TECHNICAL REPORT

ISO/TR 17452

First edition
2007-04-15

Intelligent transport systems — Using UML for defining and documenting ITS/TICS interfaces

Systèmes intelligents de transport — Usage de UML pour définir et documenter les interfaces ITS/TICS

Reference number
ISO/TR 17452:2007(E)

Contents

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

In exceptional circumstances, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide by a simple majority vote of its participating members to publish a Technical Report. A Technical Report is entirely informative in nature and does not have to be reviewed until the data it provides are considered to be no longer valid or useful.

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

ISO/TR 17452 was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

Introduction

ISO 14817 specifies the formats and procedures used to define information exchanges within the ITS/TICS sector. Such information arises through the development of the architecture for a particular application standard and the subsequent specification of the detailed data concept instances that arise in association with the architecture's interfaces. This Technical Report illustrates the steps involved in such development.

In the development of standards, it is often the case that working groups have a well-formed perception of the conceptual context within which their standard is to be applied. This is the case because many standards are the result of a refinement and consensus of requirements based on recent practice. The formal process for the identification of the requirements is streamlined to capitalize on this available body of knowledge.

For completeness, we begin with the capture of requirements. These requirements need be only those which directly affect the standard. The context of a real-world system that incorporates the standard would include a much wider range of requirements; however, we are focusing on that aspect of standards which produces data elements and other concept instances which will be registered in a data dictionary or registry. The methodology is derived from processes used in the development of software-intensive systems.

Intelligent transport systems — Using UML for defining and documenting ITS/TICS interfaces

1 Scope

This Technical Report gives guidelines for using the unified modelling language (UML) for defining and documenting interfaces between intelligent transport systems (ITS) and transport information and control systems (TICS). It presents these guidelines in the context of a case study for the creation of an ITS/TICS data dictionary and submissions to the ITS/TICS data registry.

In UML [6], an interface is a collection of operations that used to specify a service of a class or component.

The ITS/TICS data registry defined in ISO 14817 builds on this definition by mapping an operation to a message, and then it extends the definition of an interface to be a dialogue (i.e. a collection of messages within an implied protocol). This Technical Report conforms to these steps.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 14817, *Transport information and control systems — Requirements for an ITS/TICS central Data Registry and ITS/TICS Data Dictionaries*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 14817 and the following apply.

3.1
automatic equipment identification
AEI
process of identifying equipment or entities that uses the surface transportation infrastructures by means of on-board equipment combined with the unambiguous data structure defined in ISO/TS 17261

NOTE “Equipment” indicates large equipment that is carried in, or forms an integral part of, a trailer or trailer-mounted unit.

3.2
automatic vehicle identification
AVI
process of identifying vehicles using on-board equipment combined with the unambiguous data structure defined in ISO/TS 17261

3.3
electronic data interchange
EDI
passing of a data message, or series of messages, between computers and/or between different software systems

NOTE 1 Within this context, an EDI message is normally compatible with the form specified in ISO 9897.

NOTE 2 EDI is an instance of an electronic data transfer transaction.

3.4
goods provider
party that provides goods to another party

NOTE A goods provider can be a manufacturer, trader, agent or individual.

3.5
information
data, documentation and other relevant knowledge organized to inform and describe

3.6
information manager
function of managing information in a system

NOTE The role of information manager can be provided by one or many actors. It can be performed internally by one or more of the system's principal actors, or can be formed commercially or altruistically by one or more third parties.

3.7
intermodal transport
movement of goods in one or more loading unit or vehicle which uses successively several modes of transport without handling of the goods themselves when changing modes

NOTE 1 Unlike multimodal transport (3.10), intermodal transport implies changing from one mode to another using the same form of loading unit.

NOTE 2 See ISO/TS 17262 and ISO/TS 17263.

3.8
journey
⟨AVI/AEI⟩ physical movement of goods from the goods provider (3.4) to the receiver (3.11)

3.9
load
that which is to be transported from the goods provider (3.4) to the receiver (3.11)

NOTE A load comprises the consignment, packaging, pallets and/or containers that are smaller than an ISO container.

3.10
multimodal transport
carriage of goods by at least two different modes of transport

cf. intermodal transport (3.7)

NOTE Multimodal transport implies that either there is more than one modal shift, or that loads may be broken into partial loads as part of a modal change.

3.11
receiver
⟨AVI/AEI⟩ one who receives goods as a result of a journey (3.8) from a goods provider (3.4)

3.12
returnables manager
function that manages the supply, maintenance and returns cycle of returnable units (3.13)

NOTE The returnables manager function may be performed by one or more of the system's principle actors or by an independent third party.

3.13
returnable unit
unit used as part of a load (3.9), which is returned to the goods provider (3.4) or to a returnables manager (3.12)

NOTE Pallets and trays are examples of a unit.

3.14
transponder
tag
electronic transmitter/responder which responds to the receipt of suitable modulated or unmodulated downlink signals and transmits predetermined information according to predefined protocols at a predetermined frequency

NOTE The transmissions can be powered from energy obtained from the downlink or can be assisted by an on-board power supply. Forms the core, but not necessarily the only, function of an item of on-board equipment. Within the AVI/AEI context, it is fitted to the vehicle or equipment. Its prime function is to provide the identity of the item, but it can also contain additional information. For some special purposes, transponders can be installed in fixed positions and read by mobile equipment.

3.15
transport
⟨AVI/AEI⟩ vehicles/aircraft/ships used to move a consignment from the goods provider (3.4) to the receiver (3.11) or to move returnable units (3.13) back through the system

3.16
transport means
vehicles, trailers, vessels, aircraft or combination thereof which perform the journey (3.8) to deliver the consignment to the receiver (3.11) or to return returnable units (3.13), together with the driver/pilot/crew physically conducting the journey

3.17
transport operator
function that owns and/or manages the transport means (3.16)

4 Abbreviated terms

AVI/AEI	automatic vehicle and equipment identification
RCU	returnable container unit (see also 3.13)
UML	unified modelling language
VMS	variable message sign

5 Example of automatic vehicle and equipment identification

To illustrate the steps in this tutorial, we use the example of AVI/AEI for intermodal goods transport as specified in ISO/TS 17261 [3] and ISO/TS 17262 [4]. The overall application information architecture is shown in Figure 1. The key entities involved in the architecture are defined in Clause 6.

