TECHNICAL REPORT

ISO/IEC TR 14143-5

First edition
2004-04-01

Information technology — Software measurement — Functional size measurement —

Part 5: Determination of functional domains for use with functional size measurement

Technologies de l'information — Mesurage du logiciel — Mesurage de la taille fonctionnelle —

Partie 5: Détermination des domaines fonctionnels pour l'usage de mesurage de la taille fonctionnelle

Reference number
ISO/IEC TR 14143-5:2004(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web 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.

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

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

In exceptional circumstances, the joint 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 the joint 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).

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.

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.

ISO/IEC TR 14143-5, which is a Technical Report of type 2, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and system engineering*.

This document is being issued in the Technical Report (type 2) series of publications (according to the Procedures for the technical work of ISO/IEC JTC 1) as a "prospective standard for provisional application" in the field of software measurement 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/IEC TR 14143 consists of the following parts, under the general title *Information technology — Software measurement — Functional size measurement*:

— *Part 1: Definition of concepts*

— *Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1:1998*

— *Part 3: Verification of functional size measurement methods* [Technical Report]

— *Part 4: Reference model* [Technical Report]

— *Part 5: Determination of functional domains for use with functional size measurement* [Technical Report]

The following part is under preparation:

— *Part 6: Guide for use of ISO/IEC 14143 series and related International Standards*

Introduction

Functional Size Measurement (FSM) is a technique used to measure the size of software by quantifying the Functional User Requirements of the software[1)]. The first published method to embrace this concept was Function Point Analysis, developed by Allan Albrecht in the late 1970s. Since then, numerous extensions and variations of the original method have been developed. Users of these methods have made various claims regarding the usefulness and limitations of a particular method when applied to different *types* of software. Examples of these *types* of software quoted include, amongst others, Management Information Systems (MIS), Embedded Software, Process Control Software, Decision Support Software, Military Software, Engineering and Real Time Software.

The terms and the phrase "*software types*" have been loosely defined. They are widely used to differentiate between categories of user functions performed by the software (what it does), software performance issues, degrees of internal processing complexity, physical implementation requirements and development environments. There is no universally consistent definition of these terms or of the characteristics of FUR relevant to assessing Functional Size. The consequences of this are:

a) it is difficult for a potential user of a particular FSM Method to assess the Method's applicability for measuring the size of a specific set of FUR; and

b) owners and developers of an FSM Method are not able to describe the Functional Domain(s) to which the FSM Method can be applied.

This Technical Report addresses these difficulties by describing how the characteristics of FUR may be used to determine software types. The phrase "software types" for the purposes of this Technical Report is replaced by the defined term Functional Domain. A Functional Domain is defined in ISO/IEC 14143-1:1998 as "a class of software based on the characteristics of FUR which are pertinent to Functional Size Measurement."

ISO/IEC 14143-1:1998 requires that an FSM Method shall describe the Functional Domain(s) to which it can be applied. The purpose of this Technical Report is to define how Functional Domains may be defined and to provide example methods that may be used to generate Functional Domains.

To ensure that this Technical Report did not unnecessarily duplicate established Functional Domains, were these to exist in general information technology, a literature search consisting of formal library retrievals, informal library reviews, personal correspondence, conference proceedings and on-line searches was conducted between June 1995 and May 1997 referencing topics pertinent to this project. Over 700 abstracts were reviewed in addition to articles, periodicals, conference presentations and other references as background to this Technical Report.

This Technical Report satisfies the unique needs of FSM, and therefore takes a different approach from ISO/IEC TR 12182. The classification categories of ISO/IEC TR 12182 were considered, and have been mapped in the informative annex.

1) Refer ISO/IEC 14143-1:1998, *Software engineering — Software measurement — Functional size measurement — Part 1: Definition of concepts*.

Information technology — Software measurement — Functional size measurement —

Part 5: Determination of functional domains for use with functional size measurement

1 Scope

This Technical Report describes the characteristics of Functional Domains and the procedures by which characteristics of Functional User Requirements (FUR) can be used to determine Functional Domains. Two example methods for implementing these principles are provided in the Informative Annexes.

Either of the methods may be used directly, or using Functional Domains defined locally by:

— FSM Method to determine if a particular FSM Method is applicable to the Functional Domain(s) represented by their specific FUR;

— Describing, for a given set of FUR, the Functional Domain to which the FUR belong; and

— FSM Method owners and designers describing the Functional Domain(s) to which the FSM Method can be applied, as outlined in ISO/IEC 14143-1:1998.

NOTE Use of the Informative Annexes to specify Functional Domains will allow comparisons of FUR from different sources and comparisons of the applicability of FSM Methods.

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/IEC 14143-1:1998, *Information technology — Software measurement — Functional size measurement — Part 1: Definition of concepts*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14143-1:1998 and the following apply.

3.1
characteristic of FUR
a distinctive property of the FUR that is important for identifying the Functional Domain to which a specific set of FUR belongs

3.2
Functional Domain Categorization (FDC)
a process for identifying Functional Domains that conforms to the requirements of 5.3 of this part of ISO/IEC 14143

4 Abbreviations

FDC functional domain categorization

5 Requirements

5.1 General requirements for Functional Domains

A Functional Domain shall be

— distinct from all other Functional Domains, and

— described in terms of a set of characteristics that are relevant to functional size.

