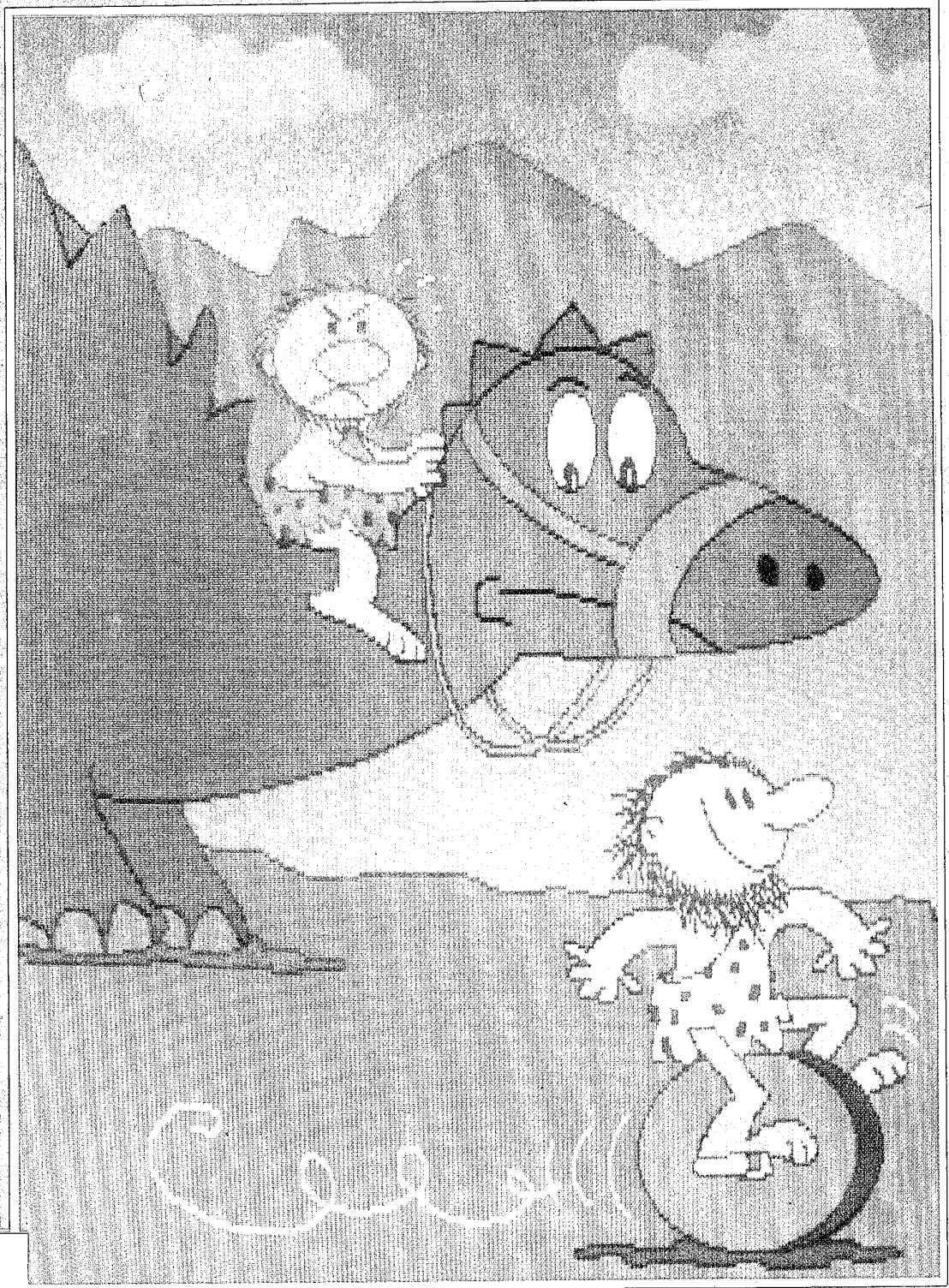
ANAIS

XXI CONGRESSO NACIONAL DE INFORMÁTICA



SUCESU'88

RIO DE JANEIRO

FOCO

SUCESU'88

XXI CONGRESSO NACIONAL DE INFORMÁTICA

Rio de Janeiro de 22 a 26 de agosto de 1988

ANAIS

VOLUME 1

Sociedade dos Usuários
de Computadores e Equipamentos
Subsidiários - RJ
Rua do Carmo, 57/6º and.
Rio de Janeiro - CEP 20011
Tel.: (021) 221-5183
Telex.: (21) 32522 - SUSU-BR

FOCO
Feiras, Exposições
e Congressos Ltda.
Rua da Ajuda, 35/7º and.
Rio de Janeiro - CEP 20040
Tel.: (021) 210-3237
TELEX.: (21) 21864 - FOCO-BR
FAX.: (021) 533-0892

ADEQUANDO METODOLOGIAS DE ANÁLISE ESTRUTURADA A ORIENTAÇÃO A OBJETOS

AUTORES:
HÉLIO BRUCK ROTENBERG
ARNDT VON STAA

ENDEREÇOS:
HÉLIO BRUCK ROTENBERG
CEFET-PR - Departamento de Informática
Rua Sete de Setembro, 3155
CEP 80230 - CURITIBA - PR
Tel.: (041) 224-5333 - ramal 180
ARNDT VON STAA
PUC-RJ - Departamento de Informática
Rua Marquês de São Vicente, 225
CEP 22453 - RIO DE JANEIRO - RJ
Tel.: (021) 529-9523

CURRICULUM VITAE:
HÉLIO BRUCK ROTENBERG — Mestre em Informática (PUC/RJ - 1987), Engenharia de Software e Sistemas de Informação, Professor Assistente do Dept.º de Informática do CEFET/PR.
ARNDT VON STAA — PhD em Ciência da Computação, Engenharia de Software, Ambientes de Desenvolvimento e Controle de Qualidade de Software, Professor Associado do Dept.º de Informática da PUC/RJ.

RESUMO:
Propõe-se um modelo do processo de desenvolvimento de sistemas de programação que aproveita os benefícios apresentados pela Programação Orientada a Objetos. A descrição do modelo abrange atividades do desenvolvimento desde a fase de especificação até a fase de projeto, sendo esta descrição ilustrada por um exemplo.

PALAVRAS-CHAVE:
Programação Orientada a Objetos, Engenharia de Software, Metodologias de Projeto.

INTRODUÇÃO

Vários ideais da Engenharia de Software, como a abstração de dados, polimorfismo e mecanismo de reutilização de software são propostos pelo estilo de programação próprio da Programação Orientada a Objetos — uma evolução em relação ao estilo tradicional de se programar. Para que possamos aproveitar estes benefícios e melhorar a efetividade na produção de sistemas de programação, o processo de desenvolvimento de sistemas deve sofrer adequações.

O modelo para este processo que tenta suportar e explorar estes possíveis benefícios é proposto. A figura 1 ilustra as representações que aparecem nas diversas fases do desenvolvimento, da fase de especificações até a fase de projeto, no modelo proposto.

