VOL. I

理论计算机科学

Theoretical Computer Science

上海科学技术文献出版社

Shanghai Scientific and Technological
Literature Publishing House

理论计算机科学

Vol. 1

《理论计算机科学》编辑委员会编

上海科学技术文献出版社

(沪)新登字 301 号

理论计算机科学

Vol.1

《理论计算机科学》编辑委员会编

*

上海科学技术文献出版社出版发行

(上海市武康路 2 号)

全国新华书店经销

上海科技文献出版社昆山联营厂印刷

*

开本 850×1168 1/32 印张 6.125 字数164,000

1992 年 6 月第 1 版 1992 年 6 月第 1 次印刷

印数：1—1,500

ISBN 7-80513-926-1/T•219

定 价：4.65 元

《科技新书目》259-286

《理论计算机科学》编委会

通讯地址：

上海科学技术文献出版社《理论计算机科学》编辑部
上海市武康路二号　电话：4373707　邮编：200031

序　言

近卅年来，理论计算机科学已经成为计算机科学中的一个独立分支，但要对理论计算机科学作出一个精确定义，界定它的学科范围，这常会众说纷云，难执一词。理论计算机科学最早是从研究“可计算的精确定义”开始，这种研究早于实际制造计算机10多年之久。图灵和丘奇关于能行性的论断，几十年来一直是指导计算机理论的难以逾越的准则。随着现代科学的迅猛发展，计算机科学已是信息社会的一个必不可少的工具，它的工程应用的特性，已远远超过计算机本身。理论计算机科学的内容已从算法、可计算性等抽象数学描述，转向于计算复杂性、程序设计、数据结构、机器设计理论等的研究，与此相涉的进程、类型、正确性证明等都可用计算模型给予阐明。理论计算机科学可以包括计算的数学理论，亦包括与计算机本身相涉的各种课题。

有鉴于计算机科学应用的不断拓广，计算机技术日新月异的发展，要从本质上深入领域的各个侧面，对于计算机科学的理论研究是必不可少的重要步骤。几年前国内几所高校为了解决理论与实践不相适应的矛盾，大家都想建立一个理论园地，使得一些理论计算机科学的硕果，得以孕育与成长，并以此参与国际学术界的交流，这一宗旨深蒙国内外一些著名学者的鼓励和支持。为此几年前由几个有关院校集资，并征集了国内的一些理论计算机科学中的较有价值的近著，并蒙国外的一些学者赐稿，我们编辑

了本丛刊，希望能以此对理论计算机科学的发展与交流起到一些积极推动的作用。

考虑到本书的适用性，本刊按作者原稿的文种（中、英文）刊登，中文稿件附英文摘要，英文稿件附中文摘要，这种编排是为尊重原作方便阅读。

本刊出版中得到北京计算机学院、华东工学院、贵州大学等有关院系领导的积极鼓励和支持，我们更特别感谢海外编委 Kuodou J. Huang 博士在筹建过程中的积极鼓励和不懈努力。希望本刊能自萌芽破土发展成长的过程中，得到国内外同仁惠予赐稿和指导。

《理论计算机科学》编委会

1991 年 8 月

《理论计算机科学》

Vol. 1

目　录

CONTENTS

METHODOLOGY AND TOOL FOR OBJECT-ORIENTED DATABASE DESIGN

Colette Rolland and Corine Cauvet
Université Paris I Ufr 06
17, Rue de la Sorbonne 75231 Paris Dedex 5

Christophe Proix
Université Paris VI Laboratoire Masi
4, Place Jussieu 75005 Paris

ABSTRACT

The paper deals with object-oriented database design. It aims at presenting a design methodology supported by an expert design tool.

The methodology combines an object-o riented model and a step-wise design approach.

Using the object-oriented model, the database is modeled as a society of objects interacting through events. The database schema is also viewed as a collection of objects including static objects (entities and domains) and dynamic objects (actions and events). Each object of the database schema has structural and behavioural properties which are encapsulated in the object scheme description. Each object belongs to one of the predefined object types of the model.

Design is organized in three steps: identification, structuration and refinement applying to static objects as well as dynamic objects.

The tool aims at supporting "intelligently" the design process itself. It helps the designer by inferring design solutions, by proposing alternative problem solving ways and by checking design products.

1. Introduction

Programming languages are more and more intensively object-oriented [GOLD. 83], [ROCH. 86], [BOBR. 83] and [MOON. 80]. Experiences in object-oriented programming demonstrate an increased productivity of programming teams due to the modularity of programs, their adaptability and readibility.