The context of Figure 1 is the information associated with the journey of goods from the goods provider to the end user. In this journey, the goods will form a load. The load can pass through several transport mode changes and other handling processes. In each instance, ISO/TS 17262 is applicable to an automated handling process.

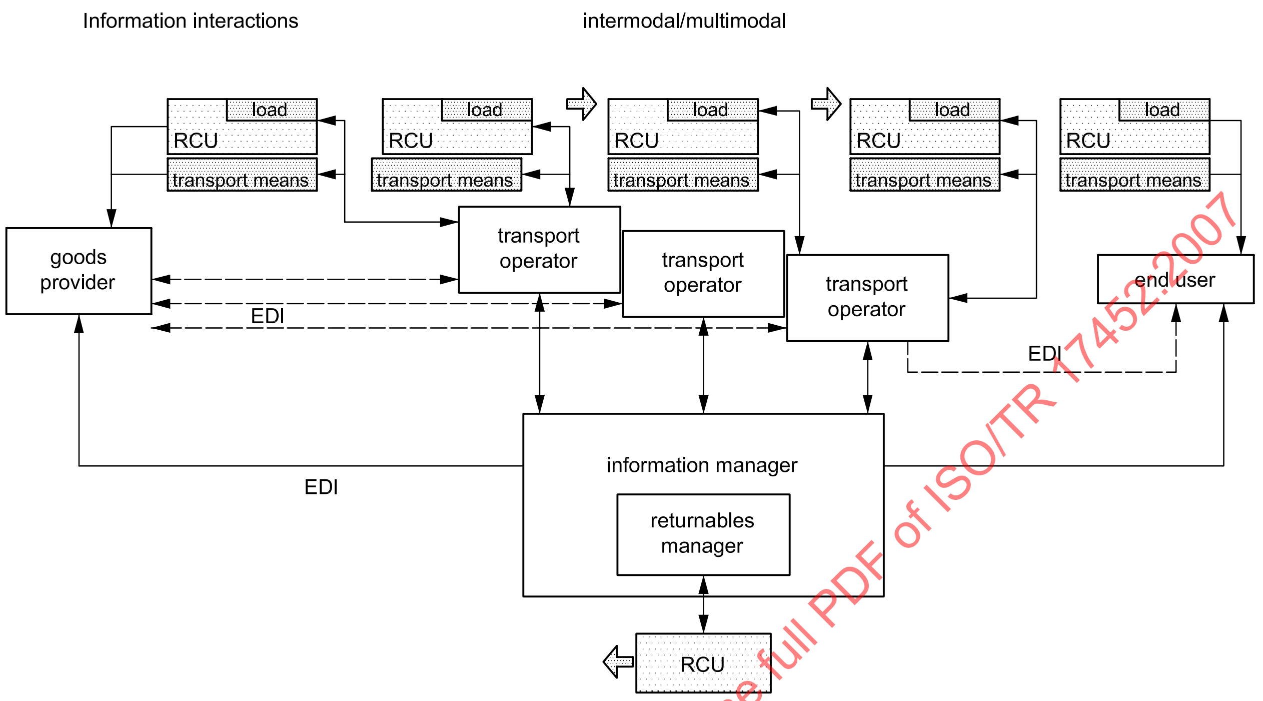

Figure 1 — Schematic diagram of the application information architecture for the journey of goods from goods provider to receiver

6 Developing the data concepts in an application standard

This tutorial employs UML [6] and illustrates how a process framed around UML (e.g. [7]) can be employed to develop the behavioural descriptions of an application architecture [7], necessary to capture the interface data concepts. In the case of standards development, some of these concepts will be the focus of an application standard.

The whole process can be broken into a sequence of steps (Figure 2), each of which has its own set of modelling artifices in UML. It is a straightforward matter to present an example in which the last step reveals a set of standard data elements. In practice the process is iterative, involving trial and error, and the steps are not always revisited in the same order.

The application information architecture shown in Figure 1 serves to justify the development of a standard because it identifies the widespread applicability of the standard. However, it is not sufficiently developed for defining the data concepts. This tutorial need only focus on a single goods-handling application function in order to illustrate the process of architecture development which culminates in data concept definition. The application is described in the first step of the process of Figure 2, in which the requirements are defined by a use case.

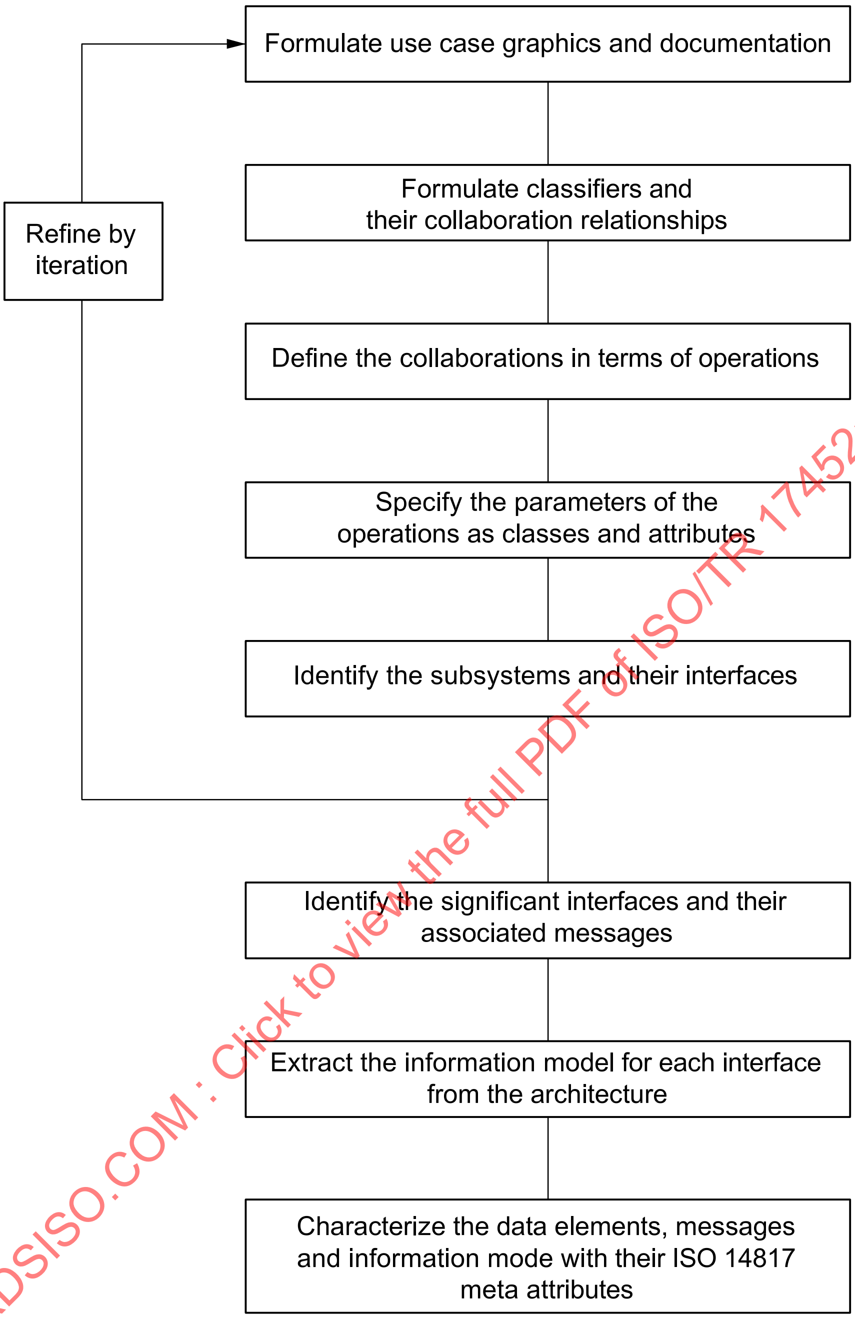
Formulate use case graphics and documentation
Formulate classifiers and their collaboration relationships
Define the collaborations in terms of operations
Specify the parameters of the operations as classes and attributes
Identify the subsystems and their interfaces
Refine by iteration
Identify the significant interfaces and their associated messages
Extract the information model for each interface from the architecture
Characterize the data elements, messages and information mode with their ISO 14817 meta attributes