A FASE DE ESPECIFICAÇÃO

A fonte de informação da fase de especificação, geralmente, constitui-se do mundo da aplicação, logo totalmente informal e não estruturado. O modelo propõe os diagramas de fluxo de dados e suas decomposições hirárquicas para representar o mundo da aplicação [3] e a partir deles, a obtenção do "modelo de objetos" do sistema.

Obs.: Modelo de objetos de um sistema é um modelo do sistema expresso através de objetos e de fluxos de dados e mensagens entre este objetos (vide exemplo na figura 5).

A escolha dos DFDs foi motivada pelo uso corrente deste tipo de representação, pelo particionamento hierárquico do sistema por eles possibilitado e pela facilidade de "tradução" destes diagramas para o modelo de objetos.

Esta tradução está baseada na idéia de que os objetos do modelo serão originados dos depósitos de dados, em conjunto com os processos correspondentes. A correspondência deve ser identificada ao olhar-se os depósitos de dados como futuros objetos, que poderiam requerer e sofrer ações. A análise da correpondência de processos em relação a objetos deve ser feita no nível mais detalhado de cada processo. Por exemplo, se determinado processo X aparecer decomposto nos processos X1, X2 e X3 num DFD mais detalhado, são os processos X1, X2 e X3 que devem ser analisados. Propomos o seguinte método para a transformação:

Obs.: Usaremos o termo objeto para designar tanto um objeto particular como para designar um objeto genérico (classe de objetos).

a. Identificam-se os objetos:

"Os depósitos de dados que aparecem nos DFDs são candidatos a darem origem a objetos."

Os depósitos de dados transitórios possivelmente não originarão objetos. Após a identificação das operações correspondentes a cada depósito de dados (item b), deve-se analisá-los identificando aqueles que não possuem operações correspondentes que lhes dê autonomia para ser um objeto. Estes passarão a fazer parte de outro objeto. Além da identificação dos objetos deve-se fazer uma listas de seus possíveis atributos.

b. Identificam-se as operações (ações) de cada objeto

"Analisam-se os processos em seu nível mais detalhado identificando a que objetos correspondem, isto é, de que objetos podem ser considerados como ações por eles requeridas ou sofridas."

Se determinado processo em seu nível mais detalhado corresponder a mais de um objeto deve-se tentar decompô-lo em processos menores de maneira a poder acomodá-los nos diversos objetos envolvidos.

Se determinado processo originário dos DFDs ou da decomposição proposta no parágrafo anterior não corresponder à ação de nenhum objeto, um novo objeto deve ser criado para acomodá-lo. Existem duas situações em que esta não correspondência ocorre:

— O processo corresponde a ações de uma entidade externa. Neste caso o objeto que vai ser criado para acomodar o processo recebe o nome desta entidade externa.
— O processo tem autonomia. Neste caso o objeto teria o nome do próprio processo.

Neste ponto deve-se controlar a coesão [6] de cada objeto, notando principalmente ausência de operações nos objetos. Constata-se que nos objetos com características passivas geralmente ficam faltando operações de consulta [2].

c. Representam-se objetos, suas operações e as entidades externas.
d. Completa-se a representação com os fluxos de dados:

Analisam-se os DFDs e o modelo de objeto que obtivemos de maneira a identificar os fluxos de dados entre objetos e entre objetos e entidades externas. Os relacionamentos entre os processos serão indicados pelos fluxos de dados entre as operações dos objetos. O mesmo acontece entre processos e entidades externas que terão seus relacionamentos indicados por fluxos de dados entre operações dos objetos e entidades externas.

Nos processos que foram desmembrados para serem acomodados em diversos objetos deve-se estabelecer fluxos de dados de maneira a simular o processo original.

Obs.: a ausência de alguma operação em determinado objeto para acomodar o recebimento de algum fluxo de dados indica que devemos controlar a coesão deste objeto.

e. Definem-se as interfaces dos objetos

"Operações que não recebem fluxos de dados ou só os recebem do objeto a que pertencem são candidatas a serem transformadas em operações privativas ao objeto."

Analisam-se quais das operações existentes em cada objeto não necessitam fazer parte de sua interface, transformando-as em operações privativas ao objeto.

f. Avaliação final.

Todas as atividades devem ser avaliadas. Já indicamos alguns casos em que o controle de coesão deve ser feito. Mas um controle global desta fase do processo é possível e necessário. Pode-se fazer o controle da coesão do projeto como um todo propiciando que avaliemos as especificações sob um outro ponto de vista — não mais o dos diagramas de fluxo de dados. Permite que avaliemos a especificação sob o ponto de vista do modelo de objetos. Permite também a comparação entre os dois modelos.

Obs.: A identificação de objetos não se restringe à fase de especificação. A medida que avançamos no processo de desenvolvimento e começamos a tratar com aspectos do mundo computacional novos objetos podem surgir, o que reafirma a não lineriedade do processo de desenvolvimento [1].

A DEFINIÇÃO DA ARQUITETURA

Como o sistema já tem sua estrutura apresentada pelo modelo de objetos, a fase de arquitetura trata somente de aspectos como o da definição do escopo do sistema a ser implementado e da possível divisão do sistema em subsistemas. O resultado desta fase é o "modelo de objetos limitado".

Propomos também uma outra atividade para esta fase, após a obtenção do "modelo de objetos limitado" — a definição dos tipos de interfaces entre os objetos e as entidades externas (agora externas ao modelo de objetos limitado) e entre os subsistemas (caso existam). Procura-se representar estas interfaces como sendo objetos independentes. Estes objetos se relacionariam como os objetos do modelo e com as entidades externas, mantendo assim a independência entre o aplicativo e a interface.

A FASE DE PROJETO

Como resultado desta fase, pretende-se obter o "modelo de objetos a ser implementado", modelo este que considera as possíveis hierarquias entre objetos e a possível reutilização de classes já existentes na biblioteca. Além disto apresenta os fluxos de controle entre objetos e entre objetos e entidades externas através da representação das trocas de mensagens.

Obs.: a hierarquia que consideraremos será a que representa o mecanismo de herança simples [4].

Para que se atinja este modelo:

a. analisa-se o modelo obtido na fase anterior procurando identificar:
   — a possibilidade de um objeto genérico (classe de objetos) não estar representando todas as suas hipotéticas instâncias.
   — a existência de similaridade entre os objetos, isto é, objetos do modelo estarem representando objetos semelhantes do mundo da aplicação ou mesmo do mundo computacional. Ex.: diversas interfaces do tipo janela, com pequenas diferenças entre si.

Existem duas alternativas para a hierarquização do modelo, a saber:

— a criação de uma superclasse abstrata (que não possui instância) contendo as operações e estruturas de dados comuns a todas as instâncias, e subclasses que acomodam as operações e es-

truturas de dados que se diferenciam nas diversas instâncias.
- a manutenção da classe mais representativa como superclasse e as outras como subclasses.