In databases, the idea of representing in database schemas both structural and behavioural properties of objects is not new [ROLL. 79], [ROLL. 82], [GUST. 82] and [BROD. 82] and demonstrated its advantage in database application development. The current tendancy is to base database design on models that are able to capture the semantic of the real-world with more precision and naturalness. Many semantic data models have been proposed TAXIS [MYLO. 80], SHM [SMIT. 77], SDM [HAMM. 81], SHM+ [BROD. 82]. Except some differences in the formalization and in expression of some constraints, the models provide similar concepts of object classification, aggregation and generalization.

These two trends highlight some convergence both in the domain of databases and in the field of programming

languages to an object-oriented approach.

In databases, this approach starts to be implemented in DBMS[KIM. 88],[LECL. 88] and [PENN. 87].

Nevertheless there does not exist methodologies to support object-oriented database design.

Based on our experience in traditional databases design, we started to develop a methodology combining object-oriented concepts (the O*-model), with a step-wise design approach (the O*-method). Experiencing the methodology on real applications leads us to develop a computer aided design tool (the O*t-ool) to assist the designer at each step of the methodology.

This paper is organized as follows. We present in section 2 the basic object-oriented concepts of the O*-model and the basic rules to manipulate concepts for designing database schemas. In section 3 we review the three main steps of the O*-method and the subsequent sub-steps. Section 4 focusses on the O*-tool to explain how it supports designers in the design of O*-schemas.

2. Basic Object-oriented Concepts

In this section we review the basic object-oriented concepts of the O*-model and their use for designing object-oriented databases.

2.1 Object-Oriented Concepts, Object-Oriented Paradi-gm, O* Schema

The model provides object-oriented concepts and an object-oriented paradigm.

Concepts are predefined types of objects which allow to

modelize any conceptual entity of the object-oriented database as an object. An ordinary integer or string is as much an object as a complex entity as a hotel or a customer; a single value like a price of a room as well as a complex value like an address are also regarded as objects. Actions such as room-reservation or room-request and events such as request-arrival or room-availability are considered as objects too.

According to the objett-oriented paradiam any object is considered as having structural, behavioural and inherited properties that are encapsulated in the object desoription of the database conceptual schema, socalled O*–schema. As model-concepts are viewed as objects, the object-oriented paradigm applies also to concepts.

Thus, any object is structured, active, encapeulated and typed. We describe in turn these four characteristics of objects.

2.2 Object Structure

For the structuration of objects, O*–provides two forms of structural abstraction to relate objects: aggregation and greuping.

Aggregation [SMIT. 77] is a form of a bstraction in which arelationship between component objects is considered as a higher level aggregate object. This is the part-of relationship. For example, ROOM may be an aggregate of components ROOM–NUMERO, ROOM–PRICE and ROOM–CATEGORY.

Grouping is a form of abstraction in which a relationship between member objects is considered as a higher level set object. This is the member-of relationship. For example the set HOTEL–UNION is a group of HOTEL members.

Static schemes are diagrammatic tools to aid structure design and to graphically represent the objects and the structural relationships. The graphic notations for the two forms of abstraction are examplified in Figure 2.1.

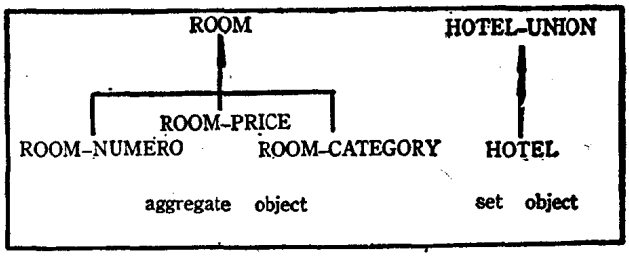

Figure 2.1 Static schemes

2.3 Object Behaviour, Encapsulation

For the design of behavioural properties of objects, O* provides two concepts: action and event.

Actions are elementary database operations which alter objects (e. g. add a new room, modify a reservation ···). The state change of an object can activate an event. Events represent elementary state changes in the object database that must trigger certain actions (e. g. when a room-request arrives, create the room-reservation if possible and otherwise postpone the request).

Therefore, interaction among the different objects consist of event sharing. No other form of communication is allowed.

The description of actions is defined in the action-text. The description of the event occurrence condition is defined in the event-predicate. Action triggering is described in the triggerdeclaration. Action-texts, event-predicates and trigger declarations, if expressed in an executable code, can be assimilated to methods. The description of the objectb ehaviour is

encapsulated with the description of its structure.

Action-texts, event-predicates and trigger-declarations are part of the definition of the object. However, as methods, they are not visible from outside the object. For each event activated by an object there are corresponding actions that execute actiontexts on related objects. An object can react to an action by activating an event. Events constitute the interface of objects.

