TESE DE DOUTORADO EM SISTEMAS MECATRÔNICOS

CONCEPTUAL FRAMEWORK FOR CLOSED-LOOP INSPECTION BASED ON THE COORDINATE MEASUREMENT TECHNOLOGY ADHERENT TO STEP-NC

CRISTHIAN IVAN RIAÑO JAIMES

Brasília, Junio de 2021

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

CONCEPTUAL FRAMEWORK FOR CLOSED-LOOP INSPECTION BASED ON THE COORDINATE MEASUREMENT TECHNOLOGY ADHERENT TO STEP-NC

CRISTHIAN IVAN RIAÑO JAIMES

TESE DE DOUTORADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM SISTEMAS MECATRÔNICOS

BRASÍLIA/DF, 16 JUNIO DE 2021

Riaño Jaimes, Cristhian Ivan
CONCEPTUAL FRAMEWORK FOR CLOSED-LOOP INSPECTION BASED ON THE COORDINATE MEASUREMENT TECHNOLOGY ADHERENT TO STEP-NC / CRISTHIAN IVAN RIAÑO JAIMES. –Brasil, 2021.
219 p.

Orientador: Alberto José Alvares

Tese (Doutorado) – Universidade de Brasília – UnB
Faculdade de Tecnologia – FT
Programa de Pós-Graduação em Sistemas Mecatrônicos – PPMEC, 2021.

1. Industry 4.0. 2. CAD/CAM/CAI. 3. Smart Manufacturing Systems. 4. STEP-NC. 5. Robotics. I. Alberto José Alvares, orientador. II. Universidade de Brasília. III. Faculdade de Tecnologia.

Dedicatória

No céu meu pai Luis Ernesto e minha mãe Carmen Miryam
Minhas adoráveis filhas Maria Camila e Maria Paula
Minha Amada Esposa Maria Fernanda
Minha querida irmã Emilce

CRISTHIAN IVAN RIAÑO JAIMES

Agradecimentos

Em primeiro lugar a Deus por sua constante presença e principalmente por me permitir avançar neste propósito.

Ao Professor Alberto Alvares pela orientação para elaboração deste trabalho.

Ao Programa de Pós-Graduação em Sistemas Mecatrônicos da Universidade de Brasília por propiciar as condições de elaboração de este trabalho.

Ao CAPES pela concessão da bolsa de doutorado.

Aos colegas do GRACO/GIAI/LabPRER e amigos da pós-graduação que direta ou indiretamente colaboraram para a execução deste trabalho.

CRISTHIAN IVAN RIAÑO JAIMES

ABSTRACT

The demands for quality and productivity in manufacturing parts with complex design specifications pose new challenges for manufacturing processes and quality inspection systems. The challenges are collecting, processing, transmitting, and storing data from the manufacturing and inspection process and refining each process related to the product life cycle. Concepts such as integration and interoperability, relevant within the digital context, show barriers that prevent their full implementation within the current manufacturing scenario. Implementing a closed manufacturing loop allows inspection results to be fed back into the digital chain and used to make decisions in the design, planning, and manufacturing phases that reduce uncertainties in manufacturing.

Dimensional and geometric inspection allows the generation of data containing traces of manufacturing. These correctly processed data can provide knowledge about causes of deviation of the manufactured part and manufacturing conditions that can be improved. This thesis work is framed within this perspective and presents a solution for integrating data generated in the different phases of the life cycle of a product. Data integration occurs within a closed-loop manufacturing architecture through a neutral, extensible, syntactically homogeneous language that allows both linking design information, manufacturing, measurement results, and supporting the flow through Computer-Aided Technologies (CAx) systems. As a result of the research, interoperable integration architecture based on the ISO10303 standard and its application protocols is presented, covering design specifications, manufacturing requirements, and dimensional and geometric inspection information exchange.

A computational implementation is developed using the Java environment to manipulate EXPRESS schemas, generate libraries with entities, functions, methods and develop an application that allows reading, writing, and modifying neutral STandard for the Exchange of Product model data (STEP) exchange files. The methodology followed through a practical conceptual framework is exposed to generate new applications based on the International Standards Organization (ISO) 10303 standard. Three case studies are presented to verify integration and interoperability within the closed-loop manufacturing architecture. The results reveal new lines of research that are proposed for future work.

RESUMO

As demandas por qualidade e produtividade na fabricação de peças com especificações de projeto complexas representam novos desafios para os processos de fabricação e sistemas de inspeção de qualidade. Os desafios são coletar, processar, transmitir e armazenar dados do processo de fabricação e inspeção e refinar cada processo relacionado ao ciclo de vida do produto. Conceitos como integração e interoperabilidade, relevantes no contexto digital, apresentam barreiras que impedem sua plena implementação no atual cenário de manufatura. A implementação de uma malha fechada de manufatura permite que os resultados da inspeção sejam realimentados na cadeia digital e usados para tomar decisões nas fases de projeto, planejamento e fabricação que reduzem as incertezas na fabricação.

A inspeção dimensional e geométrica permite a geração de dados contendo vestígios de fabricação. Esses dados processados corretamente podem fornecer conhecimento sobre as causas do desvio da peça fabricada e as condições de fabricação que podem ser melhoradas. Este trabalho de tese se enquadra nesta perspectiva e apresenta uma solução de integração de dados gerados nas diferentes fases do ciclo de vida de um produto. A integração de dados ocorre em uma arquitetura de manufatura em malha fechada por meio de uma linguagem neutra, extensível e sintaticamente homogênea que permite vincular informações de projeto, manufatura, resultados de medição e suportar o fluxo por meio de sistemas CAx (tecnologias assistidas por computador). Como resultado da pesquisa, a arquitetura de integração interoperável baseada no padrão ISO10303 e seus protocolos de aplicação é apresentada, cobrindo especificações de projeto, requisitos de fabricação e troca de informações de inspeção dimensional e geométrica.

Uma implementação computacional é desenvolvida usando o ambiente Java para manipular esquemas EXPRESS, gerar bibliotecas com entidades, funções, métodos e desenvolver uma aplicação que permite ler, escrever e modificar arquivos neutros de troca STEP. A metodologia seguida por meio de uma estrutura conceitual prática é exposta para gerar novas aplicações baseadas na norma ISO 10303. Três estudos de caso são apresentados para verificar a integração e interoperabilidade dentro da arquitetura de manufatura em malha fechada. Os resultados revelam novas linhas de pesquisa que são propostas para trabalhos futuros.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms

IDEF0 Integration DEFinition language 0. 33, 51, 74–78, 84, 96

IGES Initial Graphics Exchange Specification. 18, 19

IoT Internet of Things. 33

IR Generic Integrated Resources. 90, 92, 97

ISO International Standards Organization. i, 2

MBD Model-Based Definition. 5, 22, 32, 34, 38, 68

MIP Metrology Interoperability Project. 71

MVC Model-View-Controller. 100

NIST National Institute of Standards and Technology. 2

PDES Product Data Exchange Specification. 19

PLM Product Lifecycle Management. 3, 10, 31

PMI Product Manufacturing Information. 31, 34, 69

QIF Quality Information Framework. 12, 21–23, 32–34, 36, 89

SDAI Standard Data Access Interface. 24, 29, 36, 52, 58, 60, 68, 101

STEP STandard for the Exchange of Product model data. i, ii, 2, 7, 8, 24, 32, 33, 35–37, 50, 76

UoF Unit of functionality. 69–72, 85–90, 92, 94, 98, 99, 101, 106, 114, 126

XML Extensible Markup Language. 12, 15, 51, 59, 106, 118

XSLT Extensible Stylesheet Language Transformations. 23

Chapter 1

Introduction

In the digital age, driven by Industry 4.0, new requirements call for advanced manufacturing integration and interoperability. These integration and interoperability requirements create new challenges, for example, creating a digital chain to integrate each of the phases of a product life cycle. Total integration proposes to collect data from the processes in a dynamic way, generate information that can support decision-making, and establish communication in a language that is understandable for each agent involved in the process. The problem of interoperability is evident when the digital chain wants to include technological solutions from different owners. The manufacturing industry needs to adopt a neutral data structure to exchange data in its processes without losing information, which operates dynamically within an architecture.

By overcoming interoperability barriers, feedback measurement information aids decision-making in the design, planning, and manufacturing phases. Dimensional and geometric inspection enables the generation of data that contains information about the deviation or uncertainty of a manufactured part and data that covers relevant manufacturing aspects associated with manufacturing conditions. If a more specific measurement is made, the information associated with the final appearance of the part and its surface properties can also be determined. Some decisions may include setting manufacturing parameters, selecting suitable cutting tools, defining machining strategies, and setting up and sequencing operations. Features such as repeatability, precision, and quality in part manufacturing are positively influenced in a controlled manufacturing environment.

1.1 Background and Problem Motivation

The variability and uncertainties in part can come from different sources with different natures; The aim is to reduce these values and maintain the geometric and dimensional characteristics within the projected values and thus guarantee the correct functioning of the piece. The primary motivation of the project is to propose an integration architecture for the different phases of the product life cycle, share design, manufacturing, and measurement data through a neutral data structure, extensible and interpretable by the different technologies associated with the different phases of the product life cycle.

Advanced manufacturing requires significant data management and processing of the information generated during the manufacturing process. Manufacturing requires full integration of CAD/CAPP/CAIP/-CAM/CAI (computer-aided design, process planning, inspection planning, manufacturing, and inspection) systems to operate from an interoperability environment, regardless of the technologies used in each system. Data shared between systems must be interpreted, analyzed, and processed to generate information that can support decision-making, implement process control, virtualize and automate manufacturing (BERNSTEIN et al., 2017; HEDBERG; HELU, 2017).

Using Coordinate-Measuring Machine (CMM) as a computer-aided inspection tool, Computer-Aided Inspection (CAI) provides the necessary conditions for Closed-Loop Manufacturing (CLM) with feedback of measurement data. It remains a challenge to find a neutral, standardized, and interoperable data structure that covers all phases of the inspection process and integrates the results of the manufacturing chain. The reasons stated above and the need to find a solution that meets the demands presented by Industry 4.0 justify the development of a strategy for feedback of inspection results within a closed manufacturing cycle (ZHAO et al., 2010a).

Different efforts to overcome the interoperability barriers present in the measurement system are promoted in the international context. The ISO highlighted that the main problem in integrating measurement systems is the variability in the definitions, since it directly affects the consistency of the knowledge obtained in the measurement(ISO 17450-2, 2012). Different corporate practices with a commercial tendency to use proprietary solutions, various international standard options for exchange, and the myriad of technologies involved in inspection made this task even more complicated. An interoperability architecture is required for the integration of specific information (RAY; JONES, 2006).

In the 1990s, research already reported on the impact of computer technologies on the performance of manufacturing processes and the role of computer systems in the integration of machine tools, robotics, and production systems (GUNASEKARAN et al., 1994). A study commissioned by National Institute of Standards and Technology (NIST) (NIST, 1999) quantified the industry penalty for the lack of an interoperable architecture that avoids the need to exchange formats for sharing information between CAx systems. This study reported that the US automotive industry spends $1 billion a year to solve interoperability problems. The study also reported that up to 50% of this expense is destined to face data file exchange problems, part of the item used to develop information interpreters. In the search for a solution, approaches have emerged that involve open and neutral architectures to avoid using bidirectional translators and provide some stability in representing information. An example of a successful open standard is STEP, a set of extensible standards that define a neutral representation for data generated throughout the lifecycle of a product (GALLAHER et al., 2002).

The report NIST (2004) presents the discussion on the interoperability of measurement and the challenge that standards must assume to operate in this digital environment addressed by the team of the metrology interoperability project of Automotive Industry Action Group (AIAG). The purpose is to find a solution to reduce the product development cycle, manufacturing time, and costs through the interoperability of systems supported by software, hardware, and components used in automated metrology (RIPPEY, 2005).

In the report (NIST 2006), the NIST continued this discussion and wrote some interoperability issues related to current standards. The report presents projects in interoperability testing and self-integration research, with the goal that future standards rely on the use of formal logical representations that will enable the automation of many integration tasks (RAY; JONES, 2006).

The Digital Twin machining concept uses models of parts, tools, accessories, and operations and tool paths. The complete model is loaded for real-time simulation (HARDWICK, 2017). As for actual machining proceeds, the simulated model is updated to show the same Digital Twin results. Therefore, it is possible to analyze the results from remote locations, smartphones, and browsers. Measurements can be made on Digital Twin, and alerts can be sent if tolerances are not being met (LU et al., 2020).

These efforts present significant opportunities for reusing knowledge throughout the design process. Research is focused on getting information from both the physical and digital worlds using the same medium within this perspective. It is essential to create a solution that covers the different life cycle stages and supports decision-making. The variety of patterns in this space makes it difficult to converge towards a shared vision within the manufacturing environment. It is also necessary to formalize different aspects of the product and provide techniques for mapping information into Product Lifecycle Management (PLM) patterns. In this work, the solution focuses on combining data to provide a decision-making platform that meets inspection requirements and feeds the manufacturing system's results.

1.2 Research Questions and Hypothesis

A previous review of integration issues in CAx systems that support the product lifecycle raised concerns about closed-loop manufacturing architectures. The most relevant questions lie in the role of inspection systems and how data integration happens. Each measurement on a manufactured part provides traces of the manufacturing conditions and the identified causes of deviation and uncertainty. The data acquired in the measurement process requires an adequate structure that supports both descriptive and structural information and must handle precise and homogeneous semantics to transfer and interpret them.

An initial stage of exploratory research glimpsed the line taken by observing the requirements and demands of the digital age for CAx systems involved in the life cycle. Digital data is required to communicate, optimize, simulate, experiment, and evaluate the impact of decisions made in the manufacturing process through performance metrics. This digital context is an adequate justification to encourage the generation of solutions conducive to obtaining an integrated and interoperable measurement process with active participation within the digital world.

1.2.1 Hypothesis

The hypothesis adopted in this project consists of verifying that by implementing an integrated and interoperable dimensional and geometric inspection with a neutral data structure compatible with information and manufacturing data, the results and analysis can be feedback within a system of in CLM. The main problem to be solved is to manage the measurement results within the manufacturing chain dynamically. The development of a dimensional inspection architecture with a neutral, integrated and interoperable data structure, operating in a closed-loop, seeks to obtain the following advantages that challenge current interoperability barriers:

- Feedback on the inspection results with the different CAx systems.
- Ensure the consistency and integrity of the information captured in the measurement process.
- Avoid switching formats between CAx systems.
- Harmonize measurement data with manufacturing information to enable the development of new control strategies.
- Eliminate format incompatibility that results in loss of information.
- Eliminate restricted access to specific technologies.
- Facilitate the storage of information generated during the product's life cycle.

1.3 Purpose Of The Study

Different architectures for CLM allow to optimize the process and promote sustainable systems. CLM execution requirements change progressively in this digital age, but it is still paramount to implement full integration and maintain interoperability between CAx systems (BRODSKY et al., 2015). This work aims to digitize and share manufacturing and inspection information within a closed-loop manufacturing. This digitized information will incorporate advanced functionality into traditional manufacturing systems and streamline manufacturing processes.

The data collected in the measurement is the knowledge base to optimize different aspects of the manufacturing process. In addition to the manufacturing conditions, the data allows identifying functional improvements within a design perspective. Dimensional and geometric adjustments are applied to design, and the knowledge generated is the basis for the development of new projects. Through closed-loop manufacturing, the correct operation of the system is guaranteed by controlling the manufacturing environment, treating sources of uncertainty, reducing human intervention, and improving levels of precision and accuracy, favorably influencing the quality and process productivity.

Despite the already existing connection between CAD/CAPP/CAM/CNC systems, interoperability barriers prevent fully meeting integration requirements. The problem is visible if the manufacturing chain contains the Computer-Aided Inspection Planning (CAIP) and CAI systems. However, the integrated CAD/-CAPP/CAIP/CAM/CAI/CNC system needs a homogeneous data exchange architecture, mechanisms to encode process information, systems to store and share information, and means to virtualize processes.

The data of a part undergoes a series of transformations and losses in each of the CAD/CAPP/CAIP/-CAM/CAI/CNC systems, caused by incompatibilities of proprietary formats. Parameter changes made by any of the CAx systems may not be noticed in subsequent processes(HARDWICK, 2016; HARDWICK et al., 2013). There may be a difference between the product obtained and the designed one at the end of the manufacturing chain. Therefore, it is necessary to use an integration architecture with a neutral data structure accessible by any technology (XU; NEWMAN, 2006). An open, neutral, and extensible data model is an effective mechanism for solving a data incompatibility problem and building a dynamic knowledge base (ZHAO et al., 2010b). A vision still under study uses the STEP-NC standard to support intelligent and interoperable production, distributed across a global network with STEP-NC compliant data interpretation and autonomous manufacturing workstations (RIPPEY, 2005).

The purpose of the study incorporates planning aspects of the inspection process, data analysis, and reporting of the dimensional measurement of prismatic parts manufactured by machining. The inspection process is carried out in the final stage of the product's manufacturing cycle when required to verify compliance with the design specifications. The manufacture of prismatic parts should preferably be done in an integrated CAD/CAPP/CAM/CNC environment to ensure error compensation operates under the same manufacturing conditions. The data analysis and the inspection results report will be feedback to the design, planning, and manufacturing phases. The compensation strategies seek to associate uncertainties with causes and apply corrective actions that improve the process. It may also be possible to update the tolerance values, which allow compliance with the project specifications. Tolerance values are calculated within a compensation strategy, and statistical methods are used to determine them. The inspection strategy can be applied to other manufacturing processes, such as additive manufacturing and turning.

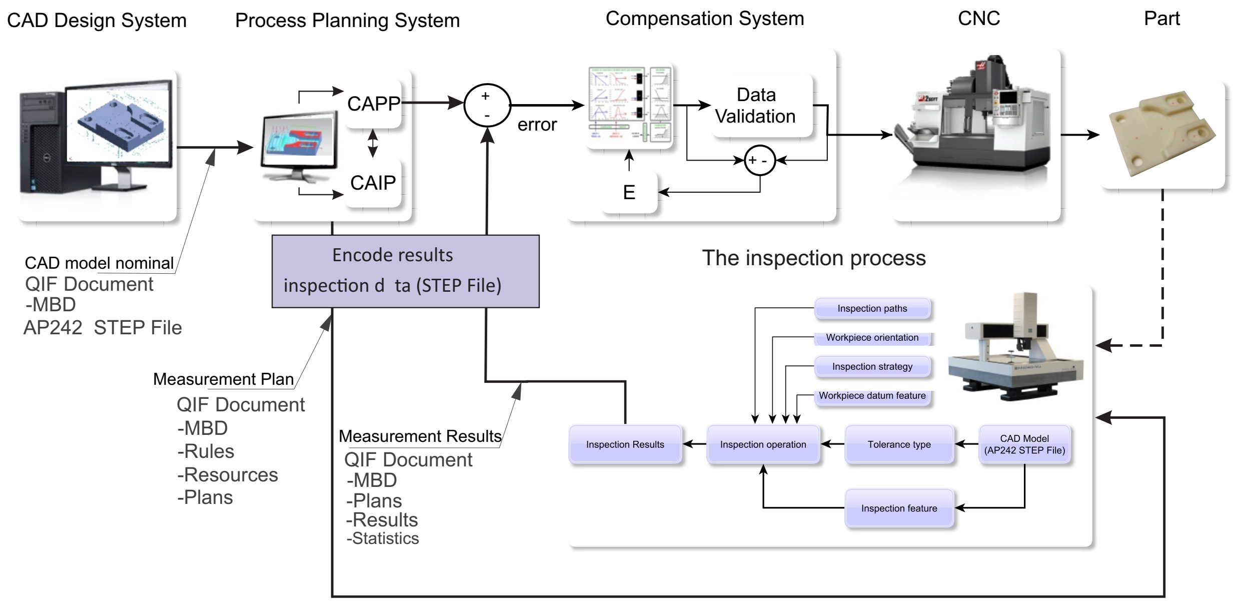


Figure 1.1: Components of the closed-loop manufacturing and inspection system.

The Figure 1.1 presents the components of the closed-loop manufacturing and inspection architecture. This diagram summarizes the purpose of the study, the projected data flow, and how each system that makes up the product life cycle is interrelated. Computer-Aided Design (CAD) model data shared within the digital manufacturing chain through a neutral format, ensuring interoperability and preserving information integrity. For example, from the STEP AP242 file, a Model-Based Definition (MBD) generated in the QIF MBD format is used within the computer-aided inspection system. The CAIP also adheres to this architecture to improve the conditions to perform the measurement.

1.4 Aims and Objectives

The general objective is to develop an architecture to implement dimensional and geometric inspection integrated within a closed manufacturing loop, adhering to the ISO 130303 standard. The development of this architecture seeks to obtain the following advantages that defy current interoperability barriers:

- Feedback inspection results between different CAx systems.
- Ensure the consistency and integrity of the information obtained in the measurement process.
- Avoid switching between the formats that support the CAx system, the product's life cycle (design, planning, manufacturing, and inspection)
- Harmonize measurement data with manufacturing information to enable the development of new control strategies.
- Eliminate format incompatibility that results in loss of information.
- Eliminate restricted access to different technologies.
- Facilitate the storage of the information generated during the product life cycle.
- Bidirectional transmission of information.
- It unifies the information in a single file, avoiding the problematic manipulation of the product data and generating multiple versions of the project.
- Link inspection and manufacturing data to define new error compensation strategies.
- Generate knowledge by capturing acceptable practices that improve the quality control system and promote process improvement.
- Systematize decision-making that in many measurement processes depends on the experience of the operator.

The specific objectives of the investigation are summarized below:

- Identify the different variables related to CAD/CAPP/CAIP/CAM/CAI systems (computer-aided design, process planning, inspection planning, manufacturing, and inspection).
- Define the integration architecture for integrated dimensional and geometric inspection in closed-loop with the manufacturing system.
- Define the methodology to implement the closed-loop integration architecture.
- Implement the integration architecture for dimensional and geometric inspection united into a closed-loop manufacturing system.
- Prove the integration and interoperability of the architecture with the traceability of a part design.

- Propose a method of analysis of measures and correction of errors that contributes to improving the manufacturing process
- Evaluate and synthesize the integration architecture results for dimensional and geometric inspection integrated into a CLM system.

1.5 Thesis Contributions

The thesis presents a methodology for the development of applications based on the ISO 10303 standard. The development of an application that allows creating a project to integrate design, manufacturing, and inspection data within a single STEP neutral exchange file, shared within a closed loop of manufacturing and inspection, allows verifying the methodology. The swap file supports the transfer of data that enables process integration in the different phases of the product life cycle.

The computational solution obtained uses the neutral data structure to support the information of the manufacturing closed loop, allowing the data to be accessible at all levels, such as managerial, operational, and physical. An architecture for data flow enables the digital manufacturing chain and integrates inspection results to support part error correction and process continuity.

It was necessary to adopt a syntactically and semantically homogeneous data structure to transfer the information between the CAx systems involved in the life cycle. This document describes the information modeling process, the technological tools used, and the procedures required to create systems adhering to the ISO10303 standard.

The main contribution is information modeling with three components: functional, conceptual, and implementation. Each of these models incorporates integration and interoperability requirements. The functional model describes the high-level activities involved in both the inspection and manufacturing closed loop. The conceptual model expresses the knowledge of the application universe and allows relating the knowledge described with implementation activities. The implementation model describes a methodology to interpret the conceptual model, translate reference concepts to coded lines in a programming environment, enabling the implementation and development of specific solutions adhering to the STEP standard.

1.6 Structure of Thesis

This section presents the structure and a synthesis of the content found in the document.

Chapter 2 presents a review of the literature to contextualize the integration and interoperability problem facing manufacturing systems, the challenges posed by Industry 4.0, and the structures that support closed manufacturing ties. The most relevant data structures in manufacturing and measurement and alternatives to feedback the measurement results are also presented.

Chapter 3 formally describes the arguments for selecting the STEP standard data architecture to solve the integration problem and the challenges to overcome in its use. The data structure of the STEP standard and the information modeling for data exchange are presented, and the strategies and techniques for its implementation.

Chapter 4 presents an introduction to information modeling, its importance in the development of the implementation, and the proposal's scope through functional and conceptual models. The Application Interpreted Model (AIM) model is presented as a tool to translate conceptual models into implementation and programming schemes.

Chapter 5 presents the development obtained through examples and case study, the process of interpreting the STEP standard for implementation purposes is explained. The development environment and its characteristics and the results obtained are shown.

In Chapter 6, the conclusions of the work are presented. New lines of research derived from the observations and results obtained are given as recommendations for future work. Research contributions are listed and exposed in this section as well.

The Appendices section contains a series of documents generated during development related to a section to support its development or provide more details than described. Appendix A.1 presents the deployment of the functional model with more specific details on the relationship of the activities for the closed-loop manufacturing and inspection integration model. Appendix B.2 presents the neutral STEP file generated through the textual interface built in this investigation with the structure of a manufacturing project. Also included is an extract of the final file (the Original file contains more than 3000 lines), nurtured with entities through STEP Machine. Appendix C.3 shows some definitions in EXPRESS-G of entities that are part of the descriptive model. Appendix D.4 presents the machine code in Dimensional Measuring Interface Standard (DMIS) format, generated for automatic inspection through CMM. Appendix F.5.1 presents a Matlab script created to obtain the angle of deviation of the axis of a machine concerning a feature of the part design to identify causes of errors in the manufacturing process. The appendix shows the dimensional inspection result performed with a CMM machine for twenty-eight different parts. The annex I.1 presents the EXPRESS schema of the `shape_tolerance_schema entity`, used as an example in explaining descriptive methods used by the ISO10303 standard.

Chapter 2

Reference Models: Literature Review

This chapter presents a literature review with the challenges faced by manufacturing processes in the current scenario brought by industry 4.0. A description of the CLM based architectures and the integration issues associated with the new requirements is made. The patterns of data structure available to share information are described, and through the correlated works, the scope of the investigation is introduced.

2.1 Advanced Manufacturing in the Digital Context

Manufacturing and quality processes require an architecture that supports the integration of CAx systems to meet the information virtualization demands driven by the fourth industrial revolution. This section presents integration and interoperability, which helps develop an architecture that supports the digital chain. Within this context, it is necessary to define a strategy for acquiring and sharing data. The measurement results are homogenized both semantically and syntactically to be managed by the systems involved in the product life cycle. Dimensional metrology data is integrated to create a compensation strategy and correct manufacturing errors identified in the inspection process. A frame of reference is presented that highlights the contributions in this line of research.

There is a large amount of work related to CLM that allows building knowledge about the bases of process integration and the problems of information flow that currently face the systems. The most relevant issues to be exposed to the bibliographic review in this section are: to contextualize the problem within the requirements of industry 4.0 and find an architecture that meets the current demands for the exchange of manufacturing and inspection information. Table 2.1 presents a summary of the literature with relevant results in this research area.

Table 2.1: Syntheses of the literature review

Author	Research context	Comments
(LI, 2013a)	Architecture development for product information exchange.	It shows some research developed based on product information standards in context. It presents an implementation methodology to integrate different product information from a standardized data model. Emphasizes the importance of using EXPRESS and SDAI (ISO 10303 -22 Standard Data Acces Interface) for computational implementations. The contribution of research and a computational tool based on STEP.
(DANJOU, 2015)	Integration and interoperability in a CLM.	It makes a study of the context of the application of STEP-NC standard to solve the problem of integration and interoperability of current processes. The research develops a data model proposal to implement closed-loop manufacturing based on a STEP-NC model and PLM systems for CAD/CAM.
(NEWMAN et al., 2008)	Interoperable manufacturing.	Based on the literature review, the author presents a strategy to create an interoperable manufacturing environment. The STEP-NC standard is considered the integration mechanism, reporting how bidirectional information happens between CAx systems in the digital chain. The article raises questions and challenges for the future of digital manufacturing within the global vision of interoperability.
(XU* et al., 2005a)	Context of intelligent integration of CAD/-CAPP/CAM/CNC systems.	It details why the emergence of the STEP standard was allowed to exchange information. It presents a compilation of research on integrating and exchanging data in manufacturing systems using the ISO 10303 standard.
(BRECHER et al., 2006)	Integration of closed-loop inspection with manufacturing processes.	Contextualizes the integration of inspection tasks in a closed loop with machining operations, presenting data structures based on the ISO 14649-16 part. It presents the exchange of information in the inspection processes using DMIS and the exchange of measurement results with Dimensional Markup Language (DML).
(ALVARES, 2005)	CAD/CAPP/CAM systems integration and interoperability.	It presents a methodology for CAD/CAPP/CAM integration to manufacture rotational parts with collaborative, interoperable, and intelligent functions via the Web. The systems are integrated through the concept of features.
(ZHAO et al., 2008a)	Inspection integration based on the ISO 10303 standard.	Contextualizes the integration of inspection tasks in a mesh with machining operations, presenting data structures such as the ISO14649 part 16 standard and Dimensional Measuring Interface Standard DMIS to exchange information associated with inspection and Dimensional Markup Language DML processes to exchange measurement results.

Continued on next page

Table 2.1 – continued from previous page

Author	Research context	Comments
(ŽIVANOVIĆ; GLAVONJIĆ, 2014)	Methodology for implementing the STEP-NC standard.	They propose a methodology for implementing the STEP-NC standard as a data structure in machine tool programming. The methodology is verified in two environments.
(QIN et al., 2017)	Methodology for implementing the STEP-NC standard.	The article presents a review of the models used to represent tolerance information to determine whether, with an EXPRESS model, it is possible to implement complete representation and tolerance information semantically. The review includes an analysis of the different representation models integrated into software such as AutoCAD and SolidWorks.
(ZHAO et al., 2006)	Model for representing geometric tolerances in integrated measurement processes.	The author proposes a multilayer model for representing geometric tolerances based on XML and XML schemas. Through EXPRESS, the requirements are modeled, and the levels and layers of the model are confirmed. With a case study, they detail the definition of data in the XML file.
(ŽIVANOVIĆ; GLAVONJIĆ, 2014)	Methodology for implementing the STEP-NC standard.	They propose a methodology for implementing the STEP-NC standard as a data structure in machine tool programming. The methodology is verified in two environments..
(BRECHER et al., 2006)	Closed-loop CAD/-CAPP/CAM systems with inspection tasks based on the STEP and STEP-NC standard.	It presents the data flow based on the STEP and STEP-NC standard to perform the measurement integration in machining sequences. It shows an overview of part 16 and its relationship to other standards such as AP219 to integrate inspection results within the manufacturing chain.
(PEAK et al., 2004)	Complementary technologies in the integration of the STEP standard,	The article presents a description of WEB-oriented technologies as they are XML and UML, to assist in the integration and interoperability of systems that intervene in the life cycle of a product and that have a data structure based on the STEP standard. Implementing STEP methods, such as EXPRESS and p21, provide standardization and enable the use of more widespread languages to achieve integration with web-based solutions.
(BHANDARKAR; NAGI, 2000)	Product representation and recognition of features.	The article uses the product representation capability to obtain geometry and topology information for a part using the AP224 application protocol. The research's main objective is to develop a system using STEP definitions for extracting features to convert design information into manufacturing data.
(XU et al., 2006)	STEP-NC for manufacturing interoperability.	The article introduces the STEP-NC standard and developed technologies based on the standard. It presents an analysis of the pattern from a functional perspective that includes bidirectional communication, data flow, and features.

Continued on next page

Table 2.1 – continued from previous page

Author	Research context	Comments
(ZHAO et al., 2012)	Integration of metrology processes using the QIF standard.	The article presents an alternative for metrology data exchange based on the Quality Information Framework (QIF) standard. The article introduces the data structure, definitions of features, and interchange model using Extensible Markup Language (XML) schemas for data encoding.
(MORSE et al., 2016)	Integration of data generated from the four main activities that support a quality measurement system using the QIF standard.	The article describes the data structure of the QIF standard that supports the flow of data generated in the design project in the definition of tolerances and measurement. The author gives a brief description of the XML language used in the QIF standard to exchange information on activities: product definition, measurement planning, measurement execution, analysis, and reporting of quality data.
(VEEN et al., 2017)	Optimization integration in closed-loop control.	The article focuses on the closed-loop control performance of a motion system component while ensuring that mechanical requirements are met. Based on an example, it is shown that this leads to non-trivial and non-intuitive designs that provide better performance. The structure allows the rapid development of prototype designs with additive manufacturing techniques.
(BAGSHAW; NEWMAN, 1999)	Feedback from quality information.	The authors have defined an integrated inspection system with analysis of production data. This involves creating geometry based on features, an operation plan, creating an inspection procedure, automatically analyzing the tolerances of features, and an expert system to determine error causes.
(ZHOU et al., 1995)	Method for estimating and compensating errors in the manufacturing process.	The work presents an artificial intelligence system based on linguistic rules. The fuzzy controller is combined with a multilayer neural network that estimates and compensates for manufacturing process errors in Computer Numerical Control (CNC) machining.
(ANJANAPPA et al., 1996; ANJANAPPA et al., 1990)	Method for estimating and compensating errors in the manufacturing process.	The work presents an algorithmic and heuristic approach to create a methodology by which the error sources can be qualified using only the inspection report. The focus of the work is on determining the deterministic dimensional errors produced during the assembly operations.

Continued on next page

Table 2.1 – continued from previous page

Author	Research context	Comments
(RENTOUL et al., 1994; MEDLAND; MULLINEUX, 1993)	Method for estimating and compensating errors in the manufacturing process.	The authors analyze how inferences about manufacturing errors can be made from comparing inspection points with a solid model of the desired product. The approach deals with forming a hierarchy of stages within a typical manufacturing process and trying to match the computer model's inspected points. This approach was validated with the aid of the RASOR modeler constraint and identified plausible manufacturers resulting from manufacturing errors.

In an attempt to deliver the best results, the manufacturing industry includes advanced technologies to improve product quality, save resources, and meet user demands. Higher levels of complexity need technologies to acquire information from the design, process planning, manufacturing, and measurement processes. The concept of intelligent manufacturing is related to the fourth industrial revolution, also known as industry 4.0, where different technologies, mainly digital, converge to take advantage of data and generate information to promote the evolution, optimization, and interoperability of processes (LEE et al., 2015).

Information management as through the different digital technologies currently available makes advanced manufacturing considered smart manufacturing. It is evident that manufacturing systems are not ready to manage information and need intelligent tools and data structures that facilitate integration (LI, 2013b; ALVARES, 2005). The most relevant issues being studied and seeking to promote solutions for advanced manufacturing can be classified into the following categories:

- Ensure human knowledge of processes.
- Share process information and operating conditions.
- Pursue the quality and sustainability of products.
- Administer and manage information.
- Create technological advances in measurement, sensors, and data acquisition.

Resolving part of the previous questions traces the way for the current manufacturing systems to be inserted in the context of industry 4.0 and allow the improvement of processes with a sustainable model of integration and interoperability (LEE et al., 2014).

Cyber-Physical System (CPS) is strongly linked with advanced manufacturing systems as it incorporates two essential functions: advanced connectivity and intelligent data management. Despite technological advances, the manufacturing and inspection processes still do not achieve the complete integration and interoperability of their systems, but there is already a significant evolution in sharing information based on neutral files philosophy. The following section presents archives evolution based on free access and a neutral structure. Within the CPS architecture, it is necessary to acquire information from reliable sources of the process, machines, and equipment. It is also necessary to consider the data structure, methods for acquisition, management, transfer procedures, and transmission protocols.

Measurement systems are based on physical principles to generate data from an environment or process. Different technological advances are used to improve the measurement's quality, with different possibilities in selecting a device. The problem with the integration of measurement results lies in the conversion of data to useful information. The methods to generate helpful information for a process from measurement results is a problem that has recently aroused much interest and maintains a growing trend in research linked to Big Data. Significant information is created from reliable data sources, intelligently managed, and computationally analyzed to build relevant information with a high degree of knowledge (LI, 2013b).

A data architecture adhering to the industry 4.0 paradigm, based on the concept of CPS, demands advanced connectivity, intelligent tools, and knowledge obtained from the conversion of data to information. From this perspective, manufacturing processes need changes in the treatment of data in their systems. With better use of available digital technologies and the physical resources associated with each process, it is possible to obtain greater control of production in each phase that makes up a product's life cycle.

In the path of integrating information obtained from a manufacturing and inspection process, it is necessary to think beyond using the information only to improve the process of manufacturing a part and its quality; think about how this information can be integrated and used by other systems that share the same operating conditions. Learning from other processes with similar operating conditions and using good practices to improve our system will insert future projects from sustainability (ALVARES, 2005). This research focuses on finding a method to structure the data that is compatible with the new requirements of industry 4.0 and that can guarantee interoperability regardless of the technology involved in the product life cycle.

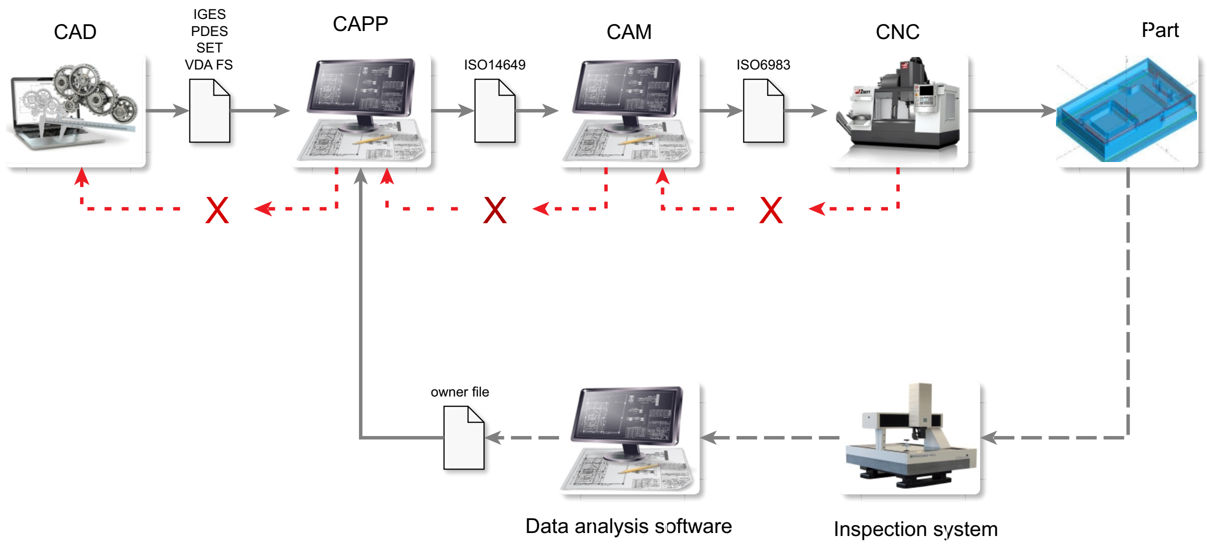


Figure 2.1: Outline of a conventional inspection and manufacturing process.

Outline of a conventional inspection and manufacturing process: In conventional manufacturing processes, as shown in Figure 2.1, tool paths are generated by Computer-Aided manufacturing (CAM) systems, where part information is extracted from a virtual model created from a CAD system. The tool paths are stored in a computerized numerical command file based on the ISO 6983 standard for processing on the machine tool. Despite the connection between each CAD/CAPP/CAM/CNC system, it is not easy to follow product evolution.

The product information undergoes a series of transformations in each of the CAD/CAPP/CAM/CNC systems, caused by incompatibilities between proprietary formats. Changes in the parameters of a product made by any of the CAD/CAPP/CAM/CNC systems may not be noticed in later processes. At the end of the manufacturing chain, there may be a difference between the product obtained and the one designed. This architecture makes corrective actions difficult. The problem is greater if the manufacturing chain, CAIP, and CAI planning processes are inserted. The exchange of information for each process is not guaranteed, and production errors can be repeated due to the difficulty of storing successful practices.

Despite technological advances, the manufacturing and inspection processes still do not achieve the complete integration and interoperability of their systems, but there is already a significant evolution in sharing information based on neutral files philosophy. The following section presents archives evolution based on free access and a neutral structure.

2.2 Machine Data Monitoring

In the manufacturing chain, some variables can provide critical information for the improvement of the process. The impact of decisions in the planning phase directly affects the yield and quality of the process. Much information is verified long after the machining process has ended, representing a problem. Mtconnect emerged as an alternative in response to the need to capture, monitor, and analyze the manufacturing environment data. This correctly processed data generates knowledge of the process and allows corrections in real-time (SHIN et al., 2016; OLIVEIRA, 2017).

The MTConnect standard enables equipment to provide data in (XML) format. The manipulation of data in a non-proprietary format generates new ways to improve processes, optimizing and increasing productivity. MTConnect defines a common language and structure for communicating manufacturing equipment monitoring data (SOBEL, 2014).

The MTConnect architecture consists of the following components:

- `Adapter`: Optional software component that connects or agent to a device.
- `Agent`: A software element that receives data from the adapter or a controller with MTconnect support makes that data available in XML format through HTTP request.
- `Device`: A device capable of operating. A device can consist of a set of components that provide data to the application. The device is a separate entity with at least one component or data item providing information about the device.
- `Application`: A process or set of processes that access the MTConnect Agent to perform a task.

To meet higher interoperability levels, MTConnect is based on the most relevant industry standards, maximizing the number of tools available for its implementation. MTConnect is composed of the following parts:

- `Header`: Protocol-related information.
- `Components`: The building blocks of the device.
- `DataItems`:The description of the data available on the device
- `Streams`: : A set of Samples, Events or Condition for components and devices.
- `Assets`: An Asset is associated with the manufacturing process that is not a component of a device, can be removed without impairing the device's function, and can be associated with other devices during the life cycle.
- `Samples`: A spot measurement of a data item that is changing continuously.
- `Events`: Discrete changes in the state that can have no intermediate value. They indicate the state of a specific attribute of a component.
- `Condition`: Information that the device provides as an indicator of status and ability to function. A condition can be Normal, Warning, Failed, or Not available. A single type of condition can have multiple Faults or Warnings at any given time. This behavior is different from Events and Samples in that a data item MUST only have a single value at any given time.

The `MTConnectDevices` structure provides descriptive information about each device reached by the agent and specifies the available data items. The `MTConnectStreams` structure contains a series of sample values, events, and conditions for devices and their components. A `MTConnectAsset` document contains relative information about a machine tool asset that does not belong directly to it and can be located on another device. The structure of the `MTConnectError` document contains information related to an error that occurred when processing the request.

2.3 Data Structure for Product Representation

A product goes as through several phases to get from idea to natural physical element. Within this document, the life cycle is associated with the design and development period from the design specifications until the piece's manufacture. Subsequent phases, such as placing on the market and following, are ignored. The stages contemplated in the development cover the functional activities of design, process planning, inspection planning, manufacturing, and inspection aided by a computer (ALI et al., 2005). How data transmission is commonly carried out in these phases represents a major integration problem. The most relevant problems arising from these phases are:

- Loss of data and information.

- Incompatibility of formats and problems associated with proprietary software.
- Restricted access to certain technologies.
- Unidirectional data transmission.
- Difficulty manipulating data and making corrections.
- Difficulty in storing data and information resulting from good practices.

There are two options to solve a data transmission problem: use a transducer in each system or use a neutral format to exchange information, compatible with the technologies associated with each system. Transducers are considered dedicated programs that allow interconnection between systems. In the exchange of information with transducers, each system has a dedicated transducer for each connection. The interconnection with transducers tends to be variable when the number of systems involved is small; when the number increases, the possibility of having transducers for each connection may be impossible to satisfy.

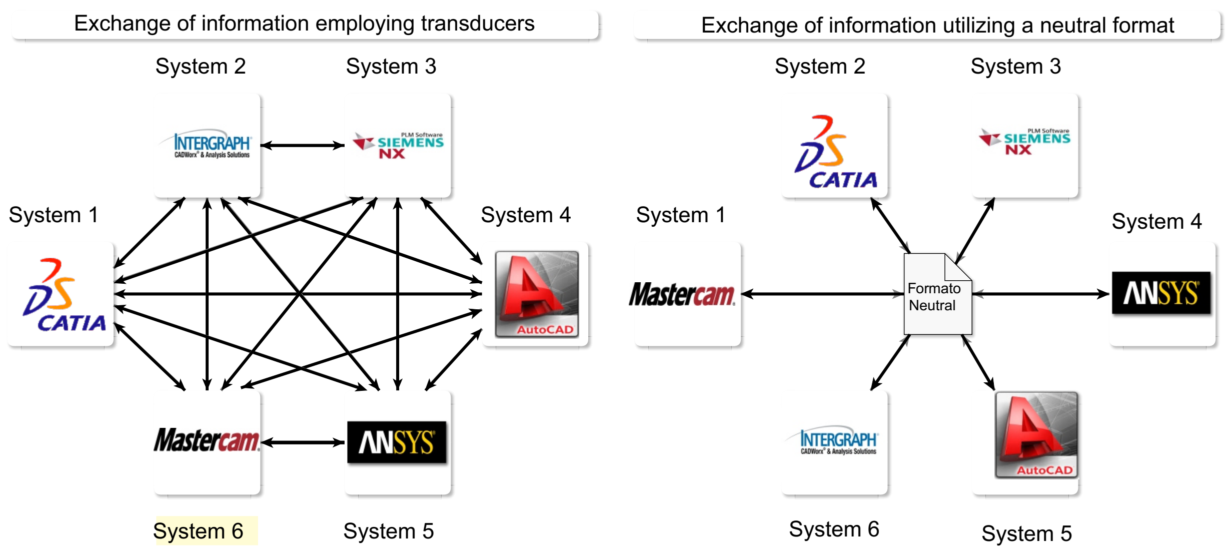


Figure 2.2: a) Exchange of information employing transducers, b) Exchange of information utilizing a neutral format.

Figure 2.2 presents the two solutions; for the first scheme, the equation 2.1 gives the total number of transducers.

$$N = 2 \times \frac{n!}{2!(n-2)!} = n \times (n-1) \tag{2.1}$$

n: Represents the number of systems. N: Represents the total number of translators required.

The exchange of information without a neutral data structure, as shown in the first scheme of Figure 2.2, the number of required transducers is thirty. When using the exchange structure based on a neutral format, the number of transducers is calculated by equation 2.2.

$$N = 2 \times\ n \tag{2.2}$$

As shown in Figure 2.2, the necessary transducers are twelve using a neutral format for exchanging information. Table 2.2 shows a comparison of the number of transducers required for the transmissions shown in Figure 2.2.

Table 2.2: Comparison of the number of transducers required for data transmissions.

Number of systems involved: (n)	Number of transducers with straight transmission $n \times (n-1)$	Number with file-neutral transmission: $2 \times\ n$
4	12	8
8	56	16
12	132	24
16	240	32

The development of solutions based on the concept of neutral files allowed the evolution of new data structures. Some structures are described below.

2.3.1 Initial Graphics Exchange Specification (IGES)

Initial Graphics Exchange Specification (IGES) was created based on the concept of neutral file driven by CAD software providers to provide the ability to transfer data between different systems. IGES allows describing the modeling data in a neutral format, which CAD/CAM systems can read. It contains information entities for the construction of the IGES model, necessary parameters for the definition of the IGES model, and the relationship between the model's entities (IGES, 2001). The architecture of a IGES file contains six subsections structured in the following order:

- **Flag Section**: The section indicates the type of file format (binary or compressed).
- **Start Section**: Required for the start section, provides a readable start for the file.
- **Global Section**: It contains information that describes the preprocessor and information needed to manipulate the file in post-processing.
- **Directory Entry Section**: It contains a set of fixed records (twenty fields of eight characters on two consecutive lines of eighty characters) for each entity in the file. The purpose is to provide an index to the file and store attribute information for each entity.
- **Parameter Data Section**: The section contains the parameter data associated with each entity. Can contain pointers to any combination of one or more entities
- **Terminate Section**: The section specifies from a single record the number of records from each of the previous sections to verify their compliance.

The IGES standard, in its appearance in 1980, had a limited scope only to only CAD systems; since that date, it has evolved and is now capable of providing security for consistent implementations. Its use has been adopted as the American national standard, accepted according to ANSI. Y14 (*American National Standards Institute*).

2.3.2 PDES (Product Data Exchange Specification)

It is a standard format for encoding all information about a product required for manufacturing purposes (design and planning steps). The Product Data Exchange Specification (PDES) describes a complete product, including the geometric aspects of the images and manufacturing features, tolerance specifications, material properties, and finishing specifications (ZEID, 1991; HORST; MCSPADDEN, 2006).The architecture is structured in three layers, defined as:

- `Application Layer`: This layer contains all descriptions and information for various application areas. It operates as an interface between PDES and the user. The descriptions and applications are formally expressed within the PDES through the information modeling techniques known as the reference model. Both IDEF1 and ICAM (Incident Cause Analysis Method), NIAM (National Integrated Assessment Model) are used as reference models.
- `Logical Layer`: This layer is intended to provide a consistent description of the constructor data, both generic and application, containing the exchange information. The goal is to ensure no redundancy in generic data structure and the relationships supported by the range of applications.
- `Physical layer`: The physical layer deals with the formats, structures, and interchange of files data. The goal is to establish and maintain efficiency in file size and processing time to avoid problems similar to those experienced using IGES.

2.4 Data Structure for Measurement Representation

2.4.1 Dimensional Measuring Interface Standard (DMIS)

ISO 22093: 2011 defines a neutral language for communication between information systems and dimensional measurement equipment (DME) called DMIS (Dimensional Measuring Interface Standard). DMIS is an execution language for part measurement programs that provides a format for exchanging metrological data such as features, tolerances, and measurement results (ISO. . . , 2011).

DMIS transmits product and equipment definitions together with the process information needed to perform dimensional measurement using CMM. DMIS contains the product definition for nominal features, construction of features, dimensional and geometric tolerances, functional datum, and part coordinate system; it also communicates equipment parameters with various sensors. Measurement, capabilities, and parameters of a machine. DMIS generates the DME measurement and movement instructions to check the conformity of a product and validate the control of the manufacturing process. Besides, DMIS guides the analysis of coordinate data to report and mark the results of measurements that determine the product's quality or process (ISO. . . , 2011).

To assist implementation, DMIS defines functional subsets of DMIS applications that ensure interoperability and validate compliance with DMIS. Also, DMIS addresses the associativity of DMIS product definitions with computer-aided design (CAD) information (ISO..., 2011).

DMIS was originally developed by CAM-I (Computer-Aided Manufacturing-International), and its version 2.1 in 1991 became an ANSI standard. The standard's continuity was guaranteed by the DMSC (Dimensional Metrology Standards Consortium). DMIS provides declarations, functions, and instructions for implementation (HORSFALL, 2007).

A DMIS language is based on a large quantity of commercial software for CMM. DMIS is considered a neutral language for communication between information systems and CMMs. As a basis for interoperability to DMIS is limited between CMMs, mas or DMIS can function as an input program format that encodes necessary instructions for CMMs to execute automated measurement actions. Alternatively, control the CMM machine interprets the high-level DMIS instructions in low-level movements (ZHAO et al., 2012).

DMIS is the only standard that combines measurement features and operating instruction information within the same measurement process definition. It is a language for controlling dimensional measurement equipment, which includes an input and an output language. The output language serves as an `log` of commands, action settings and results report with: data, features and real and nominal point tolerances. However, it does not define complete features of measurement equipment. Measurement equipment resource data is needed to complete the effectiveness of DMIS (HOCKEN; PEREIRA, 2016; ZHAO et al., 2012; HORSFALL, 2007).

2.4.2 ISO 14649— Part 16: Data for touch probing based inspection

This part of ISO 14649 specifies the technology-specific elements and data required for shape and dimensional inspection tasks. This standard is based on other protocols ISO 14649 and ISO 10303, required as NC process data for machining. The general process data described in ISO 14649-10 describes the interface between a computer numerical controller and the inspection programming system. It can be used for inspection operations on all types of machines, whether they are specific measuring machines or for in-line inspection during the machining process on milling machines, machining centers, or lathes with tools capable of automatic positioning.

Part 16 is a document that seeks to integrate the inspection process into the ISO 10303-238 data structure. The standard provides definitions for the phases of medication within the manufacturing program. The measuring medication in part 16 is limited to the IMO inspection carried out with a tactile sensor using the same fabrication machine. The functions provided are essential, but the possibilities of feedback, storage, and control in real-time create other challenges (XU; NEE, 2009; NEWMAN et al., 2008).

The Figure 2.3 shows the data structure defined in ISO14649-16 for the touch_probing entity interconnected with the ISO 10303-238 data structure. The touch_probing entity is defined within the workingstep entity. The definition of inspection operations and measurement strategies is beyond the scope of the document. Besides, other contactless inspection methods, such as manual and optical measurement techniques, are within the defined range of ISO14649-16. Part 16 needs more studios to measure operations of the ex-

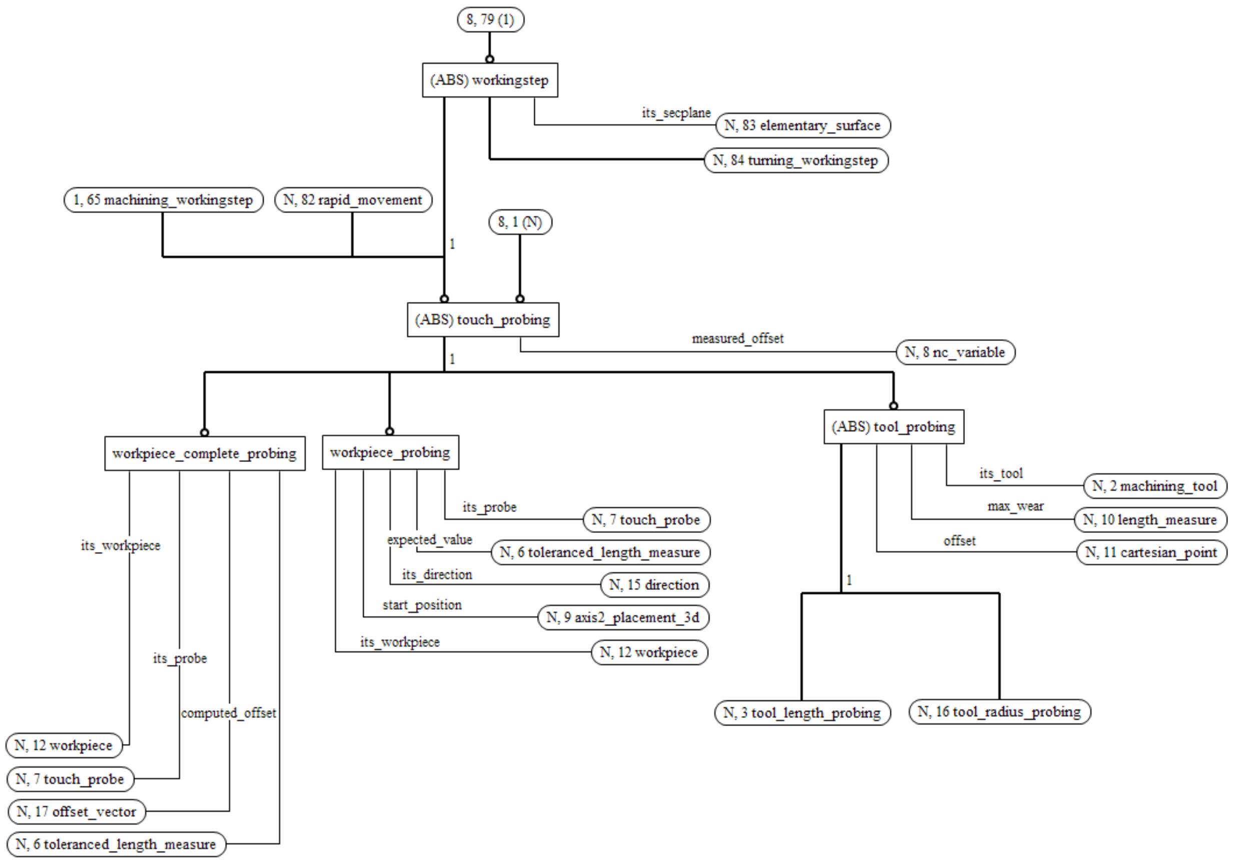


Figure 2.3: ISO 14649-16 data model within ISO 10303-238 data structure

act coordinates with other turning and milling operations. The focus of Part 16 is limited to the definition of tolerances, measurement tasks, and inspection features (ZHAO et al., 2008a).

2.4.3 QIF (Quality Information Framework)

Complete cyber-physical systems are needed to compile digital manufacturing data and support operations to guarantee quality. Methods are needed to organize and associate quality information, including measurement plans, results, pie geometry, models, resources, statistical analysis. The QIF (Quality Information Framework) is an ANSI standard that supports the range of digital concepts in engineering applications that encompass: product design, manufacturing, and quality inspection. The XML standard (Extensible Markup Language) contains libraries of XML schemas, which guarantee the integrity and interoperability of information based on corporate implementation models (DMSC, 2015).

Complete cyber-physical systems are needed to compile digital manufacturing data and support operations to guarantee quality. Methods are needed to organize and associate quality information, including measurement plans, results, pie geometry, models, resources, statistical analysis. The QIF (Quality Information Framework) is an ANSI standard that supports the range of digital concepts in engineering applications that encompass: product design, manufacturing, and quality inspection. The XML standard (Extensible Markup Language) contains libraries of XML schemas, which guarantee the integrity and in-

teroperability of information based on corporate implementation models.

The QIF standard defines an integrated set of information models that enable the exchange of metrology data throughout the entire measurement process for manufacturing quality, from product design to inspection planning (MAHMOUD, 2016). The QIF information models are contained in files written in the XML Schema Definition Language (XSDL). Additional restrictions to guarantee the data's quality are contained in the files recorded in the XSLT format (EXSensible Stylesheet Transformation) (DMSC, 2015).

At the core of the QIF architecture is the reusable QIF library that contains definitions and components referenced by the application areas, thus ensuring interoperability and extensibility. Features of the QIF standard are harmonized with the definitions of the ASME Y14.5 measurement Features, and definitions of QIF measurement Features are mapped to DMIS Features. The exchange of quality information and measurement data models is based on the standard structures already developed for manufacturing applications, such as the STEP standard. QIF is relevant to several of the ISO 10303 standards, especially STEP AP242 and AP219. The STEP structure generates a quality data model that corresponds to the standard data flow within the product's life cycle. This strategy supports the general manufacturing system giving interoperability and direct applicability in the resulting data models (DMSC, 2015).

The QIF information flow starts with the generation of CAD + PMI data `product manufacturing information` exported as QIF MBD application data `Model-Based Definition`. Quality planning systems import the MBD and generate both plans `whats` and `resources`, `rules`, and export plans `whats` and `hows`. Programming systems import plans to generate DME-specific execution programs or generate instructions to guide the inspection. The dimensional measurement equipment runs the programs, evaluates the manufactured or assembled part's characteristics, and exports the measurements. The analysis system, which is typically a statistical process control, imports simple results in parts to generate a multi-part batch analysis as QIF statistical data (DMSC, 2015).

2.4.3.1 Information and Application Model QIF

The QIF includes information models for six different application areas related to manufacturing quality. The data flow in the scheme shown in Figure 2.4 shows the five relevant activities in the measurement process, which are:

- Product definition.
- Determine measurement requirements.
- Define the measurement process.
- Perform the measurement process.
- Quality data analysis and reporting.

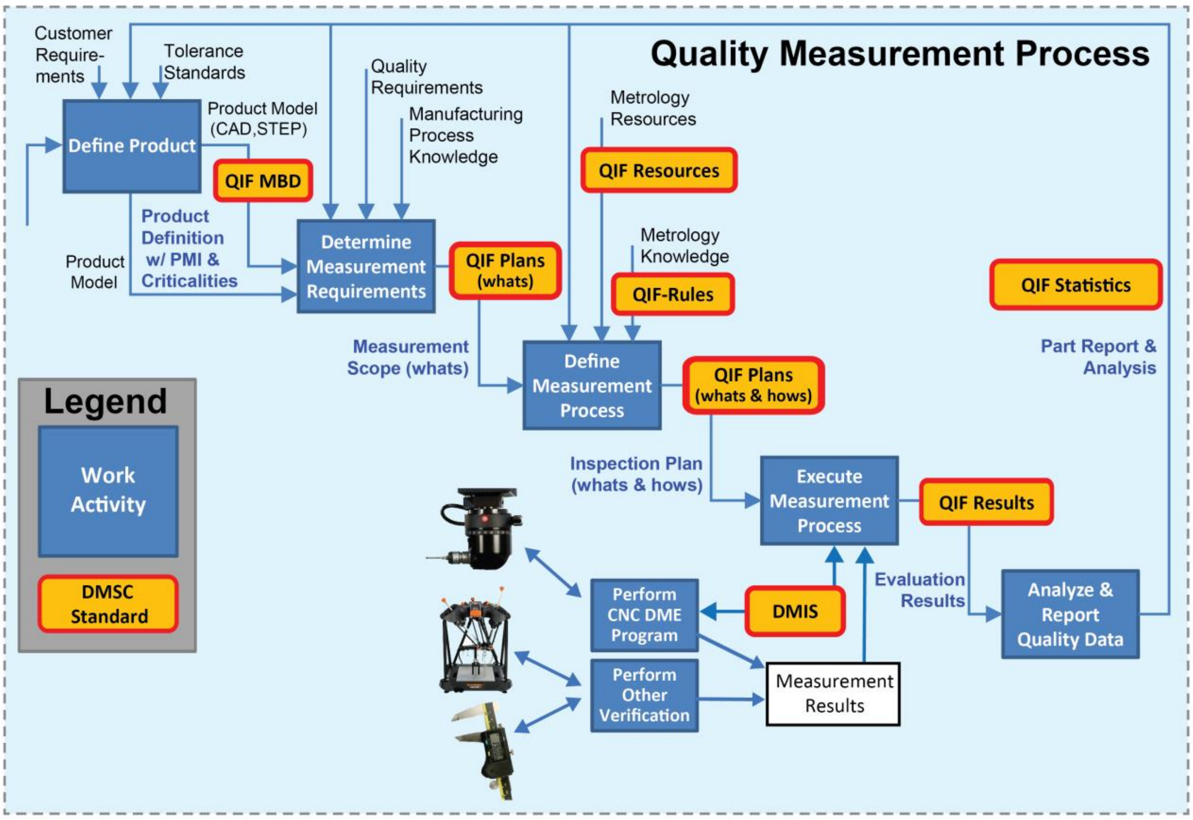


Figure 2.4: QIF Model based on information flow
Source: Adaptation (DMSC, 2015)

2.4.3.2 XML implementation

The QIF standard provides a formal set of rules for interacting with the data structure. With the advantages obtained from the implementation in XML language and the checks made available by the Extensible Stylesheet Language Transformations (XSLT) language, the QIF standard transfers a large part of a load of quality data to determine the syntactic and semantic validity elements. This verification of the structure of an instance file according to the QIF model can be done with automatic tools, free or low cost. Four checks of information quality in documents of QIF instances are available through the use of XSLT. The files that contain the XSLT checks are organized into a tree of five files with `Check.xsl` as the `root.Check.xsl` imports `CheckFormat.xsl`, `CheckQuality.xsl` and `CheckSemantic.xsl`. Each of these three files, import `CheckLibrary.xsl`.

QIF 2.1 contains two-level redundancy checking. For the low level, each type of element that describes a list, set, or collection includes a mandatory attribute "n", which is the number of items contained in the list, set, or collection. If any of the low-level "n" counts do not match the actual number of items in a list, set or the collection or the optional `ValidationCounts` element is present, there is an internal inconsistency in the QIF document, and the document is not valid.

A model is built using XML schemas, but the complete model can consist of items with information from several schema files. The complete model is defined using a top-level (or root) schema file that uses subordinate schema files. In XSDL, definitions in other files are indicated by an include directive, and child files can include other children.

2.5 STEP Compliant Deployment Environment

This section introduces the working environment used in the deployment. JSDAI is a solution that enables the construction of code within the object-oriented programming perspective, and that operates on the Java development environment. The tool is presented within a useful tool approach to generate solutions aimed at integrating manufacturing data with inspection and interoperability between related systems in the different phases of the life cycle.

2.5.1 JSDAI Implementation Environment

JSDAI is a set of tools and libraries for developing applications based on the ISO 10303 standard. The tools can be used for any other data structure modeled in the EXPRESS language (ISO 10303-11), PLIB (ISO 13584), and `IEC61360- Standard_data_element_types`. The tools are aimed at developers with knowledge of software, data structure, and familiarity with the STEP standard models.

JSDAI implements the Standard Data Access Interface standard (SDAI, ISO 10303-22) (See 3.4.2), mainly interconnected with the Java programming language for Standard Data Access Interface (SDAI), such as Internet/Intranet extensions. The SDAI standard, like Application Programming Interface Application Programming Interface (API), provides a mechanism for working with STEP data (GHANTA, 2012).

The JSDAI work environment allows reading, write and execute data for object-oriented programming, defined by a model based on the EXPRESS language, providing mechanisms for:

- Establish bidirectional relationships between models and entities.
- Create quick access to specific properties and attributes of the instances.
- The connection between the application level and the logic level of the architecture.
- Support other APIs in parallel.
- Access to SDAI libraries.
- Map from ARM model to AIM models.
- Exchange application data in different formats STEP, XML, and SDAI.
- Store information in different repositories with the possibility of multiuser and remote access.

The EXPRESS data model is compiled to represent java classes so that entities are accessed similarly to java classes. The core of STEP standard libraries is accessible through JSDAI and customizable to generate specific application models compiled in java libraries and made available to similar applications (LKSOFT, 1999). Figure 2.5 shows the structure of the JSDAI kernel.

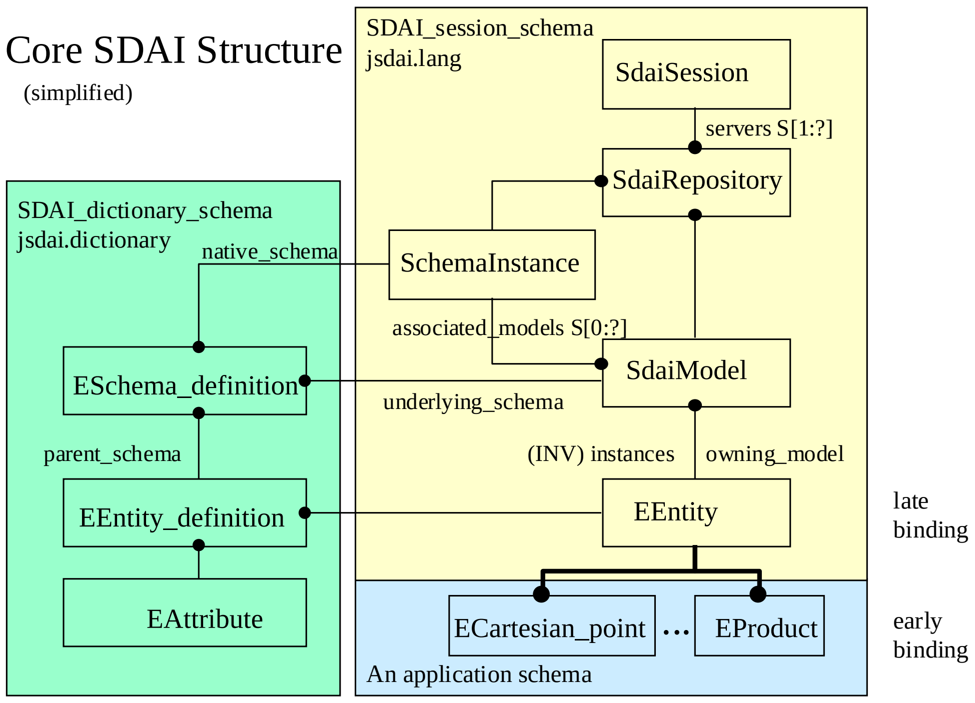


Figure 2.5: JSDAI Core Structure
Source: (LKSOFT, 1999)

The toolkit includes four components that can be used to develop custom STEP implementations. The JSDAI kernel contains a runtime platform combined with technologies and tools for development with support for early and late bindings of EXPRESS schemas.The main parts of JSDAI are:

- **JSDAI Runtime Platform**: It is an API to work with the STEP architecture; the EXPRESS compiler JSDAI results are the base in charge of manipulating the application data through each EXPRESS schema. The JSDAI Runtime environment creates the `SDAI_dictionary_schema`, `SDAI_session_schema`, and SDAI operations, as defined in the ISO 10303-22 standard, as well as using extensions for p21 files, network access (see Figure 2.6), mapping operations, event support, and others.

  The EXPRESS JSDAI compiler converts the application schemas, both as a population of dictionaries for access late binding and a Java package `sdai.Sxxx` with interface and Java classes for access early binding.

- **JSDAI Schema Library**: This library provides packages in the form of jsdai.Sxxx, data dictionary for early binding java interface, and classes for EXPRESS schemas. The EXPRESS JSDAI compiler automatically generates the JSDAI library in jsdai_library.jar.

- **SDAI File**: Like formats such as STEP-File (p21) and STEP-XML (p28), the purpose of the SDAI format is to define a structure for exchanging data, defined in the EXPRESS language. The main difference from the SDAI format is given by the binary encoding optimized for fast reading and writing. The SDAI format is suitable for both the exchange and backup of **SdaiRepository**. Other

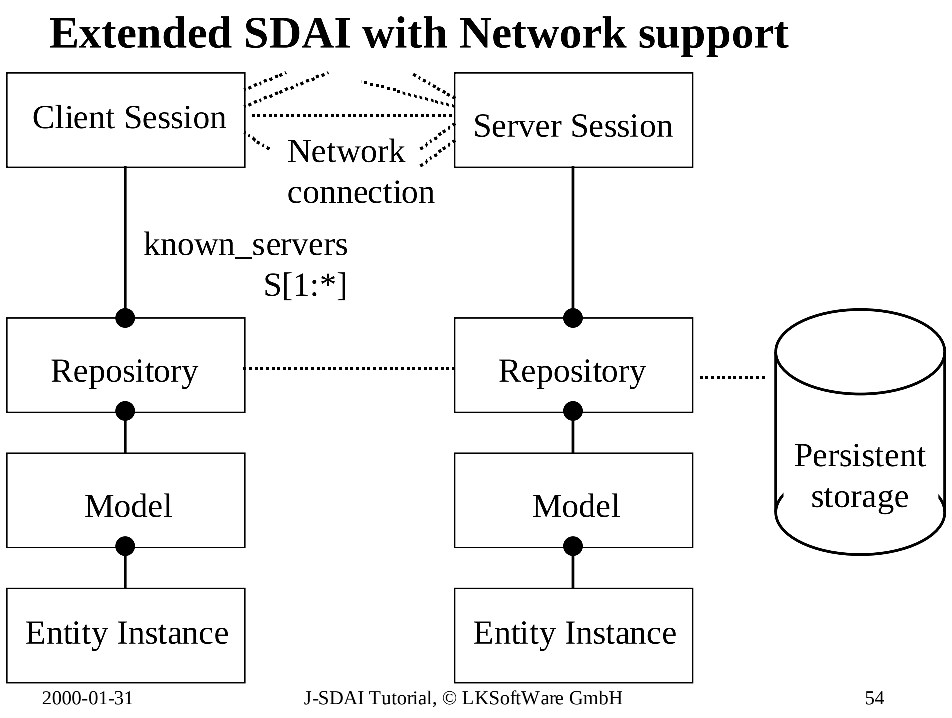


Figure 2.6: SDAI with support for network access
Source: (LKSOFT, 1999)

Table 2.3: Java packages for EXPRESS schemas

Java package	EXPRESS schema
jsdai.lang	SDAI_session_schema
jsdai.dictionary	SDAI_dictionary_schema
jjsdai.Mapping	SDAI_mapping_schema
jsdai.SXxx	Application schema "Xxx" (AIM)
jsdai.MXxx	Application schema "Xxx" (ARM)

Source: (LKSOFT, 1999)

features of the format are built-in understanding (zip), detailed information on EXPRESS schemes, integration of external documents, and Open-documented formatting.

- **JSDAI Express Compiler**: The EXPRESS compiler's purpose is to create a valid schema for JSDAI from analyzing one or more EXPRESS schemas. This is done by generating the necessary dictionary information (sdai_dictionary_schema) and Java classes. The input to the compiler is the `SHORT NAMES` of the EXPRESS entities and the list of complexities. The most significant feature of the JSDAI EXPRESS Compiler is operating directly on reduced EXPRESS schematics.

- **JSDAI ExpressDoc**: Since the Express schemas are valid in the JSDAI environment, the Express-Doc tool can generate an HTML format representation of the EXPRESS schema, providing a resource for navigation and visualization of the schemas. In an integrated way, it provides navigation resources for types and subtypes of complex entities, a list of attributes and properties of related schemes, creates an index of all named data, lists the Java methods available for access early binding, and comments on each data type and attributes.

2.5.2 JSDAI for Eclipse

The eclipse platform provides a Java-based IDE integrated development environment. The plug-in allows users to create, edit and compile EXPRESS files through a graphical environment. The highlights of the JSDAI integration with Eclipse are:

- Create, edit and compile EXPRESS schematics.
- Develop JSDAI applications.
- Generate documentation (HTML) of EXPRESS schematics.
- Create EXPRESS-G schematic diagrams

JSDAI for Eclipse offers several features for working with EXPRESS schemas and application development. The features of JSDAI for Eclipse are:

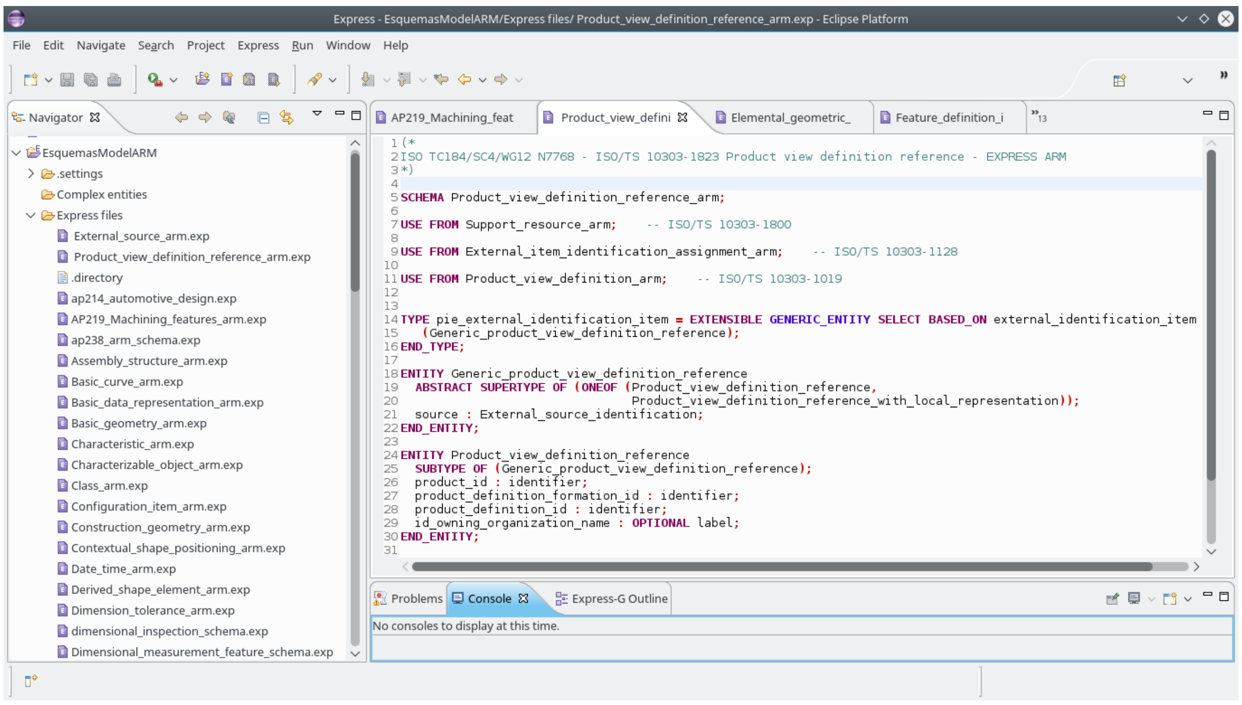
- EXPRESS editor.
- EXPRESS compiler.
- Editor EXPRESS-G.
- ExpressDoc.
- Java Express development.
- p21 Editor.
- Data validation.