A Functional Domain should be named using Information Technology industry recognized terms, extended as appropriate to indicate their local origin, e.g. 'Organization X Real-Time'. This allows the Functional Domains to gain acceptability within the context of FSM (and potentially other software areas where the requirement for software classifications exists, such as for software development and maintenance).

NOTE The explicit use of industry terms such as: real time, MIS and process control, without further qualification in the titles of the sample Functional Domains has been avoided to this point, because their use (or misuse) in the Information Technology industry generally includes quality and technical considerations in addition to FUR. An explicit statement may need to be made to reinforce that technical and quality characteristics of the software are not part of a defined Functional Domain. For example, the term "real time" in industrial use may imply timing constraints (technical or qualitative), specific hardware (technical), immediate feedback (quality), communication with non-human users (software/hardware), safety criticality and reliability (quality), fault tolerance (quality), plus actual FUR (e.g., position control of flight surface in an aircraft).

A set of FUR may be categorized as belonging to more than one Functional Domain.

5.2 General requirements for characteristics of Functional Domains

A Characteristic of a Functional Domain shall:

a) inherit all of the characteristics of Functional Size, as defined in ISO/IEC 14143-1:1998,

b) be exhibited by Functional User Requirements,

c) be independent of methods to develop software, and

d) be independent of quality and technical requirements.

5.2.1 Requirements for an FDC method

An FDC method should:

a) be repeatable;

b) be consistent;

c) result in descriptions of Functional Domains that have the same meaning at some future time as when they were derived;

d) be structured to facilitate understanding; and

e) define characteristics of a Functional Domain.

NOTE Annexes A and B show examples of FDC methods. Other methods may be possible.

6 Procedures

6.1 Determining the Functional Domain for a given set of FUR

To determine the Functional Domain for a set of FUR:

a) identify the sets of characteristics that the set of FUR exhibits, using an FDC method;

b) match the sets of characteristics against the sets of characteristics in reference Functional Domains, as defined using the selected FDC method;

c) determine the Functional Domain(s) applicable to the set of FUR.

6.2 Determining the applicability of an FSM Method to a particular Functional Domain

To determine the applicability of an FSM Method to a particular Functional Domain:

a) identify the characteristics that the FDC method defines for the particular Functional Domain;

b) for the selected FSM Method, for each BFC Type, identify which of the above characteristics are recognized;

c) compare the characteristics of the Functional Domain with those recognized by the FSM Method; and

d) if the characteristics exhibited by the Functional Domain:

   1) are a subset of characteristics that the FSM Method recognizes, then the FSM Method is applicable to the Functional Domain,

   2) are superset of characteristics that the FSM Method recognizes, then the FSM Method is partially applicable to the Functional Domain, or

   3) do not correspond to any characteristics that the FSM recognizes, then the FSM Method is not applicable to the Functional Domain.

7 Example FDC methods

Based on the procedures described in clause 6, the Informative Annexes are examples of separate and self-contained FDC methods that show different approaches to determining Functional Domains. Either of these methods, or any other structured method that conforms to clause 5.3 of this part of ISO/IEC 14143, may be used to help determine the Functional Domains for which an FSM Method is applicable.

ISO/IEC 14143-1:1998 requires that an FSM Method describe the Functional Domain(s) to which it can be applied. Owners of such FSM Methods should describe the Functional Domain(s) to which they apply using the concepts of the methods of the Informative Annexes or another comparable structured method. Alternately, owners of an FSM Method may use these methods to generate locally defined and named Functional Domains.

NOTE 1 The concepts of existing FSM Methods may or may not correspond exactly to the concepts of the methods of the Informative Annexes, as many FSM Methods were designed and defined before these methods were developed.

Informative Annex A gives an example of a ‘top-down,’ pragmatic approach to determine Functional Domains based on extensive literature research of categorizations used within and outside of the software engineering world, in alliance with software engineering concepts used to distinguish what is normally understood by Functional Domain.

Informative Annex B gives an example of a ‘bottom-up’ approach using established software engineering concepts to define a set of BFC Types (as defined in ISO/IEC 14143-1:1998) to determine Functional Domains.

NOTE 2 Informative Annex B gives an example of a ‘bottom-up’ approach using established software engineering concepts to define a set of BFC Types (as defined in ISO/IEC 14143-1:1998) to determine Functional Domains.

Annex A
(informative)

CHAR Method to determine Functional Domains

A.1 Purpose

The purpose of this Informative Annex is to:

a) define an FDC method,

b) use this method to generate example definitions of various specific Functional Domains, and

c) test the method against examples of FUR and some existing candidate FSM Methods.

ISO/IEC 14143-1:1998 requires that:

a) FSM Methods measure Functional Size by identifying the 'Base Functional Component Types (or 'BFC Types') that they recognize within the FUR and assigning values that are used to calculate the Functional Size,

b) a BFC Type is a defined category of 'elementary units of FUR defined by and used by an FSM Method for measurement purposes', and

c) FSM Methods shall describe the Functional Domains to which they can be applied.

This Informative Annex:

a) can be used to describe or to define various practical Functional Domains from FUR, and

b) can be used by the designers of FSM Methods to state which characteristics are recognized by the BFC types of their FSM Method.

