics for Software Development", ", *Proceedings of ICSE-19*, 1997.

[10] Daich, G.T. and Dawood, M.; "Metrics Tools: Size", *CrossTalk*, April 1995.

[11] Dawood, M. and Kingsbury, J.; "Metrics Tools:Quality – Defect Tracking", *CrossTalk*, May 1995.

[12] Eick, S., Loader, C., Long, M., Votta, L., and Wiel, S., "Estimating Software Fault Content Before Coding", *Proceedings of ICSE-14*, 1992.

[13] Emam, K.E.; Drouin, J., and Melo, W.; *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*, IEEE Computer Press, 1997.

[14] Erickson, D.R. and Steadman, T. A., "Metrics Tools:Effort and Schedule", *CrossTalk*, March 1995.

[15] Fenton, N., Pfleeger, S.L. and Glass, R.; "Science and Substance: A Challenge to Software Engineers", *IEEE Software*, 11(4), 1994.

[16] Fenton, N. and Pfleeger, S.L.; *Software Metrics: A Rigorous & Practical Approach*, ITP, Cambridge, 2° edition, 1996.

[17] Fenton, N. and Hall, T.; "Implementing Effective Software Metrics Programs", *IEEE Software*, 14(2), 1997.

[18] Gaffney, J., Cruickshank, R., Werling, R., and Felber, H.; *The Software Measurement Guidebook*, Thomson Computer Press,1995.

[19] Giles, A.E. and Daich, G.T.; "Metrics Tools", *CrossTalk*, February 1995.

[20] Giles, A.E. and Daich, G.T.; “Universal MetricsTools”, *CrossTalk*, February 1995.

[21] Giles, A.E. and Barney, D.; “Metrics Tools: Software Cost Estimation”, *CrossTalk*, September 1995.

[22] Grady, R.B., “Work-Product Analysis:The Philospher’s Stone of Software”, *IEEE Software*, 7(2), 1990.

[23] Hester, A.M., Borges, R., and Ierusalimschy, R.; “CGILua: A Multi-Paradigmatic Tool for Creating Dynamic WWW Pages”, *Anais do XI Simpósio Brasileiro de Engenharia de Software*, 1997.

[24] Hammer, T., Rosenberg, L., Huffman, L. and Hyatt, L., "Measuring Requirements Testing: Experience Report”, *Proceedings of ICSE-19*, 1997.

[25] Henry, S. and Selig, C., “Predicting Source-Code Complexity at the Design Stage”, *IEEE Software*, 7(2), 1990.

[26] Hudepohl, J., Aud, S., Khoshgoftaar, T., Allen, E., and Mayrand, J.; “Emerald: Software Metrics and Models on the Desktop”, *IEEE Software*, 13(5), 1996.

[27] Humphrey, W.S.; *A Discipline for Software Engineering*, Addison-Wesley, Reading, MA, 1995.

[28] Jones, C.; *Applied Software Measurement*, McGraw-Hill, New York, 1991.

[29] Kitchenham, B., Pfleeger, S.L., and Fenton, N.; “Towards a Framework for Software Measurement Validation”, *IEEE Transactions on Software Engineering*, 21(12),1995.

[30] Kokol, P.; “PROMIS: A Software Metrics Tool Generator”, *ACM Sigplan Notices*, May 1995

[31] Leite, J. et alii.; “Draco-PUC: A Technology Assembly for Domain Oriented Software development”, *III ICSR-IEEE*, Brazil, 1994.

[32] Lima. M.A.A., Barrére, T.S., Prado, A.F. e Couto, A. A.; “Ambiente CASE com Múltiplas Visões de Requisitos e Implementação Automática utilizando o Sistema Transformacional Draco”, *Anais do XI Simpósio Brasileiro de Engenharia de Software*, 1997.

[33] Meira, C.A.A., Costa, C. R. e Masshurá, S.M.; “A Construção de um Gerador de Programas Aplicativos segundo Conceitos de Análise de Domínios”, *Anais do X Simpósio Brasileiro de Engenharia de Software*, 1996.