Figure 2.7 shows the work environment of the EXPRESS editor. This tool allows semantic and syntactic verification of EXPRESS schemes (LKSOFT, 1999).

The project window is the environment that Eclipse enables so that JSDAI applications can be developed similar to a Java application. The EXPRESS-G editor (see 2.8) allows converting the EXPRESS schematics into a graphical representation of the entities relationships (LKSOFT, 1999).

Other additional JSDAI extensions that support functionality for STEP implementations are:

- The STEP application data can be stored and exchanged in several ways:
  - STEP File, ISO 10303-21
  - STEP XML, ISO 10303-28
  - SDAI File


Express - EsquemasModelARM/Express files/ Product_view_definition_reference_arm.exp - Eclipse Platform
File Edit Navigate Search Project Express Run Window Help
Navigator
AP219_Machining_feat
Product_view_defini
Elemental_geometric_
Feature_definition_i
EsquemasModelARM
.settings
Complex entities
Express files
External_source_arm.exp
Product_view_definition_reference_arm.exp
.directory
ap214_automotive_design.exp
AP219_Machining_features_arm.exp
ap238_arm_schema.exp
Assembly_structure_arm.exp
Basic_curve_arm.exp
Basic_data_representation_arm.exp
Basic_geometry_arm.exp
Characteristic_arm.exp
Characterizable_object_arm.exp
Class_arm.exp
Configuration_item_arm.exp
Construction_geometry_arm.exp
Contextual_shape_positioning_arm.exp
Date_time_arm.exp
Derived_shape_element_arm.exp
Dimension_tolerance_arm.exp
dimensional_inspection_schema.exp
Dimensional_measurement_feature_schema.exp
Problems
Console
Express-G Outline

```
(*
ISO TC184/SC4/WG12 N7768 - ISO/TS 10303-1823 Product view definition reference - EXPRESS ARM
*)

SCHEMA Product_view_definition_reference_arm;

USE FROM Support_resource_arm;    -- ISO/TS 10303-1800

USE FROM External_item_identification_assignment_arm;    -- ISO/TS 10303-1128

USE FROM Product_view_definition_arm;    -- ISO/TS 10303-1019


TYPE pie_external_identification_item = EXTENSIBLE GENERIC_ENTITY SELECT BASED_ON external_identification_item
   (Generic_product_view_definition_reference);
END_TYPE;

ENTITY Generic_product_view_definition_reference
  ABSTRACT SUPERTYPE OF (ONEOF (Product_view_definition_reference,
                               Product_view_definition_reference_with_local_representation));
  source : External_source_identification;
END_ENTITY;

ENTITY Product_view_definition_reference
  SUBTYPE OF (Generic_product_view_definition_reference);
  product_id : identifier;
  product_definition_formation_id : identifier;
  product_definition_id : identifier;
  id_owning_organization_name : OPTIONAL label;
END_ENTITY;
```

No consoles to display at this time.

Figure 2.7: EXPRESS editor supported by Eclipse JSDAI

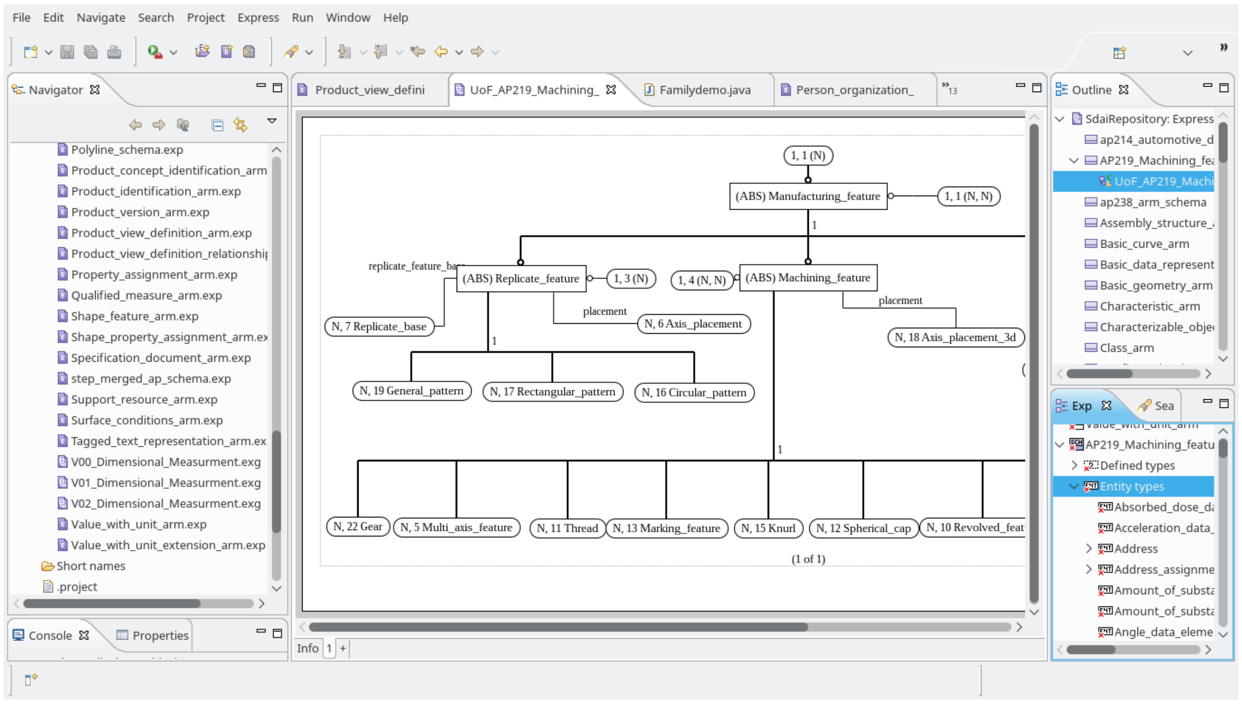


Figure 2.8: EXPRESS-G editor supported by Eclipse JSDAI

– Local repositories

– Remote Multi-User Repositories through JSDAI Database

• JSDAI takes advantage of SDAI concepts to organize and control application data with essential extensions:

- Separation of the physical repository (SdaiRepository) and logical grouping (Schema Instance) of application instances.
- Workspace concept (SchemaInstance + SdaiModel) to control access rights and logical grouping of data.
- Direct support of data distributed across multiple repositories for reuse across multiple workspaces.
- Explicit support for AP-Interoperability (between different schemes) using short EXPRESS forms.
- Full transaction support; control of confirmation and cancellation operations.
- To support multiple users. SDAI transactions are mapped to database transactions.

2.5.3 STEP-NC, JSDAI and Java

JSDAI is a set of tools and libraries for developing applications based on the ISO10303 standard. JSDAI is an implementation of the SDAI ISO10303-22 standard, strongly related to the JAVA programming language. In this project, JSDAI along with Java programming was used to support the implementation of the solution. The JSDAI core presented in Figure 2.9 is supported by a set of components that support application development to read or write STEP-Format data from or to STEP files. The components were described in section 2.5.1 and consist of the JSDAI Runtime platform, JSDAI Schema Library, SDAI File, JSDAI EXPRESS Compiler JSDAI ExpressDoc. This set of components plus various Part 21 extensions allow for consistent development of the solution.

Figure 2.9 presents the structure that supports object management in SDAI. `SdaiSession` is the starting root of all SDAI activity, `SdaiTransaction` controls the simultaneous update of changes, `SdaiRepository` is the physical storage container for `SchemaInstances` and `SdaiModels`. The `SdaiModel` groups of entity instance according to an EXPRESS schema. Other classes like `Schema Instance` are used for logical grouping models and defining a valid set of an EXPRESS schema. `Entity Entent` is a helper class for grouping entity instances of an entity data type. `SdaiException` is responsible for informing the execution status in case of error conditions. The entities of `SdaiSession, SdaiRepository, SdaiModel, SchemaInstance`, etc., in the `jsdai.lang` package do not operate like regular entities. Only specified access and manipulation methods are available. Dictionary and mapping entities are generally only accessible by early binding.

The terms early binding and late binding define how entity instances, their attributes, and relationships behave. Early binding works with specific entity types and attributes known at compile-time, late binding works with relative entity types and attributes specified with parameters at run time. Both types of binding require the EXPRESS definitions implicitly. For early binding, this is done by converting an EXPRESS schema into Java classes and interfaces; then, they are bound to the project within the Java programming environment via the Java Compile Path. However, for late binding, the EXPRESS specification becomes a set from the `jsdai.dictionary` package. The most important entities in the dictionary schema required for late binding access are `schema_definition`, `entity_definition`, and attribute.

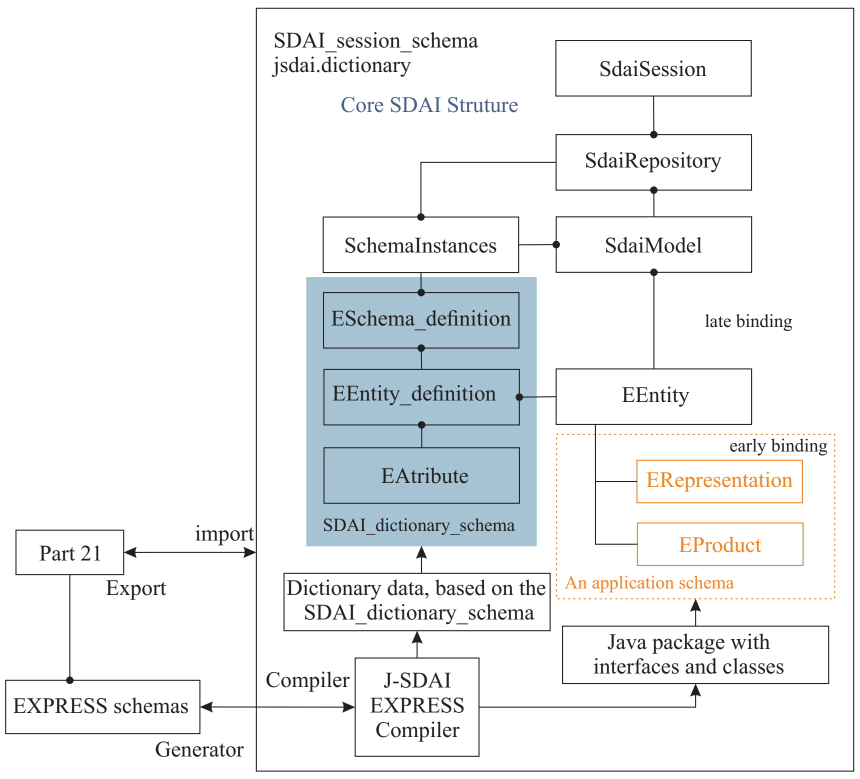


Figure 2.9: Core SDAI Structure.
Source: Adaptation (LKSOFT, 1999)

JSDAI supports early and late binding for application entities in parallel. Most of the applications addressed in our case study are identified in the ARM models and are specific entities. For this application, early link scheduling is addressed. Late binding programming generally requires working with dictionary instances and application instances in parallel.

2.5.4 Alternative Tools for Feature Extraction

An alternative for the development of solutions with objectives within a particular context, such as the particular case of extracting characteristics and dimensional specifications from a CAD model, in addition to the already exposed JSDAI API, there is STEPTools as an option to support framework development.

2.5.4.1 STEP Tools,Inc

STEP Tools, Inc. Started in 1990 as a worldwide community of experts to develop protocols for sharing product model data between companies. In the second phase, from 2000 to 2015, they create a series of software alternatives for manipulating product models. Currently, it is possible to share models in the digital chain of measurable products in real-time. STEP Tools, Inc provides a connection for machine tools, simulation servers known as digital twins that allows real-time analysis and correction of machining results.

2.5.4.2 ST-Developer

ST-Developer is an SDK (Software Development Kit) that allows the construction of STEP and EXPRESS applications, in addition to adding support for XML files, STEP Part 28, CIS / 2 (CIMsteel Integration Standard), IFC (Industry Foundation Classes) and different STEP application protocols. ST-Developer contains a wide selection of C ++ and Java class libraries, with schematics documented in HTML, which support the application protocols, AP203, edition 3 of AP224, AP219, AP221, AP236, AP238, AP239, AP240, and IFC 2x3.

2.5.4.3 ST-Developer for Java Programming

ST-Developer has a Java programming environment that consists of an EXPRESS compiler that can generate JAVA classes and a set of executable base files (stdev.jar), which provide functions such as reading and writing instances for the exchange of STEP part 21 files. Java applications can be developed around EXPRESS information models. Using the express2java compiler, Java classes for each model definition are generated, with which data sets can be created and manipulated.

2.5.4.4 Siemens NX for Manufacturing

Siemens provides a range of CAM, robotic machining, hybrid additive manufacturing, connectivity on the shop floor, and inspection design and programming tools. In inspection solutions, Siemens provides an interface that allows programming the CMM machine to make measurements with minimum setup times. The programming of the CMM in NC is oriented to use PH20 probes, visualize the probe head in the simulation and define the coordinate system, datum, and reference points for the measurement. The analysis of measurement data in NX CMM is programmed through a graphical interface that shows the part results.

With the Siemens PLM Software, and integration between manufacturing and inspection is possible. The NX ™ software's CMM Inspection provides a solution for offline programming. In the NX environment, project information, data analysis, and measurement results interact, providing a study environment. The process includes defining features, generation of trajectories, and validation to end with analyzing measurement data. The model's Product Manufacturing Information (PMI) information includes GD&T definitions and 3D annotations, which are the central elements of both manufacturing and inspection programming. The process can become automatic if a specific programming method is applied, including standardized trajectories, tools, and models.

2.6 Integrated Inspection System STEP-Compliant

This section presents an alternative solution, framed in the concept of interoperability, to integrate the results obtained in the measurement of a part with product data in the design, planning, and manufacturing phases. The measurement results are homogenized both semantically and syntactically to be handled by each CAx system. Dimensional metrology data is used to create a compensation strategy and correct manufacturing errors identified in the inspection process. An analysis is performed based on a case study that involves the definition and extraction of design specifications used for both manufacturing and measurement. The solution presented allows the use of measurement data within a closed manufacturing loop, ensuring the integrity of the information.

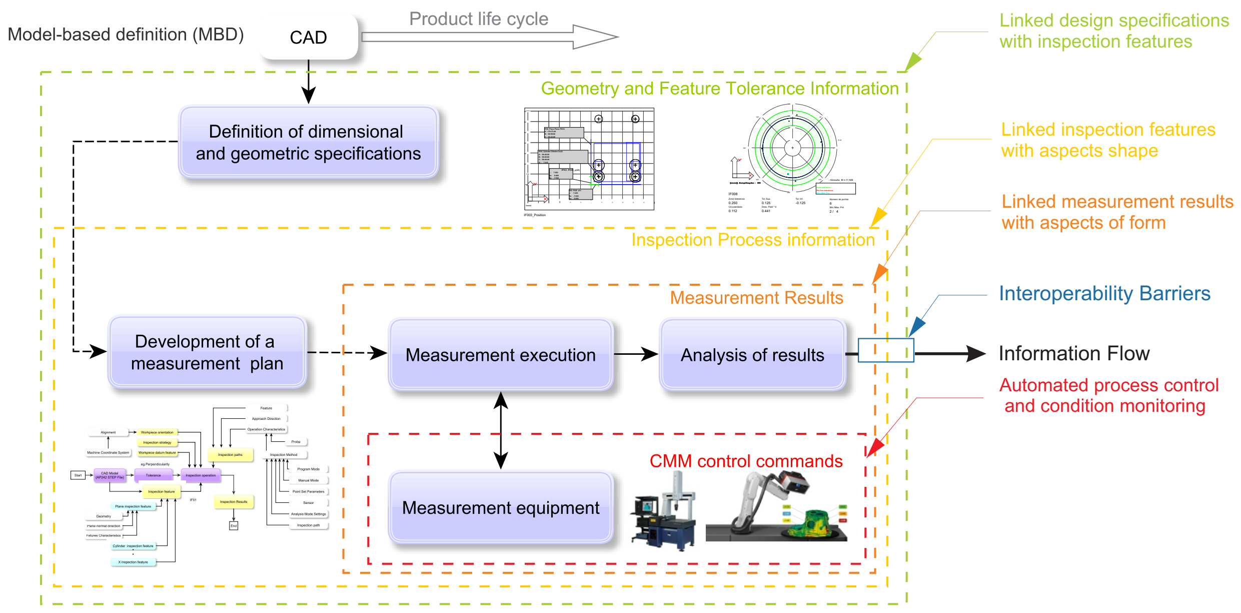
2.6.1 Interoperability Barriers of Measurement Systems

In the advanced manufacturing environment, complete process integration is created to cooperate and respond in real-time to particular demands and production specifications (DAVIS et al., 2012). Within this context, the dimensional and geometric inspection processes open their way as primary verification and validation tools for the specifications of a product and sources of information for the manufacturing environment (ZHENG et al., 2018). The measurement generates data that adequately processed information on both the manufacturing conditions and the already conventional parameters used in the quality control derived from comparing the current model with the nominal model(LU; XU, 2019).

The fourth industrial revolution demands that the inspection acquire data and extract useful information from it to improve the manufacturing process and meet the design requirements in manufacturing parts (LASI et al., 2014; RAY; JONES, 2006; BRUNNERMEIER; MARTIN, 1999). Different processes such as additive and subtractive manufacturing can use the data extracted from dimensional and geometric inspection to optimize processes. In addition, state-of-the-art concepts such as the digital twin, real-time monitoring, automatic compensation, and traceability also require measurement data. Together, these technologies seek to obtain a balance of productivity and quality that promotes the sustainability of the processes (BORTOLINI et al., 2018; ZHENG et al., 2018; CAMPOS; HARDWICK, 2006).

Inspection processes face several limitations, some of which are shared with other CAx systems (computer-aided technologies) that prevent complete integration within Closed-loop manufacturing (CLM) (BRECHER et al., 2006; ZHAO et al., 2011). Usually, the measurement system comprises four major phases: definition of dimensional and geometric specifications, unfolding of a measurement plane, measurement execution, and dice analysis. As a result, interoperability barriers they find between each stage of the inspection process (QIN et al., 2015). Figure 2.10 shows the main interoperability barriers faced by measurement systems(HEDBERG et al., 2016; XU; NEWMAN, 2006). As shown in Figure 2.10, the main problem affecting interoperability is the lack of homogeneous data related to the information generated in each phase of the inspection process. These barriers that still exist slow down the complete integration of the inspection processes with the CAx systems that make up a product's life cycle (MORONI; PETRÒ, 2018; GALLAHER et al., 2002).

In the last ten years, significant progress has been made to enable the use of information in the different phases of manufacturing, these results being adopted by some standards such as STEP, QIF, and MTConect (ISO 10303-242, 2014; ISO 10303-238, 2007; VIJAYARAGHAVAN et al., 2008). Recent projects, studies, and tests of the concept demonstrated that the implementation of standards with product definition based on Model-based definition (MBD) reduces the time to go from the design phase to manufacturing, impacting not only the consumption of resources but also the time spent also improving the quality of the final part (RUEMLER et al., 2017; GUNASEKARAN et al., 1994).



Model-based definition (MBD)
CAD
Product life cycle
Linked design specifications with inspection features
Geometry and Feature Tolerance Information
Definition of dimensional and geometric specifications
Linked inspection features with aspects shape
Linked measurement results with aspects of form
Inspection Process information
Measurement Results
Interoperability Barriers
Development of a measurement plan
Measurement execution
Analysis of results
Information Flow
Automated process control and condition monitoring
CMM control commands
Measurement equipment

Figure 2.10: Interoperability barriers of measurement systems
Source: (Riaño Jaimes; ALVARES, 2019)

It is still challenging to find a neutral, standardized, and interoperable data structure that covers all phases of the inspection process and facilitates the feedback of results within closed-loop manufacturing. This line seeks to create a solution alternative implementing a closed-loop inspection to cover all phases of the inspection process using a neutral data structure that homogenizes information semantically and syntactically to integrate the measurement results with manufacturing data (ZHAO et al., 2008b; XU* et al., 2005b).

2.6.2 Closed-loop Inspection

The measurement was included in the manufacturing processes to verify that the manufacturing results are within the projected specifications. Dimensional Metrology Data evolved to support different tasks such as surface reconstruction and deduction of mathematical models. The new approaches such as Digital Twin and the manufacturing challenges created with the digital era and Internet of Things (IoT) depend to no small extent on the information deduced from the measurement process and the existing communication between each system involved in the production life cycle of a part. Currently, these processes present integration in the application layer, plus the interoperability barriers in the logical and physical layers of the architecture persist (LU et al., 2015; XU* et al., 2005b).

Figure 2.11 presents the functional model Integration DEFinition language 0 (IDEF0) that shows the flow of information through the dimensional and geometric inspection system's four functional activities. The objective is through a neutral data structure to guarantee the integrity of the information and allow real-time access to the data within a closed-loop inspection. The architecture creates a global data model integrated semantically and syntactically using neutral standards such as the Standard for data exchange of the product model (STEP) and the Quality Information Framework (QIF).

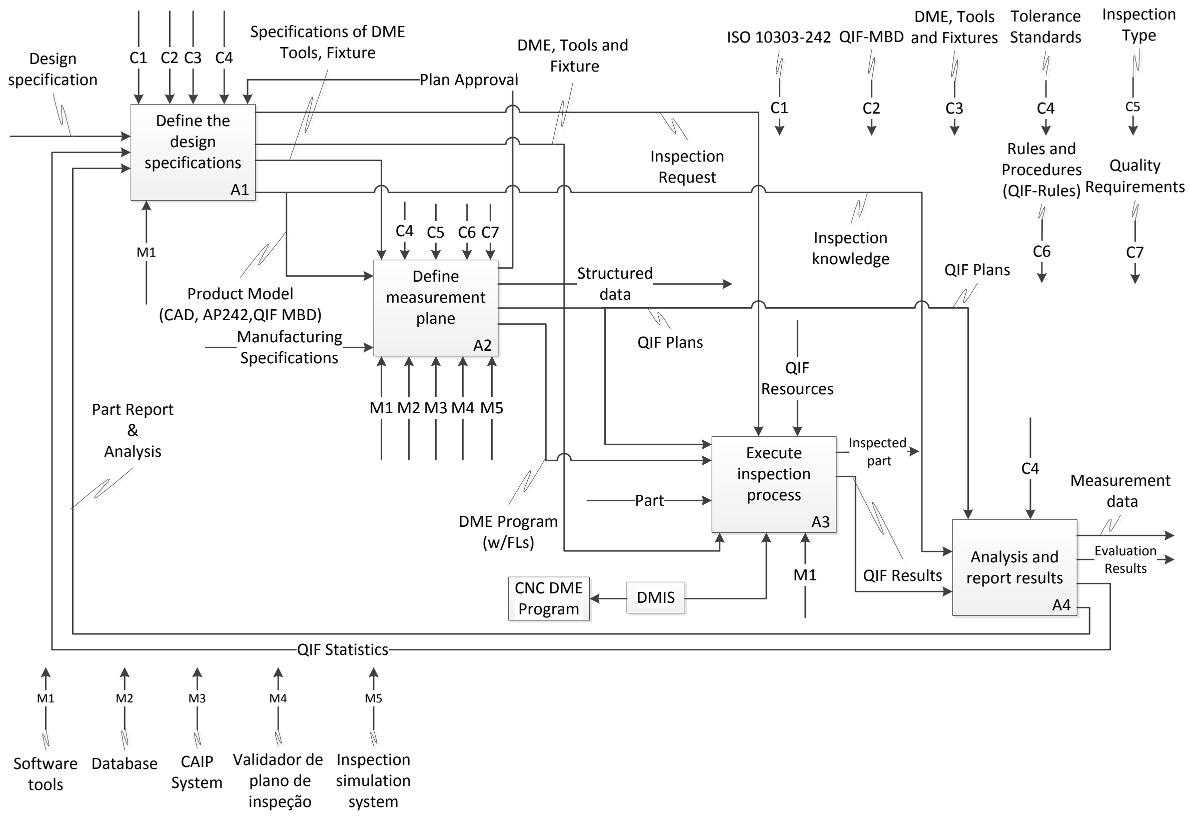


Figure 2.11: IDEF0 Diagram to detail the inspection system
Source: (RIAÑO et al., 2017)

In this architecture, the 3D Digital model, considered the fundamental basis of integration, contains both a solid and metadata with GD&T design specifications. This model is reconstructed and updated with the information contained in a file encoded in ASCII code. Through its application protocols AP203, AP214, and AP242, the STEP standard structures the product information and shares it with other systems adhering to the standard. Similarly, the QIF MBD definition model has a digital data format that includes information necessary for quality processes and PMI (RIAÑO et al., 2017).

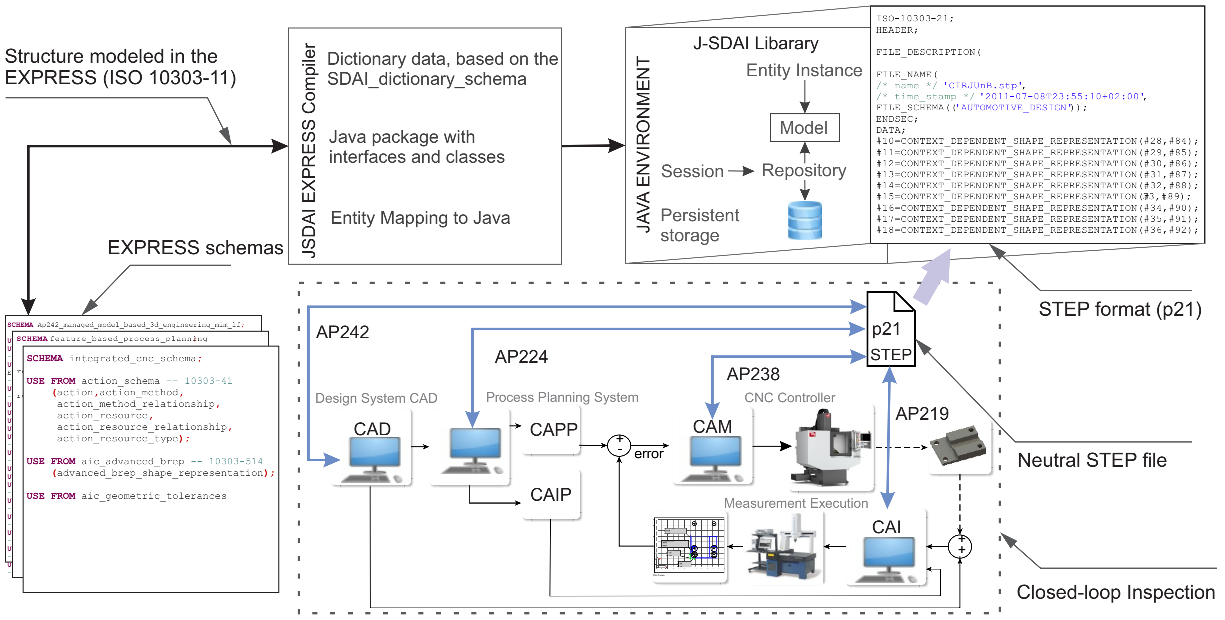
In Figure 2.11, the functional block (A1) extracts the geometry of a part and the design specifications that need conformity verification. According to the given specifications, the selection of equipment and tools to perform the measurement are received by the functional block (A2) responsible for executing the inspection plan. This block (A2) uses data from the piece's manufacturing process to include the data structure information about equipment, tools, configurations, and program to perform the dimensional measurement. The functional block (A3) contains the flow of information for the execution of the inspection. Finally, the functional block (A4) analyzes and generates a QIF format result for feedback, storage, and sharing within an integrated manufacturing environment.

2.6.3 Method for Interoperability of Measurement Data

It is necessary to relate the manufacturing decisions and the impact obtained in the results and create a mechanism to identify the causes that produce the variation in the manufactured parts. A subsequent analysis will allow the generation of corrective actions to compensate for the error. In the current interoperability context, most scenarios seek the exchange of information without solving the underlying problems that create the barriers, thus promoting a technological dependence to implement integrated and interoperable solutions.

To solving the interoperability problem, it is necessary to use mechanisms to structure the information. The EXPRESS language specified in part ISO 110303-11 is a description method used in the STEP standard to facilitate the interpretation of information models and application protocols (see 3.3.1). The EXPRESS is designed to be used independently of technology as an information modeling language, allowing and representing data, defining restrictions, rules, functions, and procedures. Within language, the focus is the definition of entities that represent objects. Each entity is defined in terms of its properties, which are characterized by domain specifications and restrictions.

JSDAI is an Application Programming Interface (API) for read-write and run-time manipulation of object-oriented data defined by an EXPRESS data model (See 2.5.1). The JSDAI tools. It is used with any data structure modeled in the EXPRESS (ISO 10303-11), PLIB (ISO 13584), and IEC61360 Standard data element types language (PAN et al., 2005). JSDAI can store and exchange data with STEP applications in various forms such as STEP XML - ISO 10303-21 (part21), STEP XML - ISO 10303-28 STEP XML, among other formats. JSDAI can organize and control application data, such as physical separation in repositories and logical grouping between instances and schemas (LI et al., 2011).



```
ISO-10303-21;
HEADER;

FILE_DESCRIPTION(

FILE_NAME(
/* name */ 'CIRJUnB.stp',
/* time_stamp */ '2011-07-08T23:55:10+02:00',
FILE_SCHEMA(('AUTOMOTIVE_DESIGN'));
ENDSEC;
DATA;
#10=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#28,#84);
#11=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#29,#85);
#12=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#30,#86);
#13=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#31,#87);
#14=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#32,#88);
#15=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#33,#89);
#16=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#34,#90);
#17=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#35,#91);
#18=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#36,#92);
```

```
SCHEMA Ap242_managed_model_based_3d_engineering_mim_lf;
```

```
SCHEMA feature_based_process_planning
```

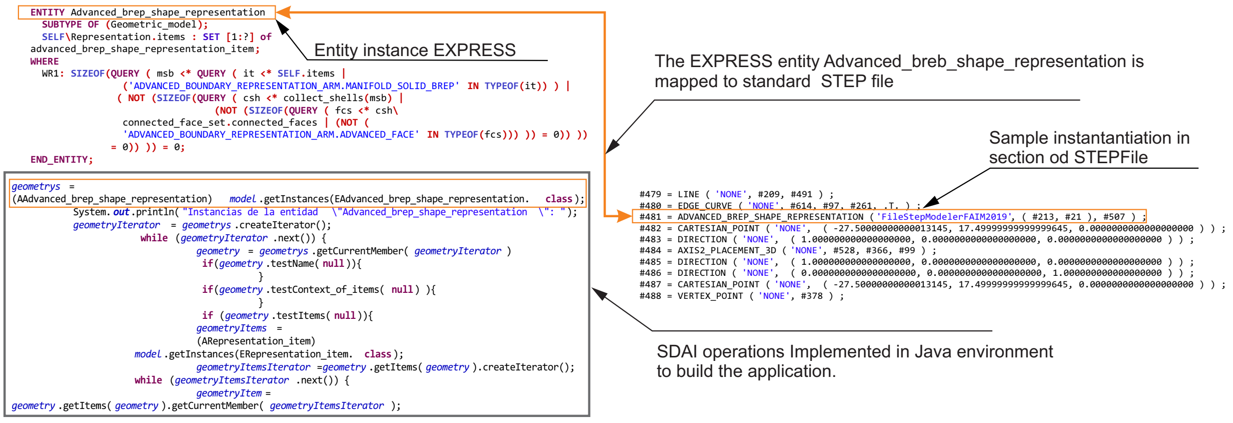
```
SCHEMA integrated_cnc_schema;

USE FROM action_schema -- 10303-41
    (action,action_method,
     action_method_relationship,
     action_resource,
     action_resource_relationship,
     action_resource_type);

USE FROM aic_advanced_brep -- 10303-514
    (advanced_brep_shape_representation);

USE FROM aic_geometric_tolerances
```

Figure 2.12: Diagram of the approach for interoperable data access.
Source: (Riaño Jaimes; ALVARES, 2019)

Figure 2.12 presents an integrated approach for transmitting specific information and based on harmonizing data generated in the manufacturing life cycle of a part. Interoperability is achieved by using a neutral language to work with the data provided by the STEP, DMIS, and QIF standards within closed-loop manufacturing. Access through different technologies is possible as there is a semantic and syntactic integration of the data exchanged between CAx systems due to the STEP standard's adoption in each of the systems. Most of the neutral standards that support the manufacturing processes converge in adopting the eXtensible Markup Language (XML) format as a mechanism for representing and transporting data with other environments, making possible the work with repositories and database.

Figure 2.12 shows the approach applied for reading, writing, and storing using a SDAI repository as a record in the dedicated database, leaving the responsibility of validating and processing this stored information directly to the application. This approach allows the data to be easily separated and integrable with existing data models in each system. Within an object-oriented programming environment such as JAVA, JSDAI libraries construct models when processing input files in STEP format (p21) (SOUZA et al., 2015). The STEP standard defines several structured methods to support the exchange of product information specified in the different application protocols, such as ISO 10303-21, ISO 10303-22, ISO 10303-23, ISO 10303-24. The Standard Data Access Interface (SDAI) defines a low-level interface for data programming defined in the EXPRESS language. This comprehensive set of SDAI operations in ISO 10303-22 are implemented in JAVA to build the application (See 3.4.2). These operations can also be implemented in specific programming languages such as C and C++.

An EXPRESS language data structure is created within the application based on the schemas of each application protocol wrapped in the digital chain. This structure allows importing into the JAVA programming environment the packages with interfaces and classes containing both the mapped entities and the data dictionary based on each STEP application protocol's schemas. Within the JAVA environment, each element is linked in the project and using the operations defined in SDAI to focus on developing the application. In this way, the functions and activities rules are available for reading and write operations within an object-oriented programming environment. The same data structure is used both to manipulate and to create new files fed with inspection results.


```
ENTITY Advanced_brep_shape_representation
  SUBTYPE OF (Geometric_model);
  SELF\Representation.items : SET [1:?] of
advanced_brep_shape_representation_item;
WHERE
  WR1: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items |
                ('ADVANCED_BOUNDARY_REPRESENTATION_ARM.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) |
               ( NOT (SIZEOF(QUERY ( csh <* collect_shells(msb) |
                                 (NOT (SIZEOF(QUERY ( fcs <* csh\
                connected_face_set.connected_faces | (NOT (
                'ADVANCED_BOUNDARY_REPRESENTATION_ARM.ADVANCED_FACE' IN TYPEOF(fcs))) )) = 0)) ))
              = 0)) )) = 0;
END_ENTITY;
```

```
geometrys  =
(AAdvanced_brep_shape_representation)   model .getInstances(EAdvanced_brep_shape_representation.   class );
          System. out .println( "Instancias de la entidad  \"Advanced_brep_shape_representation  \": ");
          geometryIterator  = geometrys .createIterator();
                   while  (geometryIterator .next()) {
                            geometry  = geometrys .getCurrentMember( geometryIterator )
                             if(geometry .testName( null )){
                                     }
                             if(geometry .testContext_of_items(  null) ){
                                     }
                             if  (geometry .testItems( null )){
                            geometryItems  =
                            (ARepresentation_item)
                  model .getInstances(ERepresentation_item.  class );
                            geometryItemsIterator =geometry .getItems( geometry ).createIterator();
                  while  (geometryItemsIterator  .next()) {
                            geometryItem =
geometry .getItems( geometry ).getCurrentMember( geometryItemsIterator  );
```

```
#479 = LINE ( 'NONE', #209, #491 ) ;
#480 = EDGE_CURVE ( 'NONE', #614, #97, #261, .T. ) ;
#481 = ADVANCED_BREP_SHAPE_REPRESENTATION ('FileStepModelerFAIM2019', ( #213, #21 ), #507 ) ;
#482 = CARTESIAN_POINT ( 'NONE',  ( -27.50000000000013145, 17.49999999999999645, 0.0000000000000000000 ) ) ;
#483 = DIRECTION ( 'NONE',  ( 1.000000000000000000, 0.0000000000000000000, 0.0000000000000000000 ) ) ;
#484 = AXIS2_PLACEMENT_3D ( 'NONE', #528, #366, #99 ) ;
#485 = DIRECTION ( 'NONE',  ( 1.000000000000000000, 0.0000000000000000000, 0.0000000000000000000 ) ) ;
#486 = DIRECTION ( 'NONE',  ( 0.0000000000000000000, 0.0000000000000000000, 1.000000000000000000 ) ) ;
#487 = CARTESIAN_POINT ( 'NONE',  ( -27.50000000000013145, 17.49999999999999645, 0.0000000000000000000 ) ) ;
#488 = VERTEX_POINT ( 'NONE', #378 ) ;
```
Entity instance EXPRESS
The EXPRESS entity Advanced_breb_shape_representation is mapped to standard STEP file
Sample instantantiation in section od STEPFile
SDAI operations Implemented in Java environment to build the application.

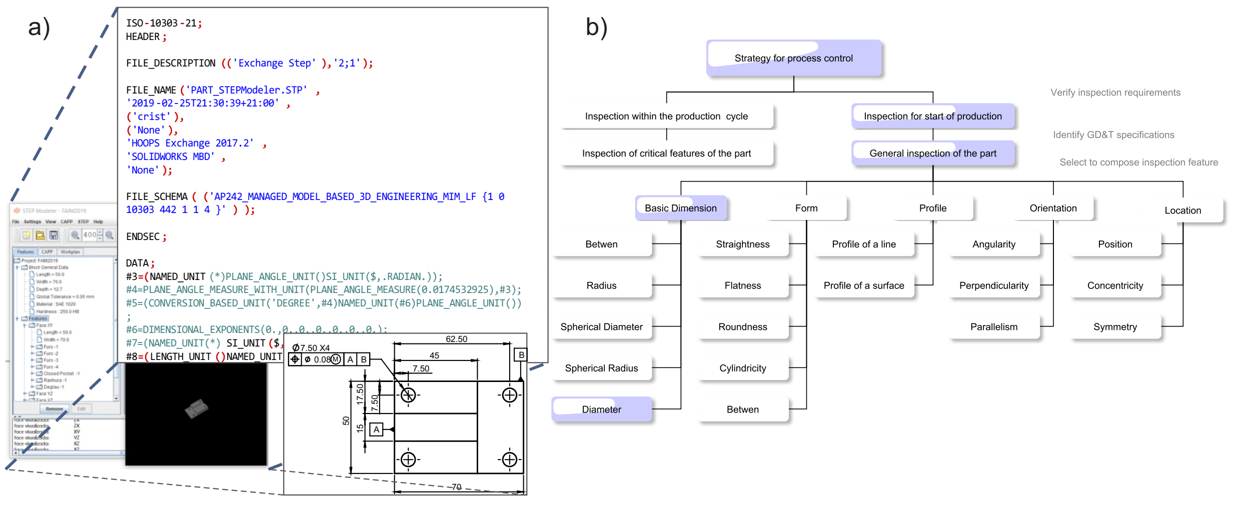
Figure 2.13: Code snippet showing the processing of the AP242 STEP File.
Source: (Riaño Jaimes; ALVARES, 2019)

Figure 2.13 shows an example of how the information generated in the inspection system is encoded in a STEP file compatible with the different CAx systems. The AP242 file contains the GD&T specifications of the part necessary to generate an inspection plane. The EXPRESS schemes include the set of entities, rules, and functions that support each application protocol; in our case, we use the AP219, AP238, and AP242 schemes. From the EXPRESS diagrams, dynamic libraries are created and used within the JAVA application to read, write, and update data in STEP files. These dynamic libraries allow the information generated in each system to be encoded within the same data structure, regardless of their nature. Figure 2.13 contains the code fragments to exemplify how each EXPRESS entity is mapped to a STEP file using a JAVA programming environment to perform these operations.

2.6.4 Integrated Inspection System architecture

The inspection data to be integrated is obtained using a coordinate measuring machine (CMM). The approach described in the previous section allows homogenizing the data within a neutral format to be shared in a digital chain. Some causes of the manufacturing process uncertainty results are defined in the knowledge base and are used within the quality control system to associate the errors with attributable causes. Figure 2.14 shows the scheme used to control the process. The objective of control is to stabilize the process and avoid the causes of errors. The measurement can be required both within a production cycle when it is desired to control the error of a specific critical feature, and at the beginning of new production, it is necessary to verify each characteristic of a part. The results are used to adjust the manufacturing environment.

Manufacturing features are linked to inspection features. Geometric inspection features include line, circle, arc structures, and dimensional inspection built from basic shapes. Figure 2.14 shows the scheme followed in composing an inspection function that allows verifying GD&T specifications.



```
ISO-10303-21;
HEADER;

FILE_DESCRIPTION (('Exchange Step' ),'2;1');

FILE_NAME ('PART_STEPModeler.STP' ,
'2019-02-25T21:30:39+21:00' ,
('crist'),
('None'),
'HOOPS Exchange 2017.2' ,
'SOLIDWORKS MBD' ,
'None');

FILE_SCHEMA ( ('AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF {1 0
10303 442 1 1 4 }' ) );

ENDSEC;

DATA;
#3=(NAMED_UNIT (*)PLANE_ANGLE_UNIT()SI_UNIT($,.RADIAN.));
#4=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(0.0174532925),#3);
#5=(CONVERSION_BASED_UNIT('DEGREE',#4)NAMED_UNIT(#6)PLANE_ANGLE_UNIT())
;
#6=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.,0.);
#7=(NAMED_UNIT(*) SI_UNIT ($,
#8=(LENGTH_UNIT ()NAMED_UNIT
```

Figure 2.14: Scheme for the design specifications control.
Source: (Riaño Jaimes; ALVARES, 2019)

The inspection is performed on a coordinate measuring machine (Mitutoyo Crysta-Plus M 574). The objective is to explicitly control the Circularity and Position features defined in the project shown in Figure 2.14 (a). The GD&T specifications are extracted following the drilling shown in Figure 2.14 (b). To measure, it is necessary to create a set of operations composed by inspection strategy, reference indications on the part, measurement trajectories, and orientation.

The procedure to define an inspection operation starts with the extraction of the design specifications of the AP242 STEP File. Then the inspection features associated with the tolerances of both circularity and position are created. For each feature, an inspection operation is designed. The workpiece orientation, inspection paths, and inspection strategy data are set up. Figure 2.15 presents the scheme to be followed and all the information required to configure an inspection operation.

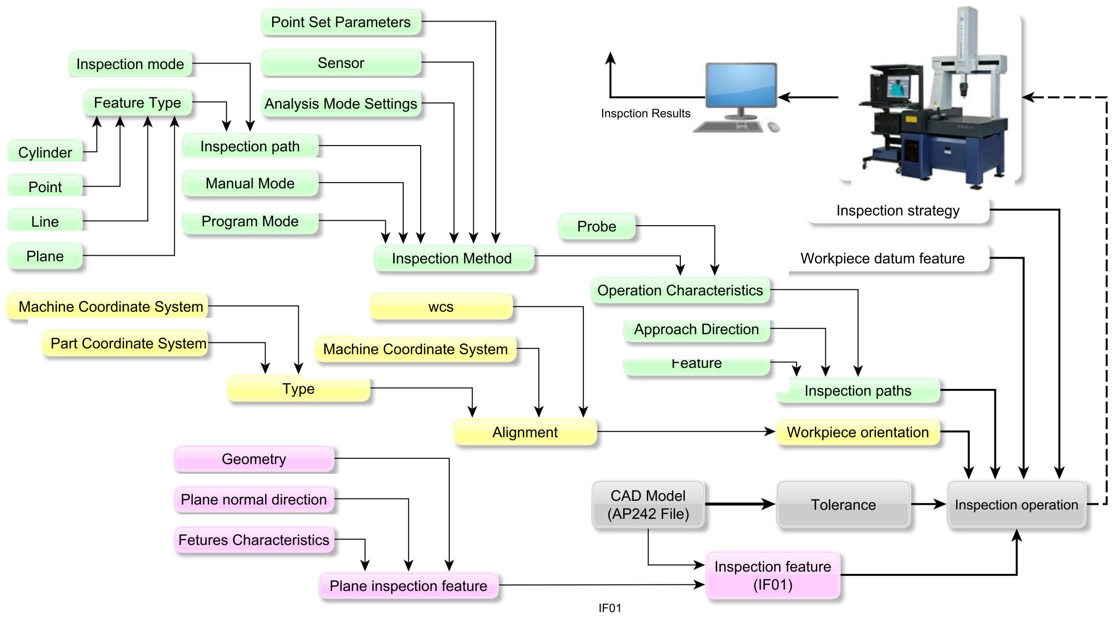


Figure 2.15: Procedure for defining inspection operations.
Source: (Riaño Jaimes; ALVARES, 2019)

Table 2.4 summarizes the data obtained in the measurement process, which needs to be analyzed to generate the corrective action according to the uncertainty value found. The amount is verified and compared to the design specifications. If the value is close to the nominal value and within the compliance zone, the function is accepted. When out of spec, it is necessary to determine the uncertainty ranges to apply the type of correction compensating for the upper or lower tolerance limit. The manufacturing tolerances are adjusted and stored in the execution schedule to be considered in the following part.

The measurement results obtained define new strategies that can be configured directly in the model MBD. Modifications in the design, adjustment of tolerances, and changes in the manufacturing process can correct the error obtained. The neutral structure allows encoding the results in different neutral formats and compatible with the other sites involved in the piece's life cycle. The created STEP file is interpreted in any CAD/CAPP/CAIP/CAM/CNC system adherent with ISO 10303-238, 242.224.

The technique uses neutral data structures, eliminating technological dependencies and enabling an alternative to overcome the interoperability barriers that hinder measurement results feedback. The solution presented in this section seeks to share homogeneous data within a compatible data structure for different CAx systems. The data is assigned using a standard to exchange standardized data that share the same

Table 2.4: Inspection feature definition

Inspection feature	Measuring feature	Tol.Type	Tol.Value (IT/mm)	Datum	Deviation	Oversize
IF02	Cylinder	Circularity	0.25	-	0.005	0.005
IF03	Cylinder	Position	0.08	IF01, IF06	0.388	0.308
IF05	Cylinder	Circularity	0.25	-	0.021	-
IF07	Cylinder	Position	0.08	IF01, IF06	0.373	0.293
IF08	Cylinder	Position	0.08	IF01, IF06	3.86	3.780
IF10	Cylinder	Circularity	0.25	-	0.008	-
IF12	Cylinder	Position	0.08	IF01, IF06	0.686	0.606
IF13	Cylinder	Circularity	0.25	-	0.020	-

syntax and semantics in each of the systems involved in the manufacturing cycle.

The architecture is projected on the models based on definition, involving STEP File, and looking to meet information requirements of measurement processes to integrate with technologies adherent to the 4.0 industry perspective. Industry 4.0 concepts promote the integration and interoperability of systems. The model presented is designed to create an architecture that integrates both the manufacturing and inspection processes in the new digital era.

2.6.5 Measurement Plan

The measurement plans contain all the information necessary to control the CMM machine to execute and evaluate the measurement. The inspection plan is essentially designed to assist measurement tasks through a procedure that establishes both an execution program and post-processing of data. The plan produces a set of measuring points and a list of features along a path to be inspected. The most relevant task in the plan is the extraction of information, which is hampered by the absence of a standard that harmonizes the definitions included in DMIS, I++ DME, AP219, ISO 14649, Part 16, DML, QMD, among other standards used in the measurement (XIA et al., 2011).

Figure 2.16 shows a model that defines the dimensional and geometric measurement features. Basic geometric features include line, circle, and arc structures. One or more geometric features make up dimensional inspection features. The Figure 2.16 represents through the diagram the hierarchy of organization of inspection features within the AP219 application protocol. At a more high level are the Units of Functionality, made up of various application objects containing entities. This hierarchical structure characteristic of the STEP standard allows defining procedures to access information contained in a STEP file (p21).

2.6.6 Feature Inspection

It is necessary to respect the restrictions of the measurement process given by the tools ability to position themselves on the points. The inspection plan tries to solve the problem using the knowledge base to generate measurement options, create an appropriate point distribution, select the appropriate probe (tool probe) and define the best trajectory based on the priority criteria, time estimation, and resource utilization.

The nominal geometry of the part that needs to be measured is contained in its features. The features

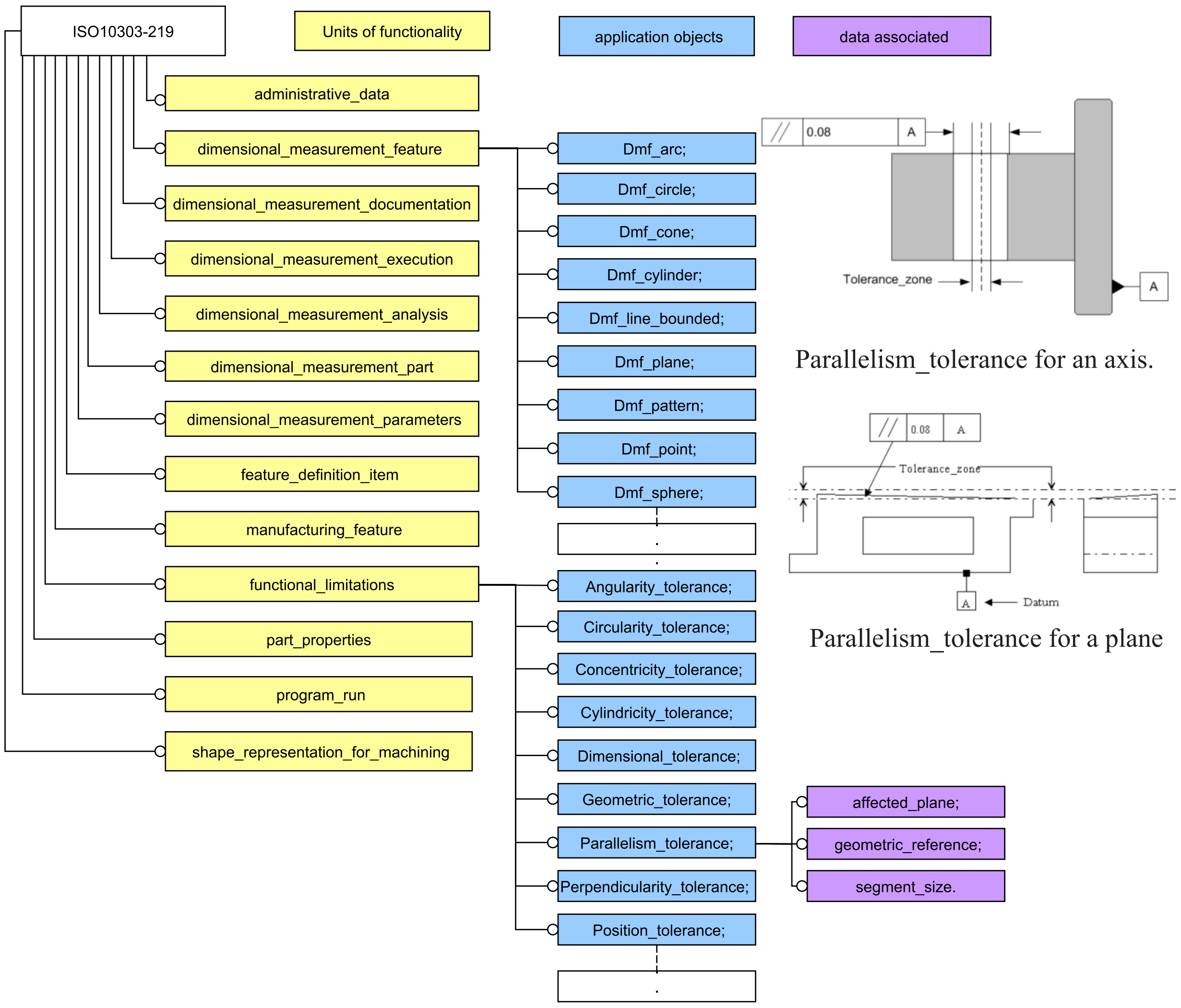


Figure 2.16: GD&T definitions AP219
Source:adaptation (ISO 10303-219, 2007)

have associated with their properties, information such as dimensions, shape, and position. There are several methods for defining features, some of which are listed below:

- `Automatic recognition of features`: Recognizes the geometry features through a Test function executed in the `CMM` and interprets it appropriately through a predefined feature model.
- `Definition of features using templates`: Within the measurement plane, a template of feature is filled. The method is commonly used when a measurement plan for similar features already exists or if programming is done outside the machine.
- `Definition of features extracting information from a CAD file`: Loading an existing CAD model, the features are extracted within the measurement plane. It helps build the measurement plan for the `CMM` machine in an offline manner and can define or hypothetically construct the features when in practice; they are not made available by the machine (e.g., include the measurement of an intersection of two features).

The inspection plan can be considered as a TSP (trade traveler problem). Assuming that a custom has N display positions, the distances between each position can be determined. A data is captured in each position, passing only once. The distance traveled in each displacement can be minimized by selecting a suitable tray that connects all the points to be measured in an optimal path.

2.6.7 Part Setup and Tool Probe Configuration.

In the probe configuration, the orientation, dimension, and structure are defined. A set of probes with different orientations and structures can be defined to meet the inspection requirements. The main issue to be resolved is identifying the probe's geometric constraints and finding the most significant number of faces of the part accessible with each probe. The previous analysis depends directly on the part's Setup, so it implies finding each Setup the most significant number of faces possible to inspect with the probe restrictions. If the inspection plan involves many probes and requires a different Setup, the measurement process can take more time and increase costs.

The problem of configuring the fixation and palpation system is addressed in parallel. It starts with the recognition of the piece's geometric model. It continues with an accessibility analysis that can include techniques based on artificial intelligence and ends with determining the complex palpation system that requires a spatial configuration of the piece in the CMM workspace.

2.6.8 Trajectory Planning

In inspection planning, the generation of the trajectory directly impacts the efficiency and performance of the measurement. The result of applying a sampling strategy generates a series of measurement points located on the part. This phase's problem is solved by finding an optimal trajectory for a probe that follows each of the points to be measured, avoiding collisions.

2.6.9 Inspection Strategies

Most assemblies are made up of multiple parts, which is a great challenge, especially for complex product development. It is necessary to identify the GD&T specifications of critical parts to ensure a proper fit. By identifying and classifying the different manufacturing process errors, uncertainty can be minimized by defining some additional constraints on the part design. Verification of these GD&T requirements occurs when manufacturing is complete; in this context, the importance of a closed-loop inspection control system to feedback the measurement results and strategies minimizes the final product error.

In 1982, ANSI introduced standards with specifications for geometric and dimensional tolerances. ANSI (1982) assumed that engineering drawings would be the only mechanism for communicating project specifications (SHAH et al., 1998). In 1994 ASME (The American Society of Mechanical Engineers) categorized tolerances into six classes: Size, Shape (flatness, straightness, circularity, cylindricity), Orientation (parallelism, perpendicularity, angularity), Position (location, concentricity), Profile (circular, total) and Finished (line, surface) (ASME Y14.5M, 1994).

Different investigations address strategies to correct errors in the manufacture of parts based on the inspection results analysis. Alvares (1990) proposes two scenarios for the analysis and automatic correction of geometric errors in machined parts in his feedback strategy. The first scenario consists of defining corrective actions for the parts at the beginning of production. The correction is made through the NC program and "altered" data, where the identified error is mainly caused by the mismatch of the machine with the errors in the programmed geometric definitions. The second scenario is for parts in the production cycle. Correction is made using tool data and "corrupted" data. The quality feedback system performs a tolerance analysis to determine the geometric elements in error. Each element was inspected and identified with the geometric definition of the corresponding element in the NC program. Figure 2.17 shows the analysis strategy feedback options and the automatic correction of geometric errors in machined parts (ALVARES, 1990).

The quality feedback system presents, as a strategy, tolerance analysis for the detection and prevention of errors in a global way. For CMM not to negatively influence the CLI with its error, its measurement uncertainty must be at least 1/5 to 1/10 of the lowest manufacturing tolerance of the produced part (ALVARES, 1990).

Statistical Process Control is a tool used to improve the quality of parts in the production cycle. Determining process variability, especially in manufacturing systems, is of great importance, as it makes it possible to comply with established quality requirements and achieve continuous process improvement. The causes of natural (quantifiable) and attributable (non-quantifiable) variability lead to controllable and uncontrollable variability. In this line of action, statistical process control plays a relevant role in monitoring, preventing, and supervising equipment involved in the process(KOLOSOWSKI et al., 2015).

Some of the causes of variability are already defined as a change in the raw material quality, problems with the equipment's operation, change in the measuring instruments, and tool wear. The control method seeks to associate the causes with attributable factors of variability (KOLOSOWSKI et al., 2015). A typical algorithm for CEP implementation can be described as follows:

- Stabilize the process, identifying and eliminating the attributable causes of variability.
- In case of insufficient process capability, improve the process by reducing its natural variability.
- Maintain a capable and stable process, identifying and removing special causes.
- Periodic analysis of the capability of the process for its continuous improvement.

Bagshaw (2002) presents research on machine tool error classification and describes methods to diagnose these manufacturing errors through diagnostic approaches, condition monitoring, and manufacturing data analysis (BAGSHAW; NEWMAN, 2002).

The production errors that affect the part geometric features are classified into two categories: errors that can affect individual features and those errors that affect a group more features large for the component. Individual feature errors include: cutting tool errors (misalignment, tool wear, and deflection error), programming errors feature size error, position/orientation error features, and interpolation error) among other errors (conditions, vibrations, and deflections of the part). The Combined feature errors category includes

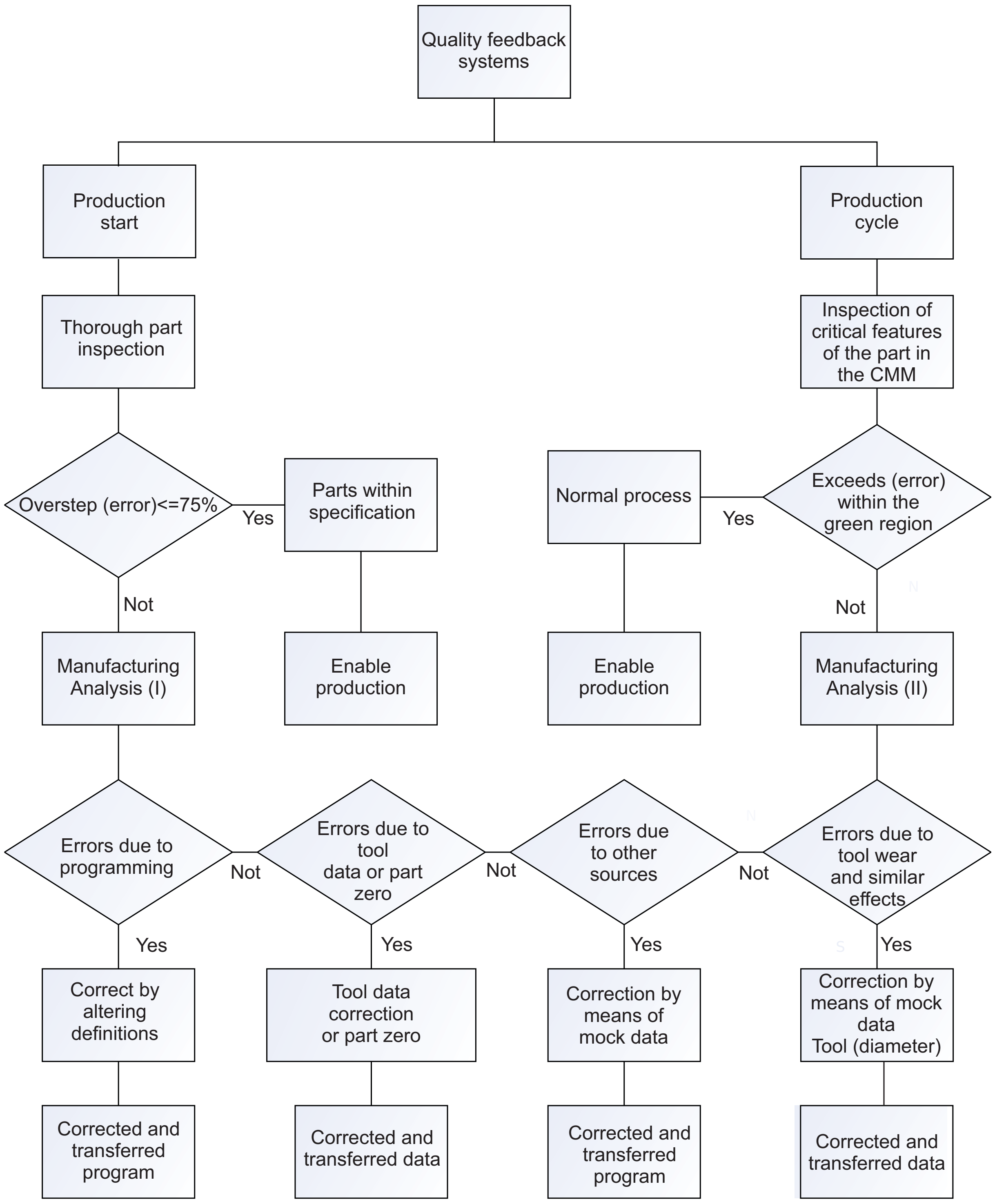

Quality feedback systems
Production start
Production cycle
Thorough part inspection
Inspection of critical features of the part in the CMM
Overstep (error)<=75%
Yes
Parts within specification
Normal process
Yes
Exceeds (error) within the green region
Not
Not
Manufacturing Analysis (I)
Enable production
Enable production
Manufacturing Analysis (II)
Errors due to programming
Not
Errors due to tool data or part zero
Not
Errors due to other sources
Not
Errors due to tool wear and similar effects
Yes
Yes
Yes
Yes
Correct by altering definitions
Tool data correction or part zero
Correction by means of mock data
Correction by means of mock data Tool (diameter)
Corrected and transferred program
Corrected and transferred data
Corrected and transferred program
Corrected and transferred data

Figure 2.17: Feedback options
Source: Aadaptation (ALVARES, 1990)

machine tool errors (axis calibration errors, servo interpolation errors, fatigue, thermal distortion, random stochastic errors), errors due to clamping, and workpiece work (configuration errors between components and machine), among other errors (dimensional errors of the material).

2.6.10 Error Compensation

The inspection system is responsible for acquiring values of the manufactured part's geometry through measurements made at various points located in different dimensional planes. The values allow for a reconstruction of the surface that the quality control system will use. The acquired data is analyzed and processed by a specialist system to quantify the errors, identify the causes and generate actions to correct them. Expert systems can base their operation on sets of rules, fuzzy logic, intelligent techniques, and knowledge to facilitate the generation of corrective actions in the manufacturing process. In-process compensation is generated directly in the program sent to the machine tool control system, which contains the execution routines for machining a new part. The inspection is repeated after machining; this reduces the error in each interaction of the closed-loop system.

The relevant issues to be resolved with the implementation of the closed-loop manufacturing system are focused on defining a mechanism to calculate the uncertainty values, determine the errors produced in the manufacture of the part, identify the possible sources of error and compensate the process to achieve minimize the differences between the designed part and the manufactured part. The quality of the information obtained in the measurement that enables the question solution depends directly on the type of inspection applied.

Two primary inspection techniques are commonly used when measuring a part. The first technique is called OMI (*On-Machine Inspection*) or OMM (*On-Machine Measurement*); it is characterized by being performed on the same machine tool and during the part manufacturing process. The second inspection technique is performed with a CMM coordinate measuring machine and outside the manufacturing line in a controlled environment. Selecting the appropriate type of inspection is essential to define the feedback strategy and the integration architecture. It is important to note that each inspection technique, due to its nature, has characteristics that can affect or produce variances in the measurement results. Some of the characteristics found in the IMO inspection are:

- Inspection times are reduced due to not moving the part to another machine. In addition to saving time, it also represents a reduction in the equipment involved in the measurement.
- Both the accessories and fasteners and the reference points and coordinate system used in the machining are used in the measurement, managing to minimize the errors given in the displacement of the part.
- The inspection can be carried out at different stages of the part machining process, enabling compensation during manufacture.

The OMI inspection technique's main disadvantage is given by the equipment used and the environment in which the measurement is made. The capability of the CNC machine and the disturbances acquired by the manufacturing environment (temperature, vibrations, dirt, calibration errors) define the quality of the inspection. Errors added to the process by different variables challenging to control in the manufacturing environment may not be detected.

Inspection using CMM machines is considered the essential technique in determining geometric error.

The main difference with the OMI inspection is that the measurement is carried out in a controlled environment, managing to minimize external disturbances and compensate for temperature. For a long time, coordinate measuring machines were responsible for providing the necessary quality control systems. Various technological advances, especially in the area of metrology, benefit from this inspection technique. Another relevant feature derived from the inspection with CMM machines is identifying the errors added by the CNC machine and the machining conditions through the post-processing of information. The main disadvantage faced by inspection using CMM is integrating the measurement results with other processes involved in the production cycle, making it difficult to overcome the barriers that prevent full integration within the closed manufacturing loop.

2.6.10.1 Sources of Uncertainty with Possible Correction

It is necessary to define a methodology based on the elements involved in the control architecture to compensate for the error. The elements maintain a relationship and connection; for this reason, changes or decisions in an element can affect the overall result. From the moment the part begins its life cycle, variances and uncertainties coexist, classified into the following types:

1. *Correlation uncertainty*: Occurs when projecting requirements with necessary specifications for the part and its functional performance.
2. *Specification uncertainty*: When a design requirement is mistranslated or with a critical fit above what is necessary for the functional performance of the part, known as uncertainty in the conformity analysis of the part.
3. *Manufacturing uncertainty*: In this set, the uncertainties have a different nature; a classification could be as follows:
   - CNC machine tool
   - NC programming
   - Machining process
   - Other factors

   In Manufacturing Uncertainty, a part of the error is always present and inherent to the process or machine, related to random sources of error (ALVARES, 1990). On the other hand, one part is the systematic and progressive errors compensated to improve and maintain the geometric quality of the piece. Some factors attributable to this uncertainty are:

   - Thermal effects, caused by the ambient temperature or by the heat generated inside the machine, affecting the machine's structure, mainly the whole, is responsible for a portion of the progressive and random errors.
   - Tool wear is responsible for progressive errors.
   - Static and dynamic stiffness of the machine tool, random errors.

- Geometric quality of the machine tool, its components, and assembly determine the cutting tool's position error (random, systematic, and progressive errors).

4. *Uncertainty in the measurement*: it depends on the method, instruments, or tools used to capture geometric information. A measurement out of parameters, methods not suitable for capturing geometric requirements or specifications, and measurement processes with errors above what is acceptable contribute to measurement uncertainty. The wrong interpretation of the inspection feature, wrong measurement strategy, the accuracy of the measurement does not match the specifications of the tolerances, and wrong part setup are some events that produce wrong measurements.

Total uncertainty contributes to each phase of the product life cycle given by the design, manufacturing, and inspection processes. Knowing these elements, their interrelation in the architecture, makes it possible to establish a feedback methodology. The sum of all types of uncertainty determines the maximum error for a specific machine or process under certain operating conditions.

The process variability presents normal distribution when the process is over control, being the deviation errors only random. The lack of control of the process is due to certain factors where, in addition to random errors, there is the presence of systematic errors or progressive errors. Progressive errors, in their essence, have the exact nature of systematic errors; they have a dynamic character, evolving gradually over time due to the variation of some greatness. When the process is under control, the level of variability will determine the manufacturing uncertainty that will depend on the constructive characteristics of the machine tool and the means used in its manufacture (manufacturing, production, and assembly), in addition to the tools and devices used (ALVARES, 1990).

Most of the error factors, referring to CNC machine tools and the machining process, that influence the uncertainty of a part can only be avoided or minimized. Among the actions conducive to reducing the sources of error are: preheating of the machine, lubrication of shafts, minimizing the internal heat sources that affect the process, air conditioning of the environment, protection against radiation, air circulation, good selection of the material of the tool, alter parameters related to machining conditions, part material and errors in the part program (ALVARES, 1990).

2.6.11 Closed Loop Control

The control strategy within the closed loop of manufacture and inspection seeks to compensate the input data to the CNC control system with the results of a comparison between the projected nominal part and the inspected part to meet the project's quality requirements.

The main challenge of closed-loop control is to provide the mechanisms for integrating data generated in the quality inspection system and ensure the interoperability of the architecture. The context of interoperability, in practice, is currently focused on the exchange of information, leaving aside the barriers given by restrictions on access and technological independence. An integrated approach focused on data harmonization is needed to transmit unambiguously the information generated during a part's manufacturing cycle.

The perspective of interoperability proposed in this research seeks to achieve an architecture within a line of digital concepts, which interconnects the data generated in the different CAD/CAPP/CAIP/CAM/C-NC/CAI systems and provides a neutral mechanism for bidirectional data flow with support for the three layers: application, logic, and physics. Interoperability is achieved through a data connection from the STEP, QIF standards which, in addition to sharing the focus based on features, encompass the major areas involved in the chain with product design, planning, manufacturing, and inspection. The integrated approach focuses on semantic and syntactic integrity that allows the validation and verification of data exchanged between CAx systems.

The architecture's core is given by the libraries provided by the STEP and QIF standards to support the manufacturing and inspection processes in the digital chain. The libraries contain the definitions and components referenced in the areas involved that can be integrated into web/Internet applications or directly within other existing standards, thus providing an interoperability and extensibility environment.

The data flow starts with a CAD model with GD&T and PMI information in a control closed-loop system. The model data is shared in the manufacturing chain in the AP242 format. Using the libraries provided by STEP and QIF, they are exported as application data to QIF-MBD and AP238. CAPP / CAIP planning systems import STEP and QIF-MBD files to generate plans with resources, rules, and mechanisms to be used. CAM / CAI systems import plans to generate specific execution programs for the CNC controller and the dimensional measurement equipment (DME). After manufacturing the part, the DME runs the programs to evaluate the part (or assembly) characteristics and export the measurement results. Based on statistical process control, the analysis system imports/exports the results in the format QIF-Statistics/QIF-Results. The compensation system makes the necessary changes to the STEP and QIF-MBD files to reduce manufacturing errors.

2.6.12 Error Calculation

During manufacturing, different factors interact so that different nature errors (static, dynamic, and progressive) are added. The geometric errors present in parts manufactured in CNC machining centers can be classified into four major groups. The proposed inspection strategy seeks to identify, monitor, and control the errors that a part acquires during the manufacturing process to meet the expected GD&T requirements (ALVARES, 1990; YANDAYAN; BURDEKIN, 1997).

The geometric errors present in the parts processed when machining with CNC machines, using cutting tools with defined geometry, can be attributed to four error sources: machine tool, NC programming, machining process, and other factors(ALVARES, 1990; YANDAYAN; BURDEKIN, 1997).

The errors present in the dimensions of machined parts initially reveal the variability of the manufacturing process. This variability/dispersion has a normal distribution when the process is under control, with errors due only to random factors. The lack of control of the process is due to certain factors where, in addition to random errors, there are also systematic errors and the third type of error called, in this work, progressive errors. Progressive errors, in their essence, have the exact nature of systematic errors; however, they have a dynamic character gradually evolving, due to the variation of some magnitude of the process (temperature, tool wear), unlike the errors systematic processes that have a permanent character(ALVARES, 1990; YANDAYAN; BURDEKIN, 1997).

Identifying the source of the error is based on the results of the inspection of a problem that is too complex and difficult to solve due to different factors influencing the measurement. An approximation can be achieved by implementing a mechanism for monitoring the manufacturing process in function of the manufacturing conditions, together with machine-iron models built with geometric tests; It would also be necessary to develop a global model that represents the system, and clearly, the model would be composed of non-linear parameters. It is possible to achieve compensation for geometric error determining the uncertainty globally, the product of a comparison of the nominal model with the accurate model, and making corrections on the data structure interpreted by the machine irons CNC control(ALVARES, 1990).

There are several factors, within these four groups of errors, that determine the manufacturing uncertainty, the most relevant being the following:

- Thermal effects, caused by the ambient temperature and the heat generated inside the machine (chips, friction, motors), affect the machine's structure.
- The machine's measurement and control systems are responsible for a significant part of the progressive errors and a portion of random errors.
- tool wear (progressive errors).
- static and dynamic stiffness of the machine tool (errors random);

Thermal deformations and tool wear cause the main factors of errors due to the machine/process, these errors being much more representative in roughing operations than in finishing operations. However, when a roughing operation presents a significant error (in the order of a tenth of one mm or more), in the finishing operation, this error will cause a lower dimensional surface quality than the desired quality, since the overmeasurement of material present the most, will influence the geometric quality of the piece(ALVARES, 1990).

In addition to the error factors related to the machine/process, there may be errors due to NC programming and other factors. These last two groups of error sources may appear superimposed on dependent errors directly from the machine/process. The identification and separation of these different sources of errors, based on dimensional inspection, is a problem that is too complex and difficult to solve (ALVARES, 1990).

The problem of calculating the error in this proposal is limited to machine tool geometric errors when prismatic parts are machined. The correction actions are coded and communicated to the CNC control of the machine tool. The analysis of inspection results is carried out to satisfy objectives such as identifying errors, trends based on the GD&T specifications given, and diagnosing the causes of the error to define, as far as possible, a correction strategy(ALVARES, 1990).

2.7 Summary

Developing a strategy to implement a closed manufacturing and inspection loop requires knowing and understanding the different aspects related to the application of the processes. This chapter presents aspects of the structure and exchange of data used for manufacturing and measurement processes. The interoperability problems derived from the exchange of information between non-homogeneous and often proprietary data structures are also approached to identify aspects that allow forging suitable selection criteria for the data structure. It is also noted that some researchers consulted have shown that the STEP standard is a viable option to support the transfer of data and product information. The definition-based model makes it possible to relate different contexts to the exact definition. The STEP standard inherits this concept and takes it as its fundamental unit called feature to relate information generated in the different life cycle phases. A feature can have both manufacturing and inspection reporting relationships. This concept enables the reuse of information to express different contexts.

Chapter 3

Integration Method Adherent to STEP Standard

3.1 Introduction

The need to integrate information has led to the emergence of standards and, as a result of this evolution, the STEP standard is accepted internationally as a complete standard for use as a product data exchange mechanism. The ISO 10303 standard defines a neutral format for the exchange of product data, which allows the transmission of information between CAx systems. This chapter formally describes the arguments for selecting the STEP standard data structure as a mechanism that solves data integration between CAx systems. The data structure of the STEP standard, its architecture, and the information model for data exchange are presented. The strategies and techniques for its implementation are presented based on AAM, ARM, and AIM models.

3.2 STEP (Standard for the Exchange of Product Model Data)

ISO 10303 is an ISO standard for the computer-interpretable representation and exchange of product manufacturing information. Its official title is: "Automation systems and integration — Product data representation and exchange" It is known informally as STEP, which stands for "STandard for the Exchange of Product model data". The STEP standard, inspired by the structure of a database system with ANSI/ SPARC architecture, provides a representation of product information and provides the necessary definitions to exchange, store, transfer, and access product information through different technologies (KERN et al., 1997a; KERN et al., 1997b). One goal of the STEP is to provide a data format that can be used by different software and systems involved in the product life cycle. In Figure 3.1, a schematic of the architecture of the STEP standard is presented (HANDBOOK, 2006).