Para decidir entre uma e outra alternativa, deve-se ter em mente o conceito de coesão de cada classe ou de cada objeto. A decisão nem sempre é clara e muitas vezes é indiferente.

b. consulta-se a biblioteca de classes:

A consulta visa identificar classes que possam ser reutilizadas no sistema que se está projetando. Para isto procura-se identificar semelhanças semânticas e sintáticas entre os objetos do modelo ou seus atributos e classes da biblioteca. Ex: existir uma classe "janela" em determinada biblioteca que seja similar à janela que estamos considerando.

Num segundo passo, procuram-se classes que tenham interfaces similares às dos objetos que estamos considerando no sentido puro de reaproveitamento de código já escrito. Ex: existir uma classe "retângulo" que tenha algumas operações em comum com as janelas que estamos considerando.

Deve-se verificar também a existência de classes que representam o tipo de algum atributo dos objetos que estamos considerando. Ex: existir uma classe "lista" que possa representar o tipo do atributo "lista de itens" da classe "requisição de compras".

c. Integram-se as classes identificadas ao modelo. A integração pode se dar de três formas:
   - uso direto da classe quando esta corresponde exatamente a algum objeto do modelo ou ao tipo de algum atributo,
   - uso de herança quando existem pequenas diferenças, criando ou não subclasses abstratas conforme o item a.
   - atribuindo uma nova instância da classe aos atributos de determido objeto, quando a classe corresponder somente a parte do comportamento do objeto.

d. Representam-se estas informações no modelo, na linguagem da figura 2.

e. Identificam-se as trocas de mensagens entre objetos e entre objetos e entidades externas.

CONCLUSÃO

Foi apresentado um modelo desde a fase de especificação até a fase de projeto de um sistema. Este modelo é completado pela fase de implementação que pode ser subdividida em projetos detalhado dos métodos dos objetos e codificação.

O modelo foi aplicado em projetos de complexidade convencional e tem se mostrado eficiente [5]. Tem propiciado, também, implementações em estruturas similares ao modelo do mundo da aplicação — o desenvolvedor tem conseguido trabalhar com o mesmo tipo de entidades em todas as fases do desenvolvimento. Os sistemas obtidos apresentam alto grau de modularidade e a reutilização de componentes no seu desenvolvimento tem crescido com o aumento da biblioteca.

Em especial, devido à homogeneidade de conceitos apresentada e a existência de heurísticos para as suas várias fases, parece bastante promissor a sua automatização, já se obtendo os primeiros resultados.

REFERENCIAS BIBLIOGRÁFICAS

1. BOOCH, G. *Software Engineering with ADA*. Menlo Park, Benjamin/Cummings, 1983.
2. BOOCH, G. Object-Oriented Development. *IEEE Transactions on Software Engineering. SE12* (2): 211-21, fev.1986.
3. GANE, C. & SARSON, T. *Análise Estruturada de Sistemas*. Rio de Janeiro, Livros Técnicos e Científicos, 1983.
4. GOLDBERG, A. & ROBSON, D. *Smalltalk-80, The Language and Its Implementation*. Menlo Park, Addison Wesley, 1983.
5. ROTENBERG, H. B. *Programação Orientada a Objetos: Um Enfoque de Engenharia de Software*. Dissertação de Metrado, Rio de Janeiro, Depto. de Informática, PUC/RJ, 1987.
6. STAA, A. Em busca de Uma Definição de Coesão. In: *XIV Seminário de Software e Hardware,* Salvador, 1987. *Anais,* 487-96, BA, 1987.

ANEXO 1

A seguir, apresentamos um exemplo da aplicação do processo de desenvolvimento proposto. O exemplo parte dos diagramas de fluxo de dados usado como exemplo em [3] (figura 3).

A partir do diagrama apresentado, obtemos o modelo de objetos do sistema. Mostramos duas representações do modelo completo. A primeira (figura 4) mostra todos os objetos com as respectivas operações, o que a torna uma representação com execessivo número de detalhes. Na segunda (figura 5), são omitidas as operações e detalhes dos fluxos de dados, dando uma melhor visão do sistema como um todo.

Após estas representações completas, foram feitas as representações individuais de cada objeto e de cada entidade externa (exemplos nas figuras 6 e 7). Estas representações individuais permitem que possamos seguir trabalhando com cada uma das entidades em separado, nos abstraindo do restante do sistema. O controle de coesão individualizado fica bastante facilitado.

A fase seguinte é a da definição da arquitetura. Primeiramente a definição do escopo do sistema (figura 8), seguida pela definição das interfaces entre este sistema e entidades externas (exemplo na figura 9).

Na fase de projeto procuramos hierarquizar o modelo, reutilizar classes já existentes e definir os fluxos de mensagens.

Apresentamos a hierarquia entre as interfaces do tipo janela usada no exemplo (figura 10) e um exemplo de definição de fluxo de mensagens (figura 11).

