COMPONENTE GERENCIADOR DE DADOS CONFIGURÁVEL

Maurício Gonçalves V. Ferreira (Mestre)
Instituto Nacional de Pesquisas Espaciais (INPE)
e-mail: mauricio@gama.ccs.inpe.br

Tatuo Nakanishi (Dr.)
Universidade de Mogi das Cruzes (UMC)
e-mail: tatuo@lac.inpe.br

João Bosco S. Cunha (Dr.)
Universidade de Mogi das Cruzes (UMC)
e-mail: bosco@lac.inpe.br

Resumo

A enorme ênfase em produtividade nesta década tem levado a indústria de software a propor mecanismos que acelerem o desenvolvimento de sistemas de software principalmente utilizando a reutilização. Desta forma a proposição de um Componente Gerenciador de Dados (CGD) Configurável tem por objetivo a sua reutilização por diferentes aplicações evitando-se que as mesmas precisem conter códigos específicos para acessar a base de dados. O aspecto importante da proposta é a substituição da produção de códigos para acessar a base de dados pela configuração de um componente de software durante o desenvolvimento de um sistema. O CGD é portanto uma camada de software que fica entre a aplicação e o banco de dados, facilitando os trabalhos dos desenvolvedores de software nas metodologias orientadas a objeto. O CGD recebe as solicitações do aplicativo para armazenar ou recuperar objetos, e ele realiza esta tarefa, seja interfaceando com um banco de dados relacional ou com um sistema de arquivos.[8]

Palavras-chave: Banco de dados, Configurável, reutilização, Objeto-persistente

Abstract

The great emphasis on the productivity during this decade led the software development industry to adapt processes which speed up the software system development, mainly, by means of software reutilization. Therefore, the proposed Configurable Data Manegemet Component (CGD) aims at its reutilization by several applicatives avoiding the need for them to have a special code to access the databases. An important feature is that CGD can be configured in much a way as to be used by various applicatives. CGD is therefore, a software layer between the applicative and the database, making the object oriented software development process easier. CGD receives requests from an applicative to store or retrive the objects. It performs this task both ways interfacing a relational database or an archiving system.

Keywords: Database, configurable, reusable, persistent object

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

1 - INTRODUÇÃO

O barateamento crescente do hardware e o aparecimento dos microcomputadores pessoais tem levado a tecnologia computacional a mais diferentes categorias de pessoas. propiciando uma demanda crescente por software. Apesar das inúmeras metodologias que surgiram na última década, o desenvolvimento de software continua sendo um processo caro e demorado. Um dos maiores problemas é como aumentar a produtividade das equipes de desenvolvimento, mantendo-se a qualidade do produto. A reutilização de software é uma tecnologia que aponta nesta direção.[10]

A enorme ênfase na produtividade tem levado à necessidade de melhorar a qualidade do software. Esta qualidade assume um novo significado nesta década na medida que se acentua a competitividade na indústria de software. Os usuários finais tem considerado outras características, além da simples ausência de defeitos. Critérios de qualidade, tais como, facilidade de utilização, portabilidade, facilidade de efetuar modificações e principalmente a reutilização, passaram a ser tratados mais rigorosamente.

A reutilização é sem dúvida o critério que vem tendo maior impacto na produtividade. Embora todos tenham falado sobre ela desde os anos setenta, a reutilização de software ainda é pouco explorada pelas empresas. Nos sistemas atuais dificilmente um módulo de uma aplicação é utilizado na confecção de uma outra. Isto acontece porque as aplicações são desenvolvidas sem a preocupação de montar um esquema apropriado visando a reutilização.[15,16]

Com o paradigma de Orientação a Objetos um grande salto foi dado em direção à reutilização. Um enfoque muito grande tem sido dado na produção de bibliotecas de classes que servem de base para a produção de diferentes aplicações. Mas só o fato de se ter os recursos de uma biblioteca de classes ainda não esgota o que se pode fazer na exploração da reutilização.

A reutilização, ainda limitada nos sistemas atuais, pode acarretar a existência de códigos similares em vários módulos do sistema. Assim diferentes módulos poderão estar sendo implementado por trechos de códigos semelhantes para fazer o mesmo tipo de operações como: acesso a Banco de Dados, interface com o usuário, controle das tarefas, etc... [9,14]

Com o objetivo de minimizar o desenvolvimento de software das aplicações para acessar o banco de dados, o presente trabalho mostra que uma camada de software poderá ficar entre a aplicação e o banco de dados. Esta camada pode ser reutilizada por várias aplicações para acessar a base de dados. O detalhe importante é que o desenvolvedor da aplicação trocará a produção de software específica para acessar o banco de dados por serviços disponíveis desta camada.

A origem desta camada veio da proposta de Coad[5], isto porque Coad cria um componente (CGD - Componente Gerenciador de Dados) só para atender as solicitações de acesso a banco de dados. Para o CGD de Coad, cada classe definida dentro da aplicação tem seus métodos próprios para armazenar os objetos pertencentes a esta classe. Já para o CGD proposto as classes definidas na aplicação não terão métodos para armazenar seus objetos, elas utilizarão os serviços de armazenamento disponíveis no CGD. Desta forma então, o CGD passa a ser capaz de armazenar qualquer objeto da aplicação. Uma forma de implementar esta capacidade é construir esta camada de maneira que ela seja configurável, ou seja, que ela possa ser previamente configurada para atender a vários objetos de uma dada aplicação.

