INTERNATIONAL STANDARD

ISO 13374-2

First edition
2007-07-15

Condition monitoring and diagnostics of machines — Data processing, communication and presentation —

Part 2: Data processing

Surveillance et diagnostic d'état des machines — Traitement, échange et présentation des données —

Partie 2: Traitement des données

Reference number
ISO 13374-2:2007(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) 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.

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 13374-2 was prepared by Technical Committee ISO/TC 108, *Mechanical vibration, shock and condition monitoring*, Subcommittee SC 5, *Condition monitoring and diagnostics of machines*.

ISO 13374 consists of the following parts, under the general title *Condition monitoring and diagnostics of machines — Data processing, communication and presentation*:

— *Part 1: General guidelines*

— *Part 2: Data processing*

The following part is envisaged:

— *Part 3: Communication requirements*

Introduction

The various computer software systems written for condition monitoring and diagnostics (CM&D) of machines that are currently in use cannot easily exchange data or operate in a plug-and-play fashion without an extensive integration effort. This makes it difficult to integrate systems and provide a unified view of the condition of machinery to users. The intent of Parts 1 to 3 of ISO 13374 is to provide the basic requirements for open CM&D software architectures which will allow CM&D information to be processed, communicated, and displayed by various software packages without platform-specific or hardware-specific protocols.

Condition monitoring and diagnostics of machines — Data processing, communication and presentation —

Part 2: Data processing

1 Scope

This part of ISO 13374 details the requirements for a reference information model and a reference processing model to which an open condition monitoring and diagnostics (CM&D) architecture needs to conform. Software design professionals require both an information model and a processing model to adequately describe all data processing requirements. This part of ISO 13374 facilitates the interoperability of CM&D systems.

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 13374-1:2003, *Condition monitoring and diagnostics of machines — Data processing, communication and presentation — Part 1: General guidelines*

ISO/IEC 14750:1999, *Information technology — Open Distributed Processing — Interface Definition Language*

3 CM&D information architecture requirements

3.1 Overview

An information architecture describes all the data objects and their properties (or attributes), property data types, data object relationships, reference data and data documents for a given system or application. An open CM&D information architecture specification shall describe their content for each of the five layers shown in Figure 1.

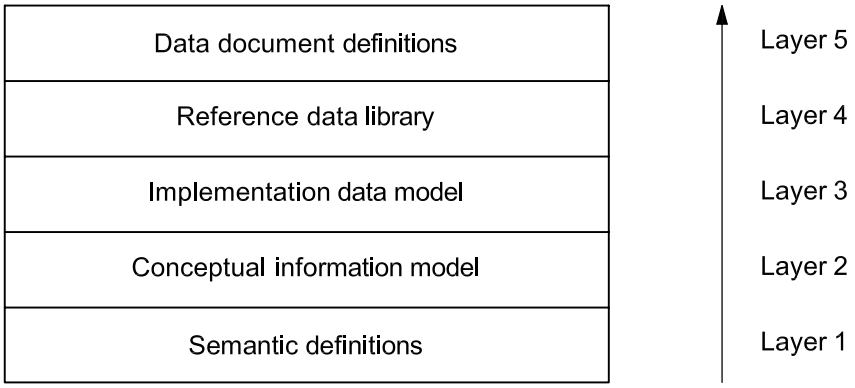

Figure 1 — CM&D information architecture layers

3.2 Semantic definition requirements

To ease understanding among various parties utilizing the information architecture, an open CM&D information architecture specification shall provide a set of semantic definitions for each major data object in the system. Non-formal description terminology, such as English language definitions of the data objects, may be used. Formal descriptions utilizing ontological schemas, such as the proposed Resource Description Format (RDF) of the World Wide Web Consortium (W3C), may also be utilized.

3.3 Conceptual information model requirements

A conceptual information model is an integrated definition of all the primary data objects relevant to CM&D, along with their major properties (also called “attributes”), property data types, object interrelationships, and object or relationship constraints. An open CM&D information architecture specification shall provide a non-proprietary conceptual information model, sometimes referred to as a “schema”, which is independent of how the data are physically stored or accessed. This conceptual information schema is a blueprint of the location of various data elements, which facilitates system integration and data integrity. Using a conceptual schema, an implementation data model can be verified.

The Unified Modelling Language (UML) has emerged as the software industry's dominant modelling language and is now an International Standard, ISO/IEC 19501. UML includes a standardized class diagram representation for information modelling (see Annex B for additional information about UML).

3.4 Implementation data model requirements

Based on the conceptual information model, an implementation data model provides the exact representation of data elements that shall be transferred or stored. An open CM&D information architecture specification shall provide an open implementation data model that conforms to its conceptual data model.

The open CM&D implementation data model shall allow for the integration of many sources of machinery information, support peer-to-peer databases, allow user-defined look-up entries, and utilize standardized timestamps and engineering units. The schema should support unique enterprise, site, and database or data source identifiers to differentiate data taken at different physical locations. The schema should also support unique, system-wide identifiers for plant segments containing machinery (service segment locations) in a parent-child hierarchy. Also, the schema should support a unique asset-specific identifier to allow individual component monitoring and tracking in a parts hierarchy.

The basic framework of storing enterprise, site, site database, process or machine segment information (such as physical orientation, drive type, mounting type and shaft coupling type), asset nameplate data (including entries such as rated speed, rated power, make and model, and bearing or other component information), measurement locations, data measurement sources, transducers, transducer orientation, engineering units, signal processing post-scaling types, ordered lists, and alarms should be specified in the schema. At a data level, the schema should support formats for communicating historical single-valued numeric data,

Fast-Fourier Transform (FFT) spectra data, constant percentage band (CPB) spectra, time waveforms, sample-based test data, thermographic images and binary large objects. The schema should support a date/time notation that references back to a specific instance in time, using the Gregorian (common era, or CE) calendar, with a lexical representation based upon ISO 8601[1] referenced to universal coordinated time (UTC) and that also stores the local time zone offset.

This specification can be specified using various schema definitions. The file description schema format has been used for years in the scientific programming community. It maps the format for ASCII or binary data files, which can be exported from a computer system or imported into a computer system. A complete record format description is published which specifies the data fields contained in the file, their exact location in relation to the other data fields, whether the fields are in ASCII or binary format, and the exact data format – real floating point, integer, character, varying character string – of each field.

The relational information schema format is the definition language for relational database management systems. The relational method is analogous to a blue-print drawing which defines the various “room names” or (tables) where data will be stored, the data “contents” or (columns) in the rooms, each data column's exact data format (scientific floating point, integer, varying character string, etc.), whether or not a data column can be empty or not (not null) and each data row's unique “key” (primary key) which uniquely identifies it. A table can be related to another table by including a “reference” (foreign key) to it.

An Extensible Markup Language (XML) schema definition (XSD) is a recommended definition language. XML is a project of the World Wide Web Consortium (W3C), and the development of the specification is being supervised by their XML Working Group. XML is a public format written in the Standard Generalized Markup Language (SGML), using ISO 8879, for defining descriptions of the structures of different types of electronic documents. The version 1.0 specification was accepted by the W3C as a recommendation on 10 February 1998. On 3 May 2001, the W3C issued an XML schema as a W3C recommendation. XML schemas define shared markup vocabularies, the structure of XML documents which use those vocabularies, and provide hooks to associate semantics with them. By bringing datatypes to XML, XML schemas increase the utility of XML to the developers of data interchange systems. XML schemas allow the author to determine which parts of a document may be validated, or identify parts of a document where a schema may apply. Further, as XML schemas are XML documents themselves, they may be managed by XML authoring tools (see Annex B for additional information about XML).

Regardless of which information schema format is chosen, the implementation data model shall define a minimum set of data elements that should be included in the schema for conformance. In addition, a list of optional elements may be included.

3.5 Reference data library requirements

To effectively utilize an implementation data model, standard entries for various look-ups need to be stored in a reference data library. An open CM&D information architecture specification shall provide an open reference data library which conforms to its implementation data model. The specification should support populating and maintaining industry-standard taxonomies and codes for the reference data and allow both suppliers and end-users to add industry-specific and customer-specific entries to the library using database-unique entries. The specification shall also support the creation of a standard set of reference codes in various languages as required.

The reference data library specifies all code tables for segment (machine/component) type codes, asset type codes, measurement location type codes, engineering unit type codes, sampling test codes, diagnostic/prognostic event codes, health codes, failure codes and root cause codes. The library also houses engineering unit codes, measurement location type codes and mounting orientation codes.

3.6 Data document definition requirements

An open CM&D information architecture specification shall also specify the format for the publication of a data document. The specification allows the reference data library to be read or written in a standardized way and supports applications which need data import/export capability.

Specifications which utilize the file description schema format as their implementation data model will probably utilize the same specification for the publication of an ASCII or binary data document. The complete record format description shall be published, specifying the data fields contained in the file, their exact location in relation to the other data fields, whether the fields are in ASCII or binary format, and the exact data format – real floating point, integer, character, varying character string – of each field.

Specifications which utilize the XML schema format as their implementation data model will probably utilize the same format for the publication of an XML data document. An XML schema provides the definition of the XML document and XML parsers and validation tools can verify the syntax of the document's content.

3.7 Compliant specifications

An open CM&D information architecture specification shall utilize the information architecture as described in subclauses 3.1 to 3.6. MIMOSA, a non-profit association, publishes an open CM&D information specification which is compliant with the above requirements. The specification is known as the MIMOSA Open Systems Architecture for Enterprise Application Integration (OSA-EAI™) specification. Annex A describes this specification in more detail.

4 CM&D processing architecture requirements

4.1 Overview

A processing architecture describes all the interactions or transactions which are between modules internal to the software system itself, external from end-user interactions, or external from other software system interactions. An open CM&D processing architecture specification shall utilize the processing architecture shown in Figure 2. This architecture is defined as blocks of data processing functionality. After each block in the system has been properly configured, the basic data are converted into digital form in Data Acquisition (DA) and are processed in various ways as they are transformed into actionable information, resulting in Advisory Generation (AG). As the processing progresses from DA to AG, data from preceding blocks need to be transferred to subsequent blocks and additional information acquired from or sent to external systems. Similarly, as the data evolve into information, both standard technical displays and graphical presentation formats are required. In many applications, data archiving is required in order to maintain a history of the output of each block. The DA, DM and SD blocks are responsible for assessing data quality. Output should be identified as good, bad or undetermined.

This part of ISO 13374 does not address the impact of errors and their propagation within and across the various CM&D processing blocks. Sources of such errors include instrumentation calibration, environmental noise, signal conditioning and processing, computational rounding, human-induced inputs, and their combined effects.