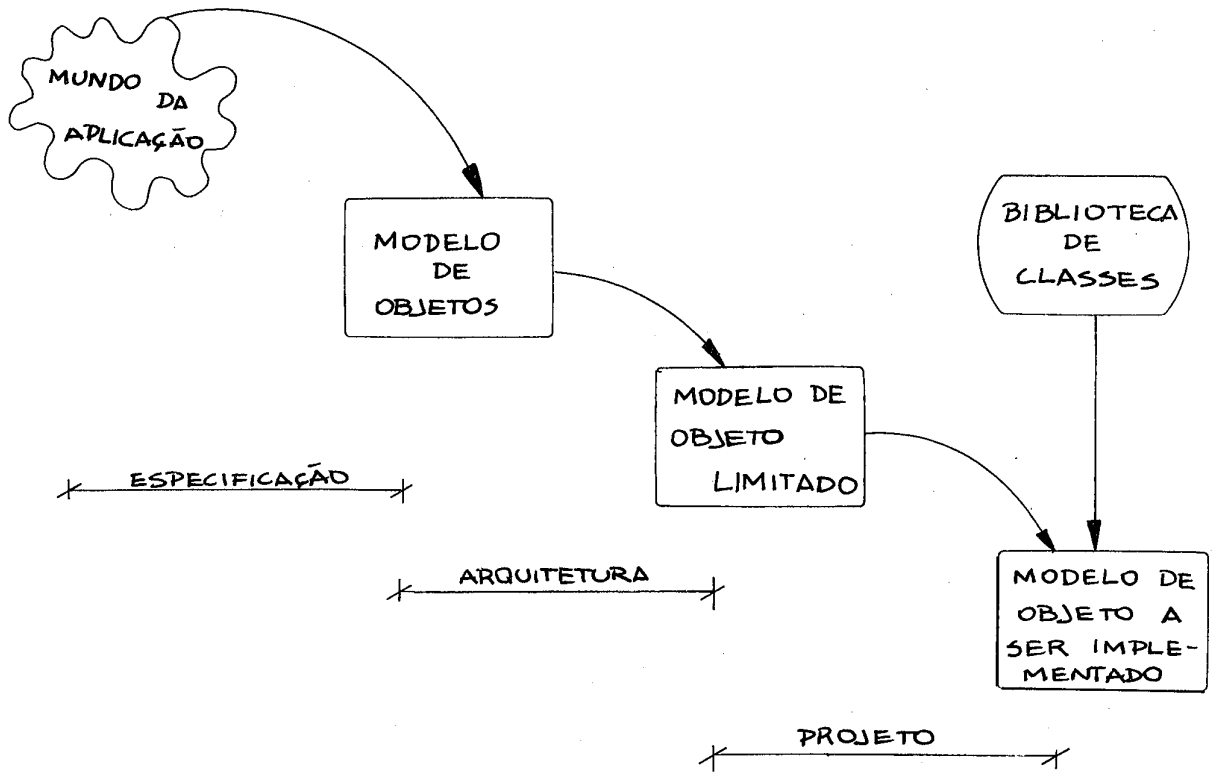
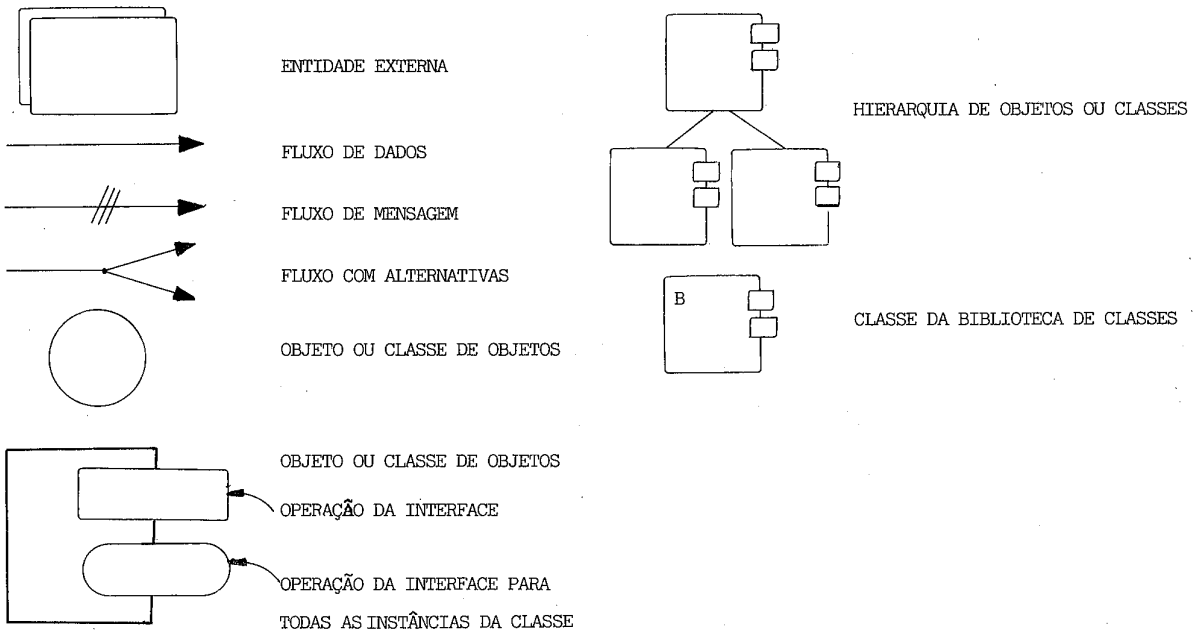