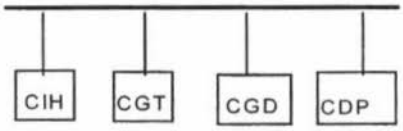
Pode-se dizer que o CGD configurável proposto é uma extensão do modelo de Coad e a opção pela metodologia de Coad foi feita com base nos resultados comparativos entre

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

metodologias obtidos por outros pesquisadores[17].

1.1 - Modelo de sistema de software proposto por Coad

Coad [5] sugere que as responsabilidades do sistema sejam modeladas e representadas no Componente Domínio do Problema (CDP). Por exemplo em um sistema de *Folha de Pagamento*, as classes e os objetos com as responsabilidades para *Calcular o salário, Cadastrar funcionários, Calcular deduções* ,estarão no CDP. Coad sugere a criação do Componente Interface Humana (CIH) para o CDP interfacear com o usuário. Coad propõe a criação do Componente Gerenciador de Dados (CGD) para o CDP armazenar seus dados permanentes. Na aplicação Folha de Pagamento os objetos permanentes são por exemplo *funcionários*, *impostos*, etc... Coad propõe a criação do Componente Gerenciador de Tarefas (CGT) com a finalidade de gerenciar as tarefas do CDP. Para a aplicação Folha de Pagamento as seguintes tarefas precisam ser gerenciadas: *Calcular o salário, Cadastrar funcionários, Calcular deduções, etc...* Veja Fig.1.1.



CIH - Componente Interface Humana

CGT - Componente Gerenciador de tarefas

CGD - Componente Gerenciador de dados

CDP - Componente Domínio do Problema

Fig. 1.1 Modelo de sistema de sofware proposto por Coad

A divisão de um sistema de software em componentes facilita a reutilização porque ajuda a organização das atividades de reutilização. Por exemplo, a aplicação, que é representada pelo Componente Domínio do Problema, para fazer a interface com o usuário poderá reutilizar, no seu desenvolvimento, as classes já definidas para o CIH. Se a aplicação precisar fazer acesso à base de dados poderá, no seu desenvolvimento, reutilizar classes já definidas para o CGD. Se a aplicação precisar gerenciar tarefas ou processos que vão estar rodando em um PC ou numa Workstation, poderá, no seu desenvolvimento, reutilizar as classes já definidas para o CGT.

Na figura 1.1, a ligação dos 4 componentes é representada por um barramento que simboliza a troca de mensagens entre eles. É através desta troca de mensagens que o CDP poderá utilizar os recursos do CIH para fazer interface com o usuário, poderá utilizar os recursos do CGT para controlar suas tarefas e poderá utilizar os recursos do CGD para fazer acesso à base de dados.

Como o objetivo deste trabalho é apresentar um **Componente Gerenciador de Dados Configurável**, o CGD proposto por Coad será descrito com mais detalhes.

1.2 - O Componente Gerenciador de Dados de Coad

A proposta de Coad[5,6] sugere que cada objeto do CDP tenha um serviço do tipo "salve-me", e esse serviço deve ser herdado de uma classe superior do CGD. Na execução

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

deste método o objeto saberá em qual plataforma ele será armazenado, se é um arquivo, um BDR ou um BDOO.

O exemplo de Coad a seguir mostra uma aplicação bem simples com os seguintes objetos relacionados: cliente, transação, transação de venda, item, e produto. Um *cliente* realiza uma *transação de compra*. Dentro desta transação ele compra *itens, os quais relacionam-se* com *produtos*. Veja Fig. 1.2

Para construir o CGD, Coad propõe que cada um desses objetos tenha os serviços necessários para se armazenar. Estes serviços são ***especializações*** de uma classe superior com o nome de *ObjectTagFormat*.

A classe *ObjectTagFormat* contém serviços do tipo *arm. atributos, arm. conexões* , etc... ,e cada classe do CDP como *cliente, transação, item, produto,* redefinem estes serviços porque evidentemente cada um dos objetos dessas classes tem seus próprios atributos e suas próprias conexões.

Em suma, na proposta de Coad para a implementação do CGD a reutilização ocorre baseada na propriedade de herança dos serviços da classe *ObjectTagFormat,* mas como esses serviços necessitam de redefinições a **reutilização implica ainda na implementação de códigos de programa**.