Figure 2 — Steps in the development of the architecture items and data concepts

6.1 Use case

As stated in [6],

The use case construct is used to define the behaviour of a system or other semantic entity without revealing the entity's internal structure. Each use case specifies a sequence of actions, including variants, that the entity can perform interacting with actors of the entity.

A use case captures the essence of a service requirement. The user of the service (an actor) can be someone or it can be another system outside the target system. As stated in [6],

An actor defines a coherent set of roles that users of an entity can play when interacting with the entity. An actor may be considered to play a separate role with regard to each use case with which it communicates.

In our example (Figure 3), the purpose of the use case is to provide instructions to the driver of the transport means for a load (e.g. at the marshalling yards where an intermodal change is to occur). The primary actor is the driver. Depending on your point of view, you might regard the transponders as within the system boundary or outside, but it is more meaningful here to describe them as actors that are necessarily involved in the functional requirement to instruct a driver where to proceed.

Depending on the scope of a system there can be many use cases, which would not be identified independently. Their specification affects the rest of the architecture and details of the architecture can in turn affect their specification.

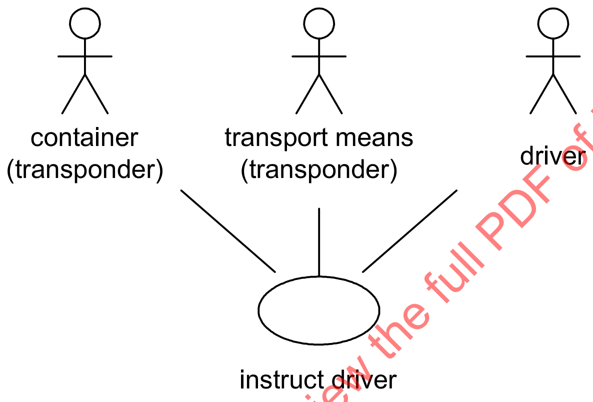

Figure 3 — Use-case diagram for "instruct driver"

The graphical view of Figure 3 alone is not a sufficient specification of the use case. Documentation must give details on the flow of events that delivers the result to the user and in particular the interactions with the actors. For example, when the vehicle approaches the entrance gate of the marshalling yards, the transponders on the vehicle and the vehicle load are interrogated for identification. The identifiers are processed by a control system that has access to database data about the load and data about the vehicle itinerary. The input of these data would be covered by other use cases that would involve the freight forwarder and the transport operator as actors. The control system uses special-purpose systems to determine where the load is to be unloaded and stacked. The location and path directions from the gate to the unload point are then output to the driver (e.g. using a VMS).

In the subsequent elaboration, we will only deal with the information and control for the load management. Similar elaboration would be required for the vehicle management (before and after unloading). However, all the necessary data concepts are identified in this restricted analysis.

The documentation of the use case sets the basis for all the remaining steps. Below is a complete sequence of steps that can be undertaken to ensure that all the situations have been covered. Pre-conditions and post-conditions might be too detailed for most architecture developments. The idea of extension points is to structure the collection of use cases, if possible. For example, in some circumstances a service might have to be expanded by actions that amount to the specification of a separate use case which is an extension of the basic service. The sequence is as follows:

a) use-case name;

b) brief description;

c) flow of events;

d) basic flow;

e) alternative flows;

f) special requirements;

g) pre-conditions;

h) post-conditions;

i) extension points.

6.2 Classifiers

Using the detailed descriptions of all the available use cases, it is possible to take the next step in the architecture definition. This step is to identify the classifiers that will enable the delivery of the services defined in the use cases. We use the inclusive term classifier to include class, component and subsystem. The important commonality of these modelling artifices is that they have structural and behavioural features. Loosely speaking, components and subsystems can be seen as groupings of classes. The most important classifier is the class, which is a description of a set of objects that share the same attributes, operations, relationships and semantics. Components and subsystems are also described by their interfaces (operations). Standard data elements will mostly be defined because they are contained in messages that cross important interfaces.

It is useful to recognize the different purposes that a class serves.

— An information class defines objects, which will store data relevant to the operation of the system and the actors and maintain that data with database-like services.

— A control class defines objects whose primary purpose is to implement the functions of the system.

The process should produce an economic set of classifiers that will effectively and efficiently meet the requirements of all the use cases. An example set which spans the implications of our example use case is shown in Figure 4. The attributes and operations of each classifier have to support a number of collaborative activities. The primary behaviour of these classifiers is control. The auxiliary information classes are defined in a subsequent step, although shipment in Figure 4 is an information class.