[34] Offen, R.J. and Xia, G.; *SQUATter-M Conceptual Requirements Document*, JRCASE Research Report, No. 96/13.

[35] Offen, R.J. and Jeffery, R.; “Establishing Software Measurement Programs”, *IEEE Software*, 14(2), 1997.

[36] Ogasawara, H., Yamada, A. and Kojo, M., “Experiences of Software Quality Management Using Metrics through the Life-Cycle”, *Proceedings of ICSE-18*, 1996.

[37] Paulk, M.C., Curtis,B., Chrissis, M.B.;Humphrey, W.S.; *Capability Maturity Model for Software*, CMU/SEI-91-TR-24, Software Engineering Institute, 1991.

[38] Pfleeger, S.L.; “Lessons learned in building a corporate metrics program”, *IEEE Software*, 10(3), 1993.

[39] Pfleeger, S.L., Jeffery, R., Curtis, B., and Kitchenham, B.; “Measuremeny based Process Improvement”, *IEEE Software*, 11(6), 1994.

[40] Pfleeger, S.L., Jeffery, R., Curtis, B, and Kitchenham, B.; “Status Report on Software Measurement”, *IEEE Software*, 14(2), 1997.

[41] Rombach, H.D., “Design Measurement:Some Lessons Learned”, *IEEE Software*, 7(2), 1990.

[42] Royce, W., "Pragmatic Software Metrics for Iterative Development*”, Proceedings of ICSE-19*, 1997.

[43] Schwarting, D.C and Guy, E.; *Guidelines for WWW-based Metrics Automation*, Informal Technical Data, STARS-PV03-A033/001/00

[44] Selby, R.W, Goldstein, A., and Hart, H; “The Amadeus Measurement System: STARS’s Automated, Integrated Approach to Quality, Management, and Process Measurement”, April 1993.

[45] Staa, A.v.; *Manual de Referência:Talisman: Ambiente de Engenharia de Software Assistido por Computador*, 1993.

[46] Tsuda, M., Morioka, Y, Takadachi, M. and Takahashi,M., "Productivity Analysis of Software Development with na Integrated CASE Tool”, *Proceedings of ICSE-14*, 1992.

[47] *The WISE Project Management System Homepage*, http://research.ivv.nasa.gov/ projects/WISE/index.html.

[48] *The Web-Project Homepage*, http://www.wproj.com/index.htm.

[49] *Timelog for the Personal Software Process Homepage,*. http://mats.gmd.de:8080/ clemens/java/timelog/

[50] *journyx Webtime Homepage,* http://www.journyx.com/products.html.

[51] *MetricCenter Homepage*, http://www.distributive.com/mchome.htm.

[52] *Software Metrics Management Information System (SMMIS) Homepage*, http://www. army.mil/swmetrics/smmis.htm.

[53] *NetProblemTracker Homepage*, http://www.netresultscorp.com/topic_home.html.