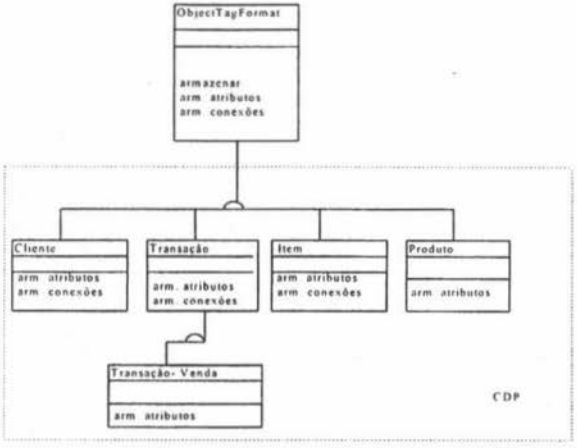


Fig.1.2 Definição dos métodos do CDP para acessar o CGD

1.3 - O objetivo deste trabalho

Para eliminar a necessidade de implementação de códigos para a redefinição dos serviços de armazenamento como visto no item anterior, é necessário eliminar os serviços de armazenamento nas classes *cliente, transação, item, produto e transação de venda* mostrados na figura 1.2, ou seja, os serviços *arm. atributos e arm. conexões*. A idéia consiste basicamente em concentrar em um único **objeto servidor** a capacidade de armazenar

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

qualquer objeto do CDP, de tal forma que, cada objeto não precise utilizar códigos próprios para se armazenar. Este **objeto servidor** é na verdade o topo de uma **camada de software** que ficará entre a aplicação e o Banco de Dados. Essa camada armazenará os objetos *Cliente*, Transação, etc, e terá a responsabilidade de levantar os atributos e as conexões particulares de cada objeto para serem armazenados .

Se esta **camada** for capaz de armazenar **qualquer objeto**, a reutilização será maximizada, porque todos os objetos da aplicação utilizarão esta camada para se armazenar, sem a necessidade de conter códigos para fazer esta operação. Uma forma de implementar esta capacidade é construir a camada de maneira que ela seja configurável, ou seja, que ela possa ser previamente configurada para atender a uma gama de objetos diferentes. Incorporando-se essa camada ao CGD, torna-o também configurável e capaz de identificar qual objeto ele tem que armazenar ou recuperar e fazer a operação. Note-se que o CGD continua existindo, só que agora ele é capaz de saber qual objeto está querendo ser armazenado e quais são os atributos e conexões deste objeto devem ser salvos. Desta forma, os serviços *arm. atributos e arm. conexões* que foram vistos na proposta de Coad, mostrados na fig 1.2, não serão mais necessários.

Tendo-se um CGD configurável a reutilização é maximizada na parte do sistema referente ao armazenamento e recuperação de dados porque não será mais necessário a implementação de códigos para estas tarefas. Ao invés disso, é necessário **configurar o CGD para atender os diferentes tipos de objetos**.

Durante o desenvolvimento de um sistema de software, para qualquer que seja a aplicação, basta então incorporar o CGD e configurá-lo de acordo com os objetos da aplicação que necessitam de armazenamento. Maiores detalhes serão apresentados nos próximos capítulos.

2 - ORIGEM DO TRABALHO

No capítulo 1 foi apresentado o modelo de sistema proposto por Coad. Dentro desse modelo os componentes CIH, CGT e CGD são implementados de acordo com o CDP. O ideal seria ter **sistemas configuráveis** para atender várias aplicações (CDP), ou seja, o CIH, o CGT e o CGD seriam configuráveis para atender tanto o CDP folha de pagamento quanto o CDP da vídeo locadora. Aproveitando a idéia de Coad em dividir um sistema em quatro componentes, Cunha [7] em sua tese de doutorado propõe um passo mais a frente na direção de reutilização, ou seja, um **sistema de software configurável** para várias aplicações. mostrado com mais detalhes no item a seguir.

2.1 - Sistemas configuráveis

Cunha[7] sugere que, no sistema de software proposto por Coad, Fig 1.1, os componentes CIH, CGD e CGT sejam configuráveis de acordo com o CDP que estivesse sendo executado naquele momento. Veja fig 2.1.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

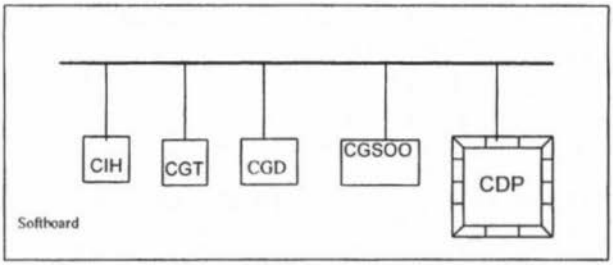


Fig. 2.1 Proposta de Cunha para sistemas configuráveis

De acordo com a proposta de Cunha, para um componente ser **configurável** ele tem que se adaptar à aplicação que está sendo executada naquele momento. O CDP continua o mesmo proposto por Coad, ele é uma aplicação como por exemplo uma folha de pagamento, um software para vídeo locadora, etc... A única diferença é que esta aplicação pode ser trocada, ou seja, retirar o software para vídeo locadora e inserir o software folha de pagamento, de tal forma que, os outros componentes se adaptem a esta mudança. Se por exemplo, o CDP fosse uma aplicação para folha de pagamento, o CIH teria que mostrar as telas para cadastrar um funcionário, o CGT iria gerenciar tarefas do tipo calcular salário bruto e o CGD iria armazenar e manter dados dos funcionários existentes na empresa. Se o CDP fosse um software para vídeo locadora o CIH iria mostrar telas para cadastrar clientes, o CGT iria gerenciar tarefas do tipo *verificar quais fitas foram alugadas* e o CGD iria armazenar e manter os dados dos clientes da vídeo locadora.