Each test request in the list of test requests shall

a) be numbered so that it can be uniquely identified,

b) identify the corresponding statement within the FSM Method, if applicable,

c) identify the performance property to which the test refers,

d) state the verification method to be used for the verification test, and

NOTE Verification methods are described in Annex B.

e) be precisely described in accordance with the provisions of A.2.

A.2 Definitions

A.2.1 CHARs

Specific characteristics of FUR selected for the Informative Annex A method.

A.2.2 CHAR Group

A collection of related CHARs that is used to determine a distinctive Functional Domain.

A.3 Characteristics of FUR relevant to FSM (CHARs)

A.3.1 Background

Based on the CHAR Group method, there are three CHAR Groups that distinguish Functional Domains. Through the evaluation and rating described in A.3.2, a set of FUR can be evaluated to find its specific Functional Domain.

A.3.1.1 The CHAR Group model

There are three groups of CHARs, each of which contains a variable number of CHARs. Each CHAR can only belong to one CHAR Group and is evaluated separately:

a) **Control- and Communication-Rich**

The FUR dictate that the software must operate concurrently with or control the users of its external world. The CHARs are:

1) Response: The requirements of the software contain timing constraints that translate into FUR.

2) Interfaces: Software interfaces to control or communicate with external objects or other external software are critical.

3) System Management: Software manages external environments to detect out of bounds / emergency data values, upon which its processing sequence can be adjusted (e.g., high priority stimulus can interrupt or alter processing of services).

NOTE Whether the software monitors the external environment or the external environment provides the stimulus for data monitoring, are two ways of implementing the same Functional User Requirement.

b) **Data-Rich**

The data architecture, relationship requirements, persistence of the data beyond the completion of a transaction prescribed by the FUR is a dominant consideration for the software. The CHARs include:

1) Complex Data: Complex data or control relationships/interdependencies requirements are an important part of the FUR.

2) Persistence: The persistence, or logical storage, of data is an important part of the FUR.

NOTE Data persistence in this Informative Annex refers to the requirement(s) that entered/polled data be retained over a period of time it strictly considers whether or not data must be retained beyond the single, completed transaction.

c) **Manipulation- and Algorithm-Rich**

The FUR specify that the software must perform particular types of algorithmically intensive services and / or complex operations. The CHARs include:

1) Manipulation: High manipulation of data (e.g. dominance of different logical functions operating on simple data).

2) Scientific/engineering: Scientific/engineering, mathematical or logical algorithms required (e.g. involves precision and accuracy, statistical analysis, etc.).

NOTE For example, Monte Carlo simulations are complex statistical algorithms required in some software estimation models.

3) Adaptive: Software is required to be adaptive (i.e. software dynamically changes its behavior or logic based on external or events) or software has business process rules that adjust based on date, time, season or other external considerations.

NOTE For example, life insurance actuarial software contains rules that govern the composition of algorithms used in particular instances. The algorithms are not static.

A.3.2 Procedure to evaluate FUR or group of FUR to determine the Functional Domain(s)

Given a set of FUR, the procedure for determining the Functional Domain to which they belong is carried out in three steps. First, an evaluation and rating of each CHAR is made and recorded in Table A.1. Second, the resultant CHAR ratings are used to establish the ratings for the CHAR Groups (see Table A.2). Finally the combination of CHAR Group ratings will determine the Functional Domain of the FUR (see Table A.3).

a) Evaluate the importance of each CHAR for a given set of FUR by rating each as negligible, present or dominant. Table A.1 is provided to increase the consistency and objectivity when rating the CHARs. It provides example FUR for each category, in answer to the following question for each CHAR:

Do the FUR exhibit this CHAR?

Table A.1 — CHAR evaluation

CHAR	**Rating = Negligible** (representing < 3 % of requirements)	**Rating = Present** (representing 3 % to 50 % of requirement)	**Rating = Dominant** (representing > 50 % of requirement)	**Rating** (Negligible Present or Dominant)
Control- and Communication-Rich				
1. Response	batch reporting	enquiring on a bank balance	monitoring pipeline pressure	
2. Interfaces	emulating an arithmetic calculator	processing a credit card payment	controlling a fuel injection system	
3. System Management	processing monthly payroll	autopilot	controlling a nuclear reactor	
Data-Rich				
4. Complex data	registering frequent fliers	managing organizational finances	monitoring organized crime	
5. Persistence	searching the Internet	navigating a website	monitoring an experimental rocket	
Manipulation- and Algorithm-Rich				
6. Manipulation	checking in to a hotel	managing inventory	online medical diagnosis	
7. Scientific / engineering	monitoring an online shopping trolley	interpreting data from Hubble Space Telescope	predicting the weather	
8 Adaptive	checking for spelling errors	scanning email for viruses	controlling missile trajectory	

b) Based on the CHAR results obtained from step 1, use Table A.2 to establish the ratings for the CHAR Groups.

