TECHNICAL REPORT

ISO/TR 14813-4

First edition
2000-12-15

Transport information and control systems — Reference model architecture(s) for the TICS sector —

Part 4: Reference model tutorial

Systèmes de commande et d'information des transports — Architecture(s) du modèle de référence du secteur TICS —

Partie 4: Manuel de modèle de référence

Reference number
ISO/TR 14813-4:2000(E)

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The main task of technical committees is to prepare International Standards, but in exceptional circumstances a technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when a technical committee has collected data of different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Technical Reports are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Attention is drawn to the possibility that some of the elements of this part of ISO/TR 14813 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/TR 14813-4, which is a Technical Report of type 2, was prepared by Technical Committee ISO/TC 204, *Transport Information and Control Systems*.

This document is being issued in the Technical Report (type 2) series of publications (according to subclause G.3.2.2 of Part 1 of the ISO/IEC Directives, 1995) as a “prospective standard for provisional application” in the field of transport information and control systems because there is an urgent need for guidance on how standards in this field should be used to meet an identified need.

This document is not to be regarded as an “International Standard”. It is proposed for provisional application so that information and experience of its use in practice may be gathered. Comments on the content of this document should be sent to the ISO Central Secretariat.

A review of this Technical Report (type 2) will be carried out not later than three years after its publication with the options of: extension for another three years; conversion into an International Standard; or withdrawal.

ISO/TR 14813 consists of the following parts, under the general title Transport Information and Control Systems — TICS Reference Architecture:

— *Part 1: TICS Fundamental Services*: This document presents the definition of 32 TICS fundamental services that are the informational products or services or applications areas provided to a TICS user.

— *Part 2: Core TICS Reference Architecture*: This document describes an abstract object-oriented system architecture based on the TICS Fundamental Services.

— *Part 3: Example Elaboration*: This document refines the Core TICS Reference Architecture (Part 2) with some emphasis on traffic management.

— *Part 4: Reference Model Tutorial*: This document describes the basic terms, graphical representations and modelling views exploited in the object-oriented definition of the architecture development of Parts 2 and 3.

— *Part 5: Requirements for Architecture Description in TICS Standards*: Requirements for Architecture Description in TICS Standards: This document describes the terminology and form to be used when documenting or referencing aspects of architecture description in TICS standards.

— *Part 6: Data Presentation in ASN.1:* This document establishes the use of ASN.1 as the normal syntax notation to be used in standards for the TICS sector and a common message form for such ASN.1 based data elements.

Introduction

ISO/TC204/WG1 is a working group whose prime objectives are to provide services to ISO TC204 and its working groups. A specific mission of WG1 is to:

> “Provide ISO TC204, its working Groups, related bodies and those involved in the TICS sector, with a reference model of Conceptual Reference Architecture(s) that show the structure and inter-relationships of the sector ...”

It is expected that there may well be more than one single TICS Architecture approach to be considered and documented and that existing architecture approaches will have previously-produced documentation developed according to disparate standards and conventions.

It is also implicit in the work being undertaken by WG1, that working group members will require a clear, well-structured understanding of the work of the following participant groups:

- Other TC 204 Working Groups
- CEN TC 278 Working Groups
- Japanese initiatives
- European Road Transport and Traffic Telematics programs
- US Intelligent Transportation Systems program
- Australian initiatives
- Canadian Initiatives

Full documentation of all possible architectural approaches is obviously not feasible given the high level of resources required to carry this out. Indeed full documentation and description of all possible approaches is undesirable as an item for Standardisation.

A defined and consistent approach is however required to facilitate the specification of architecture requirements to enable a clear view to be developed and presented of the work of each participant group This document is one of a set of WG1 documents intended to respond to stated WG1 objectives regarding the production of a TICS Reference Architecture.

In order to document an architecture, graphical and textual components of a model are required. WG1 has adopted a methodology based on the Unified Modelling Language (UML) for documenting the TICS Reference Architecture. A tutorial on the UML is provided in ISO/TR 14813 Part 4. UML is a visual modelling language for building object-oriented and component-based systems. A commercially available Computer Aided Software Engineering (CASE) tool has been used by WG1 to document the Architecture. While the tool is a commercial product, UML is open and non-proprietary.

Transport information and control systems — Reference model architecture(s) for the TICS sector — Part 4: Reference model tutorial

1 Scope

The architecture of an information and control system merges hardware and software considerations into a coordinated and integrated system view. The system architecture is a high level abstraction, or model, of the system. A system architecture should embrace both today's applications and the applications that are expected in the future. Architecture begins with the definition of the conceptual services (e.g. Part 1 - TICS fundamental services). There are several identifiable stages of system architecture development.

- Reference architecture
- Logical architecture
- Physical architecture

The reference architecture is generic and non-prescriptive and captures the concepts of the system. A logical architecture elaborates the functions that will provide the conceptual behaviour, and in so doing it provides some detail about the modularity. A physical architecture is reached when the actual distribution of the system modules is defined, thus leading to important implications for communications.