Figure 4 includes a number of associations between the classifiers. These arise because of the collaborations between the classifiers which are required in the delivery of the service(s).

AEI reader — An AEI reader component collaborates with a transponder component attached to a load to obtain the appropriate identification. It also collaborates with a monitoring point controller object to inform it of the arrival of the vehicle with a load.

Transponder — A transponder component collaborates with a reader component to provide its stored data.

Monitoring point controller — A monitoring point controller object controls the readers in its area and passes reader information on to the marshaller.

Marshaller — A marshaller object collaborates with monitoring point controller objects to receive notification of the arrival of a vehicle and load. It determines the location within the yard where the load is to be stacked and requests the display of messages to command the stacking.

Display manager — A display manager object manages all the VMS display units in its area.

Transhipper — A transhipper object controls the loading and unloading of non-road vehicles in a multi-modal environment. It also determines the load to be loaded on an empty road vehicle and collaborates with a marshaller object to arrange the movement of the vehicle to the load point. (The scope of this example does not elaborate this class of object.)

Shipment — This information class defines the database which contains the freight-forwarder information about the loads being received and dispatched from the yard.

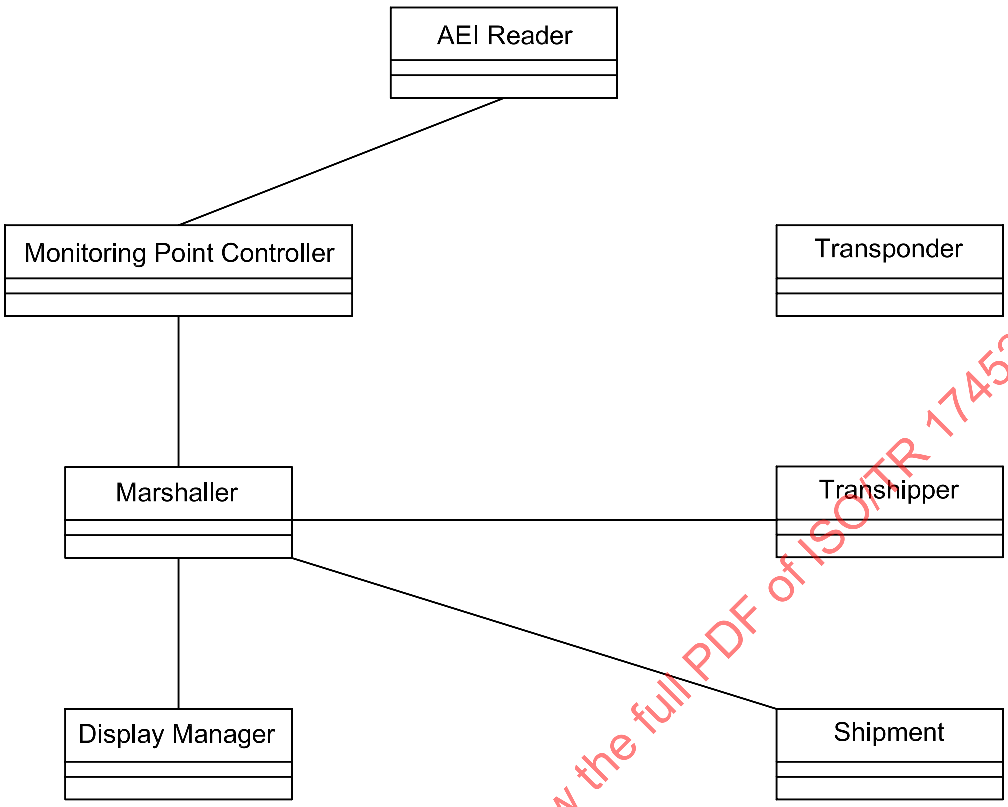

Figure 4 — Classifiers of the application architecture

The following steps serve to elaborate the framework in Figure 3.

6.3 Collaborations

This step is necessary before the specification of the classifiers in the previous step can be completed. The reason is that collaboration between classifiers is executed in terms of the operations that they support. This is illustrated by the interaction diagram shown in Figure 5. This shows the complete set of interactions necessary to perform the service defined by the use case in Figure 3.