The STEP standard provides the communication mechanisms to meet the new paradigms that restrict data conversion and support a product's information throughout the manufacturing chain. The STEP standard is constantly evolving, and new documents and processes are being incorporated to cover larger areas

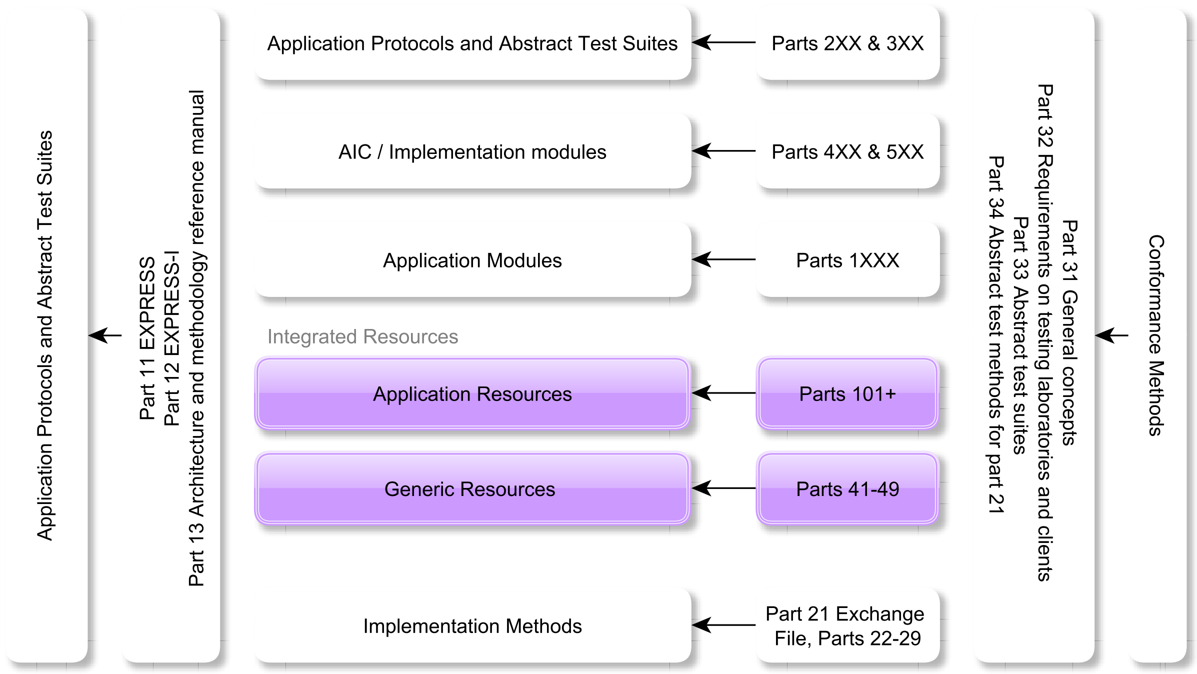

Application Protocols and Abstract Test Suites
Part 11 EXPRESS
Part 12 EXPRESS-I
Part 13 Architecture and methodology reference manual
Application Protocols and Abstract Test Suites
Parts 2XX & 3XX
AIC / Implementation modules
Parts 4XX & 5XX
Application Modules
Parts 1XXX
Integrated Resources
Application Resources
Parts 101+
Generic Resources
Parts 41-49
Implementation Methods
Part 21 Exchange File, Parts 22-29
Part 31 General concepts
Part 32 Requirements on testing laboratories and clients
Part 33 Abstract test suites
Part 34 Abstract test methods for part 21
Conformance Methods

Figure 3.1: Structure of STEP
Source: Adaptation (NASR et al., 2016)

of application (PRATT, 2001). The pattern is formed by a set of patterns organized in three layers which are:

- **Application layer**: It is an information model formed by Application Protocols (AP) and Application Interpreted Constructs (AIC) developed to be used in a specific application area. It provides the standard data specifications and semantics used to exchange product data between two or more applications. (Series 200 STEP documents).
- **Logical layer**: They are product information models created to describe all the domains of interest, unique and unambiguous. This layer is the library of product information models called Integrated Resources (Irs), which describes all the domains of interest. (Series 40 and 100 STEP documents).
- **Physical layer**: Implementation method that deals with mapping application schemes for a given computing technology (STEP Documents from Serie 20).

Table 3.1 summarizes the STEP documents series based on the standard architecture layers (Adaptation).

Based on the three-tier architecture shown in Table 3.1, developing applications that adhere to the standard is possible. The application layer supports the development of application-specific reference models. Application models are commonly developed in IDEF0. The logical layer is responsible for specifying the format of the definitions with the creation of reference models. Reference models are created based on a neutral structure and independent of any programming language such as EXPRESS and XML.

Table 3.1: STEP documents organized in the architecture composed of logical, physical, and application layers.

STEP Document Type	Document Identifying Number	Layer
Introductory	0 - 9	
Description methods	Serie 10	
Implementation methods	Serie 20	Physical
Compliance testing methodology	Serie 30	
Integrated Resources (IRs)	Serie 40 e 100	Logic
Application protocols (APs)	Série 200	Application
Interpreted application constructs	Serie 500	Application
Abstract test suites	Serie 300	

The physical layer defines the construction path of the communication structure of the STEP file through implementation methods. The most widely used implementation methods are based on the ISO 10303-22: 1998 SDAI.

The main reasons that justify the selection of the STEP standard in the development of the CAD/CAPP/CAIP/CAM/CAI/CNC systems integration architecture proposed in this project are the following:

- The generic and neutral nature of the standard, regardless of technologies and owners.
- Interpretable computational architecture that facilitates the automation of processes.
- Consistency in the data that provides robustness in the implementation.
- Different methods for accessing, storing, and transferring product data.

STEP considered an ISO standard, brings together the standardization effort that involves developing a technology that provides neutral methods and tools for the management of both product information and its specification. These standardized methods and tools allow the description, validation, and treatment of the information generated within a closed loop of manufacturing and inspection. The application protocols are developed strictly within this STEP technological approach. The application protocols intensely involved, such as AP238, AP242, and AP219, are specified in the EXPRESS modeling language, constituting a library of reusable concepts to develop a data model covering the scope of this project quite useful for processes that use techniques CAD or CAM. On the other hand, there are data transfer and exchange protocols neutrally through a standard data access interface.

3.3 Description Methods

Due to its complexity and extension, the STEP standard demands techniques that facilitate its interpretation and analysis for both humans and computer systems. The mechanisms used in the ISO 130303 standard are called Description methods and provide a generic language structure for formal specification of data regardless of the technologies used. It is possible to express cutouts of the standard for implementation legibly through the description methods and apply methods to check for inconsistencies. The

EXPRESS language is a language developed to partition, extract, integrate, and subdivide the STEP standard material's various contents.

3.3.1 EXPRESS Language

The STEP standard provides support through implementation methods classified in its logical layer. The data structure is developed in a modeling language based on neutral architecture and independent of any technology. The EXPRESS language is the standard mechanism recommended for defining the ARM model and supported by the ISO 10303-11 standard. Each entity's domain is completely specified in the ARM model, structuring the information to make the computational implementation. In this phase, the Abstract Test Suite (ATS) methodology, which consists of creating a test to assess and verify that the application's needs and requirements are completely satisfied, is also defined.

The EXPRESS language constitutes a fundamental element for describing and implementing the STEP standard since the protocols, tools, and methods are largely described in the EXPRESS language. This language inherits data modeling features such as NIM and IDEF1X, the imperative programming language constructs of Pascal, ADA, C ++, and some SQL language elements. The EXPRESS language manages to express the behavior and the logical relationships within a description of the data model that enables the automatic exploitation of the descriptions through computational tools. Its structure in textual language, specified by an orderly, consistent and homogeneous grammar, can be processed by computer means (PLANTEC, 2007).

The EXPRESS language is specified in ISO 110303-11 "Description methods: The EXPRESS language reference manual" is a description method used by STEP to facilitate the interpretation of information models and application protocols. EXPRESS is designed to use as an information modeling language, regardless of technology, allowing, in addition to data representation, to define restrictions, rules, functions, and procedures (KERN et al., 1997a). The language focuses on the definitions of the entities, which represent objects. An entity's definition is made in terms of its properties, characterized by specifications and domain restrictions. In EXPRESS, the concept of entity is the same concept of class in the object-oriented model, supporting different data types, as shown in Figure 3.2.

A `SCHEMA` can be considered a diagram within a domain that reflects an application context. Within this context, constants, data types, entities, or object classes with their respective relationships and restrictions are declared. Appendix I.1.1 shows the schematic in EXPRESS language used to define the dimensions of the part.

The `ENTITY` contains a detailed description of a class or object composed of a set of attributes or properties. Aggregations express the multiplicity assigned to properties. Some properties are necessarily mandatory in the construction of the object and are described in the form of explicit attributes (whose value must be defined when creating an instance), derivatives (calculated value product of an expression), or inverses (value defined based on another entity, define the cardinality of the association for the entity's context).

It is possible to restrict the domain of the value of attributes and instances of an entity. There may be a unique restriction that prevents more than one entity, a domain restriction created for a value of a type or

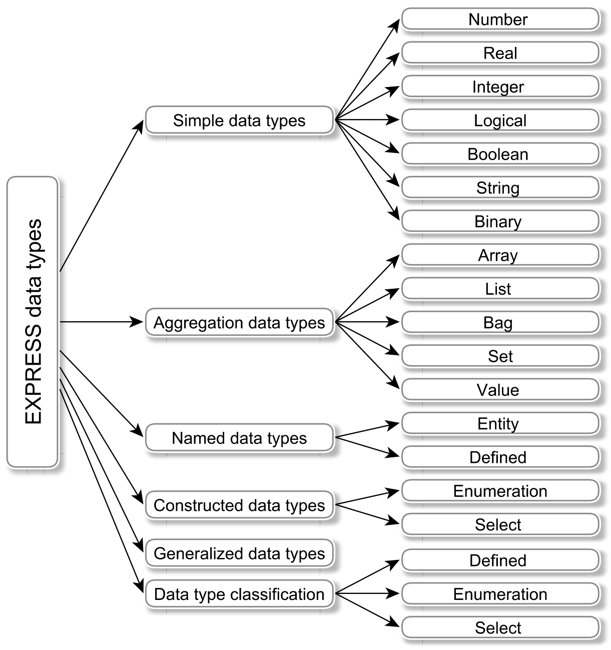


Figure 3.2: Data types used by the EXPRESS language

attribute of an instance, and a global restriction that expresses a restriction on all instances. Constraints are specified through a procedural language described as procedures or functions that allow operators such as assignment, conditional, and interactive instructions used to construct arithmetic operations.

The EXPRESS language binds an object-oriented principle, which allows entities to share attributes and methods. The entity can inherit attributes from one or more entities. A subtype inherits all local properties and constraints from its supertypes. Subtype entities can override an inherited attribute if it contributes to being a more specific entity.

EXPRESS has a set of predefined functions that can be used to request information on the type of attributes and variables and perform arithmetic and trigonometric calculations on the data. The `TYPEOF` function returns the set that includes the name of the type of the variable that is passed as an argument and those of all its supertypes.

A scheme can use content from other schemes; therefore, it is possible to describe schemes in a general way intended to be referenced or specialized by other more specific schemes. There are two different concepts; through the `USE` keyword, the entities specified in the schema used can be exploited independently. When using the `REFERENCE` keyword, the referenced schema's entities can only be used in the context of another entity, such as an attribute type. In general, `USE` facilitates the redefinition and enrichment of schemas, particularly inheriting attributes and properties, and `REFERENCE` enables association between schemas.

Modeling using the EXPRESS language is structured in schemes, entities, attributes, relationships, and constraints. Through entities, functions, and procedures, a schema describes the set of conditions for establishing a domain the product model. Entities can be evaluated to determine whether they belong to the domain or context. Entities and their attributes are declared within a schema to represent an object or concept. The attributes define an entity's important properties and the type (real, integer, string, etc.). The attributes can have relationships with other entities. Relationships are bidirectional, but one of the two directions is emphasized (FENG; YANG, 1995).

```
SCHEMA shape_tolerance_schema;
REFERENCE FROM measure_schema   -- ISO 10303-41
 (derive_dimensional_exponents,  dimensional_exponents,    length_measure_with_unit,   measure_with_unit,
  measure_value,   plane_angle_measure_with_unit);
                *
TYPE area_unit_type = EXTENSIBLE ENUMERATION OF
  (circular,
   square,
   rectangular);
END_TYPE;
                *
ENTITY geometric_tolerance_relationship;
  name : label;
  description : text;
  relating_geometric_tolerance : geometric_tolerance;
  related_geometric_tolerance : geometric_tolerance;
END_ENTITY;
                *
RULE subtype_exclusiveness_geometric_tolerance FOR
(geometric_tolerance);
WHERE
  WR1: SIZEOF(QUERY (gt <* geometric_tolerance | NOT (type_check_function(gt,
['SHAPE_TOLERANCE_SCHEMA.ANGULARITY_TOLERANCE'] , 2 ) ))) = 0;
END_RULE;
                *
FUNCTION sts_get_product_definition_shape
 (input : geometric_tolerance_target) : product_definition_shape;
CASE TRUE OF ('SHAPE_DIMENSION_SCHEMA.DIMENSIONAL_LOCATION' IN TYPEOF(input)) :
RETURN(input\shape_aspect_relationship.relating_shape_aspect\shape_aspect.of_shape);
('SHAPE_DIMENSION_SCHEMA.DIMENSIONAL_SIZE' IN TYPEOF(input)) :
RETURN(input\dimensional_size.applies_to\shape_aspect.of_shape);
('PRODUCT_PROPERTY_DEFINITION_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(input)) :
                 RETURN(input);
               ('PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPECT' IN TYPEOF(input)) :
                 RETURN(input\shape_aspect.of_shape);
  OTHERWISE : RETURN(?);
END_CASE;
END_FUNCTION;
END_SCHEMA;  -- shape_tolerance_schema
```

Figure 3.3: Example code for an EXPRESS language schema.
Source:(ISO 10303-41, 2018)

EXPRESS entities are defined in terms of attributes or properties considered important in implementation or use. Attributes can be represented through entities. The EXPRESS schema provides the structures for defining data, types, entities, rules, and functions.The EXPRESS language supports the definition of rules and the writing of functions, using arithmetic operators, logical operators, expressions, numeric functions, aggregate operators, methods description, and set of objects (ZHAO et al., 2011). Figure 3.3 shows a code made in EXPRESS language.

ISO 10303 also specifies a graphical representation for buildings in the EXPRESS language called EXPRESS-G. This description method contained in part 11 of the standard is a graphic form of EXPRESS that brings together the following functions: represent entities, show the type of data, attributes, relationships, and hierarchy. There are three types of graphic symbols in the EXPRESS-G diagram:

- **Definitions**: These are symbols that denote simple data types, constructors, and schemas. Definitions are denoted by boxes that include the definition name.

- **Relationships**: Symbols that describe relationships that exist between definitions. The relationships between the items are indicated by the lines that join the boxes. Different line styles provide information about the type of definition or relationship.

- **Composition**: Symbols that allow a diagram to be displayed on more than one page.

Lines of various styles connect the definition symbol. In EXPRESS-G, the relationship is marked with an open circle in the indicated direction. For an inheritance relationship, the direction indicated is for the subtype; that is, the circle is in the line's final subtype. The Figure 3.4 shows a formal representation in EXPRESS-G for the given level of the entity `Representation_item`. The specifications for the entity contain all the checks identified for the level shown.

EXPRESS is designed as a generic language for transmitting data information, useful for modeling data objects of any type. It has characteristics of definition languages used in databases and programming languages, but it is not a programming language. The EXPRESS language is generic with no purpose other than information modeling tasks, independent of any particular system or programming; its only declarative nature does not define methods applied to those entities in an application context. Its proximity to the object-oriented programming perspective allows the description of an object's data structure that in EXPRESS is called an entity equivalent to a class within the programming perspective. An entity's properties are called attributes and are defined through simple data types (e.g., STRING and INTEGER) or aggregates (e.g., ARRAY, LIST, and SET).

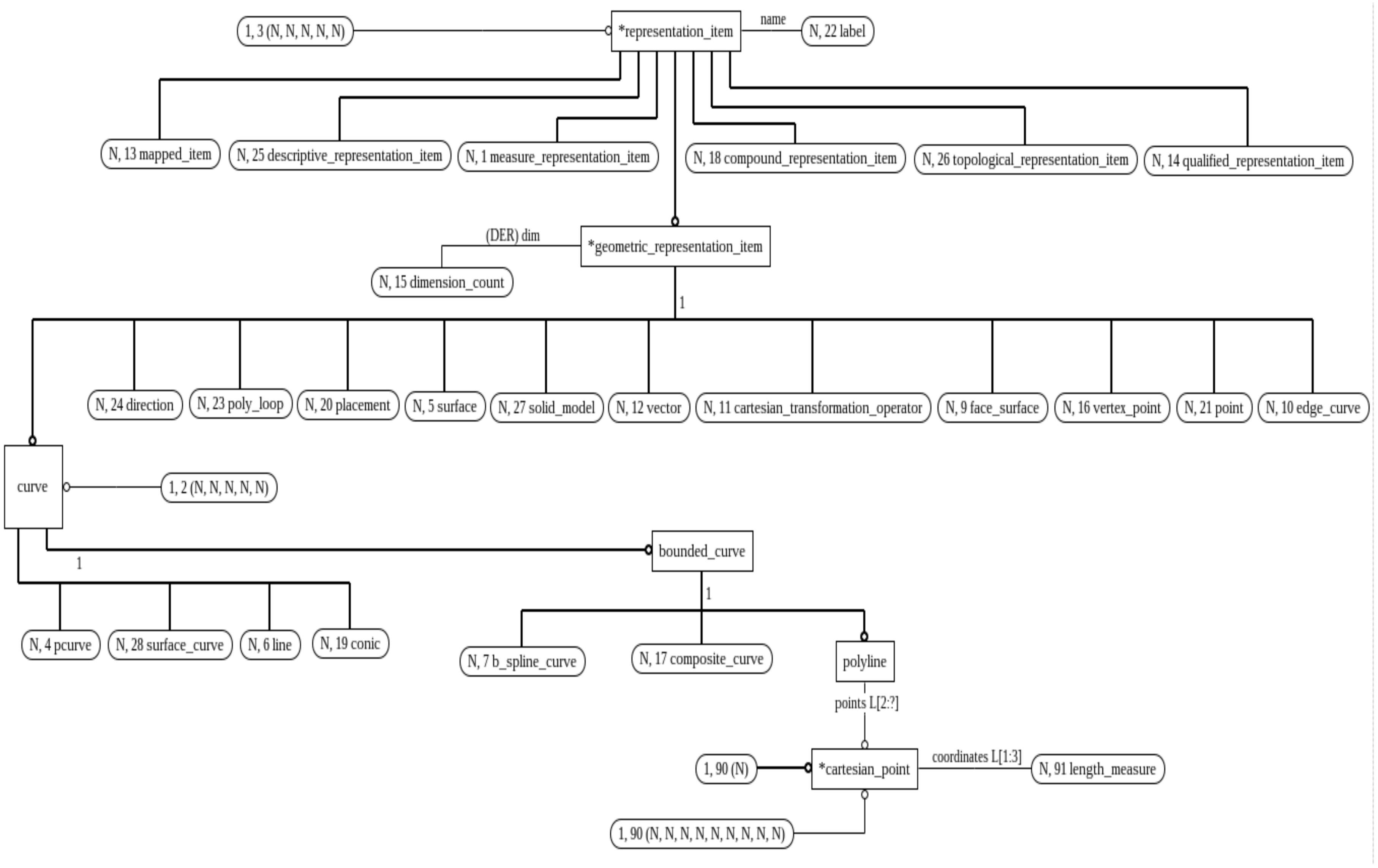
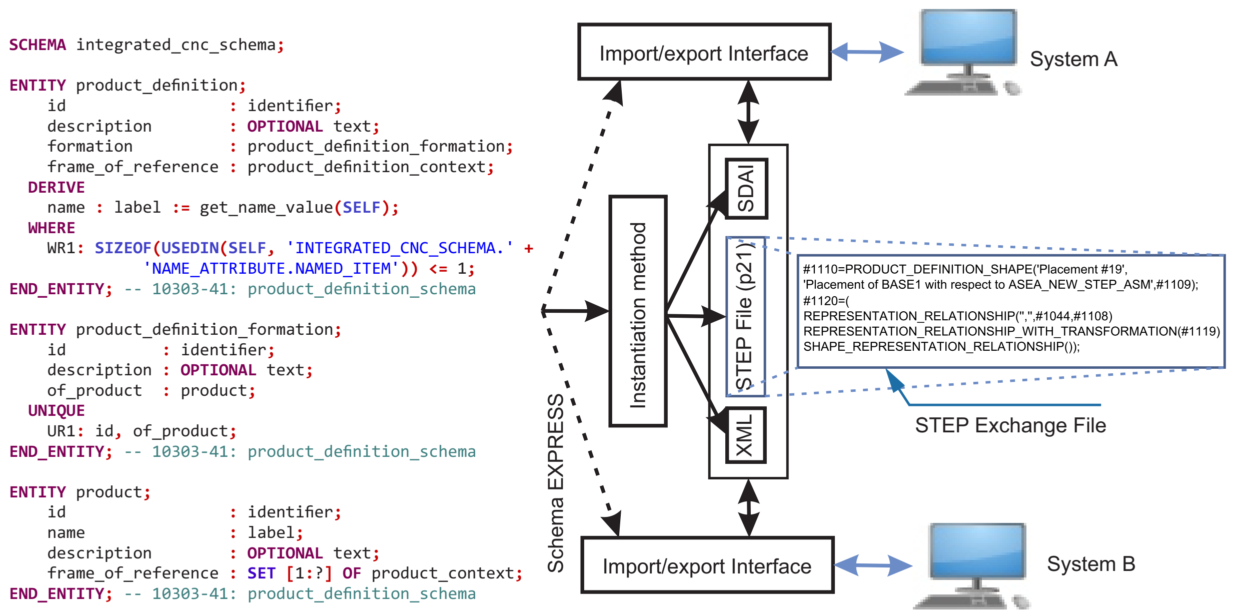


Figure 3.4: Formal representation in EXPRESS-G for the entity `Representation_item`

The inheritance relations are applied in subtypes and supertypes within the EXPRESS schemas, constituting parent/child relationships between the entities contained in a schema. The EXPRESS information model is organized in schemas. A schema is a logically complete set of definitions of entities that serve to subdivide large information models. In addition to data type, types of constraints on instances, and entities, the schema includes both a broad set of methods for describing constraints and setting rules on an entity's attributes and EXPRESS functions that can be used to calculate derived attribute values.

3.4 Implementation Methods

The previous section presented the STEP standard's description methods, in which the EXPRESS language is its primary tool. The STEP standard defines the implementation methods to take advantage of these schemas defined in the description methods in a computerized or automated way. The implementation methods are intended to define the procedures for exchanging and sharing EXPRESS schema instances between systems. The exchange happens through neutral files (SDAI, p21, or XML), syntactically homogeneous shared between heterogeneous systems.



```
SCHEMA integrated_cnc_schema;

ENTITY product_definition;
    id                 : identifier;
    description        : OPTIONAL text;
    formation          : product_definition_formation;
    frame_of_reference : product_definition_context;
  DERIVE
    name : label := get_name_value(SELF);
  WHERE
    WR1: SIZEOF(USEDIN(SELF, 'INTEGRATED_CNC_SCHEMA.' +
            'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; -- 10303-41: product_definition_schema

ENTITY product_definition_formation;
    id          : identifier;
    description : OPTIONAL text;
    of_product  : product;
  UNIQUE
    UR1: id, of_product;
END_ENTITY; -- 10303-41: product_definition_schema

ENTITY product;
    id                 : identifier;
    name               : label;
    description        : OPTIONAL text;
    frame_of_reference : SET [1:?] OF product_context;
END_ENTITY; -- 10303-41: product_definition_schema
```

```
#1110=PRODUCT_DEFINITION_SHAPE('Placement #19',
'Placement of BASE1 with respect to ASEA_NEW_STEP_ASM',#1109);
#1120=(
REPRESENTATION_RELATIONSHIP('','',#1044,#1108)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#1119)
SHAPE_REPRESENTATION_RELATIONSHIP());
```

Figure 3.5: Systems Interaction via STEP file.

Implementation methods are mechanisms for structuring the information specified in the application protocols. The EXPRESS language describes a scheme through entities, functions, and procedures but does not define an implementation method. The STEP standard defines several structured methods, which are defined to describe all the instances that support the exchange of product information specified by the application protocols. The implementation methods are known as SDAI, classified in international standards: ISO 10303-21, ISO 10303-22, ISO 10303-23, ISO 10303-24; to facilitate product data integration when implementing implementations for specific applications. This section presents the methods most commonly used in developing applications based on the STEP standard.

In a closed-loop of manufacturing and inspection, it is essential to guarantee the interaction between these two or more systems, some with different purposes but sharing the same data structure. Data persistence, information transfer, and integration are concepts that guarantee interoperability. There are two ways to implement interoperability, first through a specific link that adapts to system integration needs and requires interfaces that support this integration. The second is a standardized interface that uses a standardized, reusable protocol to establish the integration. This second approach is implemented in the STEP standard to represent information employing a neutral and standard format that contains the instances created from schemas in EXPRESS language. The integrity of the information contained within a neutral file is assured, and the intelligibility because the generated file maintains technological independence and system specifications, minimizing problems in integrating systems.

The Figure 3.5 shows a diagram of how two systems interaction happens, adhering to the STEP standard. Each system must have an interface to import and export neutral files depending on the selected implementation (p21, XML, or SDAI). The EXPRESS schema is the means of describing the data to be exchanged. This schema contains the definitions and is shared to generate implementation files with instances that maintain a syntactically homogeneous structure. Code generators can be developed from EXPRESS schemas. The implementation neutral file gathers instances representing the encoded information ready to be transmitted and interpreted by another system.

3.4.1 XML Representations of EXPRESS Schemas and Data

As a result of an effort to satisfy the integration and interoperability requirements to take advantage of the internet's advantages, XML (eXtensible Markup Language) emerged. The language allows defining the grammar to exchange structured information between different applications, including WEB applications. With the ISO 10303-28 standard (Implementation methods:XML representation of EXPRESS schemas and data), the procedure for using XML to represent the information contained in EXPRESS schemata is defined (ISO 10303-28, 2007).

For implementations from EXPRESS schemas, ISO 10303-28 specifies a set of declarations for the XML language based on the EXPRESS language syntax within two perspectives: one late binding the second early binding. The late binding context specifies a simple set of brand declarations independent of EXPRESS schemas, representing data for any schema. The early binding context is for specifying the results of generating sets of brand declarations independent in the EXPRESS schema.

Figure 3.6 shows an EXPRESS schema and its respective XML markings based on the architecture exposed in ISO 103003-28. The set of markup declarations defined in the standard are intended for a formal specification for XML documents formatting. The late binding is a simple markup declaration that can be used to represent data from an EXPRESS schema based on the early Bindings markup and a formal requirements definition architecture.

3.4.2 STEP Data Access Interface

The exchange of information in the different phases of the product life cycle promotes interoperability in the systems. The STEP standard aims to describe and share data in each phase of the life cycle; it needs

```
SCHEMA my_schema;
 ENTITY an_entity;
      attr1 : STRING;
 END_ENTITY;
END_SCHEMA;
```

(a) EXPRESS scheme

```
 <schema_id>my_schema</schema_id>
 <entity_decl>
   <entity_id>an_entity</entity_id>
   <explicit_attr_block>
     <explicit_attr>
       <attribute_id>attr1</attribute_id>
       <base_type><string/></base_type>
     </explicit_attr>
   </explicit_attr_block>
  </entity_decl>
</schema_decl>
```

(b) XML markup corresponding to the EXPRESS schema

```
 <schema_id>my_schema</schema_id>
 <entity_decl>
   <entity_id>an_entity</entity_id>
   <explicit_attr_block>
     <explicit_attr>
       <attribute_id>attr1</attribute_id>
       <base_type><string/></base_type>
     </explicit_attr>
   </explicit_attr_block>
  </entity_decl>
</schema_decl>
```

(c) XML markup set late binding for the EXPRESS schema

```
<My_schema-schema id="id1" express_schema_name="My_schema"
express_schema_identifier="My_schema Edition 2">
  <An_entity id="id2">
    <An_entity.attr1><string>an attr1 value</string></An_entity.attr1>
  </An_entity>
</My_schema-schema>
```

(d) XML markup set early binding for the EXPRESS schema.

Figure 3.6: Example of XML declarations based on EXPRESS language.
Source: (ISO 10303-28, 2007)

a standard interface to implement the exchange of information from an integrated database. The standard ISO 10303-22 Implementation methods: Standard data Access interface specifies a data access interface's functional characteristics. This interface is known as the Standard Data Access Interface (SDAI). The SDAI specifies the operations available to an application to acquire and manipulate data whose structure is defined using ISO 10303-11 (EXPRESS). According to the EXPRESS schemas, linking the EXPRESS data with a programming language refers to mapping constructions and generating a specific data structure. The programming language definitions, the entities defined in the EXPRESS schemas are converted to C, C++ classes, or Java. The goal of linking EXPRESS data with a particular programming language enables developing solutions that promote integration and sharing at all life cycle stages. SDAI standardizes the procedure, regardless of any programming language and systems, to access the data stored in a database.

A closed-loop of manufacturing and inspection operates on different phases of the life cycle, and some of them coincide; it is required to share information in each phase of the life cycle. It is necessary to implement the exchange of information within an integrated database or resource that guarantees both the persistence of the information and its integrity. SDAI is neutral but has implementations for some programming languages like C++, C, or IDL. The standard allows two types of implementation SDAI generic (or late) and SDAI specialized. Generic (or late) SDAI enables access to data regardless of source EXPRESS schema; this late binding approach does not map EXPRESS entities to classes using EXPRESS entity dictionaries to access data. Specialized SDAI that depends on EXPRESS source schema.

In an SDAI session, the manipulated instances are created and accessed through a model that interprets a source EXPRESS schema's description. This entity that references the model is called `SDAI_Model` and is stored in a repository. The `SDAI_Model` entity establishes a correspondence that links a stored resource, either a physical medium, data bank, or available memory. Two or more `SDAI_Model` entities can exist and share data regardless of their repository; this enables other applications with services available in multiple databases during one session. A dictionary contains the description of the instances handled by an `SDAI` session. The `SDAI_dictionary_schema` dictionary is created in order to describe the semantics of an EXPRESS schema. This dictionary contains the definitions in the source EXPRESS schema and is not modifiable by applications that access them.

An application using SDAI creates a session in which it has the stored repositories available. This data is open and can be read, modified, or operated with this information to develop a purpose. During the session, the data in the repositories is constantly updated to maintain cohesion in the information. Figure 3.7 presents an architecture for establishing an SDAI session.

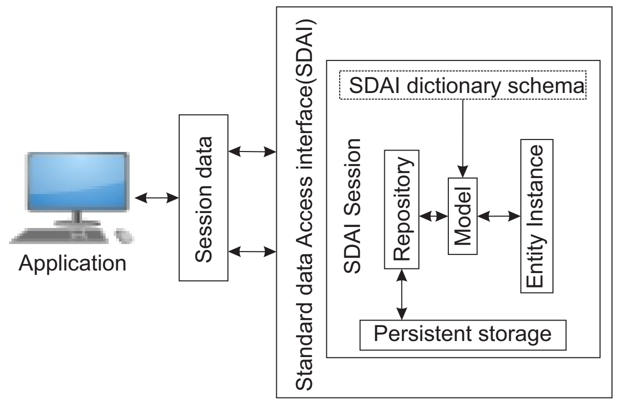


Figure 3.7: The architecture of an SDAI session.

3.4.3 STEP File as Exchange Structure

The neutral exchange file defined in the ISO 10303-21 standard (Part 21: Implementation methods: Clear text encoding of the exchange structure) known as STEP file or p21 contains instances created based on an EXPRESS schema. Each STEP file explicitly references the schema in its content. The standard presents a complete interchange structure that contains Formal definitions, Tokens, Structured data types, and sections. The exchange structure encodes the information in two large sections: Header and Data section.

The Header section is the first section; it is responsible for identifying the file's origin, the EXPRESS schema that contains information applicable to the exchange structure. The Figure 3.8 presents a code snippet from a STEP exchange file that emphasizes the Header section.

The EXPRESS language contains the specifications that allow encoding the information in the STEP File. Figure 3.9 shows a diagram with the entities that make up the Header section; it also shows the EXPRESS specification to create an instance of the `file_schema` entity and the encoding result within the STEP file exchange structure.

The second section, called the Data section, contains the instances that are transferred by the exchange structure. Each data section contains instances of entities corresponding to an EXPRESS schema specified in the Header section. Each instance has an integer number as an identifier that is unique within the

```
ISO-10303-21;
HEADER;
/* Generated by software containing
 * JSDAI (TM) from LKSoft (www.lksoft.com, www.jsdai.net)
 * JSDAI Runtime Version 4.3.0 2011-12-15T17:41:51
 */
FILE_DESCRIPTION(
/* description */ ('Program to generate AP238 p21 file'),
/* implementation_level */ '2;1');
FILE_NAME(
/* name */ ' ',
/* time_stamp */ '2021-01-11T15:21:05',
/* author */ ('Cristhian Ivan Riano Jaimes','Alberto J. Alvares'),
/* organization */ ('Universidade de Brasilia (UnB)'),
/* preprocessor_version */ ' ',
/* originating_system */ 'JSDAI MULTIPLE Version 4.0.0 (Build 270, 2011-12-15T17:42:49)',
/* authorization */ 'cristhianivanrj');
FILE_SCHEMA(('INTEGRATED_CNC_SCHEMA'));
ENDSEC;
```

Figure 3.8: Code snippet from a STEP file showing the Header section.

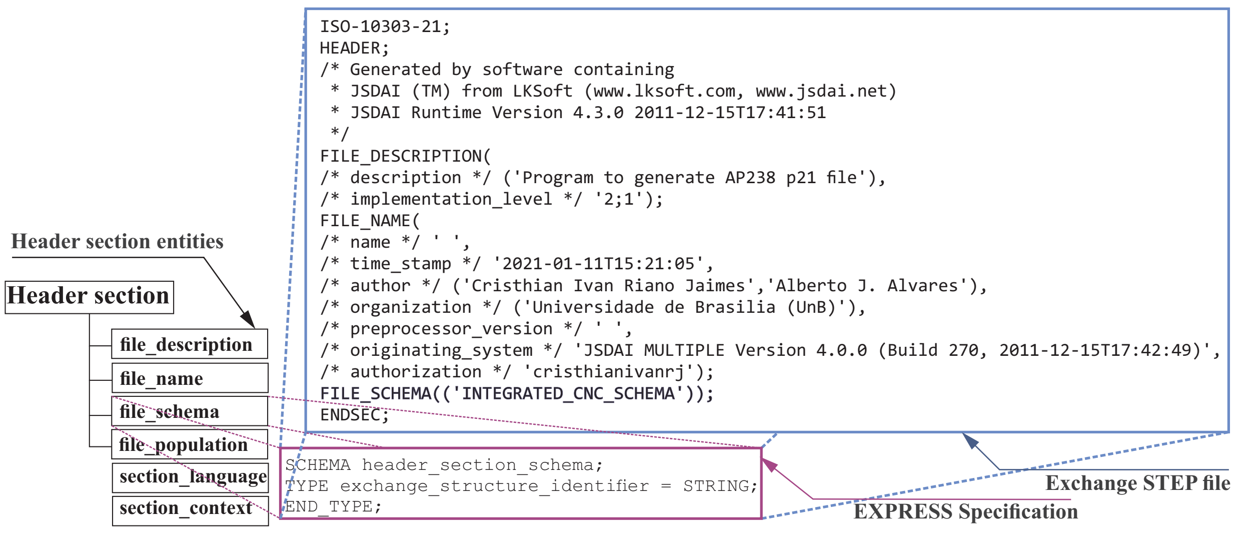
exchange structure; for this reason, the instances within the exchange file do not need to have a specific order within the list of instances, and if they can refer to the name of an instance before it is defined in the exchange structure. An exchange file can additionally contain instances defined by the bone user if they can include instances that are not part of the EXPRESS scheme specified in the HEADER section. An entity instance comprises a unique numeric identifier, the entity's name, and a list of values for each entity's explicit attribute. The derived attributes and restrictions, functions, rules, and algorithms described in the EXPRESS scheme are not contained in the STEP exchange file. Figure 3.10 shows a code snippet from a STEP exchange file that contains a small part of the Data section.

The Figure 3.11 shows a diagram that describes the relationship between entities necessary to define a`machining_workplan` entity, followed by the EXPRESS specification that describes the `mach ining _workplan` entity, and finally, a clipping of the Data section of a STEP file exchange that contains the created instances, emphasizing the `machining_workplan` entity instance.

3.4.3.1 Mapping from EXPRESS to the exchange structure

The instances in the exchange STEP file correspond to a data structure defined in EXPRESS. The data types defined in EXPRESS need to be properly mapped for the exchange structure. The EXPRESS language includes `TYPE` and `ENTITY` declarations, `CONSTANT` declarations, constraint specifications, and algorithm descriptions. The interchange file only contains `TYPE` and `ENTITY` declarations; for this reason, only these definitions are mapped in the interchange structure. Simple data types like Integer, String, Boolean, Logical, Real, Binary, Number, List, Array, Set, Bag, Enumeration, Simple defined types, and Select data types are mapped with values for the interchange structure.

A simple entity instance is an entity instance that is not an instance of a SUBTYPE of any entity data type and is fully described by a single EXPRESS entity declaration. All other instances whose description involves more than one entity declaration are called complex entity instances even when only one of them



```
ISO-10303-21;
HEADER;
/* Generated by software containing
 * JSDAI (TM) from LKSoft (www.lksoft.com, www.jsdai.net)
 * JSDAI Runtime Version 4.3.0 2011-12-15T17:41:51
 */
FILE_DESCRIPTION(
/* description */ ('Program to generate AP238 p21 file'),
/* implementation_level */ '2;1');
FILE_NAME(
/* name */ ' ',
/* time_stamp */ '2021-01-11T15:21:05',
/* author */ ('Cristhian Ivan Riano Jaimes','Alberto J. Alvares'),
/* organization */ ('Universidade de Brasilia (UnB)'),
/* preprocessor_version */ ' ',
/* originating_system */ 'JSDAI MULTIPLE Version 4.0.0 (Build 270, 2011-12-15T17:42:49)',
/* authorization */ 'cristhianivanrj');
FILE_SCHEMA(('INTEGRATED_CNC_SCHEMA'));
ENDSEC;
```

```
SCHEMA header_section_schema;
TYPE exchange_structure_identifier = STRING;
END_TYPE;
```

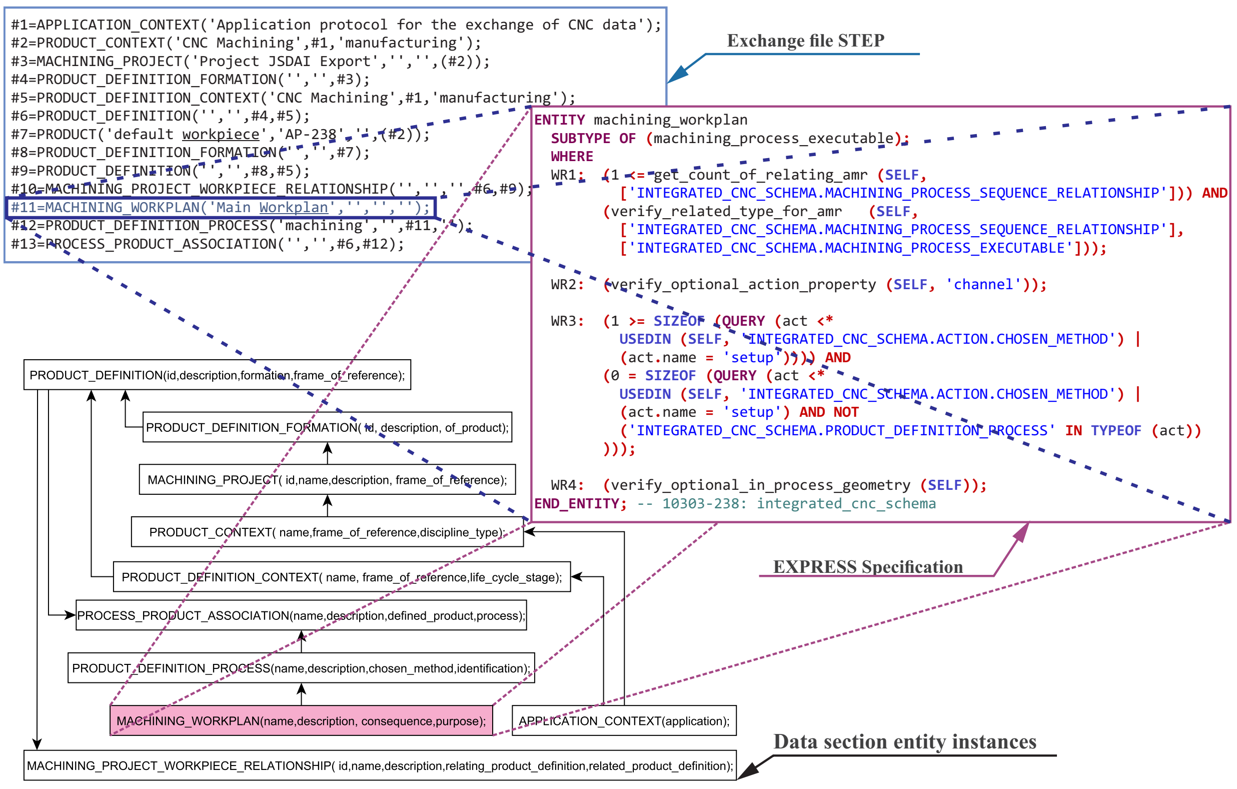
Figure 3.9: The encoding process the Header section in the exchange STEP file.

```
#1=APPLICATION_CONTEXT('Application protocol for the exchange of CNC data');
#2=PRODUCT_CONTEXT('CNC Machining',#1,'manufacturing');
#3=MACHINING_PROJECT('Project JSDAI Export','','',(#2));
#4=PRODUCT_DEFINITION_FORMATION('','',#3);
#5=PRODUCT_DEFINITION_CONTEXT('CNC Machining',#1,'manufacturing');
#6=PRODUCT_DEFINITION('','',#4,#5);
#7=PRODUCT('default workpiece','AP-238','',(#2));
#8=PRODUCT_DEFINITION_FORMATION('','',#7);
#9=PRODUCT_DEFINITION('','',#8,#5);
#10=MACHINING_PROJECT_WORKPIECE_RELATIONSHIP('','','',#6,#9);
#11=MACHINING_WORKPLAN('Main Workplan','','','');
#12=PRODUCT_DEFINITION_PROCESS('machining','',#11,'');
#13=PROCESS_PRODUCT_ASSOCIATION('','',#6,#12);
```

Figure 3.10: Code snippet from a STEP file showing the Data section.

contains explicit attributes. A simple entity can be an instance of a SUPERTYPE as long as it is not an instance of any SUBTYPE, but every instance of a SUBTYPEe is always complex. Each entity contains some explicit attributes that are mapped in the exchange structure. They are built in the order defined by the corresponding EXPRESS specification. The type of values is expressed implicitly by giving the value in textual format. The first parameter must be the value of the first explicit attribute. There can be explicit attributes declared as optional and do not require having a given value in the entity instance. Figure 3.12 shows a simple entity instance, its attributes, and values mapped to the exchange structure.

There are cases where an attribute can be an instance of another entity. The referenced instance is mapped to the attribute in the exchange structure by the entity's instance name. The referenced entity must be defined within the Data section. The entity instance attributes are given by the inheritance relationship of the attribute value list of the subtype entity instance concatenated with its supertype instances. Figure 3.13 shows this situation where the product instance in its `frame_of_reference` attribute references instance #2. The referenced instance `product_context` must also be defined within the data section.

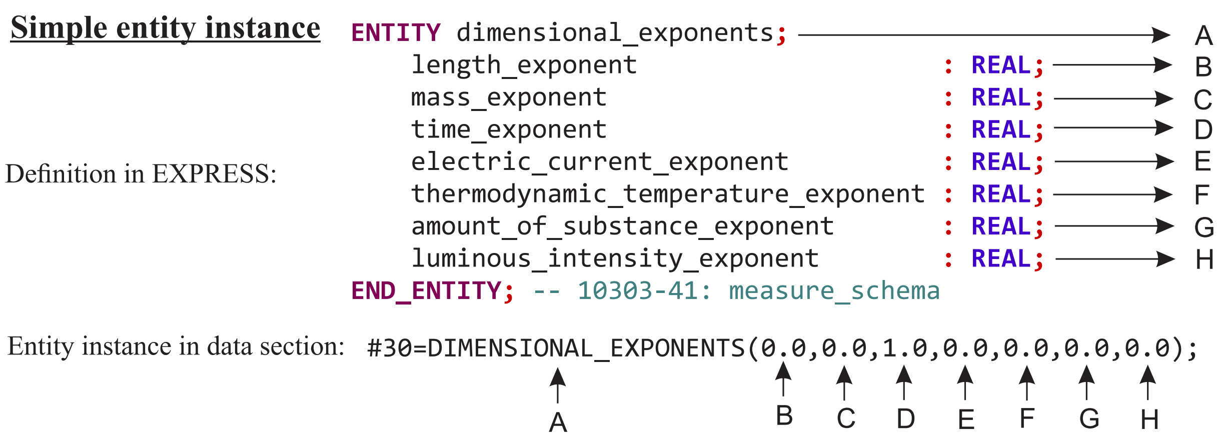


```
#1=APPLICATION_CONTEXT('Application protocol for the exchange of CNC data');
#2=PRODUCT_CONTEXT('CNC Machining',#1,'manufacturing');
#3=MACHINING_PROJECT('Project JSDAI Export','','',(#2));
#4=PRODUCT_DEFINITION_FORMATION('','',#3);
#5=PRODUCT_DEFINITION_CONTEXT('CNC Machining',#1,'manufacturing');
#6=PRODUCT_DEFINITION('','',#4,#5);
#7=PRODUCT('default workpiece','AP-238','',(#2));
#8=PRODUCT_DEFINITION_FORMATION('','',#7);
#9=PRODUCT_DEFINITION('','',#8,#5);
#10=MACHINING_PROJECT_WORKPIECE_RELATIONSHIP('','','',#6,#9);
#11=MACHINING_WORKPLAN('Main Workplan','','','');
#12=PRODUCT_DEFINITION_PROCESS('machining','',#11,'');
#13=PROCESS_PRODUCT_ASSOCIATION('','',#6,#12);
```

```
ENTITY machining_workplan
  SUBTYPE OF (machining_process_executable);
  WHERE
  WR1:  (1 <= get_count_of_relating_amr (SELF,
          ['INTEGRATED_CNC_SCHEMA.MACHINING_PROCESS_SEQUENCE_RELATIONSHIP'])) AND
        (verify_related_type_for_amr   (SELF,
          ['INTEGRATED_CNC_SCHEMA.MACHINING_PROCESS_SEQUENCE_RELATIONSHIP'],
          ['INTEGRATED_CNC_SCHEMA.MACHINING_PROCESS_EXECUTABLE']));

  WR2:  (verify_optional_action_property (SELF, 'channel'));

  WR3:  (1 >= SIZEOF (QUERY (act <*
          USEDIN (SELF, 'INTEGRATED_CNC_SCHEMA.ACTION.CHOSEN_METHOD') |
          (act.name = 'setup')))) AND
        (0 = SIZEOF (QUERY (act <*
          USEDIN (SELF, 'INTEGRATED_CNC_SCHEMA.ACTION.CHOSEN_METHOD') |
          (act.name = 'setup') AND NOT
          ('INTEGRATED_CNC_SCHEMA.PRODUCT_DEFINITION_PROCESS' IN TYPEOF (act))
        )));

  WR4:  (verify_optional_in_process_geometry (SELF));
END_ENTITY; -- 10303-238: integrated_cnc_schema
```

Exchange file STEP
EXPRESS Specification
Data section entity instances
PRODUCT_DEFINITION(id,description,formation,frame_of_reference);
PRODUCT_DEFINITION_FORMATION(id, description, of_product);
MACHINING_PROJECT(id,name,description, frame_of_reference);
PRODUCT_CONTEXT(name,frame_of_reference,discipline_type);
PRODUCT_DEFINITION_CONTEXT(name, frame_of_reference,life_cycle_stage);
PROCESS_PRODUCT_ASSOCIATION(name,description,defined_product,process);
PRODUCT_DEFINITION_PROCESS(name,description,chosen_method,identification);
MACHINING_WORKPLAN(name,description, consequence,purpose);
APPLICATION_CONTEXT(application);
MACHINING_PROJECT_WORKPIECE_RELATIONSHIP(id,name,description,relating_product_definition,related_product_definition);

Figure 3.11: Example of entity instantiation of the `machining_workplan`.



Simple entity instance

Definition in EXPRESS:

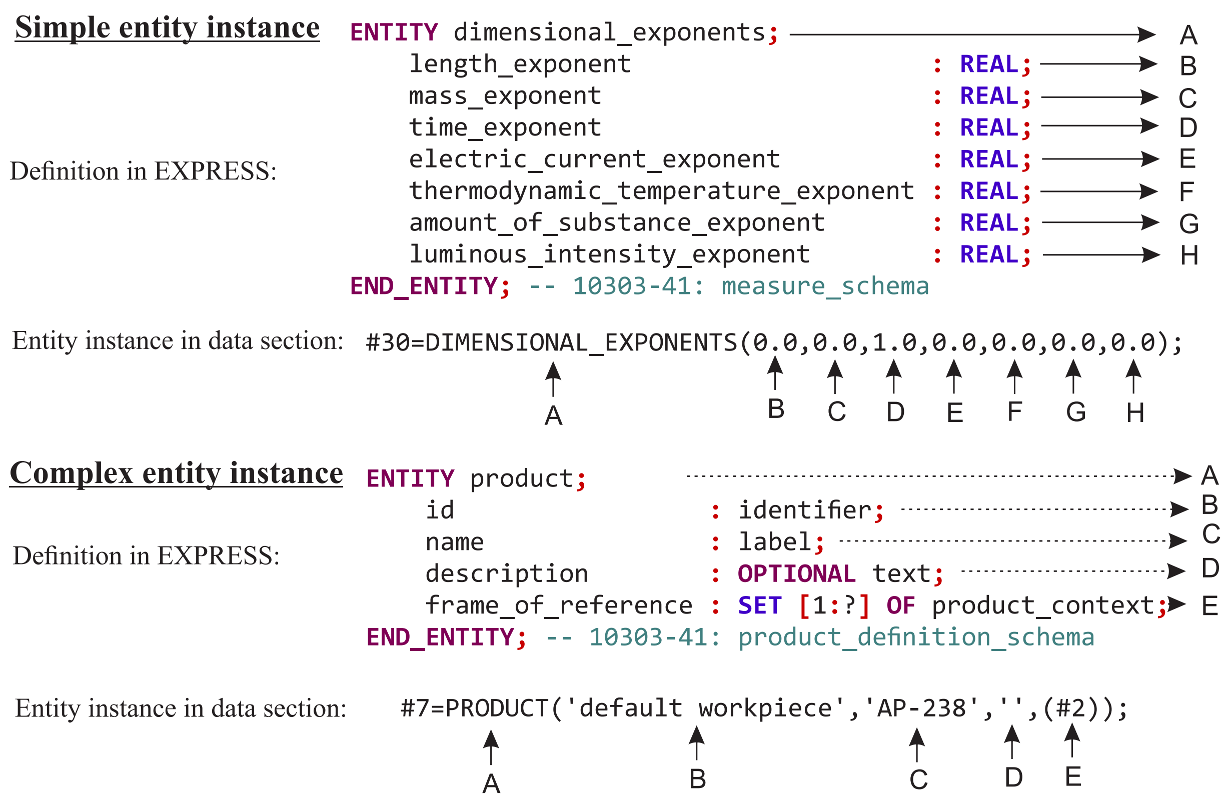
```
ENTITY dimensional_exponents;                                  A
   length_exponent                    : REAL;                  B
   mass_exponent                      : REAL;                  C
   time_exponent                      : REAL;                  D
   electric_current_exponent          : REAL;                  E
   thermodynamic_temperature_exponent : REAL;                  F
   amount_of_substance_exponent       : REAL;                  G
   luminous_intensity_exponent        : REAL;                  H
END_ENTITY; -- 10303-41: measure_schema
```

Entity instance in data section:

```
#30=DIMENSIONAL_EXPONENTS(0.0,0.0,1.0,0.0,0.0,0.0,0.0);
     A                   B   C   D   E   F   G   H
```

Figure 3.12: Example of Simple entity instance.

An entity is considered complex when the set of values of its attributes described by an EXPRESS entity declaration is composed of sublists with other entities attributes. Each instance name of an entity in its attributes identifies a value of a partial attribute in the complex entity instance. Therefore, the set of attributes evaluated through other entities partially conforms to the complex entity's set of values. Each value of the complex entity identified by an instance name is part of the entity's attributes to be described. The attribute value list contains associated sublists. This set represents a list of attribute values for the complex entity instance.



```
ENTITY dimensional_exponents;
    length_exponent                       : REAL;
    mass_exponent                         : REAL;
    time_exponent                         : REAL;
    electric_current_exponent             : REAL;
    thermodynamic_temperature_exponent    : REAL;
    amount_of_substance_exponent          : REAL;
    luminous_intensity_exponent           : REAL;
END_ENTITY; -- 10303-41: measure_schema
```

```
#30=DIMENSIONAL_EXPONENTS(0.0,0.0,1.0,0.0,0.0,0.0,0.0);
```

```
ENTITY product;
    id                  : identifier;
    name                : label;
    description         : OPTIONAL text;
    frame_of_reference  : SET [1:?] OF product_context;
END_ENTITY; -- 10303-41: product_definition_schema
```

```
#7=PRODUCT('default workpiece','AP-238','',(#2));
```

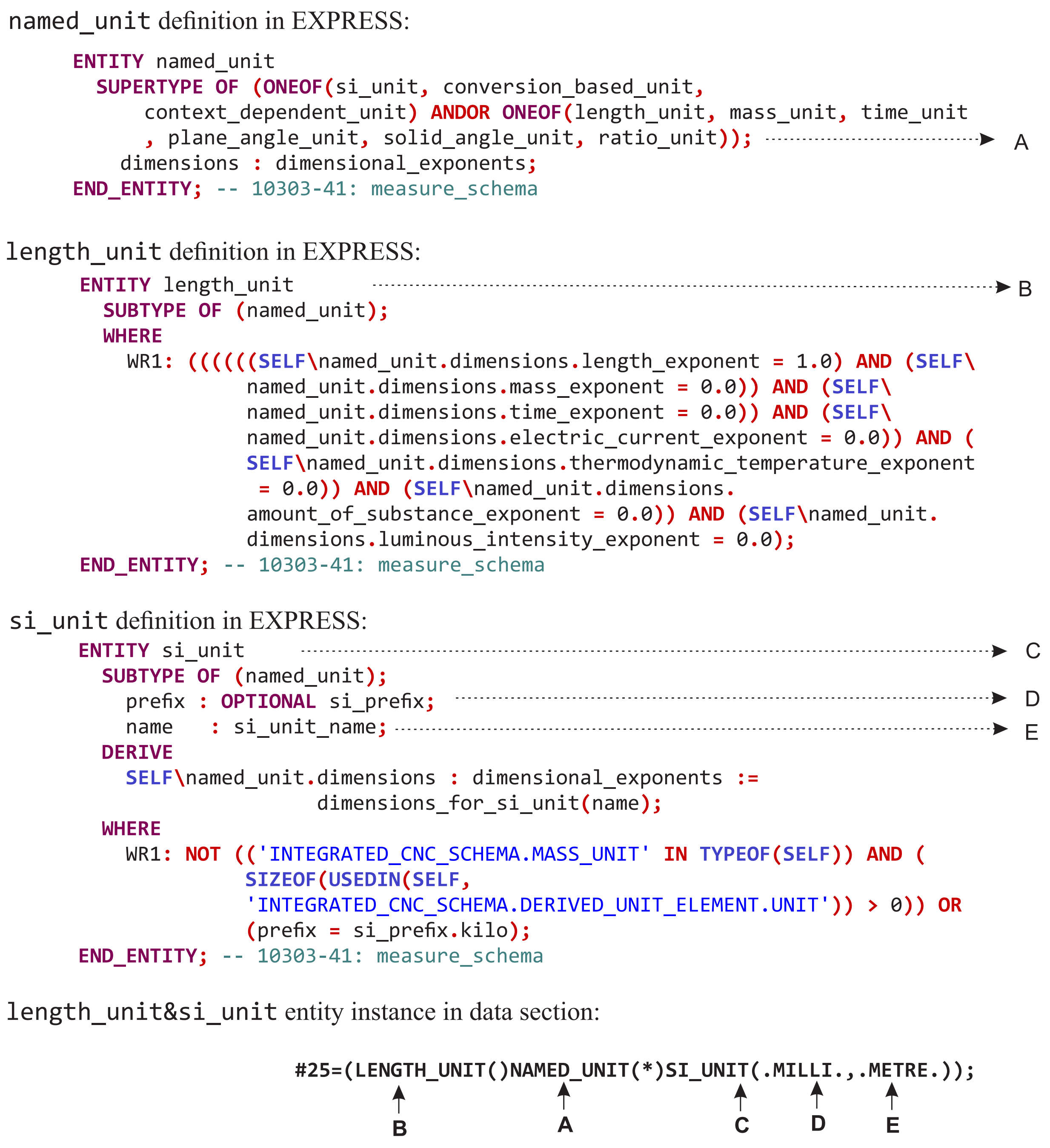
Figure 3.13: Example of Mapping of EXPRESS entity data types.

The external mapping is used when there are several possibilities to include attribute values; for example, when in the entity hierarchy, it includes one or more supertypes using `AND` or `ANDOR` operators to specify its subtypes. The list of attribute values is considered a sublist of attribute values of the main list that defines the entity. The Figure 3.14 shows an example of the mapping of subtypes related by `ANDOR`. The `length_unit&si_unit` entity instance in the data section identified as #25 is `named_unit`, `length_unit` and `si_unit` combined.

An instance can be the context for defining dependent instances and is only visible outside the context if it is explicitly defined. Creating contexts helps represent dependency relationships between entities and reusing entities between schemas.

3.5 Integrated Resources

Integrated features are standard constructs that define groups of objects made up of EXPRESS schemes that incorporate libraries or descriptions of reusable concepts in multiple applications. Integrated resources are composed of two classes, generic integrated resources, and integrated application resources. Generic integrated features are descriptions without dependencies within an application category. The entities described in the generic application resources are responsible for promoting the entities described in the generic integrated resources to highlight applications categories.The integrated resource modules can be developed according to the need of the Application Reference Model (ARM) model (ISO 10303-238,



`named_unit` definition in EXPRESS:

```
ENTITY named_unit
  SUPERTYPE OF (ONEOF(si_unit, conversion_based_unit,
      context_dependent_unit) ANDOR ONEOF(length_unit, mass_unit, time_unit
      , plane_angle_unit, solid_angle_unit, ratio_unit));
    dimensions : dimensional_exponents;
END_ENTITY; -- 10303-41: measure_schema
```

`length_unit` definition in EXPRESS:

```
ENTITY length_unit
  SUBTYPE OF (named_unit);
  WHERE
    WR1: (((((SELF\named_unit.dimensions.length_exponent = 1.0) AND (SELF\
              named_unit.dimensions.mass_exponent = 0.0)) AND (SELF\
              named_unit.dimensions.time_exponent = 0.0)) AND (SELF\
              named_unit.dimensions.electric_current_exponent = 0.0)) AND (
              SELF\named_unit.dimensions.thermodynamic_temperature_exponent
               = 0.0)) AND (SELF\named_unit.dimensions.
              amount_of_substance_exponent = 0.0)) AND (SELF\named_unit.
              dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY; -- 10303-41: measure_schema
```

`si_unit` definition in EXPRESS:

```
ENTITY si_unit
  SUBTYPE OF (named_unit);
    prefix : OPTIONAL si_prefix;
    name   : si_unit_name;
  DERIVE
    SELF\named_unit.dimensions : dimensional_exponents :=
                    dimensions_for_si_unit(name);
  WHERE
    WR1: NOT (('INTEGRATED_CNC_SCHEMA.MASS_UNIT' IN TYPEOF(SELF)) AND (
              SIZEOF(USEDIN(SELF,
              'INTEGRATED_CNC_SCHEMA.DERIVED_UNIT_ELEMENT.UNIT')) > 0)) OR
              (prefix = si_prefix.kilo);
END_ENTITY; -- 10303-41: measure_schema
```

`length_unit&si_unit` entity instance in data section:

```
#25=(LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
```

Figure 3.14: Example of the external mapping of subtypes related by ANDOR.

2007).

3.6 Application Interpreted Constructs

Within the group of data modeled in EXPRESS, some structures maintain some similarity. In other words, it has data structures with common areas for different applications. These common areas are grouped into interpreted application constructions like independent modules of data valid for various applications. AIC are data specifications that satisfy a need for product data to achieve various contexts. Interpreted application constructs specify a data structure and the semantics used to exchange common product data between two or more application protocols. The AP with similar information requirements are compared semantically to determine standardized functional equivalences within Application Interpreted Constructs. AIC are used by both application protocols and are available for use by other future AP. When it is necessary to satisfy standard data requirements, Application Interpreted Constructs provide the standardized mechanisms in the ISO 10303-50 parts (XU, 2009; ISO 10303-238, 2007).

3.7 Conformance Classes

Conformance Classes are specified in the Application Activity Model (AAM) model to highlight the needs of the industrial application. With the conformity classes, the requirements are specified to meet the established objectives. Compliance classes make it easier for users to decide the type of functions they want to admit to their systems. The AP238 application protocol contains four conformity classes.

- Tool path programming (CC1)
- Closed-loop programming (CC2)
- Feature-based programming (CC3)
- Generative programming (CC4)

These conformity classes are defined so that each class includes all the options specified by the previous class. The support for a particular compliance class requires the support of all the options specified in that class (ISO 10303-238, 2007).

3.8 Abstract Test Suites

The specifications based on application protocols are complex activities of little automated analysis and design with several error possibilities. Compliance testing is part of an application protocol and needs to be defined from the project's beginning. Abstract, tests as a procedure for evaluating a component of the application protocol are independent of the test implementation. The development of abstract tests is a complex task that is not strictly specified. The definition for implementing the tests is specific to each application protocol. The standard provides guides that indicate the implementation focus, but not the strict test definition.

To verify that the compliance classes are satisfactorily fulfilled, a test methodology is required. The test methodology is specified according to the ARM model. The methodology identified the means to validate and verify the AIM model. Compliance testing covers two parts of the STEP standard: compliance testing structure and methodology, which describes how to test the implementations and are standardized in the ISO 10303-30 series. The second part makes up the abstract tests. The abstract tests contain the set of possibilities necessary to test implementing a STEP application protocol. Each abstract test case specifies the input data to be provided to the implementation under test and how to assess that implementation's capabilities (BENAVENTE, 2011; PLANTEC, 2007).

According to the expected results, performing the abstract tests consists of specifying the means to evaluate an AIM model. In other words, it is a simulation-based on a data architecture that comprises pre-processing and post-processing. The pre-processing is in charge of coding in STEP format or producing functional calls from SDAI to perform instantiation in the AIM model. The post-processor reads the STEP format data or performs SDAI operations and encodes the data in a specific output format. The syntactic analyzes consist of verifying the product according to the rules of representation of the data specified in the ISO10303-21 standard or according to the use of the SDAI operations specified in ISO10303-22 standard. Structural analyses consist of verifying that the data is correctly constructed and that all relationships (global and local) are respected according to the AIM model. The semantic analysis consists of assessing the consistency of data within the domain and context of the application.

3.9 Application Protocols

Application protocols are considered conceptual models that provide information on necessary product data within an industrial context. Many of the components of an application protocol are designed in specific terminology for an application domain. The information commonly provided is EXPRESS diagrams, integrated resource specifications, compliance tests, and implementable data specifications of the STEP standard. AP are the central component of the STEP architecture developed primarily to support and facilitate the development of AP.

3.9.1 Application protocol AP242

The transition from 2D product models to a 3D representation occurred in response to the digital age demands and the need to advance in quality and productivity. Currently, many manufacturing processes use 3D models to plan their manufacture and obtain better use of manufacturing resources. The MBD is the product of integrating a data structure with a descriptive information model. Within this data structure, not only information about dimensions and geometries is linked, but also additional information such as GD&T has space to be transmitted. The MBD plays an essential role in advanced and intelligent manufacturing, supports and leads the digital Thread concept that refers to the digital integration of the product life cycle (Design, manufacturing, and inspection).

The AP242 (Industrial automation systems and integration — Product data representation and exchange — Part 242: Application protocol: Managed model-based 3D engineering) application protocol extends and updates the representation capabilities of AP214 (Industrial automation systems and integration — Product data representation and exchange — Part 214: Application protocol: Core data for automotive mechanical design processes), maintaining the fundamental structure inheriting the information contained in the AP214. The information contained in the AP214 is linked in the AP242 to use the same definitions, take advantage of the description of concepts, and have syntactic and semantic homogeneity.

The AP242 protocol inherits product representation, application context, geometric context, and shape. The data model is based on the predefined structure of the Unit of functionality (UoF) measured data in AP214. The STEP standard as a neutral data format was developed to exchange and share product information. ISO 10303-242 was published in 2014 as an application protocol that includes AP203(Industrial automation systems and integration — Product data representation and exchange — Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies) and AP214. The AP 242 was developed using a modular architecture; it allows the implementation of an intelligent manufacturing system because the model is associated with PMI (dimensioning geometric and tolerances, annotation, symbols) as a semantic representation.

3.9.2 Application protocol AP238

This part of the ISO 10303 standard specifies using the integrated features necessary to define manufacturing requirements using machining utilizing computer numerical control. The associated data for the following activities are within the scope of AP238:

- mechanical parts for manufacturing;
- manufacturing process descriptions, including manufacturing operations, sequences of operations, and associated information as defined in ISO 14649;
- the AS-IS and TO-BE shapes of a mechanical part;
- manufacturing features of a part;
- manufacturing tolerance requirements of a part;
- tool requirements for machining operations;
- tool paths for machining operations;
- manufacture of mechanical products using manufacturing processes defined in ISO 14649;
- manufacturing product discipline view

3.9.2.1 Information requirements

Information requirements are specified as a set of functionality units, application objects, and application assertions. The information required for manufacturing through numerical control machining and associated processes is specified. These assertions pertain to individual application objects and relationships between application objects.

The UoF specified in this application protocol are:

- **measure**: Specifies the representation of physical quantities by their value and unit and the permitted variation of the quantity. The objects listed in this UoF are defined in ISO 14649-10 but are extended to the ISO 10303 standard with additional information requirements. The measurement definitions in ISO 14639-10 do not explicitly specify a unit; instead, it is defined for each quantity (mm for length, degrees for angles), and only length parameters can be qualified with a tolerance. ISO 10303-238 extends these definitions to allow for tolerances in other units of measurement.

- **project**: Specifies the start of the interpretation of a machining program and additional management information about machining.
- **workpiece**: Specifies the mechanical product that a machining program must produce. The description can include the material, finish, properties, and shape of the product.
- **manufacturing feature**: Specifies the information needed to identify forms of interest in mechanical products. These shapes represent volumes of material that are removed by machining operations. This UoF also specifies the information needed to describe a machining feature using a 2D profile scanned along a path and information describing the upper, lower, and other limits of the feature.
- **executable**: Specifies the information needed to describe the control flow of a part program and actions outside of machining that can be performed by numerical control. They include sequential, parallel, and conditional flow and logical expressions and variable elements necessary to describe how a mechanical product can be positioned and oriented relatively on any machine tool to execute a part-program.
- **operation**: Specifies the information needed to describe machining aspects independent of technology, which can be done by numerical control. The technological specification aspects of machining actions are described for the milling and turning processes.
- **toolpath**: Specifies the information needed to describe the cutting tool offset as a pre-calculated path or set of parameters that can be converted into an exact movement by the numerical control.
- **process data for milling**: specifies the information needed to describe the milling, tools, strategies, and specific aspects of machining actions that a numerical control can perform. It includes the information needed to describe strategies and process parameters.
- **cutting tools for milling**: This UoF specifies the information needed to describe the milling and tool requirements for machining actions.
- **manufacturing feature for turning**: UoF specifies the information needed to identify forms of interest for a mechanical product. The shapes represent volumes of material removed by turning operations or the result of a series of turning operations.
- **process data for turning**: This UoF specifies the information needed to describe specific aspects of turning, which can be done by numerical control. Includes definitions for describing turning process strategies and parameters.
- **cutting tools for turning**: Specifies information to describe specific turning requirements for machining actions.

- **geometric dimensioning and tolerancing**: The UoF specifies the information to describe the geometric dimensions and the allowable variations for manufacturing using turning. Besides, this UoF specifies information to describe geometry tolerances concerning parallelism or perpendicularity and geometry tolerances with no data on references to linearity or flatness. It includes the information needed to describe simple reference data, common reference data, objective data, and tolerance range.

3.9.3 Application protocol AP219

Dimensional inspection can be done at any stage of a product's life cycle to verify project specifications compliance. The STEP standard through the AP219 application protocol defines the context, scope, and information requirements for analyzing and reporting information from dimensional inspection results. The protocol's primary focus is to provide a connection to inspection programs based on ISO 22093 (DMIS 4.0), web-based analysis applications, and MIP-assisted practice reporting Metrology Interoperability Project (MIP) (ISO…, 2011). The AP219 protocol provides a mechanism for exchanging inspection information with standard manufacturing data structures such as the ISO 10303-224 and ISO 10303-238 application protocols. The information provided by DMIS and MIP can be mapped within AP219 entities and related to the entities used in the STEP ISO 10303 standard (ISO 10303-219, 2007).

The interpretation of the protocol's integrated features defines the relationship between the information requirements and the AIM model. AIM's provided list specifies the mechanism for integrating the necessary resources in analyzing and reporting dimensional inspection of parts or assemblies. The associated data for the following inspection activities are within the scope of AP219:

- data for administering, planning;
- data for executing dimensional inspection;
- data for archiving the results of a dimensional inspection;
- interface for capturing technical data out of the upstream application protocols;
- machining feature classification structure;
- geometric and dimensional tolerances of the parts being manufactured;
- references to standards and specifications declared in the dimensional inspection.

3.9.3.1 Information requirements

Information requirements are specified through units of functionality, application objects, and application statements. The statements refer individually to each of the objects of application or relationship existing between them. The UoF contained in this application protocol for exchanging dimensional inspection information related to mechanical product definitions are:

- `administrative_data`: This UoF contains the information used to administer product data. Administrative information is associated with programs for performing dimensional measurement.
- `dimensional_measurement_analysis`: Defines a collection of possible preferences for calculating the parameters of tolerances and properties for the measured data.
- `dimensional_measurement_documentation`: This UoF provides entities to specify documents that are directly related to product data. Documents can be specific to an operation on the part being manufactured or a property given at a particular stage in the manufacturing process.
- `dimensional_measurement_execution`: The UoF contains the information for execution and the points defined for inspection operations on a part. The data analysis of the points is done to determine parameters and compare the tolerances.
- `dimensional_measurement_feature`: This UoF has the information needed to identify shapes representing volumes of material that need to be dimensionally inspected on a part.
- `dimensional_measurement_part`: The UoF Contains information necessary to identify the part and its input properties in the dimensional inspection functions.
- `dimensional_measurement_parameters`:It contains the parameter information needed to record and store the inspection results.
- `feature_definition_item`:Provides information for creating a machining feature, additionally identifying the relationship between machining feature and shape aspects.
- `feature_profile`: Contains information for identifying 2D shapes. Features 3D is created by scrolling through the Feature_profile.
- `manufacturing_feature`: The UoF contains the information needed to identify shapes representing volumes of material that must be removed from a part by machining.
- `functional_limitations`: This UoF contains the information necessary to identify the critical quantities in the measurement relationships between aspects of the shape of a piece, relationships between the shape of a piece and a shape reference, and the acceptable deviation of the quantity relationship for manufacture.
- `part_properties`: It contains the description of the characteristics of the part being measured. The characteristics specify requirements for dimensional inspection applied before making the inspection or during the parts inspection.
- `program_run`: UoF specifies the program used on the coordinate measuring machine to produce dimensional measurement data.
- `shape_representation_for_machining`: It contains the physical definition of the initial and final shape of a piece. This definition is given via parametric methods using features, geometries, and topologies.

3.10 Summary

The concepts related to the STEP neutral data exchange structure useful for an integration and implementation perspective were addressed. In the architecture of the STEP standard, two approaches are highlighted: the functional and the conceptual. The STEP standard satisfies integration requirements due to its consistent data structure and supports interoperability due to its technological independence. The STEP standard provides an exchange structure and controls the data to be exchanged through its standardized definitions. The EXPRESS language is emphasized among the description methods that support the data structure since it is a descriptive language used in the logical layer of the STEP architecture and responsible not only for describing concepts, relationships, and attributes but also for expressing knowledge through schematics. Each application protocol contains a schema that conveys the scope. The chapter presents the application protocols used in developing the proposal, such as AP219, AP238, and AP242.

Chapter 4

Conceptual Framework Proposal for Closed-Loop Inspection

4.1 Introduction

This chapter presents the scope of the information model created for the development of this project. The information model shows three aspects, data structure, functionality, and implementation models. The project's purpose covers the integration scenario that supports the design, manufacturing, and measurement activities.

4.2 Information Modeling

Information modeling through language consistently describes the requirements and specifications used to create information storage, reading, writing, and retrieval operations. Information modeling specifies the data requirements used within an application domain by representing concepts, relationships, constraints, rules, and operations within a particular semantics that covers the application's domain of discourse. The information model provides a neutral, sharable, stable, and organized information structure to understand the requirements and specifications of an application domain and can be accessed by any application regardless of the technology and technique used.

This chapter introduces information modeling that covers three models (AAM, ARM, and AIM). The functional development uses IDEF0 schemas in the construction of the AAM model. The EXPRESS schemas show the conceptual development represented by the ARM models, and mapping tables relate the definitions to implementation activities through the AIM model.

4.3 Functional Modeling

The AAM is a method used to describe the activities associated with a given application or implementation context. The AAM model provides a structure to list the functional activities involved in the specific application process. The method is used in the STEP standard to show the scope. In the AAM model made in the IDEF0, the functional blocks that make up the system are presented. The IDEF0 method provides the tools to do complete functional modeling to define the information flow and establish a hierarchy of activities.

An AAM model consists of requirements defined within a block expressed as inputs, outputs, controls, and mechanisms. The conceptual model is created with levels and sublevels to present the functional activities associated with the system in detail. The model allows seeing the system's basic requirements; at this level, the number of related entities is less. All subsequent steps in the deployment path are based on this model, and therefore it is essential to define the activities and relationships of the system comprehensively and precisely.

AAM model development covers inspection process planning, data analysis, and dimensional measurement results of prismatic parts manufactured by machining. The inspection process is carried out in the final stage of the product's manufacturing cycle when required to comply with the design specifications. Manufacturing prismatic parts should preferably be done in an integrated CAD/CAPP/CAM/CNC environment to ensure error compensation is performed under the same manufacturing conditions.

Two typical manufacturing scenarios considered for developing the AAM model are: when it is necessary to adapt the manufacturing environment for batch production of a new part. This scenario requires a thorough verification of each feature that makes up the design, analyzes the results, and provides feedback on decisions to correct the causes of deviation in the final piece until the project complies with the design specifications. The second scenario contemplated in the AAM model consists of correcting deviations detected in the middle of the production by a batch of parts; the inspected feature is the feature that is outside the design specifications. The data extracted from the measurement allows detecting the source of the error and re-feeding the results for correction or compensation in the design. The data returned for the CAD model are tolerance values adjusted to the new manufacturing conditions that allow satisfying the design requirements. The data analysis and the inspection report are feedback to the manufacturing process, with updated tolerance values, which meet the project specifications. Tolerance values are calculated within a compensation strategy and statistical methods for their determination. The inspection strategy can be applied in other contexts, such as turning and additive manufacturing processes with appropriate settings.

The implementation of the inspection platform is developed by mapping the schemas made in the EXPRESS language to the SDAI data model using JSDAI API to create JAVA classes of related entities. The STEP Modeler tool has integrated CAD / CAPP/CAM systems and forms the basis for implementing the closed-loop of manufacturing and inspection. Other computational tools can be included to aid integration. Figure 4.1 shows the development scenario, including the tools available for its implementation. In the diagram, it is possible to observe the data flow and the essential elements of a closed-loop inspection system. The red lines represent the interoperability barriers that make integration difficult, causing ambiguities, loss of data consistency, and gaps in the two-way communication of information.