Table A.2 — CHAR Group evaluation

CHAR Group	Rating = Negligible (no CHAR present, or sum of CHARs < 3 % of requirement)	Rating = Present (at least one CHAR present and sum of CHARs < 50 % of requirement)	Rating = Dominant (one CHAR dominant or sum of CHARs > 50 % of requirement)	Rating (Negligible, Present or Dominant)
Control- and Communication-Rich	Use results for *Response, Interfaces* and *System Management* CHARs from Table A.1			
Data-Rich	Use results for *Complex Data* and *Persistence* CHARs from Table A.1			
Manipulation- and Algorithm-Rich	Use results for *Manipulation*, *Scientific/Engineering* and *Adaptive* CHARs from Table A.1			

c) Use the CHAR Group ratings established in step 2 to select the appropriate Functional Domain from Table A.3.

Table A.3 — Functional Domains

Functional Domain	Control- and Communication-Rich	Data-Rich	Manipulation- and Algorithm-Rich
Pure Data Handling System	negligible	dominant	negligible
Information System	negligible	dominant	present
Data Processing System	negligible	present	present
Controlling Information System	present	dominant	negligible
Controlling Data System	present	present	negligible
Complex Controlling Information System	present	dominant	present
Non-Specific (Complex) System	present	present	present
Simple Control System	dominant	negligible	negligible
Control System	present	negligible	present
Complex Control System	dominant	negligible	present
Data Driven Control System	dominant	present	negligible
Complex Data Driven Control System	dominant	present	present
Pure Calculation System	negligible	negligible	dominant
Controlling Calculation System	present	negligible	dominant
Scientific Information System	negligible	present	dominant
Scientific Controlling Data Processing System	present	present	dominant

NOTE This table provides examples of functional domains and is not intended to be an all-inclusive list.

A.3.3 CHAR summary

Figure A.1 summarizes the CHAR evaluation procedure outlined in A.3.2.

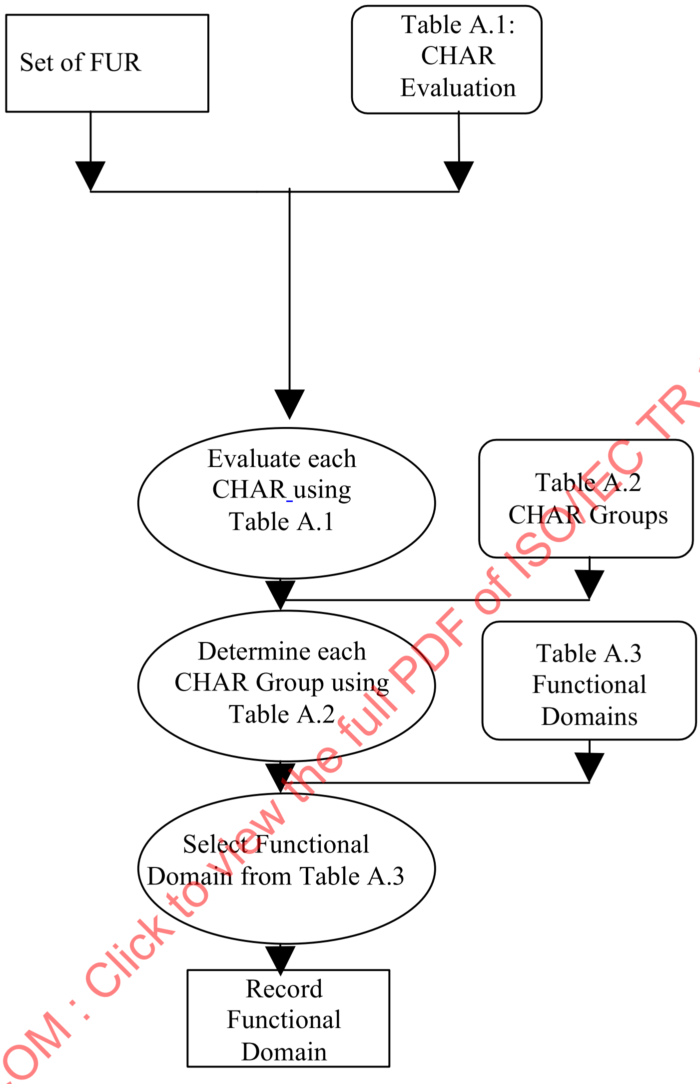

Figure A.1 — Functional Domain evaluation process

A.4 Examples of evaluating FUR to determine the Functional Domain

A.4.1 Example 1: word processing FUR

A.4.1.1 Word processing FUR

The following logical requirements constitute the FUR for this application:

Inputs:

a) Accept a numeric value between 1 and 132 called MAXPOS.

b) Accept a character input stream called INSTREAM.

c) Preserve the order and length of input words in the output.

d) Reduce each internal input break to a single character output break.

e) Start each output line with a word.

f) Fill each output line with as many complete words as possible.

g) Build output lines containing no more than MAXPOS characters preceding the new-line.

h) Break output lines only where the input breaks.

i) Save input for later reference and use.

Outputs:

j) Return a completion code corresponding to normal input.

k) Return a completion code corresponding to invalid MAXPOS.

l) Return a completion code corresponding to empty input.

m) Return a completion code corresponding to break only input.

n) Return a completion code corresponding to oversize word.

o) Return a line-by-line output form of the input character stream.