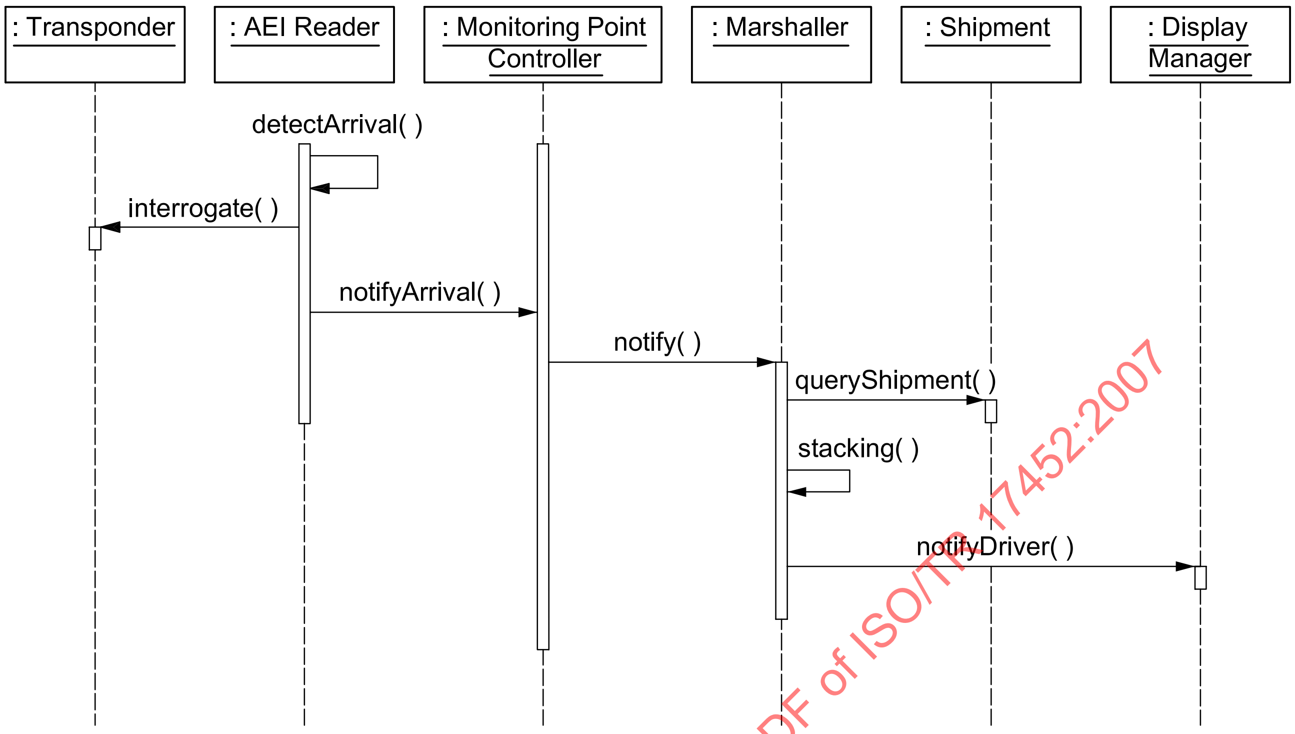

Figure 5 — Interaction between classifiers

6.4 Parameters of the operations

6.4.1 General

This step defines the information necessary to perform the control actions described in Figure 5. Information classes and their attributes are defined. These will source the parameters of the operations. The parameters are used in the detailed implementation of the operations. The information classes and attributes required for our example use case are added to the control classes of Figure 4 to produce Figure 6.

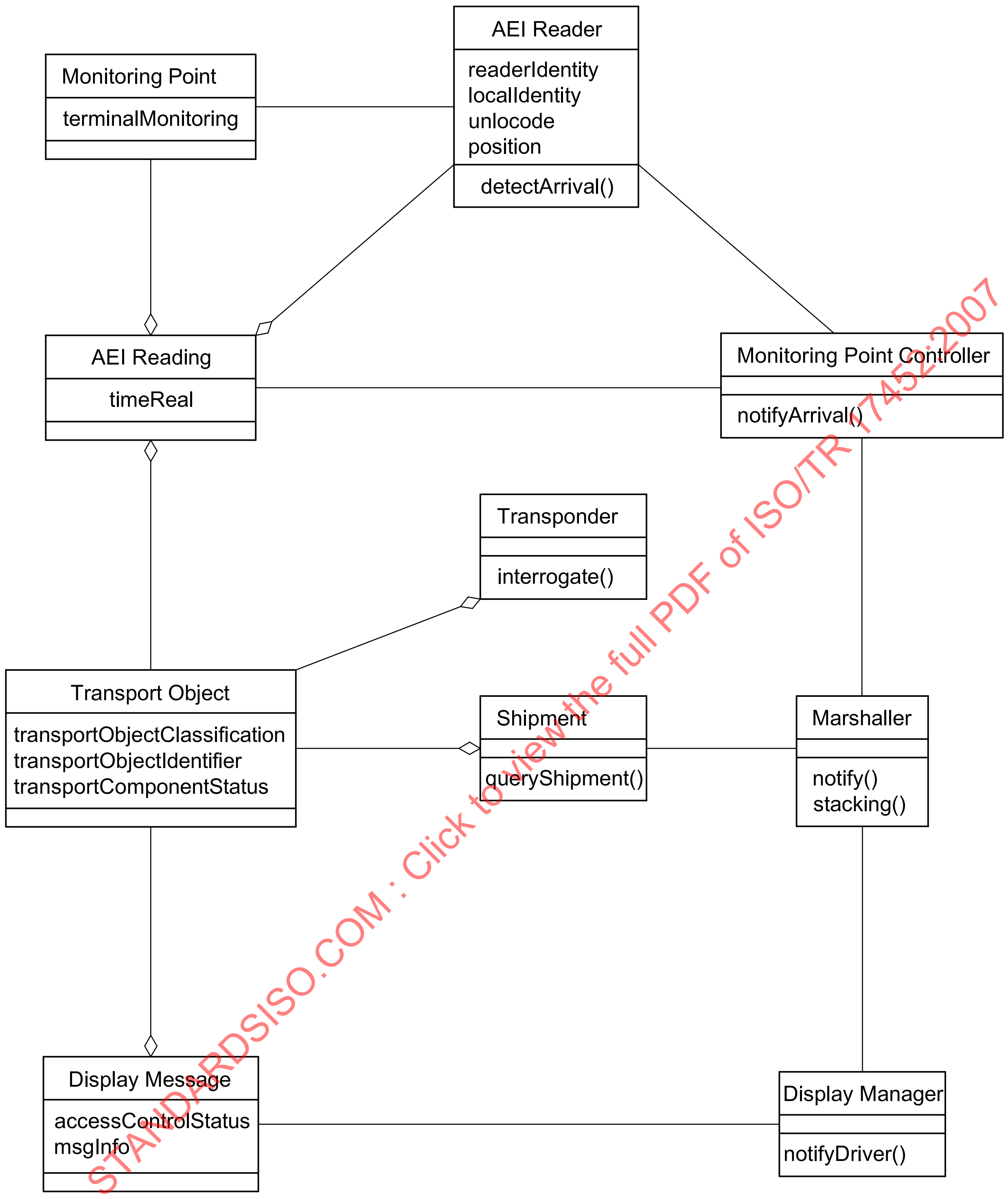

Figure 6 — Information classes and the main classifiers: class attributes

The following sections define the data concepts that will populate the appropriate ITS data registry.

In the first place, the control class AEI Reader is augmented with attributes that record its identity and location.

AEI Reader

`readerIdentity`	a Global Manufacturer Identifier (from ISO 14816)

`localIdentity`	a locally determined reader identity for efficient implementations
`unlocode`	specifies the geographical location of the AEI reader or the terminal monitoring point
`position`	location of the AEI reader relative to a reference point defined by the installation manager

The information class *Monitoring Point* is defined to represent points accessible to a marshaller object where the monitoring of transport objects is performed. It has an attribute which defines the role of the monitoring point.

NOTE *Monitoring Point* is also known as *Terminal Access Control Point*.

Monitoring Point

`terminalMonitoring`	the type of monitoring which the marshaller uses for the purpose of conducting business processes for the control and monitoring of transport means, load units and goods items by AEI.

The information class *Transport Object* is defined to represent the load being delivered by the transport means, or the transport means itself. It has several attributes that describe the real-world object.

NOTE The name has been copied from [4] and might be confusing because of the use of the word “object” in a class name.