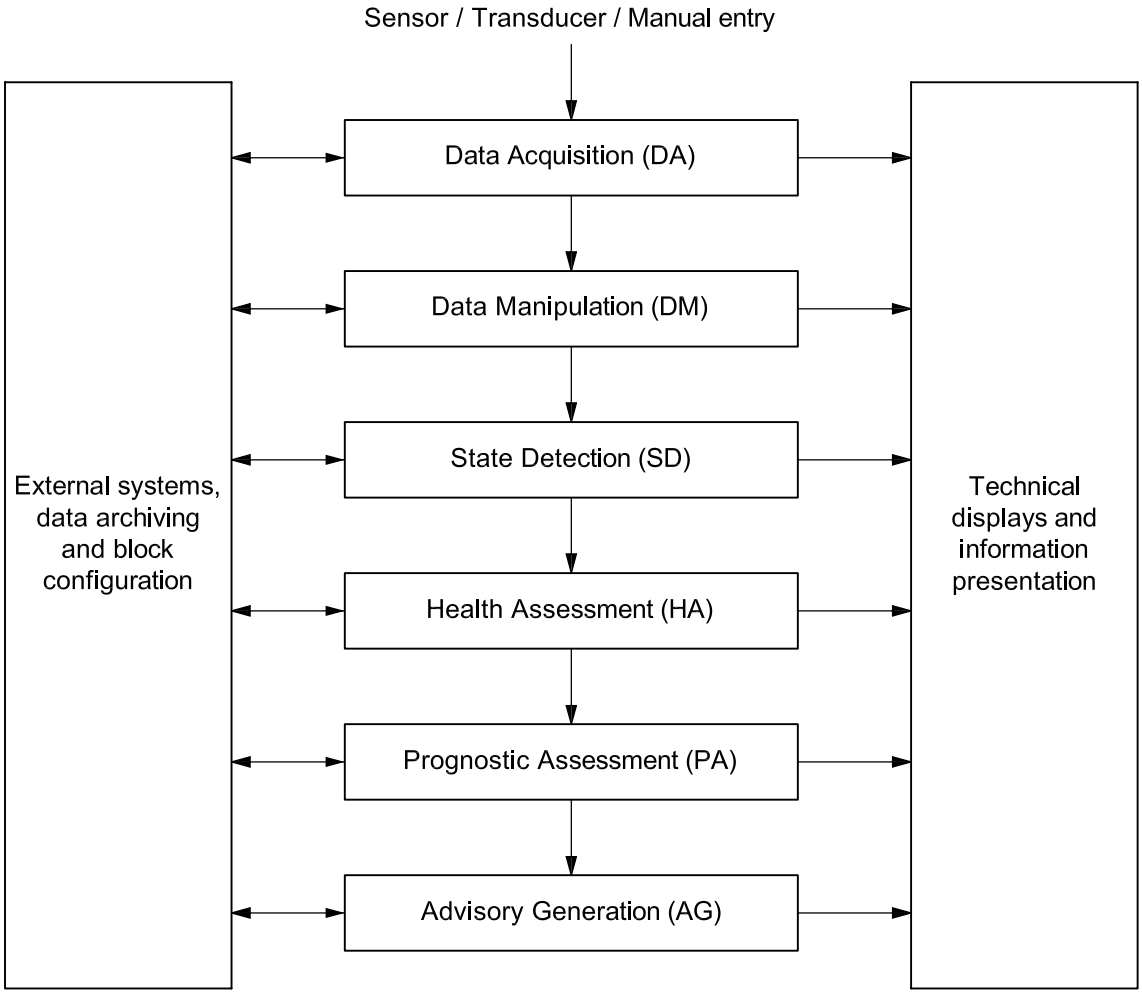

Figure 2 — Data processing block diagram

4.2 Data Acquisition (DA) blocks

As detailed in Figure 3, the DA block has been generalized to represent the software module that provides system access to digitized data entered automatically or manually. The DA module may represent a specialized data acquisition module that has analog feeds from legacy sensors, or it may collect and consolidate sensor signals from a data bus. Alternatively, it might represent the software interface to a smart sensor. The DA module is basically a server of calibrated digitized sensor data records.

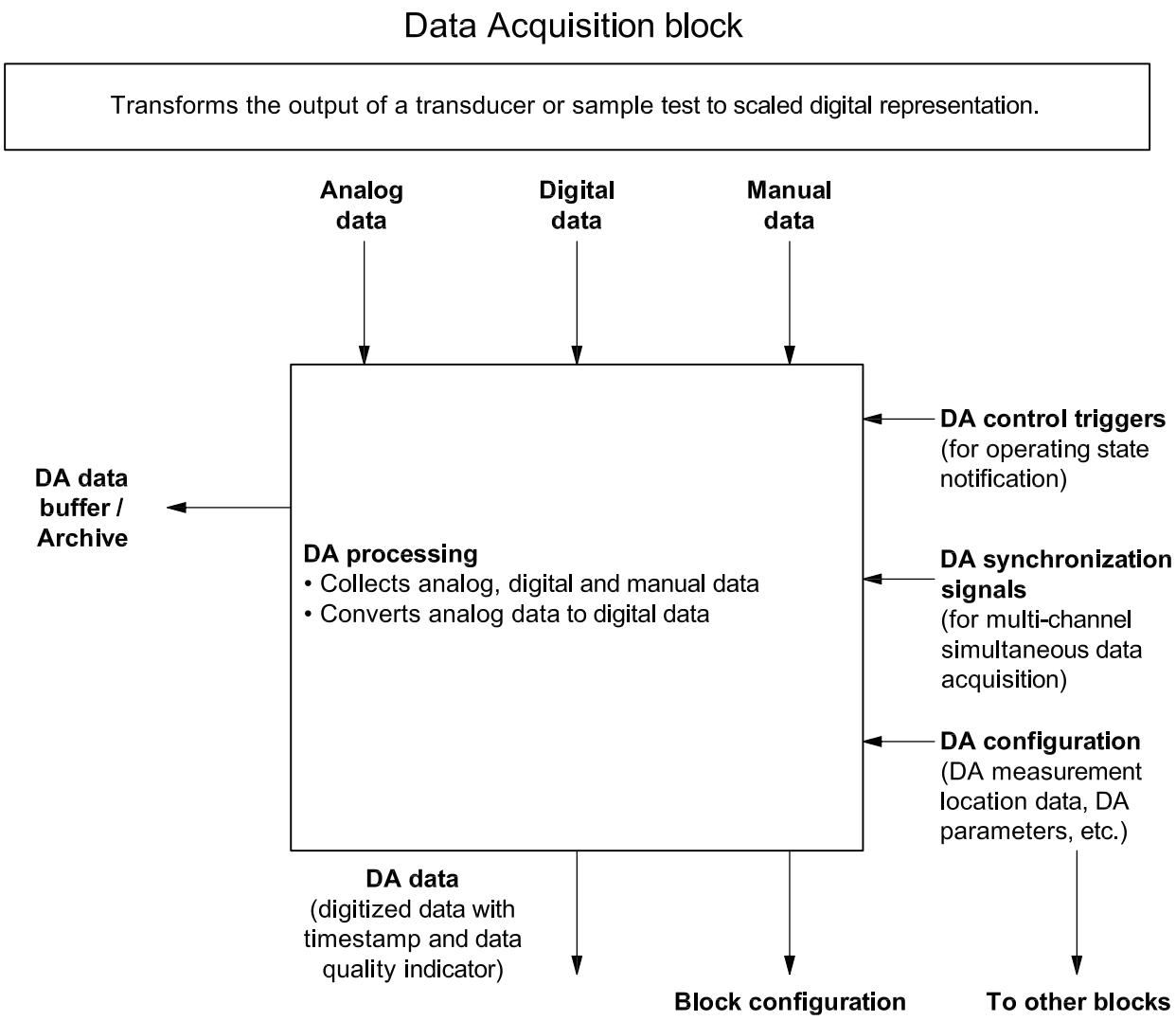

Figure 3 — Data Acquisition block

The output of all DA blocks shall contain the following:

— digitized data;

— time-order/time-reference data, normally referenced with UTC and local time zone;

— data quality indicator (e.g. “bad”, “good”, “unknown”, “under review”, etc.).

Examples of digitized data include:

— floating point values for scalar data;

— magnitude and time series for dynamic data;

— thermal radiation data with digitized image for thermographic data;

— sample test results for lubricating fluid/air/water sample data.

4.3 Data Manipulation (DM) blocks

As detailed in Figure 4, the DM block processes the digital data from the DA block to convert it to a desired form which characterizes specific descriptors (features) of interest in the machine condition monitoring and diagnostic process. Often the functionality within this layer consists of some signal processing algorithms.

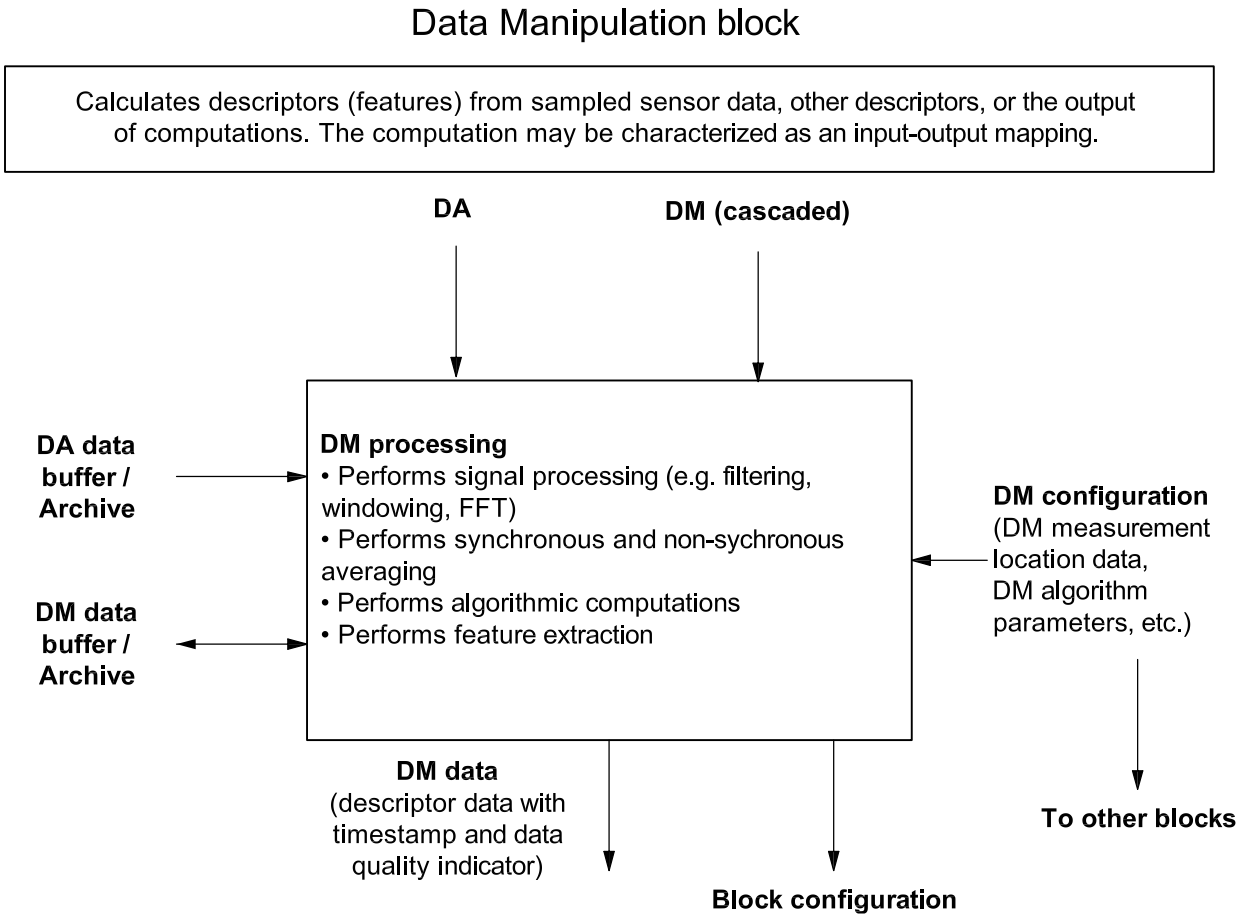

Figure 4 — Data Manipulation block

This block may contain speciality processing functions such as Fast Fourier Transforms, wavelets or simple average values over a time interval.

Examples of the descriptor outputs of the DM block include:

— extracted feature;

— conversion from time domain to frequency domain and vice versa;

— calculated, non-interpretative values;

— virtual sensor (differential pressure from inlet and outlet pressures);

— integrating acceleration to velocity/double integration to displacement;

— filtering;

— normalization;

— time series including sample rate.