FIGURA 1 - MODELO PROPOSTO PARA O PROCESSO DE DESENVOLVIMENTO



NOTAÇÃO GRÁFICA DA LINGUAGEM PROPOSTA PARA REPRESENTAR O MODELO DE OBJETOS

FIGURA 2

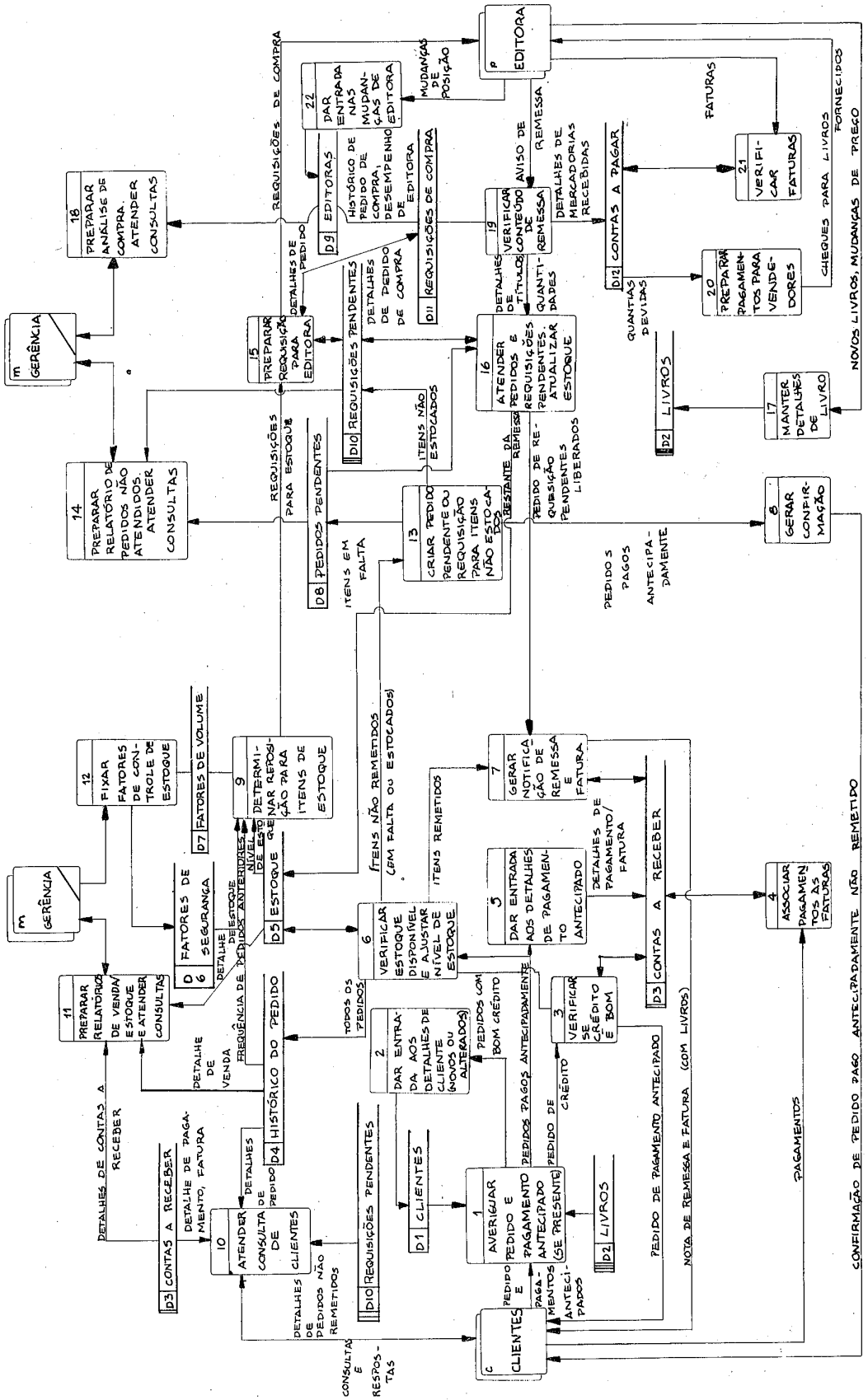


FIGURA 3 _ DIAGRAMA DE FLUXO DE DADOS