This technical report develops a TICS Reference Architecture. The objective in defining a TICS Reference Architecture is to provide a concise reference point which is both educational and a framework for the standards process. The Reference Architecture will be used by the Working Groups to develop their own logical and physical architectures in a cohesive manner.

This Part introduces the model that is applied in developing the Reference Architecture in Parts 2 and 3. A tutorial on the application of the model is provided using examples from the TICS sector.

2 Modelling an architecture

In order to document an architecture, graphical and textual components of a model are required. A unified process and the Unified Modelling Language (UML)[1] developed by Ivar Jacobson, Grady Booch and James Rumbaugh addresses this requirement for the software industry. The result is a component-based process that is use-case driven, architecture-centric, iterative, and incremental.

In Parts 2 and 3 of this technical report the abstraction that is the TICS Reference Architecture is described in four views of the Unified Modelling Language:

1. Use Case diagram
2. Class diagram
3. Package diagram
4. Sequence (Interaction) diagram

The UML views have been developed to support methodologies for building systems with the object model (e.g. systems whose software is programmed in object-oriented languages). Therefore it is necessary to begin this tutorial by introducing some basic concepts of object-oriented (OO) modelling.

Although ISO/TR14813-5 states that architectures may be developed using either OO or process-oriented methods, WG1 has chosen to use the OO method.

3 Why object-oriented?

The Transport Information and Control Systems (TICS) Reference Architecture is being developed by ISO/TC204 Working Group 1 (WG1) with the following objectives.

- to be a concise statement of what TICS is, and thus help develop consensus within TC204 and wider circles
- to act as a framework for the standards process in TC204
- to serve an educational purpose and lend itself to promulgation in media such as the Internet.

A reference architecture is non-prescriptive of technology and deployment organisation.

In the future, WG1 expects a number of standards developments such as data dictionaries, model sub-system architectures, and message standards to update the reference architecture and to evolve it into various architectures.

Thus the choice of both model and process for the architecture development were important decisions. Current and future requirements include:

- information models
- functional models
- dynamic models
- implementation models

Older system development practices are disjoint in the way that all these requirements can be met. For example, functional and data flow models must be combined with a separate model for information, and these two types of model do not integrate conveniently. Object-oriented modelling accommodates these requirements harmoniously.

Modern computer based system engineering for software development automates the object-oriented model. The process also incorporates other important models. The same tools used for system engineering are suitable for the development of architectures.

By applying the object-oriented model and process for the Reference Architecture a highly effective base is established for future developments.

4 The Unified Modelling Language

The Unified Modelling Language is a visual modelling language for building object-oriented and component-based systems. The UML was recently developed by Rational Software Corporation and its partners and is the successor to the modelling languages found in the Booch, OOSE/Jacobson, OMT, and other system engineering methods. UML has been progressed through the Object Management Group (OMG) for adoption as an OMG standard.

The developers of the UML recognised that the choice of what model projections one creates has a profound influence upon how a problem is approached and how a solution is shaped. They maintain that

- Every complex system is best approached through a small set of nearly independent views of a model - no single view is sufficient.

- Every model can be expressed at different levels of fidelity.

- The best models are connected to reality.

Rational Software Corporation, as well as many others, are incorporating the UML into their development process and products, which cover disciplines such as business modelling, requirements management, analysis & design, programming, and testing.

WG1 has adopted a methodology based on UML for developing and documenting the TICS Reference Architecture and is using a commercially available Computer Aided Software Engineering (CASE) tool to do this. While a CASE tool is a commercial tool, UML is open and non-proprietary.

Abstraction, which is to focus on relevant details while ignoring others, is the key to the development of a reference architecture. This determined the selection of four particular elements from the UML for the modelling task:-

1. Use Case

2. Class

3. Package

4. Sequence (Interaction)

These elements provide the multiple perspectives of the reference architecture.

- Use case diagrams define the architecture boundary, the external actors and the services provided.

- Class diagrams define the abstract elements that comprise the reference architecture. These elements underpin the dynamic model of the system.

- Package diagrams are the means by which model (architecture) elements can be grouped. Packages provide a hierarchical organisation as well as a network of package references.

- Sequence (Interaction) diagrams describe how the implied objects of the system cooperate to provide the services defined in the use case.

Each element view is presented as a diagram. The underlying model integrates these views into a self-consistent reference architecture. The modelling elements are explained in more detail in the following sections.

It is expected that some of the other types of diagrams defined in the UML (e.g. implementation diagrams) may be used by Working Groups to develop more detailed architectures.

5 Object-oriented modelling elements: class and object

The main purpose of modelling is to prepare generic descriptions that describe many specific particular items. In object-oriented modelling the most important generic description is called a **class** and a specific particular item belonging to a class is called an **object**.

Table 1 lists three particular objects (instances of a class called Intersection). The corresponding real life situation is depicted in the schematic of Figure 1.

There are at least three conceptual levels involved in the example. The first conceptual level is the class. This is used in architecture and design. The second conceptual level involves the software objects belonging to the class (e.g. Intersection) which arise in the development of a system conforming to the architecture. The third conceptual level is that of the real world where there is roadway pavement corresponding to the higher level intersection concepts. The higher levels are abstractions that represent the real world objects for particular purposes such as system architecture and system implementation.