A.4.1.2 Word processing FUR Functional Domain evaluation result

Table A.4 — Word processing FUR evaluation

<table>
<tr><th>CHAR</th><th>Rating
(see Table A.1)</th><th>CHAR Group</th><th>Rating
(see Table A.2)</th><th>Functional Domain
(see Table A.3)</th></tr>
<tr><td>Response</td><td>Negligible</td><td rowspan="3">Control- and Communication-Rich</td><td rowspan="3">negligible</td><td rowspan="8">Data Processing System</td></tr>
<tr><td>Interfaces</td><td>Negligible</td></tr>
<tr><td>System Management</td><td>Negligible</td></tr>
<tr><td>Complex Data</td><td>Negligible</td><td rowspan="2">Data-Rich</td><td rowspan="2">present</td></tr>
<tr><td>Persistence</td><td>Present</td></tr>
<tr><td>Manipulation</td><td>Present</td><td rowspan="3">Manipulation- and Algorithm-Rich</td><td rowspan="3">present</td></tr>
<tr><td>Scientific/Engineering</td><td>Negligible</td></tr>
<tr><td>Adaptive</td><td>Negligible</td></tr>
</table>

A.4.2 Example 2: human resources FUR

A.4.2.1 Human resources FUR

The following logical requirements constitute the FUR for this application:

a) Maintain and store information about employees and their dependants.

b) Maintain and store information about available jobs.

c) Maintain and store information about job assignments.

d) Provide a means to access the information at any point in time.

e) Pass all modifications to payroll status to payroll daily.

A.4.2.2 Human resources FUR Functional Domain evaluation result

Table A.5 — Human resources FUR evaluation

CHAR	Rating (see Table A.1)	CHAR Group	Rating (see Table A.2)	Functional Domain (see Table A.3)
Response	Negligible	Control- and Communication-Rich	negligible	**Information System**
Interfaces	Negligible			
System Management	Negligible			
Complex Data	Present	Data-Rich	Dominant	
Persistence	Present			
Manipulation	Present	Manipulation- and Algorithm-Rich	present	
Scientific/Engineering	Negligible			
Adaptive	Negligible			

A.4.3 Example 3: traffic light FUR

A.4.3.1 Traffic light FUR

The following logical requirements constitute the FUR for this application:

a) Traffic light will change based on number of cars waiting, light location and preset timing.

b) Maintain and store timing file for day periods, days and holidays.

c) Signals from emergency vehicles interrupt regular program logic.

d) Train signal prevents traffic lights from turning green via switches.

e) Send traffic statistics on weekly basis to the Department of Transportation (DOT).

A.4.3.2 Traffic light FUR Functional Domain evaluation result

Table A.6 — Traffic light FUR evaluation

CHAR	Rating (see Table A.1)	CHAR Group	Rating (see Table A.2)	Functional Domain (see Table A.3)
Response	Present	Control- and Communication-Rich	Dominant	**Complex Data Driven Control System**
Interfaces	Present			
System Management	Present			
Complex Data	Present	Data-Rich	present	
Persistence	Present			
Manipulation	Negligible	Manipulation- and Algorithm-Rich	present	
Scientific/Engineering	Present			
Adaptive	Present			

A.4.4 Example 4: automated teller banking FUR

A.4.4.1 Automated teller banking FUR

The following logical requirements constitute the FUR for this application:

Maintained Entities:

a) Customer

b) Savings Account

c) Cheque

d) Transaction (images)

Interfaces:

e) Software accesses tables in terminal controller to obtain date, branch-id, and status (on or off line), but there is no user access to this

f) No other files accessed

Transactions/ Functions:

g) Add Customer

h) Change Customer

i) Remove Customer

j) Customer Inquiry

k) Open Account

l) Close Account

m) Cheque Deposit

n) Cheque withdrawal

o) Savings deposit

p) Savings Withdrawal

q) Transfer between two account numbers

r) Cheque inquiry

s) Savings Inquiry

t) Print statement

u) Passbook update

v) Functions j) – s) must complete within 10 seconds, or a timeout will occur.

A.4.4.2 Automated teller banking FUR Functional Domain evaluation result

Table A.7 — Automated teller banking FUR evaluation

CHAR	Rating (see Table A.1)	CHAR Group	Rating (see Table A.2)	Functional Domain (see Table A.3)
Response	Present	Control- and Communication-Rich	Present	**Complex Controlling Information System**
Interfaces	Negligible			
System Management	Negligible			
Complex Data	Present	Data-Rich	Dominant	
Persistence	Present			
Manipulation	Present	Manipulation- and Algorithm-Rich	Present	
Scientific/Engineering	Negligible			
Adaptive	Negligible			

A.5 Mapping some common software ‘types’ to the CHAR Group model

As a test of the model, the characteristics of some terms for commonly understood ‘types’ of software are mapped against the CHAR Group model. The list of software types given below in Table A.8 includes terms occurring in or which are synonyms of terms in ISO/IEC 12182 ‘Information Technology – Categorization of Software’. Not all of the terms in the latter were used as most of the categories of ISO/IEC 12182 are irrelevant to the purpose of this standard.

NOTE The characteristics chosen for these ‘types’ of software may not necessarily be present in all specific instances of the software ‘type’.