Apêndice I: Exemplos das telas do sistema de medição instanciado

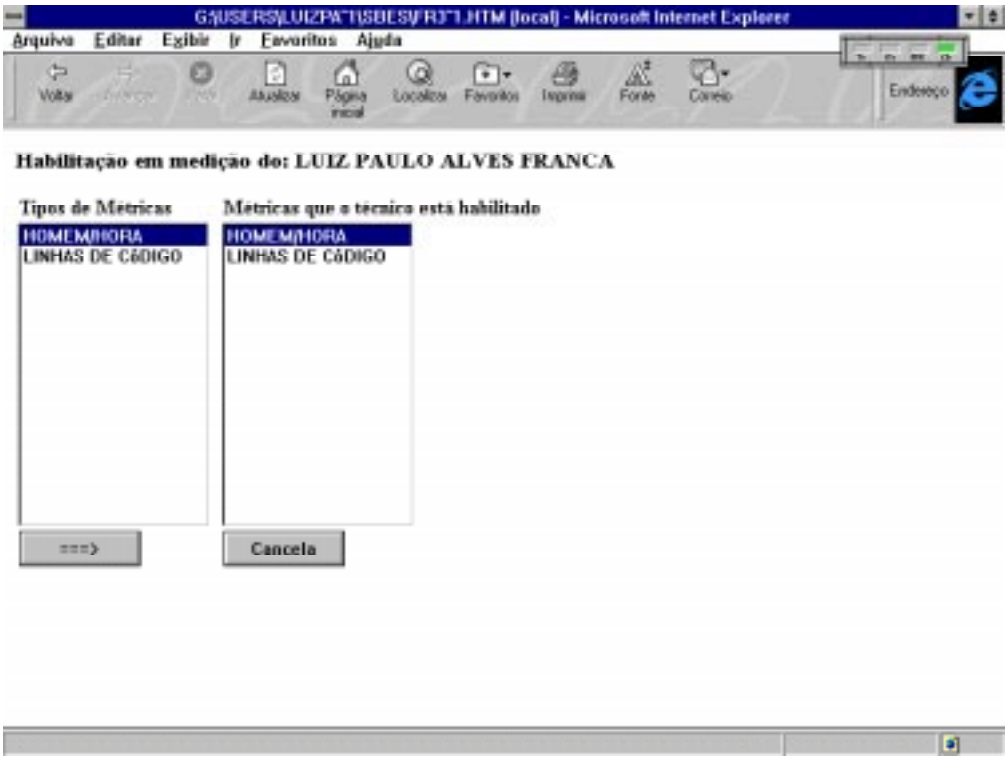

Figura 10. Tela de habilitação de medição

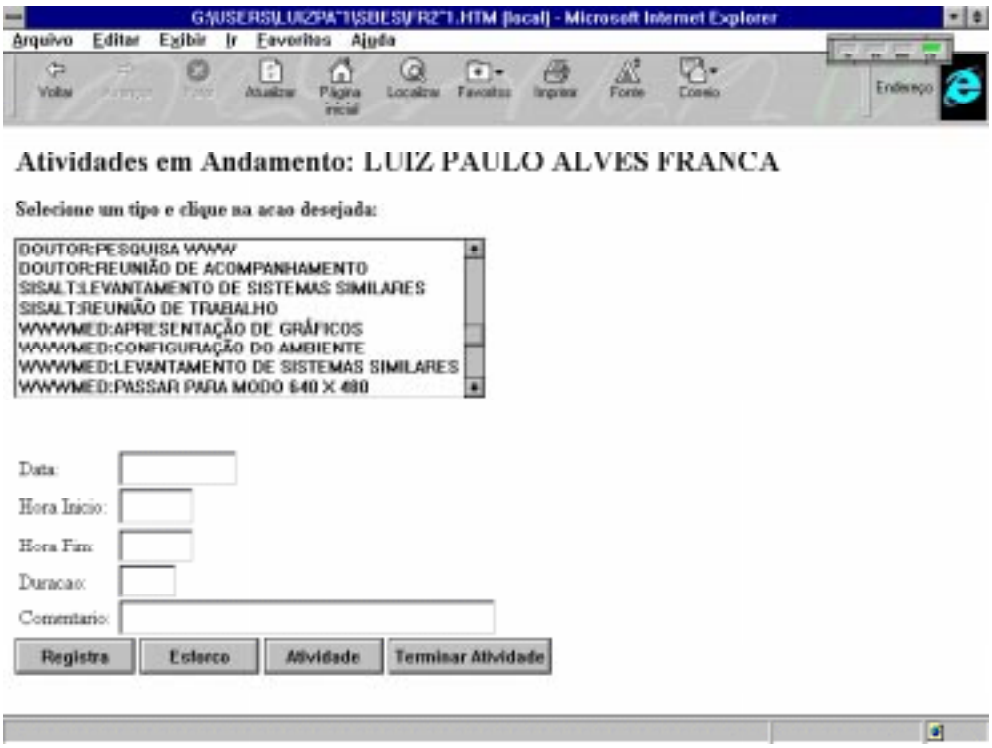

Figura 11. Tela de registro de execução de atividade