Dynamic schemes are diagrammatic tools for the design of behavioural properties of objects. Figure 2.2 gives the graphic notation for actions and events related to an object.

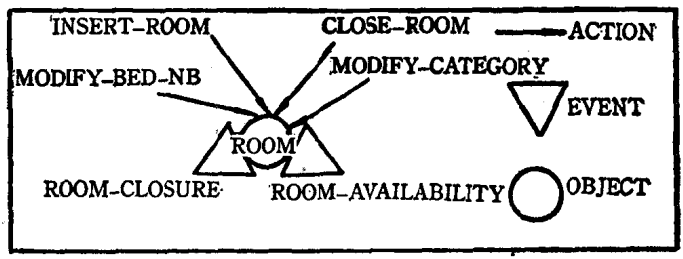

Figure 2.2 The ROOM object dynamic scheme

2.4 Type, Type Hierachy, Meta-type

Typing is a form of abstraction in which a collection of objects is considered as a higher level object type. An object type is a precise characterization of all structural and behavioural properties shared by each object in the collection. An object is an instance of an object type if it has the structure and behaviour defined in the type. Typing represents an instance-of relationship between an object type in a O*-schema and an object in a database. For example the object type ROOM described by its static scheme in Figure 2.1 and its dynamic scheme in Figure 2.2 defines aggregate objects

in the database like ⟨14, 250, "***"⟩ with action values like "insert ⟨14, 250, "***"⟩" and event values like "on delete ⟨14, 250, "***"⟩".

O*-schema is a collection of types which are regarded as objects.

The classification of types can be expressed in a generalization hierarchy. Genera-lization is a form of abstraction in which a set of types is viewed as one generic type. A type hierarchy is a hierarchy of types in which an edge between a pair of nodes represents the is-a relationship. The lower level node is a specialization of the higher level node. For example the generic type RESORT may be a generalization of types SKI-RESORT and SEA-RESORT. Structural and beha vioural properties for a type are inherited by all its specialized types.

The graphic notation for an "is-a" relationship is illustrated in the hierarchical scheme in Figure 2.3.

Any type of a O*-schema must belong to a model predefined type (also called meta-type). Meta types are object types classified in a generalization presented in Figure 2.4.

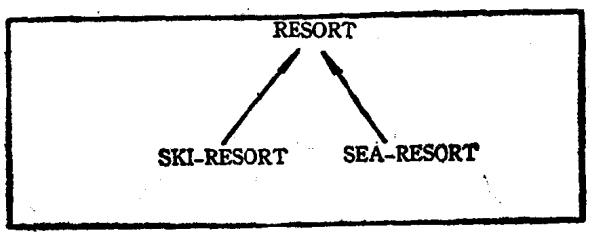

Figure 2.3 Graphic representation of an is-a hierarchy

The root of the hierarchy is the model defined type OBJECT.

The instances of the type METATYPE are the set of

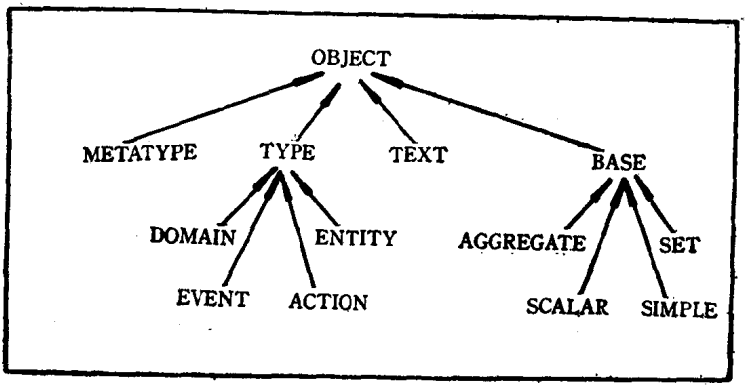

Figure 2.4 Model meta-type hierarchy

model predefined object types. ENTITY, ACTION, EVENT and DOMAIN are examples of METATYPES instances.

The objects in a O*-schema are instances of TYPE i. e. instances of ENTITY, ACTION, EVENT and DOMAIN. For example, the object type ROOM (see Figures 2.1 et 2.2) is an object of the predefined type ENTITY.

The instances of predefined type TEXT represent action-texts, event-predicates and trigger declarations.

The instances of the predefined type BASE are objects used to build other objects.

2.5 O*-Schema, O*-Database

A O*-database can be viewed as a society of objects which interact through events.

A O*-schema describes the society of objects and their interactions. It is composed of object definitions such as:

(i) each O*-schema object represents a database object type. Its definition requires three parts:

-a static part (ST) related to the object structure,

-a dynamic part (DY) including action and event definitions,