Table A.8 — Analysis of some commonly understood software 'types'

ISO/IEC 12182 software type	Functional Domain	Control- / Communication Rich	Data-Rich	Manipulation Rich
Management Information System (Business transaction processing), Decision Support	Information System	Negligible	Dominant	Present
Word Processing, Geographic Information System	Data Processing System	Negligible	Present	Present
Automated Teller Banking	Controlling Data System	Present	Present	Negligible
Business (Business Enterprise)	Complex Controlling Information System	Present	Dominant	Present
Military Command and Control	Non-Specific (Complex) System	Present	Present	Present
Real Time: Embedded, Device (Printer, Disc, etc.) Driver	Simple Control System	Dominant	Negligible	Negligible
Real Time: Embedded, Avionics, Message router	Complex Control System	Dominant	Negligible	Present
E-mail, Emergency dispatch call/receipt, Operating System	Data Driven Control System	Dominant	Present	Negligible
Process Control (Control System)	Complex Data Driven Control System	Dominant	Present	Present
Scientific, Standard math/Trig. Algorithms	Pure Calculation System	Negligible	Negligible	Dominant
Engineering	Controlling Calculation System	Present	Negligible	Dominant
Self-learning (Expert or Artificial Intelligence), Statistical, Spreadsheet, Secure Systems, Actuarial	Scientific Information System	Negligible	Present	Dominant
Safety Critical	Scientific Controlling Data Processing System	Present	Present	Dominant
(no corresponding type)	Pure Data Handling System	Negligible	Dominant	Negligible
(no corresponding type)	Controlling Information System	Present	Dominant	Negligible
(no corresponding type)	Control System	Present	Negligible	Present

NOTE This analysis shows that many common types of software are NOT distinguished by FUR, but by other types of requirements, notably quality constraints.

A.6 Mapping of existing candidate FSM Methods to the CHAR Group model

As a test of the CHAR Group Model, four existing Candidate FSM Methods, the IFPUG Function Point Analysis method Release 4.1 unadjusted, the NESMA Definitions and counting guidelines for the application of function point analysis, the MkII FP method and the COSMIC FFP method are mapped to the BFC Types in Table A.9 below.

NOTE These four methods have been included because they are the methods currently being standardized via the PAS process or as a new work item within SC7.

Table A.9 — Mapping of candidate FSM Methods to the CHAR Group model

CHAR	IFPUG 4.1 unadj.	NESMA unadj.	Mk II	COSMIC FFP
Response	Via elementary processes across boundary	Via elementary processes across boundary	Via Logical Transactions	Via Entry, Exit BFC's
Interfaces	Via elementary processes across boundary	Via elementary processes across boundary	Via Logical Transactions	Via Entry, Exit BFC's
System Management	Via elementary processes across boundary	Via elementary processes across boundary	Via Logical Transactions	Via Entry, Exit BFC's
Complex Data	Not addressed	Not addressed	Not addressed	Not addressed
Persistence	Via ILF, EIF BFC's	Via ILF, EIF BFC's	Via entity reference in Logical Transaction	Via Read, Write BFC's
Manipulation	Considered as part of processing logic for EI, EO, EQ	Considered as part of processing logic for EI, EO, EQ	Not addressed	Not addressed
Scientific/Engineering	Not addressed	Not addressed	Not addressed	Not addressed
Adaptive	Not addressed	Not addressed	Not addressed	Not addressed

This analysis suggests that these four methods specifically address certain Functional Domains of the CHAR Group model more explicitly than others. Some of the CHARs are addressed within an existing BFC type of a method, some could not be distinguished and others are not addressed by any of the methods.

Annex B
(informative)

BFC type method to determine Functional Domains

B.1 Purpose

The purpose of this Informative Annex is to:

a) Define an FDC method,

b) Use this method to generate example definitions of various specific Functional Domains, and

c) Test the method against examples of FUR and some existing candidate FSM Methods.

ISO/IEC 14143-1:1998 requires that:

a) FSM Methods measure Functional Size by identifying the 'Base Functional Component Types (or 'BFC Types') that they recognize within the FUR and assigning values that are used to calculate the Functional Size,

b) A BFC Type is a defined category of 'elementary units of FUR defined by and used by an FSM Method for measurement purposes', and

c) FSM Methods shall describe the Functional Domains to which they can be applied.

This Informative Annex:

a) can be used to describe or to define various practical Functional Domains from FUR, and

b) can be used by the designers of FSM Methods to state which characteristics are recognized by the BFC types of their FSM Method.

The set of BFC Types (the 'BFC Type Model' or 'the Model') defined in this Informative Annex provides a 'meta-model' of FUR of software. The Model is founded on the components of established 'structured' information system and data analysis methods such as those of Chen, Codd, Constantine & Myers, DeMarco, Jackson, Warnier and Yourdon, but extended pragmatically based on additional observations of actual software components. A vital selection criterion when defining the set of BFC Types has been that the level of granularity, or abstraction of the BFC Types is such that they can be recognized and defined unambiguously.

B.2 The model of BFC Types

B.2.1 The model

The Model of BFC Types of this Informative Annex relies on an understanding of how software is normally structured into 'layers' and 'segments' within layers, and hence on the definition of the 'boundary' of a software segment. A full explanation of these terms is given in Clause B.8.