In the design phase, the STEP Modeler system, developed in UFSC, is inherited. The STEP Modeler system generates a file in neutral format with product information. It is necessary to evaluate the integrity of the information required for closed-loop inspection, such as shape specifications, tolerances, and their relationship with geometry. For the planning, execution, analysis, and generation of the inspection process

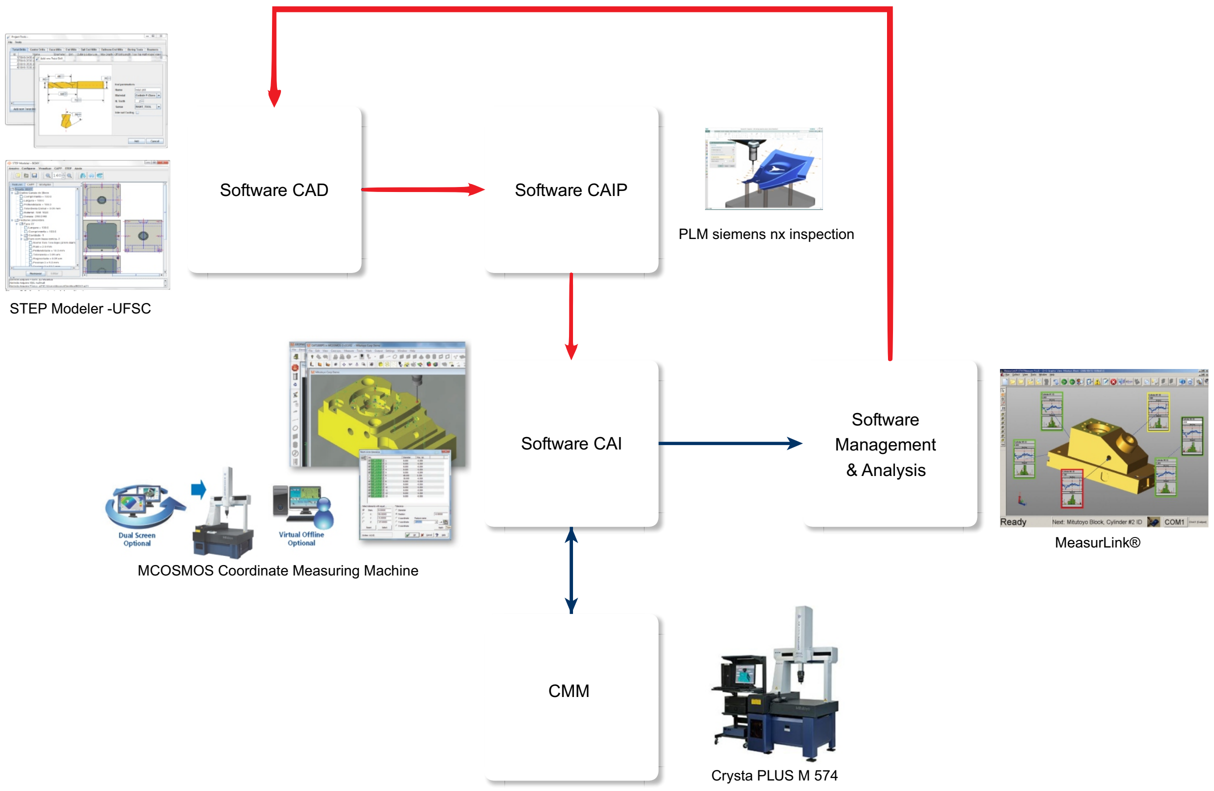


Figure 4.1: The development scenario.

results, tools and a proprietary system will be used to adhere to the information in the STEP file (p21).

4.3.1 Architecture for Inspection Integration in Closed-Loop Manufacturing

Figure 4.2 presents a general diagram with the necessary resources to create a closed-loop manufacturing and inspection system. The system is fully integrated with digital technologies and adheres to the concept of Industry 4.0. In the integration model, the selected data structure is provided by the STEP standard operating within a closed-loop architecture. The STEP standard provides the support, resources, and product definitions to standardize in different areas of the digital industry. The dimensional and geometric inspection process is performed with CMM to obtain the measurement data. With the open, neutral, and extensible data model, exchanging information in the manufacturing processes is possible.

The model includes activities related to the fabrication of a part that spans different architecture levels, such as application, logic, and physics. The main problem was to avoid the transformations that the information undergoes in each of the CAD/CAPP/CAIP/CAM/CNC/CAI systems, caused by incompatibilities of the formats, some of them proprietary. The AAM model represents the functions, resources, constraints, and requirements used to execute each linked process in the manufacturing and inspection chain. The model also establishes the application context and the exchange data structure used between each functional block to integrate the processes in the digital chain.

The main activities related to part manufacturing and inspection within the digital manufacturing chain are modeled in IDEF0. The model aims to present the set of activities considered in the implementation

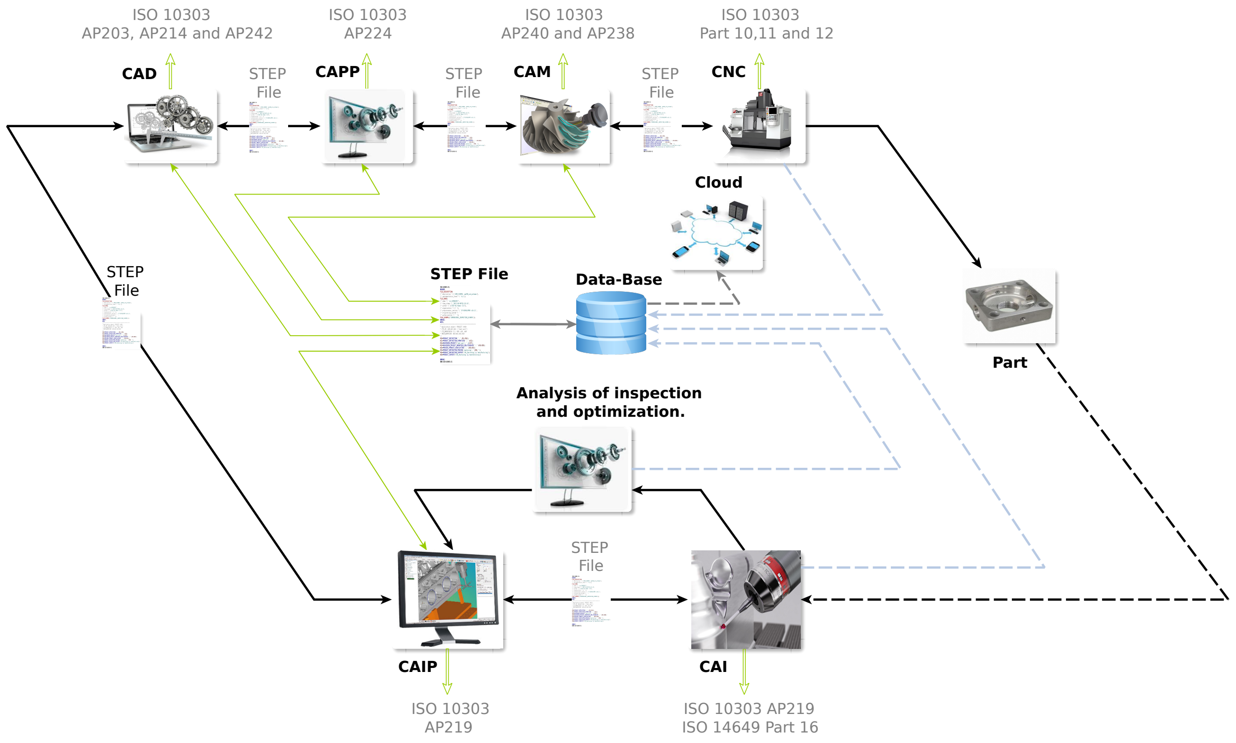


Figure 4.2: Digital chain for the integration of closed-loop manufacturing and inspection
Source:(JAIMES et al., 2018)

and defines the specifications for the planning and execution of the dimensional and geometric inspection. CAD/CAPP/CAM/CAIP/CAI integration is based on four functional activities: design, planning (machining and inspection), process control (machining and inspection), and analysis of results to generate corrections. The information is structured in a physical STEP file, encoded according to ISO 10303-21, which provides the control structure for the execution sequence of a STEP-NC-based program. The file contains the strings of Workingsteps and functions related to the operation of the machine. Workingstep is defined as specific operations carried out on a machine tool CNC, subdividing each machining operation into the steps necessary to carry out the operation.

4.3.2 Information Model for Integration of Manufacturing and Inspection Data

The main activities related to manufacturing and inspection of parts operating within a digital manufacturing chain are modeled using the IDEF0 language. The model presents the set of activities considered for implementation within the manufacturing context, and the specifications for the planning and execution of the dimensional and geometric inspection processes are defined. CAD/CAPP/CAM/CAIP/CAI integration is based on four functional activities that are: design, manufacturing planning and inspection, control of both manufacturing and inspection processes, and analysis of results. The information is structured in a physical STEP file, built with the ISO 10303-21 standard, which provides the control structure for a STEP-NC-based program's execution sequence. The file contains the `workingstep` sequences and functions related to the operation of the machine. The `workingstep` is defined as specific operations performed on a CNC machine tool, subdividing each machining operation into the steps necessary to operate.

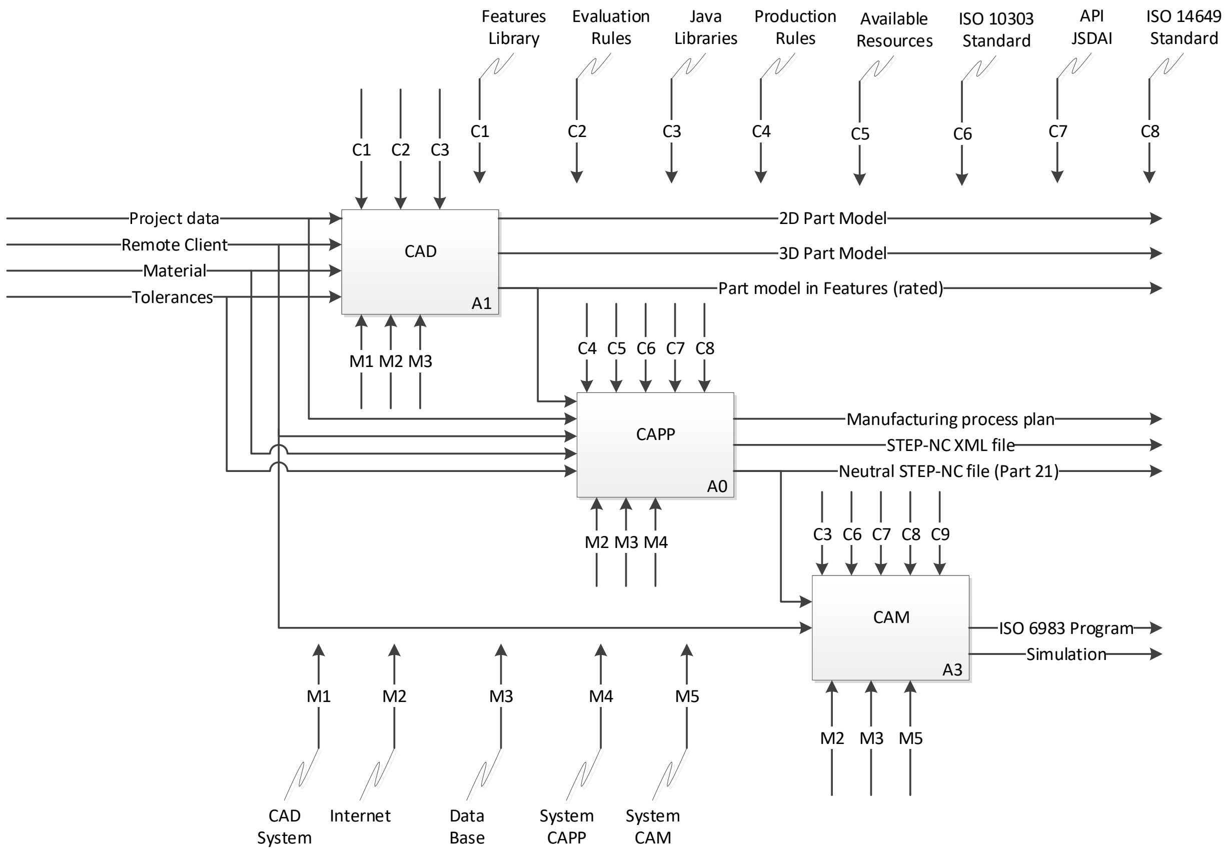


Figure 4.3: IDEF0 STEP Modeler developed at the Federal University of Santa Catarina (UFSC). Source:Adaptation (BENAVENTE, 2011)

The IDEF0 diagram of the integrated system shown in Figure 4.3 presents the functional blocks inherited from the "STEP Modeler" system developed at the University of Santa Catarina. The CAD function block uses a features modeling approach and provides support for the design of prismatic parts. The Computer-Aided manufacturing (CAPP) function block aims to support the part process manufacturing by planning through tasks that generate and organize `workingstep`, determine the support points, create the plan, and create the file Physical STEP-NC file. The CAM function block has the purpose of interpreting the physical file and making it possible to manufacture the part. The procedures to read the neutral file are performed in addition to post-processing according to ISO 6983 and simulating the movement (BENAVENTE, 2011).

The AAM model for the dimensional and geometric inspection process is integrated with the STEP Modeler system. CAIP/CAI functional activities are incorporated to perform the dimensional and geometric inspection in a closed-loop adherent to the STEP-NC standard.

Figure 4.4 presents the functional activities that make up the CAD/ CAPP/CAIP/CAM/CAI integration model. The model shows the list of the CAD/CAPP/CAIP/CAM/CNC/CAI digital manufacturing chain’s main functional blocks. This section is of interest to introduce the functional blocks that represent computer-aided planning and inspection activities. It is important to note that in IDEF0 diagrams, it is

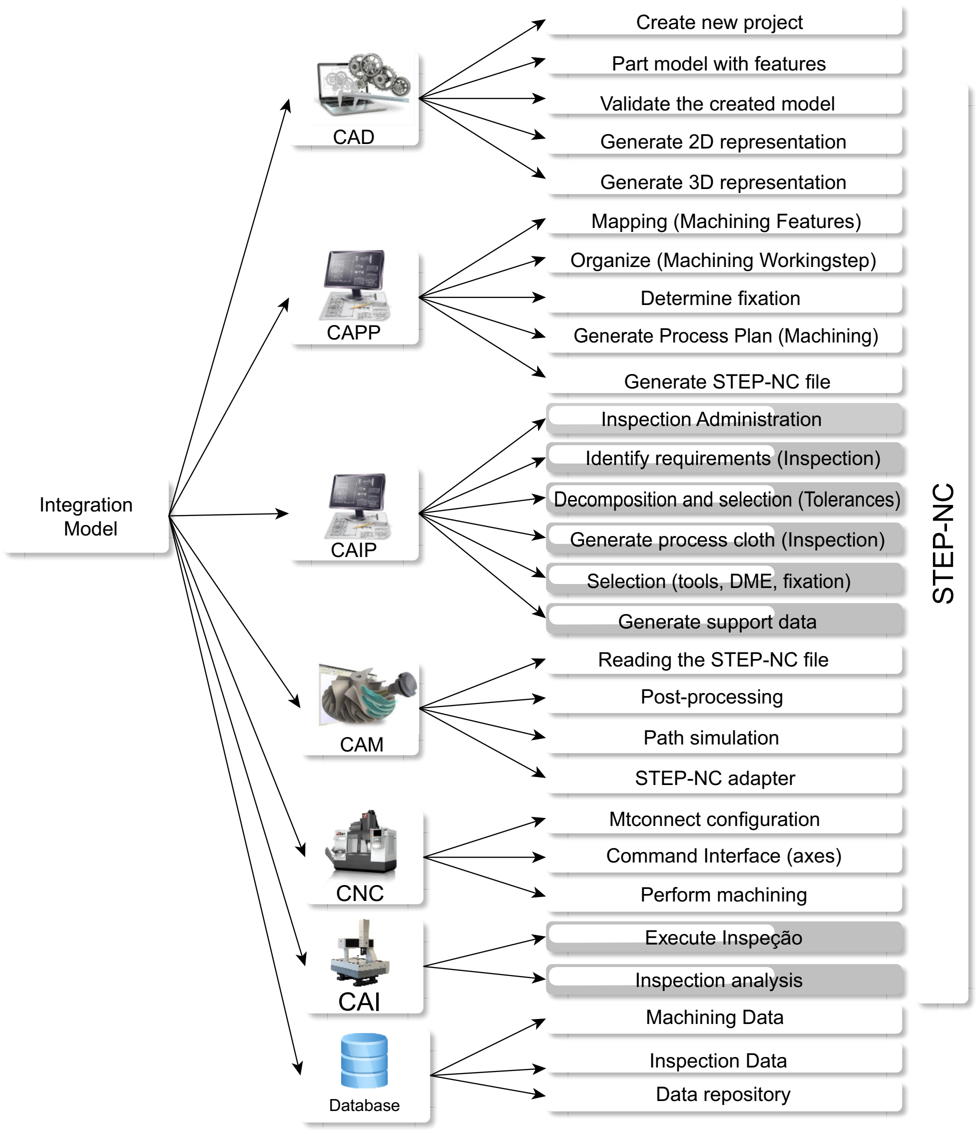


Figure 4.4: Scope of functional activities that make up the CAD/CAPP/CAIP/CAM/CAI integration model.

possible to model activities within a sequence and hierarchy, creating levels and sublevels to meet this objective. The appendix A.1 shows the complete AAM model developed by inheriting the STEP Modeler system's models and how the closed-loop inspection system is implemented.

Diagram A0 (shown in Figure 4.5) presents the general context in which the digital manufacturing chain's integration occurs. The primary input data are associated with the product design specifications, material to be used in the manufacture of the part, tolerance requirements, tools, and resources available to execute the CAD/CAPP/CAIP/CAM/CNC/CAI processes. The controls required for the integration model are a diverse set of standardized data structures that provide relationships, functions, and rules for transferring information in each process, ensuring quality standards are maintained and meeting project requirements. The mechanisms that allow the generation of products are made up of human resources, teams, applications software, databases, and knowledge. The model results are files with part design, simulations, analysis, machining routines, manufacturing plans, inspection, and parts.

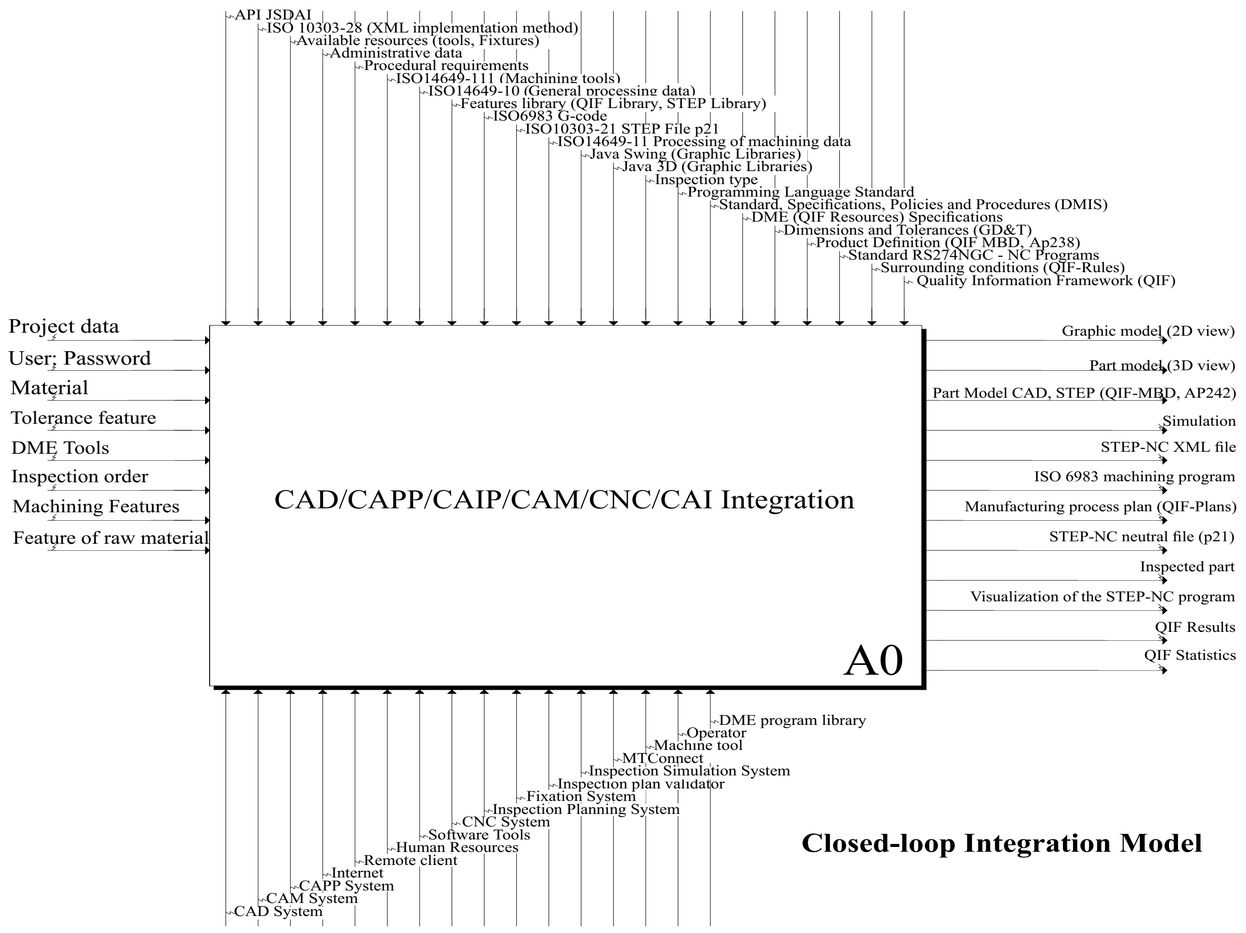
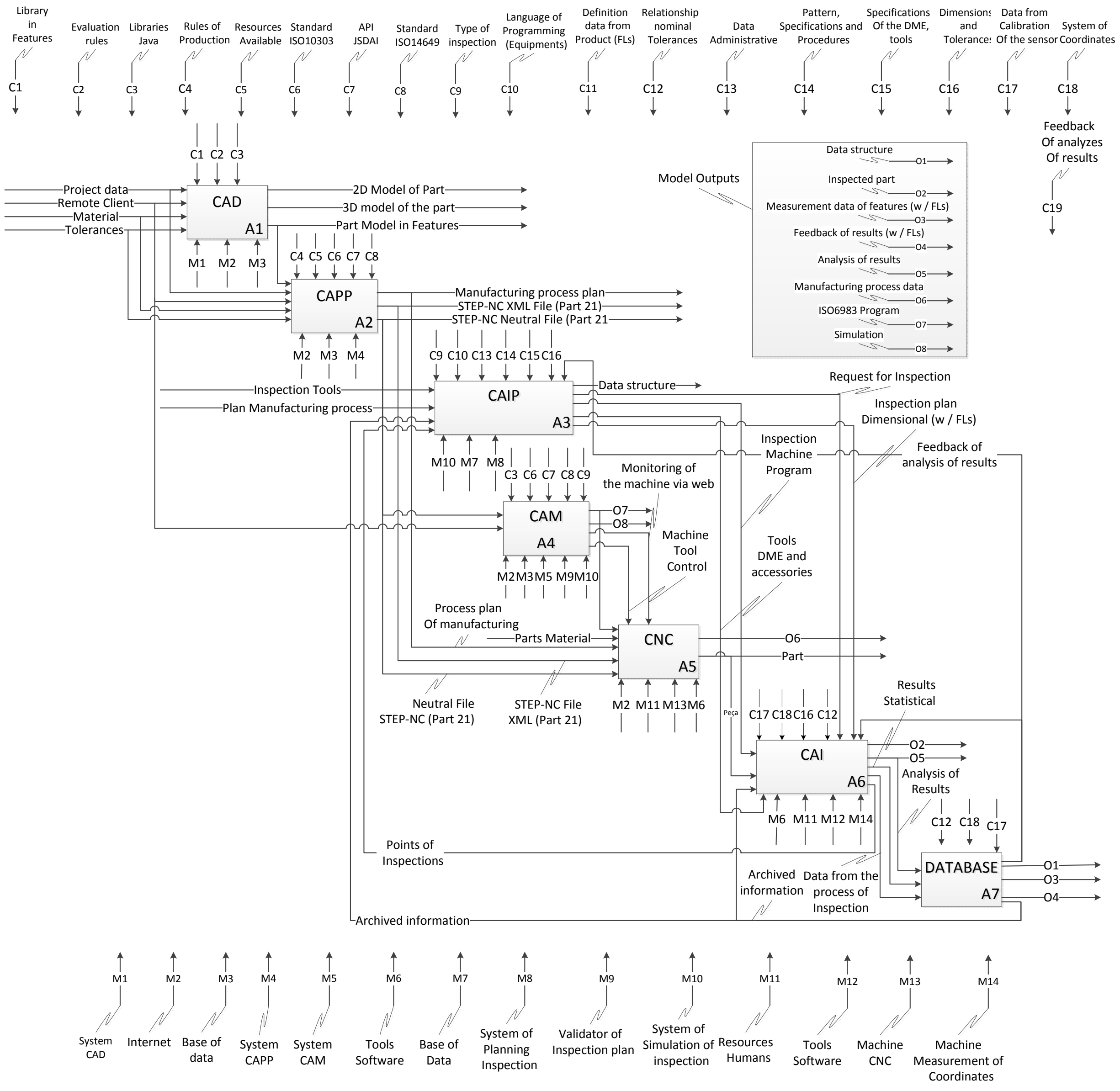

API JSDAI
ISO 10303-28 (XML implementation method)
Available resources (tools, Fixtures)
Administrative data
Procedural requirements
ISO14649-111 (Machining tools)
ISO14649-10 (General processing data)
Features library (QIF Library, STEP Library)
ISO6983 G-code
ISO10303-21 STEP File p21
ISO14649-11 Processing of machining data
Java Swing (Graphic Libraries)
Java 3D (Graphic Libraries)
Inspection type
Programming Language Standard
Standard, Specifications, Policies and Procedures (DMIS)
DME (QIF Resources) Specifications
Dimensions and Tolerances (GD&T)
Product Definition (QIF MBD, Ap238)
Standard RS274NGC - NC Programs
Surrounding conditions (QIF-Rules)
Quality Information Framework (QIF)
Project data
User; Password
Material
Tolerance feature
DME Tools
Inspection order
Machining Features
Feature of raw material
CAD/CAPP/CAIP/CAM/CNC/CAI Integration
A0
Graphic model (2D view)
Part model (3D view)
Part Model CAD, STEP (QIF-MBD, AP242)
Simulation
STEP-NC XML file
ISO 6983 machining program
Manufacturing process plan (QIF-Plans)
STEP-NC neutral file (p21)
Inspected part
Visualization of the STEP-NC program
QIF Results
QIF Statistics
DME program library
Operator
Machine tool
MTConnect
Inspection Simulation System
Inspection plan validator
Fixation System
Inspection Planning System
CNC System
Software Tools
Human Resources
Remote client
Internet
CAPP System
CAM System
CAD System
Closed-loop Integration Model

Figure 4.5: Activity model level A0

The functional activities that make up the CAD/CAPP/CAIP/CAM/CNC/CAI integration model are shown in Figure 4.6. Dimensional and geometric inspection within a closed manufacturing cycle is intended to improve the process and control its evolution. The main activities integrated into the functional blocks are listed below:

- Analyze which of the required parameters are measurable.
- Determine the acceptable uncertainty value for the measurement.
- Interpret the results and calculate the error.
- Transform the data into useful information and share it within a date of manufacture.



Library in Features C1
Evaluation rules C2
Libraries Java C3
Rules of Production C4
Resources Available C5
Standard ISO10303 C6
API JSDAI C7
Standard ISO14649 C8
Type of inspection C9
Language of Programming (Equipments) C10
Definition data from Product (FLs) C11
Relationship nominal Tolerances C12
Data Administrative C13
Pattern, Specifications and Procedures C14
Specifications Of the DME, tools C15
Dimensions and Tolerances C16
Data from Calibration Of the sensor C17
System of Coordinates C18
Feedback Of analyzes Of results C19
CAD A1
Project data
Remote Client
Material
Tolerances
2D Model of Part
3D model of the part
Part Model in Features
CAPP A2
Manufacturing process plan
STEP-NC XML File (Part 21)
STEP-NC Neutral File (Part 21)
CAIP A3
Inspection Tools
Plan Manufacturing process
Data structure
CAM A4
O7
O8
Monitoring of the machine via web
Machine Tool Control
CNC A5
Process plan Of manufacturing
Parts Material
O6
Part
Neutral File STEP-NC (Part 21)
STEP-NC File XML (Part 21)
CAI A6
Peça
O2
O5
Analysis of Results
Results Statistical
DATABASE A7
O1
O3
O4
Request for Inspection
Inspection plan Dimensional (w / FLs)
Inspection Machine Program
Feedback of analysis of results
Tools DME and accessories
Points of Inspections
Archived information
Data from the process of Inspection
Model Outputs
Data structure O1
Inspected part O2
Measurement data of features (w / FLs) O3
Feedback of results (w / FLs) O4
Analysis of results O5
Manufacturing process data O6
ISO6983 Program O7
Simulation O8
M1 System CAD
M2 Internet
M3 Base of data
M4 System CAPP
M5 System CAM
M6 Tools Software
M7 Base of Data
M8 System of Planning Inspection
M9 Validator of Inspection plan
M10 System of Simulation of inspection
M11 Resources Humans
M12 Tools Software
M13 Machine CNC
M14 Machine Measurement of Coordinates

Figure 4.6: A0 internal level activity model

4.3.3 The information model of the CAIP and CAI Systems

Figure 4.7 shows the activities related to the CAIP / CAI functional blocks, where each activity's levels and sublevels are observed. Each function must be developed and implemented using the definitions of entities provided by the STEP standard in the AP242, AP238, and AP219 application protocols to achieve the correct integration of data within the closed circuit and obtain the data model that is fully compliant with STEP-NC.

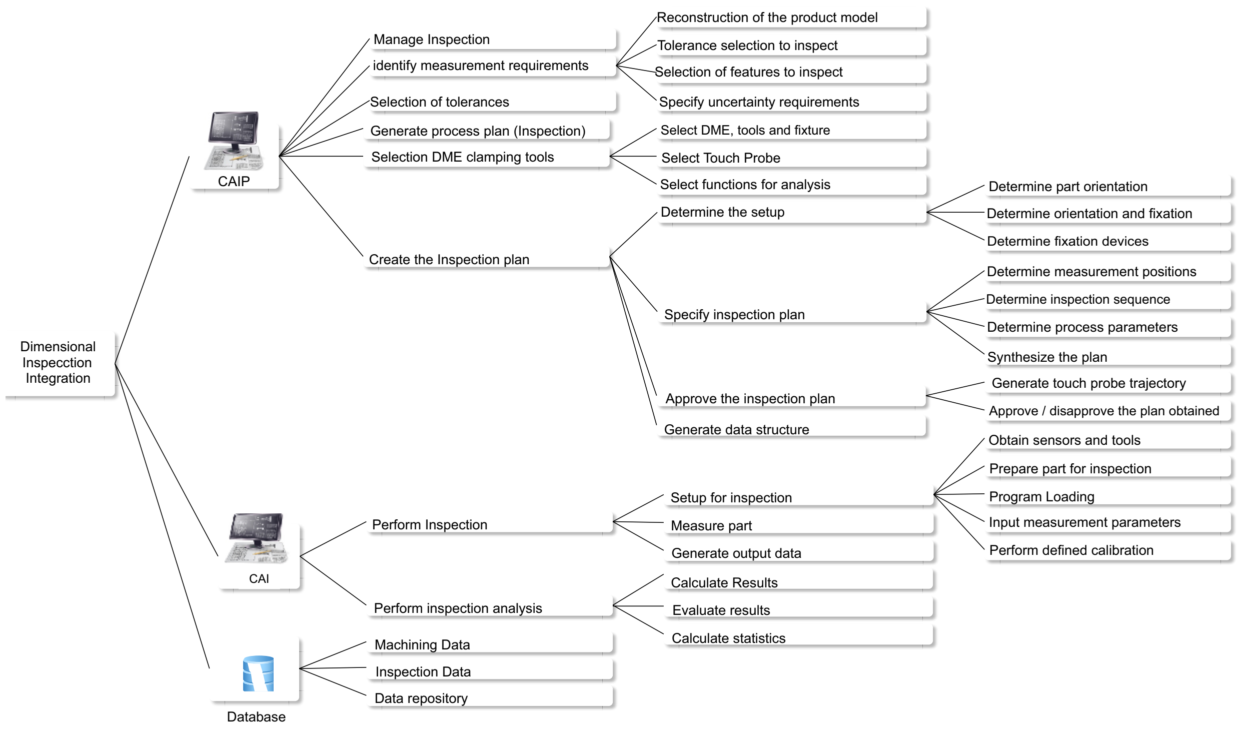

Dimensional Inspecction Integration
CAIP
Manage Inspection
identify measurement requirements
Reconstruction of the product model
Tolerance selection to inspect
Selection of features to inspect
Specify uncertainty requirements
Selection of tolerances
Generate process plan (Inspection)
Selection DME clamping tools
Select DME, tools and fixture
Select Touch Probe
Select functions for analysis
Create the Inspection plan
Determine the setup
Determine part orientation
Determine orientation and fixation
Determine fixation devices
Specify inspection plan
Determine measurement positions
Determine inspection sequence
Determine process parameters
Synthesize the plan
Approve the inspection plan
Generate touch probe trajectory
Approve / disapprove the plan obtained
Generate data structure
CAI
Perform Inspection
Setup for inspection
Obtain sensors and tools
Prepare part for inspection
Program Loading
Input measurement parameters
Perform defined calibration
Measure part
Generate output data
Perform inspection analysis
Calculate Results
Evaluate results
Calculate statistics
Database
Machining Data
Inspection Data
Data repository

Figure 4.7: Scope of activities considered in the AMM model built for CAIP/CAI systems.

The AAM for the CAIP model shown in Figure 4.8 provides a structured view of the activities involved in generating the inspection plan. The model contains the data flow and some general function definitions necessary to transform the inputs into outputs of the process. The AAM model for CAIP consists of six functional blocks, with the following activities:

- Carry out administration and archiving tasks (Block A31).
- Identify inspection requirements (Block A32).
- Decomposition of inspection features and tolerance selection (Block A33).
- Select tools, fasteners, and Dimensional Measurement Equipment (DME) functions (Block A34).
- Develop an inspection plan (Block A35).
- Generate support data (Block A36).

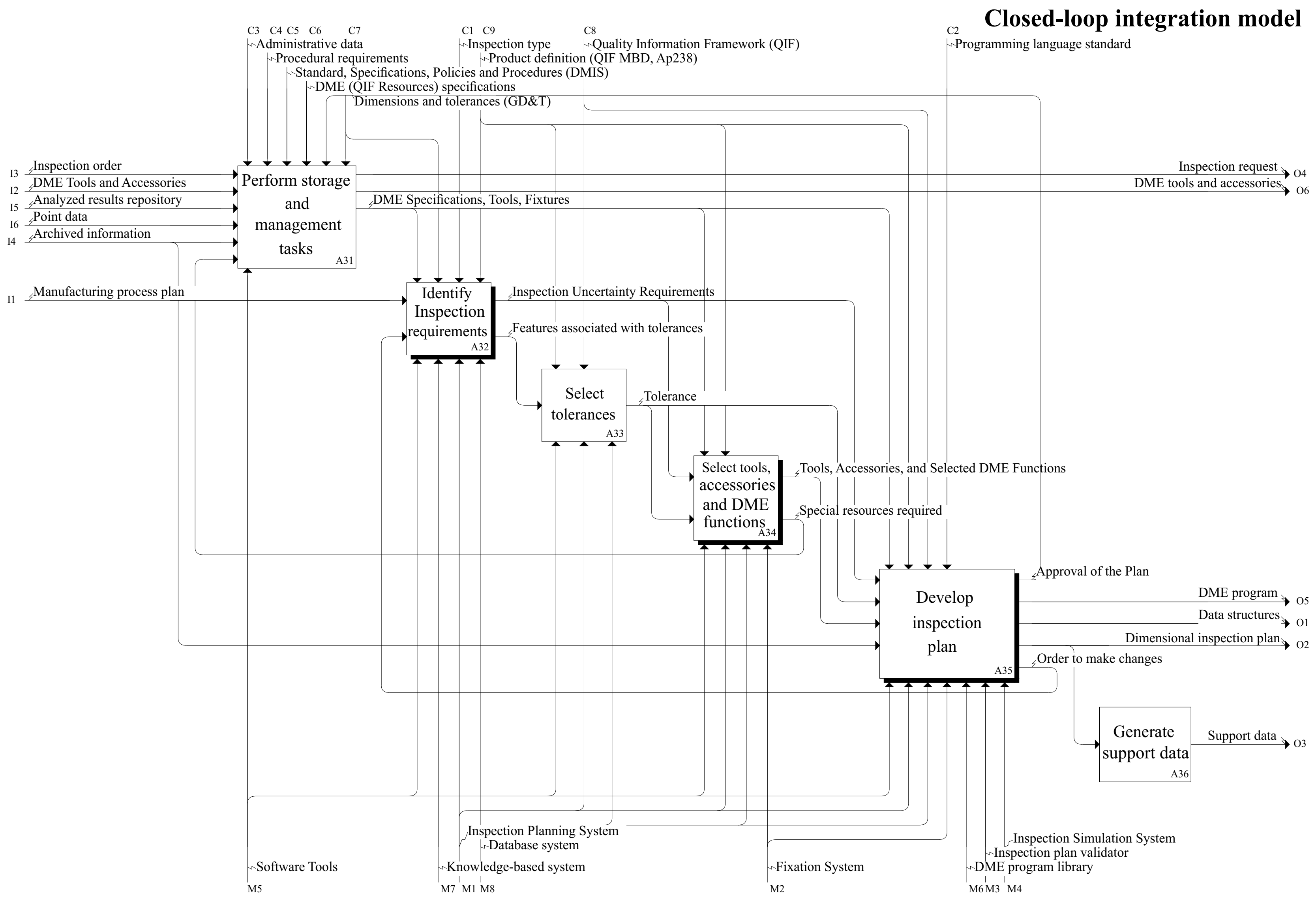

Closed-loop integration model
C3 C4 C5 C6 C7
Administrative data
Procedural requirements
Standard, Specifications, Policies and Procedures (DMIS)
DME (QIF Resources) specifications
Dimensions and tolerances (GD&T)
C1 C9
Inspection type
Product definition (QIF MBD, Ap238)
C8
Quality Information Framework (QIF)
C2
Programming language standard
I3 Inspection order
I2 DME Tools and Accessories
I5 Analyzed results repository
I6 Point data
I4 Archived information
I1 Manufacturing process plan
Perform storage and management tasks
A31
Inspection request O4
DME tools and accessories O6
DME Specifications, Tools, Fixtures
Identify Inspection requirements
A32
Inspection Uncertainty Requirements
Features associated with tolerances
Select tolerances
A33
Tolerance
Select tools, accessories and DME functions
A34
Tools, Accessories, and Selected DME Functions
Special resources required
Develop inspection plan
A35
Approval of the Plan
DME program O5
Data structures O1
Dimensional inspection plan O2
Order to make changes
Generate support data
A36
Support data O3
Inspection Planning System
Database system
Software Tools
Knowledge-based system
Fixation System
Inspection Simulation System
Inspection plan validator
DME program library
M5
M7 M1 M8
M2
M6 M3 M4

Figure 4.8: AMM model for integrating the CAIP process.

The outputs generated in each functional block are integrated into the digital manufacturing chain through feedback with useful information to improve the process. The functions and activities that make up the inspection planning are organized within the IDEF0 diagram at various levels and sublevels.

The AAM model presented in this document was developed to generate a roadmap for the computational implementation of each functional activity required to integrate the inspection results in closed-loop manufacturing. The model illustrates the degree of complexity of these activities and the solution to the problem. A solution is addressed to integrate dimensional and geometric inspection results based on the ISO 10303 standard and the STEP Modeler system. The ARM reference models contain information about the AP219 and AP238 application protocol definitions and integration methods. The AAM model presented allows the ARM model to construct the entities necessary to carry out each functional block's computational implementation. The ARM model is obtained with the definitions and terminology provided by the standard to be used to reference the computational implementation.

This model's main contribution is to present an interoperability solution and create an integration alternative that meets current manufacturing requirements and demands. Regardless of the technology used, the model details the resources necessary to structure the information that allows implementing a closed-loop inspection adhering to the STEP/STEP-NC standard. The challenges present in the project's continuation are associated with the computational implementation and the accurate selection of the technologies involved in the architecture presented in this chapter.

The AAM model presented in this section covers only one level of information, the most external. The complete AAM model that contains the sublevels of the activities can be consulted in the appendix A.1.

4.4 Conceptual Modeling

Generic conceptual models that formalize the recommended implementation practices for a given domain are called reference models. Reference models are used to guarantee a generic, efficient, extensible, and reusable structure in developing an application. The domains represented through an ARM model are extensive and may vary according to their functionality; it is necessary to delimit the domain to cover only relevant aspects of the specific application to be developed. Reference models are derived from conceptual models that continually consolidate knowledge to establish them as a reference and standard to streamline application development within a generic framework. The reference models constitute support in the modeling process by providing a list of structures that need to be related to the application domain, avoiding creating new structures outside the standard. These reference models are found within the project as a library in a neutral language intended to provide modeling support.

This section presents an abstraction of the relevant models for defining requirements within the proposed application domain. ARM models enable the standardization of development and computational implementation within a neutral, syntactically homogeneous, and interoperable architecture. Models are used to specify from a higher level the abstraction, type of objects, and their relationships within the data structure used. The ARM model provides a detailed analysis of the requirements of an industrial application. In this model, the objects or classes and entities of the system are defined. The ARM model is obtained with the definitions and terminology provided by the standard to be used to reference computational implementation. The ARM model development needs to detail the interrelated entities and know each object's definitions and attributes. An object is a mechanism that provides information and can be related, consulted, modified, and stored. Each object and element are defined together within a complex structure (MAHMOUD et al., 2016).

4.4.1 Harmonization Strategy

The STEP standard supports different implementations through UoF given in our application protocols. In section 3.9.2 it is presented a summary of the functional domain two application protocols AP238 and AP219, that specify information related to manufacturing requirements, such as information used to analyze and report inspection results. For the document, open applications for both manufacturing and inspection that share entities structured within the UoF used by AP203, AP214, AP219, AP224, AP238, AP240, and AP242 that support CAD/CAPP/ CAIP/CAM/CNC/CAI systems. The components of each protocol are protected through specific terminology for a domain of application. The main question to be solved is identifying and mapping the entities that will be used in the implementation of the wrong date (ISO 10303-203, 2011; ISO 10303-214, 2010; ISO 10303-240, 2005; ISO 10303-242, 2014).

Harmonize is a concept given to find the correspondence or agreement between similar terms used to exchange information. Figure 4.9 shows the information flow that crosses the domains of the different application protocols involved in the manufacture and inspection of a part. Harmonization makes it possible to ensure the integrity of information within this digital manufacturing chain.

The data and its properties are transmitted within a bidirectional flow through the chain of manufacture. The STEP pattern environment has specifications that can be harmonized through its functional units. A strategy to harmonize begins as a study of the UoF provided in the AP219 (ISO 10303-219, 2007). The protocol focuses on providing a relevant connection to inspection programs and creating a mechanism for the exchange and feedback of information within the standard data structure for manufacturing and used by application protocols AP224, AP242, and AP238.

4.4.2 High-Level Scheme with Entity

The UoF contained in the AP219 application protocol for exchanging dimensional inspection information related to mechanical product definitions are:

- `Administrative_data`
- `Dimensional_measurement_analysis`
- `Dimensional_measurement_documentation`
- `Dimensional_measuremen_execution`
- `Dimensional_measurement_feature`
- `Dimensional_measurement_part`

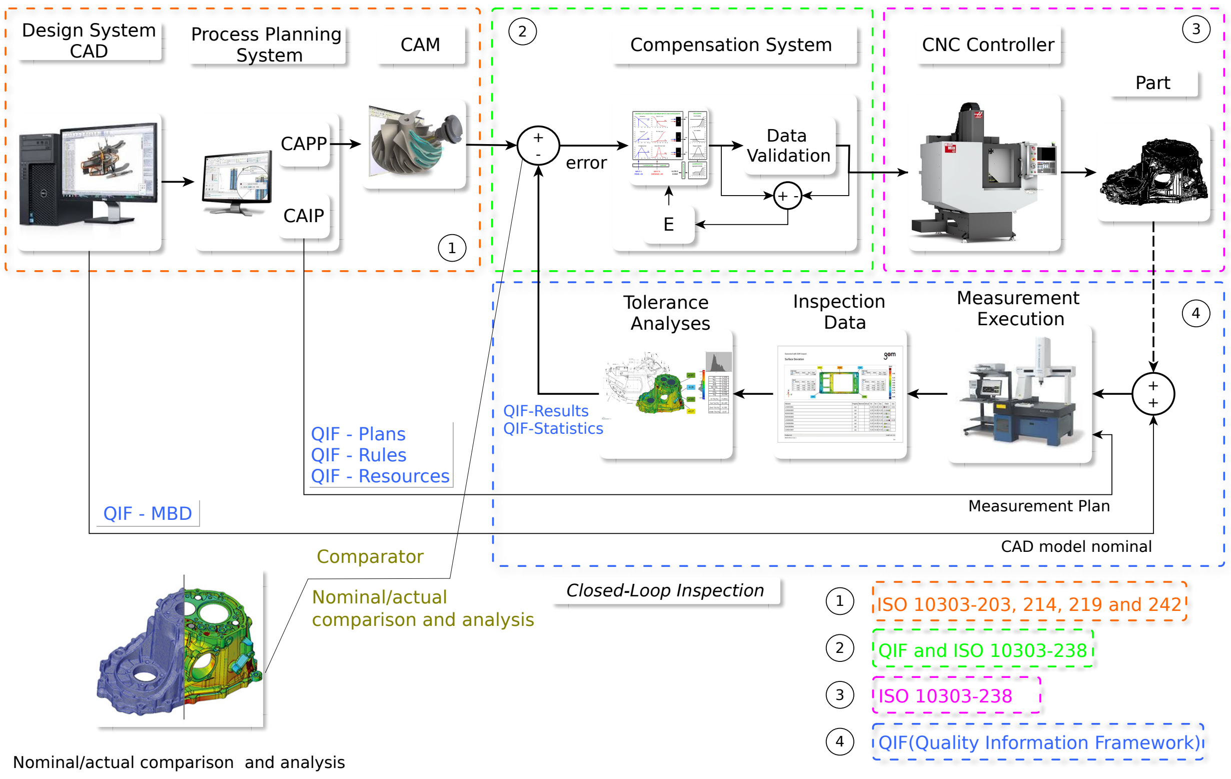


Figure 4.9: Representation of the information flow adhering to the ISO 10303 standard
Source: Adaptation (RIAÑO; ÁLVARES, 2018)

- `Dimensional_measurement_parameters`
- `Feature_definition_item`
- `Feature_profile`
- `Manufacturing_feature`
- `Functional_limitations`
- `Part_properties`
- `Program_run`
- `Shape_representation_for_machining`

The information structured in the DMIS can be mapped within the STEP standard through the AP219 protocol and the UoF that comprise it (ISO. . . , 2011; ISO 10303-219, 2007). Entities and their properties within the STEP architecture can be shared and transferred within other application protocols. With the definitions given in each UoF and based on the available EXPRESS schemes, ARM models are built that list the entities within each UoF (ISO 10303-11, 2004). Figure 4.10 shows the general relationship scheme between each UoF provided by the AP219 protocol.

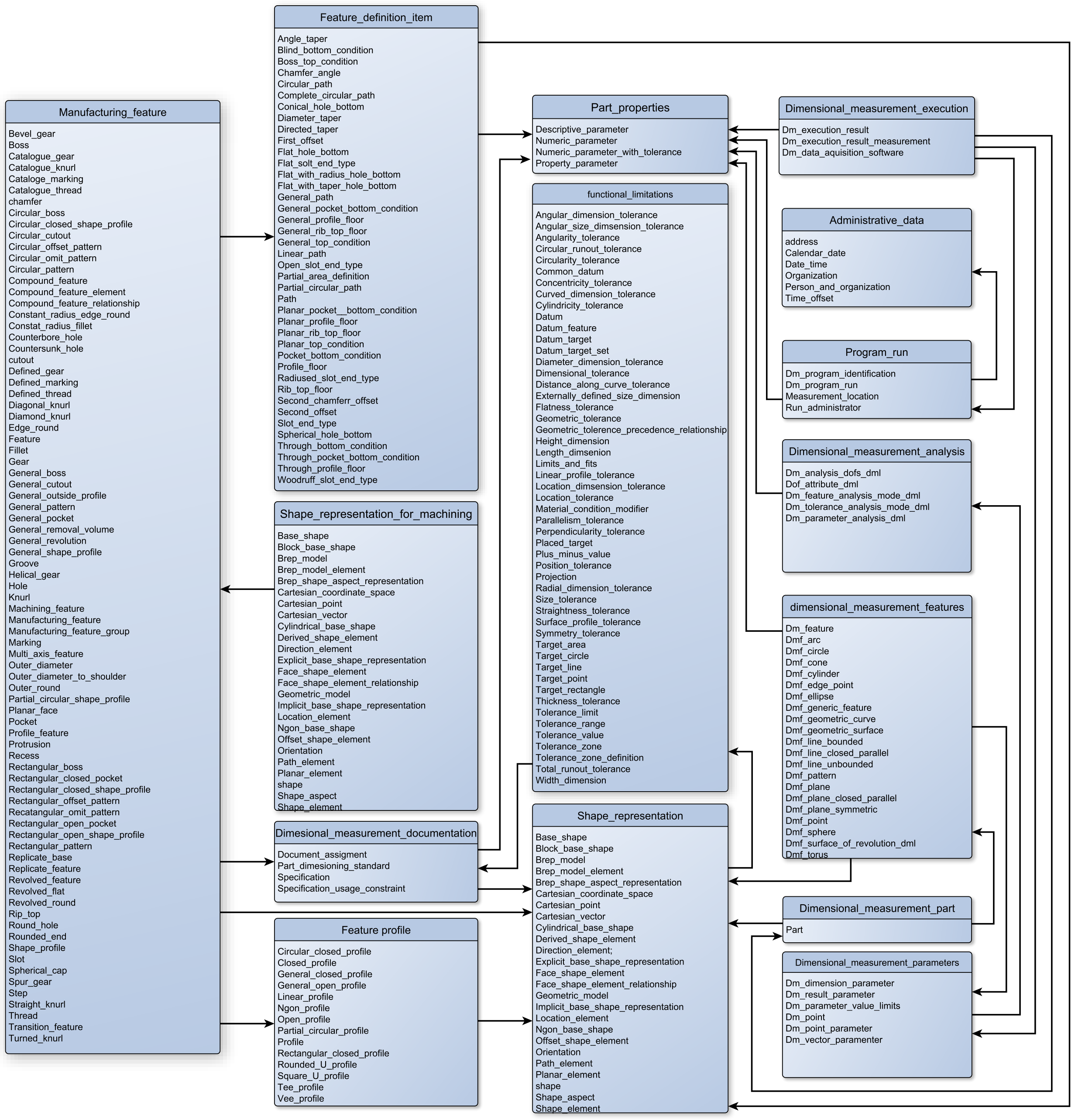


Figure 4.10: High-level relationship of the UoF provided by AP219

4.4.3 Identification of Forms for Dimensional Inspection

In the inspection process, it is necessary to define a correlation between measurement `features` and manufacturing `features`. The high-level relationship shown for AP219 in Figure 4.10 shows the exchange of information between the UoF, where product definition data, tolerances, geometry, and shape are harmonized with entities from the AP214, AP224, AP242, and protocols AP238. UoF `Dimensional_measurement_feature` associates workpiece, geometric, and topology information to build the mechanisms that identify the shapes representing the volume of material in a part that must be inspected dimensionally.The appendix presents C.3.1 the relation of entities to link data from

Shape_aspect (geometry and topology) with dm_feature_definition to identify the features of inspection. The entity dm_feature_relationship establishes a relationship between the projected parameters and those calculated with inspection results.

Shape_Aspect entity allows grouping representation items that represent aspects or components of a product's shape. The representation of a product's shapes or components represents a distinctive part of a product that can be explicitly addressed. The appendix C.3.1 presents the EXPRESS schema that describes the entity, attributes, and relationships with other entities.

An important UoF of the AP219 protocol used in the proposed integration model is called functional_limitations. The functional_limitations functionality unit contains the information necessary to identify the pieces shape relationship with the reference and shape relationship with other pieces.

The acceptable deviation value for manufacturing purposes is also defined using these entities. The appendix C.3.1.1 presents some of the entities that make up the UoF functional_limitations.In this UoF, the elements necessary to describe dimensional tolerances are integrated. The included tolerances are of two types: tolerances plus-minus and geometric.Geometric tolerances are applied to the elements defined in the schema shape_aspect shown in Figure 03 using constructors. The entity tolerance_zone allows to define within shape_aspect the value for a measure and to set the tolerance limits for geometry, orientation, location, and finishing.

Geometric tolerance is the maximum or minimum variation allowed for a position or geometric shape in the manufacturing process. The Geometric_tolerance entity is used to define and exchange critical part data for a part. The concept of tolerance feature is represented in the STEP standard as an attribute of shape_aspect to specify a type of geometric feature. For each characteristic, one of the following entities is defined:

- Angularity_tolerance
- Circular_runout_tolerance
- Circularity_tolerance
- Concentricity_tolerance
- Cylindricity_tolerance
- Flatness_tolerance
- Linear_profile_tolerance
- Parallelism_tolerance
- Perpendicularity_tolerance
- Position_tolerance
- Straightness_tolerance

The inspection run starts with the entity dm_execution_input, which contains as attributes the entities dm_program_run, part, and dm_execution_result. The dm_program_run includes information related to the environment for the execution of the inspection program. The Part entity represents the measured and linked part with the execution of the measurement program. This entity represents a shape attribute with a set of entities brep_shape_representation or shape_aspect. Each shape_aspect entity is optionally referred to a shape_element, entity, which can be replaced by any of its entity subtypes, such as direction, location, path and features. A feature can be defined as a manufacturing feature or measurement feature that is mapped by the DMIS standard. Finally, the dm_execution_result entity results from data from the execution of the measurement program. This entity refers to a set of measurement points for each parameter. The appendix C.3.2 shows the entities and attributes related to the execution of dimensional and geometric inspection.

4.4.4 Compensation Strategy

The compensation strategy proposed in this work incorporates the STEP and QIF standards data structures to create an interconnected environment between the systems involved in manufacturing and quality inspection systems. The ISO 10303: 219 (AP219) standard defines the context, scope, and information requirements necessary to analyze and report results of the dimensional inspection, creating mechanisms for bidirectional exchange of information, with the data structures used in manufacturing processes such as those provided application protocols ISO 10303-224 and ISO 10303-238.

The strategy consists of creating an interconnection between the manufacturing features, defined in the AP328 protocol with the inspection features given in the AP219 protocol and the features QIF to relate parameters such as coordinates, dimensions, positioning, tolerance, and datums. The parameters of features can be updated manually or extracted automatically from a CAD or CAM model file. The measurement is made following an inspection plan to obtain the data necessary for calculating the deviation. With the comparison of the nominal/current model, the error is calculated, and a new tolerance value is updated for each feature inspected. The new tolerances values are incorporated into the file sent to the CNC controller to compensate for the system's error. Finally, the tolerance values are updated on the STEP p21 file and sent to the CNC machine controller to execute a new machining process.

The proposed scheme to support the feedback strategy is based on the UoF provided in the application protocols ISO 10303-238 and ISO 10303-219. The STEP standard data structure supports the data flow for product definition, inspection process definition, process execution, analysis, and results reporting. The measurement process is not strictly within the scope of the ISO 10303-219 protocol, but the QIF standard allows to obtain an interconnection between dimensional inspection programs supported by ISO 22093 (DMIS 4.0), web-based analyzes, reporting of practices and standards for modeling manufacturing information such as ISO 10303-224,238,242.

The appendix C.3.1.1 shows the entities related to the definition of geometric and dimensional tolerances. It includes the mechanisms to implement GD&T features, inspect and analyze a part on dimensions, shape, and location. Tolerance information can be given to a manufacturing feature using AP238, representing a machining operation or a dimensional measurement operation given by AP219.

The appendix C.3.3 shows the EXPRESS-G schematic of the dimensional measurement features is an extraction of the entities used to integrate CNC machining and inspection, based on a relationship between the manufacturing features and inspection features. The dimensional measurement features `dm_feature` can represent a standard manufacture feature, externally composed or defined, such as cones, cylinders, ellipses, and circles.

The appendix C.3.4 shows part of the entities involved in the steps of a features machining process. The entities related to the machining and the execution of the inspection contain the entities both for relating administrative data, programs, operations, and methods of execution and the definition for using the results.

The appendix C.3.5 shows the `dimensional_measurement_representation` entities EXPRESS-G scheme part of the structure in ISO 10303-219 for dimensional measurement analysis. The entities used to define resources and provide options for analysis and calculation of tolerance parameters from measured data.

For the analysis and reporting of the results of the dimensional and geometric inspection of parts or assemblies, relationships are established between the information requirements and integration mechanisms, interpreting the integrated resources defined in the application protocols. The ISO 10303-219 protocol covers both the definition of dimensional and geometric tolerances of parts and the execution of inspection through the UoF. The UoF named `dimensional_measurement_feature` is provided in the ISO 10303-219 protocol and contains the information requirements to identify shapes representing volumes of material in part for dimensional inspection. The appendix C.3.6 shows the EXPRESS-G schema with the entities used in the UoF `Dimensional_measurement_feature`.

4.5 Application Interpreted Model (AIM)

The AIM is the final data model that meets the ARM model's requirements. The different standard constructs required in the AIM model are taken from various Generic Integrated Resources (IR) and AIC, combined to form the AIM data model's structure. AIM is the most significant data model in the STEP application protocols, as it is intended to cover the scope of the application. The AIM model represents standardized information that could be transferred and implemented.

This section presents the method for designing and implementing software components based on a conceptual description of the data. The method is based on the ISO10303 standard; work carried out over the last years by ISO to standardize the computer-interpretable representation and exchange of product manufacturing information. This standard has the technology and a method for the specification, verification, and implementation of conceptual data models. STEP technology was used, mainly the description method, the implementation methods, and the standard data model definition method for specification and coding.

The proposed method is based on a description of the data used and manipulated by code to build a manufacturing and inspection data integration strategy that can operate in a closed-loop environment. The codes created are intended to be used within specific projects with this same focus and can be scaled both functionally and technologically. It is partially configured to reflect the proposed system's particular need, creating an application within the described data domain. Here are some identified benefits of following this method:

- By following a well-defined structure, more excellent reliability is obtained. Errors are reduced by the benefits derived from know-how.
- It enables automatic code generation, making the process for generic implementations efficient. A large amount of code can be produced directly from high-level descriptions.
- The result reflects homogeneity of implementation, maintaining a clean coding without corruption due to bad programming practices.
- The code can be easily updated, being current to include new technological achievements. Maintenance and update tasks become easier within this structure.

In coding, a programming tool is used that translates the high-level specifications of the abstraction into a particular solution. The specifications describe the problem and actions to be executed by the program; they are used in different ways: text, graphic representations, or even events. The embodiment is conventionally a software component produced in a programming language. Code generation requires analysis of a high-level specification; the analysis produces an intermediate representation used in the elaboration process, either automatic or interactive; the intermediate representation is then translated into a software component. Therefore, the specification of a system is clearly distinguished from its implementation. Before implementing automatic code generation, the following considerations should be considered:

- The implementation of an automatic code generator requires resources and time; it demands a great effort in the design and development stage. The benefit is obtained when the generation code is automatically used as a development tool.
- The proposed solution is a specific case or a general case. The components generated through automatic code generation can be in greater quantity than is necessary. It is possible to systematically implement functions that are not used within the given particular case study. Performance in terms of response may suffer.
- Automatic code generation may be justified for implementations that require developments with similar characteristics, also when the project will suffer frequent alterations in its life cycle, or if several prototypes are required to obtain a final development.

It is observed that for specific solutions such as those addressed in this project, the automatic code generation may be oversized. It may be challenging to integrate already developed components and new base components to integrate manufacturing and inspection data. Using automatic code generators to reduce build time may be overlooked in this approach and require additional coding.

4.5.1 Mapping table

The mapping tables correlate the ARM reference model with builders available in the Integrated Generic Resources, Integrated Application Resources, and Application Interpreted Constructs. The complete requirements specified in the ARM model defined in the EXPRESS language are converted into manageable constructions, which will be identified in the integrated resources. The table formally establishes the origin of the components of an AIM model. Each entity and attribute found in the AIM model is specified in the correspondence table that relates the ARM model to AIM. This table monitors the process (ISO 10303-238, 2007).

Each UoF contained in the application protocols needs to be mapped to one or more AIM constructs. A summary of the UoF contained in ISO 10303-238 and IS010303-219 was given in section 3.9. The mapping tables show how IR and AIC are used to meet each application protocol's information requirements. Figure 4.11 shows the extraction of a mapping table for the Project functional unit (see 3.9.2) in the AP238 application protocol. The mapping table shows how each AP UoF and application object maps to one or more AIM constructs. The table is organized into five columns named Application element, AIM element, Rules, and Reference path, which are explained below (ISO 10303-238, 2007).

Application element	AIM element	Source	Rules	Reference path
PERSON_AND_ADDRESS	person	10303-41		
its_person	IDENTICAL MAPPING	10303-41		
its_address	personal_address	10303-41		person <- personal_address.people[i] personal_address

PROJECT	product_definition_formation	10303-41		product_definition_formation
				{ product_definition_formation.of_product ->
				product =>
				machining_project }

its_id	product.id	10303-41		product_definition_formation product_definition_formation.of_product -> product product.id
project to workplan (as main_workplan)	PATH			product_definition_formation <- product_definition.formation product_definition characterized_product_definition = product_definition characterized_product_definition <- process_product_association.defined_product process_product_association process_product_association.process -> product_definition_process => action { action.name = ‘machining’ } action.chosen_method -> action_method => machining_process_executable => machining_workplan

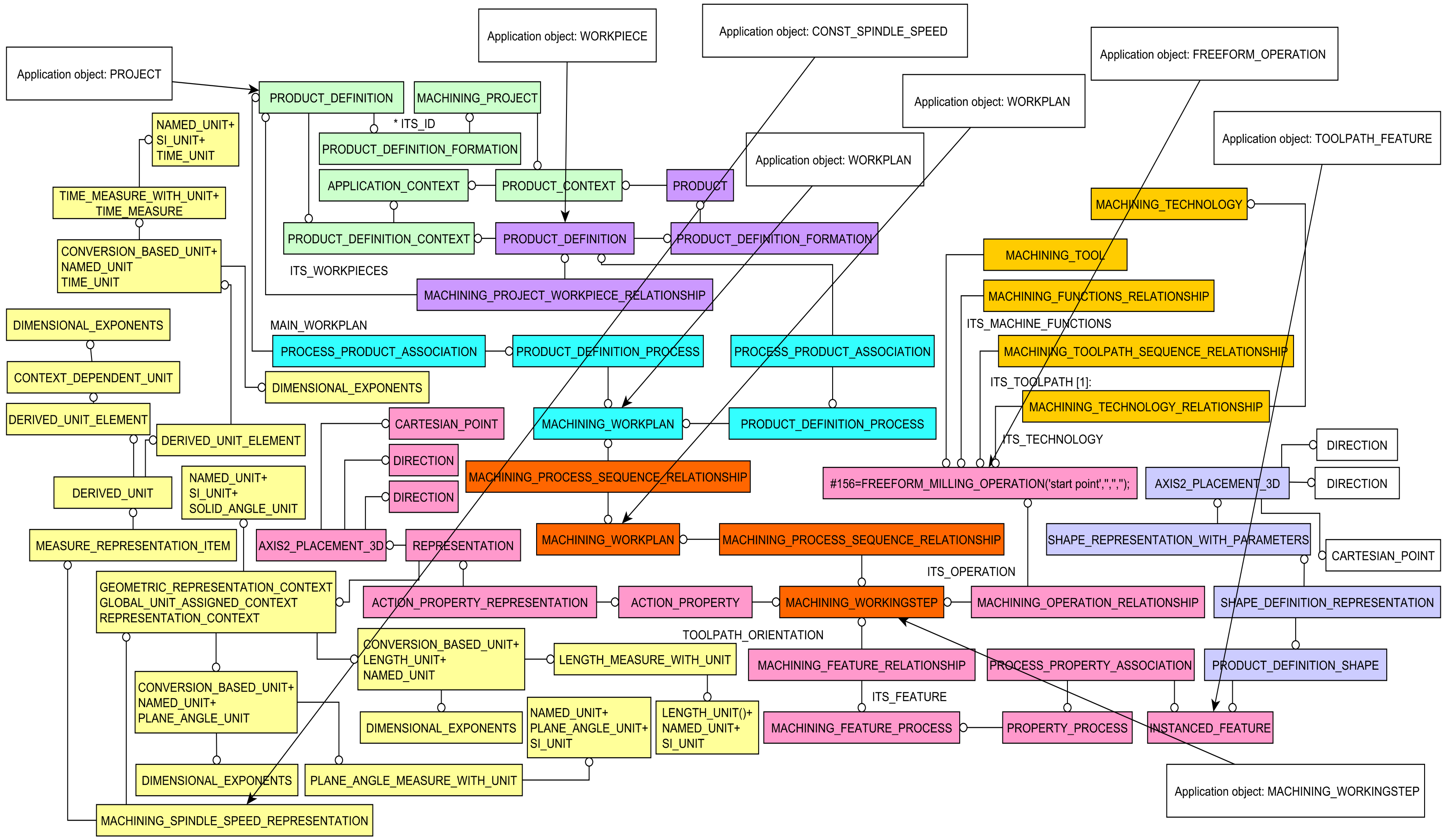
Figure 4.11: Mapping table for project UoF.
Source:(ISO 10303-238, 2007)

- **Application element**: It specifies the name of the application element corresponding to the AP's definition.
- **AIM element**: Name of an AIM element. Mapping an AIM element can result in multiple related AIM elements. Each of these AIM elements requires its line in the table.
- **Source**: This is the number that corresponds to the part that specifies the AIM element's source.
- **Rules**: One or more numbers can be given that refers to the rules that apply to the current AIM element or reference path.
- **Reference path**: Documents the function of an AIM element concerning the AIM element in the row that follows. Two or more of these related AIM elements define the interpretation of the built-in resources that satisfy the application object's requirement. For each AIM element created, a reference path to its supertype is specified from a built-in resource. This column fully describes an application object's mapping; it may be necessary to specify a reference path through various related AIM elements. The following notational conventions (see 4.1) apply to interpret reference path expressions and relationships between AIM elements.

Table 4.1: Notational conventions

Symbol	Definition
[]	Enclosed section constrains multiple AIM elements or sections of the reference path are required to satisfy an information requirement.
()	Enclosed section constrains multiple AIM elements or sections of the reference path are identified as alternatives within the mapping to satisfy an information requirement.
	Enclosed section constrains the reference path to satisfy an information requirement.
<>	Enclosed section constrains at one or more required reference path.
||	Enclosed section constrains the supertype entity.
$->$	Attribute references the entity or select type given in the following row.
$<-$	Entity or select type is referenced by the attribute in the following row.
$[i]$	Attribute is an aggregation of which a single member is given in the following row.
$[n]$	Attribute is an aggregation of which member n is given in the following row.
=>	Entity is a supertype of the entity given in the following row.
<=	Entity is a subtype of the entity given in the following row.
=	The string, select, or enumeration type is constrained to a choice or value.
$\backslash$	The reference path expression continues on the next line.
$*$	Used in conjunction with braces to indicate that any number of relationship entity data types may be assembled in a relationship tree structure.

Figure 4.12 presents an abstraction with the related entities through application objects linked to the UoF of the AP238 protocol. The data structure of some of the AP238 application protocol entities used in the development of the implementation can be seen in Figure 4.12. This structure guides creating a project within the programming environment and is used to create classes for each related instance in the schema.



Application object: PROJECT
Application object: WORKPIECE
Application object: CONST_SPINDLE_SPEED
Application object: FREEFORM_OPERATION
Application object: WORKPLAN
Application object: WORKPLAN
Application object: TOOLPATH_FEATURE
Application object: MACHINING_WORKINGSTEP
PRODUCT_DEFINITION
MACHINING_PROJECT
* ITS_ID
PRODUCT_DEFINITION_FORMATION
APPLICATION_CONTEXT
PRODUCT_CONTEXT
PRODUCT
PRODUCT_DEFINITION_CONTEXT
PRODUCT_DEFINITION
PRODUCT_DEFINITION_FORMATION
ITS_WORKPIECES
MACHINING_PROJECT_WORKPIECE_RELATIONSHIP
NAMED_UNIT+ SI_UNIT+ TIME_UNIT
TIME_MEASURE_WITH_UNIT+ TIME_MEASURE
CONVERSION_BASED_UNIT+ NAMED_UNIT TIME_UNIT
DIMENSIONAL_EXPONENTS
CONTEXT_DEPENDENT_UNIT
DERIVED_UNIT_ELEMENT
DERIVED_UNIT_ELEMENT
DERIVED_UNIT
NAMED_UNIT+ SI_UNIT+ SOLID_ANGLE_UNIT
MEASURE_REPRESENTATION_ITEM
MAIN_WORKPLAN
PROCESS_PRODUCT_ASSOCIATION
PRODUCT_DEFINITION_PROCESS
PROCESS_PRODUCT_ASSOCIATION
DIMENSIONAL_EXPONENTS
CARTESIAN_POINT
DIRECTION
DIRECTION
MACHINING_WORKPLAN
PRODUCT_DEFINITION_PROCESS
MACHINING_PROCESS_SEQUENCE_RELATIONSHIP
AXIS2_PLACEMENT_3D
REPRESENTATION
MACHINING_WORKPLAN
MACHINING_PROCESS_SEQUENCE_RELATIONSHIP
GEOMETRIC_REPRESENTATION_CONTEXT GLOBAL_UNIT_ASSIGNED_CONTEXT REPRESENTATION_CONTEXT
ACTION_PROPERTY_REPRESENTATION
ACTION_PROPERTY
MACHINING_WORKINGSTEP
MACHINING_OPERATION_RELATIONSHIP
SHAPE_DEFINITION_REPRESENTATION
ITS_OPERATION
TOOLPATH_ORIENTATION
CONVERSION_BASED_UNIT+ LENGTH_UNIT+ NAMED_UNIT
LENGTH_MEASURE_WITH_UNIT
MACHINING_FEATURE_RELATIONSHIP
PROCESS_PROPERTY_ASSOCIATION
PRODUCT_DEFINITION_SHAPE
CONVERSION_BASED_UNIT+ NAMED_UNIT+ PLANE_ANGLE_UNIT
DIMENSIONAL_EXPONENTS
NAMED_UNIT+ PLANE_ANGLE_UNIT+ SI_UNIT
LENGTH_UNIT()+ NAMED_UNIT+ SI_UNIT
ITS_FEATURE
MACHINING_FEATURE_PROCESS
PROPERTY_PROCESS
INSTANCED_FEATURE
DIMENSIONAL_EXPONENTS
PLANE_ANGLE_MEASURE_WITH_UNIT
MACHINING_SPINDLE_SPEED_REPRESENTATION
MACHINING_TECHNOLOGY
MACHINING_TOOL
MACHINING_FUNCTIONS_RELATIONSHIP
ITS_MACHINE_FUNCTIONS
MACHINING_TOOLPATH_SEQUENCE_RELATIONSHIP
ITS_TOOLPATH [1]:
MACHINING_TECHNOLOGY_RELATIONSHIP
ITS_TECHNOLOGY
#156=FREEFORM_MILLING_OPERATION('start point',",",");
AXIS2_PLACEMENT_3D
DIRECTION
DIRECTION
SHAPE_REPRESENTATION_WITH_PARAMETERS
CARTESIAN_POINT

Figure 4.12: Structure of project UoF.

4.6 Summary

After knowing the standard and understanding how the information is structured, we develop an information modeling that meets the requirements to establish a closed manufacturing and inspection loop. Two fundamental approaches stand out: the functional and descriptive approach, which made it possible to define the models useful in the implementation phase. The functional approach within a modeling environment shows the broad contexts defined by the ISO10303 standard. Understand contexts to proceed with defining roles, activities, and data flow. The information linked to each activity in the functional model is associated with a functional group of transformed or produced data within each functional block to execute an activity within the application context. The IDEF0 graphical language is the formal mechanism for presenting the functional blocks that describe an application context in a particular solution.

Representations give the conceptual approach in data schemas, whose meaning needs to be related to an application context to make sense within the model. It is a process of abstraction to give meaning to the data and understand its relationship. STEP uses schemas to define each of its application protocols; schemas contain conceptual models made up of concept relationships. Strict enforcement of this mapping is essential to ensure the integrity of the STEP interchange files information.

Chapter 5

The STEP Development and Deployment Environment

5.1 Introduction

This chapter presents the methodology to use functional and descriptive models within a programming environment that leads to generating a computational solution compatible with the STEP standard. The methodology incorporates IR, AIC, and application protocols to minimize cohesion problems and promote intelligibility, maintainability, and homogeneity in the data flow. Three typical case studies are used together with the computational solution obtained to verify the integration and interoperability of the processes.

5.2 Implementation of Case Study 1

This section presents a case study with an approach applied to a computational implementation design and construction. The main objective is to include or modify the entities with specific information from both manufacturing and inspection within the neutral STEP-data structure. Two approaches are shown, the first when we already have a conformed file and the second describes a complete description of the implementation to create new solutions with specific functionalities through the interpretation of the STEP standard. The exchange data structure has a complex structure within an object-oriented perspective, maintaining cohesion and integrity in its information. Including new information such as GD&T specifications, tool paths, tool selection, among many possibilities, requires knowledge and interpretation of the STEP standard. This solution is presented as a methodology to follow to encode a STEP interchange file that manages to solve interoperability and systems integration problems that allow adding new functions such as those presented in each phase of the life cycle.

The developed system involves coding various application objects derived from the UoFs in the application protocols, such as AP219, AP238, and AP242. As presented in the section (see 4.5), the AP238 protocol, also known as STEP-NC, in its content, the standard presents the AIM model for implementing the ARM model. The ARM model is used to describe entities, and by analyzing this model, it is possible to understand each entity's role, attributes, and relationships with other entities. Two levels are clearly differentiated through these models; the ARM model documents the information requirements present entities and attributes that can be instantiated, defines its hierarchy showing the relationship of entities, subtypes, and supertypes, and how the entities are interrelated through its attributes, as is the case with complex entities. The AIM model presents the accurate data, how the instantiation should occur according to the UoF in the protocols, and solves functionality with application objects built with entities described in the ARM model. In section 4.1 of the AP238 standard, it describes the UoF (AP238) of ARM models. UoF are a set of information requirements described as application objects. The application objects that describe the machining process are referenced in ISO14649; for more specific things like GD&T, they are defined in section 4.2 of the ISO10303-238 application protocol. The features are defined in the two standards; session 4.2 of AP238 has extra information for harmonization purposes. The AIM is derived from the mapping tables, which show the Reference Paths showing the ARM model concepts represented by AIM groups representing implementations. The EXPRESS schemas for the AIM model provide the structures used by the mapping tables.

5.2.1 STEP Standard Interpretation

Coding begins by analyzing the structure of a STEP interchange file; according to the standard, a higher-level entity is required to structure the Project. The AP238 contains the UoF called Project; this UoF specifies where this UoF specifies where the machining program's interpretation starts and management information about the machining program. The `PROJECT` and `Person_and_adress` application objects are used by the UoF Project and are defined by clause 4.3 of ISO14649-10: 2004. Each part program or data model based on ISO 14649 must include this application object that indicates the work plan to be executed after the interpretation of this model (It can include several work plans) and provide the workpieces on which actions must be performed. Section 4.2.269 of document AP238 or in clause 4.3 of ISO 14649-10: 2004 provides information about the application object `Project` from which the EXPRESS description shown in Figure 5.1 of the Project entity is extracted.

```
ENTITY project; (* m0 *)
      its_id: identifier;
      main_workplan: workplan;
      its_workpieces: SET [0:?] OF workpiece;
      its_owner: OPTIONAL person_and_address;
      its_release: OPTIONAL date_and_time;
      its_status: OPTIONAL approval;
      (* Informal proposition:
            its_id shall be unique within the part programme.
      *)
END_ENTITY;
```

Figure 5.1: Entity Description EXPRESS `Project` Entity.
Source:(ISO 10303-238, 2007)

The list of attributes of this entity is explained below:

- its_id: Project identifier, It will be unique within the parts program.
- main_workplan: The top-level workplan in this model.
- its_workpieces: The workpieces upon which actions are to be performed.
- its_owner: Optional information on the owner of the Project.
- its_release: Optional date and time reference of the Project.
- its_status: Optional attribute to indicate the current status of the Project.