4.4 State Detection (SD) blocks

As shown in Figure 5, the primary function of the SD block (sometimes referred to as “state awareness”) is to compare DM and/or DA outputs against expected baseline profile values or operational limits, in order to generate enumerated state indicators with respective boundary exceedances. The SD block generates indicators which may be utilized by the Health Assessment block to generate alerts and alarms. When appropriate data are available, the SD block should generate assessments based on operational context, sensitive to the current operational state or operational environment.

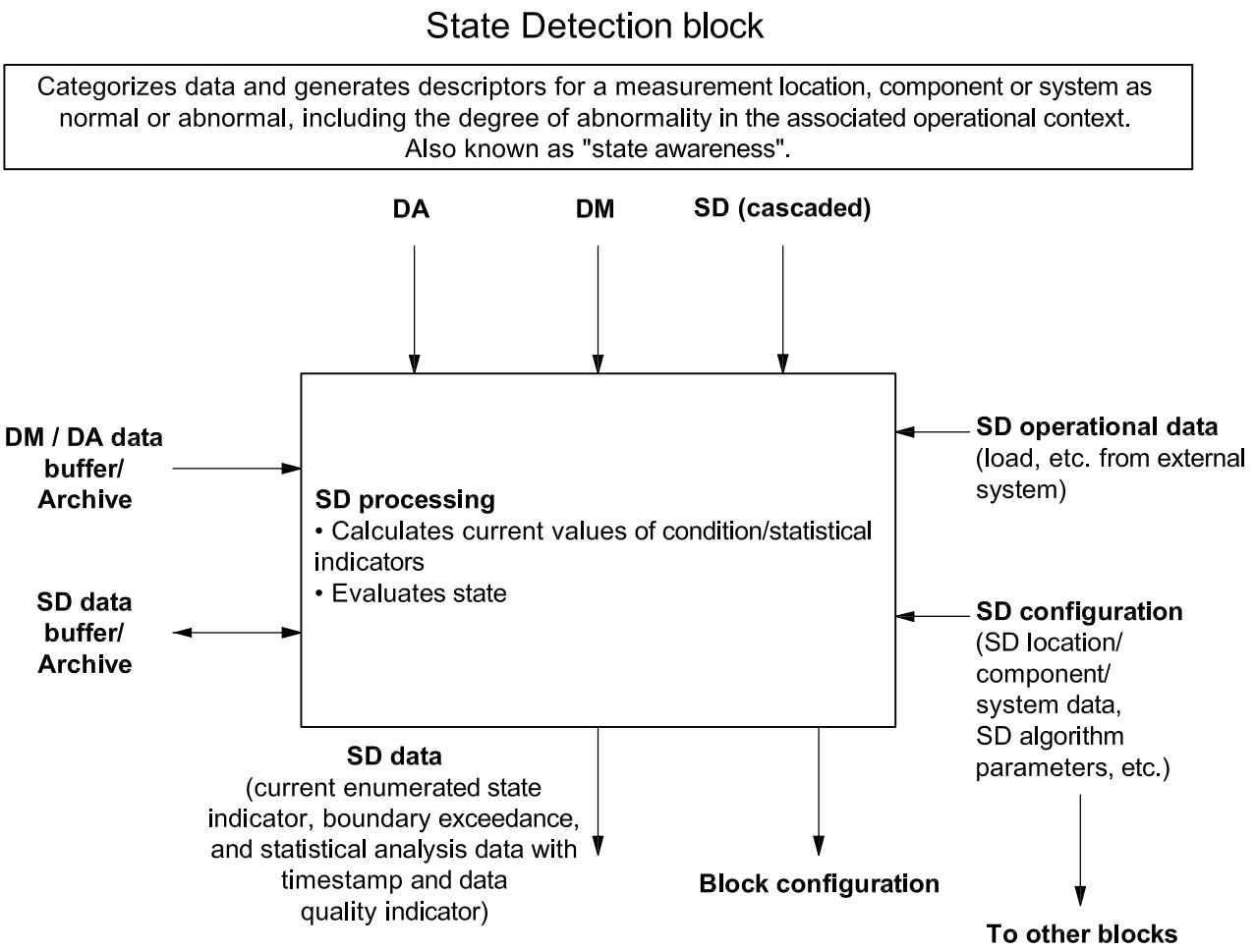

Figure 5 — State Detection block

Typically, this block of processing provides data which will contribute to a diagnosis in the health assessment block. The SD block may make use of current and historical DA and DM outputs to evaluate the current state. It may provide data manipulation and sensor module control signals, such as acquisition scheduling commands, data triggers and processing instructions.

Examples of outputs of the SD block include:

— enumerated state indicator;

— threshold boundary alerts;

— severity of threshold boundary deviation above/below;

— rate of change alert;

— degree of abnormality;

— statistical analysis using parametric and non-parametric approaches, e.g. Weibull and Gaussian distribution.

4.5 Health Assessment (HA) blocks

As shown in Figure 6, the HA block is an information block which utilizes expertise from a human or automated agent to determine the current health of the equipment and to diagnose existing fault conditions. It determines the state of health and potential failures by fusing the outputs of the DA, DM, SD and other HA blocks.

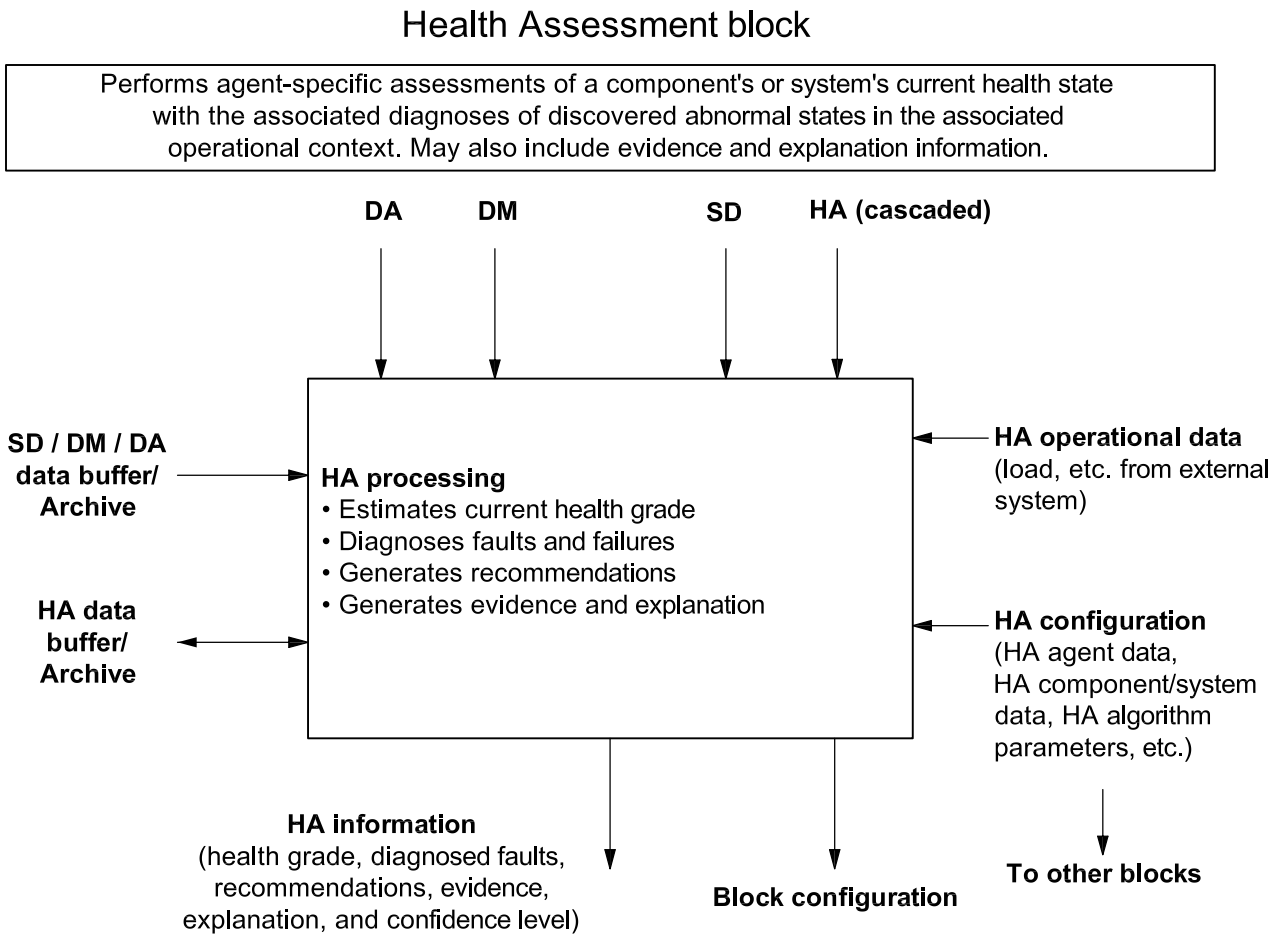

Figure 6 — Health Assessment block

An output of this block includes the component/system's current health grade and diagnosed faults and failures with associated likelihood probability. A calculation of the current risk priority number (RPN) may also be performed. Modelling of ambiguity groups and multiple hypotheses may be included in the output data structures. The HA block may also output an explanation detailing the evidence for a diagnosis or health grade.

4.6 Prognostic Assessment (PA) blocks

As shown in Figure 7, the primary function of the PA block is to project the future state of the monitored equipment using a combination of prognostic models and their algorithms, including future operational usage model(s). This block determines the future state of health and failure modes by combining the relevant outputs of the DA, DM, SD, HA and other PA blocks and applying a prognostic algorithm or model based on supplied projected operational utilization. To aid the algorithm or model, the HA block may also retrieve account historical failure data and operational history, along with projected failure rates related to operational utilization.

The prognostics layer may report health grade at a future time or may estimate the remaining life of an asset given its projected usage profile. Assessments of future health or remaining life may also have an associated prognosis of the projected fault condition. A calculation of the future risk priority number (RPN) may also be performed. An output of this block includes the component/system's future health grade and future failure events with associated likelihood probability. Modelling of ambiguity groups and multiple hypotheses may be included in the output data structures. The PA block may also output an explanation detailing the evidence for a proposed failure event or health grade.