FUR for software in any one segment is usually built up from BFC Types belonging to only one defined Functional Domain. However, the FUR for software in neighboring segments of the same layer or in neighboring layers may or may not belong to the same defined Functional Domain as the segment of interest. Consequently, when analyzing a particular set of FUR to decide to which Functional Domain(s) they belong, care should be taken first to distinguish any layers and any segments within layers of the FUR. Similarly, when deciding if a particular FSM Method can be used to measure the size of a particular set of FUR, first decide if the FUR require software in various layers and/or segments from different Functional Domains. Only then can one test if the FSM Method can be applied to all of the FUR.

The method for the determination of functional domain views FUR as follows:

a) FUR consist of a set of transactions.

b) A transaction takes data as input, processes them and outputs data as a result of the processing.

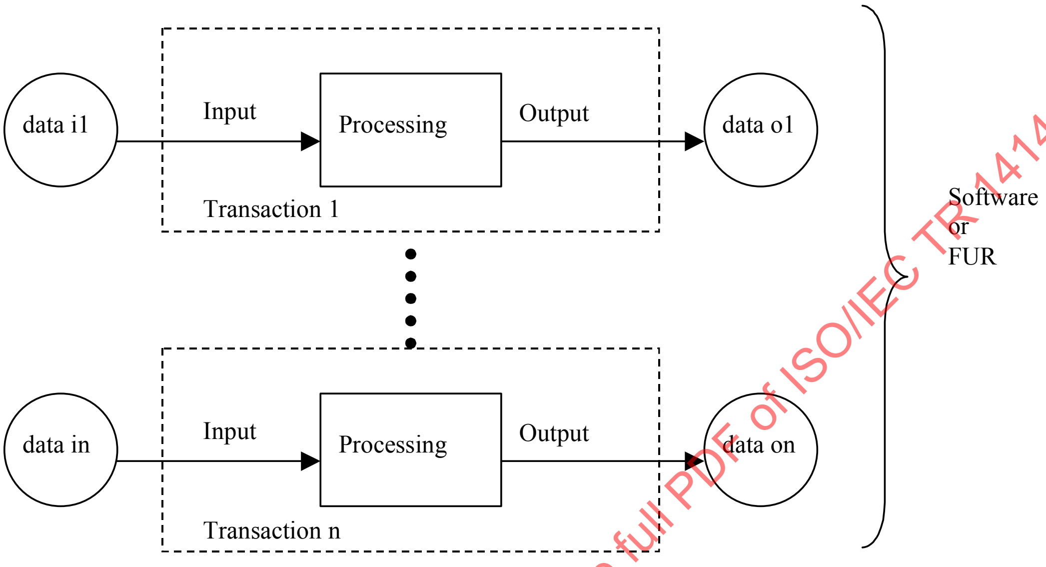

Figure B.1 — Software model

B.2.2 Decomposing FUR into BFC Types

B.2.2.1 Transaction classes

FUR state *what* the software is supposed to do once it is ‘launched’. Major subsets of FUR can be functionally decomposed into ‘Transactions’, which are sequences of input, processing and output, not necessarily in that order, triggered by events outside the software. (In the world of real-time systems, a transaction might be defined as the software’s required response to an ‘external event’.) Examples include:

— for a business application, a new customer is signed up in the real physical world, this event triggers a process for a human user to register data about the customer in the software

— a human user decides (the event) to make an enquiry to retrieve information about the new customer

— for a process control application, a hardware clock reaches a certain time (the event), which triggers the application to start a round of polling a range of sensors to gather data, compare the received data against norms, and send signals to apply corrections to machine-settings

— as for the previous process control example, but in addition the software is required to be interruptible, e.g. if a sensor detects an out-of-norm condition (the event) which requires immediate attention, the normal operation of the software is interrupted, and the software function changes to deal with the out-of-norm condition

— a message router receives an incoming message (the event), goes through a process to decide where and how to forward the message and sends the message on its way

In all cases the transaction is complete when the software is left in a self-consistent state with respect to the external event, that is, it has completed everything it was required to do as a consequence of the occurrence of the external event.

The Model distinguishes three classes of transactions (three BFC Types) depending on how the software is required to interact with the external world:

— **Type 1.** ('Passive') Once started, the normal state of the software is to do nothing. When an input is received across the software boundary from the user, the data is processed, and the processing may or may not result in data passing back across the boundary. When the transaction is complete, the software reverts to its 'do nothing' state.

— **Type 2.** ('Pro-active') Once started, the normal state is that the software must periodically sense what is happening at the other side of its boundary. Based on the conditions it detects, and its processing rules, the software may or may not pass data back across its boundary.

— **Type 3.** ('Interruptible') Once started, the software operates in Type 1 or Type 2 mode but in addition, when notified of certain types of event, it must stop whatever it is doing and accept and react immediately to the event. Depending on the processing rules associated with the event, the software may or may not pass data back across the boundary.

The above explanation addresses only the functions that the software may be *required* to perform. Each of the functions can be implemented in different ways, and the implementation may be dictated by performance or quality requirements. Examples of how each may be specified for each of the three types are, respectively:

a) "The software is required to accept an input from the keyboard, and display the symbol defined for the key on the monitor."

b) "The software is required to read the temperature at one second intervals and to generate a signal that indicates if the temperature of the coolant is outside the range of 20 to 30 degrees."

c) "The software is required to read the temperature at one second intervals and to generate a 'pump-on' signal if the temperature is above 30 degrees and a 'pump-off' signal if the temperature is below 20 degrees. At any time, on receipt of an overheat notification, the software is to unconditionally issue a 'pump-on' signal."