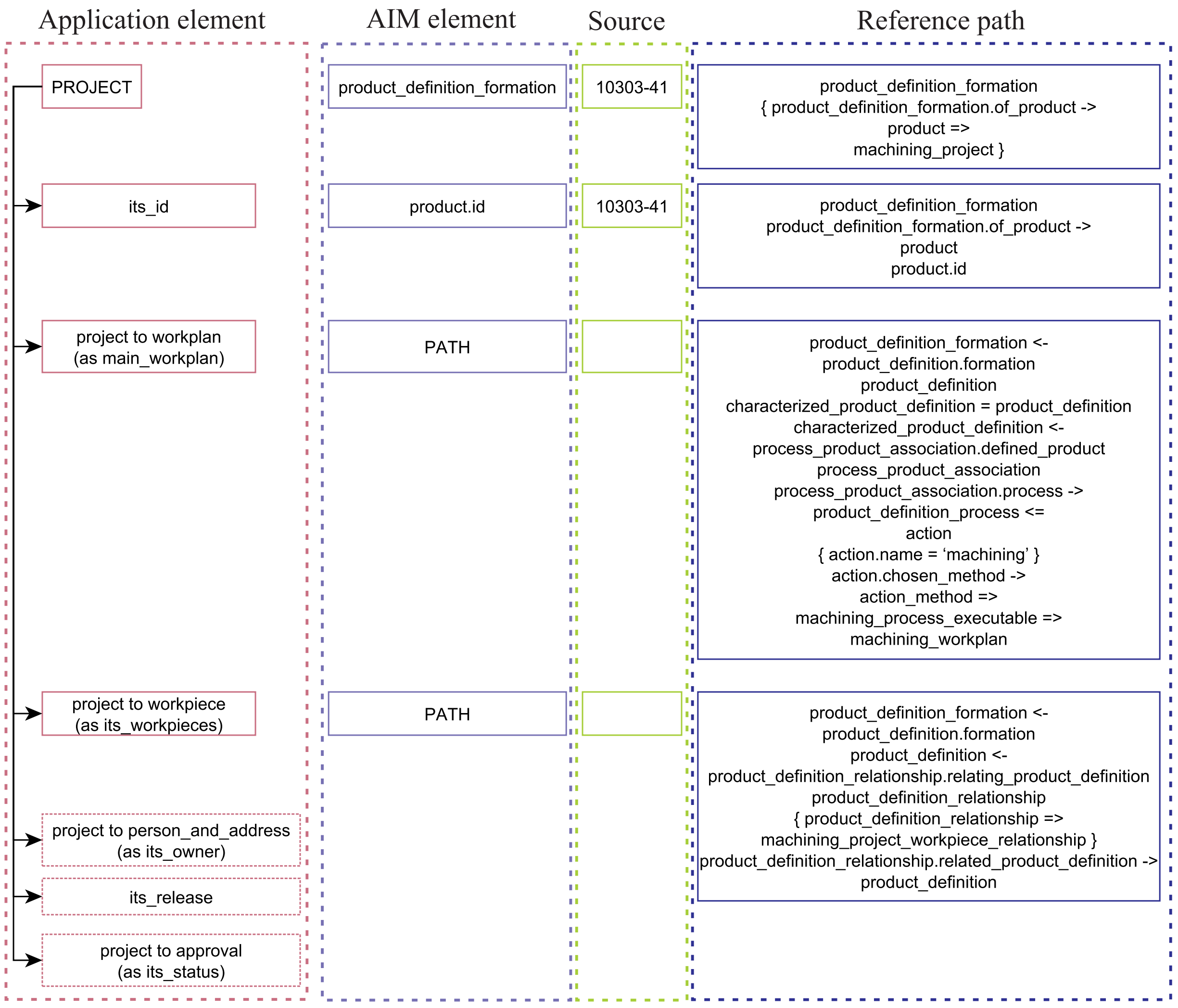
The following section to be consulted is section 4.5.1, "Application interpreted model" of the standard document AP238. This section shows how each UoF and application Object is mapped to one or more AIM constructs. In the section, it was explained how this table is organized. The information extracted from table 2- Mapping table for Project UoF of document ISO-10303-238 is illustrated in Figure 5.2.

The Figure 5.2 and the explanation given in section 4.5.1 show that the application objects names are shown in uppercase (e.g., PROJECT). Attribute names and assertions are listed after the application object they belong to and are written in lowercase (e.g., its_id, project to workplan (as main_workplan), project to workpiece (as its_workpieces), project to person_and_address (as its_owner), its_release, and project to approval (as its_status)), in the column "AIM element" shows the AIM construct element (e.g., product_definition_formation). The term 'PATH' indicates that the application assertion maps to the entire reference path. In the source column, it shows the number of the corresponding part of ISO 10303. The reference path column fully describes an application object's mapping; it may be necessary to specify a reference path through several related AIM elements.

Before starting Java programming, a schema is created with the assertions resulting from interpreting the mapping table. Figure 5.3 shows a hierarchical diagram with the entities related to the AIM constructor elements of the UoF Project. Next to the diagram is a clipping of the STEP interchange file that shows instantiating these entities and how they are freely distributed in the DATA section of the p21 file. The schema derived in Figure 5.3 describes the path obtained from the mapping table and shows how to follow to develop the project application object's computational implementation. There are several UoFs that need to be implemented to complete the objective of the case study. The process followed in interpreting the standard and obtaining a hierarchical structure that indicates the programming path is the same in most cases. In many situations, the mapping table refers to a generalized entity or a supertype entity in an attribute. In the implementation, a specialized entity is instantiated, a subtype that inherits those attributes and defines additional ones for its specialty.

5.2.2 Deployment Architecture

The functional and conceptual models were presented, a clear strategy was also identified to use the AIM models and map from logical definitions to functional units and application objects. In development follow the activities of building the physical environment. In addition to the appropriate computational resources, Java, JSDAI, and APIS programming environments running on the Eclipse integrated development environment (IDE), an implementation architecture presented in Figure is proposed. The deployment



Application element	AIM element	Source	Reference path

PROJECT	product_definition_formation	10303-41	product_definition_formation
			{ product_definition_formation.of_product ->
			product =>
			machining_project }

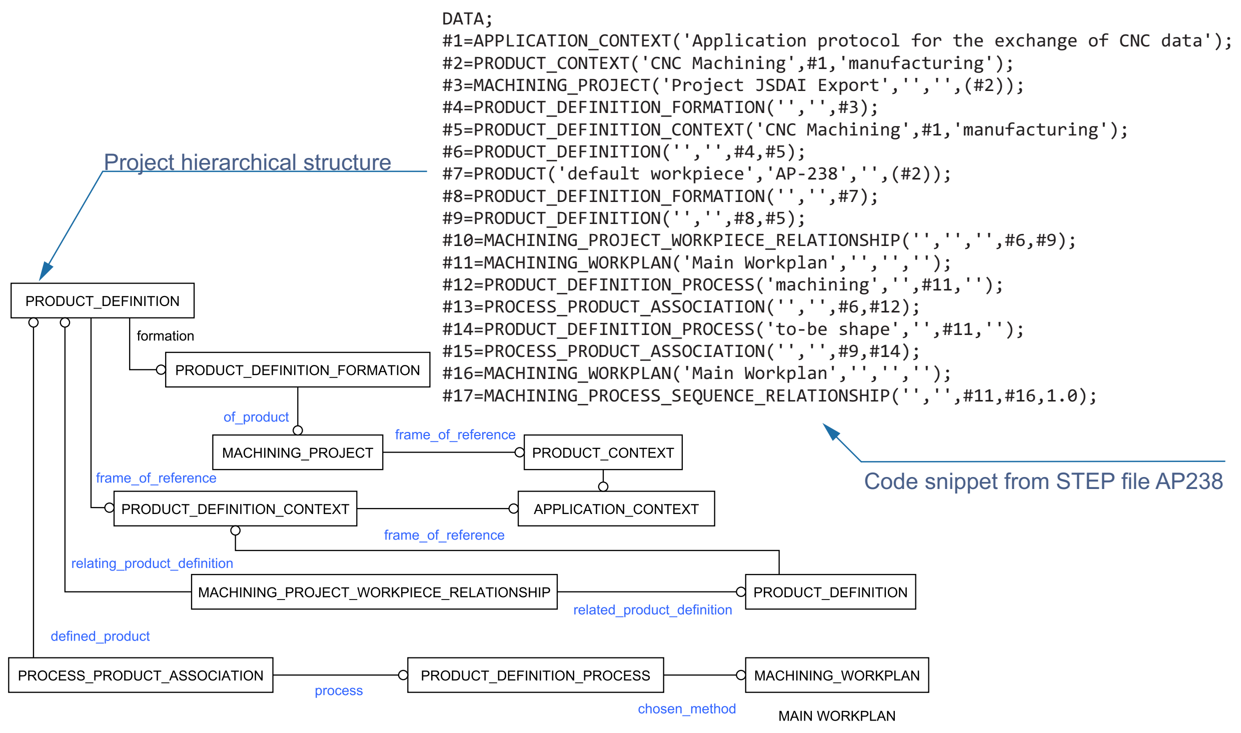
its_id	product.id	10303-41	product_definition_formation product_definition_formation.of_product -> product product.id
project to workplan (as main_workplan)	PATH		product_definition_formation <- product_definition.formation product_definition characterized_product_definition = product_definition characterized_product_definition <- process_product_association.defined_product process_product_association process_product_association.process -> product_definition_process <= action { action.name = ‘machining’ } action.chosen_method -> action_method => machining_process_executable => machining_workplan
project to workpiece (as its_workpieces)	PATH		product_definition_formation <- product_definition.formation product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship { product_definition_relationship => machining_project_workpiece_relationship } product_definition_relationship.related_product_definition -> product_definition
project to person_and_address (as its_owner)			
its_release			
project to approval (as its_status)			

Note: The information was extracted from Table 2 - Mapping table for Project UoF of document ISO-10303-238.

Figure 5.2: Mapping table for Project UoF.
Source: Adaptation (ISO 10303-238, 2007)

architecture sizes the deployment to define the programming tasks necessary to meet the case study requirements. Many of the definitions of entities given in the conceptual model need to be integrated and reused by different functional units, requiring optimal resource use. The architecture seeks to have a standard development approach, with self-describing language and facilitate maintenance, updating, or technological improvement tasks, capable of expanding and receiving functional increases with new developments.

Each specification detailed in the functional model, the descriptive definition in the conceptual model translated for application objects in the AIM model, is used in this implementation phase to build the project. Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is designed to handle specific development aspects of an application. The model contains the dynamic data structure of the application, independent of the user interface. The model component directly manages the data, logic, and rules of the application. The component is the core where the knowledge of the STEP standard



```
DATA;
#1=APPLICATION_CONTEXT('Application protocol for the exchange of CNC data');
#2=PRODUCT_CONTEXT('CNC Machining',#1,'manufacturing');
#3=MACHINING_PROJECT('Project JSDAI Export','','',(#2));
#4=PRODUCT_DEFINITION_FORMATION('','',#3);
#5=PRODUCT_DEFINITION_CONTEXT('CNC Machining',#1,'manufacturing');
#6=PRODUCT_DEFINITION('','',#4,#5);
#7=PRODUCT('default workpiece','AP-238','',(#2));
#8=PRODUCT_DEFINITION_FORMATION('','',#7);
#9=PRODUCT_DEFINITION('','',#8,#5);
#10=MACHINING_PROJECT_WORKPIECE_RELATIONSHIP('','','',#6,#9);
#11=MACHINING_WORKPLAN('Main Workplan','','','');
#12=PRODUCT_DEFINITION_PROCESS('machining','',#11,'');
#13=PROCESS_PRODUCT_ASSOCIATION('','',#6,#12);
#14=PRODUCT_DEFINITION_PROCESS('to-be shape','',#11,'');
#15=PROCESS_PRODUCT_ASSOCIATION('','',#9,#14);
#16=MACHINING_WORKPLAN('Main Workplan','','','');
#17=MACHINING_PROCESS_SEQUENCE_RELATIONSHIP('','',#11,#16,1.0);
```

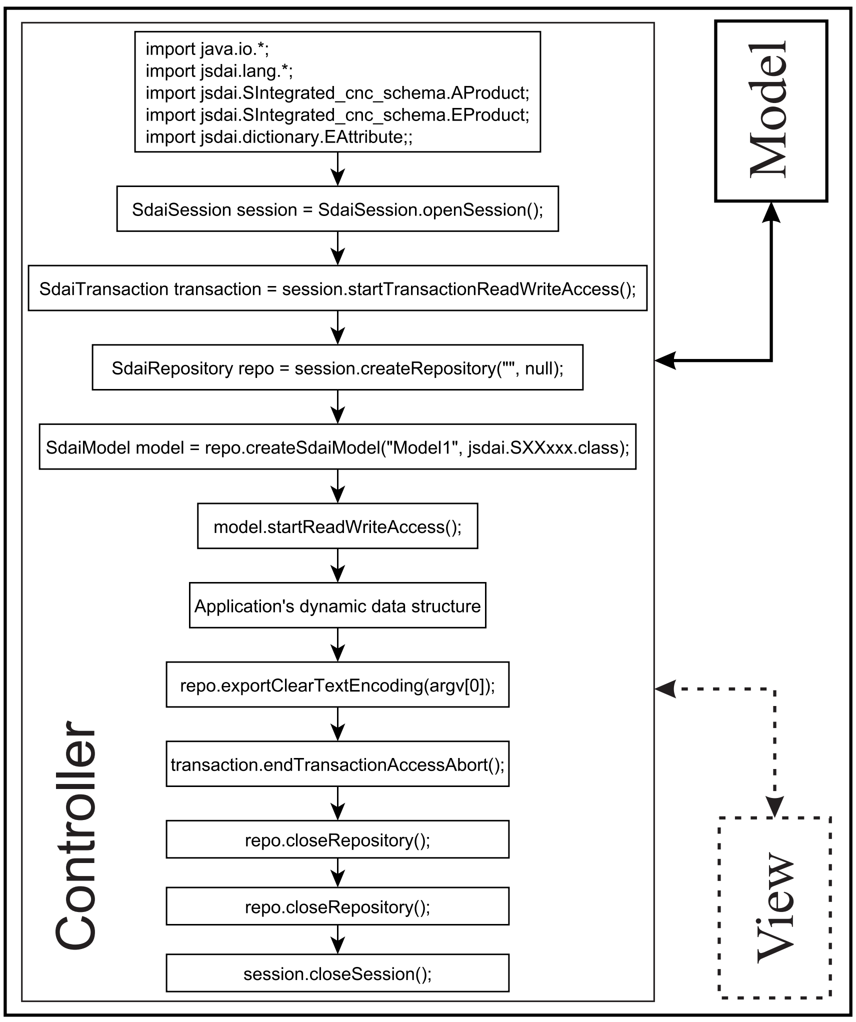
Project hierarchical structure
Code snippet from STEP file AP238
PRODUCT_DEFINITION
formation
PRODUCT_DEFINITION_FORMATION
of_product
frame_of_reference
MACHINING_PROJECT
PRODUCT_CONTEXT
frame_of_reference
PRODUCT_DEFINITION_CONTEXT
APPLICATION_CONTEXT
frame_of_reference
relating_product_definition
MACHINING_PROJECT_WORKPIECE_RELATIONSHIP
PRODUCT_DEFINITION
related_product_definition
defined_product
PROCESS_PRODUCT_ASSOCIATION
process
PRODUCT_DEFINITION_PROCESS
MACHINING_WORKPLAN
chosen_method
MAIN WORKPLAN

Figure 5.3: hierarchical diagram for UoF Project.

necessary to develop the application rests. Incremental improvements in the model component can be included and integrated to modify the scope of the project. The controller manages the program's execution, creates the project, receives events, and implements the actions to supply data to the model. The controller takes all changes to the data and manages the creation of the STEP interchange file. In Figure 5.4, a sequential structure of the controller component's main file is presented, aiming to create an SDAI session, manage the repository, and encode the STEP exchange file with the information generated in the execution of the program.

Figure 5.5 shows a diagram of the class relationships linked to the Model component of the architecture. The diagram located in the upper right corner in Figure5.5 is a guide to understanding the distribution and relationship of classes that is created when implementing a functional unit. In some cases, UoF may require more than one application object, and application objects are made up of sets of entities related to each other; some application objects use instances created in other application objects as attributes, evidencing the object-oriented perspective adopted by the STEP ISO10303 standard. Figure 5.5 details only a part of the Model component of the architecture to show the model's relationship and data structure. The entities make up the most atomic or independent element of the model and must be the first to be encoded. The entities make up a library of entities that can be used in the application objects programming or any future implementation. For each entity used, a Java class is created that allows its attributes to be instantiated, read, written, or updated. Figure 5.6 shows the Java class created for the product entity, defined in the EXPRESS schema of ISO10303-41. This same code structure is used to create the other classes that make up the project entity library.

An EXPRESS entity is represented by its own Java `EnameEntity` interface, a `CnameEntity` im-



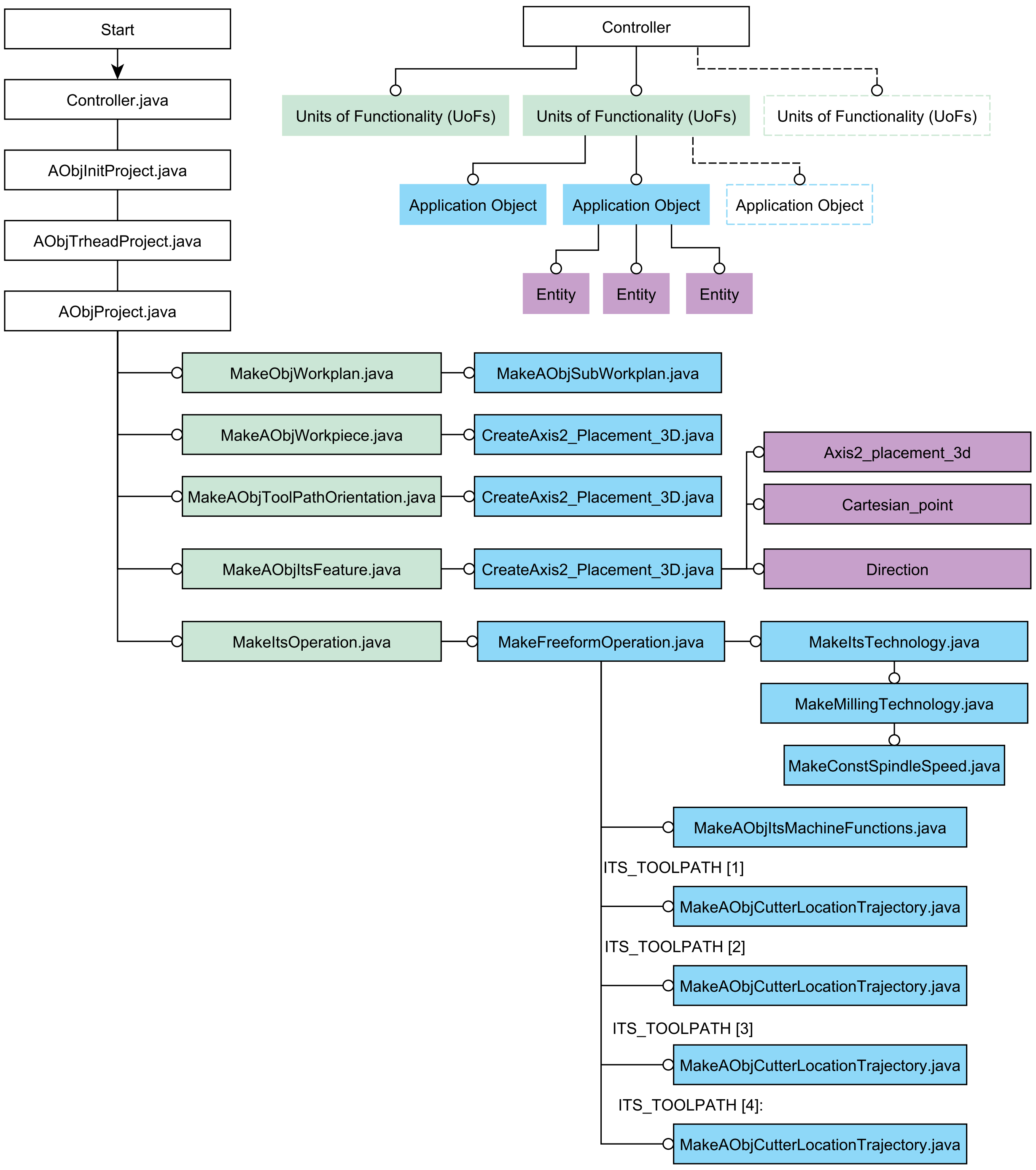
```
import java.io.*;
import jsdai.lang.*;
import jsdai.SIntegrated_cnc_schema.AProduct;
import jsdai.SIntegrated_cnc_schema.EProduct;
import jsdai.dictionary.EAttribute;;
SdaiSession session = SdaiSession.openSession();
SdaiTransaction transaction = session.startTransactionReadWriteAccess();
SdaiRepository repo = session.createRepository("", null);
SdaiModel model = repo.createSdaiModel("Model1", jsdai.SXXxxx.class);
model.startReadWriteAccess();
Application's dynamic data structure
repo.exportClearTextEncoding(argv[0]);
transaction.endTransactionAccessAbort();
repo.closeRepository();
repo.closeRepository();
session.closeSession();
```

Figure 5.4: Deployment Architecture

plementation Java class, and an `AnameEntity` aggregate Java class in early binding access. To access an attribute is done through specialized methods test, get, set, create, unset. For each attribute, whether explicit, derived, or inverse, a get method is available, and for explicit attributes, the unset and set modification methods are available. The create method is used only for attributes of aggregate type.

Four classes were needed in the JSDAI package "jsdai.lang" to manipulate entities within the Java environment and build the STEP swap files: `SdaiSession`, `SdaiRepository`, `SdaiModel`, and `EntityExtent`. The `SdaiSession` class is necessary to create, start, and end any JSDAI session; instances, relationships, and data transfer between entities happen during a session. The `SdaiSession` class is also used to initialize and end any SDAI activities, complete data transfer transactions between the program and a STEP Part 21 file and dynamically create new repositories. A `SdaiSession` class object creates a new repository for a STEP Part 21 file and establishes a transfer medium between the repository and encoding of a STEP Part 21 swap file.

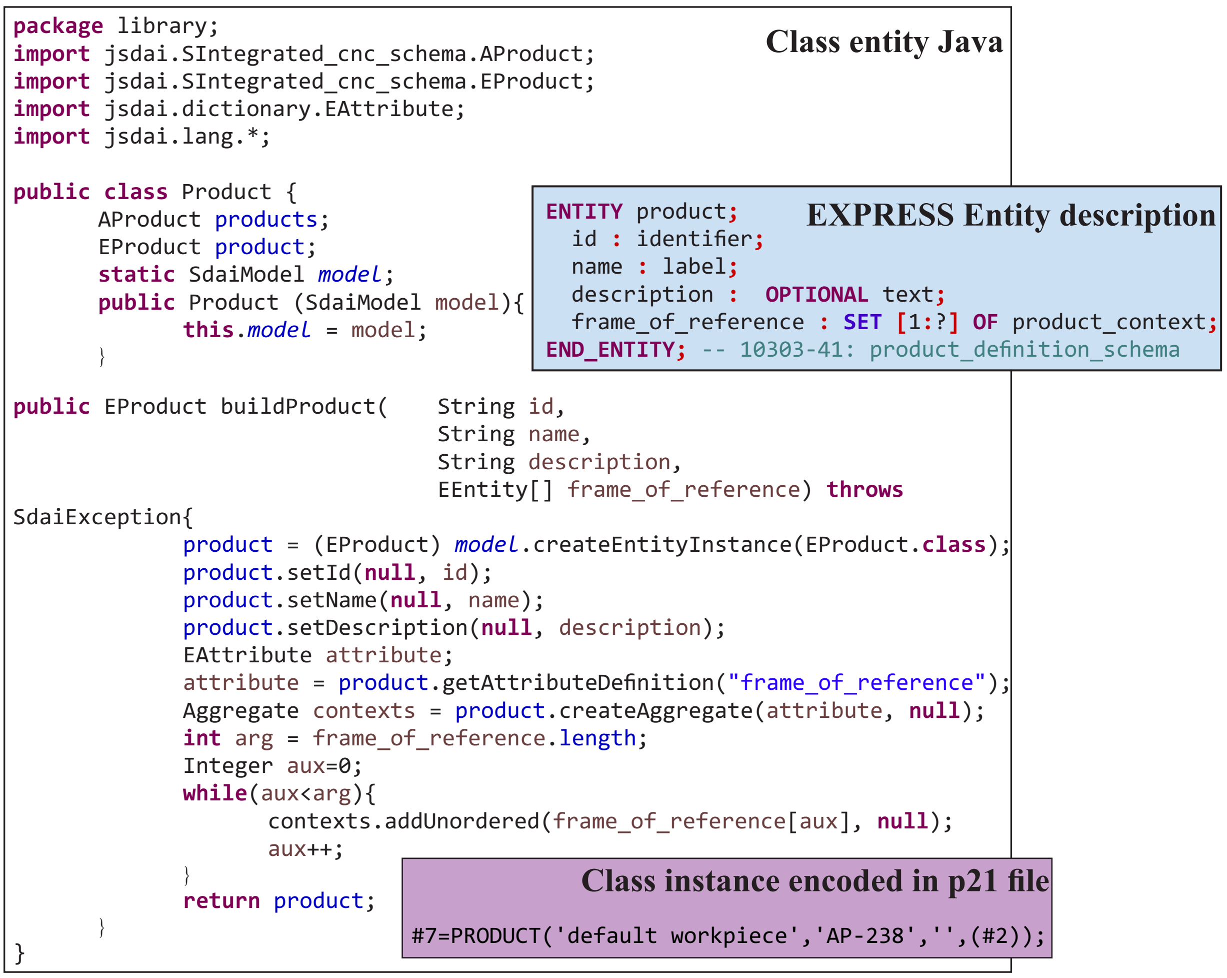
It is possible to manipulate files based on EXPRESS, build new models enriched with other schemas, consolidate a particular schema that supports the application's development. The JSDAI EXPRESS com-



Start
Controller.java
AObjInitProject.java
AObjTrheadProject.java
AObjProject.java
Controller
Units of Functionality (UoFs)
Units of Functionality (UoFs)
Units of Functionality (UoFs)
Application Object
Application Object
Application Object
Entity
Entity
Entity
MakeObjWorkplan.java
MakeAObjSubWorkplan.java
MakeAObjWorkpiece.java
CreateAxis2_Placement_3D.java
MakeAObjToolPathOrientation.java
CreateAxis2_Placement_3D.java
MakeAObjItsFeature.java
CreateAxis2_Placement_3D.java
Axis2_placement_3d
Cartesian_point
Direction
MakeItsOperation.java
MakeFreeformOperation.java
MakeItsTechnology.java
MakeMillingTechnology.java
MakeConstSpindleSpeed.java
MakeAObjItsMachineFunctions.java
ITS_TOOLPATH [1]
MakeAObjCutterLocationTrajectory.java
ITS_TOOLPATH [2]
MakeAObjCutterLocationTrajectory.java
ITS_TOOLPATH [3]
MakeAObjCutterLocationTrajectory.java
ITS_TOOLPATH [4]:
MakeAObjCutterLocationTrajectory.java

Figure 5.5: Class relationship structure on the layer Model - MVC architecture.

piler is used for the manipulation of EXPRESS schemas. Manipulating and generating other schemas can help support the project with entities from other schemas that increase functionalities that are not included in the current schema. The EXPRESS compiler parses, checks for integrity, compiles and builds a library from each schema. This library created from the EXPRESS schemas is linked to the Java development environment, where it can be accessed through the java class interface and the methods defined in these interfaces.



Class entity Java

```
package library;
import jsdai.SIntegrated_cnc_schema.AProduct;
import jsdai.SIntegrated_cnc_schema.EProduct;
import jsdai.dictionary.EAttribute;
import jsdai.lang.*;

public class Product {
      AProduct products;
      EProduct product;
      static SdaiModel model;
      public Product (SdaiModel model){
            this.model = model;
      }

public EProduct buildProduct(    String id,
                                 String name,
                                 String description,
                                 EEntity[] frame_of_reference) throws
SdaiException{
            product = (EProduct) model.createEntityInstance(EProduct.class);
            product.setId(null, id);
            product.setName(null, name);
            product.setDescription(null, description);
            EAttribute attribute;
            attribute = product.getAttributeDefinition("frame_of_reference");
            Aggregate contexts = product.createAggregate(attribute, null);
            int arg = frame_of_reference.length;
            Integer aux=0;
            while(aux<arg){
                  contexts.addUnordered(frame_of_reference[aux], null);
                  aux++;
            }
            return product;
      }
}
```

EXPRESS Entity description

```
ENTITY product;
  id : identifier;
  name : label;
  description :  OPTIONAL text;
  frame_of_reference : SET [1:?] OF product_context;
END_ENTITY; -- 10303-41: product_definition_schema
```

Class instance encoded in p21 file

```
#7=PRODUCT('default workpiece','AP-238','',(#2));
```

Figure 5.6: Java class created for the product entity

The data repository created during execution in the JSDAI session; stores the model that contains the interactions between instances, the generated data that is subsequently taken as the basis for exporting to an output STEP p21 exchange file. The generated STEP file contains each instance of the project and maintains its relationship to the STEP standard's data structure. In the model layer of the architecture, it contains a large set of classes that interact to build each unit of functionality and, as a result, generate a STEP interchange file with the encoded data.

The procedure to check the integrity of the generated file and its functionality consists of three phases. The first is to apply a test-oriented development. In this phase, each programmed application object needs to be tested in execution. It is necessary to create the classes for each entity involved in the model. For example, Figure 5.3 shows the project application object's hierarchical structure; This application object requires the following instances for its manipulation:

- `product_definition`
- `product_definition_formation`
- `machining_project`

- product_definition_context
- machining_project_workpiece_relationship
- product_context, application_context
- process_product_association
- product_definition_process
- machining_workplan

The Figure 5.3 in the upper right corner shows instantiating each class created and encoding in a neutral STEP interchange file. These instances make up the DATA section of the file, but if they are linked through the hierarchical structure that defines each application object, they will not be interpreted by other platforms.

The unit tests applied in development allow each application object to be individually analyzed, to verify its hierarchical structure, and to verify its integrity on other platforms. Within the unit tests, each class is instantiated with data extracted from a model STEP file, the values of its attributes are configured, requested, and updated to guarantee the created class behavior. Any error in the execution is easier to solve in this construction phase; the problem is that it decreases the programming time again with code to implement unit tests that check each class method. The main advantage is that unit tests ensure that each class in the project works correctly.

After an advance in development, the second method of verification consisted of developing a piece in commercial CAD software, exporting the file to the neutral STEP exchange format, in that file include new entities and definitions using the own development and checking through the commercial platform the data is interpreted appropriately.

The third verification method consisted of creating a template-type project created in our development, generating the neutral STEP exchange file, and using the STEPMachine tool from STEPTools to verify that it is imported and coded functionalities are interpreted. On this platform, the GD&T definitions, such as linking RAW part, tools, and other parameters, were defined manually and loaded on the original template file.

This case study aims to generate a consistent, semantically, and syntactically AP238 file that is interpreted by other platforms. The methodology set out to build a framework adhering to the STEP standard is a viable alternative, in force for current manufacturing technology demands and scalable future technological solutions. It is verified that interoperability barriers are overcome with the adoption of the STEP standard, the integration is correctly supported in all phases of the life cycle. Multiple functions can be developed; for example, based on making decisions for dimensional and geometric inspection, it is necessary to extract information from features, select measurement tools, create strategies and inspection plans and generate. This exposed solution methodology allows the generation of STEP file interpreters with different purposes, including the design of machine tool control, kinematic control of robots, linking new robotic structures in the manufacturing and inspection processes, etc. glimpsed among current trends.

5.3 Implementation of Case Study 2

Case study 2 aims to implement a code to read, interpret and extract information in a STEP interchange file. The STEP file contains instances with assembly data of an ASEA robot. The robot assembly comprises several parts; it is necessary to identify the number of parts, their identification, or name within the file to process that information and generate an XML structure with this information to enable a simulation based on this XML created.

The process begins with generating a virtual CAD model of the ROBOT assembly; each component of the robot involved in the kinematic control is stored as an independent part. Each part is individually positioned within the assembly, creating a functional kinematic arrangement of the robot with well-defined mates and constraints. The assembly already built is exported to the neutral STEP exchange format with two possibilities in AP242 or AP214. These two application protocols share the UoF necessary to instantiate the model with entities that relate the robot parts in a STEP exchange file format.

Knowing how the information is structured internally in the STEP interchange file is essential for extracting the data. The STEP standard's data structure represents a technological innovation to manipulate, process, and share data, but the processes to read, modify, or update these files are usually more complicated. It is possible that within this STEP file that contains information about the robot assembly, there are different configurations or views with the parts or some of the parts of the assembly to express a different context. There may be different configurations or arrangements using the parts that make up the robot, referenced in other configurations, for example, a manufacturing context and another view to express the assembly.

Internally, the data in an assembly part can be related to different structures independently to enable other application contexts. This possibility of nurturing the file with different data during the life cycle is one justification for choosing the neutral STEP interchange file to support the interoperability strategy. The parts base information does not change internally; that information is used in other contexts to express different conceptual models. Within this perspective, if we want to extract information from a specific context, such as the assembly of parts that make up the robot, it is necessary to know the conceptual model.

5.3.1 EXPRESS Entities

In the conceptual model representing the assembly application context, the product_definition entity is identified as each part's structural element. The ASEA robot assembly is made up of several parts, and from each of them, a structure is derived that involves the product_definition entity. If there are several contexts, such as manufacturing and assembly, the part's information is the same in all contexts; therefore, this entity is reused. The assembly structure will be constituted with instances of the product_definition entity representing each part that makes up the assembly.

In related geometric structures to implement assembly structures, such as the ASEA robot, there may be two alternatives:

- **Implicit relationship between assembled components**: The components of an assembly are described along with the assembly history.

This approach uses the entity representation_relationship_with_transformation.

- **Explicit representation of the assembly's geometric components**: The assembly is described with the integrated components; in other words, a part element becomes part of another assembly element used as a template in the assembly's geometry. The components are mapped through the entity mapped_item.

The two alternatives are valid to express an assembly; in cases where a kinematic study that incorporates movement is required, it may be interesting to contemplate incorporating the piece implicitly through translation and rotation movements. In this case study, the representation of implicit relationships is exposed.

The STEP standard to provide a neutral mechanism capable of describing product data throughout the life cycle incorporates Integrated generic resource: Sustainable Product structure configuration, not only for interchange files. The built-in resource defines the mechanisms for expressing the relationships between product components and assemblies.The EXPRESS definition and the explicit attributes of the `product_definition` entity are presented in Figure 5.7.

```
ENTITY product_definition
   SUPERTYPE OF (ONEOF(composite_assembly_sequence_definition , laminate_table, ply_laminate_sequence_definition ));
      id : identifier ;
      description : OPTIONAL text;
      formation : product_definition_formation ;
      frame_of_reference : product_definition_context ;
   DERIVE
      name : label := get_name_value(SELF);
   WHERE
      wr1:
         SIZEOF(USEDIN(SELF, 'STEP_MERGED_AP_SCHEMA.' + 'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY;
```

Attribute	**Type**	**Defined By**
id	identifier (STRING)	product_definition
description	text (STRING)	product_definition
formation	product_definition_formation (ENTITY)	product_definition
frame_of_reference	product_definition_context (ENTITY)	product_definition

Figure 5.7: ENTITY `product_definition`
Source:(ISO 10303-214, 2010)

Figure 5.8 shows a partial view of the product definition scheme in IS0 10303-41 and describes the subtype structure of the entities defined in this scheme. Many forms of product structure can be represented using this scheme. Two product structures are the bill of materials and the parts list structures. Parts list data structures of parts individualizes the relationship between the lower-level parts of the product structure and the higher-level assemblies in which they are contained (ISO 10303-41, 2018).

To deduce the assembly structure and using as reference the generic integrated resource ISO10303: 44 through mapping table for `parts_list`, specifically the Application element ASSEMBLY_ RELATIONSHIP, the following AIM elements are identified:

- `assembly_component_usage`
- `next_assembly_usage_occurrence`

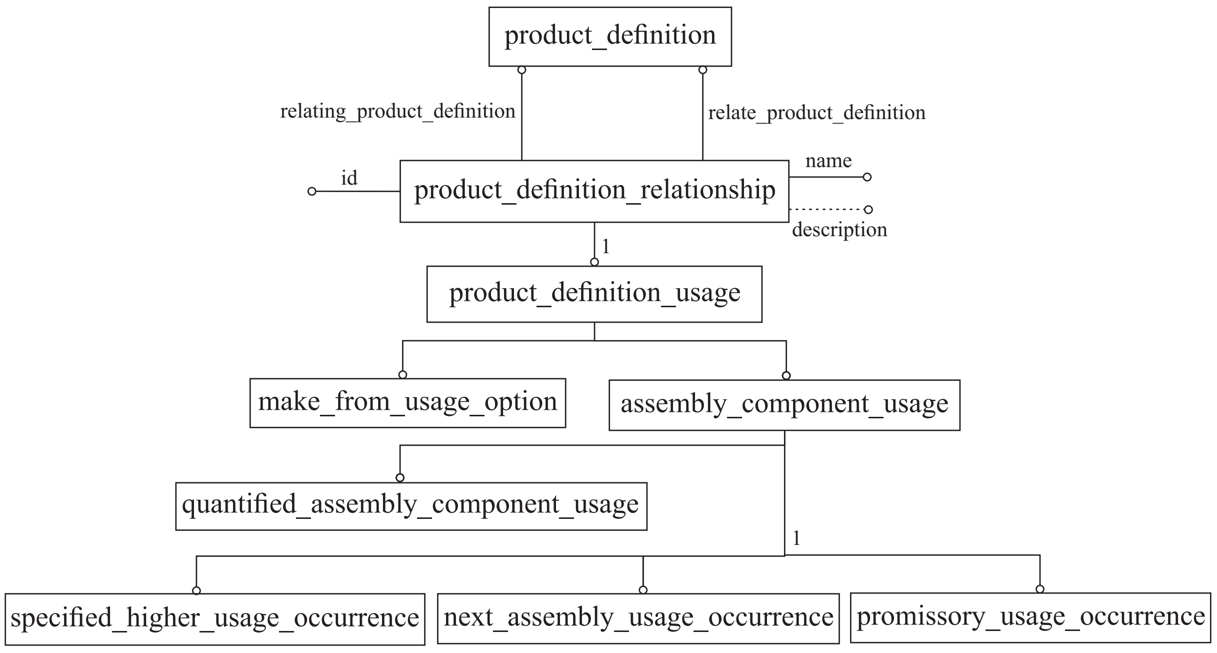

product_definition
relating_product_definition
relate_product_definition
id
product_definition_relationship
name
description
1
product_definition_usage
make_from_usage_option
assembly_component_usage
quantified_assembly_component_usage
1
specified_higher_usage_occurrence
next_assembly_usage_occurrence
promissory_usage_occurrence

Figure 5.8: Relationship of product structure entities to IS0 10303-41
Source:(ISO 10303-41, 2018)

- `quantified_assembly_component_usage`
- `specified_higher_usage_occurrence`
- `promissory_usag_occurrence`

To fully describe an application element's mapping, it may be necessary to specify a reference path through several related AIM elements. The reference path column documents the function of an AIM element and its relationship to other elements. Two or more of these related AIM elements define the interpretation of the built-in resources that satisfy the application element's requirement. For each AIM element created for use within the ISO10303 standard, a reference path to its supertype is specified from a built-in resource. The reference path for the `ASSEMBLY_RELATIONSHIP` application element is presented in Figure 5.9.

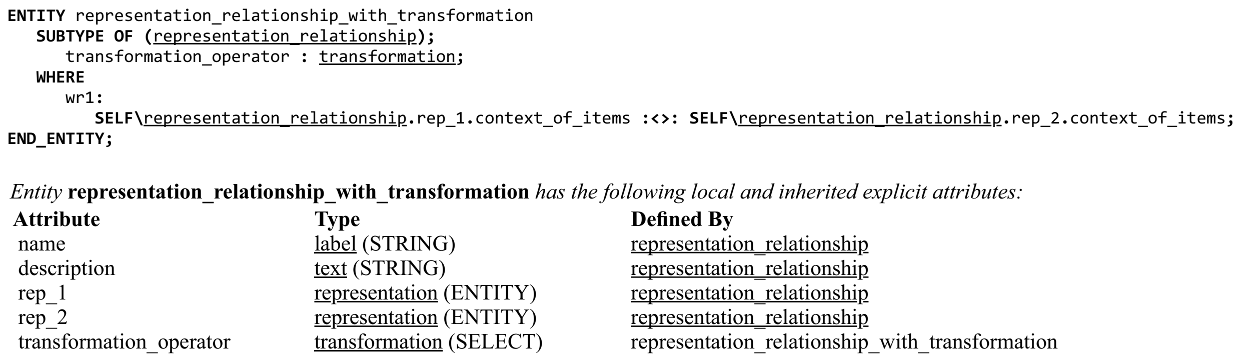
From the reference path extracted from the mapping table and presented in Figure 5.9, the following assertions can be deduced (see 4.1):

- The `product_definition_relationship.relating_product_definition` attribute references the product_definition entity
- The `product_definition.frame_of_reference` attribute references the `product_definition_context` entity
- The `product_definition_context` entity is a subtype of the `application_context_element` entity

{product_definition_relationship
product_definition_relationship.relating_product_defintion->
[product_definition
product_definition.frame_of_reference->
product_definition_context<=
application_context_element
application_context_element.name='part definition']
[product_definition<-
product_definition_context_association.definition
product_definition_context_association
product_definition_context_association.frame_of_reference->
product_definition_context<=
application_context_element
application_context_element.name='assembly definition']}

Figure 5.9: Relationship of product structure entities to IS0 10303-41
Source:(ISO 10303-41, 2018)

- The `application_context_element.name attribute` can only have the value:' part definition'
- The `product_definition` entity is referenced by `product_definition_context_association.definition`
- The `product_definition_context` entity is a subtype of the `application_context_element` entity.
- The `application_context_element.name attribute` can only have one value, 'assembly definition'



```
ENTITY representation_relationship_with_transformation
   SUBTYPE OF (representation_relationship);
      transformation_operator : transformation;
   WHERE
      wr1:
         SELF\representation_relationship.rep_1.context_of_items :<>: SELF\representation_relationship.rep_2.context_of_items;
END_ENTITY;
```

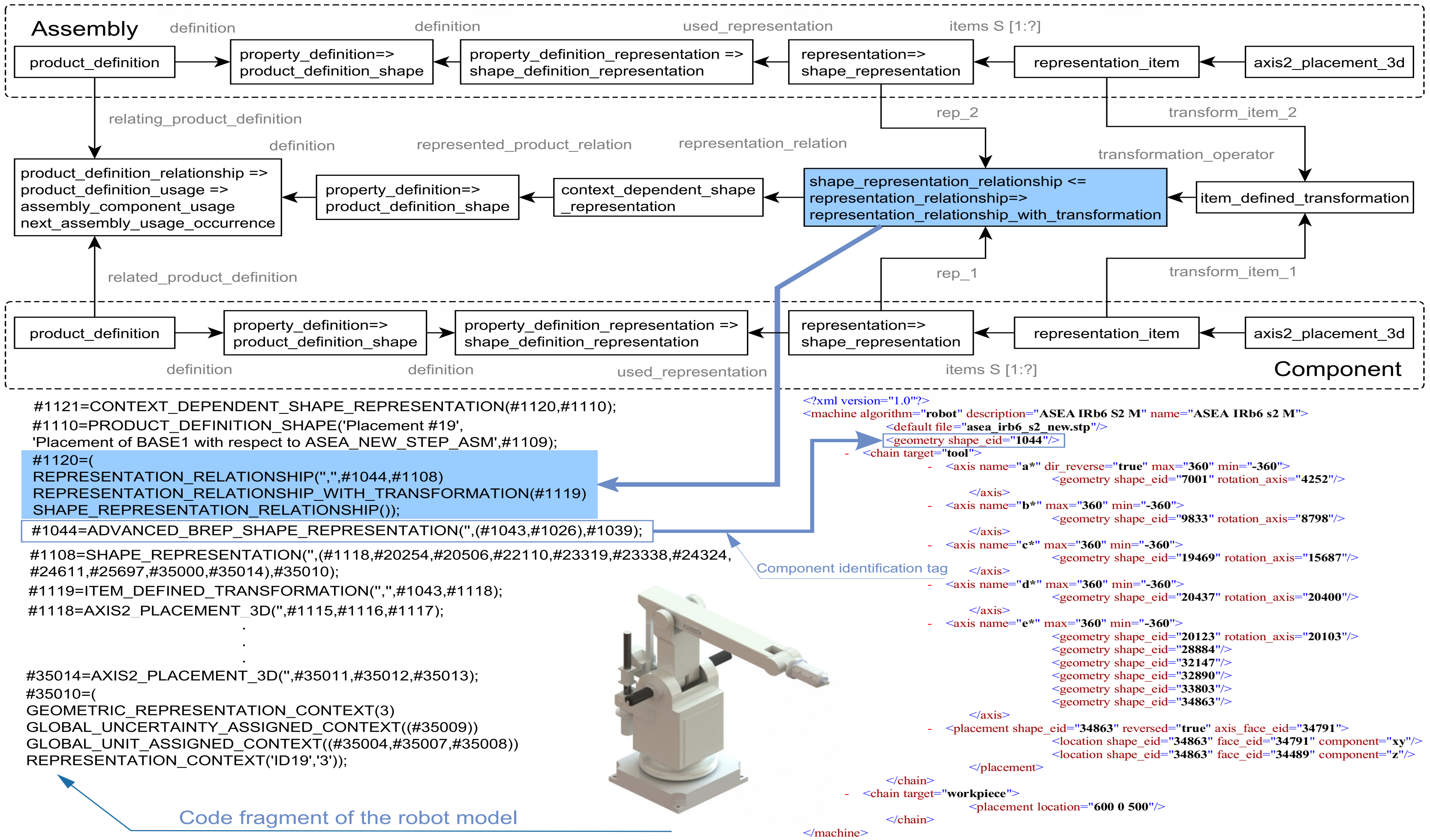
Entity **representation_relationship_with_transformation** *has the following local and inherited explicit attributes:*

Attribute	**Type**	**Defined By**
name	label (STRING)	representation_relationship
description	text (STRING)	representation_relationship
rep_1	representation (ENTITY)	representation_relationship
rep_2	representation (ENTITY)	representation_relationship
transformation_operator	transformation (SELECT)	representation_relationship_with_transformation

Figure 5.10: Entity `representation_relationship_with_transformation`
Source:(ISO 10303-41, 2018)

An assembly structure can be deduced using this mapping table and the EXPRESS definitions of each entity. The EXPRESS entities, the attributes, and relationships between the entities representing the assembly data structure contained in STEP files are illustrated in Figure 5.11. This assembly implementation approach structures each component's information and its relationships within the assembly to be shared through the STEP exchange file. The representation_relationship_with _transformation entity defines the position within an assembly (see 5.10). The relationship between the assembly and the component is given by `context_dependent_shape_representation`; this entity is responsible for distinguishing the different uses in other contexts given to a part or component of an assembly. The main entities and attributes involved in this implementation approach are described below.

A program was developed to identify tags associated with each component involved in the robot's kinematic model and create the XML file. The XML file creation starts with the STEP file processing (AP242 or AP214) to extract the helpful information to configure the parameters of the kinematic model of the robot within the XML file. The program uses the STEP standard's Express schemas to identify the explicit hierarchical structures that assemblies represent and the components and their relationships. The JSDAI Application Programming Interface (API) is used to read and manipulate object-oriented data defined in these EXPRESS data models.



Assembly
definition
product_definition
property_definition=> product_definition_shape
definition
property_definition_representation => shape_definition_representation
used_representation
representation=> shape_representation
items S [1:?]
representation_item
axis2_placement_3d
relating_product_definition
rep_2
transform_item_2
definition
represented_product_relation
representation_relation
transformation_operator
product_definition_relationship => product_definition_usage => assembly_component_usage next_assembly_usage_occurrence
property_definition=> product_definition_shape
context_dependent_shape _representation
shape_representation_relationship <= representation_relationship=> representation_relationship_with_transformation
item_defined_transformation
related_product_definition
rep_1
transform_item_1
product_definition
property_definition=> product_definition_shape
property_definition_representation => shape_definition_representation
representation=> shape_representation
representation_item
axis2_placement_3d
definition
definition
used_representation
items S [1:?]
Component
Component identification tag
Code fragment of the robot model

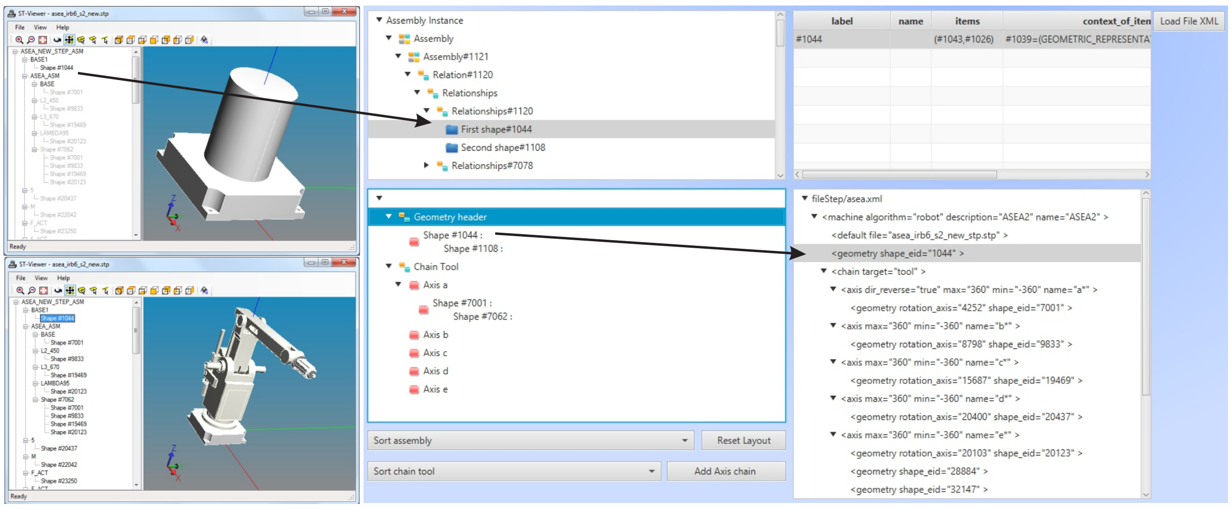
```
#1121=CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#1120,#1110);
#1110=PRODUCT_DEFINITION_SHAPE('Placement #19',
'Placement of BASE1 with respect to ASEA_NEW_STEP_ASM',#1109);
#1120=(
REPRESENTATION_RELATIONSHIP('','',#1044,#1108)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#1119)
SHAPE_REPRESENTATION_RELATIONSHIP());
#1044=ADVANCED_BREP_SHAPE_REPRESENTATION('',(#1043,#1026),#1039);
#1108=SHAPE_REPRESENTATION('',(#1118,#20254,#20506,#22110,#23319,#23338,#24324,
#24611,#25697,#35000,#35014),#35010);
#1119=ITEM_DEFINED_TRANSFORMATION('','',#1043,#1118);
#1118=AXIS2_PLACEMENT_3D('',#1115,#1116,#1117);
                          .
                          .
                          .
#35014=AXIS2_PLACEMENT_3D('',#35011,#35012,#35013);
#35010=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#35009))
GLOBAL_UNIT_ASSIGNED_CONTEXT((#35004,#35007,#35008))
REPRESENTATION_CONTEXT('ID19','3'));
```

```
<?xml version="1.0"?>
<machine algorithm="robot" description="ASEA IRb6 S2 M" name="ASEA IRb6 s2 M">
        <default file="asea_irb6_s2_new.stp"/>
        <geometry shape_eid="1044"/>
  -   <chain target="tool">
          -   <axis name="a*" dir_reverse="true" max="360" min="-360">
                      <geometry shape_eid="7001" rotation_axis="4252"/>
                </axis>
          -   <axis name="b*" max="360" min="-360">
                      <geometry shape_eid="9833" rotation_axis="8798"/>
                </axis>
          -   <axis name="c*" max="360" min="-360">
                      <geometry shape_eid="19469" rotation_axis="15687"/>
                </axis>
          -   <axis name="d*" max="360" min="-360">
                      <geometry shape_eid="20437" rotation_axis="20400"/>
                </axis>
          -   <axis name="e*" max="360" min="-360">
                      <geometry shape_eid="20123" rotation_axis="20103"/>
                      <geometry shape_eid="28884"/>
                      <geometry shape_eid="32147"/>
                      <geometry shape_eid="32890"/>
                      <geometry shape_eid="33803"/>
                      <geometry shape_eid="34863"/>
                </axis>
          -   <placement shape_eid="34863" reversed="true" axis_face_eid="34791">
                      <location shape_eid="34863" face_eid="34791" component="xy"/>
                      <location shape_eid="34863" face_eid="34489" component="z"/>
                </placement>
        </chain>
  -   <chain target="workpiece">
                <placement location="600 0 500"/>
        </chain>
</machine>
```

Figure 5.11: Graphical model with the reference paths followed in interpreting the application's requirements
Source:(ÁLVARES et al., 2020)

The structures and entities in the STEP file of the virtual robot model are processed, identified and extracted, and subsequently inserted into the XML file. The Figure 5.11 presents a graphical model with the reference paths followed in interpreting the application's requirements to generate the XML file.The reference paths extracted from the STEP standard's integrated resources allow knowing the STEP interchange file's data structure and formulate a strategy to access these entities attributes. Each entity in the neutral STEP exchange file has a specific reference path, making it possible to reuse it in different application contexts. In our case, the developed application extracts only the assembly's information and its components and presents the necessary attributes for creating the XML file.

In the upper part of Figure 5.11, the reference paths that structure the assembly information are presented. In the lower-left part, a code fragment of the robot's virtual model is shown with the information of an instance. The application creates a list of the elements and attributes of each entity. This information helps identify each element of the robot's kinematic model and configure the string in the XML file. In the lower right part of the figure, the XML file generated for the ASEA robot simulation is presented. It is observed that the tag that identifies the component entity in the model file is finally part of the XML file.



label	name	items	context_of_iten
#1044		(#1043,#1026)	#1039=(GEOMETRIC_REPRESENTA

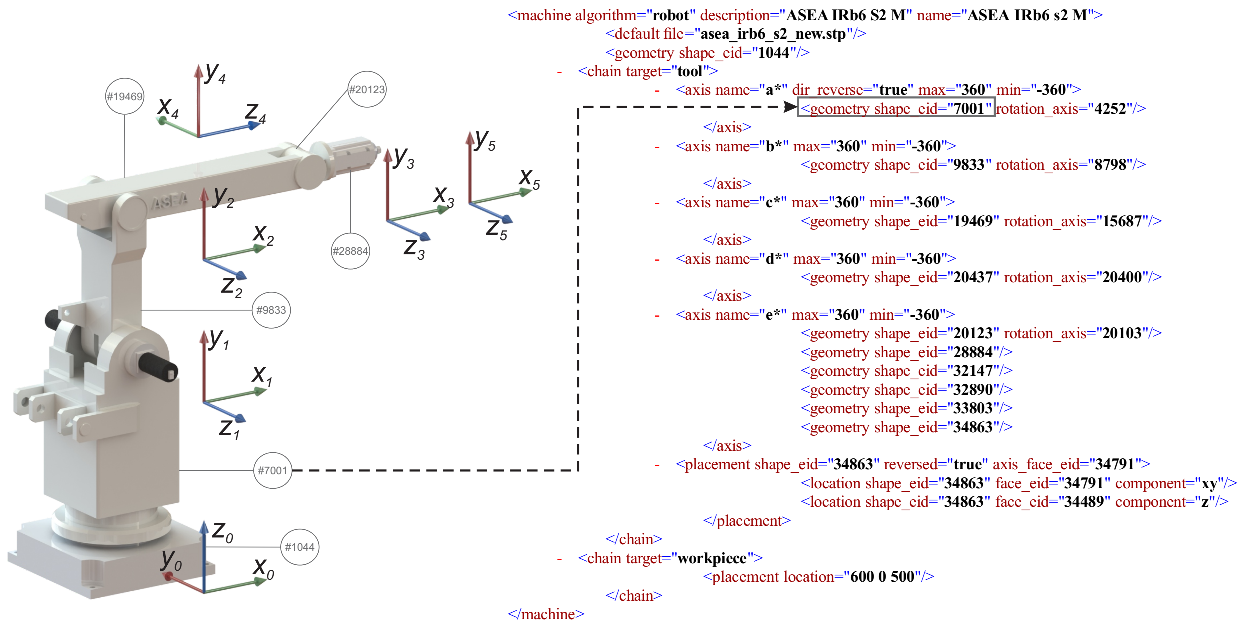
```
fileStep/asea.xml
  <machine algorithm="robot" description="ASEA2" name="ASEA2" >
    <default file="asea_irb6_s2_new_stp.stp" >
    <geometry shape_eid="1044" >
    <chain target="tool" >
      <axis dir_reverse="true" max="360" min="-360" name="a*" >
        <geometry rotation_axis="4252" shape_eid="7001" >
      <axis max="360" min="-360" name="b*" >
        <geometry rotation_axis="8798" shape_eid="9833" >
      <axis max="360" min="-360" name="c*" >
        <geometry rotation_axis="15687" shape_eid="19469" >
      <axis max="360" min="-360" name="d*" >
        <geometry rotation_axis="20400" shape_eid="20437" >
      <axis max="360" min="-360" name="e*" >
        <geometry rotation_axis="20103" shape_eid="20123" >
        <geometry shape_eid="28884" >
        <geometry shape_eid="32147" >
```

Figure 5.12: Robot assembly component reference labels
Source: (ÁLVARES et al., 2020)

An example of the process followed to generate the XML file is shown in Figure 5.12. To correctly configure the different parameters, it is necessary to identify within the STEP file of the robot model the reference labels of each kinematic chain component. Alternatively, ST-Viewer software is used to assist this identification phase. ST-Viewer presents, in addition to the virtual robot model view, information on the reference paths valid to define without mistake the geometry labels in the `shape_eid` attribute and surface labels in the `face_eid` attribute of the XML file.

The positions of each of the rotational joints of the ASEA IRB6-S2 robot are also defined. In the file, five axes (a, b, c, d, e) are created and configured to represent the degrees of freedom that the robot has. Each of the elements associated with that degree of freedom is defined in the axes (a, b, c, d, e). A position concerning the robot's fixed coordinate system is defined to locate its workspace's workpiece. The Figure 5.13 shows the XML file's final configuration, which enables virtual simulation within the

STEP-MACHINE environment.



```
<machine algorithm="robot" description="ASEA IRb6 S2 M" name="ASEA IRb6 s2 M">
          <default file="asea_irb6_s2_new.stp"/>
          <geometry shape_eid="1044"/>
   -   <chain target="tool">
          -   <axis name="a*" dir_reverse="true" max="360" min="-360">
                        <geometry shape_eid="7001" rotation_axis="4252"/>
                 </axis>
          -   <axis name="b*" max="360" min="-360">
                        <geometry shape_eid="9833" rotation_axis="8798"/>
                 </axis>
          -   <axis name="c*" max="360" min="-360">
                        <geometry shape_eid="19469" rotation_axis="15687"/>
                 </axis>
          -   <axis name="d*" max="360" min="-360">
                        <geometry shape_eid="20437" rotation_axis="20400"/>
                 </axis>
          -   <axis name="e*" max="360" min="-360">
                        <geometry shape_eid="20123" rotation_axis="20103"/>
                        <geometry shape_eid="28884"/>
                        <geometry shape_eid="32147"/>
                        <geometry shape_eid="32890"/>
                        <geometry shape_eid="33803"/>
                        <geometry shape_eid="34863"/>
                 </axis>
          -   <placement shape_eid="34863" reversed="true" axis_face_eid="34791">
                        <location shape_eid="34863" face_eid="34791" component="xy"/>
                        <location shape_eid="34863" face_eid="34489" component="z"/>
                 </placement>
          </chain>
   -   <chain target="workpiece">
                 <placement location="600 0 500"/>
          </chain>
</machine>
```

Figure 5.13: XML representation for virtual robot
Source:(ÁLVARES et al., 2020)

With the XML file already created and configured, it is stored together with the robot's virtual model in the STEP-NC Machine installation folder and the available machine tools files. The ASEA IRB6-S2 robot can be used in simulation as a machine tool. Figure 5.14 shows the environment ready to run the simulation and verify the programs generated through each architecture. Each program generated specifically for the manufacture of a part model is loaded into the STEP-NC Machine environment. Figure 5.14 shows the procedure to include the part's model to be manufactured and simulate the manufacture using the ASEA IRB6-S2 robot. The simulation allows off-line programming, verification, and validation of each architecture's results, executed avoiding using the real robot in this analysis phase, providing an excellent environment to test manufacturing strategies, analyze results, and make decisions without compromising resources. The simulation results will enable the real manufacture of each piece studied.

5.4 Implementation of Case Study 3

Dimensional measurement has come a long way to be considered an essential tool for improving advanced manufacturing processes. Previously, the process was limited to taking measures within a controlled environment to obtain uncertainty and error values. The demands of increasingly efficient and sustainable processes promote significant changes, which pose new challenges and establish several unresolved issues. Within the integration and interoperability context, the main issues arise from integrating, planning, and controlling the manufacturing and measurement systems. Solutions in defining architectures, communication mechanisms with machine controllers, and measurement with machine tools currently focus on different researches.

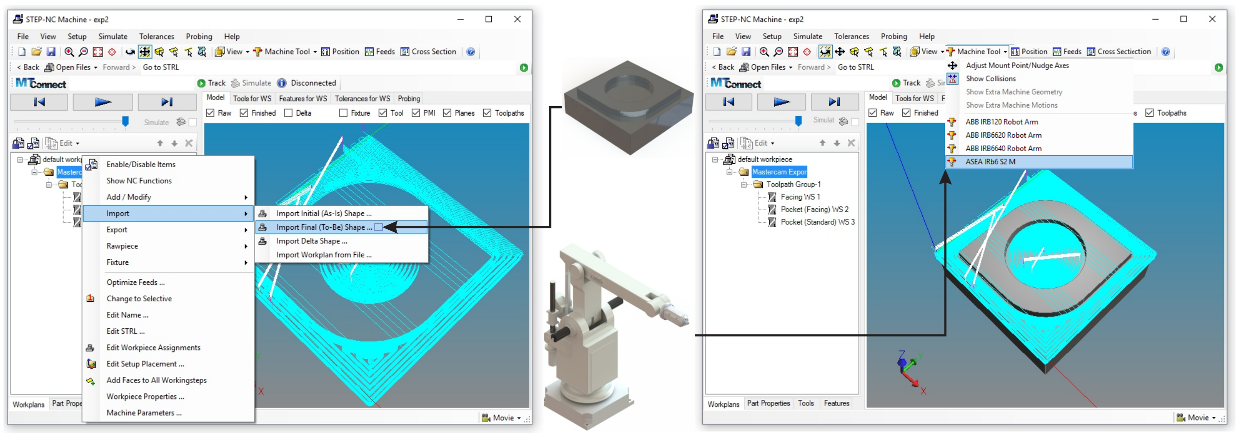

STEP-NC Machine - exp2
File View Setup Simulate Tolerances Probing Help
View Machine Tool Position Feeds Cross Section
Back Open Files Forward Go to STRL
Track Simulate Disconnected
Model Tools for WS Features for WS Tolerances for WS Probing
Raw Finished Delta Fixture Tool PMI Planes Toolpaths
Enable/Disable Items
Show NC Functions
Add / Modify
Import
Export
Rawpiece
Fixture
Optimize Feeds ...
Change to Selective
Edit Name ...
Edit STRL ...
Edit Workpiece Assignments
Edit Setup Placement ...
Add Faces to All Workingsteps
Workpiece Properties ...
Machine Parameters ...
Import Initial (As-Is) Shape ...
Import Final (To-Be) Shape ...
Import Delta Shape ...
Import Workplan from File ...
Workplans Part Properties
Movie
Adjust Mount Point/Nudge Axes
Show Collisions
Show Extra Machine Geometry
Show Extra Machine Motions
ABB IRB120 Robot Arm
ABB IRB6620 Robot Arm
ABB IRB6640 Robot Arm
ASEA IRb6 S2 M
default workpiece
Mastercam Export
Toolpath Group-1
Facing WS 1
Pocket (Facing) WS 2
Pocket (Standard) WS 3
Workplans Part Properties Tools Features

Figure 5.14: STEP-NC Machine simulation environment
Source:(ÁLVARES et al., 2020)

This sect aims to describe the methods of measuring parts, the defined control architecture, and the feedback strategy. Coordinate measurement is a flexible measurement technique compatible with other technologies that provide the results necessary to achieve the project objectives. To link the measurement results in a closed-loop with the manufacturing processes, the CMM machine (Coordinate Measuring Machines) is used to acquire dimensional information of the part. The following are aspects related to dimensional and geometric inspection.

5.4.1 Extracting CAD Information Requirements

There are different tools to create CAD models, but it is vitally important to develop a measurement plan. In the proposed application context, the part design is developed based on features. In the product data model based on the STEP standard, it is necessary to use the application protocols functional units to define the features and properties. Listed below are the UoF available in the STEP standard application protocols to support information modeling and parts design.

- *The part shape representation model*
- *Manufacturing feature model*
- *Part geometry model*
- *Part topology model*
- *Tolerances model*
- *Manufacturing part properties model*
- *Part administration data model*

Part modeling technology based on features is the main component in integrating the product life cycle processes. The STEP standard mainly uses this technology to provide a neutral, interoperable format for exchanging product information.

In extracting requirements from the CAD model, some technological solutions can support the initial phase of construction of the 3D model, specification of geometric and dimensional tolerances.

5.4.2 Typical case simulation: Inspection at the Start of production

A manufacturing case is simulated to demonstrate the architecture, the mode of operation, and the scope of the methodology exposed with the developed system. Figure 5.15 presents the projected scenario and the phases required to simulate a case for a closed-loop inspection.

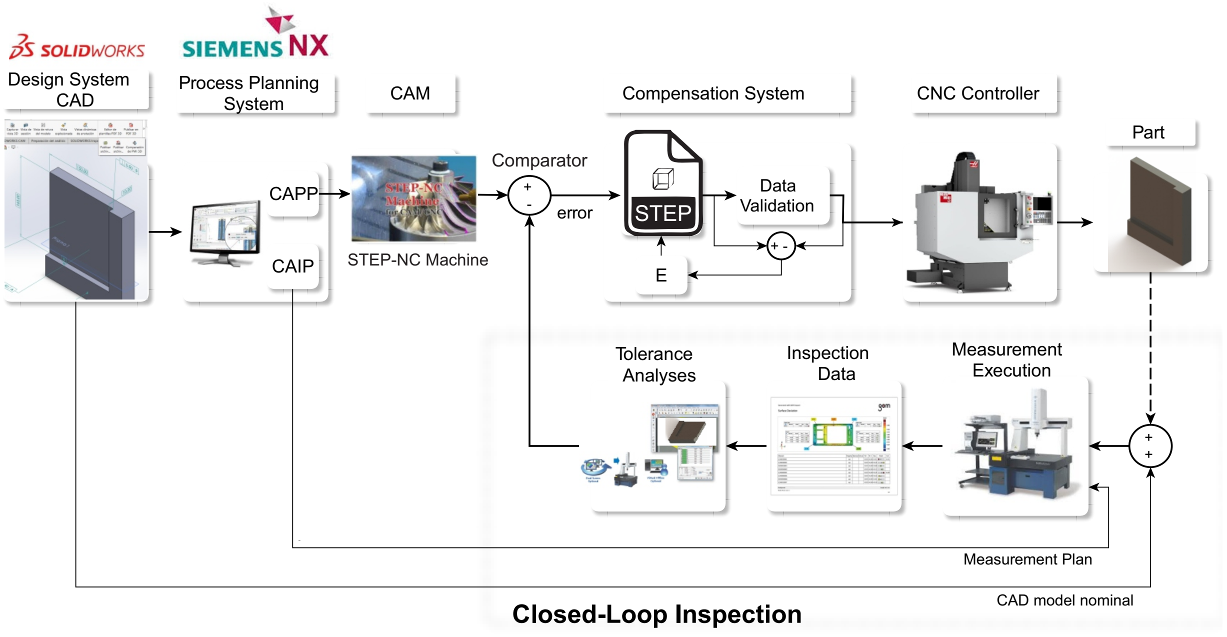


Figure 5.15: Scenario and phases of architecture

The physical configuration used as a reference for the simulation is shown in Figure 5.15. In this configuration, different technologies of proprietary CAx systems interact. The developed solution manages to be integrated into the architecture within the creation of the neutral STEP interchange file that contains the CAM process in AP238 format. The system begins with a CAD model of a prismatic part projected for fabrication. The part is created within the Solidworks design environment using sketch tools. After the part design, the DimXpert tool included in Solidworks defines the geometric tolerances: flatness, parallelism, and perpendicularity. Figure 5.16 shows the definition of geometric tolerance. DimXpert adds reference dimensions by applying dimensions in drawings to fully define manufacturing features, such as patterns, grooves, and cavities.

After having the part and its feature completely defined both dimensionally and geometrically, the definition-based model is exported, encoding the information in an AP242 format. Figure 5.16 shows the export process carried out from Solidworks. This first phase of CAD design in the architecture is carried out in proprietary software; the integration and interoperability condition is not affected since the format used for interchange is based on a neutral STEP interchange file. This model will accompany the subsequent phases and can be nurtured with new entities generated in other phases of the architecture.

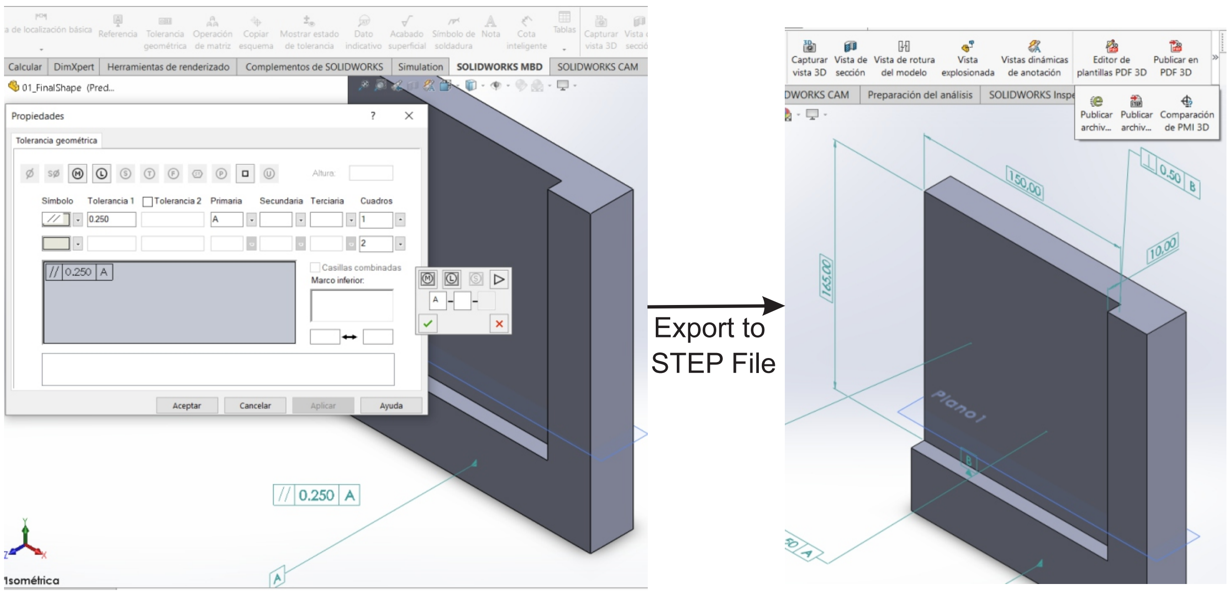


Figure 5.16: Definition of tolerances with DimXpertManager for the example part

5.4.2.1 CAD/CAPP/CAM integration

The framework is used to encode a series of tool paths are encoded to build the CAD design feature. The framework generates a neutral STEP exchange file (see annex B.2) in AP238 format containing the project, main workplan, and a workingstep with the facing operation that contains the toolpath.

The STEP-NC Machine software is used to define and nurture the file with more relevant information in the CAPP and CAM system. The neutral STEP exchange file (see annex B.2) is loaded into the STEP-NC Machine environment; the project structure is shown in the left pane of the environment as shown in Figure 5.17. This file contains the project structure and a Facing operation through toolpaths. With the environment import tools, the following parts are included in the project from independent STEP exchange formats: initial Shape, Final Shape, Delta Shape, and Tool Shape. The GD&T tolerance information is also linked, extracted from the AP242 file that contains the Final Shape.

5.4.2.2 Generation of the STEP-NC program

The STEP-NC Machine environment allows to include information relevant to the manufacturing process within the STEP neutral interchange file. The consolidated information can be exported again, and the new file will contain this helpful information for manufacturing on machines with STEP file interpreters. Due to its interoperable nature, the information contained in the file can be accessed at any stage of the life cycle and by any technology. The data structure correctly supports the manufacturing process and each of the operations necessary for advanced manufacturing.

The statements used and of possible processing or correction by the compensation system are presented below:

- Statements of geometric definitions to define points, lines, circles, and planes.

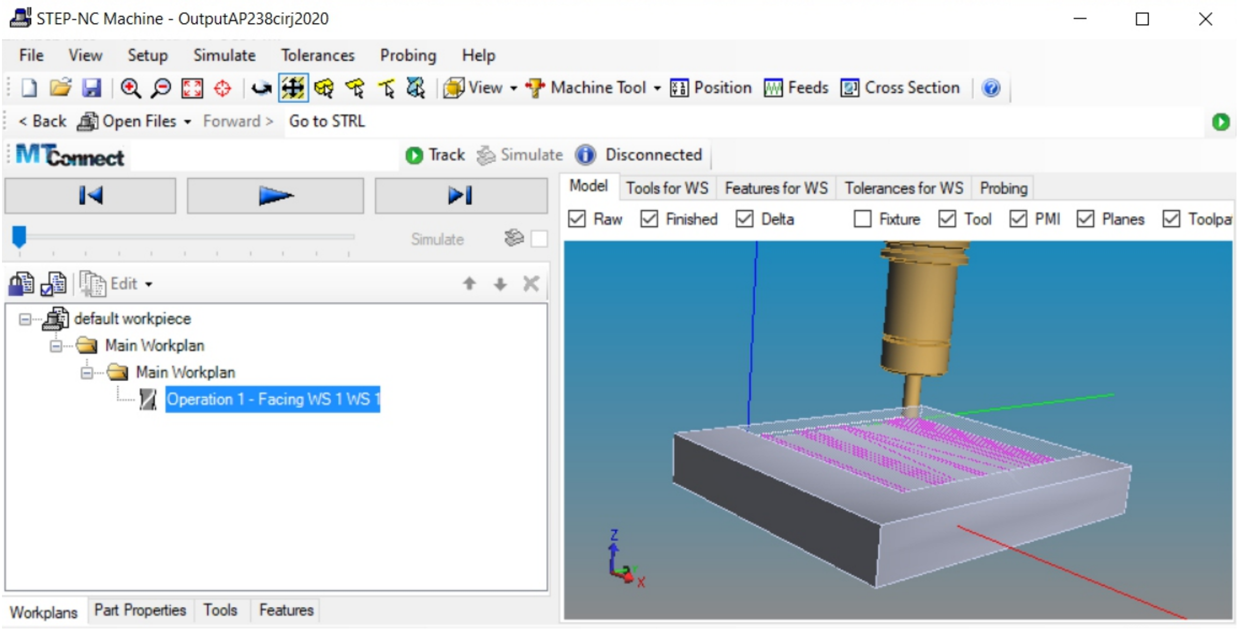

STEP-NC Machine - OutputAP238cirj2020
File View Setup Simulate Tolerances Probing Help
View Machine Tool Position Feeds Cross Section
< Back Open Files Forward > Go to STRL
MTConnect
Track Simulate Disconnected
Simulate
Edit
default workpiece
Main Workplan
Main Workplan
Operation 1 - Facing WS 1 WS 1
Workplans Part Properties Tools Features
Model Tools for WS Features for WS Tolerances for WS Probing
Raw Finished Delta Fixture Tool PMI Planes Toolpa
Z
X

Figure 5.17: Integration of the Step file project with additional information for the manufacturing process

- GD&T setting specification.
- Modification or generation of new trajectories that compensate or reduce the uncertainty in the result.