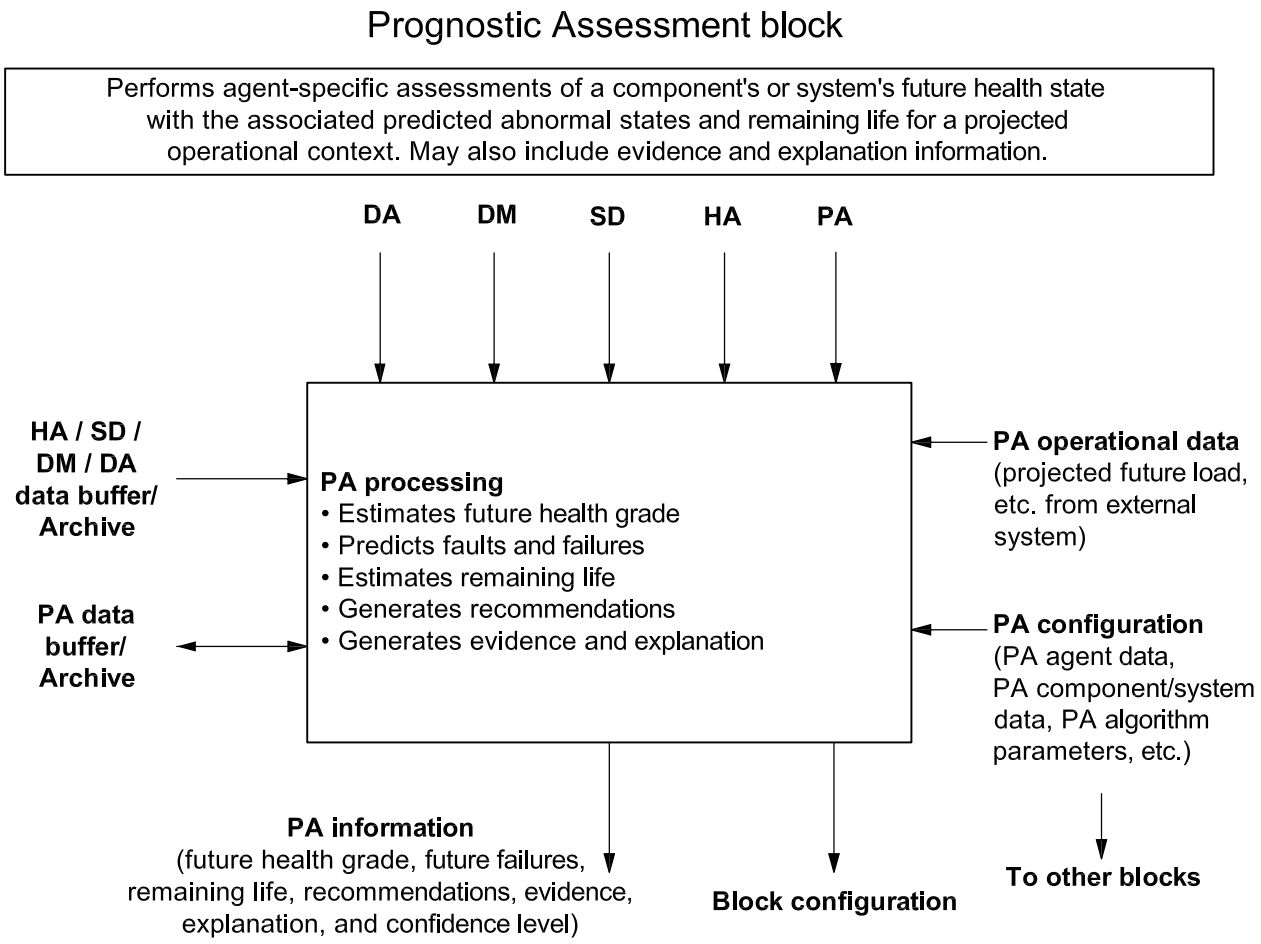

Figure 7 — Prognostic Assessment block

4.7 Advisory Generation (AG) blocks

As detailed in Figure 8, the primary function of the AG block is to integrate information from DA, DM, SD, HA, PA and other AG blocks and external constraints (safety, environmental, budgetary, etc.), and to provide optimized recommended actions and alternatives to applicable personnel or external systems. Recommendations may include prioritized operational and maintenance actions and capability forecast assessments or modifying operational profiles to allow mission completion. The decision support module needs to take into account the operational history (including usage and maintenance), current and future mission profiles, high-level unit objectives and resource constraints.

Maintenance advisories from this block should detail future maintenance work required, which may include the verification of monitoring data or the performance of additional monitoring. The structure of these advisories should be put into a “work request” format for external maintenance work management systems. Based on this request, maintenance work management systems can schedule work in advance and locate spare parts and tools required for these jobs.

Operational advisories from this block can be immediate in nature, such as the current notification of operators of alerts and resulting action steps. Other production-related advisories can be more strategic, such as sending a notice to a production planning system about the high risk of failure on a production line due to a soon-to-fail critical piece of equipment.

Capability forecast assessments from this block provide the results for requests about the likelihood of accomplishing a specific mission or production run. These assessments are critical to production forecasting systems when evaluating whether or not to accept certain missions/orders and where to assign the work, based on asset optimization principles.

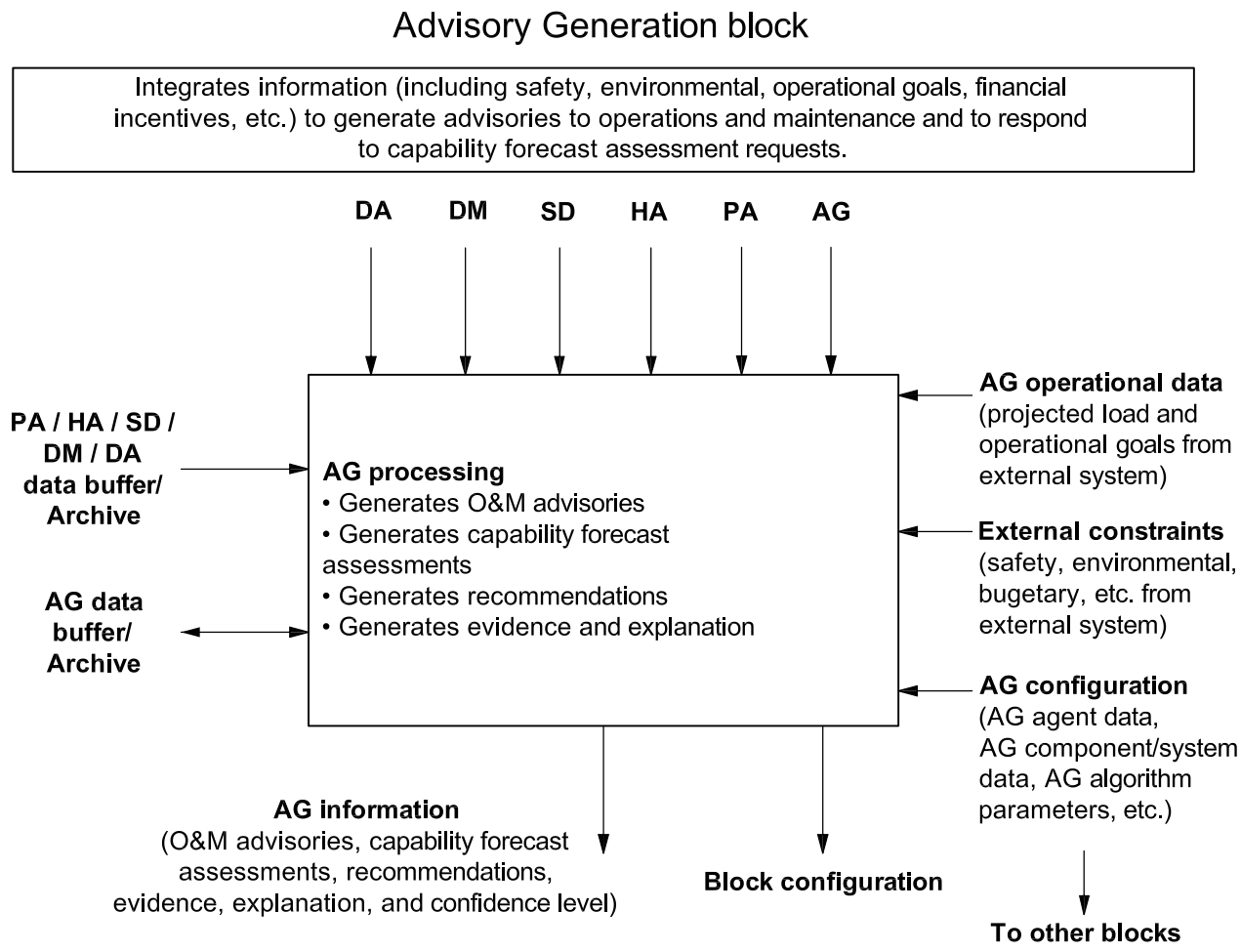

Figure 8 — Advisory Generation block

4.8 Block configuration

Each data processing block requires configuration information, some of which may be static data, and other parameters may be changed dynamically by the system during operation.

As an example, the following is a sample of the configuration of the Data Acquisition block:

a) measurement location description (measurement location table)

   1) orientation and relative position,

   2) location description;

b) monitoring intervals – dynamic vs. static

   1) on-line continuous,

   2) on-line polled,

      — default polling rate,

      — default parameters;

c) triggered vs. non-triggered

   1) set points,

   2) deadband;

d) asynchronous vs. synchronous;

e) transducer information

   1) response curve,

   2) measurement confidence,

   3) transducer electronic data sheet (TEDS) information;

f) calibration;

g) channels

   1) single or multiple channel collection.

4.9 External systems

Retrieval of previous work histories from the maintenance system and previous operational data (starts/stops/loads) from a process data historian is important in the assessment of machinery health. After a health assessment is made, the maintenance action to be taken can range from increasing the frequency of inspection, to repair or replacement of the damaged machinery or component. The effect on operations may be an adjustment of operating procedures or a request to shutdown the equipment immediately. This need for rapid communication to maintenance and operational systems requires software interfaces to maintenance management systems and operational control systems. These interfaces are useful in order to communicate recommended actions in the form of maintenance work requests and operational change requests.

4.10 Data archiving

Data archiving is an important feature during all processing of a machine condition monitoring program. Previous data trends can be analysed for statistical relevance. The data archiving system should provide rules for the archiving rate and amount of data stored. Previous advisories should be audited for accuracy and root cause information added upon its discovery.

4.11 Technical displays

Relevant technical displays showing data from each block are necessary to facilitate analysis by qualified personnel. These displays should provide the analyst with the data required to identify, confirm or understand an abnormal state.

4.12 Information presentation

Information from the HA, PA and AG blocks is displayed by this processing block. It is important that the data be converted to a form that clearly represents the information necessary to make corrective action decisions. In some cases, the user will need the ability to drill down into the SD, DM and DA technical displays when abnormalities are reported.

4.13 Compliant specifications

An open CM&D processing architecture specification shall utilize the processing architecture as described above. Specifications shall specify a processing data model/schema and an Application Programming Interface (API) description which meets ISO/IEC 14750 for the processing blocks which they expose. This will allow various data processing blocks from various suppliers to be integrated into a complete, functional system. MIMOSA publishes an open CM&D specification which is compliant with the above requirements. The specification is known as the MIMOSA Open Systems Architecture for Condition Based Maintenance (OSA-CBM™) specification. Annex A describes this specification in more detail.

Figure 9 shows how these blocks can interact with each other to form a complete integrated system. The hub of the wheel structure represents the communications medium between the modules, which may be accomplished using popular communication and middleware technologies (see Annex B for an understanding of middleware). Therefore, the modules do not need to reside on the same machine but may reside anywhere on a local or worldwide network. Open systems architecture design enables the integration of improved prognostic capability within new or existing system designs, allowing maximum flexibility for future upgrades to the system.

A module may implement the functionality of one or more data processing blocks from the processing architecture. For example, module A from Figure 10 implements the functionality of the State Detection block. Module B, on the other hand, implements functionality of two blocks: Health Assessment and Prognostic Assessment. A module may implement one or more block APIs. Modules from other layers implement one or more layer APIs. For instance, Module D from Figure 10 implements the functionality of two blocks: Data Manipulation and State Detection. In this case, the module provides information about manipulated data and computed conditions.

An example of a compliant system is shown in Figure 11. The system has the largest number of Data Acquisition (DA) blocks which feed into more complex blocks until finally one Advisory Generation (AG) block sends its maintenance and operations decision support to an external user display.