Desta forma, o sistema de software configurável seria similar a um **softboard** onde se teria o CDP que seria um "chip" que poderia ser trocado. Esta troca nada mais é do que a troca da aplicação, enquanto todos os outros componentes , CIH, CGD e CGT se adaptariam a esta nova aplicação.

Para que cada componente se adapte é preciso ficar registrado em algum lugar que, quando for trocado a aplicação vídeo locadora pela aplicação folha de pagamento uma nova interface (CIH) terá que ser mostrada, novas tarefas serão gerenciadas (CGT) e novos dados serão mantidos (CGD). O responsável por manter o registro e a ligação da aplicação com os componentes CIH, CGT e CGD é o novo componente introduzido, ou seja, o CGSOO (Componente Gerenciador de Sistemas Orientados a Objetos).

2.2 - Componente Gerenciador de Sistemas Orientado a Objeto (CGSOO)

O CGSOO é a sugestão de Cunha para que o sistema de software seja configurável. É através dele que são armazenadas as informações necessárias de uma aplicação para que a mesma possa utilizar o ambiente configurável proposto. Estas informações são chamadas de **metadados** e são armazenadas e mantidas no próprio banco de dados pelo CGSOO. Como um exemplo pode-se citar uma aplicação (CDP) Folha de pagamento que utiliza o sistema configurável proposto. Para que isto seja possível é necessário conhecer os **metadados** da aplicação Folha de Pagamento que são:

- As telas que a aplicação utiliza. Como exemplo: Tela1 = Cadastrar Funcionário e Tela2 = Rel. de Horas Trabalhadas. Estes **metadados** serão utilizados pelo CIH.
- As tarefas existentes na aplicação. Exemplo: Tar1= Cal. Sal. Bruto e Tar2 = Cal. Imp. de

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompress

Renda. Estes **metadados** serão utilizados pelo CGT.
- As classes/objetos pertencentes à aplicação. Exemplo: Funcionário, Departamento, etc... Para cada um destes objetos quais os seus atributos, quais são seus métodos, com quais objetos ele mantém conexão, e onde ele está armazenado. Estes **metadados** serão utilizados pelo CGD.

Estes **metadados** são armazenados no banco de dados pelo CGSOO. A tabela a seguir mostra um exemplo destes metadados:

	CGD					CGT	CIH
Aplicação	Classes	Atributos	Conexões	Métodos	Disp. Arm	Tarefas	Telas
Folha_pag	Funcionario	nome endereço telefone	depto	get_nome get_end get_tel	TAB_FUNC	TAR1 TAR2	TELA1 TELA2
	Depto	nome local	funcionario	Get_nome Get_local	TAB_DPTO		

tabela 2.1 - Exemplo do conteúdo dos metadados gerenciados pelo CGSOO

Baseado nestas informações armazenadas pelo CGSOO observa-se que :
- O CGD é configurável porque é possível alterar para cada aplicação os seguintes aspectos: suas classes/objetos, atributos de cada objeto, conexões, métodos e o nome do dispositivo onde será armazenado o objeto;
- O CGT é configurável porque é possível alterar para cada aplicação quais são suas tarefas:
- O CIH é configurável porque é possível alterar para cada aplicação quais telas ela utiliza.

Metadados do CGD mantidos pelo CGSOO

O enfoque deste trabalho é somente com os metadados que o CGSOO irá gerenciar para permitir que o CGD possa ser configurável, ou seja, para que o CGD possa ser utilizado por outras aplicações. Estes metadados são:
- o nome da aplicação que está sendo executada;
- as classes contidas nesta aplicação;
- os objetos destas classes;
- os atributos destes objetos;
- os métodos que acessam estes atributos (**métodos de busca**);
- os métodos que atribuem valor a estes atributos (**métodos de atribuição**);
- as conexões deste objeto com outros objetos;
- o nome do dispositivo onde está armazenado este objeto, podendo ser um arquivo, um BDR ou um BDOO [7].

Para o sistema de software proposto por Cunha ser configurável, ele necessitará que **metadados** de cada aplicação sejam armazenados no banco de dados via CGSOO. O objetivo de se ter um sistema de software configurável para várias aplicações só terá êxito se, na troca de uma aplicação por outra, todos os **metadados** da nova aplicação estiverem armazenados e mantidos no banco de dados via CGSOO. Isto porque na carga para a execução da nova aplicação as tarefas que compõem esta aplicação devem ser ativadas, as interfaces que esta aplicação mantém com o usuário devem ser carregadas e os dados permanentes que são os objetos que esta aplicação manipula devem estar disponíveis.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

3 - CGD CONFIGURÁVEL

3.1 - Objetivos do CGD

- O CGD configurável é uma camada que existe fazendo interface entre o banco de dados e a aplicação, oferecendo para a mesma os seguintes serviços para manipular seus objetos:
1 - armazenamento;
2 - recuperação e
3 - exclusão
Observação: Estes serviços são semelhantes aos acessos existentes em qualquer base de dados, a diferença é que a aplicação não precisa implementar código para armazenar um objeto. Ela simplesmente aciona o CGD.