Processing result, a program in AP238 format is generated. The appendix B.2.2 shows a code snippet AP238 program generated in the STEP-NC Machine.

5.4.2.3 CAD/CAIP/CAI integration

The integrated inspection process within the manufacturing chain plays an essential role in improving the process. The inspection is in charge of making a check of the design specifications on the manufactured product to inform the differences and through an analysis to establish the respective corrections. The inspection plan CAIP generates the plan to perform the measurement takes the structured product information from the digital model that contains the geometries and definitions necessary to represent the nominal restrictions of the product. The model developed in the CAD system specifies the nominal values, variations, limits, tolerances, and restrictions in the geometry of a part. The plans generated with CAIP systems are usually developed for CMM and can be executed to control tolerances and check geometry (NASR et al., 2016).

CAIP systems generate heuristic rules with a determined sequence to measure features. Inspection can be considered global when geometric features are grouped to create a global inspection plan or location when each feature is divided for measurement into geometric components such as (plane, circle, lines). In both cases, the planning includes the number of measurement points and the trajectory to be covered. The main activities developed in the CAIP system are: determining the number of measurement points, selecting the appropriate sensor to inspect each feature, analyzing the accessibility of each measurement point, generating the trajectories, simulating the inspection process, creating an execution plan, perform the measurement and analyze the results.

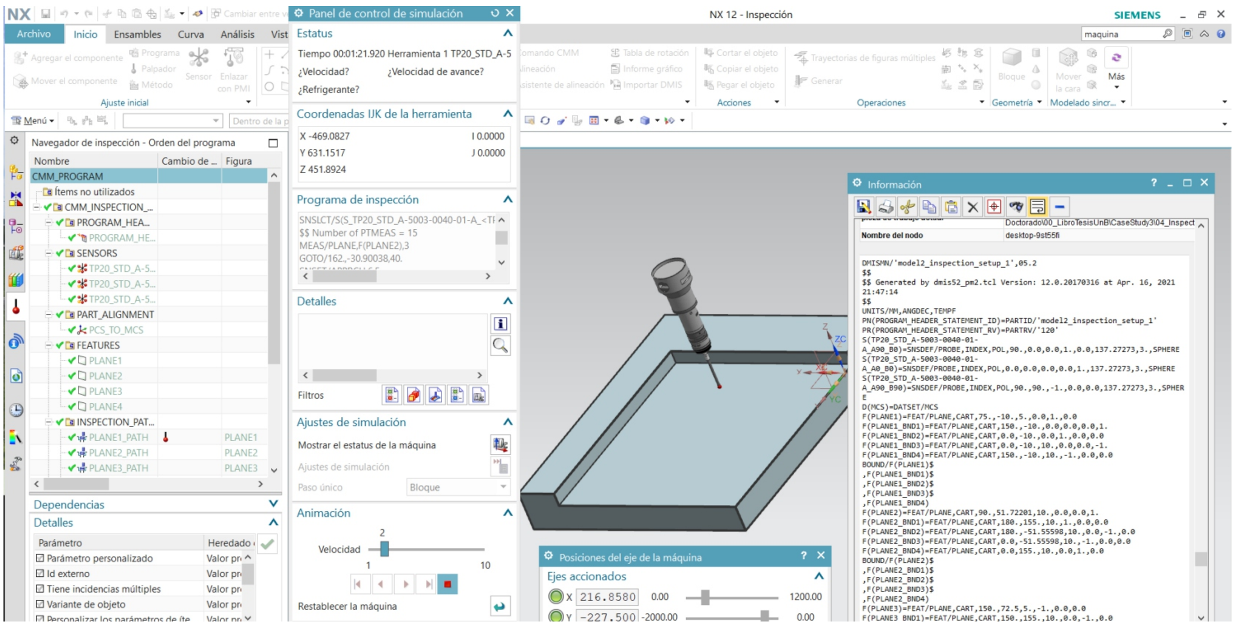
Inspection results integrated with manufacturing processes can be used as a reference for positioning (raw material, zero compensation), planning sequences of machining operations, and compensating for systematic errors. In some cases, the integration of dimensional inspection planning with execution systems is done through the DMIS protocol (Dimensional Measuring Interface Standard), which, unlike STEP/STEP-NC, product definitions are not based on features. The protocol establishes the communication between inspection programs and measurement equipment, mainly with proprietary technologies. The planning-based execution software creates a program to carry out specific measurement operations, which contains low-level commands necessary to control the equipment in charge of the inspection. I++ DME is the protocol used to conduct CMM. The measurement results are transferred to the analysis report using DML (Dimensional Markup Language), responsible for structuring the transfer between inspection devices and databases based on XML. The DMIS provides CAIP integration with CAI but does not integrate the results and analyses with the original project, which generates communication problems due to the exchange of formats and incompatibilities with the STEP environment due to not having features to manufacture the flow of unidirectional information.

The use of STEP-NC in inspection tasks allows generating automatic alternatives to move the CMM. The automatic definition of inspection strategies and operations is not included in the standard and must be defined externally by the user. The standard in this regard supports data exchange but does not provide solutions due to complexity (orientation, movement, speed). Any autonomous functionality must be developed based on the information provided by STEP and implemented in the control of the machine. AP219 was created to exchange dimensional inspection data to focus on planning inspection operations and to specify the GD&T of a product within a neutral format.

The Figure 5.18 shows the inspection planning process performed at NX-Siemens. NX ™ CMM inspection scheduling software allows generating quality inspection schedules, typically in the industry-standard Dimensional Measurement Interface Specification (DMIS) format.

Nx-Siemens provides an environment for planning both manufacturing and inspection processes. The objective is to simulate an automatic measurement environment that involves detecting tolerance specifications, defining touch probes, creating collision-free trajectories, and reconfiguring the order in which those trajectories are executed. Different part setups can be considered in the inspection process when it is necessary to access features in complex parts. The simulation is based on a CAD model exported from Solidwork to demonstrate interoperability through the neutral STEP format. From this CAD model based on definition and structure in a neutral exchange file with AP242 structure, the dimensional and geometric specifications are extracted to start the inspection planning. The planning process follows the procedure described in the section 2.6.4. When the simulations are carried out, and the configurations applied in the CAIP are approved, a DMIS output file is generated due to the process, helpful in executing the measurement process. The appendix D.4.1 contains the DMIS output file generated in the inspection process planning.

After detecting an error, the compensation strategy can be linked to the framework, in charge of reading the values susceptible to changes in the AP238 file and making the necessary modifications to correct



Tiempo 00:01:21.920 Herramienta 1 TP20_STD_A-5
¿Velocidad? ¿Velocidad de avance?
¿Refrigerante?

Coordenadas IJK de la herramienta
X -469.0827 I 0.0000
Y 631.1517 J 0.0000
Z 451.8924

Posiciones del eje de la máquina
X 216.8580 0.00 1200.00
Y -227.500 -2000.00 0.00

Información

	Doctorado\00_LibroTesisUnB\CaseStudy3\04_Inspect
Nombre del nodo	desktop-9st55fi

```
DMISMN/'model2_inspection_setup_1',05.2
$$
$$ Generated by dmis52_pm2.tcl Version: 12.0.20170316 at Apr. 16, 2021
21:47:14
$$
UNITS/MM,ANGDEC,TEMPF
PN(PROGRAM_HEADER_STATEMENT_ID)=PARTID/'model2_inspection_setup_1'
PR(PROGRAM_HEADER_STATEMENT_RV)=PARTRV/'120'
S(TP20_STD_A-5003-0040-01-
A_A90_B0)=SNSDEF/PROBE,INDEX,POL,90.,0.0,0.0,1.,0.0,137.27273,3.,SPHERE
S(TP20_STD_A-5003-0040-01-
A_A0_B0)=SNSDEF/PROBE,INDEX,POL,0.0,0.0,0.0,0.0,1.,137.27273,3.,SPHERE
S(TP20_STD_A-5003-0040-01-
A_A90_B90)=SNSDEF/PROBE,INDEX,POL,90.,90.,-1.,0.0,0.0,137.27273,3.,SPHER
E
D(MCS)=DATSET/MCS
F(PLANE1)=FEAT/PLANE,CART,75.,-10.,5.,0.0,1.,0.0
F(PLANE1_BND1)=FEAT/PLANE,CART,150.,-10.,0.0,0.0,0.0,1.
F(PLANE1_BND2)=FEAT/PLANE,CART,0.0,-10.,0.0,1.,0.0,0.0
F(PLANE1_BND3)=FEAT/PLANE,CART,0.0,-10.,10.,0.0,0.0,-1.
F(PLANE1_BND4)=FEAT/PLANE,CART,150.,-10.,10.,-1.,0.0,0.0
BOUND/F(PLANE1)$
,F(PLANE1_BND1)$
,F(PLANE1_BND2)$
,F(PLANE1_BND3)$
,F(PLANE1_BND4)
F(PLANE2)=FEAT/PLANE,CART,90.,51.72201,10.,0.0,0.0,1.
F(PLANE2_BND1)=FEAT/PLANE,CART,180.,155.,10.,1.,0.0,0.0
F(PLANE2_BND2)=FEAT/PLANE,CART,180.,-51.55598,10.,0.0,-1.,0.0
F(PLANE2_BND3)=FEAT/PLANE,CART,0.0,-51.55598,10.,-1.,0.0,0.0
F(PLANE2_BND4)=FEAT/PLANE,CART,0.0,155.,10.,0.0,1.,0.0
BOUND/F(PLANE2)$
,F(PLANE2_BND1)$
,F(PLANE2_BND2)$
,F(PLANE2_BND3)$
,F(PLANE2_BND4)
F(PLANE3)=FEAT/PLANE,CART,150.,72.5,5.,-1.,0.0,0.0
F(PLANE3_BND1)=FEAT/PLANE,CART,150.,155.,10.,0.0,-1.,0.0
```

Figure 5.18: Inspection program execution

errors.

5.4.3 Typical case simulation: Inspection within production cycle

In industrial manufacturing processes, the dimensional management of the product fulfills several roles. It is necessary to maintain a balance between productivity and quality. A quality product previously required demanding dimensional and geometric specifications, which significantly influences productivity. In the current technological circumstances and within the proposed architecture, a closed-loop inspection can ideally guarantee the quality of the product and maintain a high level of productivity. The question is that overvalued dimensional specifications cause manufacturing problems because they require capacity in machines and processes adjusted to meet these demanding specifications. Parts with demanding specifications are of better quality but more challenging to manufacture, impacting productivity. Another aspect derived from this issue is that relaxed geometric, and dimensional specifications increase productivity but sacrifice quality in the final result. The closed-loop inspection architecture seeks to find a zone of dimensional compliance that establishes a balance between productivity and quality, seeking continuous improvement of the process.

Dimensional conformance offers a robust design against dimensional variations in manufacturing and error-controlled production processes. In the previous case study, a control scheme was proposed to verify and control the errors that may occur in a part that is to be produced in series. The process is adjusted to meet these design specifications by making. In this case, we will present a scheme that can be applied to obtain dimensional conformity that maintains the balance between productivity and quality and serves as a strategy for process control compensating for errors that can be identified. The paths to achieve dimensional compliance include information generated in different phases of the life cycle. Can improve the geometric specifications of the product in the design phase, control the production and manufacturing processes and improve the way to inspect or take measurements on the part.

Every piece performs a functional role and is often part of an assembly. Not all dimensions must be demanding to fulfill a function; however, one must perform a functional analysis to identify the critical dimensions that require a more demanding specification. Computational resources can support this work of functional analysis and definition of specifications. The central limit theorem is a proper method to estimate the behavior of random variables such as the result of manufacturing a part. The theorem indicates that when the size of a bone sample and the number of pieces manufactured is large enough, the distribution of the mean follows approximately normal or Gaussian distribution. The samples accumulate around the mean and with extreme values far from that mean; this concept is known as the confidence interval.

5.4.3.1 Define the Problem

For this analysis, the inspection results obtained from the production follow-up of a part are used. The measurement was performed in a Mitutoyo CMM, and the results exported from MeasurLink to identify causes of particular variation within the process and identify opportunities for improvement that reduce process variation. The measurement is carried out on features considered critical and must be monitored. The inspection results are used to estimate future data and determine if the process is under control.

5.4.3.2 Determine the factors that affect the process of manufacturing

The statistical study is carried out to find a relationship between the errors and the causes that cause them. It also makes it possible to determine the capacity of the machine to manufacture these types of parts. Measurements are made on two linear features of the part that can provide relevant information on the process. These features represent the functional features of the part under control.

The case of process control is defined between the limits of 6σ. The figure shows a graph of measures by variable extracted from the inspection results, thrown by Measurerlink from where the limits defined for this section could be observed. The variability measure is associated with considering the Gaussian result where the $\mu \pm 3\ \sigma$ range includes approximately 99.7% of the characteristic values. The range limits define natural tolerances as intrinsic to the process. The specification limits (LIE and LSE) are considered to define the range of permissible values of the variables that are forced. The objective defined by the mean value μ will be centered concerning the specification limits. Capability index is defined as:

$$C_p = LSE - LIE/6\sigma \tag{5.1}$$

where

$$6\sigma = (UCL_{\bar{x}} - LCL_{\bar{x}})\sqrt{n} \tag{5.2}$$

and

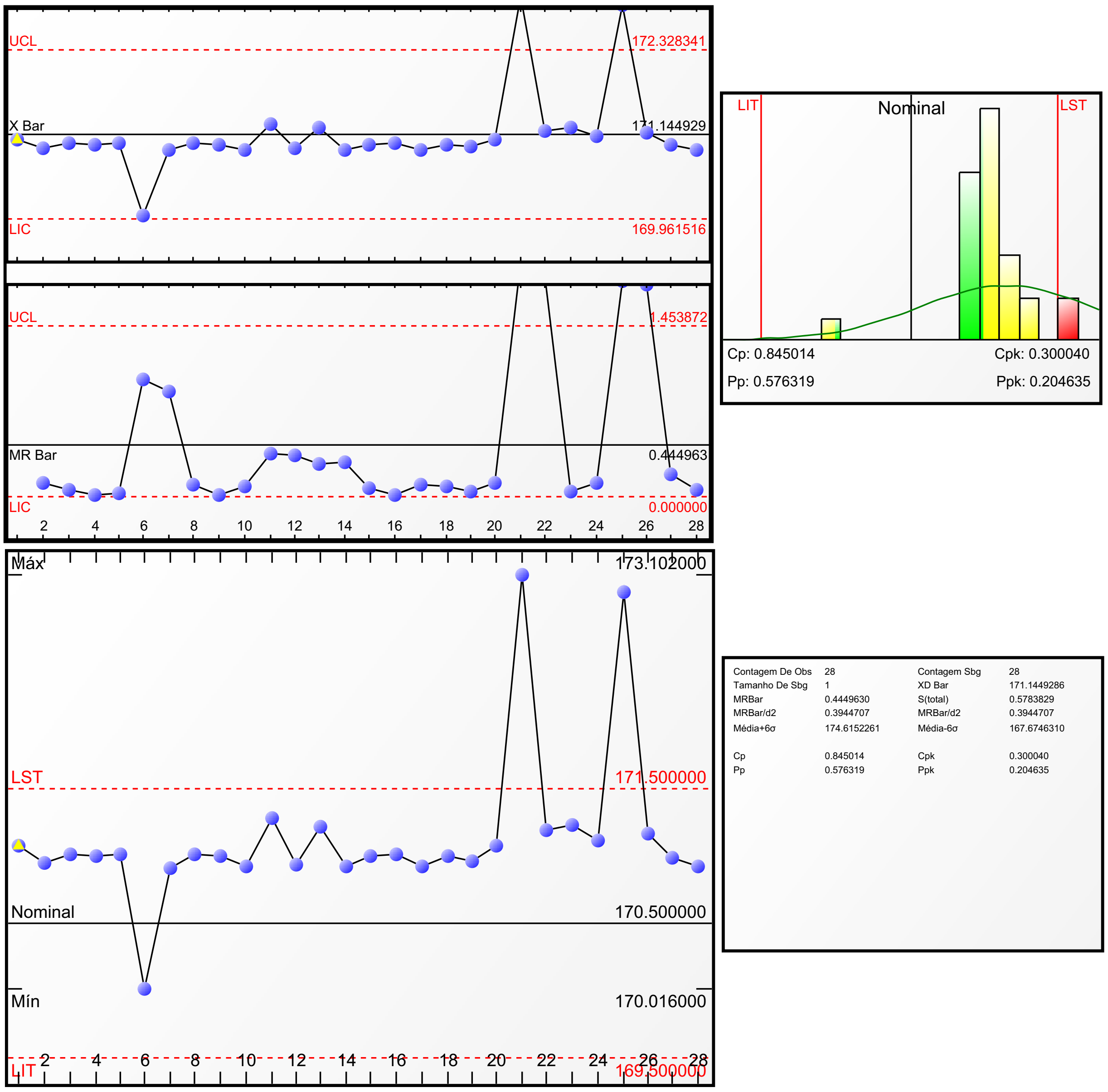
$$Tolerance_{Natural} = (UCL_{\bar{x}} - LCL_{\bar{x}})\sqrt{n} \tag{5.3}$$

If μ and σ are known, then the construction of the graph of media is immediate from its definition:

$$UCL_{\bar{x}} = \bar{x} + A_2 * \bar{R} \tag{5.4}$$

$$LCL_{\bar{x}} = \bar{x} - A_2 * \bar{R} \tag{5.5}$$

The Figure 5.19 presents a graphical result of means. An analysis of the results is necessary to obtain more information on the behavior of the process. Indeed, from the dispersion of the sample values, it is possible to estimate the dispersion of the process and follow the evolution. The standard deviation σ measures the dispersion of the values of a population, and the most frequently used sample estimators are the path R (Graph R) and the sample standard deviation S (Graph S).



Contagem De Obs	28	Contagem Sbg	28
Tamanho De Sbg	1	XD Bar	171.1449286
MRBar	0.4449630	S(total)	0.5783829
MRBar/d2	0.3944707	MRBar/d2	0.3944707
Média+6σ	174.6152261	Média-6σ	167.6746310
Cp	0.845014	Cpk	0.300040
Pp	0.576319	Ppk	0.204635

Figure 5.19: Measurement graph for the variable

Table 5.1 presents a summary of the measurement result returned by Measurlink that presents standard deviation (σ), natural process variability (6σ), and process capability (Cp) (see appendix G.6).

Table 5.1: References for quality control

Indicator	**6σ**	σ	**Cp**	**Cpk**	$\bar{X}$	$\bar{R}$
Dx_Tolerância-DSTNCE	3,4702	0,5783	0,8450	0,3000	171,1449	0,4449
Dy_Tolerância-DSTNCE	2,7533	0,4588	0,8408	0,7911	170,4408	0,4471
Ly_1 Length Toleranc-DSTNCE	3,8667	0,6444	0,6805	0,6646	170,5233	0,5524
Ly_2 Lenght_Toleran-DSTNCE	4,4777	0,7462	0,6013	0,5647	170,5608	0,6252
Lx_1 Lenght_Toleran-DSTNCE	3,5157	0,5859	0,8430	0,3711	171,0598	0,4459
Lx_2 Lenght_Toleran-DSTNCE	3,4685	0,5781	0,8229	0,2388	171,2097	0,4568

With the parameters presented in Table 5.1, it is possible to obtain the errors associated with the part. The systematic error is constant in the process and occurs in the same way in all measurements made on a quantity. The causes of systematic errors are mainly attributed to defects in the manufacturing machine or the measuring instrument. Equation 5.6 is used to calculate a systematic error value. The result obtained for the systematic error is presented in Table 5.2.

$$Error_{Sis} = V_i - \bar{X}_i \tag{5.6}$$

where V_i: Desired value $\bar{X}_i$: Mean

The value for the manufacturing error is obtained using the definition of the root mean square error with the variability values of the natural process (6σ). The values obtained in the measured characteristics allow the calculation of the manufacturing error. In detecting manufacturing errors, it is also essential to determine the angle of displacement of each axis of the machine concerning the piece's characteristics. The procedure for calculating the angle of displacement is to perform a linear regression of the data from the normal distribution of the data to obtain a line that correlates with the measurement values (see appendix F.5.1). The equation obtained for each feature will describe a slight incline calculated to determine the offset angle. The data used in this procedure is the output data generated by the Mitutoyo Measurlink software. Figure 5.20 shows the graphical result thrown for the Dx_Tolerância-DSTNCE feature. The information of each characteristic that was manufactured in the disposition of a machine tool axis is extracted. Figure 5.20 shows the result of the linear regression procedure for Feature Dx_Tolerância-DSTNCE. The Table 5.2 summarizes the result for each feature analyzed; it presents the deduced equation and the angle of displacement concerning an axis of the machine. The results obtained allow making corrective maintenance decisions that lead to improving the quality of the part.

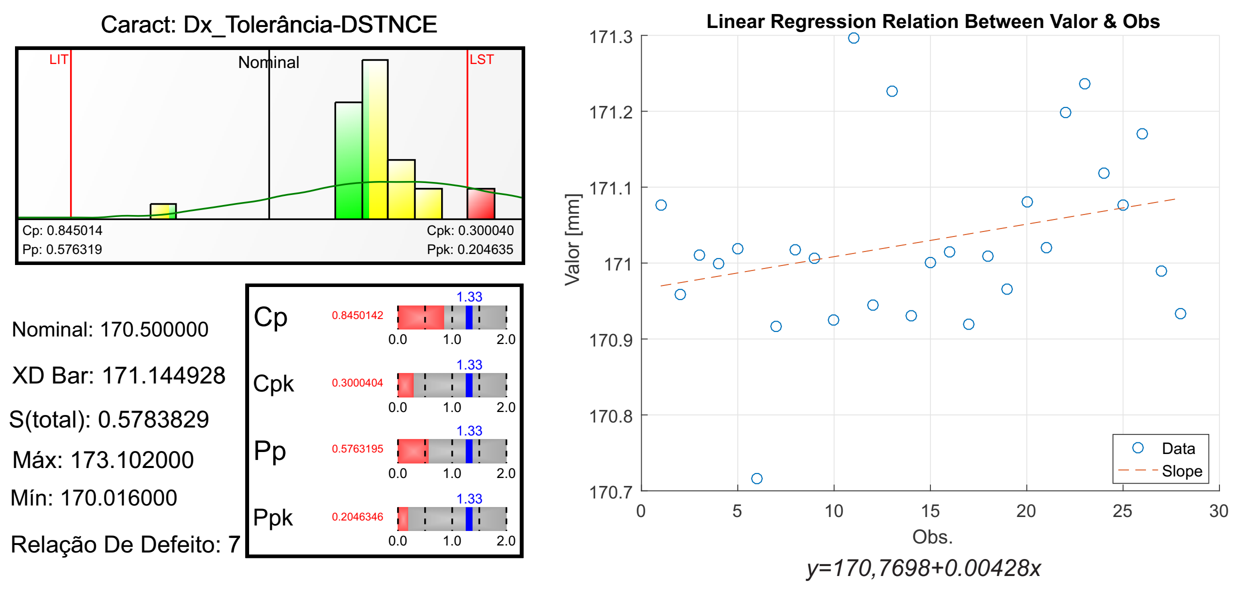


Figure 5.20: Result of the linear regression procedure for Feature Dx_Tolerância-DSTNCE

Table 5.2: Manufacturing errors found with inspection results

Indicator	$Error_{Sis}$	${\sigma_x}^2$	${\sigma_y}^2$	${\sigma_{xy}}^2$	$y = mx + b$	ϕ
Dx_Tolerância-DSTNCE	-0,6449	0,3345	–	0,5451	$y = 170,7698 + 0.00428x$	1,4818
Dy_Tolerância-DSTNCE	0,0591	–	0,2105	0,5451	$y = 170,2022 + 0,233x$	1,3381
Lx_1 Lenght_Toleran-DSTNCE	-0,5598	0,3433	–	0,7586	$y = 170,7224 + 0,0232x$	1,3329
Ly_1 Length_Toleranc-DSTNCE	-0,0233	–	0,4153	0,7586	$y = 170,8090 + 0,0276x$	1,5827
Lx_2 Lenght_Toleran-DSTNCE	-0,7097	0,3341	–	0,8911	$y = 170,1995 + 0,0223x$	1,2794
Ly_2 Lenght_Toleran-DSTNCE	-0,0608	–	0,5569	0,8911	$y = 170,2082 + 0,0243x$	1,3927

5.5 Summary

The chapter presents a computational solution adhering to the STEP standard tested in three different application contexts. In the three typical cases addressed, the neutral STEP file (p21) managed to integrate different technologies associated with the different CAx systems to transmit the data collected or generated in the process. The computational implementation presented brings together the knowledge of the standard, applied to generate a solution adhering to ISO 10303 and used within a closed-loop manufacturing architecture. The procedure described, and the case studies, allowed to test the hypothesis and a neutral data structure to overcome the integration and interoperability barriers. The methodology followed for the computational development of the application can facilitate new developments. The development environment described provides tools for manipulating the EXPRESS schemas that are the basis of all computational development.

Chapter 6

Conclusions

This thesis summarizes the study carried out on the ISO10303 standard to know the neutral, scalable data structure supporting data integration and information transfer. The current requirements that require sharing information between CAx systems can be fulfilled through STEP. Understanding the interrelation of data in the different phases of the life cycle shows that interoperability problems arise when the exchange of information is not carried out in an integrated way. The potential to use a data structure within an object-oriented perspective, the ability to reuse definitions within different application contexts without compromising data integrity shows that the standard can adapt to current and future demands. As for the developments found, they are pretty scarce but, looking at recent publications; it is possible to highlight a growing trend of interest in solutions based on the STEP standard. The main barrier encountered when studying the standard lies in the difficulty of interpreting the standard and making abstractions that can develop a rapid technological solution. However, its robust structure that bases its descriptive model on EXPRESS schemes facilitates new insertions and promotes automatic code generation; In this sense, projects associated with the STEP standard can potentially be advantageous compared to other data structures.

Updates to the STEP standard generate new EXPRESS schemes; the possibility of generating updates to the solutions lies in exploiting the potential of the EXPRESS language. Development of mechanisms to generate components based on the EXPRESS schemes that contain a conceptual description expressed syntactically and semantically, tools to create schemes that support applications that are still outside the norm are possibilities that will define the massive implementation within environments more specific.

A method was exposed to integrate data and measurement results within a closed manufacturing cycle in the present work. The technique uses neutral data structures, eliminating technological dependencies and allowing an alternative to overcome interoperability barriers that hinder feedback of measurement results. The presented solution seeks to share data with a descriptive and structural component compatible with different CAx systems. Data through the ISO10303 standard are assigned for standardized exchange, sharing the same syntax and semantics in each manufacturing cycle system.

The architecture presented is projected on the models based on definition, involving neutral STEP exchange files, and seeking to meet the information requirements of the measurement processes to integrate with technologies adhering to the perspective of Industry 4.0. The concepts of Industry 4.0 promote the integration and interoperability of systems. The model presented is designed to create an architecture that integrates both the manufacturing and inspection processes in the new digital era.

6.1 Thesis Contributions

This thesis document addresses the development of a conceptual framework for closed-loop inspection based on STEP-NC adherent coordinate measurement technology. The development methodology presented in this research allows meeting the information demands of the process, integrating the inspection results within the digital manufacturing chain, and building a closed circuit where the measurement results are the basis for making decisions that promote process improvement.

Table 6.1: Synthesis of thesis contributions

Contribution	Description	Related chapters and sections
1	Functional Model for CAD/CAPP/CAM/CAM/CHAI integration. The model contains the flow of data, activities, and relationships between the CAx systems.	Chapter 4: Conceptual Framework Proposal for Closed-Loop Inspection (AIM)
2	A conceptual model for closed-loop manufacturing and inspection. The model contains the construction fundamentals to be used as a reference in the development of other solutions adhering to the STEP standard.	Chapter 4: Conceptual Framework Proposal for Closed-Loop Inspection
3	AIM model and foundations for the transfer of requirements generated in the conceptual model in computational implementation strategies.	Chapter 4: Conceptual Framework Proposal for Closed-Loop Inspection
4	A Methodology for the development of computational solutions. The methodology includes specification, verification, and implementation of conceptual models adhering to the STEP standard.	Chapter 4: Conceptual Framework Proposal for Closed-Loop Inspection, Chapter 5: The STEP Development and Deployment Environment
5	Programming architecture with an object-oriented perspective provides libraries created for the classes related to the conceptual model coded in self-describing language for academic purposes. This architecture seeks to promote new developments on this line of research.	Chapter 5: The STEP Development and Deployment Environment
6	A interoperable inspection and manufacturing closed-loop integration architecture, with strategies for error compensation.	Chapter 5: The STEP Development and Deployment Environment

The main contributions are summarized in the Table 6.1 and detailed below:

1. **Functional Model for CAD/CAPP/CAIP/CAM / CAI integration. Describes data flow, activities, and relationships between CAx systems**: The conceptual model is the product of a literature review carried out to identify solutions for the integration and interoperability of CAx systems. In this review process, the main variables that interact within a closed manufacturing loop were identified, and a closed inspection loop was designed. The functional model describes and ranks the

activities through blocks that offer a high-level view of the project's scope. Emphasis is placed in this model on using available neutral data structures for information exchange that meet digital technological requirements and demands. This Functional model covers the entire integration aspect of closed-loop manufacturing and inspection.

2. **A conceptual model for the integration of CAx systems and construction foundations for reference in the development of other solutions adhering to the STEP standard**: The conceptual model is presented as a methodology that allows describing in a more specific way how the integration of data and information happens in the closed-loop architecture of manufacturing and inspection. The STEP standard is adopted as a means of exchange, which through the description method in EXPRESS language, manages to provide details of the data flow, coding structure, and architecture for integration in all phases of the product life cycle. This conceptual model is presented as a contribution because it is not documented as a particular solution to the context addressed but is detailed in a generalized way to facilitate abstraction to other application contexts projected adherents to STEP.

3. **AIM model and foundations for the transfer of requirements generated in the conceptual model in computational implementation strategies**: The AIM model enables computational implementations. It is a complex model and requires specialized knowledge in software development for creation in greater detail than in many situations. The necessary foundations to build this model are documented; an example of transferring the specifications given in the conceptual model to the AIM model is presented. Using built-in resources and application builders allows connecting the conceptual model with programming languages and structures. The perspective adopted by the standard is object-oriented, where hierarchies due to inheritance and complex entities are interrelated. This development cycle is presented starting from a high level until reaching instructions, classes, and instances that support the flow of information.

4. **Methodology for developing computational solutions with specification, verification, and implementation of conceptual models adhering to the STEP standard**: A viable development environment is presented to build any solution adhering to the STEP standard. Tools and requirements already proven through the case studies are described to promote particular solutions within this line. It is presented as a methodology since it was proven, and due to the state of art, very few works are developed for independent application contexts. The implementation methodology is current, scalable, and allows for incremental improvements; Important issues when we know that the ISO10303 standard is constantly evolving.

5. **Programming architecture within an object-oriented perspective with libraries created for the classes related to the conceptual model in self-descriptive language for academic purposes that promote new developments on this line**: A solution with a textual interface and development-oriented to unit tests was created to verify the methodology. It required building libraries and classes for the entities used by different functional units of the application protocols. The inheritance perspective of the STEP standard allows an instance to be used in different application contexts; in other words, libraries can be helpful for new developments since some of the UoF, and their relationship structure is implemented. They are exposed as academic contributions since the code created is self-descriptive and can be used to train and tea ch disciplines that lead to new solutions.

6. **Interoperable closed-loop integration architecture, with strategies for error compensation**: The architecture presented allows inheriting concepts of manufacturing error control and statistical process helpful control for the closed-loop of manufacturing and inspection. Two typical example cases are presented: when want to start manufacturing from a detailed analysis of a part and another when want to correct errors in the middle of the manufacturing cycle. New error control techniques can be developed in this control context.

This research with a description of procedures seeks to promote technological solutions based on exchange-neutral files that guarantee integration and interoperability. The possibilities of implementing the STEP standard for specific applications, such as integration of manufacturing data with inspection, were evaluated. A method was developed for the design and implementation of computational solutions adhering to the STEP standard. The solution obtained is specific and of particular application within a manufacturing context, but it allows knowing and interpreting the standard to develop new solutions that increase functionalities.

6.2 Recommendations for Future Work

Carrying out this research has allowed the identification of several lines of research that can generate meaningful solutions within the digital career faced by the manufacturing and inspection processes. The data structure provided by the ISO10303 standard allows for meeting various requirements in this race to achieve integration and interoperability in the processes. Some systems currently in operation will require interpreters to be linked in this digital age; new developments are required to process, analyze and make decisions with the data generated. The knowledge extracted from the product life cycle will play an essential role in optimizing the manufacturing and inspection processes since the balance between quality and productivity can be increasingly efficient.

The data structures of the STEP standard maintain their connection with the descriptive language EXPRESS that is constantly evolving; it is frequently nurtured with new definitions and generates significant updates. This EXPRESS language describes the definitions and defines the structure to transfer the data within a context adhering to the standard. Solutions for automatic code generation based on EXPRESS schemes will facilitate definitions of functional implementations. The importance may lie in promoting faster solutions, which can be easily updated, promoting the maintainability of solutions that are obsolete with new standard technological updates.

Inspection processes take relevance within the digital context, especially in the construction of knowledge; technological advances in measurement systems seek to automate the process, reduce human participation and errors produced in the measurement phase. The closed-loop inspection strategy requires solutions focused on maintaining interoperability, interpreting results, associating manufacturing errors with process causes, establishing automatic error compensation strategies, and optimizing the measurement process.

The inspection process requires even more significant contributions that support the execution of inspection processes, representation of measurement results that allow covering and linking the different measurement alternatives outside the conventional ones. The planning of the inspection process can also change due to the nature or principle of measurement. Solutions based on the neutral data structure of STEP will allow to link results automatically and transmit them to the different phases of the product life cycle, where they can be helpful in the improvement of the process.

Mechanical structures with particular configurations that provide better performance in manufacturing and inspection can be developed to interpret neutral exchange files. Automatic recognition of Features, generation of optimal trajectories, kinematic control, and configuration of process variables can be developed in the not too distant future to improve the process efficiency. Structures with hybrid manufacturing or that direct link inspection within manufacturing can take advantage of these developments.

The active participation of the manufacturing sector has generated significant contributions in constructing the knowledge contained in the application protocols. This participation leverages the development and its massive distribution of the data structure. The ISO 10303 standard is being assumed within the manufacturing context, solutions focused on maintaining technology-independent interoperability and data integration with the consistency of information would be well-received by the commercial sector.

Several neutral exchange formats are emerging, some focused on monitoring and capturing process variables, others specialized in communication. Expert systems that manage to process the amount of data generated in the manufacturing environment and generate knowledge for better productivity without forgetting quality can be developed based on intelligent techniques. These systems endowed with learning capabilities, self-diagnostics, self-correction of errors, and optimization of resources will completely transform the concept of manufacturing as we know it today. The neutral data structure provided by STEP can be the natural element to extract information.

REFERENCES

ALI, L.; NEWMAN, S. T.; PETZING, J. Development of a step-compliant inspection framework for discrete components. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Sage Publications Sage UK: London, England, v. 219, n. 7, p. 557–563, 2005.

ALVARES, A. *Uma metodologia para integração CAD/CAPP/CAM voltada para manufatura remota de peças rotacionais baseada na internet, 2005, 249f.* Tese (Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2005.

ALVARES, A. J. *A tecnologia de medição por coordenadas como base para a regulação da qualidade geométrica do processo de usinagem CNC*. Dissertação (Mestrado) — Universidade federal de santa Catarina, 1990.

ANJANAPPA, M.; DICKSTEIN, J.; ANAND, D.; KIRK, J. Automated inspection data analyzer for closed loop manufacturing. In: *Proceedings of Manufacturing International*. [S.l.: s.n.], 1990. v. 90, p. 17–25.

ANJANAPPA, M.; DICKSTEIN, J.; SUBBACHARYA, M. Computer-aided inspection data analyser. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 12, n. 2, p. 93–102, 1996.

ASME Y14.5M. Dimensioning and tolerancing. 1994. Disponível em: <https://www.asme.org/products/codes-standards/>.

BAGSHAW, R.; NEWMAN, S. Quality information feedback within a contemporary metalworking smaller manufacturing enterprise. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Sage Publications Sage UK: London, England, v. 213, n. 5, p. 533–538, 1999.

BAGSHAW, R.; NEWMAN, S. Manufacturing data analysis of machine tool errors within a contemporary small manufacturing enterprise. *International Journal of Machine Tools and Manufacture*, v. 42, n. 9, p. 1065 – 1080, 2002. ISSN 0890-6955. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0890695502000275>.

BENAVENTE, J. C. T. *Um sistema para o projeto e fabricação de peças mecânicas a distância via internet aderente à norma ISO 14649 (STEP-NC)*. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2011.

BERNSTEIN, W. Z.; JR, T. D. H.; HELU, M.; FEENEY, A. B. Contextualising manufacturing data for lifecycle decision-making. *International Journal of Product Lifecycle Management*, Inderscience Publishers (IEL), v. 10, n. 4, p. 326–347, 2017.

BHANDARKAR, M. P.; NAGI, R. Step-based feature extraction from step geometry for agile manufacturing. *Computers in industry*, Elsevier, v. 41, n. 1, p. 3–24, 2000.

BORTOLINI, M.; GALIZIA, F. G.; MORA, C. Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems*, v. 49, p. 93 – 106, 2018. ISSN 0278-6125.

BRECHER, C.; VITR, M.; WOLF, J. Closed-loop capp/cam/cnc process chain based on stepand step-nc inspection tasks. *International Journal of Computer Integrated Manufacturing*, Taylor & Francis, v. 19, n. 6, p. 570–580, 2006.

BRODSKY, A.; SHAO, G.; KRISHNAMOORTHY, M.; NARAYANAN, A.; MENASCÉ, D.; AK, R. Analysis and optimization in smart manufacturing based on a reusable knowledge base for process performance models. In: IEEE. *Big Data (Big Data), 2015 IEEE International Conference on*. [S.l.], 2015. p. 1418–1427.

BRUNNERMEIER, S. B.; MARTIN, S. A. *Interoperability cost analysis of the US automotive supply chain*. [S.l.]: DIANE Publishing, 1999.

CAMPOS, J. G.; HARDWICK, M. A traceability information model for cnc manufacturing. *Computer-Aided Design*, Elsevier, v. 38, n. 5, p. 540–551, 2006.

DANJOU, C. *Ingénierie de la chaîne numérique d'industrialisation: proposition d'un modèle d'interopérabilité pour la conception-fabrication intégrées*. Tese (Doutorado) — Compiègne, 2015.

DAVIS, J.; EDGAR, T.; PORTER, J.; BERNADEN, J.; SARLI, M. Smart manufacturing, manufacturing intelligence and demand-dynamic performance. *Computers & Chemical Engineering*, v. 47, p. 145 – 156, 2012. ISSN 0098-1354. FOCAPO 2012.

DMSC. Quality information framework (qif) – an integrated model for manufacturing quality information. 2015. Disponível em: <http://qifstandards.org>.

FENG, S. C.; YANG, Y. A dimension and tolerance data model for concurrent design and systems integration. *Journal of Manufacturing systems*, Elsevier, v. 14, n. 6, p. 406–426, 1995.

GALLAHER, M. P.; O'CONNOR, A. C.; PHELPS, T. Economic impact assessment of the international standard for the exchange of product model data (step) in transportation equipment industries. *NIST planning report*, p. 02–5, 2002.

GHANTA, S. *A STEP Implementation for Product Data Exchange: DT005A*. 2012.

GUNASEKARAN, A.; VIRTANEN, I.; MARTIKAINEN, T.; YLI-OLLI, P. The design of computer-integrated manufacturing systems. *International Journal of Production Economics*, Elsevier, v. 34, n. 3, p. 313–327, 1994.

HANDBOOK, S. A. *ISO 10303, Version 3*. [S.l.]: SCRA, 2006.

HARDWICK, M. Digital twin machining. 2017. Accedido 14-05-2018. Disponível em: <https://www.steptools.com/blog/20171011_twin_machining/>.

HARDWICK, M.; ZHAO, Y. F.; PROCTOR, F. M.; NASSEHI, A.; XU, X.; VENKATESH, S.; ODENDAHL, D.; XU, L.; HEDLIND, M.; LUNDGREN, M. et al. A roadmap for step-nc-enabled interoperable manufacturing. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 68, n. 5-8, p. 1023–1037, 2013.

HARDWICK, S. T. M. Operate orchestrate and originate. In: *Digital Manufacturing Advisory Group*. [s.n.], 2016. Disponível em: <https://www.nist.gov/sites/default/files/documents/el/msid/12_MHardwick-DMDII-O3-3.pdf>.

HEDBERG, T.; LUBELL, J.; FISCHER, L.; MAGGIANO, L.; FEENEY, A. B. Testing the digital thread in support of model-based manufacturing and inspection. *Journal of computing and information science in engineering*, American Society of Mechanical Engineers, v. 16, n. 2, p. 021001, 2016.

HEDBERG, T. D.; HELU, M. M. *Design and Configuration of the Smart Manufacturing Systems Test Bed*. [S.l.], 2017.

HOCKEN, R. J.; PEREIRA, P. H. *Coordinate measuring machines and systems*. [S.l.]: CRC Press, 2016.

HORSFALL, S. *Step by Step DMIS Programming*. Instant Publisher, 2007. ISBN 9781604580600. Disponível em: <https://books.google.com.br/books?id=orOGHwAACAAJ>.

HORST, J. A.; MCSPADDEN, S. *A Roadmap For Metrology Interoperability*. [S.l.], 2006.

IGES, A. Initial graphics exchange specification version 6.0. *ANSI Standard*, 2001.

ISO 10303-11. Description methods: The express language reference manual. 2004. Disponível em: <https://www.iso.org/standard/38047.html>.

ISO 10303-203. Industrial automation systems and integration – product data representation and exchange – part 203: Application protocol: Configuration controlled 3d design of mechanical parts and assemblies. 2011. Disponível em: <https://www.iso.org/standard/44305.html>.

ISO 10303-214. Industrial automation systems and integration – product data representation and exchange – part 214: Application protocol: Core data for automotive mechanical design processes. 2010. Disponível em: <https://www.iso.org/standard/43669.html>.

ISO 10303-219. Industrial automation systems and integration – product data representation and exchange – part 219: Application protocol: Dimensional inspection information exchange. 2007. Disponível em: <https://www.iso.org/standard/38722.html>.

ISO 10303-238. Industrial automation systems and integration – product data representation and exchange – part 238: Application protocol: Application interpreted model for computerized numerical controllers. 2007. Disponível em: <https://www.iso.org/standard/38036.html>.

ISO 10303-240. Industrial automation systems and integration — product data representation and exchange — part 240: Application protocol: Process plans for machined products. 2005. Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso:10303:-240:ed-1:v1:enl>.

ISO 10303-242. Industrial automation systems and integration – product data representation and exchange – part 242: Application protocol: Managed model-based 3d engineering. 2014. Disponível em: <https://www.iso.org/standard/57620.html>.

ISO 10303-28. Product data representation and exchange: Implementation methods: Xml representation of express schemas and data. 2007. Disponível em: <https://www.iso.org/standard/40646.html>.

ISO 10303-41. Industrial automation systems and integration — product data representation and exchange — part 41: Integrated generic resource: Fundamentals of product description and support. 2018. Disponível em: <https://www.iso.org/standard/76127.html>.

ISO 17450-2. Geometrical product specifications (gps) – general concepts – part 2: Basic tenets, specifications, operators, uncertainties and ambiguities. 2012. Disponível em: <https://www.iso.org/standard/53629.html>.

ISO 22093:2011, Industrial automation systems and integration – Physical device control – Dimensional Measuring Interface Standard (DMIS). 2011. Disponível em: <https://www.iso.org/standard/56444.html>.

JAIMES, C.; BONNARD, R.; FERREIRA, J.; ALVARES, A.; JAIMES, C. Closed loop integration model for dimensional and geometric inspection of prismatic parts based on the step-nc standard. In: *24th ABCM international congress of mechanical engineering*. [S.l.: s.n.], 2018.

KERN, V. M. et al. Manutenibilidade da semântica de modelos de dados de produtos compartilhados em rede interoperável. Florianópolis, SC, 1997.

KERN, V. M. et al. Manutenibilidade da semântica de modelos de dados de produtos compartilhados em rede interoperável. Florianópolis, SC, 1997.

KOLOSOWSKI, M.; DUDA, J.; TOMASIAK, J. Statistical process control in conditions of piece and small lot production. *Annals of DAAAM & Proceedings*, v. 26, n. 1, 2015.

LASI, H.; FETTKE, P.; KEMPER, H.-G.; FELD, T.; HOFFMANN, M. Industry 4.0. *Business & Information Systems Engineering*, Springer, v. 6, n. 4, p. 239–242, 2014.

LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, Elsevier, v. 3, p. 18–23, 2015.

LEE, J.; KAO, H.-A.; YANG, S. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp*, Elsevier, v. 16, p. 3–8, 2014.

LI, Y. *Development architecture for industrial data management*. Tese (Doutorado) — KTH Royal Institute of Technology, 2013.

LI, Y. *Development architecture for industrial data management*. Tese (Doutorado) — KTH Royal Institute of Technology, 2013.

LI, Y.; HEDLIND, M.; KJELLBERG, T. Implementation of kinematic mechanism data exchange based on step. In: *CIRP 7th International Conference on Digital Enterprise Technology. Athens, Greece*. [S.l.: s.n.], 2011. p. 152–159.

LKSOFT. Jsdai overview jsdai reference documentation. 1999. Disponível em: <http://www.jsdai.net/overview>.

LU, Y.; LIU, C.; WANG, K. I.-K.; HUANG, H.; XU, X. Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, v. 61, p. 101837, 2020. ISSN 0736-5845. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0736584519302480>.

LU, Y.; MORRIS, K. C.; FRECHETTE, S. Standards landscape and directions for smart manufacturing systems. In: IEEE. *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. [S.l.], 2015. p. 998–1005.

LU, Y.; XU, X. Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services. *Robotics and Computer-Integrated Manufacturing*, v. 57, p. 92 – 102, 2019. ISSN 0736-5845.

MAHMOUD, H. *Resource-Independent Computer Aided Inspection*. Tese (Doutorado) — University of Bath, 2016.

MAHMOUD, H.; DHOKIA, V.; NASSEHI, A. Step-based conceptual framework for measurement planning integration. *Procedia CIRP*, v. 43, p. 315 – 320, 2016. ISSN 2212-8271. 14th CIRP CAT 2016 - CIRP Conference on Computer Aided Tolerancing. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2212827116003164>.

MEDLAND, A.; MULLINEUX, G. A constraint approach to feature-based design. *International Journal of Computer Integrated Manufacturing*, Taylor & Francis, v. 6, n. 1-2, p. 34–38, 1993.

MORONI, G.; PETRÒ, S. Geometric inspection planning as a key element in industry 4.0. In: SPRINGER. *International Conference on the Industry 4.0 model for Advanced Manufacturing*. [S.l.], 2018. p. 293–310.

MORSE, E.; HEYSIATTALAB, S.; BARNARD-FEENEY, A.; JR, T. H. Interoperability: linking design and tolerancing with metrology. *Procedia CIRP*, Elsevier, v. 43, p. 13–16, 2016.

NASR, E. A.; ABDULHAMEED, O.; AL-AHMARI, A. M. *Computer-Aided Inspection Planning: Theory and Practice*. [S.l.]: Crc Press, 2016.

NEWMAN, S.; NASSEHI, A.; XU, X.; ROSSO, R.; WANG, L.; YUSOF, Y.; ALI, L.; LIU, R.; ZHENG, L.; KUMAR, S. et al. Strategic advantages of interoperability for global manufacturing using cnc technology. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 24, n. 6, p. 699–708, 2008.

OLIVEIRA, L. E. S. d. Concepção de um framework para monitoramento e teleoperação de máquinas-ferramenta cnc via internet aderente à indústria 4.0. 2017.

PAN, C.; SMITH, S. S.-F.; SMITH, G. C. Determining interference between parts in cad step files for automatic assembly planning. *Journal of Computing and Information Science in Engineering*, American Society of Mechanical Engineers, v. 5, n. 1, p. 56–62, 2005.

PEAK, R. S.; LUBELL, J.; SRINIVASAN, V.; WATERBURY, S. C. Step, xml, and uml: complementary technologies. *Journal of Computing and Information Science in Engineering*, American Society of Mechanical Engineers, v. 4, n. 4, p. 379–390, 2004.

PLANTEC, A. *Outils et méthodes de la norme STEP (Standard ISO 10303)*. [S.l.]: Lab, 2007.

PRATT, M. J. Introduction to iso 10303—the step standard for product data exchange. *Journal of Computing and Information Science in Engineering*, American Society of Mechanical Engineers, v. 1, n. 1, p. 102–103, 2001.

QIN, Y.; LU, W.; LIU, X.; HUANG, M.; ZHOU, L.; JIANG, X. Description logic-based automatic generation of geometric tolerance zones. *The International Journal of Advanced Manufacturing Technology*, v. 79, n. 5, p. 1221–1237, Jul 2015. ISSN 1433-3015.

QIN, Y.; QI, Q.; LU, W.; LIU, X.; SCOTT PAUL J.AND JIANG, X. A review of representation models of tolerance information. *The International Journal of Advanced Manufacturing Technology*, Nov 2017. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-017-1352-4>.

RAY, S. R.; JONES, A. T. Manufacturing interoperability. *Journal of Intelligent Manufacturing*, v. 17, n. 6, p. 681–688, Dec 2006. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-006-0037-x>.

RENTOUL, A.; MULLINEUX, G.; MEDLAND, A. Interpretation of errors from inspection results. *Computer Integrated Manufacturing Systems*, v. 7, n. 3, p. 173 – 178, 1994. ISSN 0951-5240. Disponível em: <http://www.sciencedirect.com/science/article/pii/0951524094900361>.

RIAÑO, C. I.; ÁLVARES, A. J. Feedback strategy for closed-loop inspection based on STEP-NC. *Journal of Physics: Conference Series*, IOP Publishing, v. 1065, n. 8, p. 082014, aug 2018. Disponível em: <https://doi.org/10.1088/1742-6596/1065/8/082014>.

RIAÑO, C. I.; ESPINDOLA, J. C.; BENAVENTE, J.; ALVARES, A. J. Integration model for closed-loop inspection based on step-nc (original title in portuguese). In: *9o Congresso Brasileiro de Engenharia de Fabricação*. [S.l.: s.n.], 2017.

Riaño Jaimes, C. I.; ALVARES, A. J. Integrated inspection system step-compliant for the exchange of dimensional metrology data. *Procedia Manufacturing*, v. 38, p. 1205–1212, 2019. ISSN 2351-9789. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2351978920302122>.

RIPPEY, W. Aiag demonstrates metrology interoperability: To save you time and money. In: *International Dimensional Workshop (IDW 2005)*. [S.l.: s.n.], 2005.

RUEMLER, S. P.; ZIMMERMAN, K. E.; HARTMAN, N. W.; HEDBERG, T.; FEENY, A. B. Promoting model-based definition to establish a complete product definition. *Journal of manufacturing science and engineering*, American Society of Mechanical Engineers, v. 139, n. 5, p. 051008, 2017.

SHAH, J. J.; YAN, Y.; ZHANG, B.-C. Dimension and tolerance modeling and transformations in feature based design and manufacturing. *Journal of Intelligent Manufacturing*, v. 9, n. 5, p. 475–488, Oct 1998. ISSN 1572-8145. Disponível em: <https://doi.org/10.1023/A:1008856818686>.

SHIN, S.-J.; WOO, J.; KIM, D. B.; KUMARAGURU, S.; RACHURI, S. Developing a virtual machining model to generate mtconnect machine-monitoring data from step-nc. *International Journal of Production Research*, Taylor & Francis, v. 54, n. 15, p. 4487–4505, 2016.

SOBEL, W. *MTConnect Standard. Part 1 - Overview and Protocol*. Version 1.3.0. [S.l.], 2014.

SOUZA, F. J.; FERREIRA, J. C. E.; MARTIN, C. A.; GASCHO, W. F. Remote machining of prismatic parts through the internet in a cnc machine compliant with the step-nc standard. In: *23rd ABCM International Congress of Mechanical Engineering-Rio de Janeiro, Brazil*. [S.l.: s.n.], 2015. p. 6–11.

VEEN, G. van der; LANGELAAR, M.; MEULEN, S. V. D.; LARO, D.; AANGENENT, W.; KEULEN, F. van. Integrating topology optimization in precision motion system design for optimal closed-loop control performance. *Mechatronics*, Elsevier, v. 47, p. 1–13, 2017.

VIJAYARAGHAVAN, A.; SOBEL, W.; FOX, A.; DORNFELD, D.; WARNDORF, P. Improving machine tool interoperability using standardized interface protocols: Mt connect. 2008.

XIA, R.-x.; LU, R.-s.; SHI, Y.-q.; LI, Q.; DONG, J.-t.; LIU, N. Caip system for vision-based on-machine measurement. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Seventh International Symposium on Precision Engineering Measurements and Instrumentation*. [S.l.], 2011. v. 8321, p. 83213V.

XU, X. *Integrating advanced computer-aided design, manufacturing, and numerical control: principles and implementations*. [S.l.]: Information Science Reference-Imprint of: IGI Publishing, 2009.

XU, X.; NEE, A. Y. C. *Advanced design and manufacturing based on STEP*. [S.l.]: Springer Science & Business Media, 2009.

XU, X.; NEWMAN, S. T. Making cnc machine tools more open, interoperable and intelligent—a review of the technologies. *Computers in Industry*, Elsevier, v. 57, n. 2, p. 141–152, 2006.

XU*, X.; WANG, H.; MAO, J.; NEWMAN, S.; KRAMER, T.; PROCTOR, F.; MICHALOSKI, J. Step-compliant nc research: the search for intelligent cad/capp/cam/cnc integration. *International Journal of Production Research*, Taylor & Francis, v. 43, n. 17, p. 3703–3743, 2005.

XU*, X.; WANG, H.; MAO, J.; NEWMAN, S.; KRAMER, T.; PROCTOR, F.; MICHALOSKI, J. Step-compliant nc research: the search for intelligent cad/capp/cam/cnc integration. *International Journal of Production Research*, Taylor & Francis, v. 43, n. 17, p. 3703–3743, 2005.

XU, X. W.; WANG, L.; RONG, Y. Step-nc and function blocks for interoperable manufacturing. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 3, n. 3, p. 297–308, 2006.

YANDAYAN, T.; BURDEKIN, M. In-process dimensional measurement and control of workpiece accuracy. *International Journal of Machine Tools and Manufacture*, Elsevier, v. 37, n. 10, p. 1423–1439, 1997.

ZEID, I. *CAD/CAM theory and practice*. [S.l.]: McGraw-Hill Higher Education, 1991.

ZHAO, F.; XU, X.; XIE, S. Step-nc enabled on-line inspection in support of closed-loop machining. *Robotics and Computer-Integrated Manufacturing*, v. 24, n. 2, p. 200 – 216, 2008. ISSN 0736-5845. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0736584506001141>.

ZHAO, F.; XU, X.; XIE, S. Step-nc enabled on-line inspection in support of closed-loop machining. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 24, n. 2, p. 200–216, 2008.

ZHAO, X.; PASUPATHY, T. K.; WILHELM, R. G. Modeling and representation of geometric tolerances information in integrated measurement processes. *Computers in Industry*, v. 57, n. 4, p. 319 – 330, 2006. ISSN 0166-3615. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166361505001442>.

ZHAO, Y.; KRAMER, T.; BROWN, R.; XU, X. *Information modeling for interoperable dimensional metrology*. [S.l.]: Springer Science & Business Media, 2011.

ZHAO, Y. F.; HORST, J. A.; KRAMER, T. R.; RIPPEY, W.; BROWN, R. J. Quality information framework–integrating metrology processes. *IFAC Proceedings Volumes*, Elsevier, v. 45, n. 6, p. 1301–1308, 2012.

ZHAO, Y. F.; PROCTOR, F. M.; HORST, J. A. *A Machining and Measurement Process Planning Activity Model for Manufacturing System Interoperability Analysis*. [S.l.], 2010.

ZHAO, Y. F.; PROCTOR, F. M.; HORST, J. A.; XU, X. Merging machining and measurement for cognitive manufacturing. In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. [S.l.: s.n.], 2010. v. 44113, p. 353–361.

ZHENG, P.; SANG, Z.; ZHONG, R. Y.; LIU, Y.; LIU, C.; MUBAROK, K.; YU, S.; XU, X. et al. Smart manufacturing systems for industry 4.0: Conceptual framework, scenarios, and future perspectives. *Frontiers of Mechanical Engineering*, Springer, p. 1–14, 2018.

ZHOU, E.; LINK, D.; HARRISON, D. An artificial intelligence system for estimating and compensating manufacturing process errors in cnc machining. In: SPRINGER. *Proceedings of the Thirty-First International Matador Conference*. [S.l.], 1995. p. 321–326.

ŽIVANOVIĆ, S.; GLAVONJIĆ, M. Methodology for implementation scenarios for applying protocol step-nc. *Journal of Production Engineering*, v. 17, n. 1, p. 71–74, 2014.

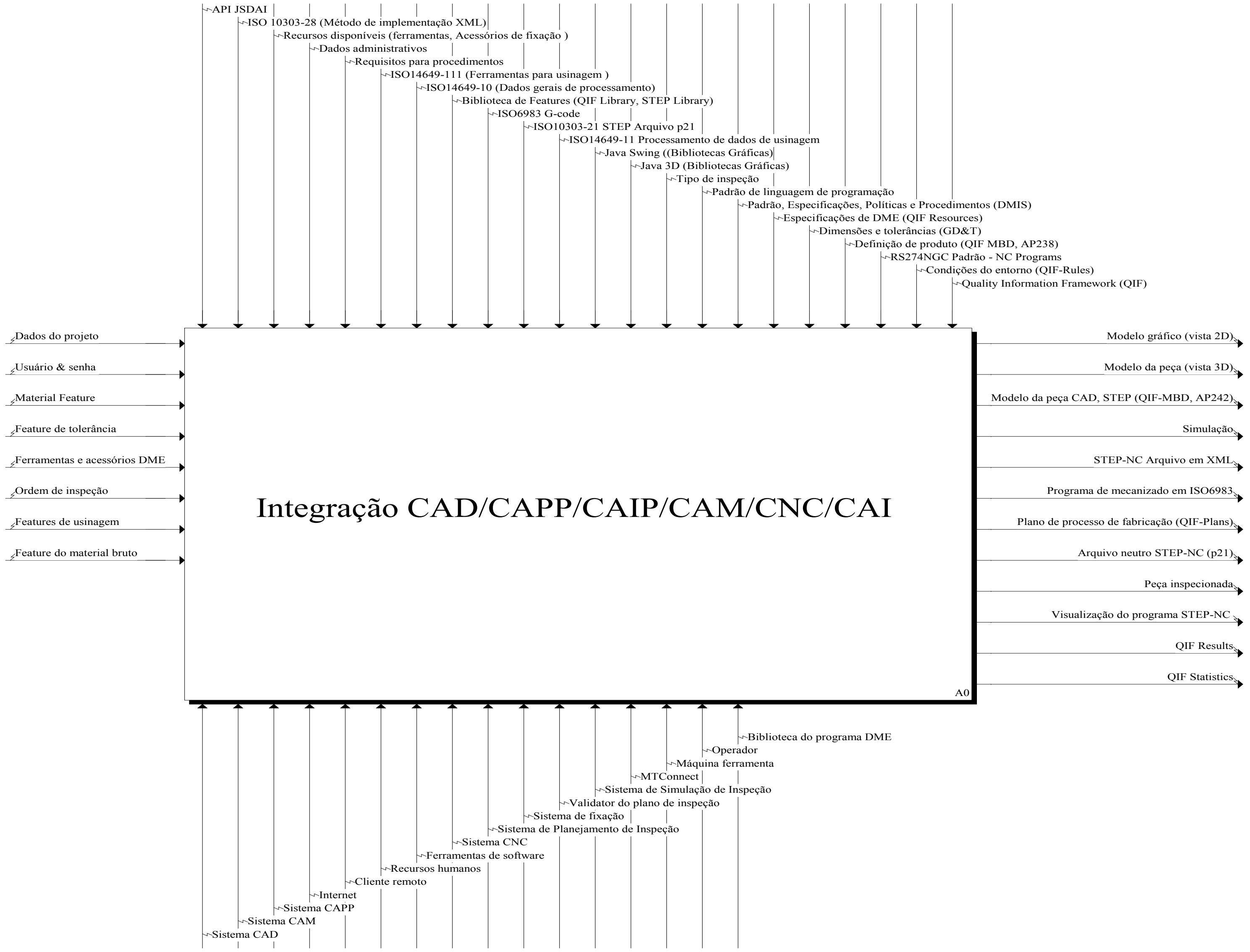
ÁLVARES, A. J.; RODRIGUEZ, E.; JAIMES, C. I. R.; TOQUICA, J. S.; FERREIRA, J. C. E. Step-nc architectures for industrial robotic machining: Review, implementation and validation. *IEEE Access*, v. 8, p. 152592–152610, 2020.

APPENDIX

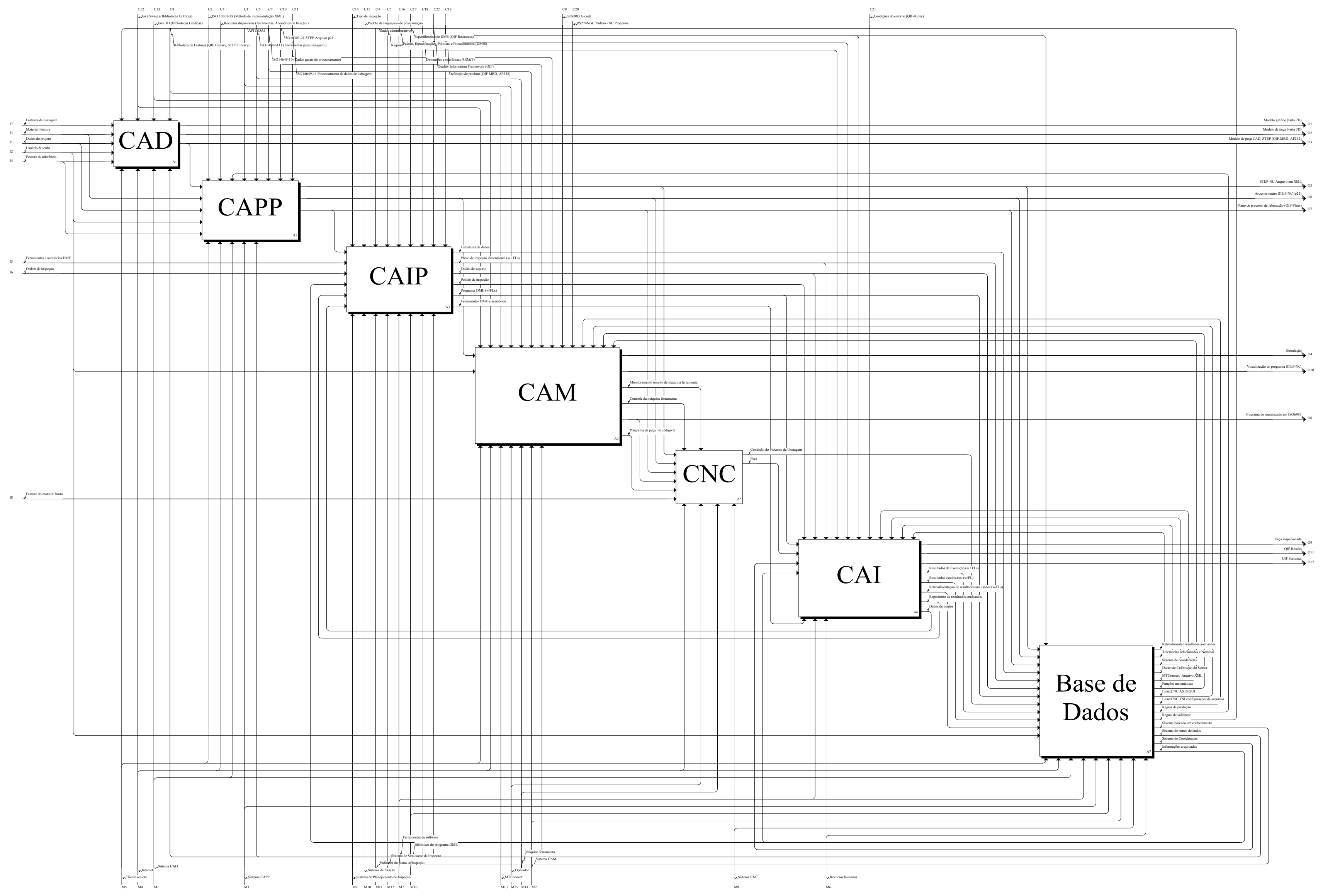
A.1 Appendix: AAM IDEF0 Model for CAx Systems Integration

A.1.1 Closed-loop Integration Model

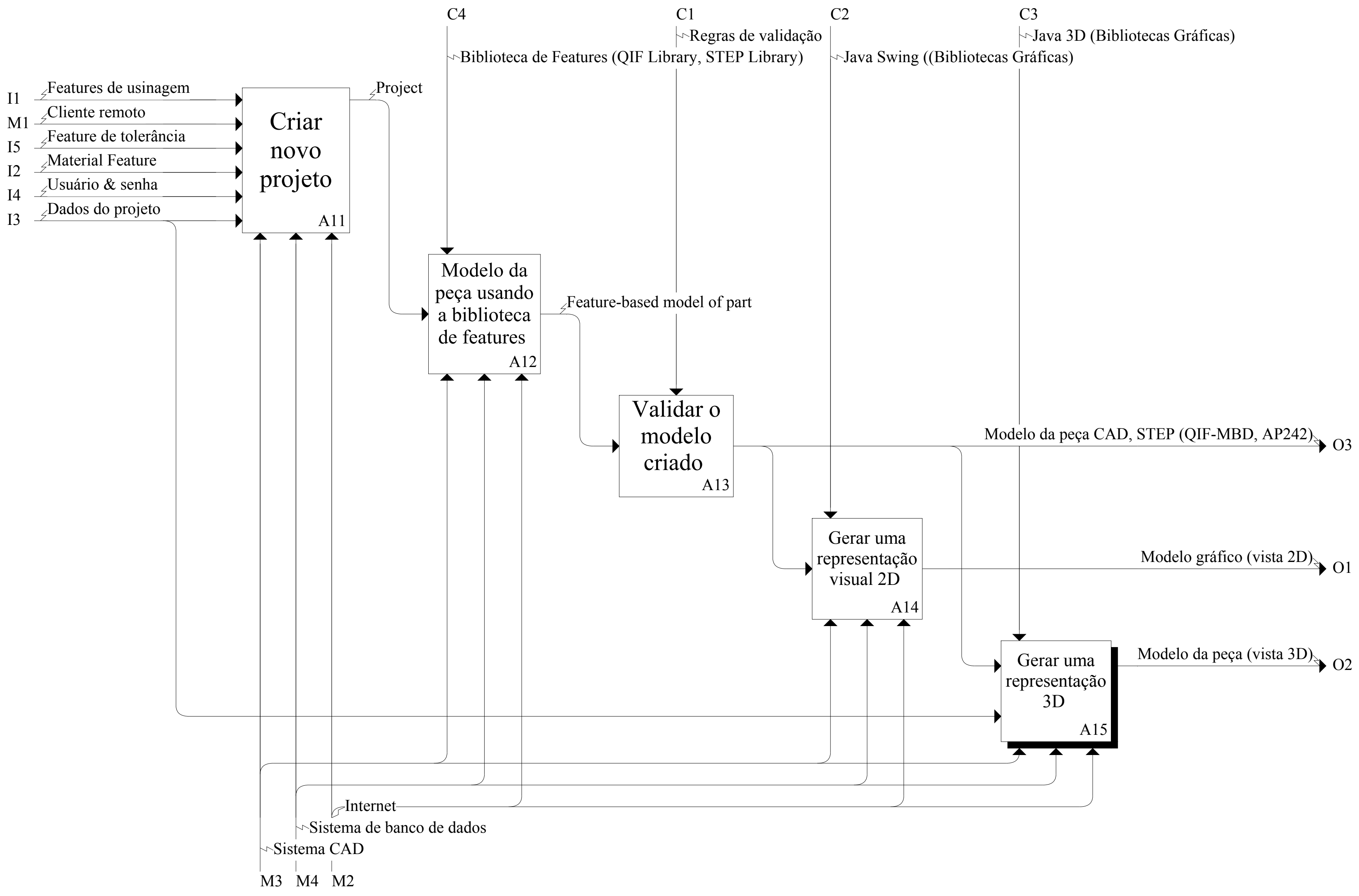
Modelo de integração em malha fechada


API JSDAI
ISO 10303-28 (Método de implementação XML)
Recursos disponíveis (ferramentas, Acessórios de fixação)
Dados administrativos
Requisitos para procedimentos
ISO14649-111 (Ferramentas para usinagem)
ISO14649-10 (Dados gerais de processamento)
Biblioteca de Features (QIF Library, STEP Library)
ISO6983 G-code
ISO10303-21 STEP Arquivo p21
ISO14649-11 Processamento de dados de usinagem
Java Swing ((Bibliotecas Gráficas)
Java 3D (Bibliotecas Gráficas)
Tipo de inspeção
Padrão de linguagem de programação
Padrão, Especificações, Políticas e Procedimentos (DMIS)
Especificações de DME (QIF Resources)
Dimensões e tolerâncias (GD&T)
Definição de produto (QIF MBD, AP238)
RS274NGC Padrão - NC Programs
Condições do entorno (QIF-Rules)
Quality Information Framework (QIF)
Dados do projeto
Usuário & senha
Material Feature
Feature de tolerância
Ferramentas e acessórios DME
Ordem de inspeção
Features de usinagem
Feature do material bruto
Integração CAD/CAPP/CAIP/CAM/CNC/CAI
A0
Modelo gráfico (vista 2D)
Modelo da peça (vista 3D)
Modelo da peça CAD, STEP (QIF-MBD, AP242)
Simulação
STEP-NC Arquivo em XML
Programa de mecanizado em ISO6983
Plano de processo de fabricação (QIF-Plans)
Arquivo neutro STEP-NC (p21)
Peça inspecionada
Visualização do programa STEP-NC
QIF Results
QIF Statistics
Biblioteca do programa DME
Operador
Máquina ferramenta
MTConnect
Sistema de Simulação de Inspeção
Validator do plano de inspeção
Sistema de fixação
Sistema de Planejamento de Inspeção
Sistema CNC
Ferramentas de software
Recursos humanos
Cliente remoto
Internet
Sistema CAPP
Sistema CAM
Sistema CAD

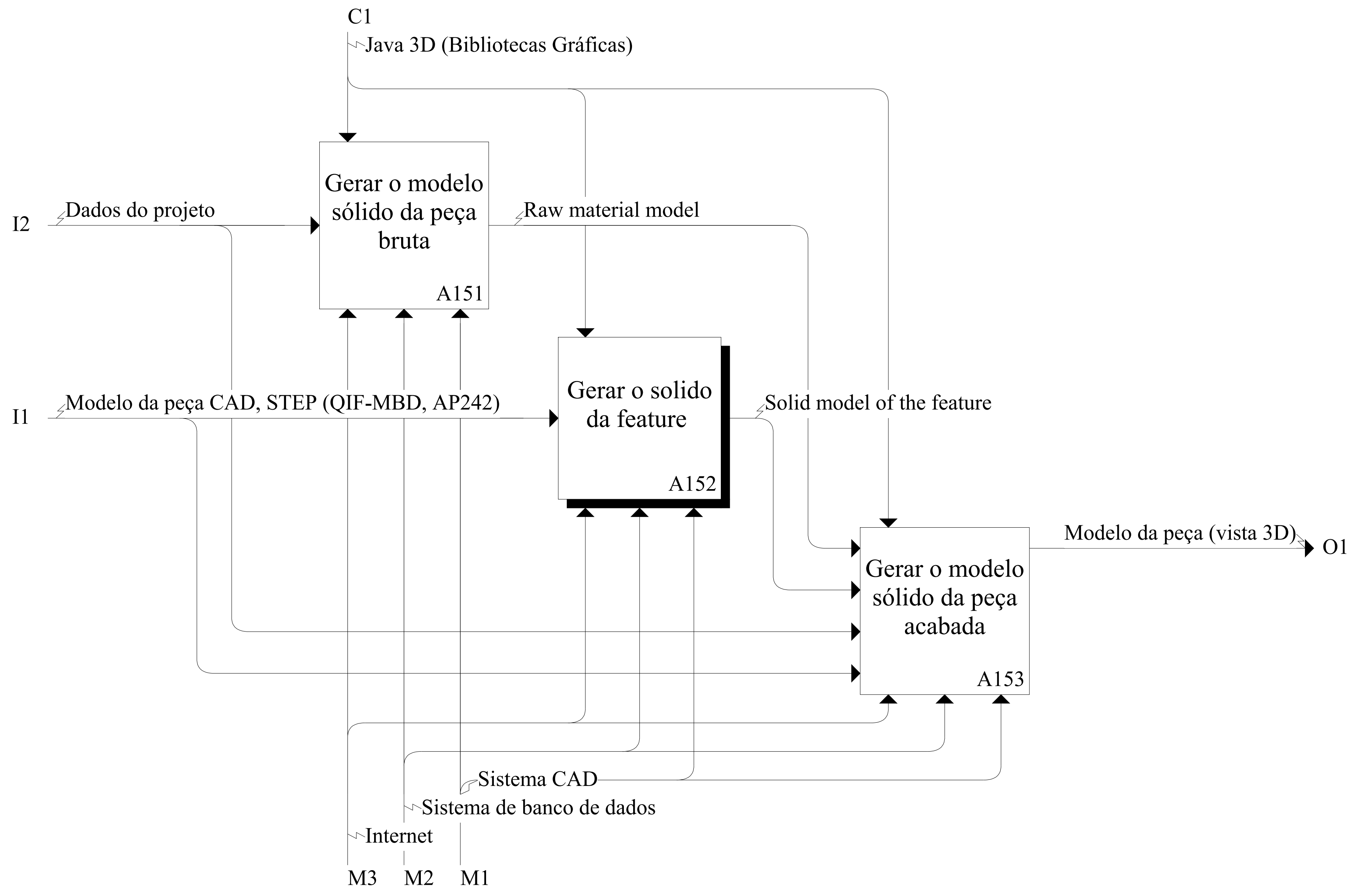
Modelo de integração em malha fechada


C12
C13
C8
C2
C3
C1
C6
C7
C10
C11
C14
C15
C4
C5
C16
C17
C18
C22
C19
C9
C20
C21
Java Swing ((Bibliotecas Gráficas)
Java 3D (Bibliotecas Gráficas)
Biblioteca de Features (QIF Library, STEP Library)
ISO 10303-28 (Método de implementação XML)
Recursos disponíveis (ferramentas, Acessórios de fixação)
API JSDAI
ISO10303-21 STEP Arquivo p21
ISO14649-111 (Ferramentas para usinagem)
ISO14649-10 (Dados gerais de processamento)
ISO14649-11 Processamento de dados de usinagem
Tipo de inspeção
Padrão de linguagem de programação
Dados administrativos
Especificações de DME (QIF Resources)
Requisit
Padrão, Especificações, Políticas e Procedimentos (DMIS)
Dimensões e tolerâncias (GD&T)
Quality Information Framework (QIF)
Definição de produto (QIF MBD, AP238)
ISO6983 G-code
RS274NGC Padrão - NC Programs
Condições do entorno (QIF-Rules)
I7
Features de usinagem
I3
Material Feature
I1
Dados do projeto
I2
Usuário & senha
I4
Feature de tolerância
I5
Ferramentas e acessórios DME
I6
Ordem de inspeção
I8
Feature do material bruto
CAD
A1
CAPP
A2
CAIP
A3
CAM
A4
CNC
A5
CAI
A6
Base de Dados
A7
Modelo gráfico (vista 2D)
O1
Modelo da peça (vista 3D)
O2
Modelo da peça CAD, STEP (QIF-MBD, AP242)
O3
STEP-NC Arquivo em XML
O5
Arquivo neutro STEP-NC (p21)
O8
Plano de processo de fabricação (QIF-Plans)
O7
Estruturas de dados
Plano de inspeção dimensional (w / FLs)
Dados de suporte
Pedido de inspeção
Programa DME (w/FLs)
Ferramentas DME e acessórios
Simulação
O4
Visualização do programa STEP-NC
O10
Monitoramento remoto de máquina ferramenta
Controle da máquina ferramenta
Programa de mecanizado em ISO6983
O6
Programa da peça no código G
Condição do Processo de Usinagem
Peça
Peça inspecionada
O9
QIF Results
O11
QIF Statistics
O12
Resultados da Execução (w / FLs)
Resultados estadísticos (w/FL)
Retroalimentação de resultados analisados (w/FLs)
Repositório de resultados analisados
Dados de pontos
Retroalimentar resultados analisados
Tolerâncias relacionadas e Nominal
Sistema de coordenadas
Dados de Calibração do Sensor
MTConnect Arquivo XML
Funções matemáticas
LinuxCNC AXIS GUI
LinuxCNC .INI configurações de arquivos
Regras de produção
Regras de validação
Sistema baseado em conhecimento
Sistema de banco de dados
Sistema de Coordenadas
Informações arquivadas
Ferramentas de software
Biblioteca do programa DME
Sistema de Simulação de Inspeção
Validador do plano de inspeção
Sistema de fixação
Sistema de Planejamento de Inspeção
Máquina ferramenta
Sistema CAM
Operador
MTConnect
Sistema CAD
Internet
Cliente remoto
Sistema CAPP
Sistema CNC
Recursos humanos
M5
M4
M1
M3
M9
M10
M11
M12
M7
M16
M13
M15
M14
M2
M8
M6