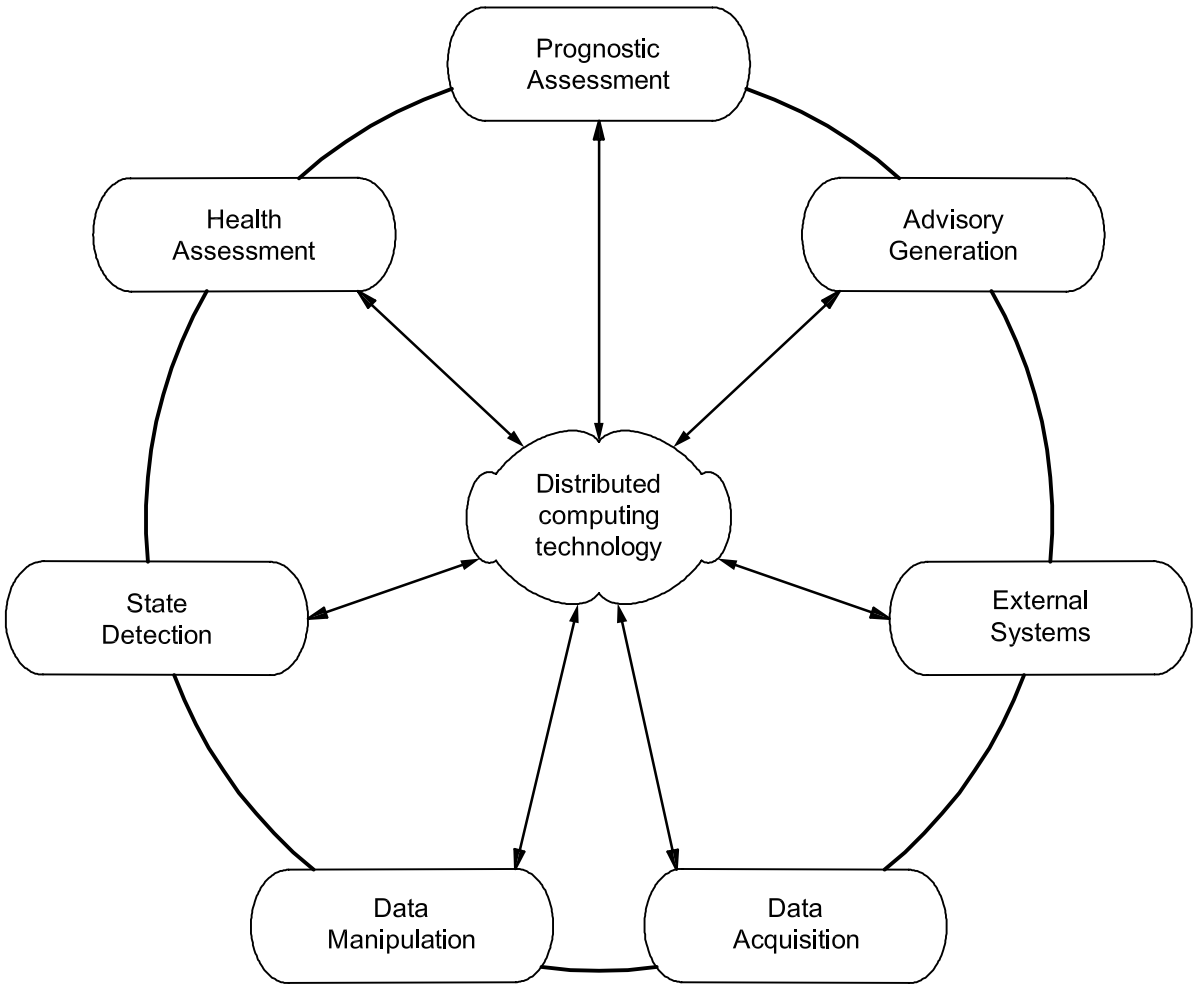

Figure 9 — Data processing flow within a standard architecture

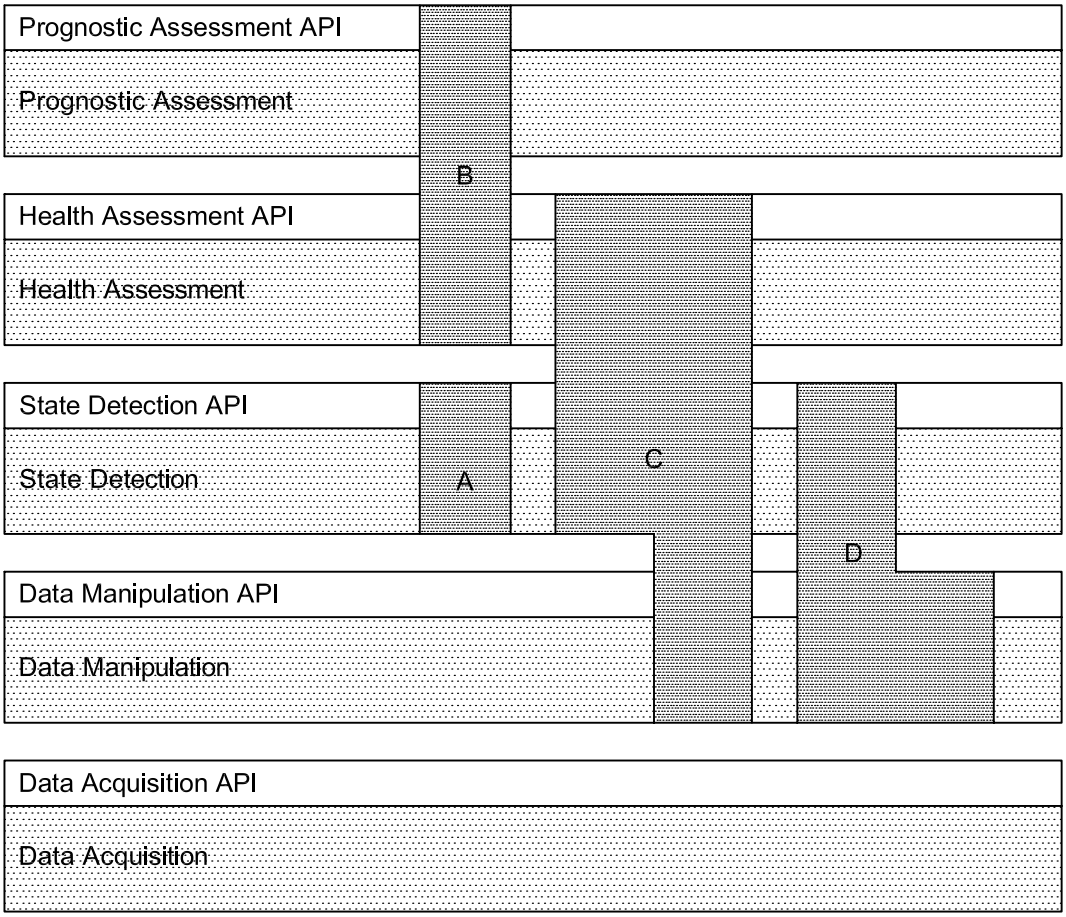

Figure 10 — Example of compliant modules

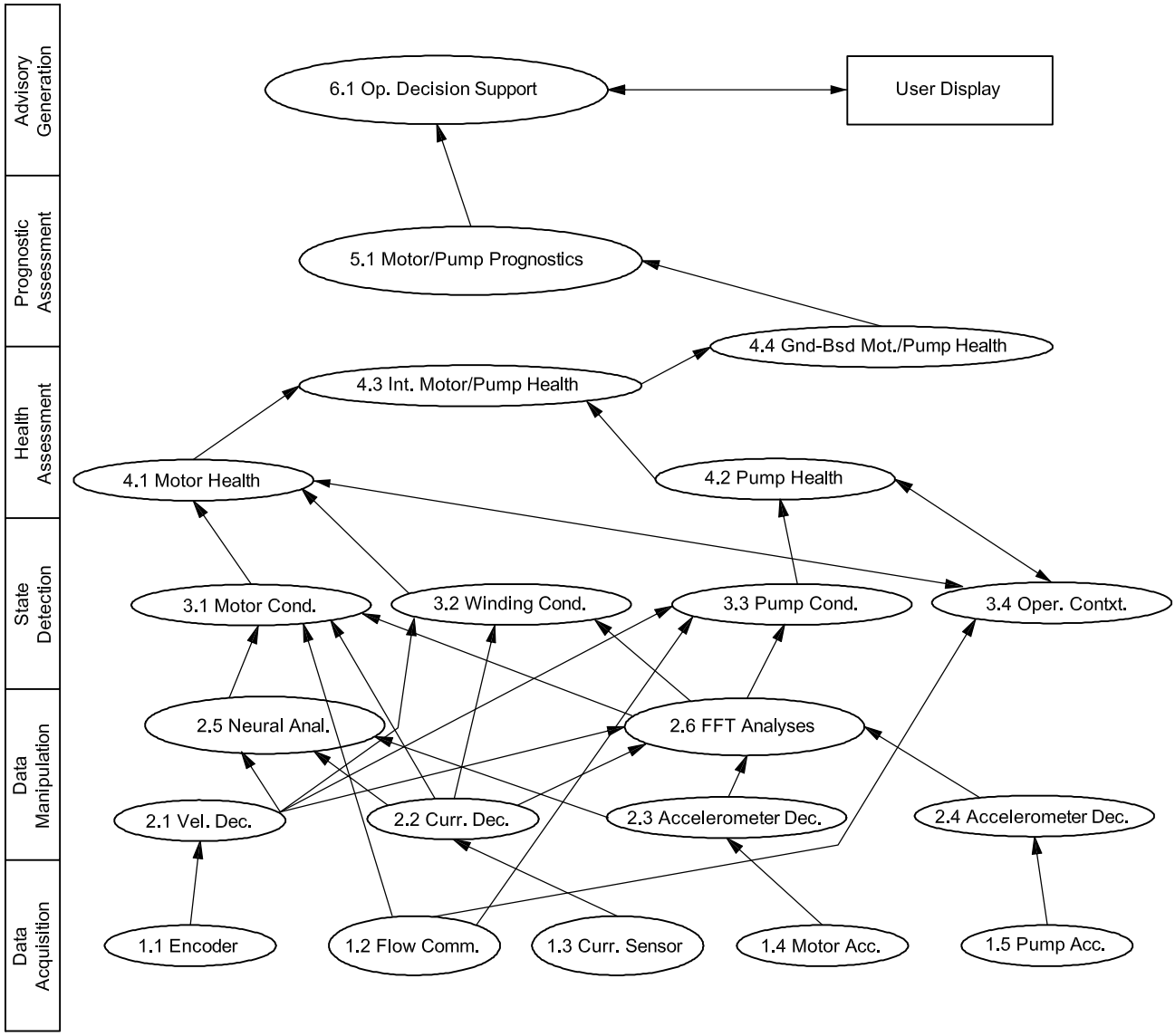

Figure 11 — Example of a compliant modular system

Annex A
(informative)

Compliant specifications

A.1 MIMOSA's OSA-EAI™ CM&D information architecture specification

A.1.1 Introduction

MIMOSA is a non-profit trade association, which develops and encourages adoption of open information standards for operations and maintenance. MIMOSA is composed of industrial asset management system providers and industrial asset system end-users who develop open information integration specifications for managing physical assets.

MIMOSA's Open Systems Architecture for Enterprise Application Integration (OSA-EAI) specification (publicly released version available for free download at http://www.mimosa.org/) is compliant as a CM&D information architecture specification with ISO 13374-1 and this part of ISO 13374, and facilitates the integration of asset management and CM&D information throughout multi-site enterprises. MIMOSA's OSA-EAI system specifications offer advantages for maintenance and reliability users, as well as technology developers and suppliers.

For users, the adoption of MIMOSA OSA-EAI specifications facilitates the integration of asset management information, provides a freedom to choose from a broader selection of software applications, and saves money by reducing integration and software maintenance costs.

For technology suppliers, the adoption of MIMOSA OSA-EAI specifications stimulates and broadens the market, allows concentration of resources on core high-value activity rather than low-value platform and custom interface requirements, and provides an overall reduction in development costs.

MIMOSA regularly updates the OSA-EAI standard to add additional capabilities. Users and technology suppliers are encouraged to refer to MIMOSA's Web site (http://www.mimosa.org/) for the latest version of the standard.

A.1.2 Architecture

A.1.2.1 Diagram of architecture

The OSA-EAI Version 3.1 architecture is shown in Figure A.1.