3.2 - Diferenças entre os serviços do CGD proposto por Coad e o CGD configurável são:

-No CGD proposto por Coad é comentado no capítulo 1 os serviços (armazenar, recuperar e excluir) estavam disponíveis para todos os objetos da aplicação , só que, cada objeto tinha que redefini-los e implementá-los, de acordo com os seus atributos internos. Já no CGD configurável proposto, todos estes serviços estarão disponíveis para todos os objetos, sem que os mesmos precisem implementar os métodos para armazenar seus atributos internos. Simplesmente o objeto aciona o CGD que será capaz de identificar quem é o objeto e armazená-lo.

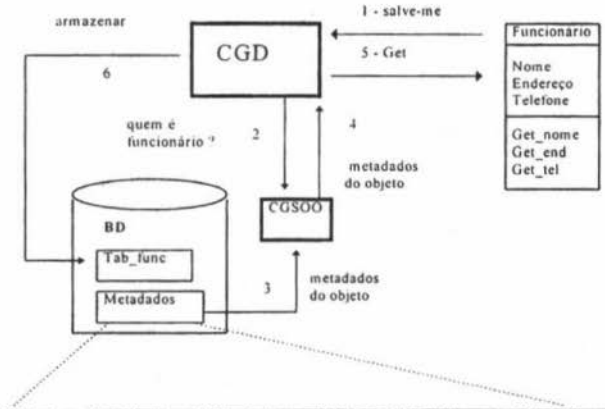
- No CGD proposto pode-se alterar os metadados de um objeto, como por exemplo, o nome do dispositivo onde ele será armazenado , seus atributos, etc..., sem precisar modificar códigos da aplicação onde este objeto está inserido. O sistema de armazenamento de dados fica transparente para o CDP (de uma aplicação).

3.3 - Funcionamento do CGD Configurável

Para se ter um CGD configurável é necessário que ele conheça os **metadados** da aplicação que está sendo executada naquele momento. O CGD é capaz de armazenar os objetos desta aplicação sem que os mesmos precisem conter códigos para fazer esta operação. O objeto simplesmente aciona o CGD através de uma mensagem para armazená-lo. O CGD é então o responsável por descobrir quem é este objeto, quais são seus atributos, seus métodos de busca a estes atributos e quais são seus relacionamentos ou conexões com outros objetos. Para recuperar estas informações o CGD aciona o CGSOO e recupera os metadados do objeto em questão. Com estes metadados o CGD tem então condições de resgatar este objeto da memória e transportá-lo para um banco de dados .

Como exemplo, dentro de uma aplicação folha de pagamento, o objeto *funcionário* com os atributos (nome, endereço e telefone) quer se salvar. Veja fig.3.1.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor



Objeto	Atributo	Met. de busca	Met. de atrib.	Disp_arm	Conexões
Funcionário	Nome Endereco Telefone	Get_nome Get_end Get_tel	Atrib_nome Atrib_end Atrib_tel	Tab_func	Depto
Cliente	Nome CPF	Get_nome Get_CPF	Atrib_nome Atrib_CPF	Tab_cli	Transação

Fig. 3.1 - Passos para armazenar um objeto pelo CGD

Os passos para o objeto se salvar são comentados a seguir:

1 - O objeto funcionário manda uma mensagem para o CGD configurável do tipo salve-me.
2 - O CGD tenta descobrir quem é o objeto funcionário, quais são seus atributos, seus métodos para acessar estes atributos e onde está armazenado este objeto mandando uma mensagem para o CGSOO.
3 - O CGSOO, por sua vez, acessa os ***metadados*** armazenados no banco de dados e entrega para o CGD os metadados do objeto funcionário.
4 - Os metadados do objeto funcionário são: Atributos(Nome, endereço, telefone), métodos de busca (GET-nome,GET-end, GET-tel), métodos de atribuição(ATRIB-Nome,ATRIB-End. ATRIB-tel), o nome do dispositivo onde está armazenado este objeto(TAB-FUNC) e as conexões ou relacionamentos que existem com este objeto(depto).
5 - O CGD com os métodos que acessam os atributos (GET-Nome, GET-end e GET-tel) consegue extrair os dados do objeto funcionário.
6 - O CGD armazena todos estes dados no dispositivo TAB_func.

3.4 - Análise e Projeto do CGD Configurável

Com o propósito de facilitar o entendimento do modelo resultante da análise. é mostrado as responsabilidades básicas do CGD.
1- Definir uma interface com a aplicação (CDP) oferecendo para a mesma os serviços comuns para acesso a um banco de dados tais como: armazenar, alterar e excluir. Implementar esses serviços, tirando então a responsabilidade da aplicação de conter código para acessar a base de dados.
2- Definir uma interface com a aplicação para recuperar na memória o objeto que está sendo armazenado, ou seja, quando for solicitado o armazenamento de um objeto, o CGD terá que

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

descobrir quais são os **metadados** deste objeto para depois buscá-lo na memória.
3- Definir uma interface com o banco de dados para armazenar os objetos da aplicação.
4- Definir uma interface com o CGSOO para recuperar os metadados de um objeto.
5- Tornar transparente para a aplicação qual o sistema de armazenamento está se utilizando(se é um arquivo, um BDR ou um BDOO)

Para atender a primeira responsabilidade cria-se uma classe interface de entrada com a aplicação *(Int. de entrada)*.