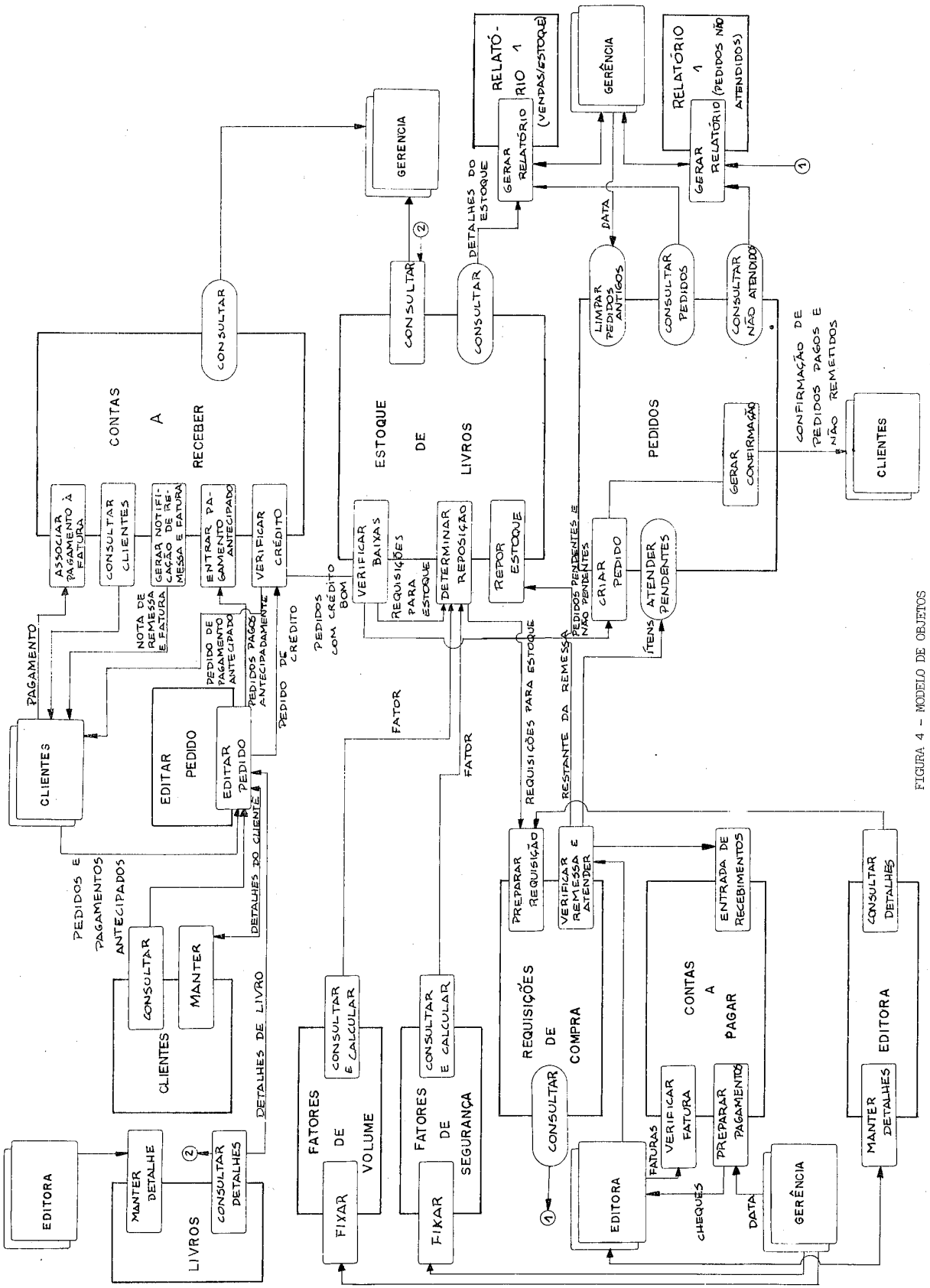


FIGURA 4 - MODELO DE OBJETOS

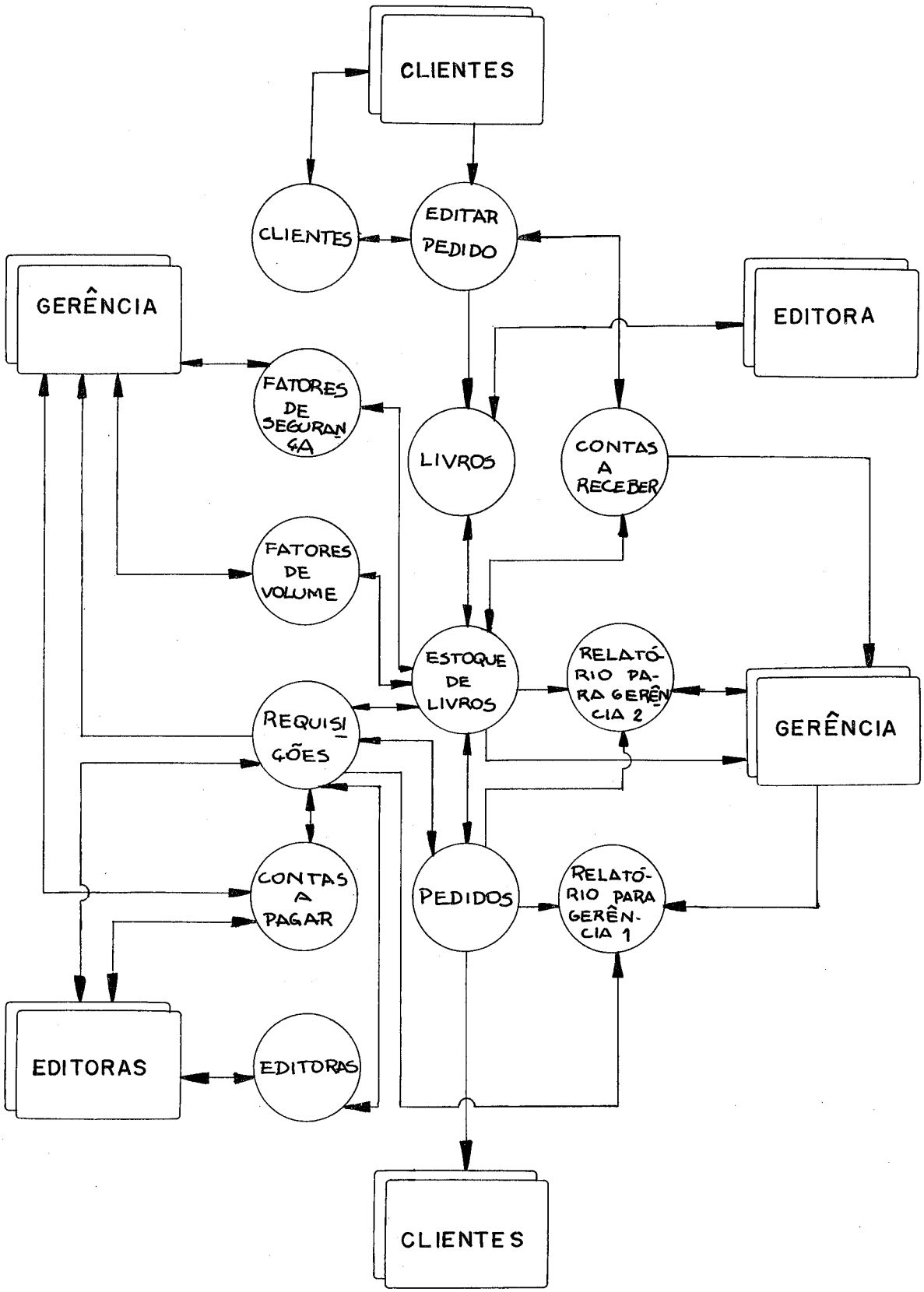


FIGURA 5 - MODELO DE OBJETOS

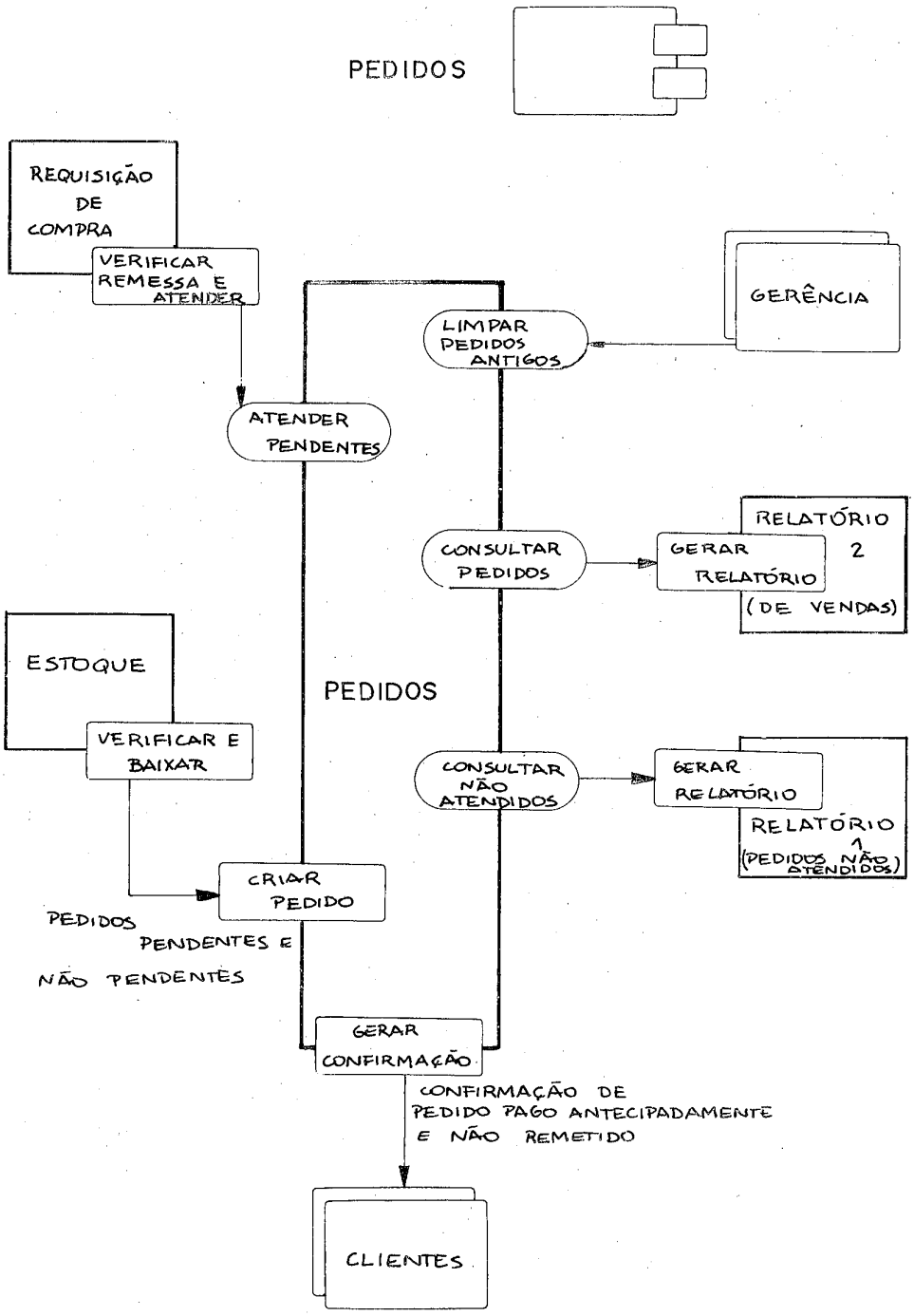


FIGURA 6 - EXEMPLO DE UM OBJETO