Transport Object

`transportObjectClassification`	a qualifier for the type of unit which is included in a transport chain
`transportObjectIdentifier`	the identifier of a transport means, package or goods item NOTE The transport object identifier normally comprises the transponder identity.
`transportComponentStatus`	a status code to indicate the operational status of the components

The information class *Display Message* is defined to represent the information provided to the driver of the transport means via a variable message display or other communication media.

Display Message

`accessControlStatus`	a code issued by the marshaller to indicate the status of the access control of a transport means, load unit or a goods item to a terminal monitoring point
`msgInfo`	a free format message with local semantics

6.4.2 Class operations

The parameters of the individual operations are now specified using the format

class_name.operation_name(*parameter_list*) : *return*

The elements of “*parameter_list*” and “*return*” are derived from the information classes, their attributes and the attributes of the control classes. Where there is a one-to-one correspondence with an element of a parameter list or a return, we use that class name or attribute name in this specification.

AEI Reader.`detectArrival( ) : vehiclePresent`

The return *vehiclePresent* does not correspond to any attribute. It takes on a value of true or false.

Transponder.`interrogate( ) : Transport Object`

The return *Transport Object* contains the information stored in the transponder about the load unit.

Marshaller.`notify arrival(timeReal, AEI Reading)`

The first element of the parameter list would be derived from a system object, not described at this application level. In using the class name *AEI Reading* as the second element of the parameter list, we imply that it also contains the attributes of the classes related by aggregation associations in Figure 6, i.e. the attributes *of Transport Object, Monitoring Point* and *AEI Reader.*

Shipment.`queryShipment( Transport Object.transportObjectIdentifier ) : forwarding`

The return *forwarding* is derived from information contained in a shipment object identified by the parameter *`transportObjectIdentifier`* of the operation *`notifyArrival`*. It is used by the marshaller control object to determine the acceptance of the load unit described by the parameter *`transportObjectIdentifier`.*

Marshaller.`stacking( forwarding)`

This operation is invoked by the marshaller object on itself to compute the stacking instructions for the load unit described in the *forwarding* parameter. The result is used in invoking the `notifyDriver` operation of the *Display Manager.*

Display Manager.`notifyDriver( Display Message, Transport Object.transportObjectIdentifier, displayUnit )`

The first element of the parameter list is the message which will provide the driver instructions on how to proceed for stacking of the load. The second element is the necessary identification of the receiver of the message, which will be included in the display. The third element is a local identifier of the VMS to display the message.

6.5 Significant interfaces

Referring back to Figure 1, we are reminded that our focus is an AVI/AEI standard that can be applied in every load transfer situation. The various requirements enable us to determine the physical subsystem analysis of the architecture. Once the significant subsystem interfaces are determined, we can identify the associated information classes and attributes which it is important to register, because these will need to be standardized.

The subsystem diagram is shown in Figure 7. The physical subsystem analysis is easy to justify in this example. Clearly the tag subsystem must be independent because it is mobile. The reader subsystem is likely to be deployed as a separate equipment package matching the tag equipment. The other subsystems follow accepted practice of separating the display system from the control system.

In Figure 7, the reader subsystem invokes the interfaces of the transponder and the marshalling yard controller subsystems. The latter invokes the interface of the VMS subsystem.

The significant interfaces specified in Figure 7 are elaborated in Figure 8. The details of the parameters are defined in 6.4.

The tag interface is significant because the reader and tag subsystems will be separate deployments, reused in every transhipment yard. The marshalling yard control interface is significant because it must conform to the requirements of the reader subsystem. Thus, while control systems can vary considerably in different types of transhipment yards, they will have to provide a standard interface for readers.

The interface provided by the VMS subsystem and used by the marshalling yard control subsystem is not significant (at least from the point of view of an AVI/AEI standard). However, the interface for the actor is significant because part of the interaction with the actor will use a standard data element (`accessControlStatus`).

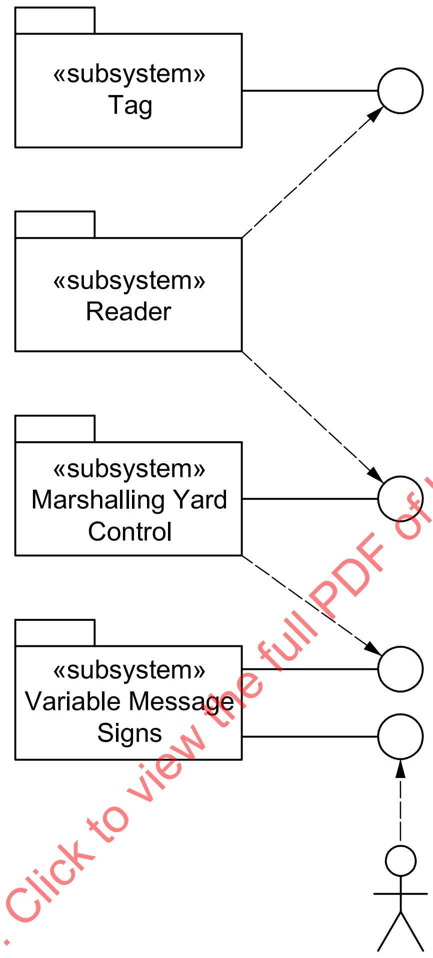
«subsystem» Tag
«subsystem» Reader
«subsystem» Marshalling Yard Control
«subsystem» Variable Message Signs