Table 1 — Example class and objects

Class	Software Objects	Real World Objects
Intersection {operations & attributes}	Object1{ data}	Main St & Route 401
	Object2{ data}	Main St & Washington St
	Object3{ data}	Main St & Route 20

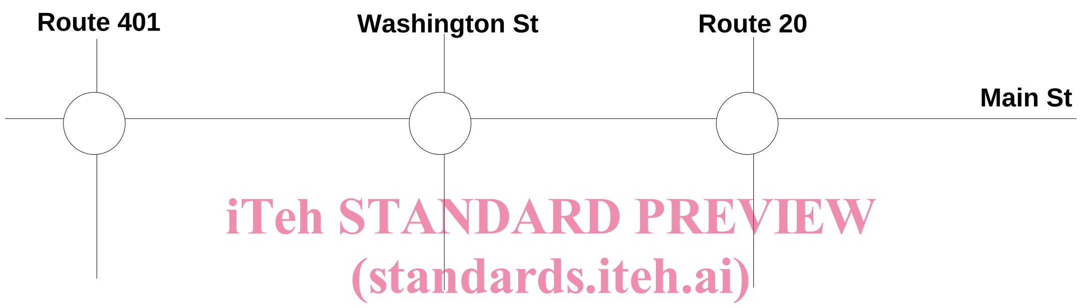

Figure 1 — A map of the real world corresponding to the three Intersection objects

In the above example the class might be defined so that information about real world intersections could be stored and updated in a system database. The reader shall see later that classes (and their implied objects) are often invented for other purposes, such as control and system interfaces.

Only classes are presented in the Reference Architecture, however the implied objects will often be referenced in the discussion of the architecture. A more complete definition of class is now given.

A **class** is the descriptor for a set of objects having similar structure, behaviour, and relationships. Class definitions consist of attributes and operations.

Attributes are the elements used to record the state of an object (e.g. the equipment deployed at an Intersection, or the traffic demand at an Intersection, or the current control phase). State changes of an object are recorded by changing the attribute values. **Static attributes** are those that apply collectively to the entire class (e.g. a count of Intersections). In Table 1 the state of the software objects is referred to as data.

An **operation** is an action that a software object performs (e.g. Change Phase at an Intersection in order to allow a different traffic movement) when it receives a particular stimulus called a **message** (e.g. a request to change phase). The collection of defined operations specifies the behaviour of a class (i.e. for all its objects). Classes may also have **static operations** (e.g. list the Intersections on a given route).

The relationships that are recorded about objects are called **associations** (e.g. each Intersection object is associated with a set of traffic Movements, and a Movement is associated with a set of Manoeuvres which can occur simultaneously, i.e. in the same traffic control phase). The various types of association are defined in a later section.

6 Abstraction

Unlike any other model, the class-object model can be applied in every stage of the development process, from architecture to system implementation. The key is the use of abstraction.

Because a reference architecture is concise and it originates at the very early stages of system development, it must use abstract classes, that is classes which reflect relevant details while ignoring others.

On the other hand, in object-oriented software design and implementation, the class element provides concrete specifications for the attributes, operations and state transitions of the software objects that will materialise the system.

Abstraction in the class-object model is a powerful means by which the strong connection between architecture, implementation, and the reality found in objects is maintained.

Abstraction is also applied in the other model views of the architecture. However it is only the elements of the class-object model which are materialised in the system implementation.

7 Model Views

Our objectives for conciseness etc. can only be met by using a graphical modelling language. We now describe the four UML model views and their diagram notation used in developing the reference architecture.

7.1 Use case diagrams

Use case diagrams are the first step in architecture development. They model the functional system requirements and scope. There are two primary components of a use case diagram, actor and use case.

An **actor** is a class external to the system. The relevant actors are those whose objects interact with the system. Thus an actor may be a role performed by a human or it may be a type of system. Although an actor is not necessarily a human it is always represented by a 'stick figure' in the use case diagrams, and labelled with its class name.

When an actor object interacts with the system, together they perform a behaviourally related transaction sequence. Each specific sequence is called a **scenario**. The abstraction of similar and closely connected scenarios is called a **use case**.

Just as the idea of classification is natural to the modelling process, so is the grouping together of strongly related use cases. This is the basis for forming a use case diagram. In the diagram a use case is represented by an ellipsis labelled with the name of the use case.

The natural way to begin an architecture specification is to identify some of the actors and the associated use cases. This is often referred to as developing the operational concept for the system. This process answers questions such as why is the system being developed and what must it do. (We might then specify some of the other types of diagrams before identifying all the actors and uses cases in iterative fashion.)

A **use case diagram** shows the relationships among the actors and the use cases. It is a graph consisting of actors and use cases, the latter enclosed by the system boundary. The system boundary separates the actors from the use cases and is shown as a dashed-line rectangle. The communication (participation) associations between the actors and the use cases, and the relationships among the use cases are shown by different types of arc in the graph.

The participation of an actor in a use case is shown by connecting the actor symbol to the use case symbol by a solid path. The actor is said to communicate with the use case.