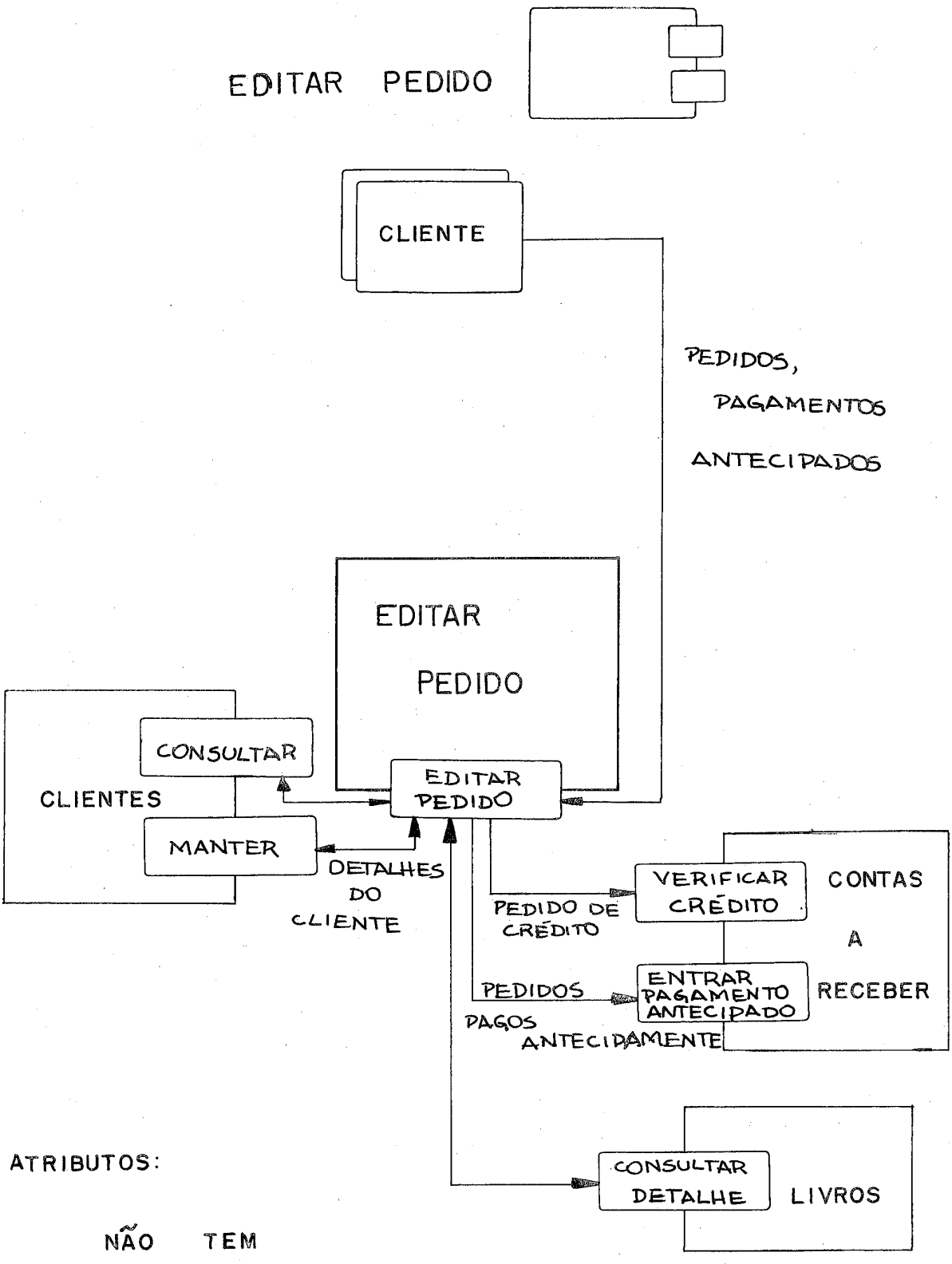


FIGURA 7 - EXEMPLO DE UM OBJETO QUE NÃO SE FORMOU A PARTIR DE DEPÒSITOS DE DADOS

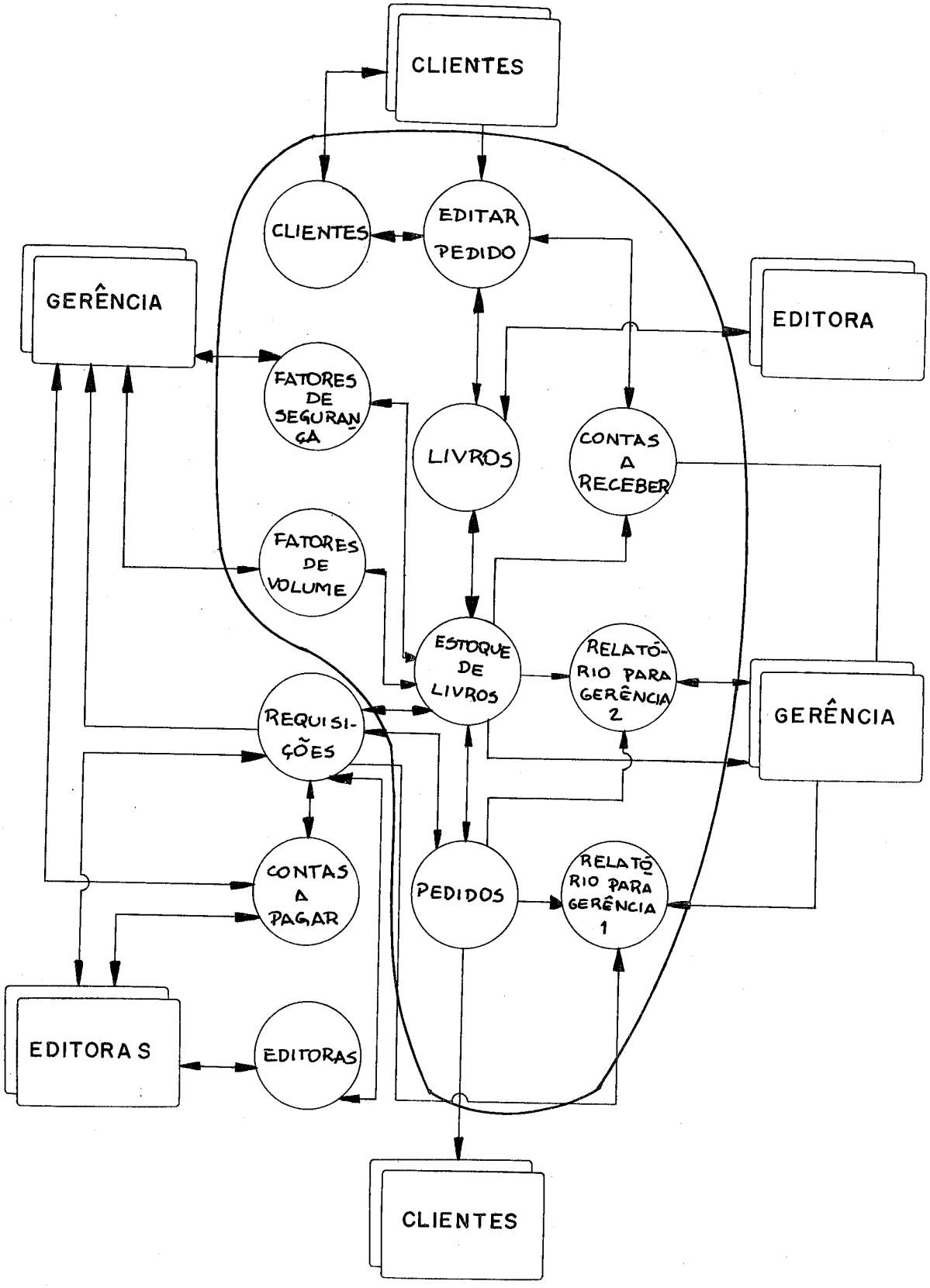
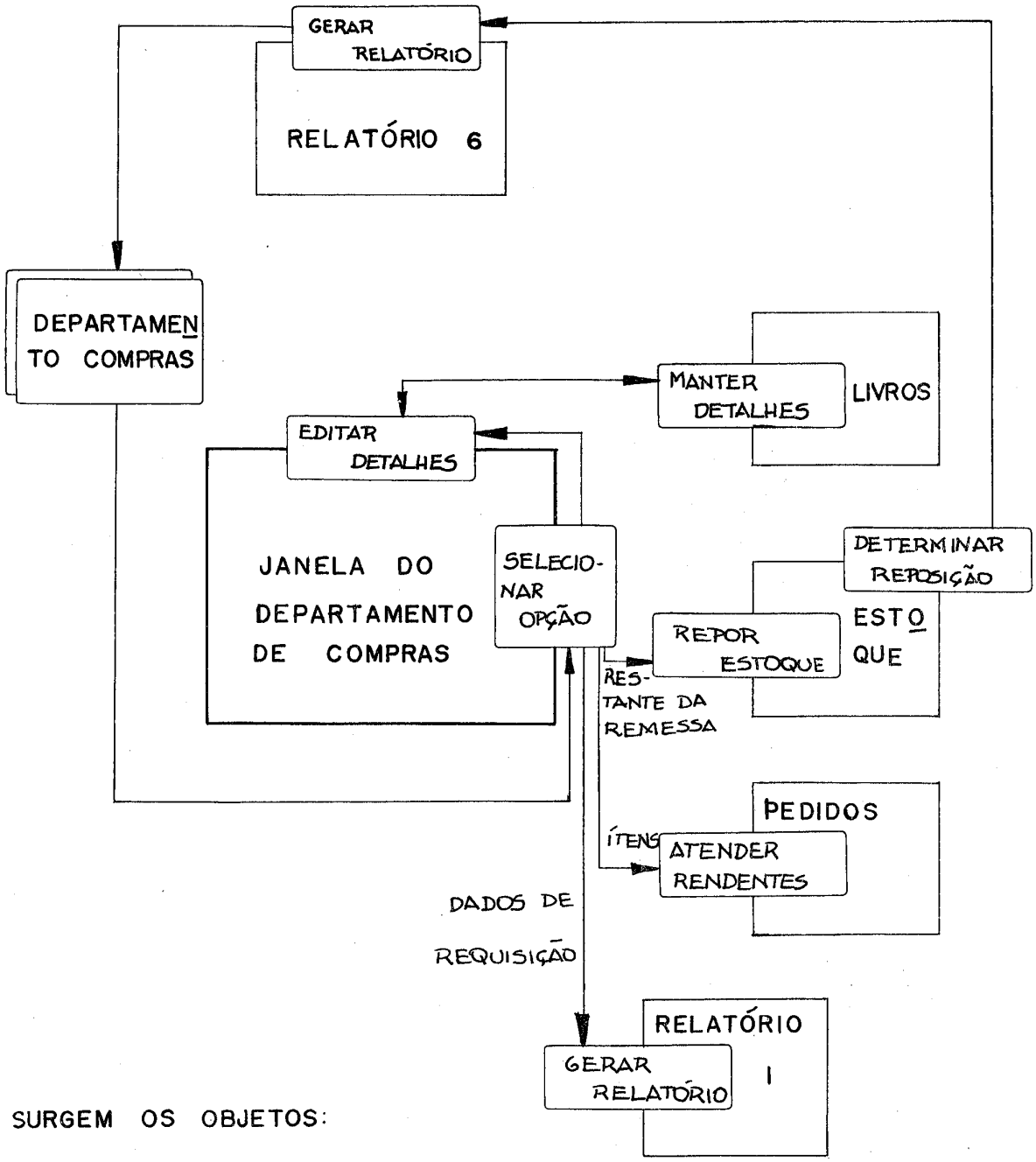


FIGURA 8 - DEFINIÇÃO DA ARQUITETURA MODELO DE OBJETOS LIMITADO

DEFINIÇÃO DAS INTERFACES DEPARTAMENTO COMPRAS-SISTEMA



SURGEM OS OBJETOS:

JANELA DO DEPARTAMENTO DE COMPRAS

RELATÓRIO 6

FIGURA 9 - EXEMPLO DE DEFINIÇÃO DE INTERFACE

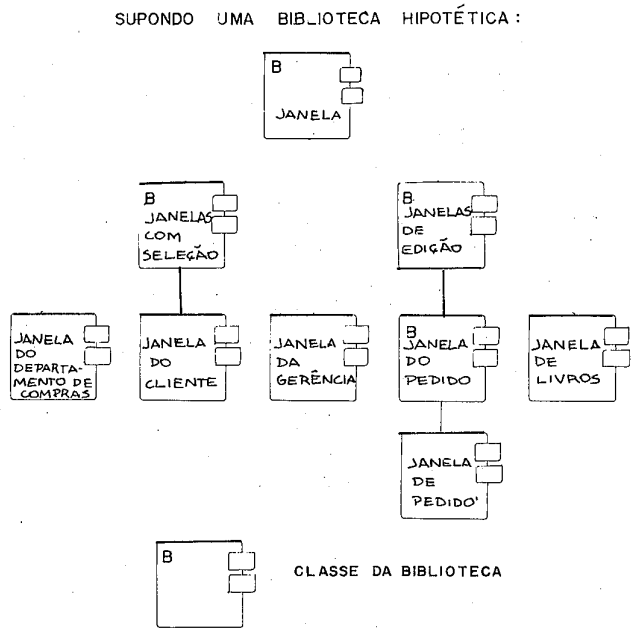
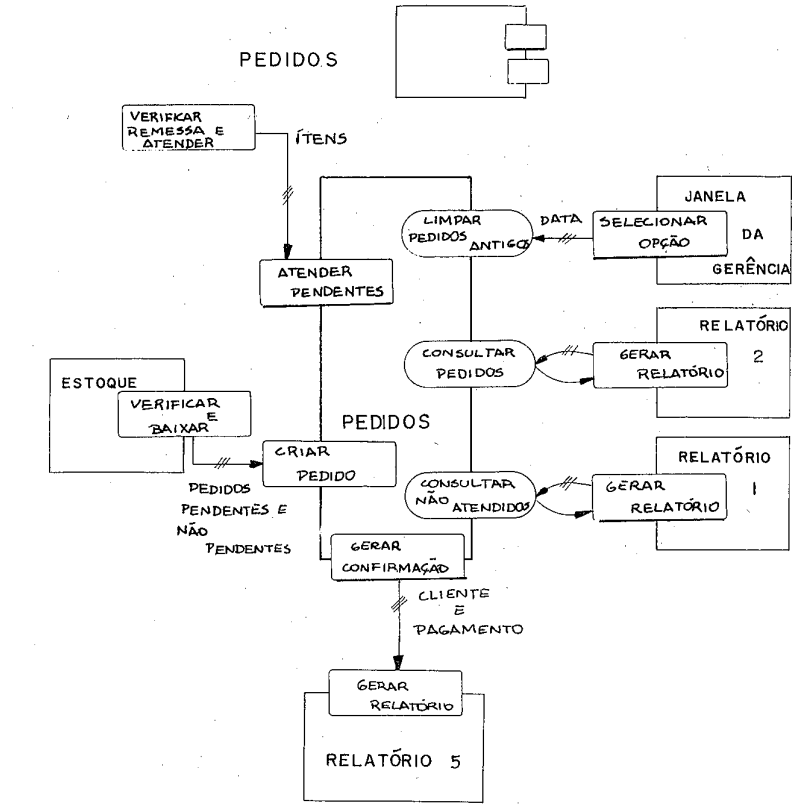


FIGURA 10 - EXEMPLO DE HIERARQUIZAÇÃO



ATRIBUTOS:

PEDIDO	INTEIRO
DATA	DATA
CLIENTE	STRING
LISTA DE ÍTENS:	LISTA
LIVRO	STRING
QUANTIDADE	INTEIRO

SUPERCLASSE: X

FIGURA 11 - EXEMPLO DE OBJETO COMPLETO, INCLUSIVE COM FLUXO DE MENSAGENS