Para atender a segunda responsabilidade cria-se uma classe interface de saída com a aplicação *(Int. de Saída)*.

Para atender a terceira responsabilidade cria-se uma classe interface com o banco de dados *(Int. com BD)*.

Para atender a quarta responsabilidade cria-se uma classe interface com o CGSOO *(Int. com CGSOO)*.

Para atender a quinta responsabilidade cria-se três classes abaixo da classe **Int. com BD** que são:

Int. ARQ - Para fazer interface com arquivos;
Int. BDR - Para fazer interface com Banco de dados Relacionais;
Int. BDOO - Para fazer interface com Banco de Dados Orientado a Objeto

No nível de projeto, o ambiente onde foi implementado um protótipo do CGD é o DELPHI 2.0, portanto foi necessário incluir mais uma classe a **TApplication** que encapsula toda a aplicação. Todas as classes definidas no CGD passam a herdar recursos do windows através da Tapplication. Desta forma o CGD no nível de análise/projeto da metodologia Coad/Yourdon[4,5] é mostrado na fig. 3.2

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

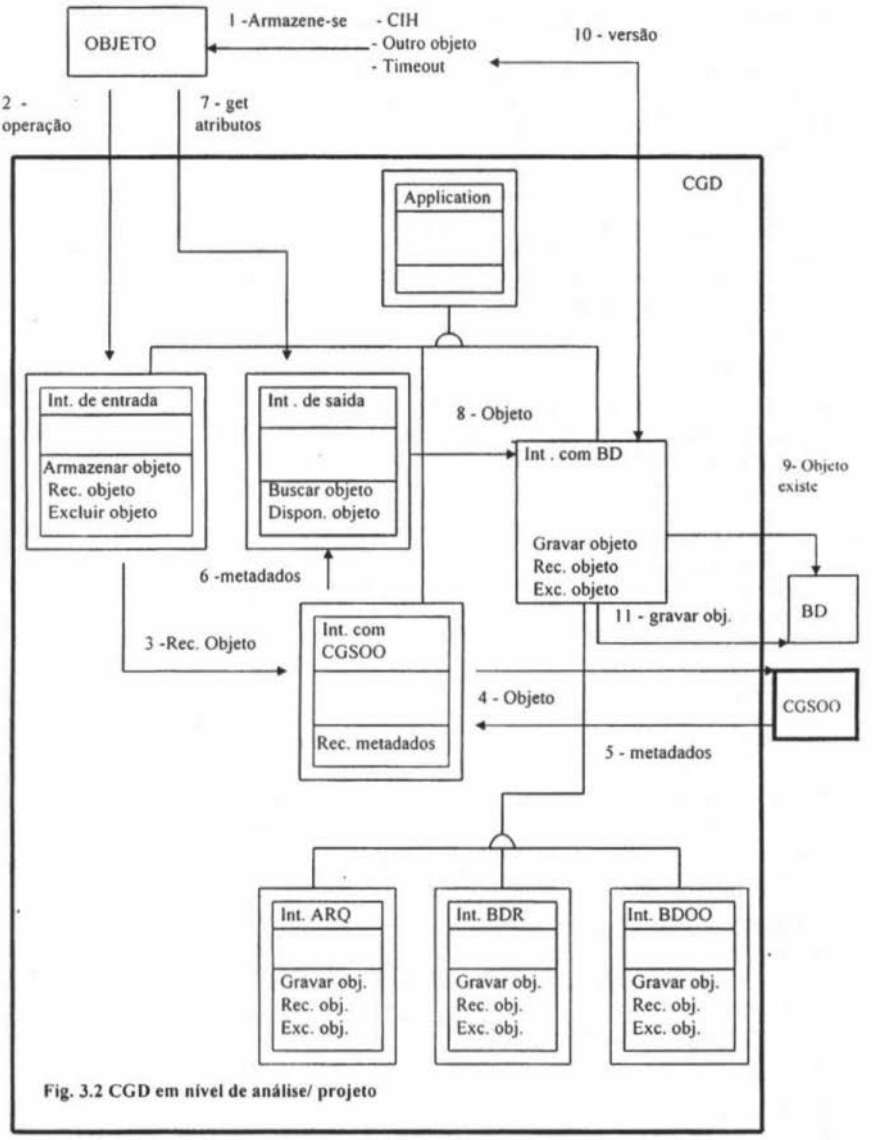


Fig. 3.2 CGD em nível de análise/ projeto

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

3.5 - Dinâmica do fluxo de mensagens entre as classes do CGD

1 - **Armazene-se:** O CIH (Componente Interface Humana), um outro objeto, ou um temporizador, vai solicitar para um objeto para ele se armazenar. Este objeto por sua vez não precisa conter nenhum código para fazer esta operação, basta ele acionar o CGD. Então é disparado a mensagem 2 - operação.
Exemplo: Armazene o objeto *funcionário*.

2 - **Operação:** Esta mensagem é originada de um objeto que pode estar solicitando as seguintes operações: armazenamento, recuperação ou exclusão. A classe **Int. de entrada** não tem todas as informações do objeto necessárias para a execução da operação. Ela precisa do auxílio da classe **Int. com CGSOO** para descobrir quem é o objeto que está querendo, por exemplo, se armazenar. Desta forma então é disparado a mensagem 3 - Rec. objeto.