<table>
<tr><td>Tech-Doc-File
For XML File
Imports & Export</td><td>Tech-CDE-Services for
Web Service Tech-XML
Clients & Servers</td><td>Tech-XML-Web for
HTTP Tech-XML
Clients & Servers</td><td>Tech-XML-Services
For Web Service
Tech-XML
Clients & Servers</td><td>EAI Application
Interoperability</td></tr>
<tr><td>Tech-Doc CRIS
Unpackaged
XML Content
Producer/Consumer</td><td>Tech-CDE-Aggregate
CRIS XML Transaction
Clients & Server Schema</td><td colspan="2">Tech-XML Atomic
CRIS XML Transaction
Client & Server Schema</td><td>XML content</td></tr>
<tr><td colspan="4">CRIS Reference data library (see 3.5)</td><td>MetaData taxonomy</td></tr>
<tr><td colspan="4">Common Relational Information Schema (CRIS) (see A.1.2.2.3)</td><td>Implementation model</td></tr>
<tr><td colspan="4">OSA-EAI Common Conceptual Object Model (CCOM) (see A.1.2.2.2)</td><td>Conceptual model</td></tr>
<tr><td colspan="4">OSA-EAI Terminology dictionary (see A.1.2.2.1)</td><td>Semantic definitions</td></tr>
</table>

Technology types [*Tech*-]

REG	(Object Registry Management)
WORK	(O&M Agent Work Management)
DIAG	(Diagnostics/Prognostics/Health Assessment)
TREND	(Operational Scalar Data & Alarms)
DYN	(Dynamic Vibration/Sound Data & Alarms)
SAMPLE	(Oil/Fluid/Gas/Solid Test Data & Alarms)
BLOB	(Binary Data/Thermography Data & Alarms)
REL	(RCM/FMECA/Model Reliability Information)
AGENT	(Agent Information and Roles)
TRACK	(Physical Asset GeoSpatial Tracking Information)

Figure A.1 — MIMOSA OSA-EAI architecture diagram

A.1.2.2 Definition of terms

A.1.2.2.1 OSA-EAI terminology dictionary

To ease understanding among all parties using MIMOSA's OSA-EAI specification, MIMOSA provides a standard set of terminology in the OSA-EAI Terminology Dictionary. This provides the basic semantic descriptions used throughout the OSA-EAI specifications.

A.1.2.2.2 Common Conceptual Object Model (CCOM)

The Common Conceptual Object Model (CCOM) provides the basic conceptual model basis for OSA-EAI. Software designers familiar with class diagrams in Unified Modelling Language can utilize this model to understand the basic classes, major attributes, and relationships between classes in OSA-EAI. CCOM V3.0 is available in PDF document form and Visio (VSD) format.

A.1.2.2.3 Common Relational Information Schema (CRIS)

MIMOSA's Common Relational Information Schema (CRIS) provides a common implementation schema which allows information from many systems to be communicated and integrated. The schema is in a relational format, commonly used in most database systems. Each table in CRIS is assigned a unique table

reference number. CRIS V3.0 is published in PDF document form, Word document form (DOC), and XML Schema (XSD) form.

Data sources which house asset information are not required to be physically redesigned to match CRIS, but must be able to translate their information into CRIS tables and columns. In order to ease this translation and for MIMOSA implementers who desire to implement a physical CRIS database, MIMOSA provides ORACLE and Microsoft SQLServer table creation scripts.

A.1.2.2.4 Common Conceptual Object Model and Common Relational Information Schema (CCOM/CRIS)

CCOM/CRIS contains standard enterprise, site, functional segment, asset and agent identification nomenclature for the corporate level of an organization or the top organizational structure of a non-profit or military body. Each **Enterprise** is associated with exactly one **Enterprise Type**. An enterprise uniquely assigns **Site Unique Integration Codes** to new **Sites** and may control one or more **Sites** (which could have formerly been controlled by other enterprises). In order for multiple enterprises to exchange MIMOSA information, every **Enterprise** must request and utilize its unique, unchanging **Enterprise Unique Integration Code**.

A **Site** is what an enterprise defines as an entity that can be decomposed into **segments** and which generates new assets, agents, databases and measurement locations. A given enterprise can contain many sites. A site can contain many segments. For facility applications, the **Site** normally represents a tangible as-built building. For industrial and manufacturing applications, this entity normally represents a physical plant. For fleet applications, this entity might represent a maintenance depot which has responsibility for assets. Some very large assets, such as an aircraft carrier, may be defined as a **Site** itself. Each **Enterprise** uniquely assigns every **Site** its unique, unchanging **Site Unique Integration Code**.

Because CCOM and CRIS are designed for multi-site collaborative asset life cycle management, nearly all tables in CRIS (except MIMOSA-owned reference tables) include a reference to the enterprise/site/database. To keep the number of primary key columns to a minimum, CRIS V3 combines the enterprise ID (also 4-bytes) and the site ID into a fixed-length 16-character MIMOSA **site code**. This site code is composed of these two 4-byte non-negative integers which are converted into their hexadecimal format (resulting in 8 characters per integer) and then concatenated into a fixed-length 16-character string.

Version 3.1 of CCOM and CRIS have also added the ability to break up functional locations at a site into multiple hierarchies of **segments** where serialized **assets** can be installed over time. In addition, CCOM/CRIS provides for a method of standard measurement location identification across various condition monitoring technologies. Trendable, scalar data such as operational temperatures, pressures and loads are modelled in CCOM/CRIS. CCOM/CRIS support dynamic data, such as time waveforms and Fast Fourier Transforms (FFTs), which are used in vibration analysis and sound monitoring. Binary data, known as Binary Large OBjects or (BLOBs), are supported for communicating drawings, reports, diagrams and photographs. CCOM/CRIS also manages sampling test data results, such as used-oil analysis test data and air quality monitoring data. CCOM/CRIS also allows the communication of diagnostic, health and prognostic information from smart systems, and eases the generation of advisory recommendations. Special maintenance and reliability tables define fields for events (actual, hypothesized, proposed), health, estimated asset life assessment and recommendations. CCOM/CRIS model maintenance and production work request scheduling and the tracking of the completion (or non-completion) of a maintenance or production job as related to an asset. CCOM/CRIS also provides the information framework for storing reliability data for assets.

A.1.2.2.5 CRIS reference database specification

CCOM/CRIS contain many “type” classes/tables, which allow users to associate types of enterprises, sites, segments, assets, agents, measurement locations, engineering units, etc. with standard numeric codes, common throughout all their various systems. MIMOSA generates and maintains industry-standard taxonomies and codes for most of these tables, but CCOM/CRIS allow both suppliers and end-users to add industry-specific and customer-specific entries to these tables. MIMOSA experts have generated a large reference database, the CRIS reference database specification in XML and several popular relational database formats. Version 3.1 of this database specification contains many useful codes which allow standardization across many disparate systems, even those from various countries. For example, the asset

type table allows standard querying of common asset types, such as “AC induction motor”, which have unchanging three-integer unique identifiers. Other standard code tables include service segment, measurement location, engineering units, sampling test codes, diagnostic/prognostic event codes, health codes, failure codes and root cause codes. This allows systems to search across various systems for common failures on certain equipment types.

A.1.2.2.6 *Tech*-Doc Producer/Consumer interface schemas

For XML document-based systems, MIMOSA provides the *Tech*-Doc Producer/Consumer interface schemas, which publish/consume CRIS data from a given technology in an XML document format. *Tech*-Doc interfaces specify the contents of an XML document, but do not specify the physical storage and transport method of the produced/consumed document. *Tech*-Doc Consumer Read-only applications will utilize a *Tech*-Doc XML document, but will not change its permanent data storage based on this information. *Tech*-Doc Consumer Write-only applications change their permanent data storage after successful file import.

A.1.2.2.7 *Tech*-CDE Aggregate CRIS XML Transaction Client/Server interface schemas

Another component of OSA-EAI is the *Tech*-Compound Document Exchange (*Tech*-CDE) Client/Server interface schemas. These XML schemas provide a common set of XML-based client/server interface definitions for request and transfer of large data sets of CRIS XML data normally physically stored in a CRIS format on a server system. The *Tech*-CDE interfaces specify the contents of a client/server data exchange, but do not specify the physical transport method.

A.1.2.2.8 *Tech*-XML Atomic CRIS XML Transaction Client/Server interface schemas

A key component of MIMOSA’s Open System Architecture for Enterprise Application Integration (OSA-EAI) is the *Tech*-XML Client/Server interface schemas. These XML schemas provide a common set of XML-based client/server interface definitions for various communication protocols, enabling proprietary systems to respond to a discrete transaction with a CRIS-formatted response. The *Tech*-XML interfaces specify the contents of a client/server data exchange, but do not specify the physical transport method.

A.1.2.2.9 *Tech*-CDE, *Tech*-XML and *Tech*-Doc technology types

All three *Tech*-interfaces (*Tech*-CDE, *Tech*-XML and *Tech*-Doc) are further sub-divided into ten (10) technology types (see Figure A.1). This allows developers to focus on supporting only what is relevant to their particular application area, such as vibration analysis or maintenance management. The technology types are listed in Figure A.1.

To refer to the entire set of 10 application technology packages, the italicized prefix “*Tech*-” is used. Each vertical application technology only has certain CRIS tables which are relevant. This allows implementers to pare down the entire CRIS specification to only the application-specific tables.

A.1.2.2.10 *Tech*-XML-Services and *Tech*-CDE-Services specifications

The OSA-EAI *Tech*-XML-Services and *Tech*-CDE-Services specifications are used for building a Service Oriented Application (SOA) server or a client which can communicate data and information between condition monitoring systems, diagnostic systems, reliability management systems, registry management systems and work management systems via the Simple Object Application Protocol (SOAP) using XML messages. The interfaces are defined using XML schemas and specified in a client/server fashion.

A.2 MIMOSA's OSA-CBM™ CM&D Processing Architecture Specification

MIMOSA’s Open Systems Architecture for Condition-Based Maintenance (OSA-CBM) specification (available for download at http://www.mimosa.org/) is compliant as a CM&D processing architecture specification. Starting with data acquisition and progressing towards decision support, the general functions of the layers are specified below. Each layer has the capability of requesting data from any functional layer as needed, however data flow will usually occur between adjacent functional layers.

a) *Layer 1 — Data Acquisition:* The data acquisition module has been generalized to represent the software module that provides system access to digitized sensor or transducer data. The data acquisition module may represent a specialized data acquisition module that has analog feeds from legacy sensors, or it may collect and consolidate sensor signals from a data bus. Alternatively, it might represent the software interface to a smart sensor (e.g. IEEE 1451.2 — compliant sensor). The data acquisition module is basically a server of calibrated digitized sensor data records.

b) *Layer 2 — Data Manipulation:* The data manipulation module may perform single and/or multi-channel signal transformations along with specialized CBM feature extraction algorithms.

c) *Layer 3 — State Detection:* The primary function of the state detection module is to compare features against expected values or operational limits and output enumerated condition indicators (e.g. level low, level normal, level high, etc.). The state detection module may also generate alerts based on defined operational limits. When appropriate data are available, the condition monitor may generate assessments of operational context (current operational state or operational environment).

d) *Layer 4 — Health Assessment:* The primary function of the health assessment layer is to determine if the health of a monitored system, subsystem or piece of equipment is degraded. If the health is degraded, this layer may generate a diagnostic record that proposes one or more possible fault conditions with an associated confidence. The health assessment module should take into account trends in the health history, operational status and loading, and the maintenance history.

e) *Layer 5 — Prognostic Assessment:* The primary function of the prognostics layer is to project the current health state of equipment into the future, taking into account estimates of future usage profiles. The prognostics layer may report health status at a future time, or may estimate the remaining useful life (RUL) of an asset given its projected usage profile. Assessments of future health or RUL may also have an associated diagnosis of the projected fault condition.

f) *Layer 6 — Advisory Generation:* The primary function of the advisory generation module is to provide recommended actions and alternatives, and the implications of each recommended action. Recommendations include maintenance action schedules, modification of the operational configuration of equipment in order to accomplish mission objectives, or modification of mission profiles to allow mission completion. The decision support module needs to take into account operational history (including usage and maintenance), current and future mission profiles, high-level unit objectives, and resource constraints.

After the specification is defined and developed for each CBM module, the modules can be constructed into a complete functional CBM system. Open systems architecture design enables the integration of improved prognostic capability within new or existing system designs, allowing maximum flexibility and upgradeability of the system.

A.3 OSA-CBM framework

The OSA-CBM framework was developed around input from the functional layer descriptions and existing and emerging standards for monitoring and maintenance, such as MIMOSA's OSA-EAI Information Schema, AI-ESTATE, and IEEE 1451.2. The OSA-CBM framework currently excludes the decision support layer since it is application specific. The presentation layer was designed as a client-only layer in order to stay open to any user interface technology; no interfaces are therefore defined in this part of ISO 13374. The first step was defining an object-oriented data model in Unified Modeling Language (UML) for each layer that was then converted into an abstract interface specification. The abstract specification can then be converted to the desired middleware language for a specific interface definition.

The UML object model defines interfaces only. For a given layer of the architecture, the data model does not prescribe the object classes that would be required for a software implementation. The focus is on describing the structure of the information that might be of interest to clients of that layer. OSA-CBM does not impose any requirements on the internal structure of compliant software modules. The architectural constraints are applied to the structure of the public interface and to the behaviour of the modules. This approach allows complete encapsulation of proprietary algorithms and software design approaches within the software module.

Annex B
(informative)

References to UML, XML and Middleware

B.1 Purpose

This annex provides a simple reference of definitions and glossary of terms on Unified Modelling Language (UML), eXtensible Markup Language (XML) and middleware services. In addition, a number of useful references have been provided along with pointers for simple tutorials and detailed tutorials on these topics.

B.2 Unified Modeling Language (UML)

B.2.1 Definition of UML

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing and documenting the artifacts of a software system. UML offers a standard way to write a system's blueprints including conceptual entities, such as business processes and system functions, as well as concrete entities, such as programming language statements, database schemas and reusable software components [27].

B.2.2 Glossary of Terms

Terms and their definitions have come from various sources, in particular from Reference [20].

Activity — A step or action within an Activity Diagram, which represents an action taken by the system or by an actor.

Activity Diagram — A glorified flowchart that shows the steps and decisions and parallel operations within a process, such as an algorithm or a business process.

Actor — A person or an external computer system that interacts with the software under design.

Association — A connection between two elements of a Model. This might represent a member variable in code, or the association between a personnel record and the person it represents, or a relation between two categories of workers, or any similar relationship. By default, both elements in an Association are equal, and are aware of each other through the Association. An Association can also be a Navigable Association, meaning that the source end of the association is aware of the target end, but not vice versa.

Association Class — A Class that represents and adds information to the Association between two other classes.

Attribute — A data field or property that represents information about a Classifier.

Base Class — A Class which defines Attributes and Operations that are inherited by a Subclass via a Generalization relationship.

Branch	A decision point in an Activity Diagram. Multiple Transitions emerge from the Branch, each with a Guard Condition. When control reaches the Branch, exactly one Guard Condition must be true, and control follows the corresponding Transition.
Class	A category of similar Objects, all described by the same Attributes and Operations and all assignment-compatible.
Class Diagram	A diagram that shows relationships between various Classes and Interfaces.
Classifier	A UML element that has Attributes and Operations, specifically, Actors, Classes and Interfaces.
Collaboration	A relation between two Objects in a Collaboration Diagram, indicating that Messages can pass back and forth between the Objects.
Collaboration Diagram	A diagram that shows Collaborations between Objects and Messages that pass along those Collaborations to carry out some behaviour.
Component	A deployable unit of code within a software system.
Component Diagram	A diagram that shows relationships between various Components and Interfaces.
Dependence	A relationship that indicates one Classifier knows the Attributes and Operations of another Classifier, but is not directly connected to any instance of the second Classifier.
Deployment Diagram	A diagram that shows relationships between various Processors.
Element	Any item that appears in a Model.
Event	In a State Diagram, this represents a signal or event or input that causes the system to take an action or switch States.
Final State	In a State Diagram or an Activity Diagram, this indicates a point at which the diagram completes.
Fork	A point in an Activity Diagram where multiple parallel control threads begin.
Generalization	An inheritance relationship in which a Subclass inherits and adds to the Attributes and Operations of a Base Class.
Initial State	In a State Diagram or an Activity Diagram, this indicates the point at which the diagram begins.
Interface	A Classifier that defines Attributes and Operations that form a contract for behaviour. A provider Class or Component may elect to Realize an Interface (i.e. implement its Attributes and Operations). A client Class or Component may then Depend upon the Interface and thus use the provider without any details of the true Class of the provider.
Join	A point in an Activity Diagram where multiple parallel control threads synchronize and rejoin.
Lifeline	A line in a Sequence Diagram that indicates the duration of an Object.
Member	An Attribute or an Operation within a Classifier.

Merge	A point in an Activity Diagram where different control paths come together.
Message	In a Sequence Diagram or a Collaboration Diagram, a communication from one Object to another, delivering information or requesting a service.
Model	The central UML artifact. Consists of various elements arranged in a hierarchy by Packages, with relations between elements as well.
Navigability	Indicates which end of a relationship is aware of the other end. Relationships can have bidirectional Navigability (each end is aware of the other) or unidirectional Navigability (one end is aware of the other, but not vice versa).
Navigable Association	An Association with unidirectional Navigability.
Note	A text note added to a diagram to explain the diagram in more detail.
Note Attachment	A dashed line connecting a Note to an element that it describes.
Object	In an Activity Diagram, an object that receives information from Activities or provides information to Activities. In a Collaboration Diagram or a Sequence Diagram, an object that participates in the scenario depicted in the diagram. In general: one instance or example of a given Classifier (Actor, Class or Interface).
Operation	A method or function that a Classifier can perform.
Package	A Model element that divides the Model into a hierarchy.
Package Diagram	A Class Diagram in which all of the elements are Packages and Dependencies.
Parameter	An argument to an Operation.
Private	A Visibility level applied to an Attribute or an Operation indicating that only code for the Classifier that contains the member can access the member.
Processor	In a Deployment Diagram, this represents a computer or other programmable device where code may be deployed.
Protected	A Visibility level applied to an Attribute or an Operation indicating that only code for the Classifier that contains the member or for its Subclasses can access the member.
Public	A Visibility level applied to an Attribute or an Operation indicating that any code can access the member.
Realization	Indicates that a Component or a Class provides a given Interface.
Sequence Diagram	A diagram that shows the existence of Objects over time and the Messages that pass between those Objects over time to carry out some behaviour.
State	In a State Diagram, this represents one state of a system or subsystem, i.e. what it is doing at a point in time, as well as the values of its data.
State Diagram	A diagram that shows States of a system or subsystem, Transitions between States, and the Events that cause the Transitions.

Static	A modifier to an Attribute to indicate that there's only one copy of the Attribute shared among all instances of the Classifier. A modifier to an Operation to indicate that the Operation stands on its own and does not operate on one specific instance of the Classifier.
Stereotype	A modifier applied to a Model element indicating something about it which cannot normally be expressed in UML. In essence, Stereotypes allow the user to define his/her own “dialect” of UML.
Subclass	A Class which inherits Attributes and Operations that are defined by a Subclass via a Generalization relationship.
Swimlane	An element of an Activity Diagram that indicates what parts of a system or a domain perform particular Activities. All Activities within a Swimlane are the responsibility of the Object, Component or Actor represented by the Swimlane.
Transition	In an Activity Diagram, a Transition represents a flow of control from one Activity, Branch, Merge, Fork or Join to another. In a State Diagram, a Transition represents a change from one State to another.
Use Case	In a Use Case Diagram, a Use Case represents an action that the system takes in response to a request from an Actor.
Use Case Diagram	A diagram that shows relations between Actors and Use Cases.
Visibility	A modifier to an Attribute or Operation that indicates what code gives access to the member. Visibility levels include Public, Protected and Private.

B.2.3 Simple tutorials on UML

— Tutorial on how to use the UML to define and build actual systems [30], [31]

— What is UML tutorial [31]

— UML Tutorial: Complex transition [32]

— A UML introduction tutorial [33]

— Borland UML tutorial [34]

B.2.4 More detailed tutorials on UML

— Object Modeling in UML: Introduction to UML, Behavioral Modeling & Advanced Modeling [35], [36], [37].

B.3 eXtensible Markup Language (XML)

B.3.1 Definition of XML

The eXtensible Markup Language (XML) is a subset of SGML that is completely described in this part of ISO 13374. Its goal is to enable generic SGML to be served, received and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML [28].

B.3.2 Glossary of terms

Terms and their definitions have come from various sources, in particular from Reference [21].

Application	A self-contained program that performs a specific function directly for the user. An example of an application is XMLwriter. It is a stand-alone program that allows a user to edit XML files as well as other text files.
Attribute	Attributes are inserted in start or empty element tags in the form attribute_name="attribute_value". They are additional information about an element, intended for interpretation by an application.
Attribute-list (ATTLIST) declaration	Attribute-list declarations are placed inside a DTD. They specify what attributes are allowed in an XML document, to which element they belong, and what the default value of an attribute may be.
Browser	A program that allows a user to interact with information stored in a variety of formats on the World Wide Web, private networks, or locally. Most browsers support a variety of information types including XML, HTML and Java. An example of a browser is Microsoft's Internet Explorer 5.
CDATA	Character data, or text that does not need to be parsed. Markup within CDATA sections will not be interpreted as markup.
Character	In XML, a character is any legal graphic character of the ISO/IEC 10646 or Unicode standard (the two standards are identical). Other legal characters include a carriage return, tab and line feed.
Child element	An element nested inside another (parent) element.
Cascading Style Sheets (CSS)	CSS are used to apply style (formatting) to HTML and XML documents.
Document type (DOCTYPE) declaration	Contains an internal DTD or pointers to an external DTD.
Document Type Definition (DTD)	A set of rules describing the structure of an XML document. The document must conform to these rules in order to be valid.
Empty tag	An element that has no content. In XML, an empty tag follows the syntax <name></name> or <name/>.
End tag	The closing tag of an element. It follows the syntax </name> and must match the name in the start tag to be a well-formed XML document.
Element type	The name that appears in a start, end or empty tag.
ELEMENT type declaration	Element type declarations are placed inside a DTD. They specify what elements are allowed in an XML document, and what their content may be.
ENTITY declaration	Declaration placed inside a DTD. They contain the abbreviation to be used for an entity and the text to be substituted for that abbreviation. They may also contain a URI if the text or data is stored at a remote location.
External DTD	A DTD that is contained in another file which may reside at a remote location.
General entity reference	An entity used in the content of an XML document. General entities follow the syntax '&name;'.

HTML (HyperText Markup Language)	One of the publishing languages of the World Wide Web. HTML consists of a set of predefined tags that tell a browser how to display text and images to the end-user.
Internal DTD	A DTD that is located within the XML document.
Java	A platform-independent, object-oriented computer language that has a similar syntactic structure to C++.
Markup	Syntax used with text to indicate formatting instructions to a processor (or parser). In XML, the characters '<' and '&' signify the start of markup.
Metadata	A definition or description of a collection of data.
Name	In XML, an element, attribute, or entity name must begin with a letter, '_', or ':', and may continue with zero or more letters, digits, '.', '-', '_', or ':'.
NOTATION declaration	Notation declarations are placed inside a DTD. They are used to identify the format of non-XML data, and name (or point to) applications that will interpret the data. An example of a NOTATION declaration is <!NOTATION gif PUBLIC “gif viewer”>.
Parameter entity reference	An entity used within the DTD. Parameter entities follow the syntax '%name;'.
Parent element	An element in which another (child) element is nested.
Parser	An application that processes a document and identifies text from markup. It interprets the markup and determines the content and characteristics of a document. An example of a parser is Microsoft's Internet Explorer 5 XML validating engine.
PCDATA	Parsed character data. Text that is not markup but is processed by a parser.
Processing instruction	Used in an XML document to embed information intended for proprietary applications.
Recursion	When something refers to itself.
Root element	The first element in a document. It may not be contained by any other element in the document, and hence forms the basis of the document's element tree.
Schema document	An XML document which defines the structure and contents of other XML documents in a similar manner to a DTD.
Start tag	The opening tag of an element. It follows the syntax <name> and must match the name in the end tag to be well-formed XML.
Tag	A start tag, end tag, or empty tag. A tag contains the name of an element.
Unicode	A system for the display, interchange, and processing of text written in any of a wide range of supported languages. The Unicode character set is a multi-byte character set that supports a wide range of international characters.
Unparsed	Contains information that cannot be interpreted as text or markup.

URI: Uniform Resource Identifier	The generic set of all names or addresses that refer to resources.
URL: Uniform Resource Locator	URIs that locate and access resources specifically via the Internet. URLs can be absolute (http://www.xmlwriter.net) or relative (doc001.xml).
Validity constraint	Rules defined in the XML specification that are imposed on an XML document via the DTD. A document is valid if it conforms to the DTD and is well-formed.
W3C (World Wide Web Consortium)	The W3C is responsible for releasing standards related to World Wide Web technology.
Well-formedness constraint	Rules defined in the XML specification related to XML syntax. An XML document can exist merely as a well-formed document (i.e. not have any validity constraints).
XML	Extensible Markup Language. It is a subset of SGML (Standard Generalized Markup Language), which uses semantic tags in a structured format. XML offers a flexible way to create information formats and to share both data and metadata with other applications and users.
XML declaration	The processing instruction situated on the first line of an XML document. Every XML document should include an XML declaration.
XML specification	A recommendation released by the World Wide Web Consortium (W3C) on 10 February 1998, as the official standard on how to write documents using XML.

B.3.3 Simple tutorials on XML

— Creating an XML document (Chapter 3 of Reference [38])

— XML tutorial from W3school.com [39]

— Free eBook crashcourse XML [40]

B.3.4 Detailed tutorials on XML

— Book XML DTD [22]

— Case studies using XML (Chapters 27, 28, 29, 30 of Reference [38])

B.4 Middleware services

B.4.1 Definition of Middleware

Middleware is an extremely simple “microkernel” — a reusable, expandable set of services and functions that many applications need in order to function well in a networked environment [24]. Middleware is the software that manages communication between clients and servers over N-tier software architecture [25]. Middleware is the network-aware system software layered between an application, the operating system and the network transport layers whose purpose is to facilitate cooperative processing [26].

B.4.2 Glossary of terms in middleware

Terms and their definitions have come from various sources, in particular from Reference [21].

Application server	A server program that allows the installation of application-specific software components in such a manner that they can be remotely invoked, usually by some form of remote object method call.
Bean-managed persistence	When an Enterprise JavaBean performs its own long-term state management, it is called bean-managed persistence.
Bytecode	In the context of Java, bytecode is the platform-independent executable program code.
Clustering	Aggregating multiple servers together to form a service pool of some kind, usually for achieving redundancy or improving performance.
Component standard	A definition of how software components cooperate and, in particular, the roles and interfaces of each. In the context of Java middleware, component standards usually include specifications of the middleware interfaces exposed to the components, and the component interfaces required by the middleware.
Container-managed persistence	When an Enterprise JavaBean server manages a bean's long-term state, it is called container-managed persistence.
CORBA	Standard maintained by the Object Management Group (OMG) called the Common Object Request Broker Architecture.
COS Naming	CORBA standard for object directories.
Data source	Term used by the JTA and JDBC specifications to refer to a persistent repository of data. It usually represents a database, but may also refer to an object that makes database connections available (i.e. a driver program).
DCOM	Microsoft's distributed component object model.
Enterprise JavaBeans (EJB)	A server component standard developed by Sun Microsystems.
Entity bean	An Enterprise JavaBean that maintains state across sessions and which may be looked up in an object directory by its key value.
Failover	The ability to respond resiliently to a component failure by switching to another component.
IDL	Interface description language, CORBA's syntax for defining object-remote interfaces.
IIOP	Internet Inter-ORB Protocol, CORBA's wire protocol for transmitting remote object method invocations.
ISAPI	Microsoft's C++ API for coding application extensions for its Internet Information Server.
Java Naming and Directory Interface (JNDI)	The Java standard API for accessing directory services, such as LDAP, COS Naming and others.
Java Transaction API (JTA)	Java API for coding client demarcated transactions and for building transactional data source drivers.
JTS	Java Transaction Service, which is the Java binding for the CORBA Transaction Service. JTS provides a way for middleware vendors to build interoperable transactional middleware.
JVM	Java virtual machine.
LDAP	Lightweight Directory Access Protocol, a protocol for directory services derived from X.500.

Middleware	Software that runs on a server and acts as either an application processing gateway or a routing bridge between remote clients and data sources or other servers, or any combination of these.
NSAPI	Netscape's C-language API for adding application extensions to their Web servers.
OMG	Object Management Group, an organization that defines and promotes object-oriented programming standards.
OODB	Object-oriented database.
OODBMS	Object-oriented database management system.
ORB	Object request broker, the primary message routing component in a CORBA product.
Passivate	To place an object in a dormant state when it is not being accessed, such that it can later be returned to an active and usable state.
Persistence	Maintaining state over a long time, especially across sessions.
Pooling	Maintaining a collection of objects, servers, connections, or other resources for ready access, so that one does not need to be created anew each time one is needed.
RMI	Remote Method Invocation, the Java standard technology for building distributed objects whose methods can be invoked remotely across a network.
RMI over IIOP	Using the CORBA IIOP wire protocol from an RMI API.
Servlet	Application extension to a Java Web server.
Session bean	An Enterprise JavaBean that does not maintain its state from one session to the next. A session bean appears to the client as if the bean was created just for that client.
Skeleton	A server-side software component that serves to relay remote calls from a client to the methods of a servant running in a server. Usually a skeleton is automatically generated by a special compiler.
SQLJ	An extended Java syntax for embedding SQL-like commands in a Java program.
Stub	A client-side software component that serves to forward remote calls to a remote server and receive the subsequent responses. Usually a stub is automatically generated by a special compiler.
Three-tier	An architecture in which a remote client accesses remote data sources via an intervening server.
Transaction manager	A software component that coordinates the separate transactions of multiple data sources, so that they behave as a single unified transaction. The transaction manager requires data source drivers that can participate in this kind of coordination. Further, it usually provides the ability to monitor transactions and provide statistics.
Transactional	When an operation has the property that it completes, or if it does not complete due to a failure, it either undoes its own effects or has the ability to complete at a later time when the failure is repaired.

B.4.3 Simple tutorials on middleware

— JDBC program example (Chapters 5 and 7 of Reference [41])

— Java Socket tutorial (Chapter 3 of Reference [43])

— RMI tutorial (Chapter 4 of Reference [43])

— Corba tutorial (Chapter 5 of Reference [43])

— JDBC program example (Chapter 8 of Reference [43])

— Simple example of COM and CORBA (Chapter 2 of Reference [44])

B.4.4 Detailed tutorials on middleware

— Application building in COM (Reference [42])

— Integrating COM-CORBA (Chapters 7-11 of Reference [44])

— Code samples of application on COM-CORBA interoperability [45]

Bibliography

[1] ISO 8601:2000, *Data elements and interchange formats — Information interchange — Representation of dates and times*

[2] ISO 8879:1986, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*

[3] ISO/IEC 9075 (all parts), *Information technology — Database languages — SQL*

[4] ISO/IEC 9506 (all parts), *Industrial automation systems — Manufacturing Message Specification*

[5] ISO/IEC 9579:2000 *Information technology — Remote database access for SQL with security enhancement*

[6] ISO/IEC 10646:2003, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

[7] ISO/IEC 10746 (all parts), *Information technology — Open Distributed Processing — Reference Model*

[8] ISO 13372, *Condition monitoring and diagnostics of machines — Vocabulary*

[9] ISO 13373-1, *Condition monitoring and diagnostics of machines — Vibration condition monitoring — Part 1: General procedures*

[10] ISO 13379, *Condition monitoring and diagnostics of machines — General guidelines on data interpretation and diagnostic techniques*

[11] ISO 13380, *Condition monitoring and diagnostics of machines — General guidelines on using performance parameters*

[12] ISO 13381-1, *Condition monitoring and diagnostics of machines — Prognostics — Part 1: General guidelines*

[13] IEEE 1451.2:1997, *Smart Transducer Interface for Sensors and Actuators — Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*

[14] ISO 14830-1, *Condition monitoring and diagnostics of machines — Tribology-based monitoring and diagnostics — Part 1: General guidelines*

[15] ISO 17359, *Condition monitoring and diagnostics of machines — General guidelines*

[16] ISO 18436-1, *Condition monitoring and diagnostics of machines — Requirements for training and certification of personnel — Part 1: Requirements for certifying bodies and the certification process*

[17] ISO 18436-2, *Condition monitoring and diagnostics of machines — Requirements for training and certification of personnel — Part 2: Vibration condition monitoring and diagnostics*

[18] ISO/IEC 19500-2, *Information technology — Open Distributed Processing — Part 2: General Inter-ORB Protocol (GIOP)/Internet Inter-ORB Protocol (IIOP)*

[19] ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

[20] http://www.tabletuml.com/Help/UMLGlossary.htm

[21] http://xmlwriter.net/xml_guide/glossary.shtml

[22] http://www.xmlxperts.com/xmlbookdtd.htm

[23] http://www.javaworld.com/javaworld/jw-04-1999/jw-04-middleware.html

[24] http://www.cra.org/Policy/NGI/draft/mid2.html

[25] http://www.prolifics.com/docs/panther/html/gl_html/glossary.htm

[26] http://iishelp.web.cern.ch/IISHelp/iis/htm/core/iigloss.htm#GlossaryM

[27] http://www.omg.org/uml/

[28] http://www.w3.org/TR/2004/REC-xml-20040204

[29] http://www.sparxsystems.com.au/UML_Tutorial.htm

[30] http://www.sparxsystems.com.au/WhitePapers/The_Business_Process_Model.pdf

[31] http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/what_is_uml.htm

[32] http://www.objectmentor.com/resources/articles/cplxtrns.pdf

[33] http://www.cragsystems.co.uk/ITMUML/

[34] http://bdn.borland.com/article/0,1410,31863,00.html

[35] ftp://ftp.omg.org/pub/docs/omg/99-11-04.pdf

[36] ftp://ftp.omg.org/pub/docs/omg/00-01-05.pdf

[37] ftp://ftp.omg.org/pub/docs/omg/00-03-03.pdf

[38] NAVARRO, A., WHITE, C. and BURMAN, L., *Mastering™ XML*, Sybex, 2000

[39] http://www.w3schools.com/xml/default.asp

[40] http://www.spiderpro.com/ebooks/kickstart_tutorial_xml.pdf

[41] CHAN, H. *et al*., *E-Commerce: Fundamentals and Applications*, John Wiley & Sons, 2001

[42] BOX, D., *Essential COM*, Addison Wesley, 1998

[43] BOGER, M., *Java™ in Distributed System*, John Wiley & Sons, 1999

[44] ROSEN, M. and CURTIS, D., *Migrating CORBA and COM Applications*, John Wiley & Sons, 1998

[45] GERAGHTY, R. *et al*. *COM-CORBA Interoperability*, Prentice Hall PTR, 1999

ICS 17.160; 35.240.99

Price based on 33 pages