TECHNICAL REPORT

ISO/IEC TR 14369

Second edition
2018-01

Information technology — Programming languages, their environments and system software interfaces — Guidelines for the preparation of language-independent service specifications (LISS)

Technologies de l'information — Langages de programmation, leurs environnements et interfaces du logiciel d'exploitation — Lignes directrices pour l'élaboration de spécifications de service indépendantes du langage (LISS)

Reference number
ISO/IEC TR 14369:2018(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This second edition cancels and replaces ISO/IEC TR 14369:1999 and ISO/IEC TR 14369:2014, of which it constitutes a minor revision.

The main changes compared to the previous edition are as follows:

— the reference section has been corrected/updated;

— editorial changes have been made to fully align with ISO/IEC Directives.

Introduction

This document is dedicated to Brian L. Meek in grateful recognition of his leadership and vision in the development of the concepts on programming language independent specifications, and his efforts in producing a set of documents in this area. Without his commitment this document never would have been published.

0.1 Background

This document provides guidance to those writing specifications of services, and of interfaces to services, in a language-independent way, in particular as standards. It can be regarded as complementary to ISO/IEC TR 10182, which provides guidance to those performing language bindings for such services and their interfaces.

NOTE 1 Here and throughout, "language", on its own or in compounds like "language-independent", means "programming language", not "specification language" nor "natural (human) language", unless explicitly stated.

NOTE 2 A "language-independent" service or interface specification can be expressed using either or both of a natural language like English or a formal specification language like VDM-SL or Z. In a sense, a specification can be regarded as "dependent" on VDM-SL, for example. The term "language-independent" does not imply otherwise, since it refers only to the situation where programming language(s) can otherwise be used in defining the service or interface.

The development of this document was prompted by the existence of an earlier draft IEEE Technical Report (IEEE TCOS-SCC Technical Report on Programming Language Independent Specification Methods, draft 4, May 1991). The TCOS draft was concerned with specifications of services in a POSIX systems environment, and as such contained much detailed POSIX-specific guidance; nevertheless it was clear that many of the principles, if not the detail, were applicable much more generally. This document was conceived as a means of providing such more general guidance. Because of the very different formats, and the POSIX-related detail in the TCOS draft, there is almost no direct correspondence between the two documents, except in the discussion of the benefits of a language-independent principles below. However, the spirit and principles of the TCOS draft were of great value in developing this document, and reappear herein, albeit in much altered and more general form.

NOTE 3 The TCOS draft has not in fact been published, as the result of an IEEE decision to concentrate activities in other POSIX areas.

0.2 Principles

Service or interface specifications that are independent of any particular language, particularly when embodied in recognized standards, are increasingly seen as an important factor in promoting interoperation and substitution of system components, and reducing dependence on and consequent limitations due to particular language platforms.

NOTE It is possible for a specification to be "independent" of a particular language in a formal sense, but still be dependent on it through inbuilt assumptions derived from that language which do not necessarily hold for other languages. The term "language-independent" here is meant in a much stronger sense than that, though complete independence from all inbuilt assumptions can be difficult if not impossible to achieve.

Potential benefits from language-independent service or interface specifications include:

— A language-independent interface specification specifies those requirements that are common to all language bindings to that interface, and hence provides a specification to which language bindings can conform.

— A language-independent interface specification is a re-usable component for constructing language bindings.

— A language-independent interface specification aids the construction of language bindings by providing a common reference to which all bindings can relate. Through this common reference it is possible to make use of pre-existing language bindings to language-independent standards

for common features such as datatypes and procedure calls, and to other language-independent specifications with related concepts.

— A language-independent service or interface specification provides an abstract specification of a service in isolation from language-dependent extensions or restrictions, and hence facilitates more rigorous modelling of services and interfaces.

— Language-independent service specifications facilitate the specification of relationships between one service and another, by making it easier to relate common concepts than is generally possible when the specifications are dependent on different languages.

— A language-independent interface specification facilitates the definition of relationships between different language bindings to a common service (such as requirements for interoperability between applications based on different languages that are sharing a common service implementation), by providing a common reference specification to which all the languages can relate.