3 - **Rec. objeto:** Esta mensagem aciona o serviço *Rec. metadados* da classe **Int. com CGSOO**. Este serviço é o responsável por recuperar os metadados armazenados no banco de dados pelo CGSOO. Para isto, esta classe dispara a mensagem **4 - objeto** para o banco de dados e pela mensagem **5 - metadados** volta os metadados do objeto. Como exemplo, os metadados do objeto funcionário são:
- seus atributos: nome, endereço e telefone;
- os métodos de busca : Get_nome, Get_end e Get_tel;
- está armazenado no dispositivo TAB_FUN
Depois de conhecer os metadados do objeto é disparado a mensagem 6 - metadados para a classe **int. de saída.**

6 - **metadados:** Ao chegar esta mensagem para a classe **Int. de saída,** esta classe conhecerá todos os atributos e métodos de busca do objeto que está querendo se salvar. Através dos métodos de busca ela consegue extrair os dados do objeto. Para isto, é disparada um nova mensagem **7-Get-atributos.**

7 - **Get-atributos:** Esta mensagem vai conter realmente a parte de dados de um objeto. Para o objeto funcionário pode estar armazenado nesta mensagem os seguintes dados referentes a nome, endereço e telefone respectivamente: "José da Silva", "Rua dos atletas, n: 123 ", "321-2323"

8 - **Objeto:** Esta mensagem vai ser simplesmente passada para a classe **Int. com BD** contendo:
- a parte de dados do objeto, no caso, informações colhidas com a mensagem anterior (7 - Get_atributos)
- os metadados deste objeto que contém o nome do dispositivo onde ele deve ser armazenado.
Exemplo: "José da Silva", "Rua dos atletas, n: 123 ", "321-2323" Dispositivo= TAB_FUNC

9 - **objeto existe:** Esta mensagem vai conter o identificador do objeto para verificar se ele já existe no banco de dados. Caso exista, é retornado a versão que se encontra este objeto. A padronização escolhida é:
V0 - é a primeira versão
V1, V2 ... São versões posteriores...

10 - **Versão:** Esta mensagem é disparada para aplicação em duas situações:

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

- Primeiro quando está se armazenando um objeto e já existe uma cópia dele no banco de dados. O CGD consulta a aplicação para a mesma definir se irá ou não gerar uma nova versão do objeto.
- Num processo de recuperação ou exclusão de um objeto podem existir várias versões do mesmo. É necessário uma interação do CGD com a aplicação para decidir qual versão será recuperada ou excluída respectivamente

11 - Gravar objeto: Esta mensagem vai conter a parte de dados do objeto mais sua versão.

4 - CONCLUSÕES

A implementação de um CGD configurável para várias aplicações dentro do ambiente de sistema de software proposto por Cunha no capítulo 2, pode trazer os seguintes benefícios:

1- Reutilização de Software: Isto porque esta camada feita sobre o banco de dados pode ser utilizada por diferentes aplicações, bastando para isto que os metadados de cada aplicação sejam armazenados no banco de dados via CGSOO. O CGD foi concebido de maneira a ser o mais genérico possível, haja visto os serviços que ele disponibiliza para a aplicação como: armazenar, recuperar e excluir objeto .

2- Aumento de produtividade: Da mesma forma que hoje os ambientes visuais existentes no mercado como Visual basic, Visual C++, DELPHI, etc... diminuem o tempo gasto do desenvolvedor de software para construir interfaces com o usuário, o CGD poderá ser utilizado pelos desenvolvedores reduzindo o tempo gasto no desenvolvimento de interfaces com banco de dados, consequentemente aumentando a produtividade.

3 - Transparência de ambiente: Para quem está utilizando uma metodologia orientada a objetos poderá ficar transparente o tipo de base de dados que está se utilizando, se é um arquivo ou um BDR . A aplicação manipula objetos e entrega para o CGD objetos para serem armazenados. O CGD se incumbe de transformar o objeto manipulado em um registro, se a forma de armazenamento for um arquivo, transformar o objeto em uma tupla de uma tabela, se a forma de armazenamento for um BDR.[1,2]

4 - Não há necessidade de implementar o código de acesso ao banco de dados: Todas as aplicações que forem cadastradas no banco de dados via CGSOO podem acionar os serviços do CGD para acessar o banco de dados, sem precisar possuir códigos para armazenar seus objetos. Passaria para o CGD como parâmetro simplesmente o objeto a ser armazenado, recuperado ou excluído.

5 - Controle de versões: Os BDOOs são tecnologias que apontam nesta direção, ou seja, permitem que se tenham mais de uma versão de um mesmo objeto. Apesar de não ter sido implementado, estas versões podem facilitar o acesso da aplicação a estados anteriores ou alternados de objetos.[11,13]

6 - Comparação do CGD com o OMG(0bject Management Group). O CGD segue a orientação geral proposta pelo OMG, ou seja, o CGD disponibiliza para a aplicação serviços para armazenar e recuperar seus objetos do banco de dados.[18]

7 - Implementação / Protótipo. A implementação do protótipo do CGD foi feita utilizando

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

DELPHI 2.0, mas somente foi desenvolvido interface com um BDR (Paradox).