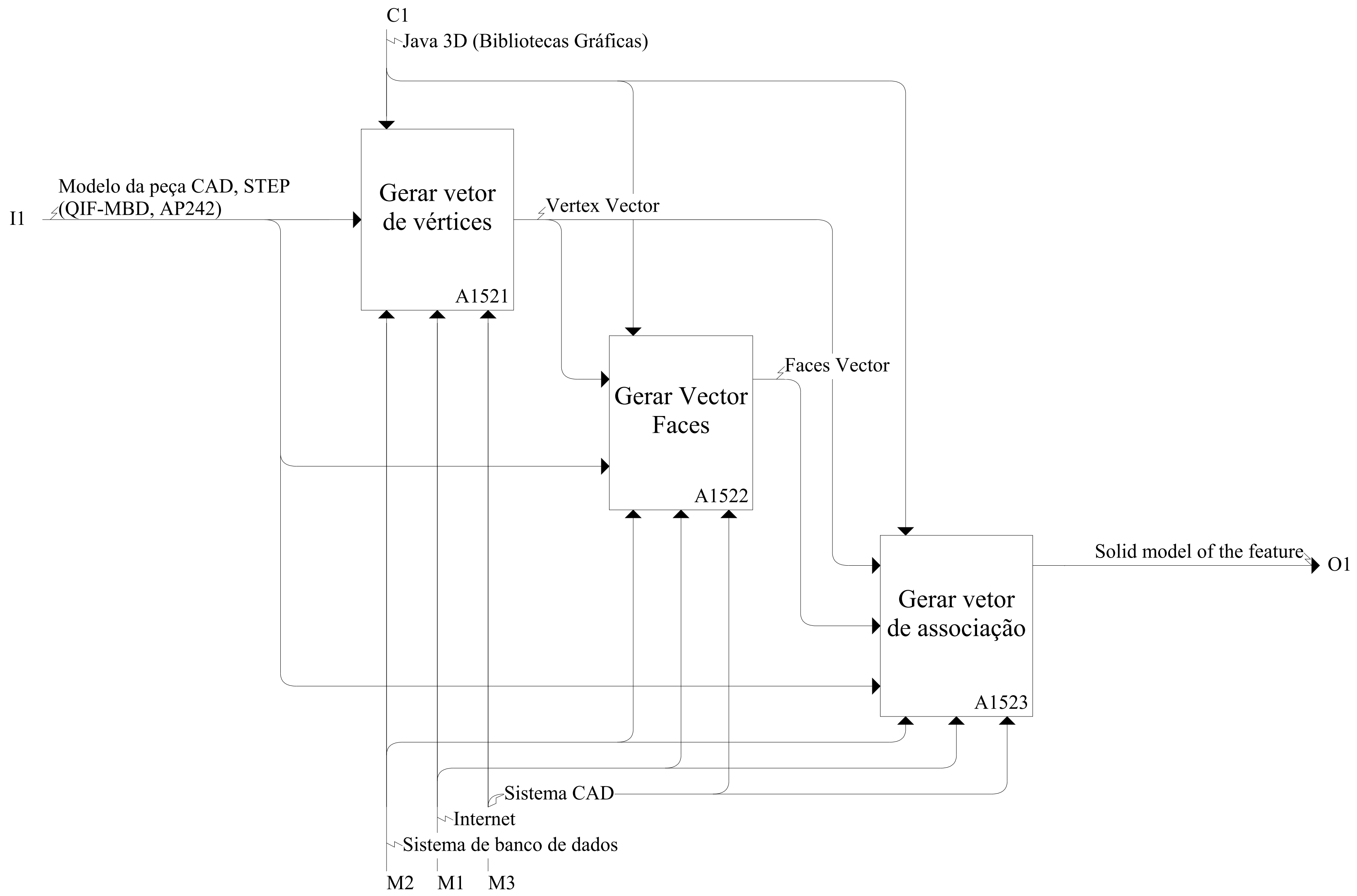
Modelo de integração em malha fechada


C4
C1
C2
C3
Regras de validação
Java 3D (Bibliotecas Gráficas)
Biblioteca de Features (QIF Library, STEP Library)
Java Swing ((Bibliotecas Gráficas)
I1
Features de usinagem
M1
Cliente remoto
I5
Feature de tolerância
I2
Material Feature
I4
Usuário & senha
I3
Dados do projeto
Criar novo projeto
A11
Project
Modelo da peça usando a biblioteca de features
A12
Feature-based model of part
Validar o modelo criado
A13
Modelo da peça CAD, STEP (QIF-MBD, AP242)
O3
Gerar uma representação visual 2D
A14
Modelo gráfico (vista 2D)
O1
Gerar uma representação 3D
A15
Modelo da peça (vista 3D)
O2
Internet
Sistema de banco de dados
Sistema CAD
M3
M4
M2

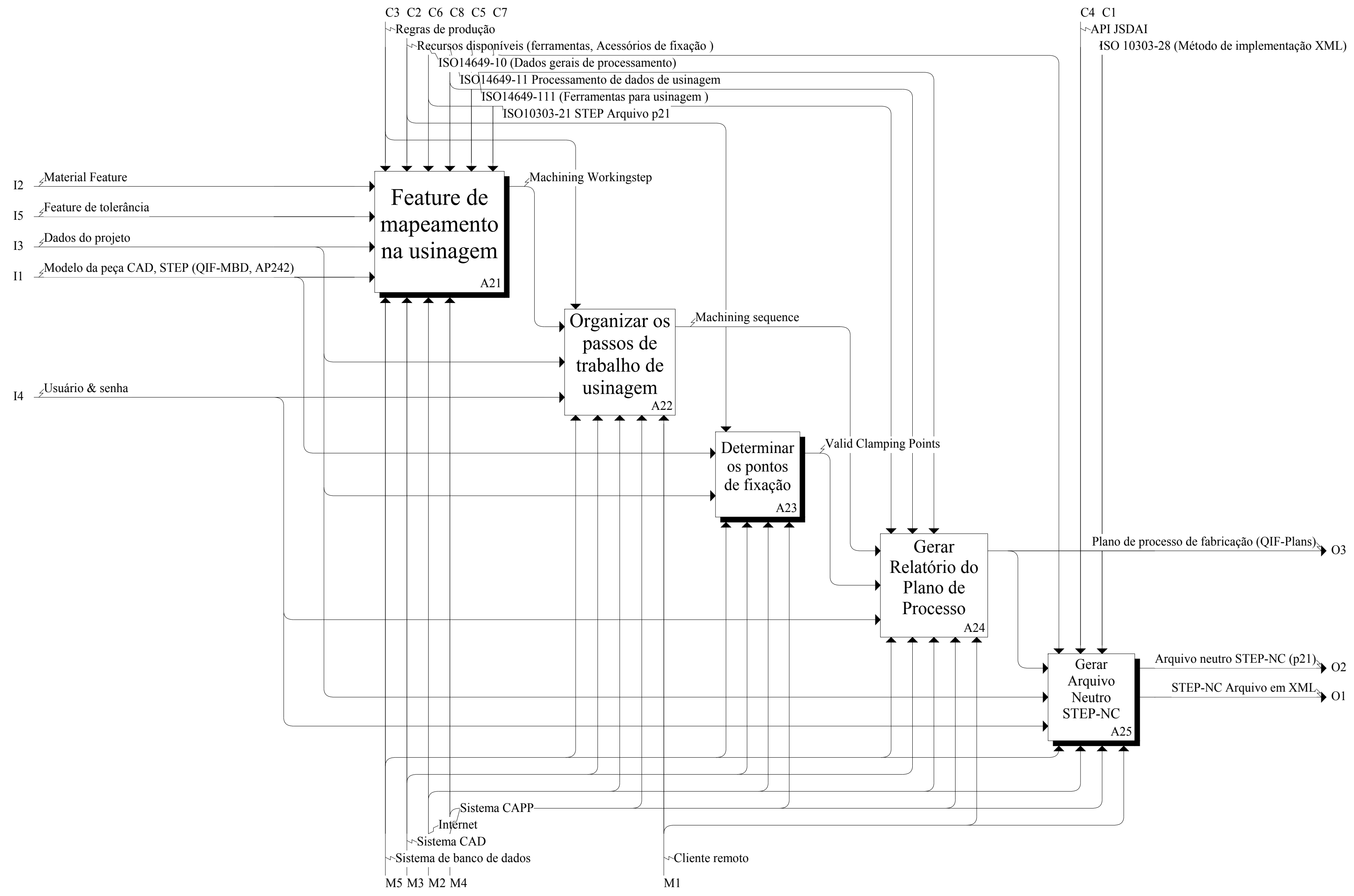
Modelo de integração em malha fechada


C1
Java 3D (Bibliotecas Gráficas)
I2
Dados do projeto
Gerar o modelo sólido da peça bruta
A151
Raw material model
I1
Modelo da peça CAD, STEP (QIF-MBD, AP242)
Gerar o solido da feature
A152
Solid model of the feature
Gerar o modelo sólido da peça acabada
A153
Modelo da peça (vista 3D)
O1
Sistema CAD
Sistema de banco de dados
Internet
M3
M2
M1

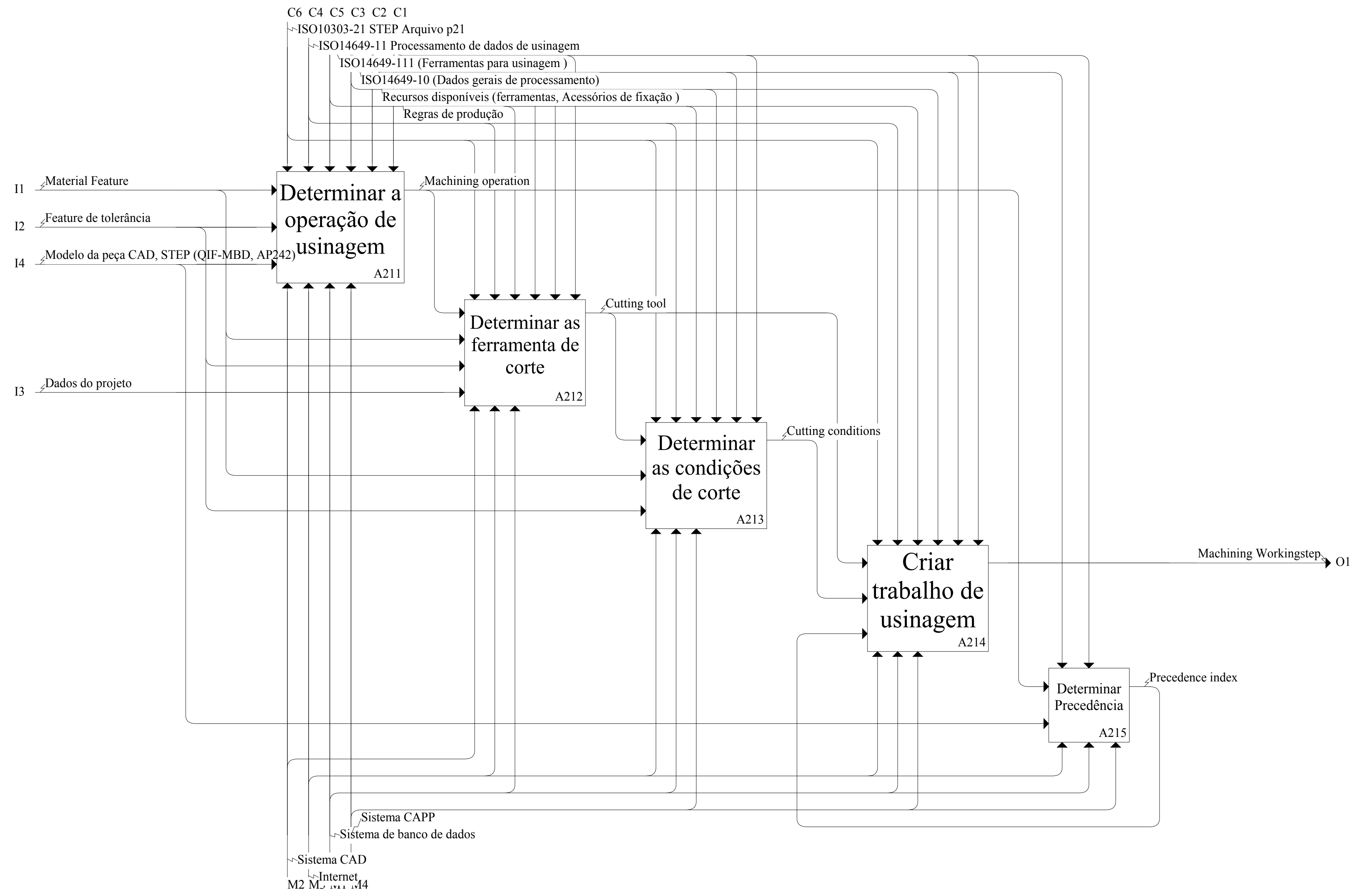
Modelo de integração em malha fechada


C1
Java 3D (Bibliotecas Gráficas)
I1
Modelo da peça CAD, STEP (QIF-MBD, AP242)
Gerar vetor de vértices
A1521
Vertex Vector
Gerar Vector Faces
A1522
Faces Vector
Gerar vetor de associação
A1523
Solid model of the feature
O1
Sistema CAD
Internet
Sistema de banco de dados
M2
M1
M3

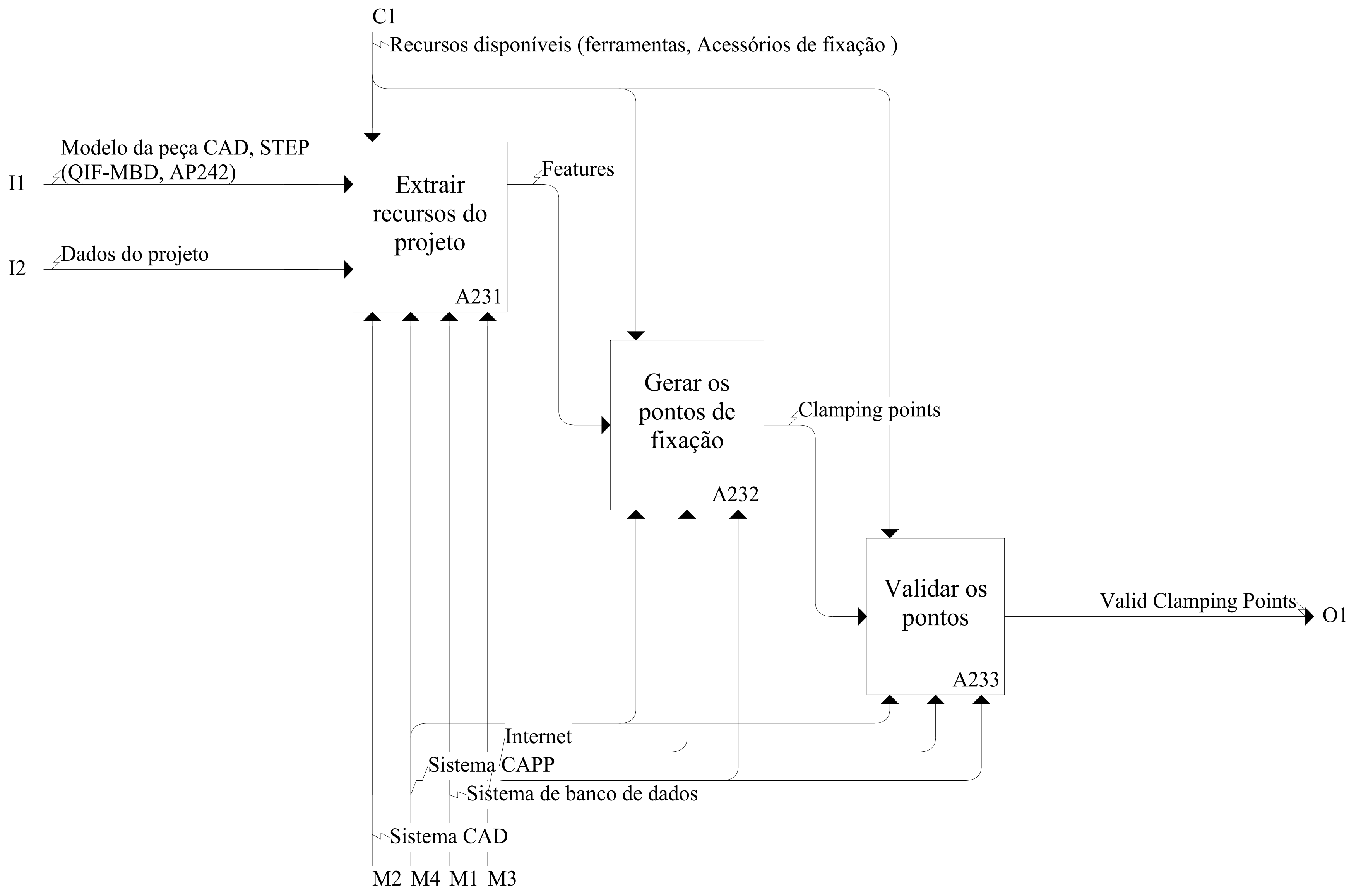
Modelo de integração em malha fechada


C3 C2 C6 C8 C5 C7
Regras de produção
Recursos disponíveis (ferramentas, Acessórios de fixação)
ISO14649-10 (Dados gerais de processamento)
ISO14649-11 Processamento de dados de usinagem
ISO14649-111 (Ferramentas para usinagem)
ISO10303-21 STEP Arquivo p21
C4 C1
API JSDAI
ISO 10303-28 (Método de implementação XML)
I2 Material Feature
I5 Feature de tolerância
I3 Dados do projeto
I1 Modelo da peça CAD, STEP (QIF-MBD, AP242)
I4 Usuário & senha
Feature de mapeamento na usinagem
A21
Machining Workingstep
Organizar os passos de trabalho de usinagem
A22
Machining sequence
Determinar os pontos de fixação
A23
Valid Clamping Points
Gerar Relatório do Plano de Processo
A24
Plano de processo de fabricação (QIF-Plans)
O3
Gerar Arquivo Neutro STEP-NC
A25
Arquivo neutro STEP-NC (p21)
O2
STEP-NC Arquivo em XML
O1
Sistema CAPP
Internet
Sistema CAD
Sistema de banco de dados
Cliente remoto
M5 M3 M2 M4
M1

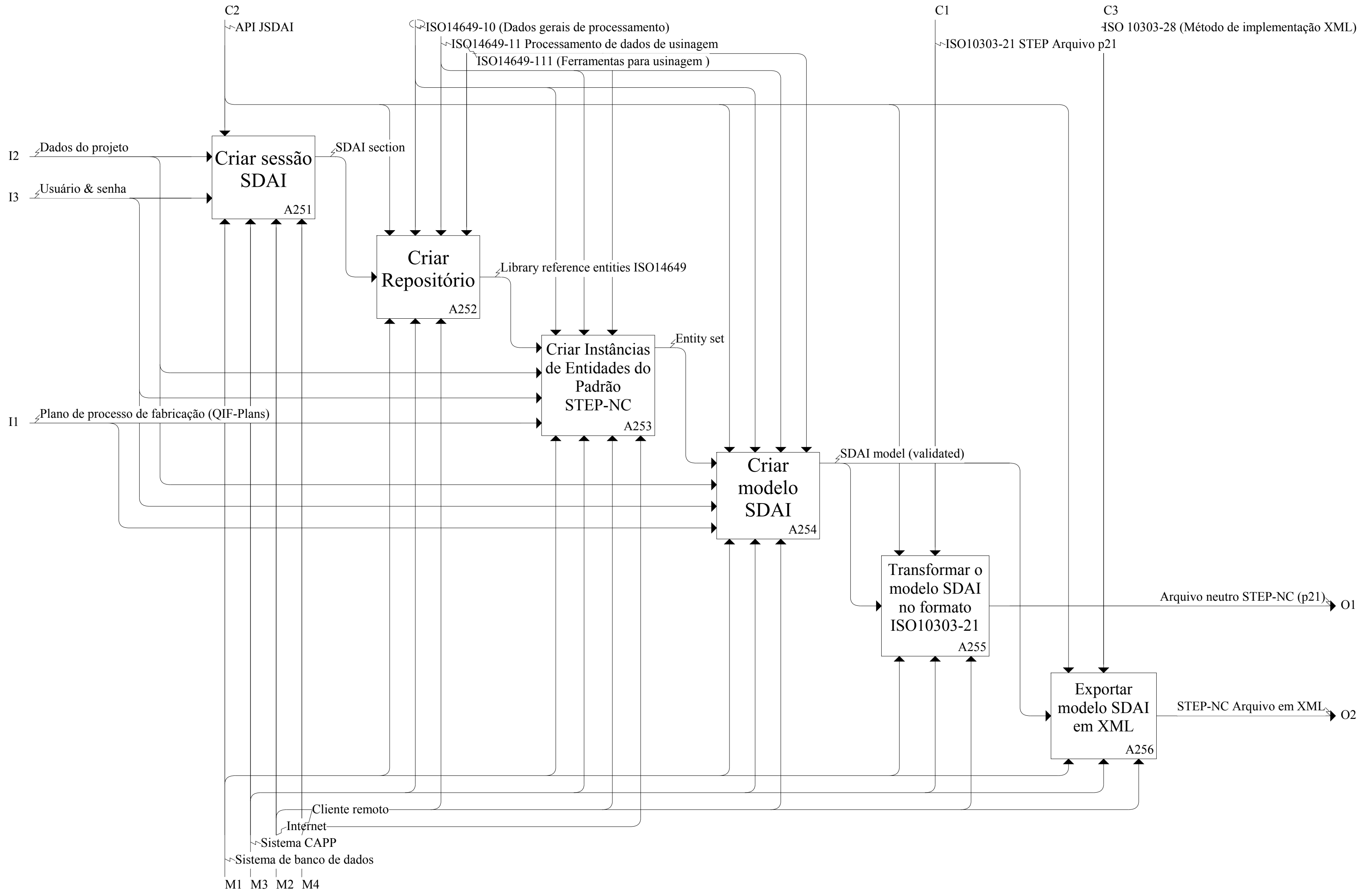
Modelo de integração em malha fechada


C6 C4 C5 C3 C2 C1
ISO10303-21 STEP Arquivo p21
ISO14649-11 Processamento de dados de usinagem
ISO14649-111 (Ferramentas para usinagem)
ISO14649-10 (Dados gerais de processamento)
Recursos disponíveis (ferramentas, Acessórios de fixação)
Regras de produção
I1
Material Feature
I2
Feature de tolerância
I4
Modelo da peça CAD, STEP (QIF-MBD, AP242)
I3
Dados do projeto
Determinar a operação de usinagem
A211
Machining operation
Determinar as ferramenta de corte
A212
Cutting tool
Determinar as condições de corte
A213
Cutting conditions
Criar trabalho de usinagem
A214
Machining Workingstep
O1
Determinar Precedência
A215
Precedence index
Sistema CAPP
Sistema de banco de dados
Sistema CAD
Internet
M2 M3 M1 M4

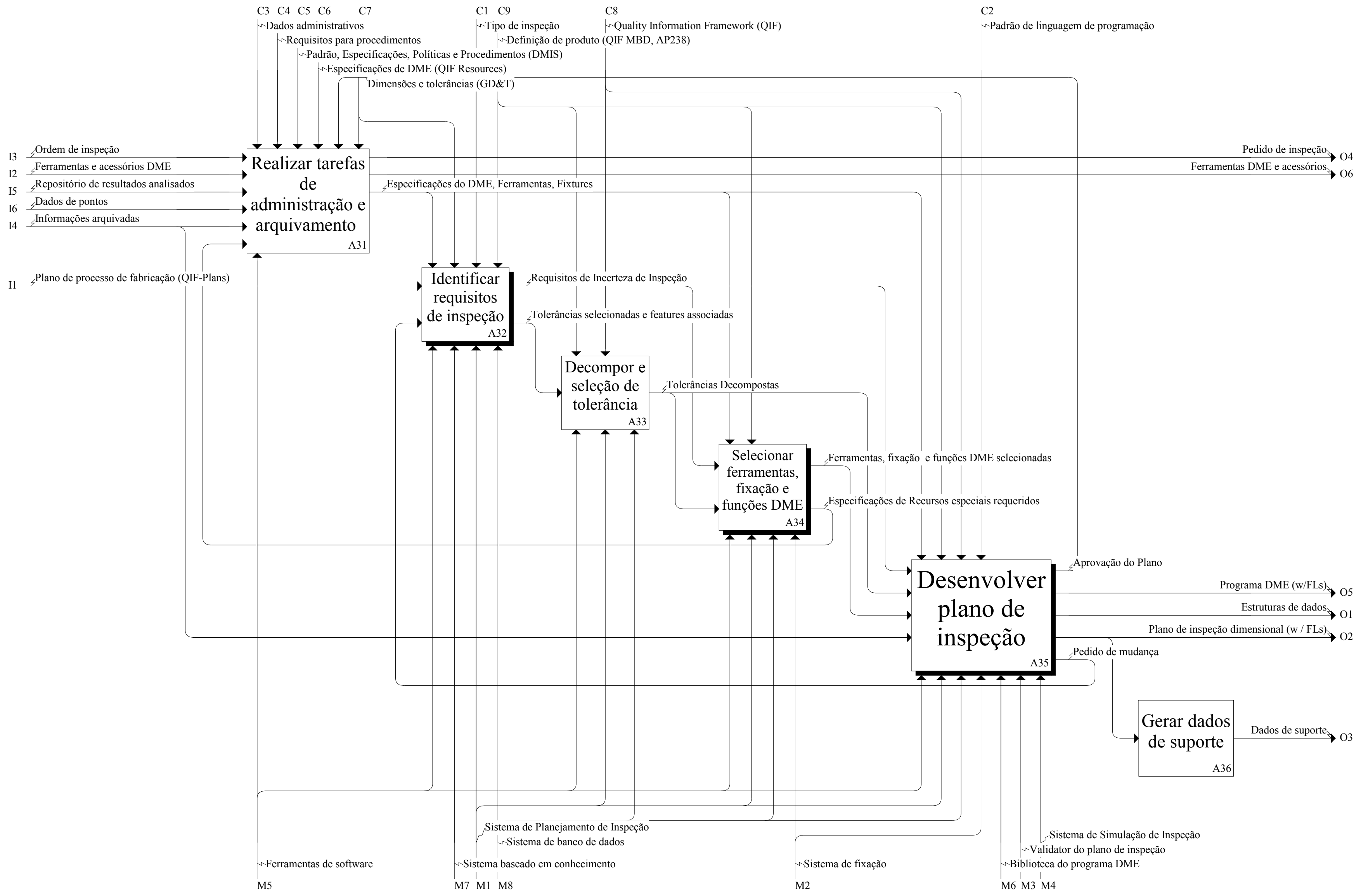
Modelo de integração em malha fechada


C1
Recursos disponíveis (ferramentas, Acessórios de fixação)
I1
Modelo da peça CAD, STEP (QIF-MBD, AP242)
I2
Dados do projeto
Extrair recursos do projeto
A231
Features
Gerar os pontos de fixação
A232
Clamping points
Validar os pontos
A233
Valid Clamping Points
O1
Internet
Sistema CAPP
Sistema de banco de dados
Sistema CAD
M2
M4
M1
M3

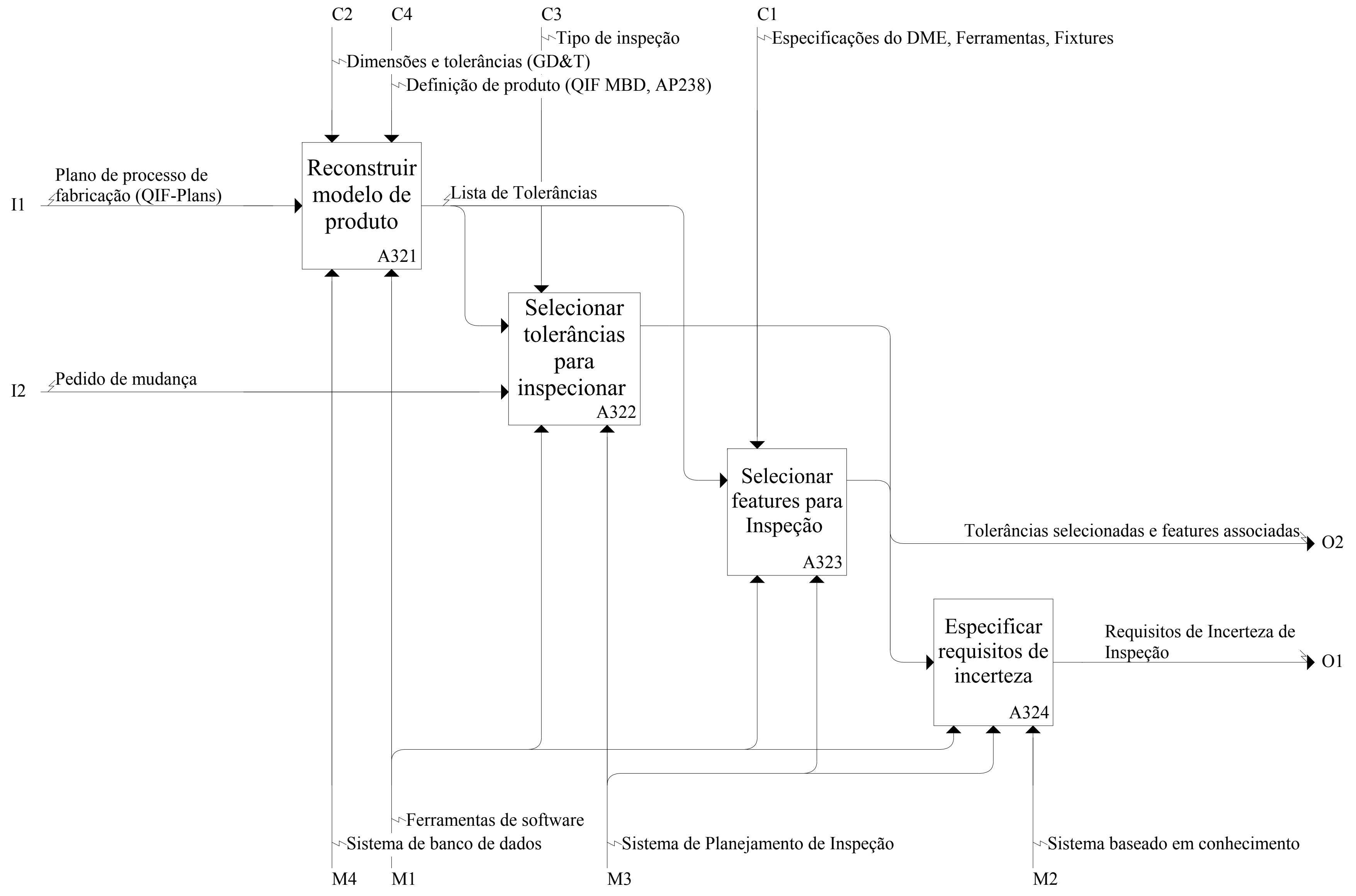
Modelo de integração em malha fechada


C2
API JSDAI
ISO14649-10 (Dados gerais de processamento)
ISO14649-11 Processamento de dados de usinagem
ISO14649-111 (Ferramentas para usinagem)
C1
ISO10303-21 STEP Arquivo p21
C3
ISO 10303-28 (Método de implementação XML)
I2
Dados do projeto
I3
Usuário & senha
Criar sessão SDAI
A251
SDAI section
Criar Repositório
A252
Library reference entities ISO14649
Criar Instâncias de Entidades do Padrão STEP-NC
A253
Entity set
I1
Plano de processo de fabricação (QIF-Plans)
Criar modelo SDAI
A254
SDAI model (validated)
Transformar o modelo SDAI no formato ISO10303-21
A255
Arquivo neutro STEP-NC (p21)
O1
Exportar modelo SDAI em XML
A256
STEP-NC Arquivo em XML
O2
Cliente remoto
Internet
Sistema CAPP
Sistema de banco de dados
M1
M3
M2
M4

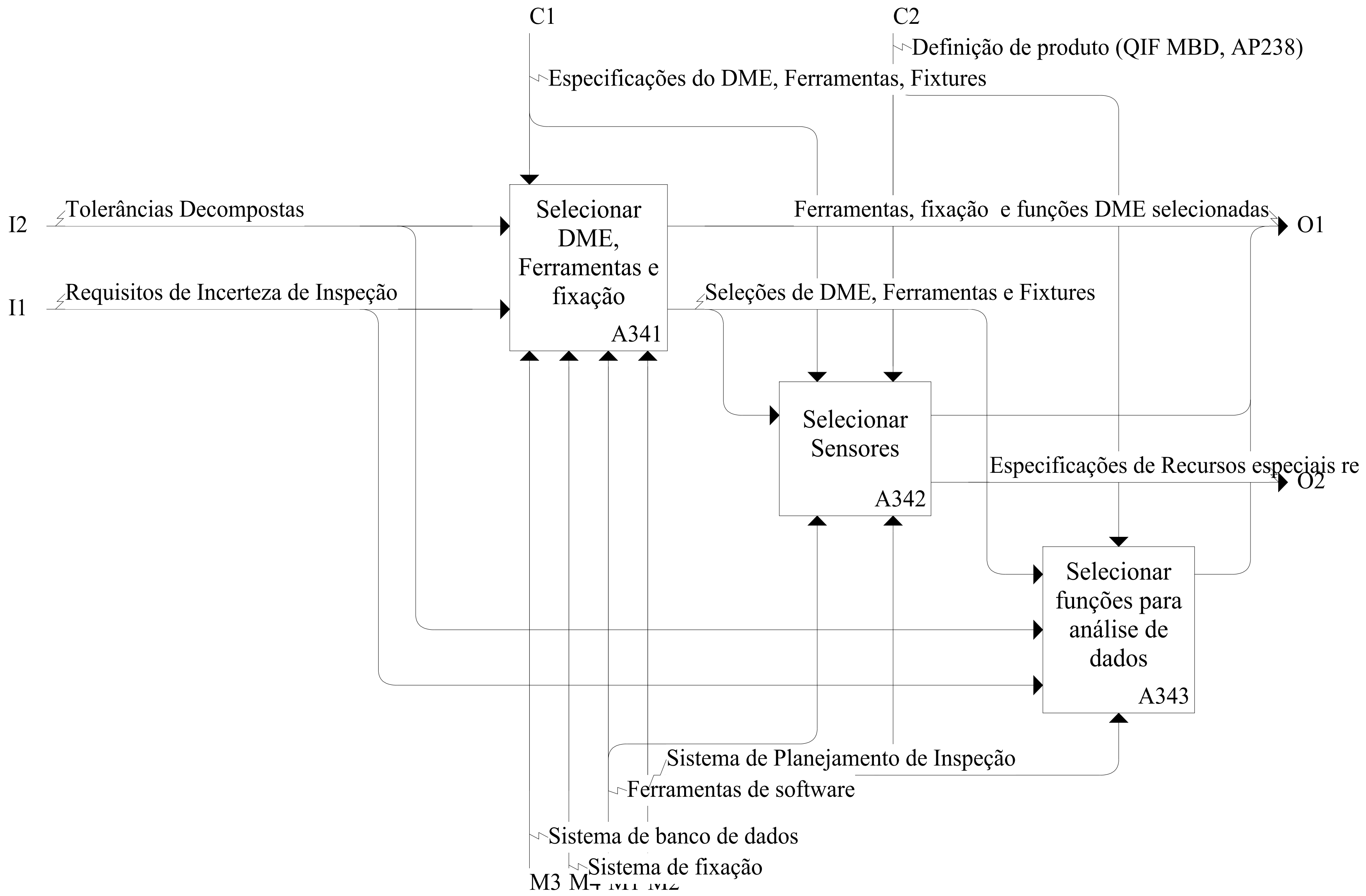
Modelo de integração em malha fechada


C3 C4 C5 C6 C7
Dados administrativos
Requisitos para procedimentos
Padrão, Especificações, Políticas e Procedimentos (DMIS)
Especificações de DME (QIF Resources)
Dimensões e tolerâncias (GD&T)
C1 C9
Tipo de inspeção
Definição de produto (QIF MBD, AP238)
C8
Quality Information Framework (QIF)
C2
Padrão de linguagem de programação
I3 Ordem de inspeção
I2 Ferramentas e acessórios DME
I5 Repositório de resultados analisados
I6 Dados de pontos
I4 Informações arquivadas
I1 Plano de processo de fabricação (QIF-Plans)
Realizar tarefas de administração e arquivamento
A31
Especificações do DME, Ferramentas, Fixtures
Pedido de inspeção O4
Ferramentas DME e acessórios O6
Identificar requisitos de inspeção
A32
Requisitos de Incerteza de Inspeção
Tolerâncias selecionadas e features associadas
Decompor e seleção de tolerância
A33
Tolerâncias Decompostas
Selecionar ferramentas, fixação e funções DME
A34
Ferramentas, fixação e funções DME selecionadas
Especificações de Recursos especiais requeridos
Desenvolver plano de inspeção
A35
Aprovação do Plano
Programa DME (w/FLs) O5
Estruturas de dados O1
Plano de inspeção dimensional (w / FLs) O2
Pedido de mudança
Gerar dados de suporte
A36
Dados de suporte O3
Sistema de Planejamento de Inspeção
Sistema de banco de dados
Ferramentas de software
Sistema baseado em conhecimento
Sistema de fixação
Sistema de Simulação de Inspeção
Validator do plano de inspeção
Biblioteca do programa DME
M5
M7 M1 M8
M2
M6 M3 M4

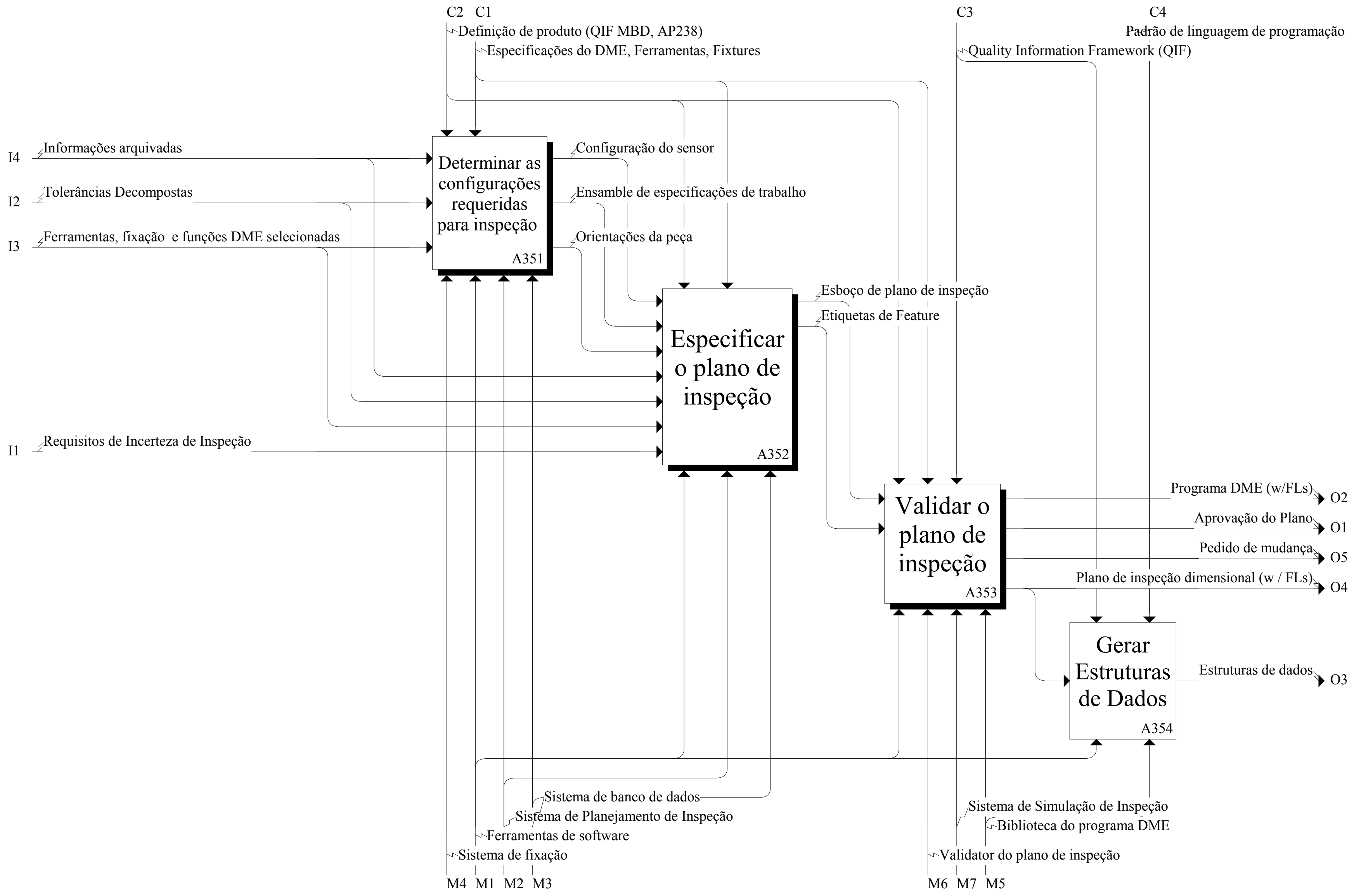
Modelo de integração em malha fechada


C2
C4
C3
C1
Tipo de inspeção
Especificações do DME, Ferramentas, Fixtures
Dimensões e tolerâncias (GD&T)
Definição de produto (QIF MBD, AP238)
I1
Plano de processo de fabricação (QIF-Plans)
Reconstruir modelo de produto
A321
Lista de Tolerâncias
Selecionar tolerâncias para inspecionar
A322
I2
Pedido de mudança
Selecionar features para Inspeção
A323
Tolerâncias selecionadas e features associadas
O2
Especificar requisitos de incerteza
A324
Requisitos de Incerteza de Inspeção
O1
Ferramentas de software
Sistema de banco de dados
Sistema de Planejamento de Inspeção
Sistema baseado em conhecimento
M4
M1
M3
M2

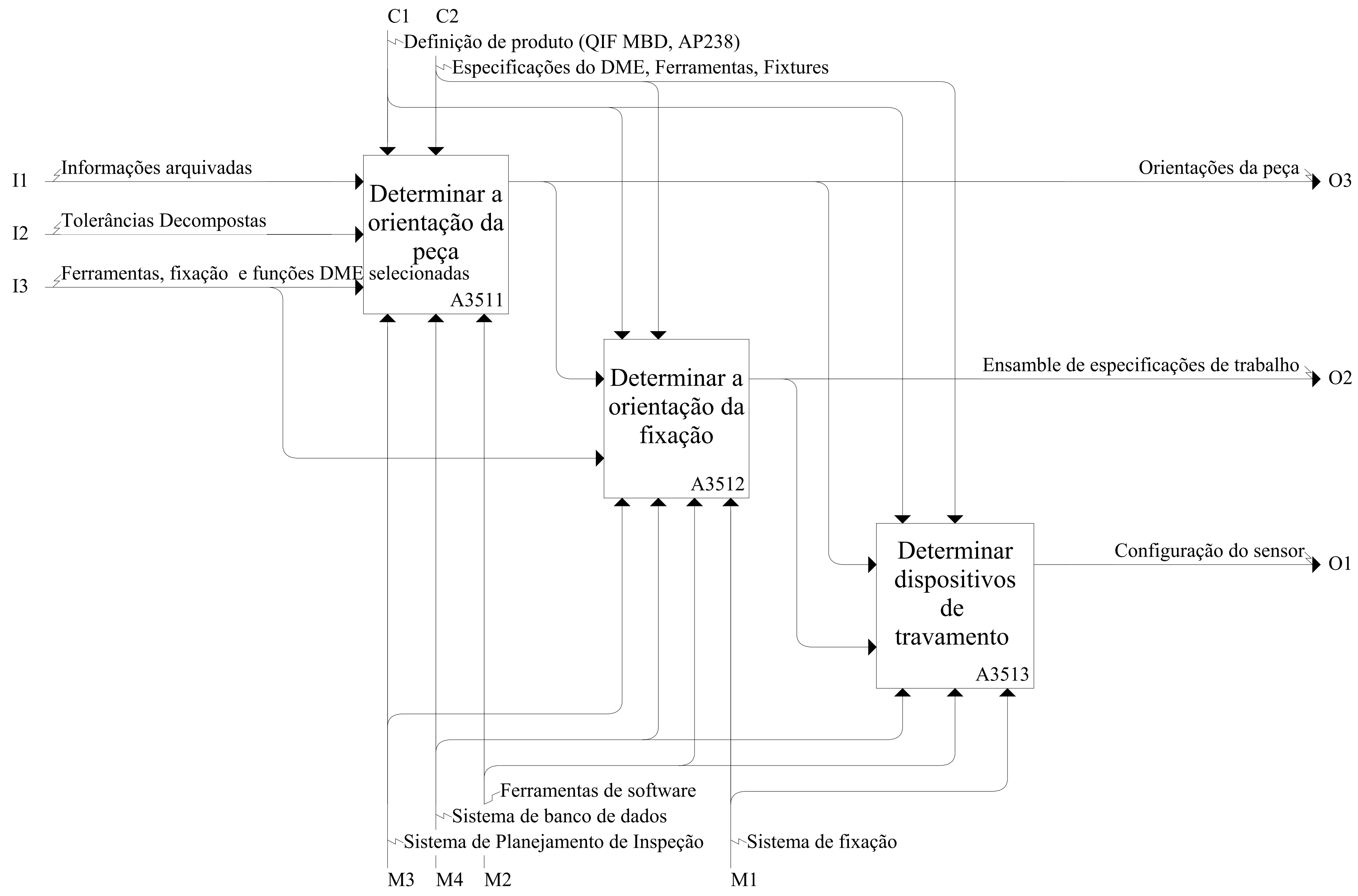
Modelo de integração em malha fechada


C1
C2
Definição de produto (QIF MBD, AP238)
Especificações do DME, Ferramentas, Fixtures
I2
Tolerâncias Decompostas
I1
Requisitos de Incerteza de Inspeção
Selecionar DME, Ferramentas e fixação
A341
Ferramentas, fixação e funções DME selecionadas
O1
Seleções de DME, Ferramentas e Fixtures
Selecionar Sensores
A342
Especificações de Recursos especiais re
O2
Selecionar funções para análise de dados
A343
Sistema de Planejamento de Inspeção
Ferramentas de software
Sistema de banco de dados
Sistema de fixação
M3
M4
M1
M2

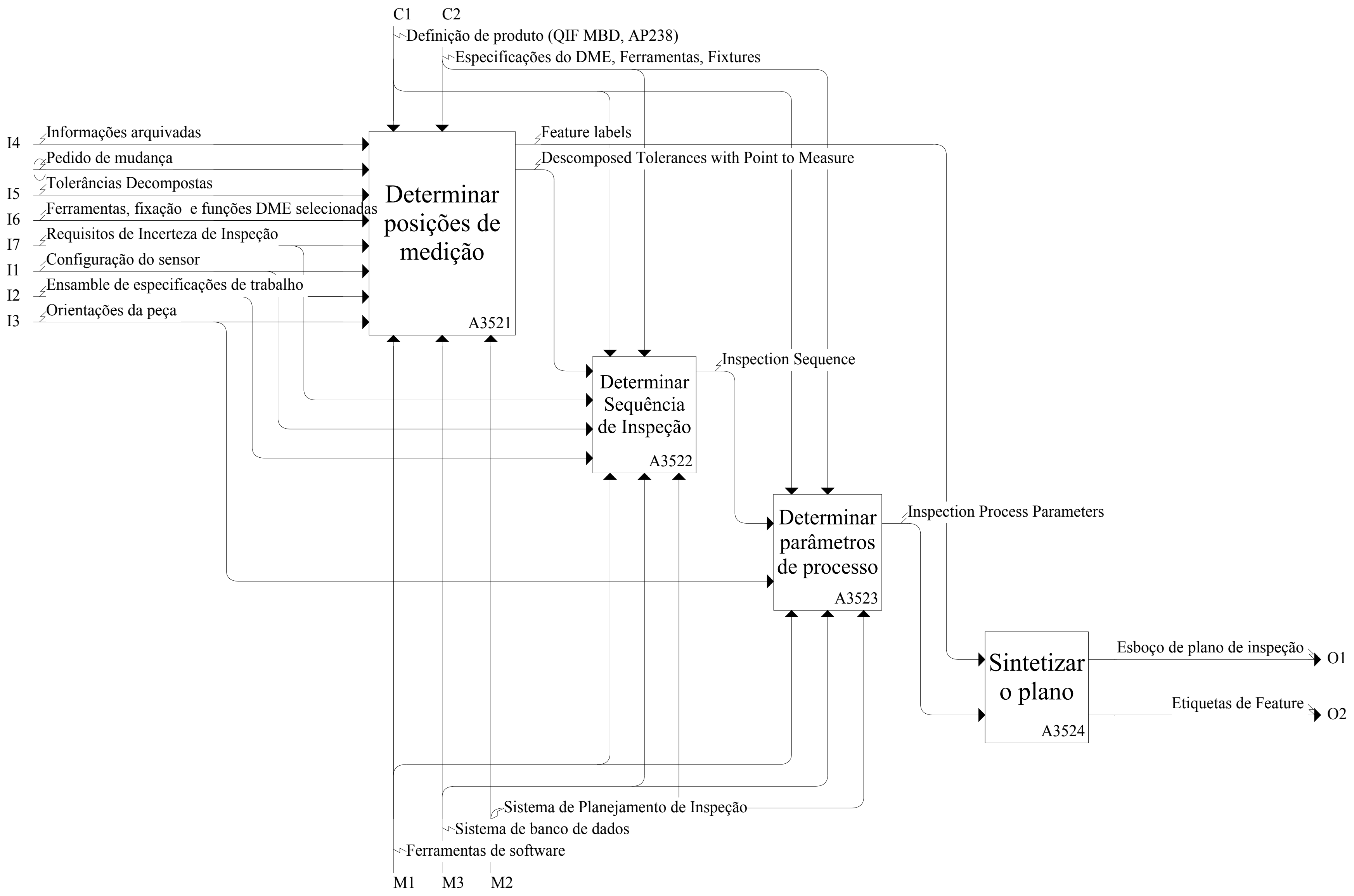
Modelo de integração em malha fechada


C2
C1
Definição de produto (QIF MBD, AP238)
Especificações do DME, Ferramentas, Fixtures
C3
Quality Information Framework (QIF)
C4
Padrão de linguagem de programação
I4
Informações arquivadas
I2
Tolerâncias Decompostas
I3
Ferramentas, fixação e funções DME selecionadas
I1
Requisitos de Incerteza de Inspeção
Determinar as configurações requeridas para inspeção
A351
Configuração do sensor
Ensamble de especificações de trabalho
Orientações da peça
Especificar o plano de inspeção
A352
Esboço de plano de inspeção
Etiquetas de Feature
Validar o plano de inspeção
A353
Programa DME (w/FLs)
O2
Aprovação do Plano
O1
Pedido de mudança
O5
Plano de inspeção dimensional (w / FLs)
O4
Gerar Estruturas de Dados
A354
Estruturas de dados
O3
Sistema de banco de dados
Sistema de Planejamento de Inspeção
Ferramentas de software
Sistema de fixação
Sistema de Simulação de Inspeção
Biblioteca do programa DME
Validator do plano de inspeção
M4
M1
M2
M3
M6
M7
M5

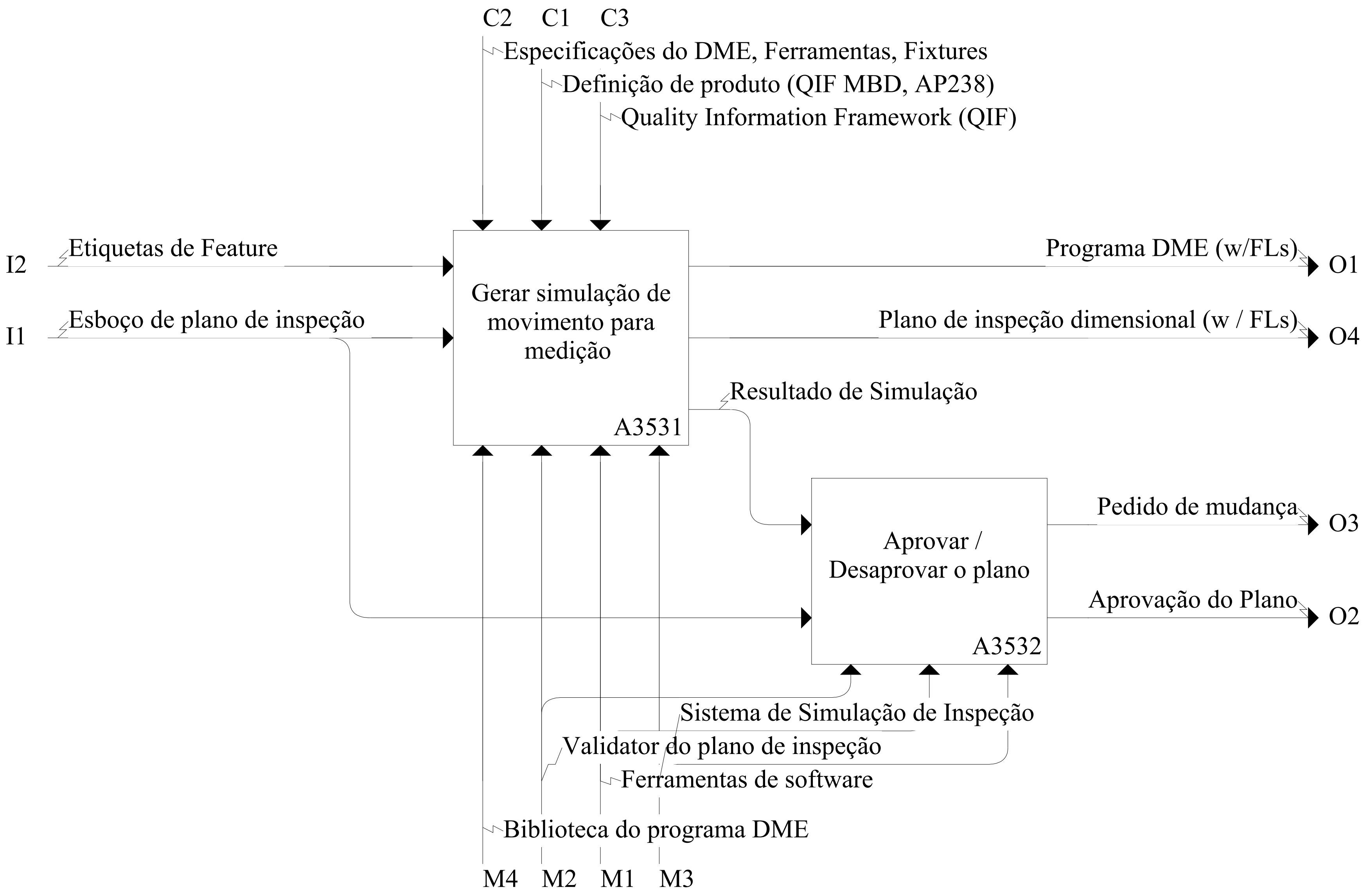
Modelo de integração em malha fechada


C1
C2
Definição de produto (QIF MBD, AP238)
Especificações do DME, Ferramentas, Fixtures
I1
Informações arquivadas
I2
Tolerâncias Decompostas
I3
Ferramentas, fixação e funções DME selecionadas
Determinar a orientação da peça
A3511
Orientações da peça
O3
Determinar a orientação da fixação
A3512
Ensamble de especificações de trabalho
O2
Determinar dispositivos de travamento
A3513
Configuração do sensor
O1
Ferramentas de software
Sistema de banco de dados
Sistema de Planejamento de Inspeção
Sistema de fixação
M3
M4
M2
M1

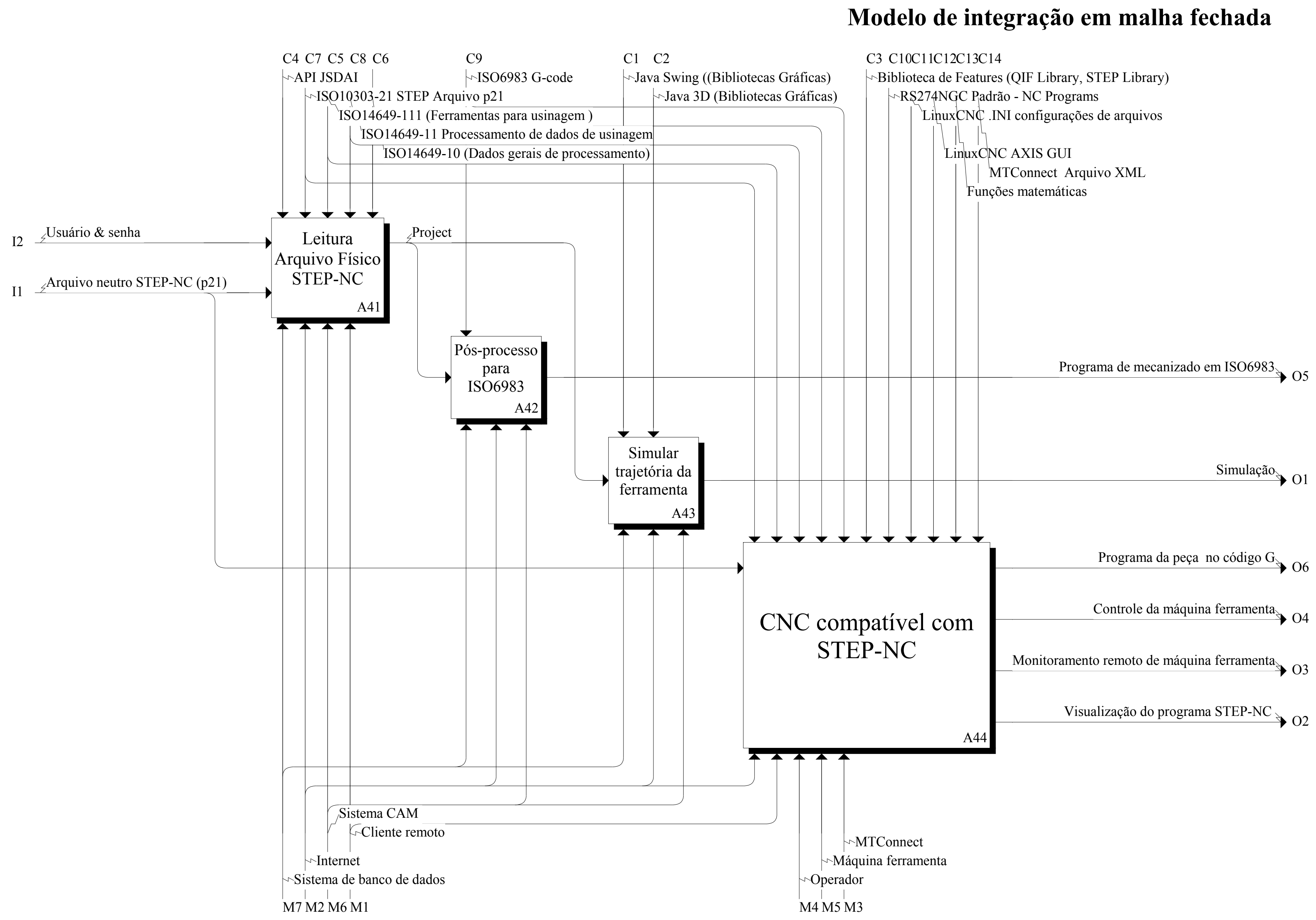
Modelo de integração em malha fechada


C1
C2
Definição de produto (QIF MBD, AP238)
Especificações do DME, Ferramentas, Fixtures
I4
Informações arquivadas
Pedido de mudança
I5
Tolerâncias Decompostas
I6
Ferramentas, fixação e funções DME selecionadas
I7
Requisitos de Incerteza de Inspeção
I1
Configuração do sensor
I2
Ensamble de especificações de trabalho
I3
Orientações da peça
Determinar posições de medição
A3521
Feature labels
Descomposed Tolerances with Point to Measure
Determinar Sequência de Inspeção
A3522
Inspection Sequence
Determinar parâmetros de processo
A3523
Inspection Process Parameters
Sintetizar o plano
A3524
Esboço de plano de inspeção
O1
Etiquetas de Feature
O2
Sistema de Planejamento de Inspeção
Sistema de banco de dados
Ferramentas de software
M1
M3
M2

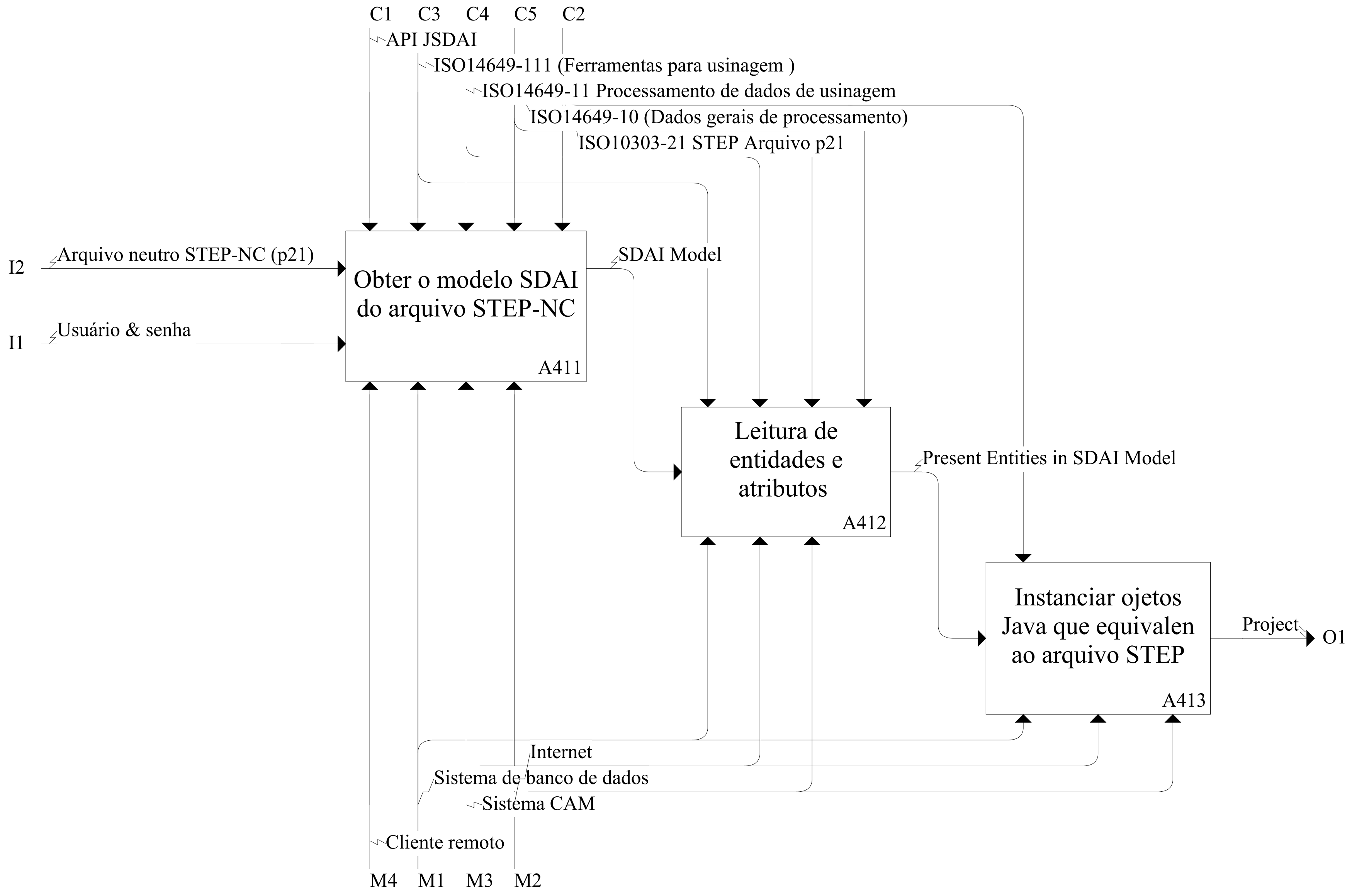
Modelo de integração em malha fechada



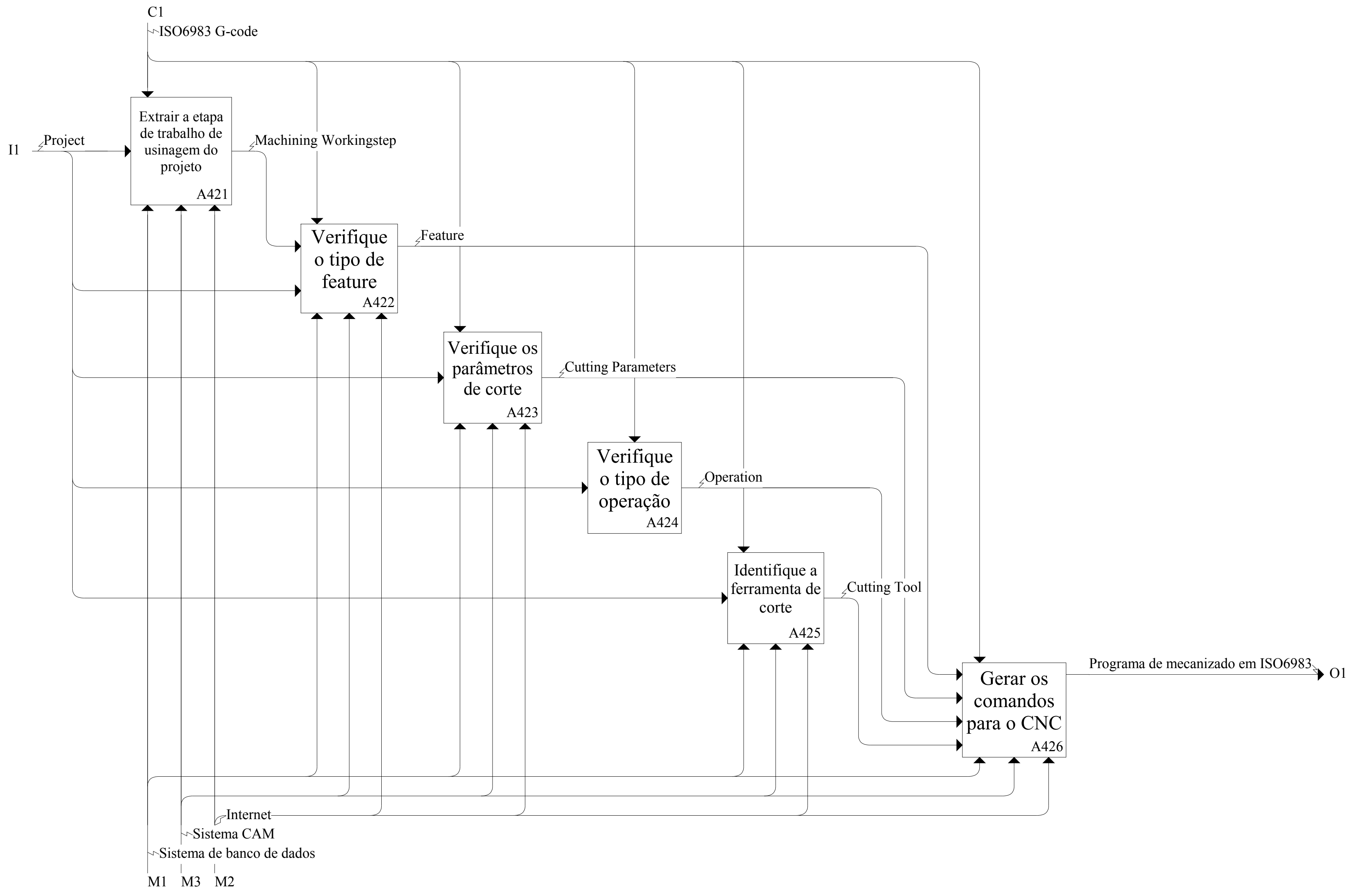
C2
C1
C3
Especificações do DME, Ferramentas, Fixtures
Definição de produto (QIF MBD, AP238)
Quality Information Framework (QIF)
I2
Etiquetas de Feature
I1
Esboço de plano de inspeção
Gerar simulação de movimento para medição
A3531
Programa DME (w/FLs)
O1
Plano de inspeção dimensional (w / FLs)
O4
Resultado de Simulação
Aprovar / Desaprovar o plano
A3532
Pedido de mudança
O3
Aprovação do Plano
O2
Sistema de Simulação de Inspeção
Validator do plano de inspeção
Ferramentas de software
Biblioteca do programa DME
M4
M2
M1
M3


Modelo de integração em malha fechada
C4 C7 C5 C8 C6
API JSDAI
ISO10303-21 STEP Arquivo p21
ISO14649-111 (Ferramentas para usinagem)
ISO14649-11 Processamento de dados de usinagem
ISO14649-10 (Dados gerais de processamento)
C9
ISO6983 G-code
C1 C2
Java Swing ((Bibliotecas Gráficas)
Java 3D (Bibliotecas Gráficas)
C3 C10C11C12C13C14
Biblioteca de Features (QIF Library, STEP Library)
RS274NGC Padrão - NC Programs
LinuxCNC .INI configurações de arquivos
LinuxCNC AXIS GUI
MTConnect Arquivo XML
Funções matemáticas
I2
Usuário & senha
I1
Arquivo neutro STEP-NC (p21)
Leitura Arquivo Físico STEP-NC
A41
Project
Pós-processo para ISO6983
A42
Simular trajetória da ferramenta
A43
CNC compatível com STEP-NC
A44
Programa de mecanizado em ISO6983
O5
Simulação
O1
Programa da peça no código G
O6
Controle da máquina ferramenta
O4
Monitoramento remoto de máquina ferramenta
O3
Visualização do programa STEP-NC
O2
Sistema CAM
Cliente remoto
Internet
Sistema de banco de dados
M7 M2 M6 M1
MTConnect
Máquina ferramenta
Operador
M4 M5 M3

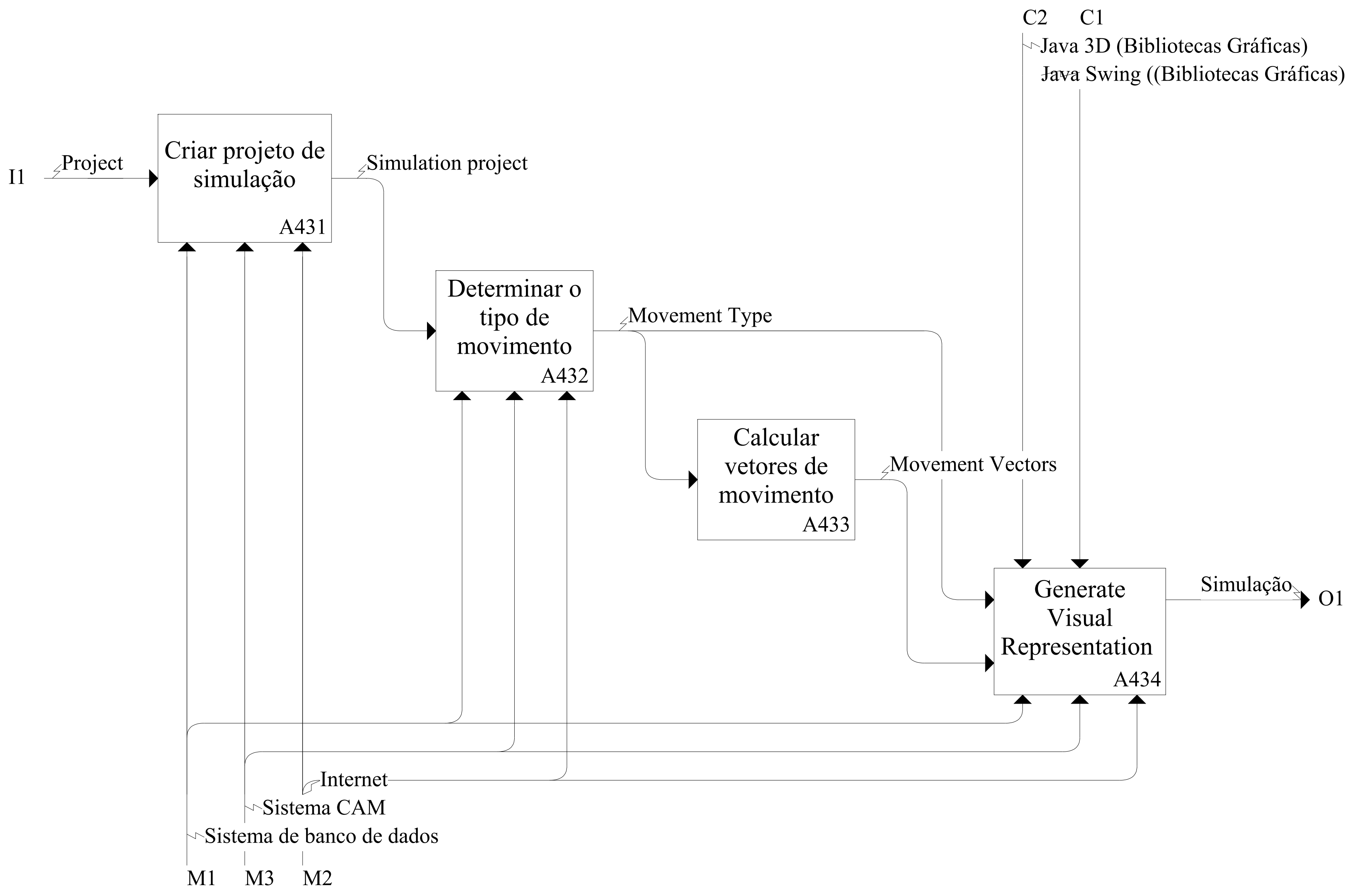
Modelo de integração em malha fechada


C1
C3
C4
C5
C2
API JSDAI
ISO14649-111 (Ferramentas para usinagem)
ISO14649-11 Processamento de dados de usinagem
ISO14649-10 (Dados gerais de processamento)
ISO10303-21 STEP Arquivo p21
I2
Arquivo neutro STEP-NC (p21)
I1
Usuário & senha
Obter o modelo SDAI do arquivo STEP-NC
A411
SDAI Model
Leitura de entidades e atributos
A412
Present Entities in SDAI Model
Instanciar ojetos Java que equivalen ao arquivo STEP
A413
Project
O1
Internet
Sistema de banco de dados
Sistema CAM
Cliente remoto
M4
M1
M3
M2