— A language-independent interface specification facilitates the definition of relations between bindings to multiple services, including the requirements on management of multiple name spaces.

— A language-independent service or interface specification brings economic benefits by reducing the effort and resources needed to ensure compatibility and consistency of behaviour between implementations of the same service in different languages or between applications based on different languages using the same interface.

Information technology — Programming languages, their environments and system software interfaces — Guidelines for the preparation of language-independent service specifications (LISS)

1 Scope

This document provides guidelines to those concerned with developing specifications of information technology services and their interfaces intended for use by clients of the services, in particular by external applications that do not necessarily all share the environment and assumptions of one particular programming language. The guidelines do not directly or fully cover all aspects of service or interface specifications, but they do cover those aspects required to achieve language independence, i.e. required to make a specification neutral with respect to the language environment from which the service is invoked. The guidelines are primarily concerned with the interface between the service and the external applications making use of the service, including the special case where the service itself is already specified in a language-dependent way but needs to be invoked from environments of other languages. Language bindings, already addressed by ISO/IEC TR 10182, are dealt with by providing advice on how to use the two documents together.

This document provides technical guidelines, rather than organizational or administrative guidelines for the management of the development process, though in some cases the technical guidelines can have organizational or administrative implications.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
client
application (typically a program in some language) which makes use of a *service* (3.13)

Note 1 to entry: The term “service user” (3.15) is often used in a similar sense, where “client” more often implies the physical computer system on which the application is running, rather than just the application itself.

3.2
datatype
set of values, usually accompanied by a set of operations on those values

3.3
formal language
formal *specification language* (3.16)

3.4
interface
mechanism by which a *service user* (3.15) invokes and makes use of a *service* (3.13)

3.5
language
programming language, not *specification language* (3.16) or natural (human) language, unless otherwise qualified

3.6
language binding
specification of the standard interface to a *service* (3.13), or set of services, for applications written in a particular programming language

3.7
language-dependent
state of making use of the concepts, features or assumptions of a particular programming language

3.8
language-independent
state of not making use of the concepts, features or assumptions of any particular programming language or style of language

3.9
language processor
entire computing system which enables a programming language user to translate and execute programs written in the language, in general consisting both of hardware and of the relevant associated software

[SOURCE: ISO/IEC TR 10176:2003, 3.1, modified — the words "denotes the" at the beginning of the definition, and the two Notes to entry were deleted.]

3.10
mapping
defined association between elements (such as concepts, features or facilities) of one entity (such as a programming language, or a specification, or a standard) with corresponding elements of another entity

Note 1 to entry: Mapping when used as a verb is a process of determining or utilizing a mapping.

Note 2 to entry: Mappings are usually defined as being from one entity into another. A language binding of a language L into a standard S usually incorporates both a mapping from L into S and a mapping from S into L.

Note 3 to entry: Depending on what is being mapped, a mapping is not necessarily one-to-one. This means that mapping an element E from system A into an element E' of system B, followed by mapping E' back into system A, does not necessarily get back to the original E. In such situations, if a two-way correspondence is to be preserved, the execution of the mappings needs to include recording the place of origin and returning to it.

3.11
marshalling
process of collecting the actual parameters used in a procedure call, converting them if necessary, and assembling them for transfer to the called *procedure* (3.12)

Note 1 to entry: This process is also carried out by the called procedure when preparing to return the results of the call to the caller.

Note 2 to entry: Marshalling can be regarded as being performed by a service user when preparing input values for a service provider, and by a service provider when preparing results for a service user, the service concerned being regarded as the procedure being called.

3.12
procedure
subprogram which can return a value

Note 1 to entry: A procedure that returns a value is sometimes called a subroutine, a procedure that does not return a value is sometimes called a function.

Note 2 to entry: Some programming languages use different terminology.

3.13
service
facility, or set of facilities, made available to *service users* (3.15) through an *interface* (3.4)

3.14
service provider
server
computer system, or set of computer systems, that implements a *service* (3.13) and makes it available to *service users* (3.15)

Note 1 to entry: In this definition, "computer system" means a logical system, not a physical system; it can correspond to part of all of one or more physical computer systems.

Note 2 to entry: The term "server" is often used in a similar sense, though sometimes implying a physical computer system that has no other function than to provide its service

3.15
service user
application (typically a program in some language) which makes use of a *service* (3.13)

Note 1 to entry: The term "client" (3.1) is often used in a similar sense, though sometimes implying the physical computer system on which the application is running, rather than just the application itself.

3.16
specification language
formal language for defining the semantics of a service or an interface precisely and without ambiguity

3.17
unmarshalling
process of receiving and disassembling transferred parameters, and converting them if necessary, to prepare the values for further use

Note 1 to entry: This process is carried out by the called procedure on receipt of the actual parameters for the call, and by the caller on receipt of the returned results of the call.

Note 2 to entry: Unmarshalling can be regarded as being performed by a service provider when receiving input values from a service user, and by a service user when receiving results from a service provider, the service concerned being regarded as the procedure being called.

3.18
Z
<mathematics>complex numbers

Note 1 to entry: See ISO/IEC 10967-1.

3.19
Z
type of formal *specification language* (3.16)

Note 1 to entry: It is pronounced "zed".

Note 2 to entry: See ISO/IEC 13568.

4 Abbreviated terms

GPD general-purpose datatypes, as defined in ISO/IEC 11404:2007

LID language-independent datatypes, as defined in ISO/IEC 11404:1996; the LID specifications used in this document are identical to the corresponding specifications in ISO/IEC 11404:2007 (GPD)

LIPC language-independent procedure calling, as defined in ISO/IEC 13886

RPC remote procedure call, as defined in ISO/IEC 11578:1996

5 Overview

5.1 Services, interfaces, service providers and service users

The concept of a "service" is a very general one. In some contexts it is customary to use it in a restricted sense, e.g. when talking about "service industries" as contrasted with "manufacturing industries". Despite such usages, almost any activity or behaviour can be regarded as a "service", if it serves some useful purpose to do so (for example, manufacturing spoons can be regarded as a service for those needing spoons).

With the concept of a service come the concepts of a "service provider" and a "service user". The provider performs the activity that constitutes the service; the user is the customer or the client for the service, for whom the service is performed. In the information technology field, the "client-server model" incorporates these concepts: the server provides, the client uses.

Between the service provider and the service user is an interface that allows them to communicate. The service user communicates through the interface the requirement for the service, and any relevant information (e.g. not only the need for spoons, but the number and size of spoons required), and the service provider communicates through the interface the response to the order for the service, and any additional information or queries (e.g. the spoons can be delivered in six days, do you want silver spoons or plastic spoons?). In the information technology field, such interfaces are usually explicit, realized in hardware or software or both. In the world in general, they are sometimes explicit, but sometimes subsumed in more general human or other interactions.

This distinction between provider and user (client and server) should not be assumed to correspond to identifiable distinct entities. The distinction, and the service interface, can be purely notional, and possibly not normally thought of in that way. The service itself can similarly not correspond to a distinct, separate activity, and again possibly not normally thought of as such; it can be subsumed in some other activity or group of activities, and can possibly be implicit.

Hence, for example, in a transaction between two parties, each one can be providing a service for the other: each is a client, and each a server. In another context, the provider is providing the service to itself; the provider is also the user. Though it is possible to subdivide the provider/user into a provider part and a user part when considering provision of the service, this can be inconvenient in other respects.

In summary, "client" and "server", are roles that are carried out, rather than elements that necessarily need to be implemented separately. Though the term "client-server" is sometimes used in the information technology field in ways that are more specific than it is used here, it is important not to carry over assumptions from particular client-server models when reading this document. It is even more important not to assume that implementation of any service, in the sense used here, needs to be done using a client-server model.

5.2 Information technology services

The history of information technology has many instances of the technology, or a product, being used for very different purposes and in very different ways from those originally envisaged. The kinds of