Figure 7 — Subsystem diagram

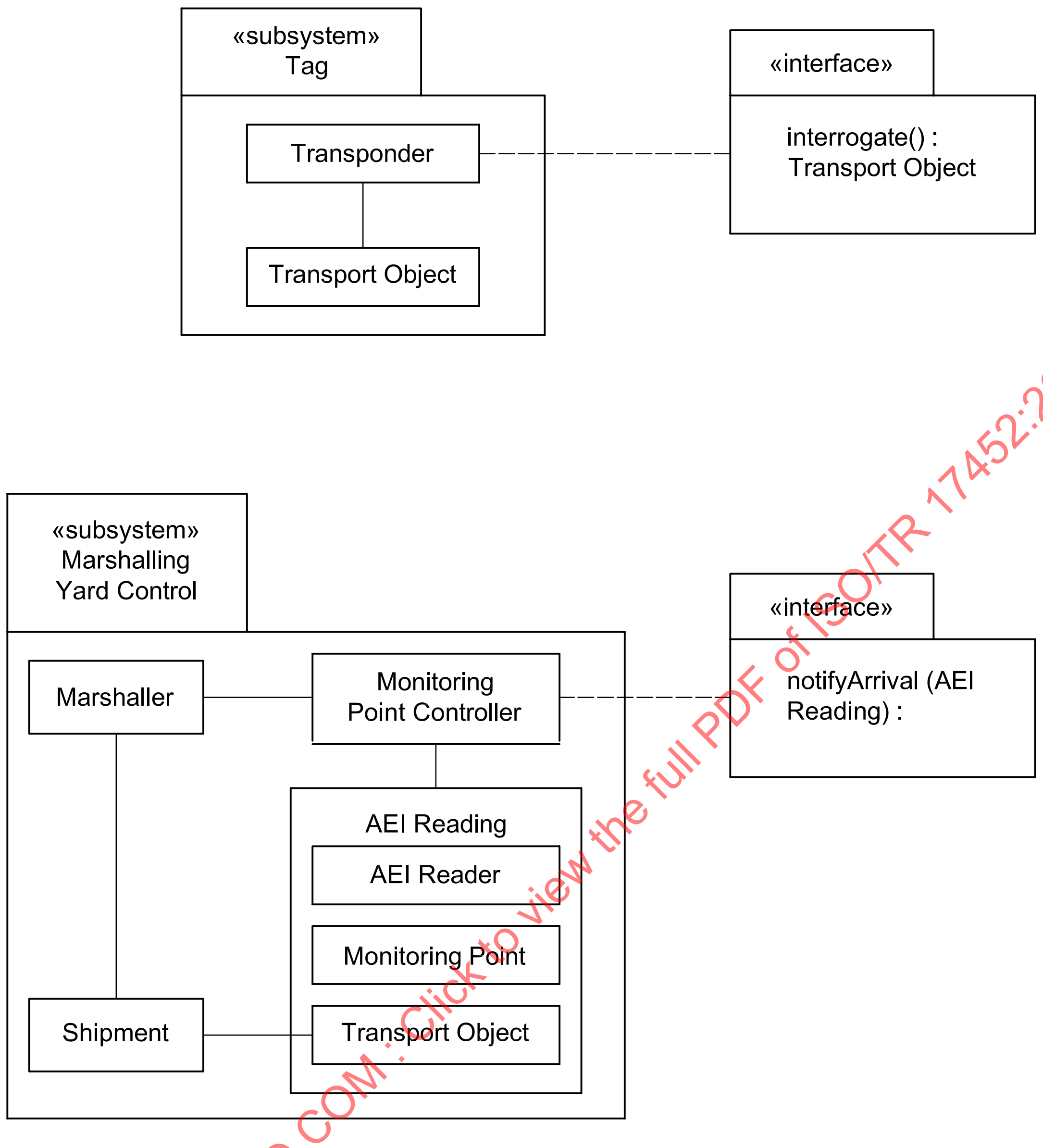

Figure 8 — Significant interfaces

6.6 Messages

In general, we will identify two messages with each interface operation: the invocation message and the return message. The latter can be null.

The example interfaces in Figure 8 have only one operation per interface. However, in general there is more than one operation associated with an interface, so the message specification has to identify the operation of the interface. Thus, the components of the messages for the tag interface are as follows:

a) `TagInterrogateInvokeMessage`

b) `TagInterrogateReturnMessage`

   1) *`transportObjectClassification`*

   2) *`transportObjectIdentifier`*

   3) *`transportComponentStatus`*

For the marshalling yard control interface, the messages are

c) `MarshallingNotifyArrivalInvokeMessage`

   1) *`timeReal`*

   2) *`readerIdentity`*

   3) *`localIdentity`*

   4) *`unlocode`*

   5) *`position`*

   6) *`terminalMonitoring`*

   7) *`transportObjectClassification`*

   8) *`transportObjectIdentifier`*

   9) *`transportComponentStatus`*

d) `MarshallingNotifyArrivalReturnMessage`

6.7 Information model for the interfaces

The information classes of the significant interfaces are few in number and all the classes are associated. Therefore, one information model is sufficient. This is shown in Figure 9.

The attributes of the classes in Figure 9 are the source of all the data elements that it would be desirable to register from the point of view of this example, i.e. from the viewpoint of AVI/AEI as described in [4].

There are other data elements that the designers of this system might wish to document in a data dictionary. However, on the evidence of this example they would not warrant entry in an ISO 14817 data register. The situation would change if other parts of the system were subject to reuse on a wider scale.

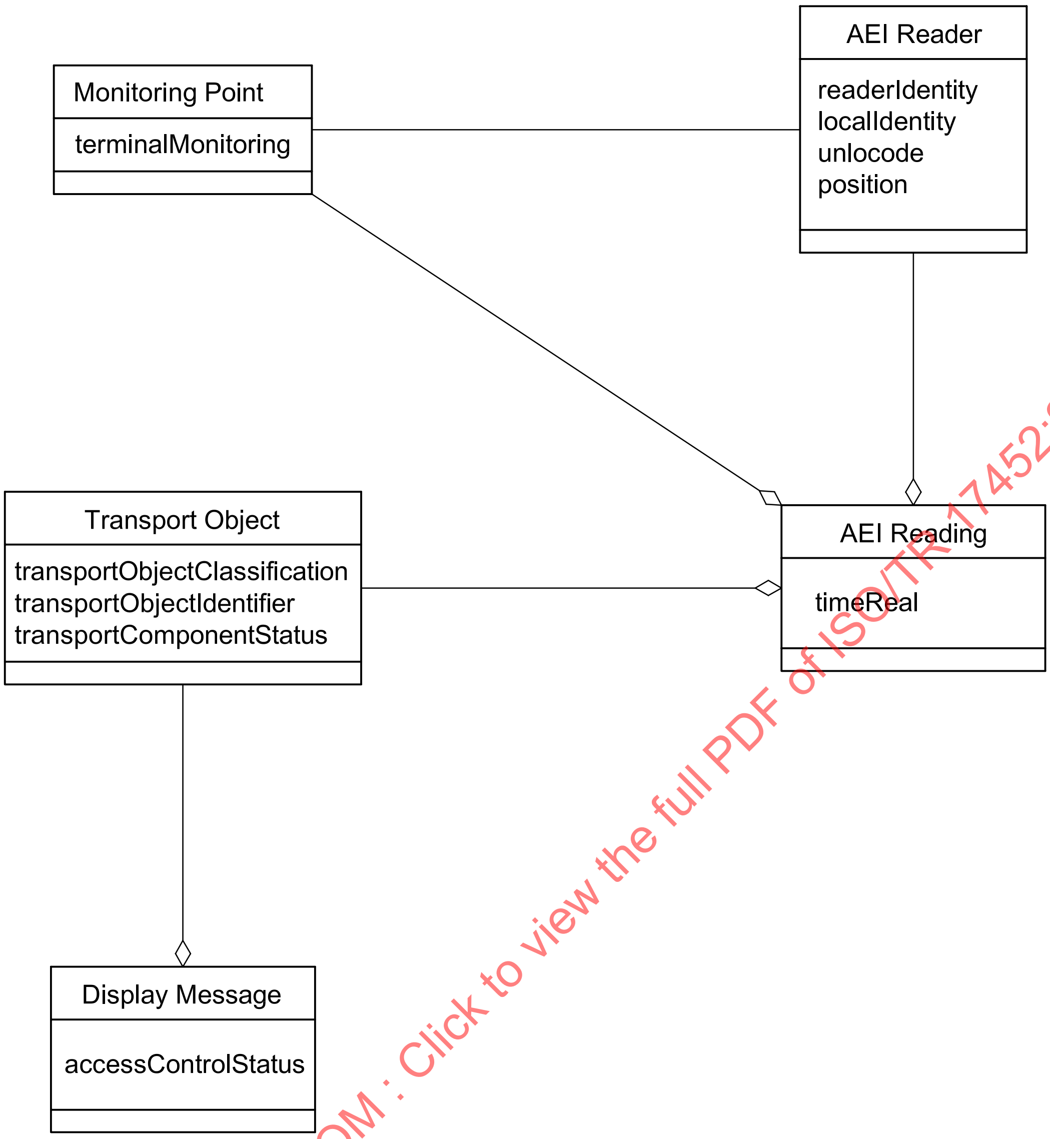

Figure 9 — Information model for the significant interfaces

7 Registering the elements

This clause illustrates the development of the registration requirements for the concepts defined in Clause 6.

For conformance to ISO 14817, the data elements would be documented in a data dictionary, and then submitted to the data registry with the required set of meta attributes filled out. The necessary meta attributes are identified in ISO 14817:2002, Annex C, Tables C.3 and C.4 for a data dictionary and Tables C.1 and C.2 for the data registry submission.

7.1 Example information model

The information model is given in Figure 9. A tabular representation of selected mandatory entries is shown in Table 1 and Table 2.

Table 1 — Object classes

Data concept type = Object class

UML name	Data concept identifier	Descriptive name	Descriptive name context	Definition	Architecture reference	Architecture name	Referenced associations
Monitoring point	1.1	Monitoring point	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	2.1, 2.2
AEI reader	1.2	AEI reader	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	2.1, 2.3
AEI reading	1.3	AEI reading	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	2.2, 2.3, 2.4
Transport object	1.4	Transport object	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	2.4, 2.5
Display message	1.5	Display message	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	2.5

Table 2 — Associations

Data concept type = Association

UML name	Data concept identifier	Descriptive name	Descriptive name context	Definition	Architecture reference	Architecture name	Referenced object classes	Roles	Multiplicity	Aggregate	Role key
Monitoring point – AEI reader	2.1	Monitoring point – AEI reader	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	1.1, 1.2	owns, connected to	1 to *	false	1 0 17262 1 3
AEI reading – monitoring point	2.2	AEI reading – monitoring point	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	1.3, 1.1	located at, owns	* to 1	true	1 0 17262 1 3, 1 0 17262 1 6
AEI reading – AEI reader	2.3	AEI reading – AEI reader	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	1.3, 1.2	product of, made by	* to 1	true	1 0 17262 1 3, 1 0 17262 1 6
AEI reading – transport object	2.4	AEI reading – transport object	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	1.3, 1.4	describes, described by	0..1 to 1	true	
Display message – transport object	2.5	Display message – transport object	AVI/AEI		AVI/AEI intermodal goods transport architecture	AVI/AEI intermodal goods transport architecture	1.4, 1.5	directed at, recipient of	0..1 to 1	true	

7.2 Data element definitions

The data elements that need to be registered correspond to the architecture attributes in Figure 9. A tabular representation of selected mandatory entries is shown in Table 3.

Table 3 — Data element entries

Data concept type = Data element

UML attribute name	Descriptive name	ASN.1 name	ASN.1 object identifier	Definition	Descriptive name context	International Standard	Data type	Format	Unit of measure	Valid value rule
`MonitoringPoint. terminalMonitoring`	`MonitoringPoint. TypeOfMonitoring. code`	TERMINAL MONITORING	1 0 17262 1 1		AVI/AEI	ISO/TS 17262	BitStringType			0-8
`AEIReader. readerIdentity`	`AEIReader. ISO14816Identifier. identifier`	READER IDENTITY	1 0 17262 1 2		AVI/AEI	ISO 14816	BitStringType			
`AEIReader. localIdentity`	`AEIReader. LocalIdentifier. Identifier`	LOCAL IDENTITY	1 0 17262 1 3		AVI/AEI	ISO/TS 17262	IntegerType1			1-65535
`AEIReader. unlocode`	`AEIReader. GeographicLocation. code`	UNLOCODE	1 0 17262 1 4		AVI/AEI	ISO ?	OctetStringType			
`AEIReader. position.xCoordinate`	`AEIReader. LocallyReferenced XPosition. number`	XPOSITION	1 0 17262 1 5		AVI/AEI	ISO/TS 17262	IntegerType		metre	
`AEIReader. position.yCoordinate`	`AEIReader. LocallyReferenced YPosition. number`	YPOSITION	1 0 17262 1 6		AVI/AEI	ISO/TS 17262	IntegerType		metre	
`AEIReader. position.zCoordinate`	`AEIReader. LocallyReferenced ZPosition. number`	ZPOSITION	1 0 17262 1 7		AVI/AEI	ISO/TS 17262	IntegerType		metre	
`AEIReading. timeReal`	`AEIReading. LocalTimeReference. number`	TIMEREAL	1 0 17262 1 6		AVI/AEI	ISO/TS 17262	IntegerType		second	