B.2.2.2 Data types recognized

Transactions involve inputting, outputting and processing of items or groups of data. The Model distinguishes various types of data items or groups i.e. BFC Types), which the software is required to recognize. Different Functional Domains may recognize only certain Data Types.

The Model recognizes the following Data Types:

— Data Element Types (DET) Synonym 'Data Attribute'

— Data Groups (a collection of Data Attributes of a single Entity-type)

— Sounds

— Images

— Moving images

— Data Relationships (a rule describing the meaning and degree of the relationship between two other data types)

NOTE The concept of a 'Data Element Type', typically defined in existing FSM Methods as 'a unique user-recognizable unit of data', could include any of the above Data Types. Future FSM Methods relying on this Model for defining BFC Types and Functional Domains may therefore need to specify more precisely which Data Types they recognize, if appropriate.)

B.2.2.3 Information creation function types

When Transactions 'process' data, that is they manipulate and compute new data, they use many types of processes, which we call 'Information Creation Function Types'. These BFC Types, which range in complexity from very simple Boolean operations to complex mathematical algorithms, vary with Functional Domain.

The Model recognizes the following Information Creation Function Types:

— Simple Boolean operations

— Simple arithmetic operations

— Standard mathematical/statistical algorithms (for example as in industry-standard spreadsheets)

— Two and three-dimensional geometry (for example as in geographic information systems)

— Complex scientific/engineering processes (for example as in aircraft wing design, or weather forecasting)

— Sets of rules for decision-making by inference, as in expert systems

— Heuristic, or self-modifying algorithms

— Business Domain-specific algorithms (i.e. to be defined in the local Business)

B.2.2.4 Data retention requirements

The Data Types of B.2.2.2 may be required to be retained for different time-periods, a feature that is Functional Domain distinguishing.

The Model distinguishes three timescales (or 'degrees of persistence') for which FUR specify retention of Data Types:

— *no retention*; Data Types are not required to be retained after the logical Transaction is complete; an example would be the data used by a simple calculator

— Data Types are required to be retained only for the duration of *processing* of the software; an example would be the accumulation of data across a batch of Transactions for output on a common report when all Transactions have been processed

— Data Types are required to be retained long-term about events outside the software - *indefinitely*, as far as the software is concerned

B.2.3 Summary of the BFC Type model

The BFC Type Model of this Informative Annex recognizes three groups of BFC Types, namely:

— three Classes of Transactions

— various Data Types

— various types of 'Information Creation Functions'

and that the various Data Types can be required to have any of three degrees of persistence for data retention.

Any Transaction is required to recognize only certain Data Types, process the Data Types according to certain types of Information Creation Functions, and to retain Data Types for certain periods.

The proposition of the Model of this Informative Annex is that all FUR can be expressed in terms of these BFC Types and the requirements for data retention, and that the Functional Domains, which need to be distinguished for practical FSM Methods, can be defined in terms of these same BFC Types. This list of BFC Types, however, is not claimed to be exhaustive and does not exclude the future provision of other BFC Types.

The Model of BFC Types is summarized in Table B.1.

Table B.1 — The model of BFC Types - a framework for defining functional domains

Software BFC Types			4. Requirement for Retention of Data Types
1. Transaction Class	**2. Data Types**	**3. Information Creation Function Types**	
— Passive — Pro-active — Interruptible	— DET. — Data groups. — Sounds. — Images. — Moving Images. — Data Relationships.	— Simple Boolean. — Simple Arithmetic. — 2 & 3-d geometry. — Std. math's/stats. — Complex Scientific / Engineering algorithms. — “Expert System” Rules. — Heuristic algorithms. — Business Domain-specific algorithms.	— None (Only duration of Transaction). — Duration of Processing. — Indefinitely.

B.3 Process for generation of Functional Domains

A unique Functional Domain can be characterized and then defined from the contents of Table B.1 by taking a combination of zero, one or more relevant BFC Types from each of columns 1, 2 and 3 and data retention requirements from column 4.

This process could lead to the possibility of generating a huge number of Functional Domain definitions, which is undesirable. In practice FSM Method designers and users need a small number of commonly encountered Functional Domain definitions. Table B.2 defines some example Functional Domains for such a list, which may be extended or refined with further study.

Some consequences of this analysis, using the definitions of Table B.2 are that:

— a Multi-User Operating System is ‘Complex Process Control Software’

— a PC Word Processor system is a special case of a ‘Business Data Processing Application’, but which also includes graphical elements as BFC Types

— a washing machine’s embedded control software and a traffic light control system software are examples of ‘Simple Process Control Software’

— an aircraft attitude control system is ‘Complex Process Control Software’, (it has to store some data indefinitely in the aircraft’s ‘black box’) but with the addition of domain-specific algorithms, i.e. flight envelope control algorithms

— the only real difference between ‘Scientific/Engineering Computation Software’ and ‘Business Data Processing Applications’ is the existence of Complex Scientific/Engineering algorithms in the former.