Desenvolveu-se um programa capaz de simular 3 aplicações(CDPs) diferentes: Controle de Estoque, Vídeo Locadora e Folha de Pagamento. Para cada uma destas aplicações foram escolhidos os principais objetos que fazem parte do domínio do problema.

Foi implementado um protótipo parcial do CGSOO para que o CGD pudesse consultar os **metadados** de cada aplicação que fosse utilizá-lo. Veja fig. 4.1

Com este ambiente implementado conclui-se que: O CGD pode ser utilizado por várias aplicações e no desenvolvimento de cada aplicação não será necessário produzir novos códigos de programa específicos para acessar o banco de dados

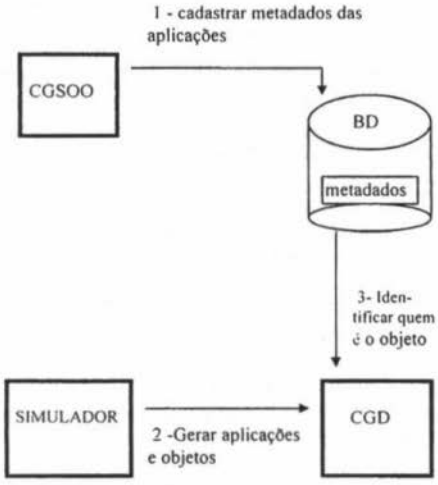


Fig. 4.1 Ambiente implementado

8- Pesquisas desenvolvidas: Existe hoje um empenho de um grupo de pesquisadores do INPE(Instituto Nacional de Pesquisas Espaciais) e da UMC(Universidade de Mogi das Cruzes) para o desenvolvimento de um protótipo da **Softboard** proposta por Cunha. Esforços tem sido feito para viabilizar esta implementação e o desenvolvimento de um CGD configurável é um deles.

9 - Conclusão Final: Com o desenvolvimento de um do CGD configurável é possível trocar o desenvolvimento de código de programa para acessar o banco de dados pela tarefa de armazenar em uma base de dados as informações (**metadados**) dos objetos que devem ser mantidos no banco de dados.

5 REFERÊNCIAS BIBLIOGRÁFICAS

[1]-Blaha, M.;Premerlani,W.; Shean, H. **Converting OO Models into RDBMS Schema**. IEEE Software, Vol(11), Issue-3, pag.28-39, may-1994

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

[2]- Carey, M. ; Haas L. **Extensible Database Management Systems**. SIGMOD RECORD. Vol(19), Number-4, pag.54-69, december -1990.

[3]-Cattell, R.G.G. **A Standard for Object-Oriented DBMS**. SIGMOD RECORD, Vol(23). Number-2, pag.480-482 ,june-1994

[4]-Coad, P.; Yourdon, E. **Analise Baseada em objetos**. 2ed. Editora Campus, Rio de Janeiro 1992.

[5]-Coad, P.; Yourdon, E. **Projeto Baseado em objetos**.
1ed. Editora Campus, Rio de Janeiro 1993.

[6]-Coad,P.; Nicola, J. **Object-Oriented Programming**. Yourdoun Press. 1993.

[7]-Cunha, J.B.S. **Uma Abordagem de qualidade e produtividade para o desenvolvimento de sistemas de software complexo utilizando a arquitetura de placa de software-Softboard**. Tese de Doutorado em Computação Aplicada., INPE - 1997.

[8]-Fafchamps, D. **Organizational Factors and Reuse.** IEEE Software, September-1994.

[9]-Frakes, W. ; Isoda S. **Sucess Factors for Systematic Reuse**. IEEE Software, Vol(11). Issue-5, pag.14-22 September-1994

[10]-Henninger S. **Using Iterative Refinement to find Reusable Software**. IEEE Software, Vol(11). Issue-5, pag.48-60 September-1994

[11]-Khoshafian,S. **Banco de Dados Orientado a Objeto**. Editora Infobook S.A. Rio de Janeiro, 1994.

[12]-Korth, H. F.; Silberschatz, A. **Sistemas de Banco de Dados**. Makron Books 1994.

[13]- Kulkarni, K. G. **Object-Oriented Extensions in SQL3**. SIGMOD RECORD , Vol(23). Number-2, pag.478-479 june-1994

[14]-Moore, J. W. **Debate on Software Reuse libraries**. Third International conference on Software Reuse. Brasil 1994

[15] Staringer W. **Constructing Applications from Reusable Components**. IEEE Software, Vol(11), Issue-5, pag.61-69 September-1994

[16]-Wayne, C.L. **Effects of Reuse on Quality, Productivity, and Economics**. IEEE Software, Vol(11), Issue-5, pag.23-30 september-1994.

[17]- Copyright the Object Agency. Inc. 1995. "**A Comparison of Object-Oriented Development Methodologies**".
http://www.ipipan.gda.pl/~marek/objects/TOA/OOMethod/mcr.html

[18] - Object Managemet Group (OMG) . http://www.objs.com/survey/omg.htm

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor