INTERNATIONAL STANDARD

ISO 13281

First edition
1997-08-15

Industrial automation systems — Manufacturing Automation Programming Environment (MAPLE) — Functional architecture

Systèmes d'automatisation industrielle — Environnement de programmation pour l'automatisation industrielle (MAPLE) — Architecture fonctionnelle

Reference number
ISO 13281:1997(E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

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.

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.

International Standard ISO 13281 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 5, *Architecture and communications*.

Annexes A to C of this International Standard are for information only.

© ISO 1997

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Organization for Standardization
Case postale 56 • CH-1211 Genève 20 • Switzerland
Internet central@iso.ch
X.400 c=ch; a=400net; p=iso; o=isocs; s=central

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Introduction

Over the recent past, manufacturing systems have become considerably more flexible and have acquired greater functionality. The numbers and types of component devices of manufacturing systems, such as NC machines, robots, automated guided vehicles and programmable controllers, have increased. Furthermore, there is a definite trend for some of these devices to be incorporated in manufacturing cells. Manufacturing engineers are thus required to develop and update programs not only for many kinds of individual devices but also for combinations of devices and ultimately manufacturing cells. Due to this fact, the difficulty of integrating and programming the control of manufacturing operations has increased.

Manufacturing programs have an intense need for a large variety of manufacturing data, including product oriented data, process oriented data, operation oriented data and management oriented data. This diversity means that manufacturing data has a much more complicated and varied schema than the usual processing data encountered in other systems, e.g., business systems. Therefore, the use and management of manufacturing databases requires a special high-level technology.

MAPLE addresses the following problems that have been traditionally recognized to be within the domain of manufacturing application programming languages for automated production:

– Manufacturing presents a diversity of tasks with widely varying requirements and constraints. Often, addressing these tasks requires programming. Because of the diversity of requirements and constraints, a variety of manufacturing application programming languages have been found necessary.

– Typically, each manufacturing application programming language has its own unique environment of development methodologies, development, debug and simulation tools, and run-time services. Because these environments are stand-alone, it is difficult to achieve convenient access to the manufacturing databases.

– As a result, it is only with difficulty that an application developer or designer may coordinate the use of differing manufacturing languages for the individual tasks of a complete project, though this is a common need.

– Similarly, it is with great difficulty that systems engineers and integrators combine programs developed using different manufacturing languages, because they use or require different run-time services.

To address these problems, a language-independent manufacturing automation programming environment (MAPLE) is being standardized. This International Standard represents the functional architecture of MAPLE as a first step towards achieving such an environment. MAPLE is a structured set of capabilities that connects the objects such as data used in advanced manufacturing technologies to the required user oriented tools.

--`,,,,`,,`,`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

This International Standard for the MAPLE functional architecture specifies the functionality and interconnection of the components in the environment. It is intended for the technical committees, subcommittees and working groups of those standardization bodies whose mandate will be to develop the standards for the MAPLE services and interfaces, as well as for the commercial developers of MAPLE.

The MAPLE environment can replace existing in-house solutions that have been created by system integrators during the last decade to solve the above mentioned problems.

This support facility for programming will need a set of functionalities that are typical for the manufacturing environment, mainly because of the specific requirements of programming automated devices. Environments to support the programming of other automated devices outside the manufacturing domain (e.g. a transportation system with unmanned trains) will have similar architectures to MAPLE. Nevertheless, the functionality and implementation of the components needed by these other environments will be different from MAPLE.

MAPLE will provide the following benefits, which will lead to considerable time and cost reduction:

- easy and quick development of manufacturing programs;
- easy and quick updating of manufacturing programs;
- unified access for distributed manufacturing databases;
- unified management of manufacturing databases;
- effective utilization of manufacturing software tools;
- provision of a framework for future manufacturing software tools and data models.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Industrial automation systems — Manufacturing Automation Programming Environment (MAPLE) — Functional architecture

1 Scope

This International Standard specifies the functional architecture of MAPLE, a Manufacturing Automation Programming Environment. MAPLE is a common vendor-independent neutral support facility for the programming of multiple manufacturing devices and controls. Thus, MAPLE offers a single environment for the programming of a number of devices and controls not necessarily made by the same company. In addition, MAPLE does not require specifying specific devices and controls at programming time.

To the extent that it is concerned with the programming of manufacturing devices and controls, MAPLE will support the following areas:

– connections between various manufacturing data and manufacturing application programs;

– management of several manufacturing databases;

– sharing of manufacturing application programs and manufacturing software tools.

The scope of this International Standard will be relevant to the following users of the MAPLE standard:

- developers of manufacturing application programs;
- operators editing manufacturing programs;
- engineers who need to refer to manufacturing data.

2 Normative reference

The following standard contains provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/TR 11065:1992, *Industrial automation glossary.*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

3 Definitions

For the purposes of this International Standard, the definitions given in ISO/TR 11065 and the following definitions apply.

3.1 Data Translator: a Manufacturing Software Tool for converting the representation of data.

3.2 Dictionary Manager: a Manufacturing Software Tool to facilitate the manipulation of the Manufacturing Data Dictionary and the Manufacturing Software Tool Dictionary.

3.3 Execution Manager: a software tool that controls the sequence of execution of internal processes of MAPLE and the external Manufacturing Application Programs and Manufacturing Software Tools.

3.4 Manufacturing Application Program: manufacturing software which has command and information connections with MAPLE (e.g. CAD systems), but is not registered in the Manufacturing Software Tools Dictionary.

3.5 Manufacturing Database: a data repository, possibly distributed, containing product, process, facility and management oriented data.

3.6 Manufacturing Data Dictionary: a collection of data schema (meta-data) describing data in the Manufacturing Database.

3.7 Manufacturing Data Manager: a Manufacturing Software Tool enabling access to the Manufacturing Databases.

3.8 Manufacturing Software Tool: a software program registered in the Manufacturing Software Tool Dictionary, which provides simple or complex, manufacturing related functionality for the support of Manufacturing Application Programs. (e.g. simulators, editors).

3.9 Manufacturing Software Tool Dictionary: a collection of descriptions of the capabilities of Manufacturing Software Tools and Manufacturing Application Programs.

3.10 MAPLE: a Manufacturing Automation Programming Environment, a common vendor-independent neutral support facility for the programming of multiple manufacturing devices and controls.

3.11 MAPLE Engine: part of the standardized core of MAPLE, a software tool for receiving and handling requests to MAPLE and the initialization of the entire environment.

3.12 Software Tool Linker: a Manufacturing Software Tool to select and sequence other Manufacturing Software Tools and Manufacturing Application Programs to fulfil MAPLE Engine requests.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

4 MAPLE functional architecture and its components

The MAPLE functional architecture, its components and interfaces between these components, and the interfaces to the outside world are shown in figure 1. To facilitate the description of the components, and (in clause 5) the interfaces, the components and interfaces shown in figure 1 are labelled with the appropriate section numbers describing them. Figure 2 shows the information and control flow between the MAPLE components themselves and the outside world. The following provides more functional detail on the architecture and components of the Manufacturing Automation Programming Environment.

The functional architecture of this environment consists of the the MAPLE Engine, the Manufacturing Data Dictionary, the Manufacturing Software Tool Dictionary, a Dictionary Manager, a Manufacturing Data Manager, an Execution Manager and a Software Tool Linker, which provide standardized functionality and interfaces to the Manufacturing Application Programs, the Data Translator, the Manufacturing Software Tools and the Manufacturing Databases. The user interface is through the Manufacturing Application Programs and the Manufacturing Software Tools.

The MAPLE Engine is the driver of the whole environment. To keep track internally of the variety of types of data being used in the Manufacturing Data Bases, a Manufacturing Data Dictionary, describing standard manufacturing data models, is provided. Similarly, to facilitate the use of Manufacturing Software Tools and Applications connected to MAPLE, especially if a number of such Tools and Applications have to be linked to accomplish a given task, a Manufacturing Software Tool Dictionary, describing the Tools' functionalities as well as their input and output requirements, is provided. The actual data handling is achieved through MAPLE's Manufacturing Data Manager.

4.1 MAPLE Engine

The MAPLE Engine provides the following functionalities:

– allow for the initialization of the entire environment;

– provide the MAPLE Interface between the MAPLE and the outside world;

– accept external requests for services of the environment, related to the Software Tool Linker, Manufacturing Data Manager, Data Translator and Dictionary Manager or any other Software Tool or Application Program connected to MAPLE;

– interpret and process external requests by calling on either the Software Tool Linker or the Execution Manager directly;

– provide status information to the user on request through a Manufacturing Application Program or a Manufacturing Software Tool;

– pass commands coming from the Execution Manager and directed to an external Manufacturing Software Tool or Application Program through the MAPLE Interface;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

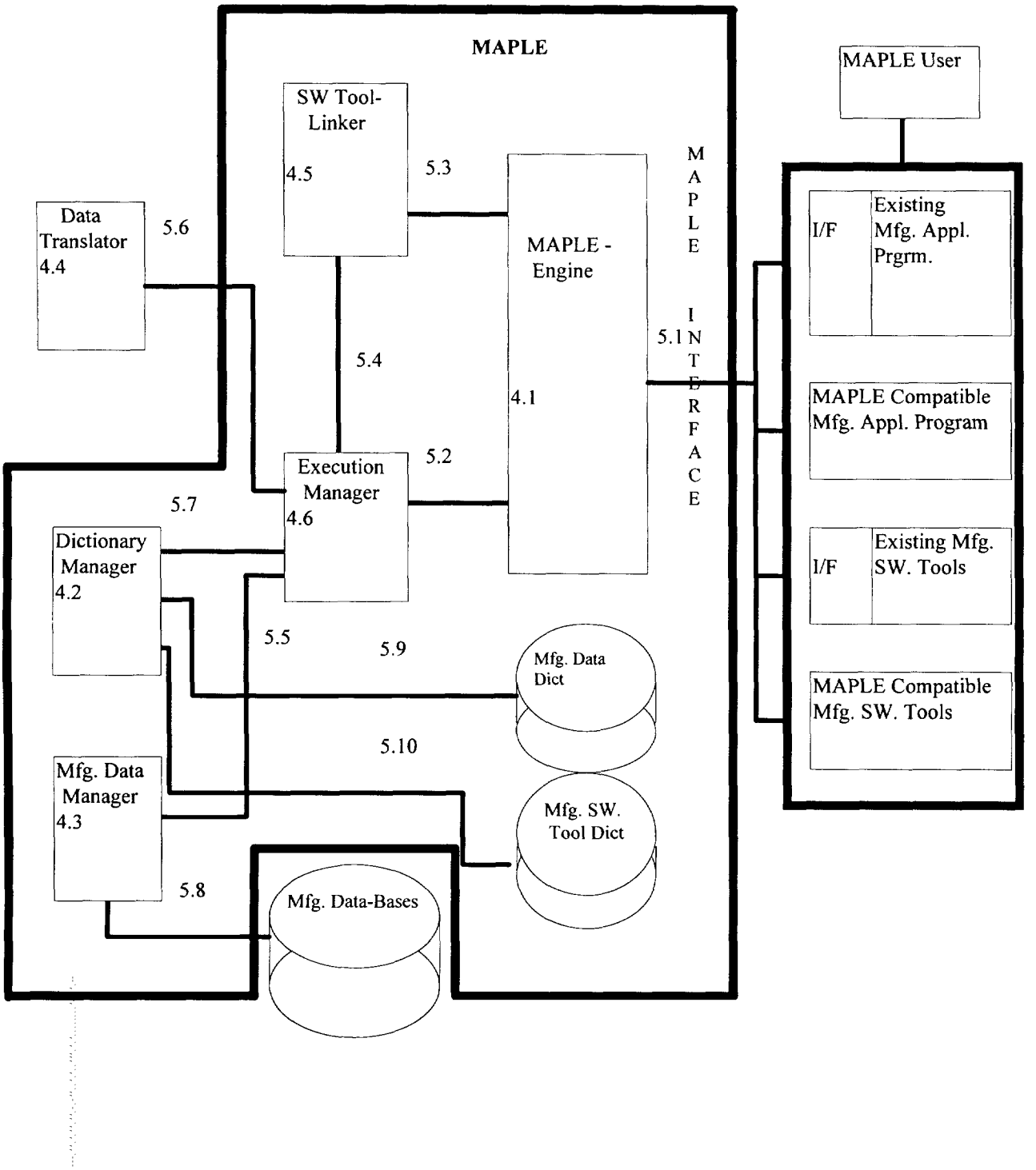

Figure 1 - MAPLE functional architecture and interfaces

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

– pass data between the Execution Manager and external Manufacturing Software Tools or Application Programs through the MAPLE Interface.

The MAPLE Engine is responsible for the handling and the control of external requests to MAPLE. For this purpose the MAPLE Engine provides an interface to all user applications and software tools connected to MAPLE. The external requests to MAPLE are interpreted by the MAPLE Engine, which decides which actions should be taken. If the external request can be handled by one single action of a software tool or application (simple request) then a request for this action is passed to the Execution Manager who will take care of the execution. If the external request involves more complex actions, the MAPLE Engine issues a request to the Software Tool Linker to create a task list of actions that will fulfil the original request. The execution of the actions appearing in the task list is managed by the Execution Manager. In reply to the original request issued to MAPLE, the MAPLE Engine will provide the requested information, data or action as well as status information on the success or failure of handling the particular request made.

4.2 Dictionary Manager

The Dictionary Manager permits manipulation of the Manufacturing Data Dictionary and the Manufacturing Software Tool Dictionary through the provision of the following functions for either one of the two dictionaries:

– adding an entry to the dictionary;

– deleting an entry from the dictionary;

– editing an item in the dictionary;

– searching for and retrieving a dictionary entry.

All activities involving either the Manufacturing Data Dictionary or the Manufacturing Software Tool Dictionary are solely handled by the Dictionary Manager. These activities consist of adding, deleting, editing and search for and retrieving of entries in either one of these two dictionaries. The requests for such activities, including any associated data, come to the Dictionary Manager from the Execution Manager, triggered by requests from a user; the Manufacturing Data Manager, which may require information regarding a Manufacturing Database access; the external Data Translator, which requires format information for one of its input or output files; or the Software Tool Linker, which requires information about the capabilities and needs of Manufacturing Software Tools, and the required formats of inputs and outputs of these tools. All these requests are controlled by the Execution Manager, who is in fact the process invoking the Dictionary Manager and taking care of data passing from the requestor to the Dictionary Manager and vice versa. The interfaces between the Dictionary Manager and the Manufacturing Data Dictionary and the Manufacturing Software Tool Dictionary are described in 5.10.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

4.3 Manufacturing Data Manager

The functionalities of the Manufacturing Data Manager can be summarized as follows:

- access data in the Manufacturing Databases (select, insert, delete, update);
- provide information concerning Manufacturing Databases;
- control user access to the Manufacturing Databases (security);
- control database integrity;
- allow maintenance of the Manufacturing Databases (e.g. create database)

The programming and control of the different component devices of manufacturing systems requires an extensive amount of manufacturing data. This data, including product oriented data, production oriented data, operation oriented data and management oriented data, tends to have a very complex data structure. As a result of this, actual manufacturing databases in most companies tend to have also a complex physical structure (e.g. distributed databases) requiring high level and special technology for the transparent use and management of this data.

The Manufacturing Data Manager provides the means of access to the Manufacturing Databases in response to requests from the Execution Manager. Requests for access to data in the Manufacturing Databases can come from the MAPLE Engine, the Software Tool Linker and the Data Translator. It accesses any required information from the Manufacturing Data Dictionary via a request to the Execution Manager. It includes the regular functions of a Database Manager such as user access control and security as well as maintenance of database integrity. It also provides information concerning the Manufacturing Databases. MAPLE and the applications relying on it, performs all its Manufacturing Database accesses (select, insert, delete, update) through the Manufacturing Data Manager.

4.4 (External) Data Translator

The Data Translator is a software tool external to the MAPLE environment, invoked by the Execution Manager and translates data from one specified data model to another. In order to perform this task, the Data Translator accesses both the input and output data models stored in the Manufacturing Data Dictionary through requests to the Execution Manager.

Typically the Software Tool Linker will have identified the need for translation and will invoke the Data Translator through the Execution Manager to perform a particular data translation task. The formats to be used within the data translation mechanism are held within the Manufacturing Data Dictionary, hence there is a bi-directional communication initiated between the Data Translator and the Dictionary Manager via the Execution Manager, to obtain the necessary format information. This activity is controlled by the Execution Manager. The next stage of the Data Translator phase is to read the data which is to be translated and to write the translated output. These data sets can be residing at intermediate storage spaces controlled by the Execution Manager or can be in the Manufacturing Databases. In the latter case the Manufacturing Data Manager is invoked.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

The method utilized for data translation is considered an implementation issue. The Software Tool Linker passes the two data formats and the Data Translator implementor may select any means to achieve the required translation (i.e. single phase [source → destination], double phase [source → neutral format, neutral format → destination] are valid).

A special case of the Data Translator is a data filter, a Manufacturing Software Tool which performs selective data translation.

4.5 Software Tool Linker

The Software Tool Linker responds to requests from the MAPLE Engine to select and sequence other Manufacturing Software Tools. The Software Tool Linker, a special manufacturing software tool itself, creates one virtual manufacturing software tool from multiple manufacturing software tools. Using this special manufacturing software tool, users can implement 'new' manufacturing software tools, by combining manufacturing software tools which have already been developed and catalogued within the Manufacturing Software Tool Dictionary.

The Software Tool Linker provides the following functionalities:

– interpret MAPLE Engine requests to determine requested functionality;

– examine through the use of the Dictionary Manager, the Manufacturing Software Tool Dictionary to identify the Manufacturing Software Tools required to achieve the requested functionality;

– provide to the Execution Manager a task list of subsequent actions to be taken to achieve the original request of the MAPLE Engine. A task description will typically contain the name of the tool to be used, the commands to be given to this tool, the location of the input data and the location where to provide the output data;

– provide status information on the linking process to the MAPLE Engine.

4.6 Execution Manager

The Execution Manager is responsible for the execution of tasks issued by the MAPLE Engine or task lists issued by the Software Tool Linker. Its functions are :

– accept tasks from the internal MAPLE components or task lists from the Software Tool Linker;

– for each single task requested by an internal MAPLE component issue control commands to the Manufacturing Software Tool selected, in order to achieve the requested functionality;

– for task lists requested by the Software Tool Linker, issue control commands in the proper sequence to the Manufacturing Software Tools selected, in order to achieve the requested functionality. In the case of Software Tools invoked external to MAPLE, the control commands are transferred through the MAPLE Engine and the MAPLE Interface;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

– accept status information from the tools running. In the case of Software Tools external to MAPLE, the status information is transferred through the MAPLE Interface and the MAPLE Engine;

– provide status information to the requesting MAPLE component or to the Software Tool Linker concerning the execution of respectively a task or a task list;

– set aside any intermediate storage space required for data to be exchanged between successive Manufacturing Software Tools. Intermediate storage for data and associated data models can be assigned either internally to the Execution Manager, or in the Manufacturing Databases and Manufacturing Data Dictionary, respectively;

– manage the data flows between Manufacturing Software Tools and intermediate storage space provided by the Execution Manager. In the case of Software Tools external to MAPLE, the data is transferred through the MAPLE Engine and the MAPLE Interface;

– keep track of the stack of tasks to be executed, following an original external request. Note that the original task can lead to several sub-tasks, to be executed before the original task can be completed. For example, a single task to call the Data Translator can lead to several sub-tasks issued by the Data Translator to receive information from the Dictionary Manager.

4.7 Manufacturing Data Dictionary

The Manufacturing Data Dictionary is basically a facility (database) to store a machine readable collection of descriptions of data schema. A description is provided for data in the Manufacturing Databases, as well as for temporary data that the Execution Manager stores as intermediate data between a set of Manufacturing Software Tools that are concatenated to achieve a desired functionality. The content of the Manufacturing Data Dictionary provides instructions on how to read and interpret the data it describes. An entry in this dictionary has two main aspects: format of the data and meaning of the data. The format of the data is information that is ultimately needed by the Manufacturing Data Manager, the meaning of the data is information needed by the Data Translator, the Software Tool Linker and other Manufacturing Software Tools and Manufacturing Application Programs. The Manufacturing Data Dictionary is a database with a main table, in which each record represents one manufacturing data entity, and other related tables needed to describe table structures and relations between tables of the manufacturing database.

Manipulation of the Manufacturing Data Dictionary is achieved through the interface with the Dictionary Manager. Essentially the Dictionary Manager reads, writes or edits the dictionary as required, to achieve the goal of add, delete, edit or search and retrieve commands. The response of the Manufacturing Data Dictionary consists of the requested dictionary content information, as well as any status information on the dictionary manipulation.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

4.8 Manufacturing Software Tool Dictionary

The Manufacturing Software Tool Dictionary provides a facility for storing a machine readable collection of Manufacturing Software Tool identifications (filenames) and descriptions of the functionality as well as the input and output requirements of the Manufacturing Software Tools connected to MAPLE. The principal use of the Manufacturing Software Tool Dictionary is by the Software Tool Linker via the Dictionary Manager. The Manufacturing Software Tool Dictionary is examined according to a particular need for information about a Manufacturing Software Tool to be employed by the Software Tool Linker. The computer readable description of functionalities is specified and the input manufacturing data and output descriptions of each software tool and each Manufacturing Application Program is referenced to entries in the dictionary. The entries will include software tools provided by MAPLE itself, such as the Dictionary Manager, the Data Translator and the Software Tool Linker.

Maintenance and updating of the Manufacturing Software Tool Dictionary is achieved through manipulations of the Dictionary Manager. The Dictionary Manager, in response to requests for adding, deleting, editing or searching and retrieving of a particular entry in the Manufacturing Software Tool Dictionary will read or write relevant entries in the dictionary. The Manufacturing Software Tool Dictionary responds with the requested information as well as status information.

For effective use of MAPLE, all application programs and software tools used in a specific manufacturing system have to be registered in the Manufacturing Software Tool Dictionary through an external request from the MAPLE user or an appropriate application program. MAPLE software and other MAPLE components such as the Dictionary Manager, Software Tool Linker, Execution Manager, Translator and Manufacturing Data Manager are also registered. This allows, for example, the Dictionary Manager to be called from a user supplied application program.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

5 Interfaces

NOTE – Clause 5 and all its subclauses are informative only, in order not to restrict the development of subsequent International Standards in this series.

The following describes the interfaces between the components of MAPLE and the Manufacturing Databases, Manufacturing Application Programs and Manufacturing Software Tools. Figure 1 shows these interfaces with labels referring to sub-clauses in this clause. Figure 2 provides detail of control and data flow through these interfaces.

5.1 MAPLE interface

This interface is one of the three interfaces between MAPLE and its surrounding. It provides the interface to Manufacturing Software Tools, Manufacturing Application Programs, and MAPLE Users who gain access to MAPLE via an appropriate Manufacturing Application Program.

Already existing Manufacturing Software Tools and Manufacturing Application Programs require an interface layer to enable them to communicate successfully with the MAPLE Engine via the standardized MAPLE interface 5.1. Any new Manufacturing Software Tools or Manufacturing Application Programs being developed can be made MAPLE compatible, through conformance to the MAPLE Interface standard, and thus will not require a separate interface layer.

5.1.1 Control between MAPLE Engine and Manufacturing Application Programs and Manufacturing Software Tools

The control consists of the following:

a) requests made by a Manufacturing Application Program for the initialization of the entire environment;

b) reply by MAPLE Engine on status of initialization request;

c) requests made by Manufacturing Application Programs, Manufacturing Software Tools or by the MAPLE User, via a Manufacturing Application Program to the MAPLE Engine for the provision of a requested functionality that can be provided by:

1) the Software Tool Linker;
2) the Manufacturing Database Manager;
3) the Dictionary Manager;

d) the MAPLE Engine returns status information on the requests received;

e) program control commands issued by the MAPLE Engine to Manufacturing Software Tools and Manufacturing Application Programs, consisting of:

1) initialize;
2) idle;
3) initiate;
4) run;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

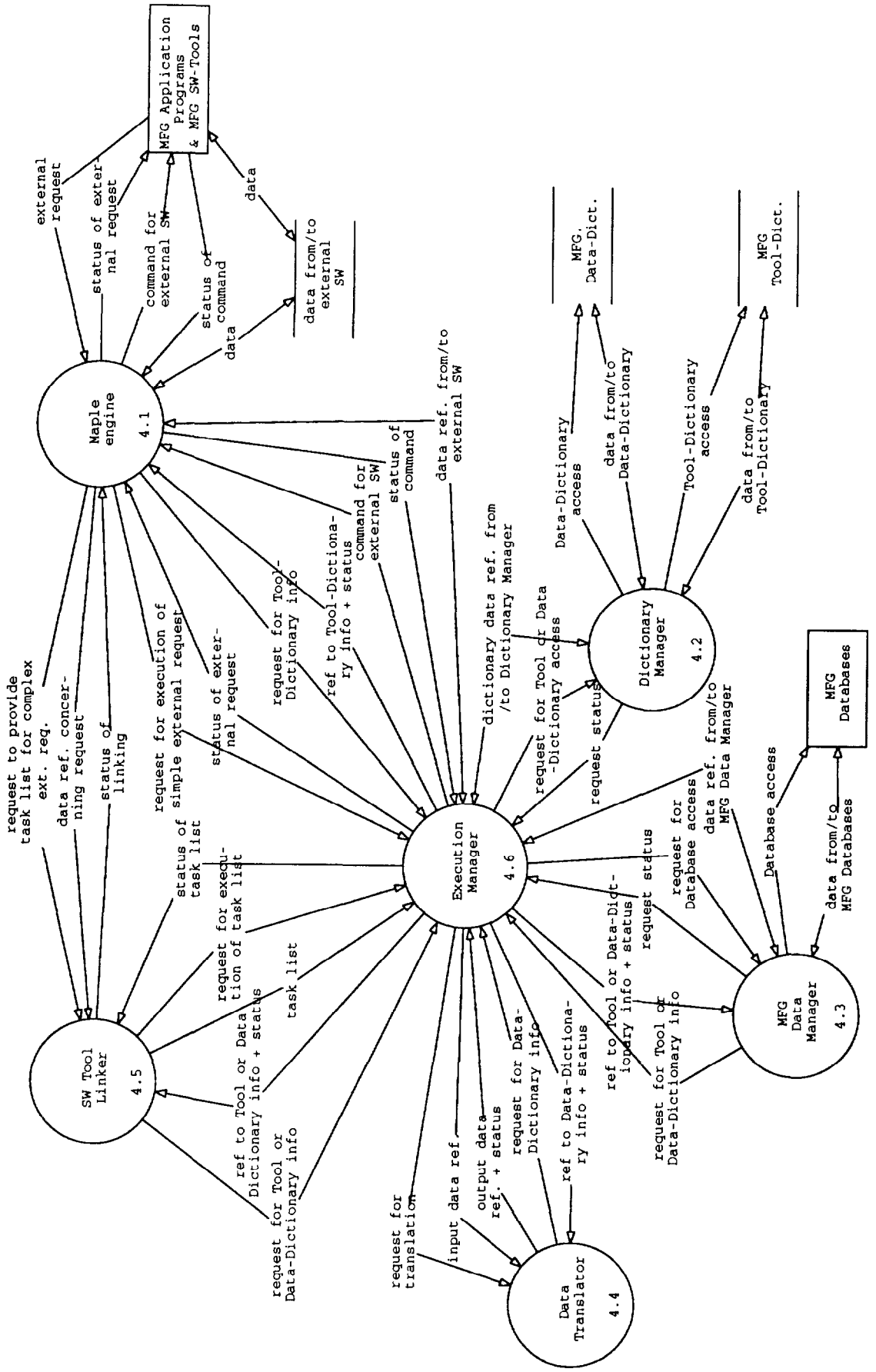

Figure 2 - Control and data flow

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

5) complete;
6) terminate;

f) information to the MAPLE Engine on status of requested program control.

5.1.2 Data transfer from MAPLE Engine to Manufacturing Application Programs and Manufacturing Software Tools

Data originating from:

- the Manufacturing Data-Bases;
- the Data Translator;
- the Manufacturing Data Dictionary;
- the Manufacturing Software Tool Dictionary;
- temporary data storage provided by the Execution Manager;
- other Manufacturing Software Tools and Manufacturing Application Programs.

5.1.3 Data transfer from Manufacturing Application Programs and Manufacturing Software Tools to MAPLE Engine

The data transferred consists of:

- data destined for the Manufacturing Data-Bases;
- data destined to the Data Translator;
- data for updating the Manufacturing Data Dictionary;
- data for updating the Manufacturing Software Tool Dictionary;
- data destined for temporary data storage provided by the Execution Manager;
- data to be sent to other Manufacturing Software Tools and Manufacturing Application Programs.

5.2 MAPLE Engine - Execution Manager interface

5.2.1 Control between MAPLE Engine and Execution Manager

The control consists of:

a) the MAPLE Engine passes on requests to the Execution Manager for functionalities to be provided by:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

1) the Manufacturing Data Manager;
2) the Data Translator;
3) the Dictionary Manager;
4) the Execution Manager, specifically, to store or retrieve data from temporary storage;

b) the Execution Manager replies with status information on these requests;

c) the Execution Manager sends program control commands for Manufacturing Application Programs and Manufacturing Software Tools, consisting of:

1) initialize;
2) idle;
3) initiate;
4) run;
5) complete;
6) terminate;

d) the MAPLE Engine replies with status information on these program control commands.

5.2.2 Data transfer from MAPLE Engine to Execution Manager

Data references are passed for:

- data destined for the Manufacturing Data-Bases;
- data destined to the Data Translator;
- data for updating the Manufacturing Data Dictionary;
- data for updating the Manufacturing Software Tool Dictionary;
- data destined for temporary data storage provided by the Execution Manager.

5.2.3 Data transfer from Execution Manager to MAPLE Engine

Data references are passed for:

- data originating from the Manufacturing Data-Bases;
- data output originating from the Data Translator;
- data originating from the Manufacturing Data Dictionary;
- data originating from the Manufacturing Software Tool Dictionary;
- data originating from temporary data storage provided by the Execution Manager.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

5.3 MAPLE Engine - Software Tool Linker interface

5.3.1 Control between MAPLE Engine and Software Tool Linker

This control consists of the following:

– the MAPLE Engine passes on requests to the Software Tool Linker for functionalities that are non-trivial, that is they require a combination of more than one software tool to be fulfilled;

– the Software Tool Linker replies with status information on these requests.

5.3.2 Data transfer from MAPLE Engine to Software Tool Linker

Meta data and/or pointers to data are required for fulfilling the request made.

5.3.3 Data transfer from Software Tool Linker to MAPLE Engine

None.

5.4 Execution Manager - Software Tool Linker interface

5.4.1 Control between Execution Manager and Software Tool Linker

The control consists of:

a) requests to the Execution Manager for:

   1) Manufacturing Data Dictionary information;
   2) Manufacturing Software Tool Dictionary information;

b) reply by the Execution Manager on the status of the requests made;

c) request by Software Tool Linker to execute a sequence of tools;

d) reply by Execution Manager on status of linkage request.

5.4.2 Data transfer from Execution Manager to Software Tool Linker

Data references are passed for:

– content of Manufacturing Data Dictionary entry requested;

– content of Manufacturing Software Tool Dictionary requested.

5.4.3 Data transfer from Software Tool Linker to Execution Manager

This data consists of a sequence of software tools to be used, their location, and location of input and output file names and locations for each of these tools.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

5.5 Execution Manager - Manufacturing Data Manager interface

5.5.1 Control between Execution Manager and Manufacturing Data Manager

This control consists of the following:

– request to the Manufacturing Data Manager to create a temporary data file of given name, size and format to be used as data buffer;

– reply by the Manufacturing Data Manager on the status of this request;

– request to the Manufacturing Data Manager to store or retrieve a specified data file of given name, size and format;

– reply by the Manufacturing Data Manager on the status of this store or retrieve request;

– request to the Execution Manager for information from the Dictionary Manager;

– reply by the Execution Manager on the status of the request to the Dictionary Manager.

5.5.2 Data transfer from Execution Manager to Manufacturing Data Manager

Data references are passed for:

– data to be stored into Manufacturing Data Bases;

– data obtained from the Dictionary Manager.

5.5.3 Data transfer from Manufacturing Data Manager to Execution Manager

Data references are passed for data being retrieved from Manufacturing Data Bases.

5.6 Execution Manager - Data Translator interface

This external interface links the external Data Translator to MAPLE.

5.6.1 Control between Execution Manager and Data Translator

The control consists of the following:

– request to the Data Translator for translation of a data file;

– status information from the Data Translator on the translation process;

– request to the Execution Manager for data models from the Dictionary Manager;

– status information from the Execution Manager on the data model request.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

5.6.2 Data transfer from Execution Manager to Data Translator

Data references are passed for:

- input and output formats of data to be translated;
- data to be translated;
- data models for input and output data.

5.6.3 Data Transfer from Data Translator to Execution Manager

Data references are passed for the translated data.

5.7 Execution Manager - Dictionary Manager interface

5.7.1 Control between Execution Manager and Dictionary Manager

The control consists of the following:

a) request to the Dictionary manager for one of the following:
   1) initialize the dictionaries;
   2) add an entry to the dictionary;
   3) delete an entry from the dictionary;
   4) edit an item in the dictionary;
   5) search for and retrieve an entry;

b) the Dictionary Manager responds with status information on the request.

5.7.2 Data transfer from Execution Manager to Dictionary Manager

Data references are passed for the dictionary data to be stored.

5.7.3 Data transfer from Dictionary Manager to Execution Manager

Data references are passed for the dictionary data retrieved.

5.8 Manufacturing Data Manager - Manufacturing Data-Base interface

This interface establishes a link between MAPLE and the external Manufacturing Data-Base.

5.8.1 Control between Manufacturing Data Manager and Manufacturing Data-Bases

The control consists of the following:

a) request to one of the Manufacturing Data Bases for one of the following:

   1) initialize the database;
   2) read a data base entry;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

3) write a data base entry;
4) delete a data base entry;

b) reply from the Manufacturing Data Base with status information on the request.

5.8.2 Data transfer from Manufacturing Data Manager to Manufacturing Data-Bases

This is the data to be stored.

5.8.3 Data transfer from Manufacturing Data-Bases to Manufacturing Data Manager

This is the data being retrieved.

5.9 Dictionary Manager - Manufacturing Data Dictionary interface

5.9.1 Access from Dictionary Manager to Manufacturing Data Dictionary

This access handles the following:

a) request to the Manufacturing Data Dictionary for one of the following:

1) initialize the dictionary;
2) add an entry;
3) copy an entry;
4) read an entry;
5) delete an entry;

b) reply from the Manufacturing Data Dictionary with status information on the request.

5.9.2 Data transfer from Dictionary Manager to Manufacturing Data Dictionary

This is the data to be stored.

5.9.3 Data transfer from Manufacturing Data Dictionary to Dictionary Manager

This is the data being retrieved.

5.10 Dictionary Manager - Manufacturing Software Tool Dictionary interface

5.10.1 Access from Dictionary Manager to Manufacturing Software Tool Dictionary

This access handles the following:

a) request to the Manufacturing Software Tool Dictionary for one of the following:
1) initialize the dictionary;
2) add an entry;
3) copy an entry;
4) read an entry;
5) delete an entry;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

b) reply from the Manufacturing Software Tool Dictionary with status information on the request.

5.10.2 Data transfer from Dictionary Manager to Manufacturing Software Tool Dictionary

This is the data to be stored.

5.10.3 Data transfer from Manufacturing Software Tool Dictionary to Dictionary Manager

This is the data being retrieved.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Annex A
(informative)

How MAPLE can succeed

This International Standard is the first of a series of MAPLE standards. The interfaces between the MAPLE and the outside, will be specified in detail in future parts of the MAPLE standard series. Only the services, not implementation details, will be specified in these future parts.

The successful introduction of the MAPLE standard will also depend highly on standardisation activities in the areas of product oriented data models (e.g. STEP), process oriented data models, management oriented data models and physical resource oriented data models.

Because the future parts of the MAPLE standard will focus only on the services provided by MAPLE and the interfaces to MAPLE, software vendors developing MAPLEs have freedom in developing the MAPLE components to meet their own customers' needs.

Software vendors developing the manufacturing application programs and software tools meant to interact with MAPLE (e.g., editors, debug and simulation tools) have to provide a MAPLE-compatible interface.

To prove and promote the feasibility of MAPLE, a prototype should be developed. Concurrently, software suppliers and system integrators should be motivated to develop the different MAPLE components and developers of software tools and manufacturing application programs should be motivated to provide an interface to MAPLE.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Annex B
(informative)

Simple examples

The following examples illustrate how MAPLE will function in typical manufacturing applications. As will be seen, simple requests to the MAPLE Engine such as requests for dictionary information or for data files in the format they are stored in, are passed on directly to the Execution Manager, All other requests are passed on to the Software Tool Linker.

B.1 Request for dictionary information

In this example, a manufacturing application program requests information from one of the two dictionaries. This request is made via interface 5.1 to the MAPLE Engine, which interprets the request, and passes it on via 5.2 to the Execution Manager and 5.7 to the Dictionary Manager. The Dictionary Manager then searches the appropriate dictionary for the entry requested, and retrieves the entry from the dictionary via interface 5.9 or 5.10. If the requested entry does not exist, an appropriate error message created by the Dictionary Manager will be relayed via the Execution Manager and the MAPLE Engine to the originating manufacturing application program, and the request terminated. Otherwise, a reference to the content from the dictionary entry will be passed from the Dictionary Manager back via the Execution Manager to the MAPLE Engine, which will then make the data available to the manufacturing application program that made the original request.

B.2 Request for data in existing format

In this example, a manufacturing application program requests data from a file FILE, with no format specified, meaning that the data is to be retrieved in the format it is stored in. Again, this request is made via interface 5.1 to the MAPLE Engine, which interprets the request, and passes it on to the Manufacturing Data Manager via the Execution Manager via interfaces 5.2 and 5.5. The Manufacturing Data Manager requests and receives data dictionary information about this file from the Dictionary Manager via the Execution Manager, using interfaces 5.5, 5.7 and 5.9. Using this information, the Manufacturing Data Manager then searches the appropriate Manufacturing Data Base for the requested file via interface 5.8, retrieves the content of the file and passes a reference to its location to the MAPLE Engine via interfaces 5.5 and 5.2. The MAPLE Engine will then obtain this data and send to the originating manufacturing application program. Any errors occurring anywhere during the entire transaction, be it during the dictionary interrogation or the data file retrieval, would be indicated by appropriate error messages generated and passed back to the requestor.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

B.3 Request for data in a specified format

This example is similar to that of B-2, except that the data is requested in a specified format, which, presumably, is different from the format it is stored in. A manufacturing application program issues a request to MAPLE to provide the content of data file FILE in format FORMR. The application program communicates this request via interface 5.1 to the MAPLE Engine. The MAPLE Engine interprets this request and, recognizing that this request is neither a request for dictionary information nor a straight database retrieval, it passes this request on to the Software Tool Linker via interface 5.3. Having received the goal, that is file FILE in format FORMR, the Linker now tries to establish the starting position, that is, where is FILE located, and what format is it in? Thus the software Tool Linker via the Execution Manager and interfaces 5.4 and 5.7 requests the Dictionary Manager for Manufacturing Data Dictionary information on file FILE. The reply will consist of the location of FILE as well as its format FORMA.

If FORMA is the same as FORMR, there is no need for the Software Tool Linker, since the request can be handled as in example B-2. The Linker therefore informs the MAPLE Engine via interface 5.3 that this request should be handled as a request for data in existing format. The MAPLE Engine now handles the task as in B-2.

If FORMA is different from FORMR, the Linker has to find an appropriate tool to convert from FORMA to FORMR. It will therefore request information from the Manufacturing Software Tool Dictionary via the services of the Execution Manager and the Dictionary Manager (interfaces 5.4, 5.7 and 5.10) on software tools that can translate either directly from FORMA to FORMR, or in general, find a chain of tools that will allow data translation from FORMA to FORMR. The reply informs of the availability and functionality of the external Data Translator. The Linker now passes a command and a reference to the input data via 5.4 to the Execution Manager and hence to the Data Translator to retrieve the desired data from the Manufacturing Data Base and translate it using the specified translator. The Execution Manager now issues a sequence of commands as follows:

a) Via interface 5.5 request the Manufacturing Data Manager to retrieve FILE in format FORMA;

b) Via interfaces 5.5 and 5.6 request reference to FILE to be sent from the Data Manager to the Translator, and for the Translator to receive FILE in format FORMA;

c) Via 5.6 request the Translator to translate FILE from FORMA to FORMR. In order to execute this command, the Data Translator will need the data models for FORMA and FORMR. It obtains these through a request to the Dictionary Manager via interfaces 5.6 and 5.7. The data models obtained from the Manufacturing Data Dictionary will be returned via interfaces 5.9, 5.7 and 5.6;

d) Receive a reference to the translated data through 5.6 and pass it on via interface 5.2 to the MAPLE Engine;

e) Via 5.2 inform the MAPLE Engine that the task has been completed.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

It now remains only for the MAPLE Engine to retrieve the translated data from a location specified in the data reference, and pass it to the originating manufacturing application program via interface 5.1. Of course, in case of any error occurring, an appropriate error message will been sent through appropriate interfaces back to the MAPLE Engine and from there to the requestor.

B.4 Request for a simulation capability

In this example a manufacturing application program has generated a sequence of robot motions which should be simulated, and the results displayed visually on a specified display device. Assume that this type of application has been used many times in the past, and hence the data schema of the output of the application program that is to be sent to the simulator has already been documented in the Manufacturing Data Dictionary.

The manufacturing application program issues a request to MAPLE via interface 5.1 to simulate the robot motions described in file FILE to be supplied by the application program, and which is in FORMA. The results are to be in the form of a visual display of the robot kinematics. The MAPLE Engine forwards this request to the Software Tool Linker via interface 5.3. The Linker in turn consults the Manufacturing Software Tool Dictionary via the services of the Execution Manager and the Dictionary Manager (5.4, 5.7 and 5.10), to find a suitable software tool to provide the kinematics display. In this case, reference to a display software tool is found. This tool resides outside MAPLE as one of the available Manufacturing Software Tools. The dictionary also indicates that the display tool requires its input in format FORMA.
Seeing that the available data is in the same format, no data translation is required, making the Linker s task straight forward. The Software Tool Linker informs the Execution Manager through 5.4 to send the robot motion output file to the simulator. The Execution Manager now proceeds as follows.

a) Through interface 5.2 and 5.1 the display tool will be initialized;

b) Through interface 5.2 to the MAPLE Engine and interface 5.1 a message is sent to the requesting manufacturing application program, indicating the location of the display tool, and requesting it to send a reference to the display data to the display tool.

c) A "Run command is sent via interface 5.2 and 5.1 to the display tool. On completion, the display tool signals the Execution Manager via 5.1 and 5.2;

d) A "Complete message is sent via 5.2 and 5.1 to the requesting manufacturing application program.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Annex C
(informative)

Case studies

Two case studies are provided. The first illustrates how the data management subsystem could be replaced by MAPLE s MAPLE Engine, the Dictionary Manager and the Manufacturing Data Manager, and how some of the functionalities of the Software Tool Linker and the Execution Manager could be implemented. The second case study illustrates the benefits to be derived if an environment like MAPLE were available to help in integration of manufacturing software.

C.1 An integrated environment for factory automation

The Flexible Manufacturing System (FMS) is an important component in today's manufacturing plants. It typically consists of a number of manufacturing work-cells each containing an intelligent, autonomous system controller and a variety of machine tools, storage systems, and flexible tool and work-piece transportation systems.

The control software for such an FMS is becoming larger and much more complex as manufacturing capability for new products is added to the existing production line, or the existing control software modules are modified to implement new functions to be performed for the additional products. The current software system implementation has become too ad-hoc to make it simple and easy to maintain.

In this case study it is assumed that a manufacturing work-cell consists of devices such as NC machines, robots, work-piece conveyers, storage systems, operator panels, sensors, actuators and so on, as well as a work-cell controller system. The work-cell also has communication channels to exchange information with components outside of the work-cell, such as an area control computers, other work-cell controller systems in other work-cells, and manufacturing line operators.

The work-cell will have functions for manufacturing planning, device selection, giving of directions to the operator, machine control, data acquisition and analysis, process monitoring and communication.

The manufacturing work-cells are connected by some transportation and communication systems. Production data and work-pieces are pooled somewhere, and are distributed when the demand requests come from the work-cells. The results of the work-cell process are returned and pooled for use in subsequent requests.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Thus the work-cell control and monitoring software systems require: a work-cell model database which contains information about the abilities, characteristics, configurations and operations of the machine tools in the work-cell, as well as about the whole manufacturing work-cell itself; an autonomous task selection capability, based on information obtained from the work-cell model database; a task decomposition capability to divide a selected task into simpler tasks, and then schedule those sub-tasks using precedence relations; device control data generation which prepares device control data or programs for the virtual manufacturing devices (VMD's); dynamic task dispatching which balances the load of each device within a work-cell while keeping the utilization rate of those devices high; work-cell monitoring, that is continuously checking the operating status of the machine tools and taking proper actions for unexpected events; and information update capabilities.

C.1.1 The data-pool system

The data-pool system described here represents a proprietary solution that has been developed to support application programs by means of data sharing, event notification and condition monitoring [Takata, et. al. 1990] [Takata 1993, 1992]. The system uses inter-process communication over the network, and hence can play the roles of the infrastructure of an autonomous distributed work-cell control systems. Even in a single work-cell system, the data-pool system is effective to keep application programs simple and highly independent from other program modules.

The data-pool system design is based on the client-server model of transaction processing, and basically consists of one server and as many clients as the user wishes. The data-pool server system works as a process running on the work-cell controller operating system, and application programs including the data-pool clients are operated as other processes on the same OS. Those client processes communicate with the data-pool server process, using the inter-process communication vehicle provided by the work-cell controller operating system.

The data-pool server consists of two major parts; the communication interface subsystem and the data management subsystem. The communication interface exchanges information with other processes in the same work-cell control system, or the processes running on the remote work-cell controllers connected with the network system.

The data management subsystem also consists of two parts; the data change monitor subsystem and the data storage subsystem. The data storage subsystem manages the association between the object name and its data provided by the client. The access and / or update requests from the clients are received and processed by the communication interface subsystem, and the data retrieved from the data storage subsystem are manipulated according to the received requests.

If the data-pool server has multiple clients, all those clients can share the data in the data-pool system by means of communicating with the same data-pool server. When some data in the data storage subsystem are changed by a request from one of the clients, the data change monitor subsystem transmits the notification messages to the clients registered in advance. Furthermore, relational expressions representing some conditions can be registered within the data-pool server, causing the data change monitor subsystem to send notification messages when the conditions are fulfilled. Using these feature, the data-pool clients need not poll the status of data or relations of interest.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

C.1.2 Main features

Some of the main features of the data-pool system are now discussed.

C.1.2.1 Data sharing

One of the primary functions of the data-pool system is data sharing. As it is very important to share data within multiple application processes, the data-pool system is prepared as a separate process and connected with the communication channel, in order to provide common data for those related processes. As there are some data which are strongly related to each other, those values should be updated simultaneously to keep the whole data system consistent. In the data-pool system, there are many application program interfaces (API's) used to change values of multiple object names within a single transaction.

C.1.2.2 Data change notification

In order to eliminate the need for data polling, and to simplify the logic and appearances of the application programs, the data change notification feature is used. Clients, which want to receive a change notification when the value of an object name is changed, can register the name of the client itself for the object name of interest. As the clients may register names of other clients as the notification destinations, the user can implement a kind of dispatcher which dispatches the events of the data changes to some processes. Furthermore, each application program can be coded such that the program starts processing only when the notification messages is received.

C.1.2.3 Automatic update of dependent data and condition monitoring

Some clients may need to know when the value of a certain object name has a certain constant value, or the value satisfies a certain condition. Using such feature, some application programs can start their processing only when the object name of interest has a certain value.

C.1.2.4 A message routing subsystem

The data-pool system can be used as a message routing subsystem in the application programs. This is a very important feature and the first step towards process abstraction, in that a message can be sent without having exact information about the message receiver process.

C.1.2.5 Automatic process invocation

This feature can be used when a message has to be sent to an application program that is not yet running.

C.1.3 Support utilities

Some application programs are supplied as utility programs of the data-pool system. These clients have very universal functionalities and adapt various types of the user applications. Using the data-pool system, users can easily build a large library of utility programs and integrate them with minimum overhead. The following are some such utilities.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

C.1.3.1 Independent requestor

This utility lets the application programmers and the system integrators access the data-pool server from the operating system command line.

C.1.3.2 Data-pool display subsystem

This display system has a character mapped display screen, and can display the values of the object names stored in the data-pool systems on its screen.

C.1.3.3 Data-pool inspector subsystem

The data-pool inspector is a tool for retrieving and updating the values and attributes of object names in the single data-pool server.

C.1.4 Application programming paradigm

Application program modules follow a programming paradigm, in order to integrate application programs systematically using the data-pool system. This is implemented using the following.

C.1.4.1 Application program control

Each application program progresses through a number of phases in a given order. These are:

- initialization to start up the program;
- idling, waiting for a start notification from the data pool;
- initiating, to secure necessary systems resources prior to start;
- running to carry out the task;
- completing, to release systems resources, and finally;
- terminating, an optional phase to terminate the execution of the application program.

Storing and using status indicators for application programs allows application program modules to be chained or executed concurrently.

C.1.4.2 Sensor inputs and actuator outputs

The read-outs of various sensors can be kept within the data-pool system. Programs can access devices attached to the work-cell through the data-pool system, instead of accessing the devices directly.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

C.1.4.3 Application program interface

This interface for the data-pool system is provided as a function library package. Thus, programmers need not know about the inter-process communication nor the network communication programming. Three interface routines are provided for: Network control and message processing; data cell structure handling; and data-pool accessing.

C.1.4.4 Message protocol

All the operations relating to the data-pool system are requested by clients using transaction messages.

C.1.5 Language processor

A factory automation processing language (FAPL) can be constructed and processed by an object oriented language processor system [Goldberg and Robson 1983], implemented as an interpreter system. The language specification of FAPL is designed to rely on the data-pool system for global data management, as well as inter-interpreter communication, process synchronization, mutual execution and condition monitoring. In other words, the FAPL language provides the application programmer view of the data-pool system.

Each process in an application program is executed by respective language interpreter processes. As the process of the FAPL language interpreter can be started from the data-pool system, the response to a certain condition can be composed in this programming language. Thus, if a condition is met, the data-pool server can invoke the language interpreter process and send a message to the interpreter process just created. While an object oriented implementation is not mandatory, and in fact. may introduce a huge execution time overhead in a message based execution mode, it is still a very effective means for achieving data encapsulation and information hiding, and especially for the control of virtual manufacturing devices.

C.1.6 Relationship to MAPLE

The data-pool system described represents a proprietary solution to a need, that MAPLE, when implemented, could satisfy in large parts in a standard manner. For instance, the two parts of the data-pool server, that is the communications interface and data management subsystem can be replaced by MAPLE's MAPLE Engine, the Dictionary Manager and the Manufacturing Data Manager. MAPLE will certainly support data sharing. Other features such as data change notification, automatic update of dependent data and condition monitoring, message routing and automatic process invocation could be incorporated in the functionalities of the Manufacturing Data Manager, the Execution Manager, the Software Tool Linker or the MAPLE Engine, as appropriate. Some of the functionalities provided by data-pool s support utilities, such as Application Program Control and Application Program Interface would be provided by MAPLE s Execution Manager and the MAPLE Interface.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

C.1.7 Bibliography

Goldberg, A. and Robson, D. (1983) "Smalltalk-80: The Language and Its Implementation," Addison Wesley.

Takata, M. (1992) "FA Software Development on Object Oriented Paradigm," Journal of the Japan Society of Precision Engineering, vol. 58, no. 10, pp. 1649-1651 (in Japanese).

Takata, M. (1993) A Programming Environment for Factory Automation , Proc. of the MAPLE 93 Symposium on Manufacturing Automation Programming Environments, Oct. 4-5, 1993, Ottawa, Canada, pp. 215-224.

Takata, M., Hasegawa, M. and Matsuka, H. (1990) " A Unified Environment for Production Operation," Proc. of Symp. on Manufacturing Application Programming Language Environments, Ottawa, Canada, May 14-15, 1990, pp. 85-94.

C.2 Industrial case study – Manufacture of printing machines

This test-case is based on a real installation in a medium-sized Swiss company, manufacturing printing machines. The system is fairly typical of modern CIM investment. For this system some components such as the CAD system, the order processing system, part of the NC-programming and most of the NC-machines already existed.

The new investment, the production planning system, the manufacturing data system with DNC, the database for supporting the operation, time control, the collection of production-data and two flexible manufacturing cells had to be integrated with the existing system parts. A new shop network based on Ethernet, was installed together with the new components. The customised Software-Development needed for integration was relatively high, as most of the components are vendor specific solutions, a typical situation in AMT,.

In spite of the consequent usage of widely accepted platforms and means of communication (defacto standards), many specific interfaces, protocols and post processors were needed.

It can clearly be seen that a Manufacturing Application Environment with suitable interfaces and tools would bring substantial benefits and cost savings, not only in the purchasing and commissioning phases, but also for future operating and training expansions.

In this study the CAD and CAM Systems share a common database for the working materials or operating means. All manufacturing data is distributed, collected and accessed through a DNC-Server via a local group servers or PC-Workstations. CNC and tool data is exchanged over CNC-modules. For machine-site programming, storing and accessing of programs is made directly from the CNC control. There are links to the database, a special link for quality control, and a provision for collecting machine data in categories such as run-rime, idle-time, setup, repairs etc. Production data is linked via a DNC-server to the Production Planning System. A time control and safety access system works with separate terminals. This part is not directly integrated into the manufacturing data and communication system and is therefore not part of the test-case.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Through the shop's network the group servers, time collecting terminals and fibre optical star couplers are directly connected to an Ethernet backbone. Older CNC machines are connected to a group server via serial lines (RS 232). Flexible manufacturing systems FMS1 and FMS2 with their own specialised server, a robot, the tool setup and the control unit of a floor transport system are connected with Thinwire Ethernet branches. Newer and important NC- machines each have on-site PC-Workstations. All these workstations are connected via fibre optical cables to two star couplers on the Ethernet backbone.

The basic protocol of the entire network is TCP/IP + NFT. It is used for file transfer between the servers and the PC workstations. All control functions, commands and message transfers, are effective only between the group servers of the PC workstations and the control equipment of the NC machines or the other workshop devices. In most of these applications there was a need for customised software.

From the 18 NC machines considered in this test-case, only 5 are identical in type with respect to data formats and communication interfaces (same manufacturer). Therefore, 14 post processors and communication link adaptations were needed. Such customised software was also needed for the robot, the FMS, the tool setup and the transport system.

As shown in Table C - 1 below, considerable savings would be achieved in such a system, if the required interfaces to tools and data were available in an environment described by the MAPLE functional architecture, and the needed tools were available to fit into this architecture. Thus the relational data management needed, could be realised on the base structure provided by MAPLE and only the individual relationships and specialities need be customised.

The operating database is a very special case, due to the multiple accesses needed by the many departments and hence system components. The savings shown would only be possible if the vendors of such systems make the necessary adaptations. The real-time tool data exchanges, including actual dimensions and cutting times, are a more general problem, which should be solved with a MAPLE tool or a standardised procedure.

Links to Production Planning, CAD and CAM systems should make use of the Manufacturing Data Manager to access the corresponding databases.

It remains an open question if the adaptation of operator interfaces by suitable application programs, could reduce the needed custom engineering. This is not taken into consideration in this test case. No savings are possible for communication protocols needed. Here a new approach in industrial communication or a much wider implementation of MMS by vendors could bring progress.

A considerable high cost component are the post-processors for the individual NC-data formats. As post processors are rewritten often by different vendors, a general available library of post-processors within MAPLE would be of big help and result in considerable savings.

Finally, commissioning, tests, instructions and documentation could be achieved with much less effort and cost in a standardised environment such as MAPLE will offer.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

Table C - 1 Estimated savings, using MAPLE

Software Item	Savings through using MAPLE	Estimated Savings in %
Data Server		
Standard Software	Base structure, Tools, Interfaces, Std. Procedures	0
Customized Software		33
Group Server 1, for older CNC Machines		
Standard Software	Tools	17
Customized Software	Architecture, Tools	16
Group Server 2, FMS Systems + Robot		
Standard Software	Architecture, Tools	17
Customized Software	Architecture, Tools, more possible with integrated Industrial Communication Standard	11
PC Workstations on 13 CNC's		
Standard Software	Architecture, Tools	29
Customized Software	Architecture, Tools, more possible with integrated Industrial Communication Standard	20
Post Processors		
14 different post processors	Library of post processors	67
Services	Commissioning, Tests, Instructions	52
	Documentation software and operating	40
Total Savings for Software and Services		**30**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST

© ISO

ICS 25.040.40

Descriptors: automation, automation engineering, production, manufacturing, computer applications, programming (computers), environment architecture, components, interfaces.

Price based on 30 pages

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Licensee=University of Alberta/5966844001, User=ahmadi, rozita
Not for Resale, 12/26/2014 07:49:56 MST