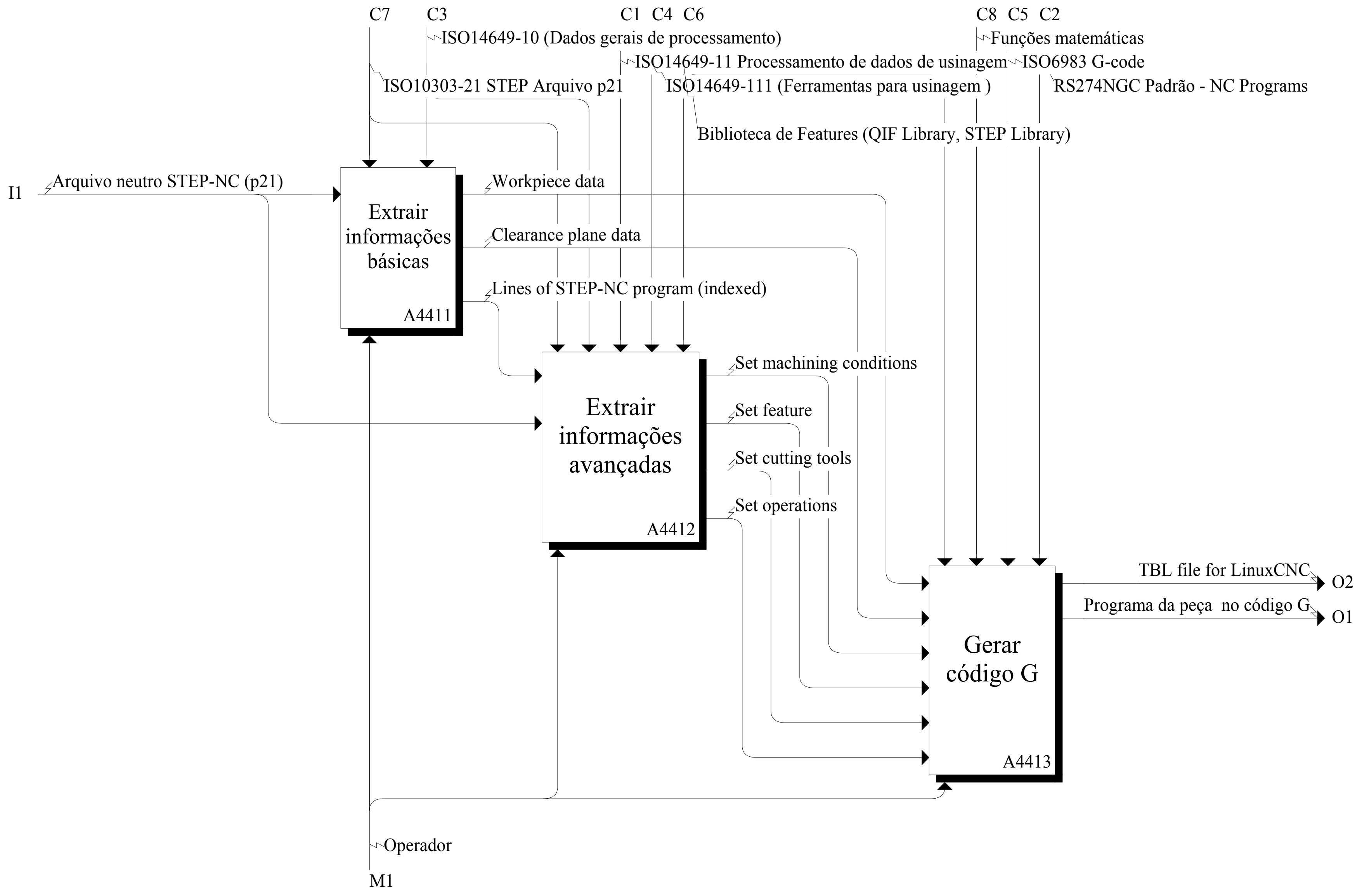
Modelo de integração em malha fechada


C1
ISO6983 G-code
I1
Project
Extrair a etapa de trabalho de usinagem do projeto
A421
Machining Workingstep
Verifique o tipo de feature
A422
Feature
Verifique os parâmetros de corte
A423
Cutting Parameters
Verifique o tipo de operação
A424
Operation
Identifique a ferramenta de corte
A425
Cutting Tool
Gerar os comandos para o CNC
A426
Programa de mecanizado em ISO6983
O1
Internet
Sistema CAM
Sistema de banco de dados
M1
M3
M2

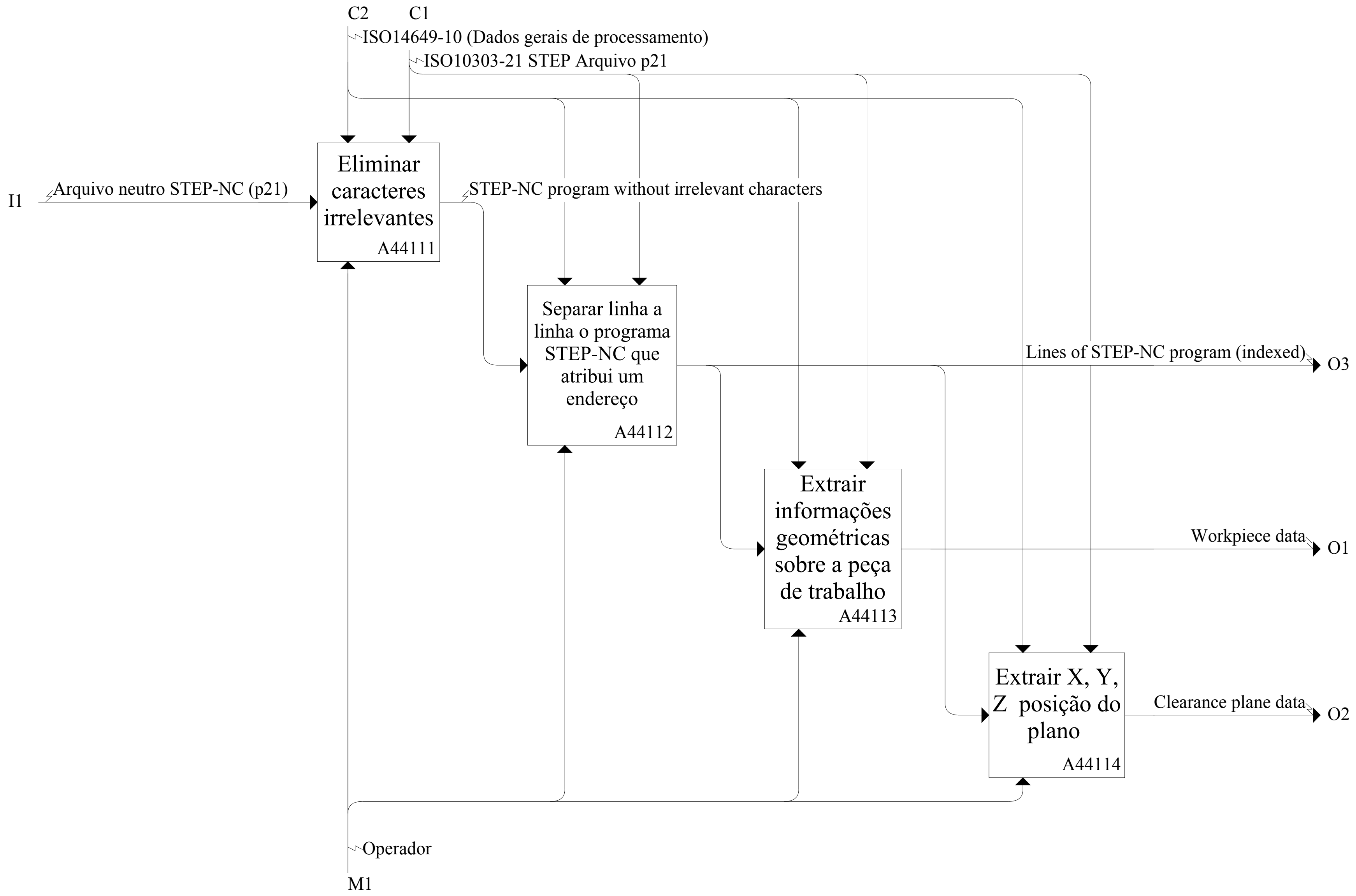
Modelo de integração em malha fechada

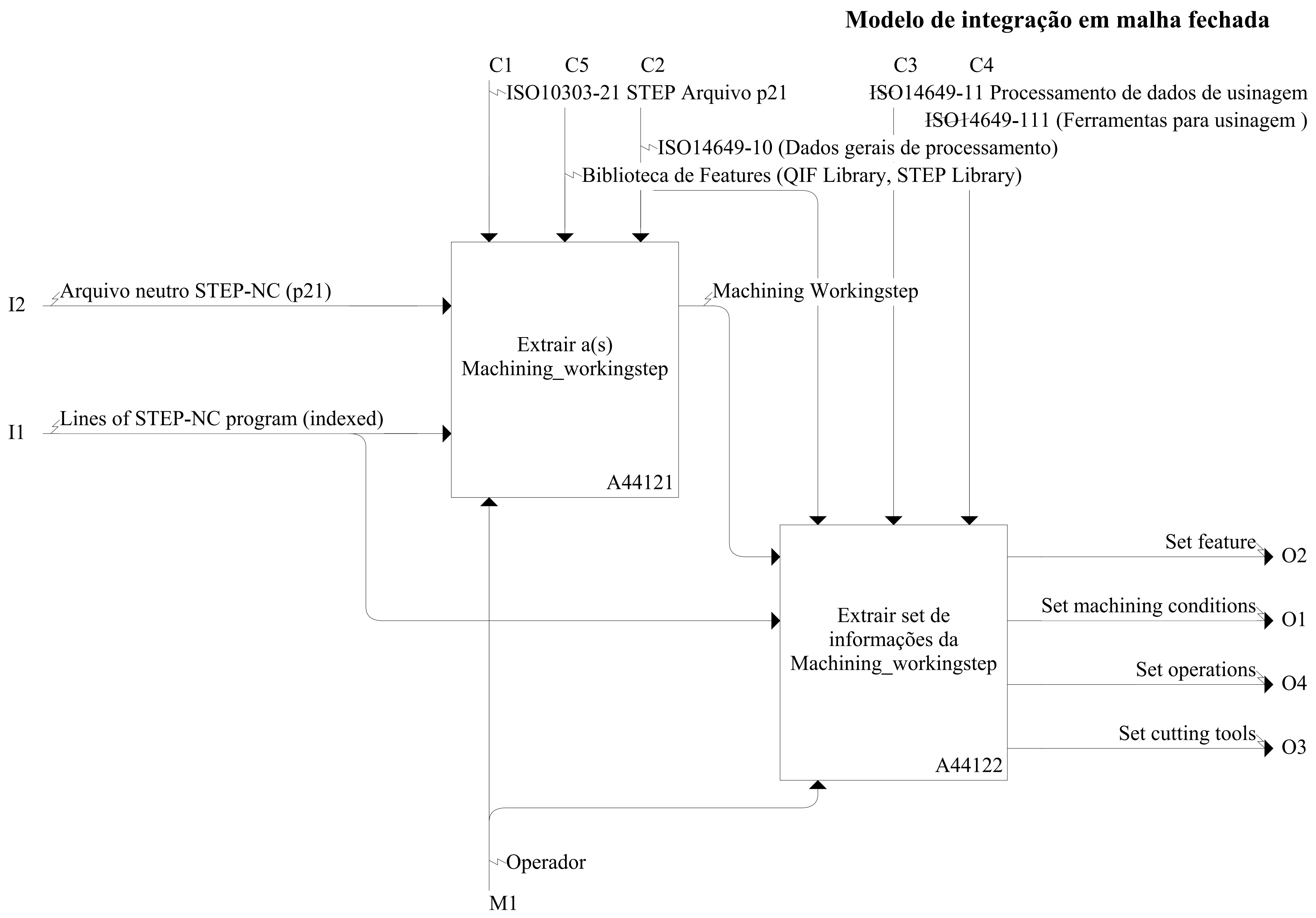

C2
C1
Java 3D (Bibliotecas Gráficas)
Java Swing ((Bibliotecas Gráficas)
I1
Project
Criar projeto de simulação
A431
Simulation project
Determinar o tipo de movimento
A432
Movement Type
Calcular vetores de movimento
A433
Movement Vectors
Generate Visual Representation
A434
Simulação
O1
Internet
Sistema CAM
Sistema de banco de dados
M1
M3
M2

Modelo de integração em malha fechada

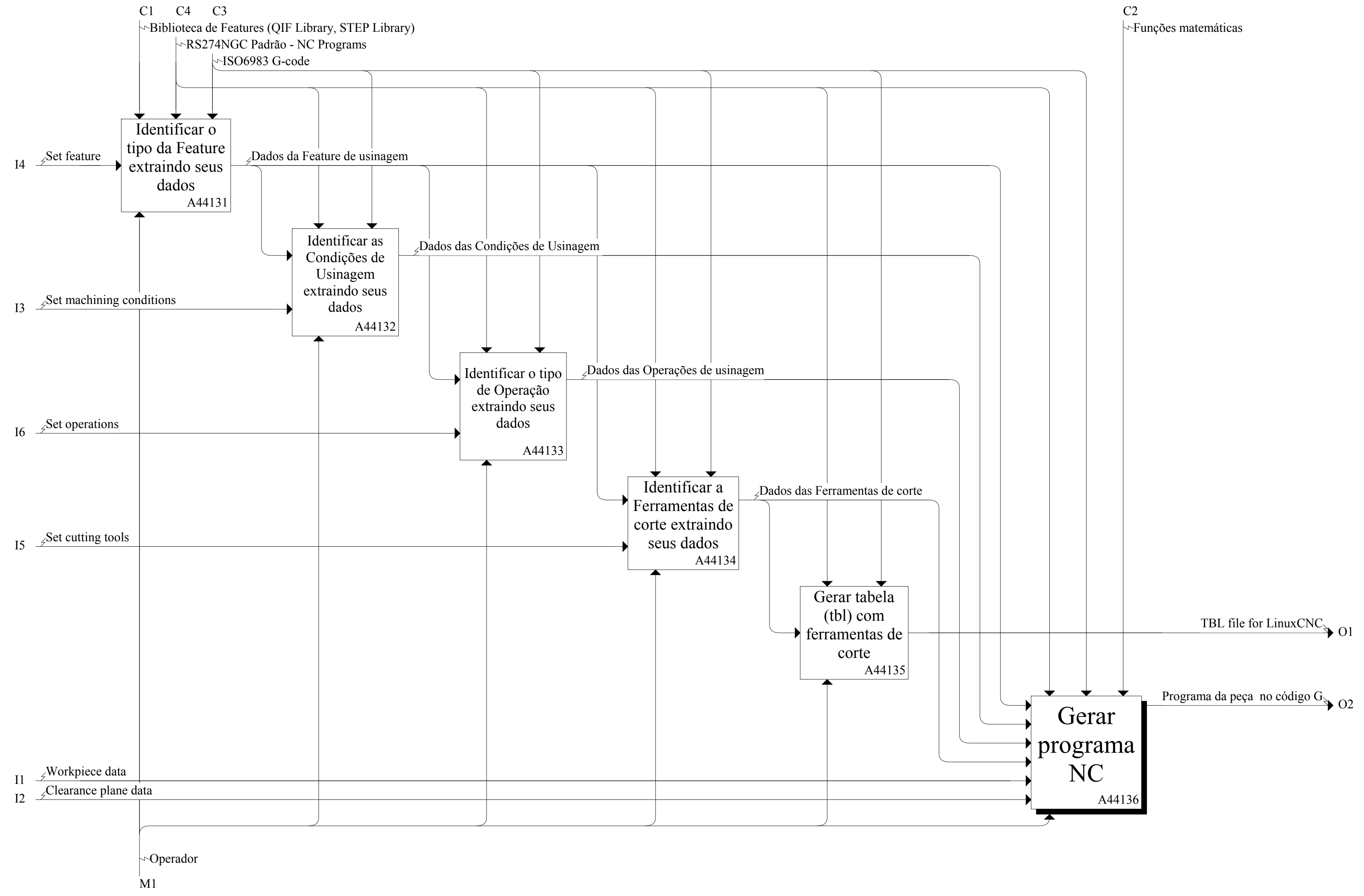

C7
C3
C1
C4
C6
C8
C5
C2
ISO14649-10 (Dados gerais de processamento)
ISO14649-11 Processamento de dados de usinagem
ISO10303-21 STEP Arquivo p21
ISO14649-111 (Ferramentas para usinagem)
Funções matemáticas
ISO6983 G-code
RS274NGC Padrão - NC Programs
Biblioteca de Features (QIF Library, STEP Library)
I1
Arquivo neutro STEP-NC (p21)
Extrair informações básicas
A4411
Workpiece data
Clearance plane data
Lines of STEP-NC program (indexed)
Extrair informações avançadas
A4412
Set machining conditions
Set feature
Set cutting tools
Set operations
Gerar código G
A4413
TBL file for LinuxCNC
O2
Programa da peça no código G
O1
Operador
M1

Modelo de integração em malha fechada

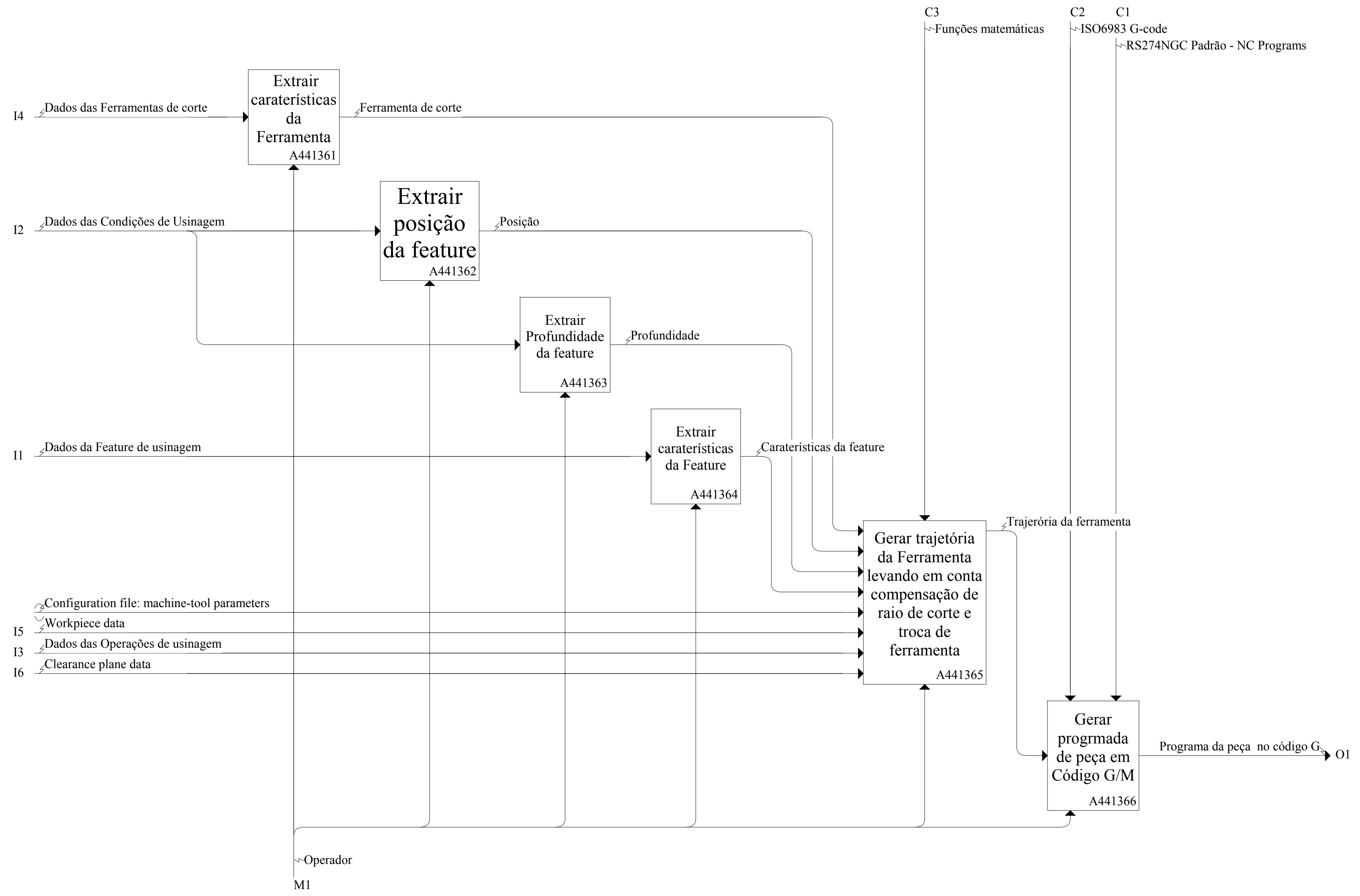

C2
C1
ISO14649-10 (Dados gerais de processamento)
ISO10303-21 STEP Arquivo p21
I1
Arquivo neutro STEP-NC (p21)
Eliminar caracteres irrelevantes
A44111
STEP-NC program without irrelevant characters
Separar linha a linha o programa STEP-NC que atribui um endereço
A44112
Lines of STEP-NC program (indexed)
O3
Extrair informações geométricas sobre a peça de trabalho
A44113
Workpiece data
O1
Extrair X, Y, Z posição do plano
A44114
Clearance plane data
O2
Operador
M1


Modelo de integração em malha fechada
C1
C5
C2
C3
C4
ISO10303-21 STEP Arquivo p21
ISO14649-11 Processamento de dados de usinagem
ISO14649-111 (Ferramentas para usinagem)
ISO14649-10 (Dados gerais de processamento)
Biblioteca de Features (QIF Library, STEP Library)
I2
Arquivo neutro STEP-NC (p21)
Machining Workingstep
Extrair a(s) Machining_workingstep
I1
Lines of STEP-NC program (indexed)
A44121
Set feature
O2
Set machining conditions
O1
Extrair set de informações da Machining_workingstep
Set operations
O4
Set cutting tools
O3
A44122
Operador
M1

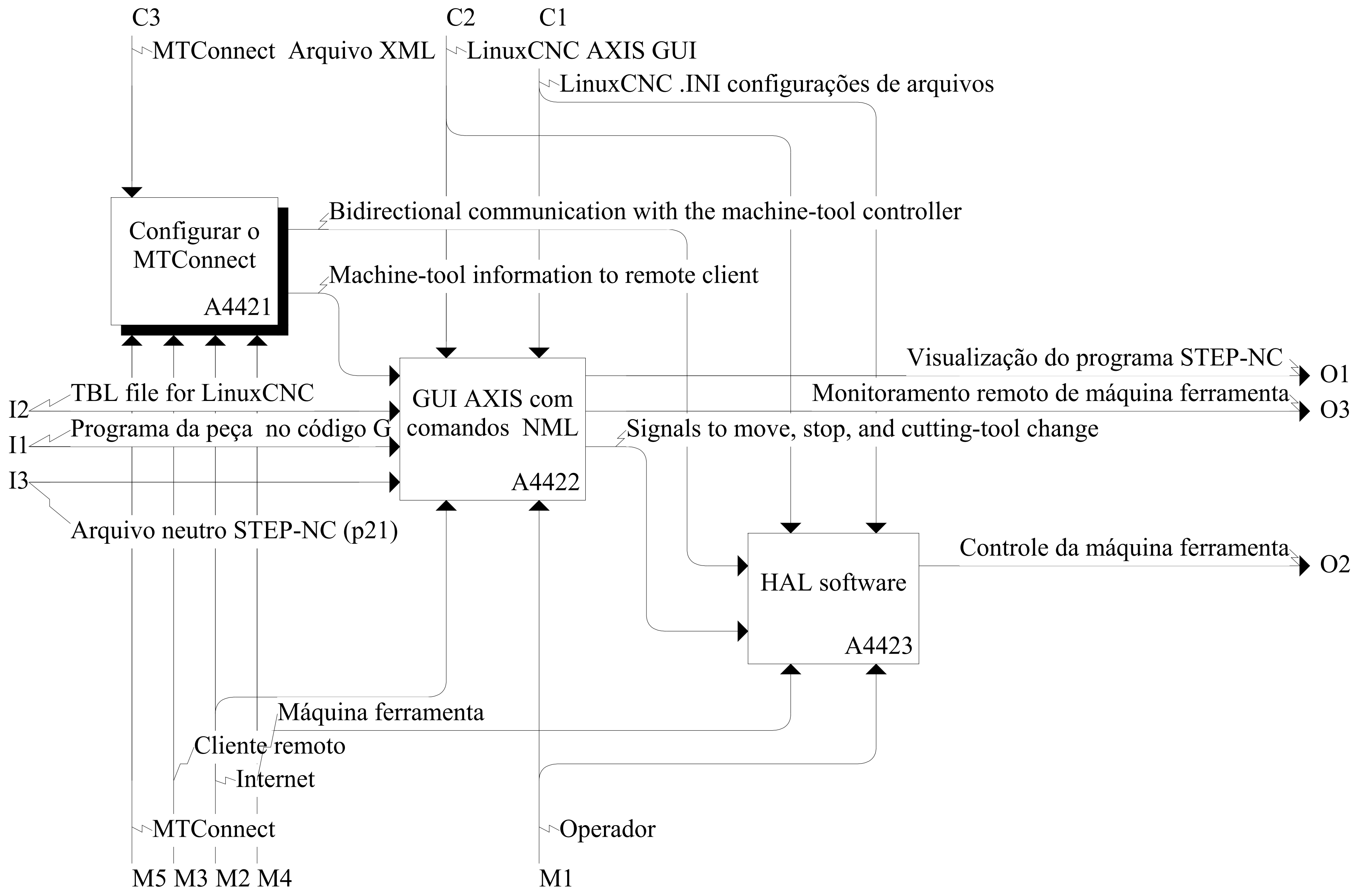
Modelo de integração em malha fechada


C1
C4
C3
Biblioteca de Features (QIF Library, STEP Library)
RS274NGC Padrão - NC Programs
ISO6983 G-code
C2
Funções matemáticas
I4
Set feature
Identificar o tipo da Feature extraindo seus dados
A44131
Dados da Feature de usinagem
I3
Set machining conditions
Identificar as Condições de Usinagem extraindo seus dados
A44132
Dados das Condições de Usinagem
I6
Set operations
Identificar o tipo de Operação extraindo seus dados
A44133
Dados das Operações de usinagem
I5
Set cutting tools
Identificar a Ferramentas de corte extraindo seus dados
A44134
Dados das Ferramentas de corte
Gerar tabela (tbl) com ferramentas de corte
A44135
TBL file for LinuxCNC
O1
Gerar programa NC
A44136
Programa da peça no código G
O2
I1
Workpiece data
I2
Clearance plane data
Operador
M1

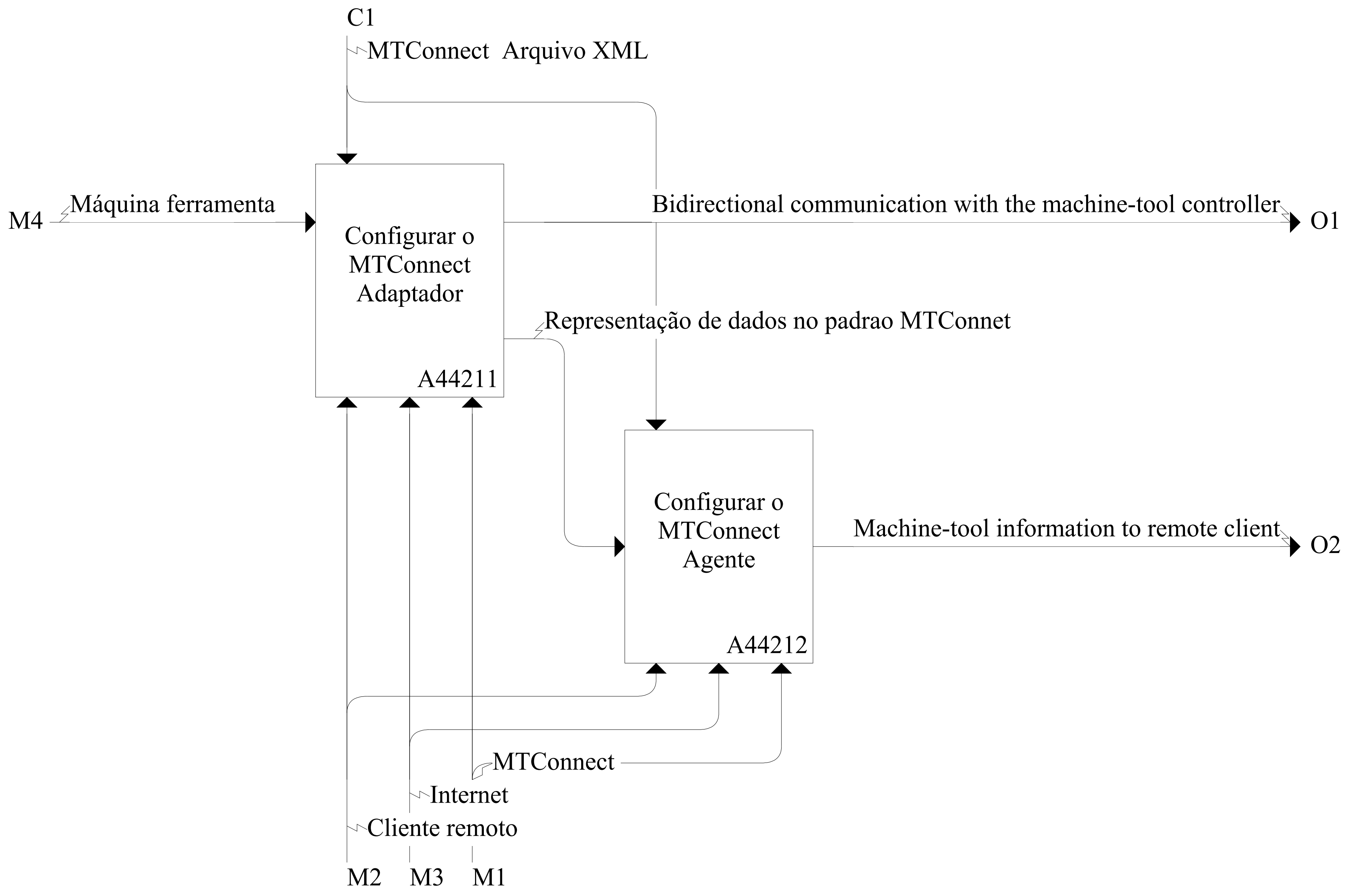
Modelo de integração em malha fechada

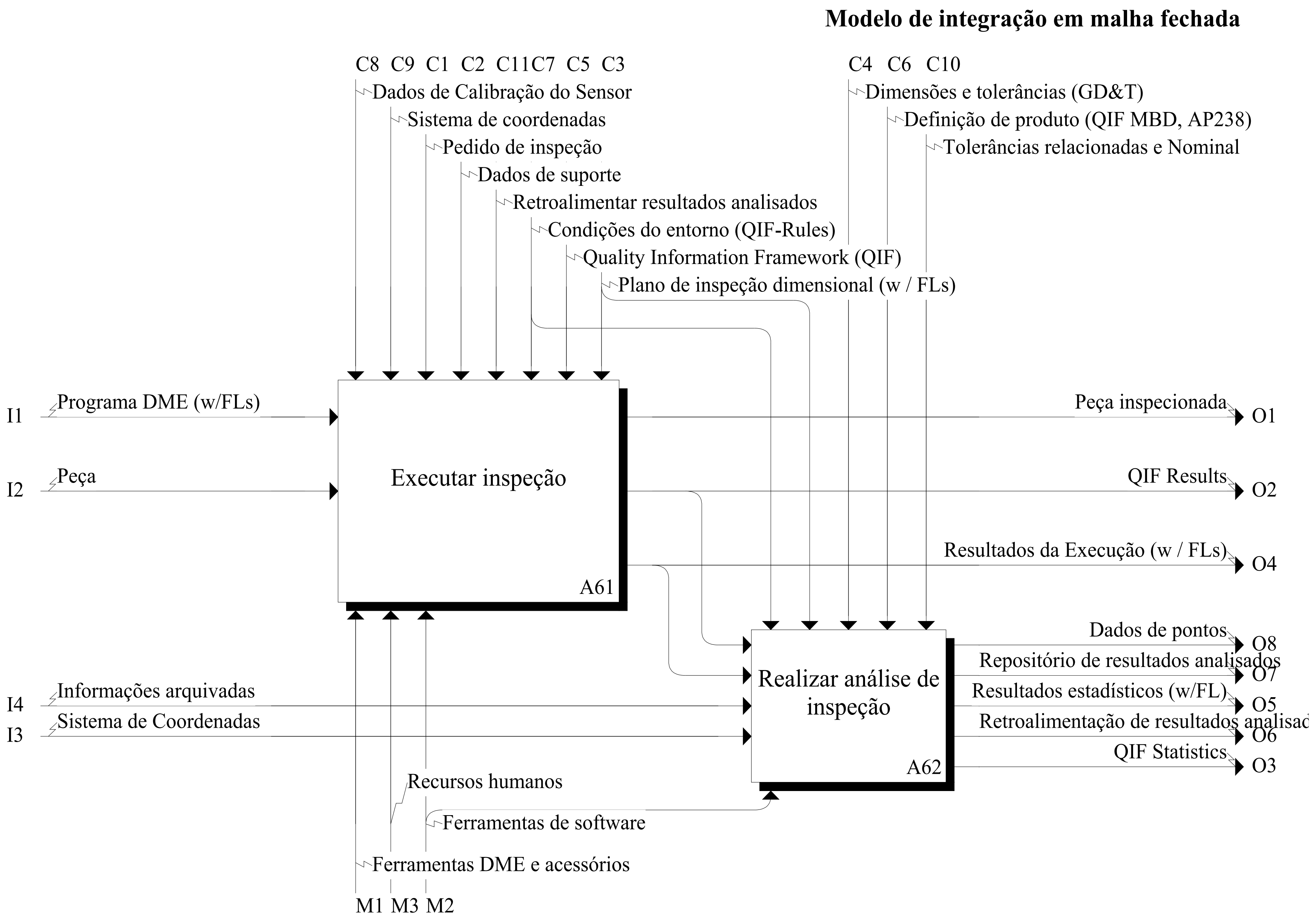

C3
Funções matemáticas
C2
ISO6983 G-code
C1
RS274NGC Padrão - NC Programs
I4
Dados das Ferramentas de corte
Extrair caraterísticas da Ferramenta
A441361
Ferramenta de corte
I2
Dados das Condições de Usinagem
Extrair posição da feature
A441362
Posição
Extrair Profundidade da feature
A441363
Profundidade
I1
Dados da Feature de usinagem
Extrair caraterísticas da Feature
A441364
Caraterísticas da feature
Gerar trajetória da Ferramenta levando em conta compensação de raio de corte e troca de ferramenta
A441365
Trajerória da ferramenta
Configuration file: machine-tool parameters
I5
Workpiece data
I3
Dados das Operações de usinagem
I6
Clearance plane data
Gerar progrmada de peça em Código G/M
A441366
Programa da peça no código G
O1
Operador
M1

Modelo de integração em malha fechada

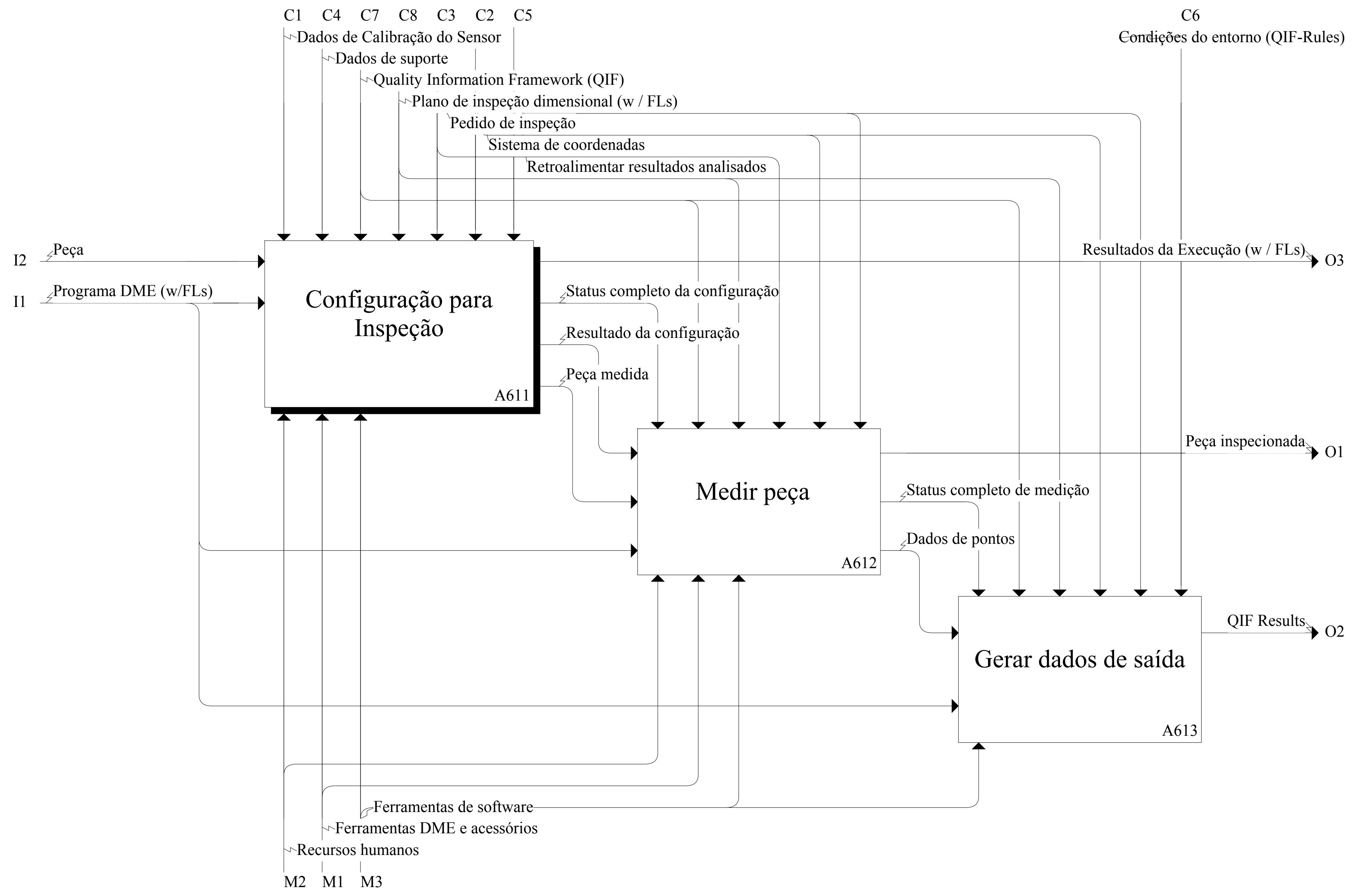

C3
MTConnect Arquivo XML
C2
LinuxCNC AXIS GUI
C1
LinuxCNC .INI configurações de arquivos
Configurar o MTConnect
A4421
Bidirectional communication with the machine-tool controller
Machine-tool information to remote client
GUI AXIS com comandos NML
A4422
Visualização do programa STEP-NC
O1
Monitoramento remoto de máquina ferramenta
O3
Signals to move, stop, and cutting-tool change
I2
TBL file for LinuxCNC
I1
Programa da peça no código G
I3
Arquivo neutro STEP-NC (p21)
HAL software
A4423
Controle da máquina ferramenta
O2
Máquina ferramenta
Cliente remoto
Internet
MTConnect
Operador
M5 M3 M2 M4
M1

Modelo de integração em malha fechada

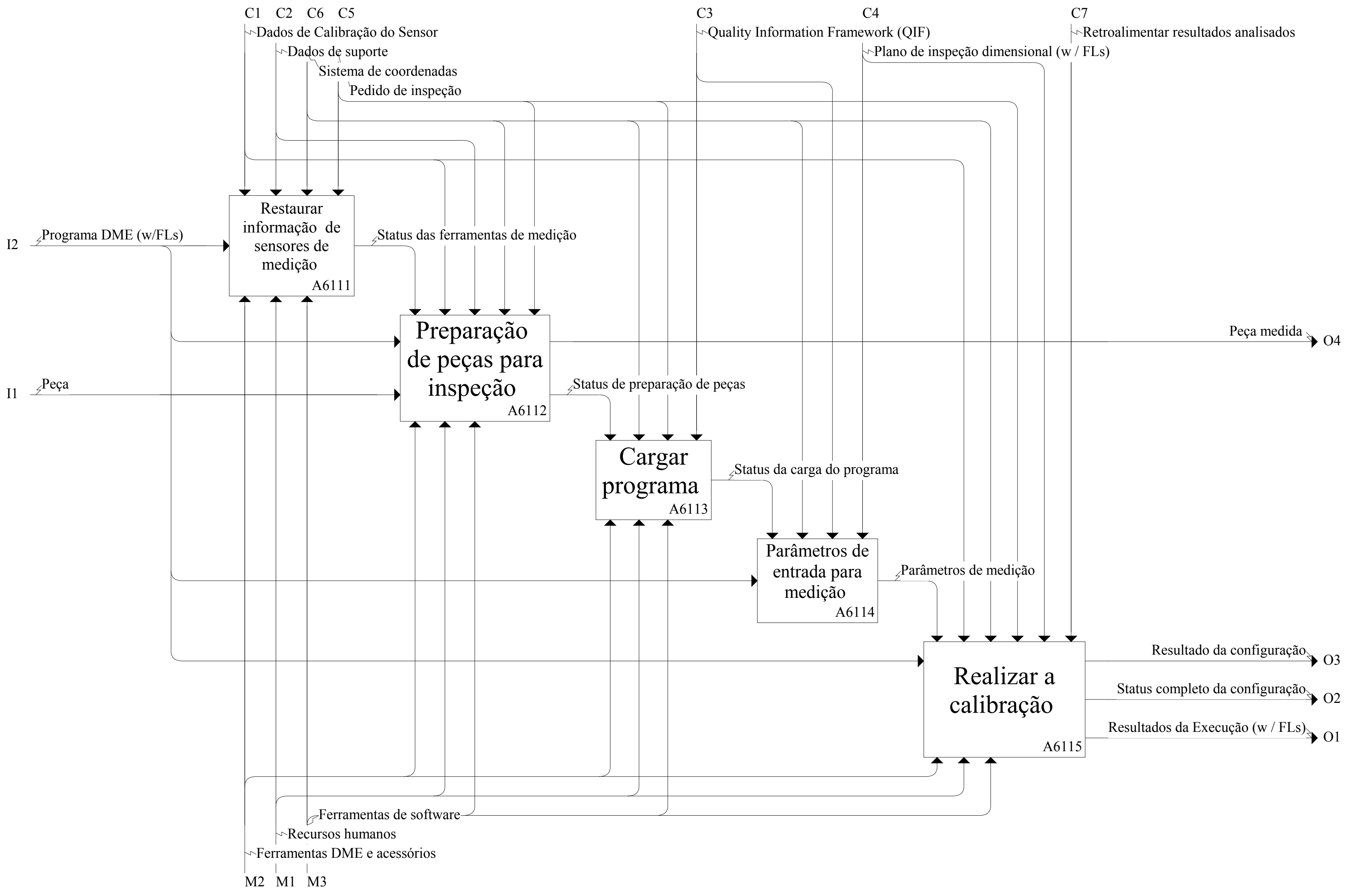

C1
MTConnect Arquivo XML
M4
Máquina ferramenta
Configurar o MTConnect Adaptador
A44211
Bidirectional communication with the machine-tool controller
O1
Representação de dados no padrao MTConnet
Configurar o MTConnect Agente
A44212
Machine-tool information to remote client
O2
MTConnect
Internet
Cliente remoto
M2
M3
M1


Modelo de integração em malha fechada
C8 C9 C1 C2 C11 C7 C5 C3
Dados de Calibração do Sensor
Sistema de coordenadas
Pedido de inspeção
Dados de suporte
Retroalimentar resultados analisados
Condições do entorno (QIF-Rules)
Quality Information Framework (QIF)
Plano de inspeção dimensional (w / FLs)
C4 C6 C10
Dimensões e tolerâncias (GD&T)
Definição de produto (QIF MBD, AP238)
Tolerâncias relacionadas e Nominal
I1
Programa DME (w/FLs)
I2
Peça
Executar inspeção
A61
Peça inspecionada
O1
QIF Results
O2
Resultados da Execução (w / FLs)
O4
Dados de pontos
O8
Repositório de resultados analisados
O7
Resultados estadísticos (w/FL)
O5
Retroalimentação de resultados analisad
O6
QIF Statistics
O3
Realizar análise de inspeção
A62
I4
Informações arquivadas
I3
Sistema de Coordenadas
Recursos humanos
Ferramentas de software
Ferramentas DME e acessórios
M1 M3 M2

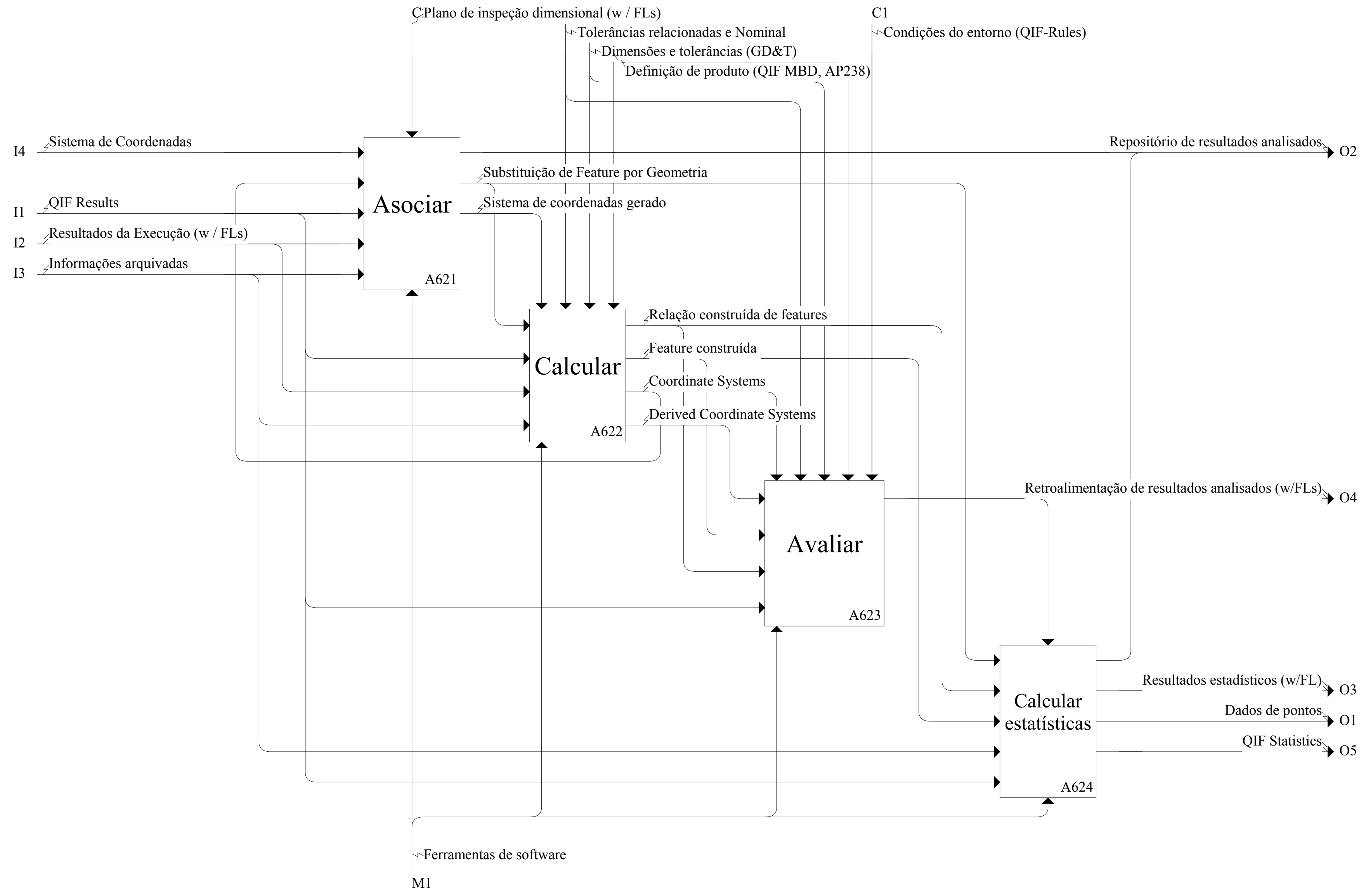
Modelo de integração em malha fechada


C1
C4
C7
C8
C3
C2
C5
C6
Dados de Calibração do Sensor
Dados de suporte
Quality Information Framework (QIF)
Plano de inspeção dimensional (w / FLs)
Pedido de inspeção
Sistema de coordenadas
Retroalimentar resultados analisados
Condições do entorno (QIF-Rules)
I2
Peça
I1
Programa DME (w/FLs)
Configuração para Inspeção
A611
Status completo da configuração
Resultado da configuração
Peça medida
Resultados da Execução (w / FLs)
O3
Medir peça
A612
Peça inspecionada
O1
Status completo de medição
Dados de pontos
Gerar dados de saída
A613
QIF Results
O2
Ferramentas de software
Ferramentas DME e acessórios
Recursos humanos
M2
M1
M3

Modelo de integração em malha fechada


C1
C2
C6
C5
Dados de Calibração do Sensor
Dados de suporte
Sistema de coordenadas
Pedido de inspeção
C3
Quality Information Framework (QIF)
C4
Plano de inspeção dimensional (w / FLs)
C7
Retroalimentar resultados analisados
I2
Programa DME (w/FLs)
Restaurar informação de sensores de medição
A6111
Status das ferramentas de medição
Preparação de peças para inspeção
A6112
Peça medida
O4
I1
Peça
Status de preparação de peças
Cargar programa
A6113
Status da carga do programa
Parâmetros de entrada para medição
A6114
Parâmetros de medição
Realizar a calibração
A6115
Resultado da configuração
O3
Status completo da configuração
O2
Resultados da Execução (w / FLs)
O1
Ferramentas de software
Recursos humanos
Ferramentas DME e acessórios
M2
M1
M3

Modelo de integração em malha fechada


C Plano de inspeção dimensional (w / FLs)
Tolerâncias relacionadas e Nominal
Dimensões e tolerâncias (GD&T)
Definição de produto (QIF MBD, AP238)
C1
Condições do entorno (QIF-Rules)
I4
Sistema de Coordenadas
I1
QIF Results
I2
Resultados da Execução (w / FLs)
I3
Informações arquivadas
Asociar
A621
Substituição de Feature por Geometria
Sistema de coordenadas gerado
Repositório de resultados analisados
O2
Calcular
A622
Relação construída de features
Feature construída
Coordinate Systems
Derived Coordinate Systems
Avaliar
A623
Retroalimentação de resultados analisados (w/FLs)
O4
Calcular estatísticas
A624
Resultados estadísticos (w/FL)
O3
Dados de pontos
O1
QIF Statistics
O5
Ferramentas de software
M1

B.2 Appendix: STEP Neutral File Exchange

B.2.1 STEP File Built Through Implementation

```

ISO-10303-21;
HEADER;
/* Generated by software containing
 * JSDAI (TM) from LKSoft (www.lksoft.com, www.jsdai.net)
 * JSDAI Runtime Version 4.3.0 2011-12-15T17:41:51
 */
FILE_DESCRIPTION(
/* description */ ('Program to generate AP238 p21 file'),
/* implementation_level */ '2;1');
FILE_NAME(
/* name */ ' ',
/* time_stamp */ '2021-04-12T03:53:13',
/* author */ ('Cristhian Ivan Riano Jaimes','Alberto J. Alvares'),
/* organization */ ('Universidade de Brasilia (UnB)'),
/* preprocessor_version */ ' ',
/* originating_system */ 'JSDAI MULTIPLE Version 4.0.0 (Build 270, 2011-12-15T17:42:49)',
/* authorization */ 'cristhianivanrj');
FILE_SCHEMA(('INTEGRATED_CNC_SCHEMA'));
ENDSEC;
DATA;
#1=APPLICATION_CONTEXT('Application protocol for the exchange of CNC data');
#2=PRODUCT_CONTEXT('CNC Machining',#1,'manufacturing');
#3=MACHINING_PROJECT('Project JSDAI Export','','',(#2));
#4=PRODUCT_DEFINITION_FORMATION('','',#3);
#5=PRODUCT_DEFINITION_CONTEXT('CNC Machining',#1,'manufacturing');
#6=PRODUCT_DEFINITION('','',#4,#5);
#7=PRODUCT('default workpiece','AP-238','',(#2));
#8=PRODUCT_DEFINITION_FORMATION('','',#7);
#9=PRODUCT_DEFINITION('','',#8,#5);
#10=MACHINING_PROJECT_WORKPIECE_RELATIONSHIP('','','',#6,#9);
#11=MACHINING_WORKPLAN('Main Workplan','','','');
#12=PRODUCT_DEFINITION_PROCESS('machining','',#11,'');
#13=PROCESS_PRODUCT_ASSOCIATION('','',#6,#12);
#14=PRODUCT_DEFINITION_PROCESS('to-be shape','',#11,'');
#15=PROCESS_PRODUCT_ASSOCIATION('','',#9,#14);
#16=MACHINING_WORKPLAN('Main Workplan','','','');
#17=MACHINING_PROCESS_SEQUENCE_RELATIONSHIP('','',#11,#16,1.0);
#18=MACHINING_WORKINGSTEP('Operation 1 - Facing WS 1 WS 1','machining','','');
#19=MACHINING_PROCESS_SEQUENCE_RELATIONSHIP('','',#16,#18,1.0);
#20=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);
#21=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);
#22=(NAMED_UNIT(*)SI_UNIT($,.STERADIAN.)SOLID_ANGLE_UNIT());
#23=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(0.01745329252),#22);
#24=(CONVERSION_BASED_UNIT('degree',#23)NAMED_UNIT(#21)PLANE_ANGLE_UNIT());
#25=(LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
#26=(NAMED_UNIT(*)PLANE_ANGLE_UNIT()SI_UNIT($,.RADIAN.));
#27=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(25.4),#25);
#28=(CONVERSION_BASED_UNIT('inch',#27)LENGTH_UNIT()NAMED_UNIT(#21));
#29=(GEOMETRIC_REPRESENTATION_CONTEXT(3)GLOBAL_UNIT_ASSIGNED_CONTEXT((#28,#24,#26))
        REPRESENTATION_CONTEXT('INCH DEGREE STERADIAN','3D'));
```

```
#30=DIMENSIONAL_EXPONENTS(0.0,0.0,1.0,0.0,0.0,0.0,0.0);
#31=(NAMED_UNIT(*)SI_UNIT($,.SECOND.)TIME_UNIT());
#32=TIME_MEASURE_WITH_UNIT(TIME_MEASURE(60.0),#31);
#33=(CONVERSION_BASED_UNIT('',#32)NAMED_UNIT(#30)TIME_UNIT());
#34=DERIVED_UNIT_ELEMENT(#33,-1.0);
#35=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);
#36=CONTEXT_DEPENDENT_UNIT(#35,'revolution');
#37=DERIVED_UNIT_ELEMENT(#36,1.0);
#38=DERIVED_UNIT((#37,#34));
#39=MEASURE_REPRESENTATION_ITEM('rotational speed',NUMERIC_MEASURE(0.0),#38);
#40=MACHINING_SPINDLE_SPEED_REPRESENTATION('spindle speed',(#39),#29);
#41=MEASURE_REPRESENTATION_ITEM('rotational speed',NUMERIC_MEASURE(-1591.0),#38);
#42=MACHINING_SPINDLE_SPEED_REPRESENTATION('spindle speed',(#41),#29);
#43=MEASURE_REPRESENTATION_ITEM('rotational speed',NUMERIC_MEASURE(-1591.0),#38);
#44=MACHINING_SPINDLE_SPEED_REPRESENTATION('spindle speed',(#43),#29);
#45=CARTESIAN_POINT('',(0.0,0.0,0.0));
#46=DIRECTION('',(0.0,0.0,1.0));
#47=DIRECTION('',(1.0,0.0,0.0));
#48=AXIS2_PLACEMENT_3D('toolpath orientation',#45,#46,#47);
#49=REPRESENTATION('',(#48),#29);
#50=ACTION_PROPERTY('toolpath orientation','machining',#18);
#51=ACTION_PROPERTY_REPRESENTATION(' ','machining',#50,#49);
#52=FREEFORM_MILLING_OPERATION('start point','','','');
#53=MACHINING_OPERATION_RELATIONSHIP('','machining',#18,#52);
#54=MACHINING_FEATURE_PROCESS('','machining','','');
#55=MACHINING_FEATURE_RELATIONSHIP('','machining',#18,#54);
#56=PROPERTY_PROCESS('machining','machining',#54,'');
#57=INSTANCED_FEATURE($,$,$,$,$,$);
#58=PROCESS_PROPERTY_ASSOCIATION('','',#56,#57);
#59=MACHINING_FUNCTIONS('','milling','','');
#60=ACTION_PROPERTY('chip removal','milling',#59);
#61=DESCRIPTIVE_REPRESENTATION_ITEM('constant','coolant off');
#62=REPRESENTATION('constant',(#61),#29);
#63=ACTION_PROPERTY_REPRESENTATION('','milling',#60,#62);
#64=ACTION_PROPERTY('through spindle coolant','milling',#59);
#65=DESCRIPTIVE_REPRESENTATION_ITEM('constant','coolant off');
#66=REPRESENTATION('constant',(#65),#29);
#67=ACTION_PROPERTY_REPRESENTATION('','milling',#64,#66);
#68=DESCRIPTIVE_REPRESENTATION_ITEM('constant','coolant off');
#69=REPRESENTATION('constant',(#68),#29);
#70=ACTION_PROPERTY('coolant','milling',#59);
#71=ACTION_PROPERTY_REPRESENTATION('','milling',#70,#69);
#72=MACHINING_TECHNOLOGY('','milling','','');
#73=MACHINING_TECHNOLOGY_RELATIONSHIP('','',#52,#72);
#74=MACHINING_TOOLPATH('start point WS 1 TP 1','cutter location trajectory','','');
#75=MACHINING_TOOLPATH_SEQUENCE_RELATIONSHIP('','',#52,#74,1.0);
#76=MACHINING_TOOLPATH('WS 1 TP 2','cutter location trajectory','','');
#77=MACHINING_TOOLPATH_SEQUENCE_RELATIONSHIP('','',#52,#76,2.0);
#78=MACHINING_TOOLPATH('WS 1 TP 3','cutter location trajectory','','');
#79=MACHINING_TOOLPATH_SEQUENCE_RELATIONSHIP('','',#52,#78,3.0);
#80=MACHINING_TOOLPATH('WS 1 TP 4','cutter location trajectory','','');
#81=MACHINING_TOOLPATH_SEQUENCE_RELATIONSHIP('','',#52,#80,4.0);
#82=MACHINING_FUNCTIONS_RELATIONSHIP('','',#52,#59);
#83=ACTION_RESOURCE_TYPE('milling cutting tool');
#84=MACHINING_TOOL('0','user defined milling tool',(#52),#83);
#85=RESOURCE_PROPERTY('tool body','user defined milling tool',#84);
#86=(LENGTH_MEASURE_WITH_UNIT()MEASURE_REPRESENTATION_ITEM()MEASURE_WITH_UNIT(LENGTH_MEASURE(
      0.0),#28)REPRESENTATION_ITEM('effective cutting diameter'));
#87=MACHINING_TOOL_BODY_REPRESENTATION('',(#86),#29);
#88=RESOURCE_PROPERTY_REPRESENTATION('effective cutting diameter','user defined milling tool',
    #85,#87);
#89=RESOURCE_PROPERTY('tool body','user defined milling tool',#84);
#90=(LENGTH_MEASURE_WITH_UNIT()MEASURE_REPRESENTATION_ITEM()MEASURE_WITH_UNIT(LENGTH_MEASURE(
    0.0),#28)REPRESENTATION_ITEM('corner radius center vertical'));
#91=MACHINING_TOOL_BODY_REPRESENTATION('',(#90),#29);
#92=RESOURCE_PROPERTY_REPRESENTATION('corner radius center vertical','user defined milling tool',
    #89,#87);
#93=RESOURCE_PROPERTY('tool body','user defined milling tool',#84);
#94=(LENGTH_MEASURE_WITH_UNIT()MEASURE_REPRESENTATION_ITEM()MEASURE_WITH_UNIT(LENGTH_MEASURE(
    0.0),#28)REPRESENTATION_ITEM('corner radius'));
#95=MACHINING_TOOL_BODY_REPRESENTATION('',(#94),#29);
#96=RESOURCE_PROPERTY_REPRESENTATION('corner radius','user defined milling tool',
    #93,#95);
#97=RESOURCE_PROPERTY('tool body','user defined milling tool',#84);
#98=(LENGTH_MEASURE_WITH_UNIT()MEASURE_REPRESENTATION_ITEM()MEASURE_WITH_UNIT(LENGTH_MEASURE(
    0.0),#28)REPRESENTATION_ITEM('corner radius'));
#99=MACHINING_TOOL_BODY_REPRESENTATION('',(#98),#29);
#100=RESOURCE_PROPERTY_REPRESENTATION('maximum depth of cut','user defined milling tool',
    #97,#99);
#101=RESOURCE_PROPERTY('tool body','user defined milling tool',#84);
#102=(LENGTH_MEASURE_WITH_UNIT()MEASURE_REPRESENTATION_ITEM()MEASURE_WITH_UNIT(LENGTH_MEASURE(
    0.0),#28)REPRESENTATION_ITEM('corner radius'));
#103=MACHINING_TOOL_BODY_REPRESENTATION('',(#98),#29);
#104=RESOURCE_PROPERTY_REPRESENTATION('corner radius center horizontal','user defined milling tool',
    #101,#103);
#105=RESOURCE_PROPERTY('tool body','user defined milling tool',#84);
#106=(LENGTH_MEASURE_WITH_UNIT()MEASURE_REPRESENTATION_ITEM()MEASURE_WITH_UNIT(LENGTH_MEASURE(
    0.0),#28)REPRESENTATION_ITEM('overall assembly length'));
#107=MACHINING_TOOL_BODY_REPRESENTATION('',(#106),#29);
#108=RESOURCE_PROPERTY_REPRESENTATION('overall assembly length','user defined milling tool',
    #105,#107);
#109=ACTION_PROPERTY('spindle','milling',#72);
#110=ACTION_PROPERTY_REPRESENTATION('rotational speed','milling',#109,#40);
#111=ACTION_PROPERTY('feedrate','milling',#72);
#112=DERIVED_UNIT_ELEMENT(#33,-1.0);
#113=DERIVED_UNIT_ELEMENT(#28,1.0);
#114=DERIVED_UNIT((#113,#112));
#115=MEASURE_REPRESENTATION_ITEM('feed speed',NUMERIC_MEASURE(0.0),#114);
#116=MACHINING_FEED_SPEED_REPRESENTATION('feed speed',(#115),#29);
#117=ACTION_PROPERTY_REPRESENTATION('feed speed','milling',#111,#116);
#118=MACHINING_TECHNOLOGY('','milling','','');
#119=ACTION_PROPERTY('spindle','milling',#118);
#120=ACTION_PROPERTY_REPRESENTATION('rotational speed','milling',#119,#42);
#121=ACTION_PROPERTY('feedrate','milling',#118);
#122=MEASURE_REPRESENTATION_ITEM('feed speed',NUMERIC_MEASURE(3.13188976377953),
    #114);
#123=MACHINING_FEED_SPEED_REPRESENTATION('feed speed',(#122),#29);
#124=ACTION_PROPERTY_REPRESENTATION('feed speed','milling',#121,#123);
#125=MACHINING_TECHNOLOGY('','milling','','');
#126=ACTION_PROPERTY('spindle','milling',#125);
#127=ACTION_PROPERTY_REPRESENTATION('rotational speed','milling',#126,#44);
#128=ACTION_PROPERTY('feedrate','milling',#125);
#129=MEASURE_REPRESENTATION_ITEM('feed speed',NUMERIC_MEASURE(6.26377952755905),
    #114);
#130=MACHINING_FEED_SPEED_REPRESENTATION('feed speed',(#129),#29);
#131=ACTION_PROPERTY_REPRESENTATION('feed speed','milling',#128,#130);
#132=ACTION_PROPERTY('basic curve','cutter location trajectory',#74);
#133=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.433070866141732));
#134=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.433070866141732));
#135=POLYLINE('',(#134,#133));
```

```
#136=REPRESENTATION('',(#135),#29);
#137=ACTION_PROPERTY_REPRESENTATION('','cutter location trajectory',#132,#136);
#138=ACTION_PROPERTY('speed profile','rapid',#74);
#139=DESCRIPTIVE_REPRESENTATION_ITEM('','rapid');
#140=MACHINING_TOOLPATH_SPEED_PROFILE_REPRESENTATION('',(#139),#29);
#141=ACTION_PROPERTY_REPRESENTATION('','rapid',#138,#140);
#142=MACHINING_TECHNOLOGY_RELATIONSHIP('','cutter location trajectory',#74,#72);
#143=ACTION_PROPERTY('basic curve','cutter location trajectory',#76);
#144=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.0393700787401575));
#145=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.433070866141732));
#146=POLYLINE('',(#133,#145,#144));
#147=REPRESENTATION('',(#146),#29);
#148=ACTION_PROPERTY_REPRESENTATION('','cutter location trajectory',#143,#147);
#149=MACHINING_TECHNOLOGY_RELATIONSHIP('','cutter location trajectory',#76,#118);
#150=ACTION_PROPERTY('basic curve','cutter location trajectory',#78);
#151=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.0393700787401575));
#152=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.0393700787401575));
#153=CARTESIAN_POINT('',(6.04724409448819,5.90543307086614,0.0393700787401575));
#154=CARTESIAN_POINT('',(6.04724409448819,5.73670866141732,0.0393700787401575));
#155=CARTESIAN_POINT('',(-0.141732283464567,5.73670866141732,0.0393700787401575));
#156=CARTESIAN_POINT('',(-0.141732283464567,5.5679842519685,0.0393700787401575));
#157=CARTESIAN_POINT('',(6.04724409448819,5.5679842519685,0.0393700787401575));
#158=CARTESIAN_POINT('',(6.04724409448819,5.39925984251968,0.0393700787401575));
#159=CARTESIAN_POINT('',(-0.141732283464567,5.39925984251968,0.0393700787401575));
#160=CARTESIAN_POINT('',(-0.141732283464567,5.23053543307087,0.0393700787401575));
#161=CARTESIAN_POINT('',(6.04724409448819,5.23053543307087,0.0393700787401575));
#162=CARTESIAN_POINT('',(6.04724409448819,5.23053543307087,0.0393700787401575));
#163=CARTESIAN_POINT('',(-0.141732283464567,5.06181102362205,0.0393700787401575));
#164=CARTESIAN_POINT('',(-0.141732283464567,4.89308661417323,0.0393700787401575));
#165=CARTESIAN_POINT('',(6.04724409448819,4.89308661417323,0.0393700787401575));
#166=CARTESIAN_POINT('',(6.04724409448819,4.89308661417323,0.0393700787401575));
#167=CARTESIAN_POINT('',(-0.141732283464567,4.72436220472441,0.0393700787401575));
#168=CARTESIAN_POINT('',(-0.141732283464567,4.55563779527559,0.0393700787401575));
#169=CARTESIAN_POINT('',(6.04724409448819,4.55563779527559,0.0393700787401575));
#170=CARTESIAN_POINT('',(6.04724409448819,4.38691338582677,0.0393700787401575));
#171=CARTESIAN_POINT('',(-0.141732283464567,4.38691338582677,0.0393700787401575));
#172=CARTESIAN_POINT('',(-0.141732283464567,4.21818897637795,0.0393700787401575));
#173=CARTESIAN_POINT('',(6.04724409448819,4.21818897637795,0.0393700787401575));
#174=CARTESIAN_POINT('',(6.04724409448819,4.04946456692913,0.0393700787401575));
#175=CARTESIAN_POINT('',(-0.141732283464567,4.04946456692913,0.0393700787401575));
#176=CARTESIAN_POINT('',(-0.141732283464567,3.88074015748032,0.0393700787401575));
#177=CARTESIAN_POINT('',(6.04724409448819,3.88074015748032,0.0393700787401575));
#178=CARTESIAN_POINT('',(6.04724409448819,3.7120157480315,0.0393700787401575));
#179=CARTESIAN_POINT('',(-0.141732283464567,3.7120157480315,0.0393700787401575));
#180=CARTESIAN_POINT('',(6.04724409448819,4.38691338582677,0.0393700787401575));
#181=CARTESIAN_POINT('',(6.04724409448819,3.54329133858268,0.0393700787401575));
#182=CARTESIAN_POINT('',(6.04724409448819,3.37456692913386,0.0393700787401575));
#183=CARTESIAN_POINT('',(6.04724409448819,4.21818897637795,0.0393700787401575));
#184=CARTESIAN_POINT('',(-0.141732283464567,3.20584251968504,0.0393700787401575));
#185=CARTESIAN_POINT('',(6.04724409448819,3.20584251968504,0.0393700787401575));
#186=CARTESIAN_POINT('',(6.04724409448819,3.03711811023622,0.0393700787401575));
#187=CARTESIAN_POINT('',(-0.141732283464567,3.03711811023622,0.0393700787401575));
#188=CARTESIAN_POINT('',(-0.141732283464567,2.8683937007874,0.0393700787401575));
#189=CARTESIAN_POINT('',(6.04724409448819,2.8683937007874,0.0393700787401575));
#190=CARTESIAN_POINT('',(6.04724409448819,2.69966929133858,0.0393700787401575));
#191=CARTESIAN_POINT('',(-0.141732283464567,2.69966929133858,0.0393700787401575));
#192=CARTESIAN_POINT('',(-0.141732283464567,2.53094488188976,0.0393700787401575));
#193=CARTESIAN_POINT('',(6.04724409448819,2.53094488188976,0.0393700787401575));
#194=CARTESIAN_POINT('',(6.04724409448819,2.36222047244094,0.0393700787401575));
#195=CARTESIAN_POINT('',(-0.141732283464567,2.36222047244094,0.0393700787401575));
```

```
#196=CARTESIAN_POINT('',(-0.141732283464567,2.19349606299213,0.0393700787401575));
#197=CARTESIAN_POINT('',(-0.141732283464567,3.03711811023622,0.0393700787401575));
#198=CARTESIAN_POINT('',(6.04724409448819,2.02477165354331,0.0393700787401575));
#199=CARTESIAN_POINT('',(-0.141732283464567,2.02477165354331,0.0393700787401575));
#200=CARTESIAN_POINT('',(-0.141732283464567,1.85604724409449,0.0393700787401575));
#201=CARTESIAN_POINT('',(6.04724409448819,1.85604724409449,0.0393700787401575));
#202=CARTESIAN_POINT('',(6.04724409448819,1.68732283464567,0.0393700787401575));
#203=CARTESIAN_POINT('',(-0.141732283464567,1.68732283464567,0.0393700787401575));
#204=CARTESIAN_POINT('',(-0.141732283464567,1.51859842519685,0.0393700787401575));
#205=CARTESIAN_POINT('',(6.04724409448819,1.51859842519685,0.0393700787401575));
#206=CARTESIAN_POINT('',(6.04724409448819,1.34987401574803,0.0393700787401575));
#207=CARTESIAN_POINT('',(-0.141732283464567,1.34987401574803,0.0393700787401575));
#208=CARTESIAN_POINT('',(-0.141732283464567,1.18114960629921,0.0393700787401575));
#209=CARTESIAN_POINT('',(6.04724409448819,1.18114960629921,0.0393700787401575));
#210=CARTESIAN_POINT('',(6.04724409448819,1.01242519685039,0.0393700787401575));
#211=CARTESIAN_POINT('',(-0.141732283464567,1.01242519685039,0.0393700787401575));
#212=CARTESIAN_POINT('',(-0.141732283464567,0.843700787401575,0.0393700787401575));
#213=CARTESIAN_POINT('',(6.04724409448819,0.843700787401575,0.0393700787401575));
#214=CARTESIAN_POINT('',(6.04724409448819,0.674976377952756,0.0393700787401575));
#215=CARTESIAN_POINT('',(-0.141732283464567,0.67497637795275,0.0393700787401575));
#216=CARTESIAN_POINT('',(-0.141732283464567,0.506251968503937,0.0393700787401575));
#217=CARTESIAN_POINT('',(6.04724409448819,0.506251968503937,0.0393700787401575));
#218=CARTESIAN_POINT('',(6.04724409448819,0.337527559055118,0.0393700787401575));
#219=CARTESIAN_POINT('',(-0.141732283464567,0.337527559055119,0.0393700787401575));
#220=CARTESIAN_POINT('',(-0.141732283464567,0.1688031496063,0.0393700787401575));
#221=CARTESIAN_POINT('',(6.04724409448819,0.168803149606299,0.0393700787401575));
#222=CARTESIAN_POINT('',(6.04724409448819,7.87401574802299E-5,0.0393700787401575));
#223=CARTESIAN_POINT('',(-0.259842519685039,7.87401574810024E-5,0.0393700787401575));
#224=POLYLINE('',(#151,#152,#153,#154,#155,#156,#157,#158,#159,#160,#161,#162,#163,
        #164,#165,#166,#167,#168,#169,#170,#171,#172,#173,#174,#175,#176,#177,#178,
        #179,#180,#181,#182,#183,#184,#185,#186,#187,#188,#189,#190,#191,#192,#193,
        #194,#195,#196,#197,#198,#199,#200,#201,#202,#203,#204,#205,#206,#207,#208,
        #209,#210,#211,#212,#213,#214,#215,#216,#217,#218,#219,#220,#221,#222,#223));
#225=REPRESENTATION('',(#224),#29);
#226=ACTION_PROPERTY_REPRESENTATION('','cutter location trajectory',#150,#225);
#227=MACHINING_TECHNOLOGY_RELATIONSHIP('','cutter location trajectory',#78,#125);
#228=ACTION_PROPERTY('basic curve','cutter location trajectory',#80);
#229=CARTESIAN_POINT('',(-0.259842519685039,7.87401574810024E-5,0.0393700787401575));
#230=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.0393700787401575));
#231=POLYLINE('',(#223,#229,#230));
#232=REPRESENTATION('',(#231),#29);
#233=ACTION_PROPERTY_REPRESENTATION('','cutter location trajectory',#228,#232);
#234=ACTION_PROPERTY('speed profile','rapid',#80);
#235=DESCRIPTIVE_REPRESENTATION_ITEM('','rapid');
#236=MACHINING_TOOLPATH_SPEED_PROFILE_REPRESENTATION('',(#235),#29);
#237=ACTION_PROPERTY_REPRESENTATION('','rapid',#234,#236);
#238=MACHINING_TECHNOLOGY_RELATIONSHIP('','cutter location trajectory',#80,#125);
ENDSEC;
END-ISO-10303-21;
```

B.2.2 A Snippet of STEP File fed With Instances Through STEP Machine

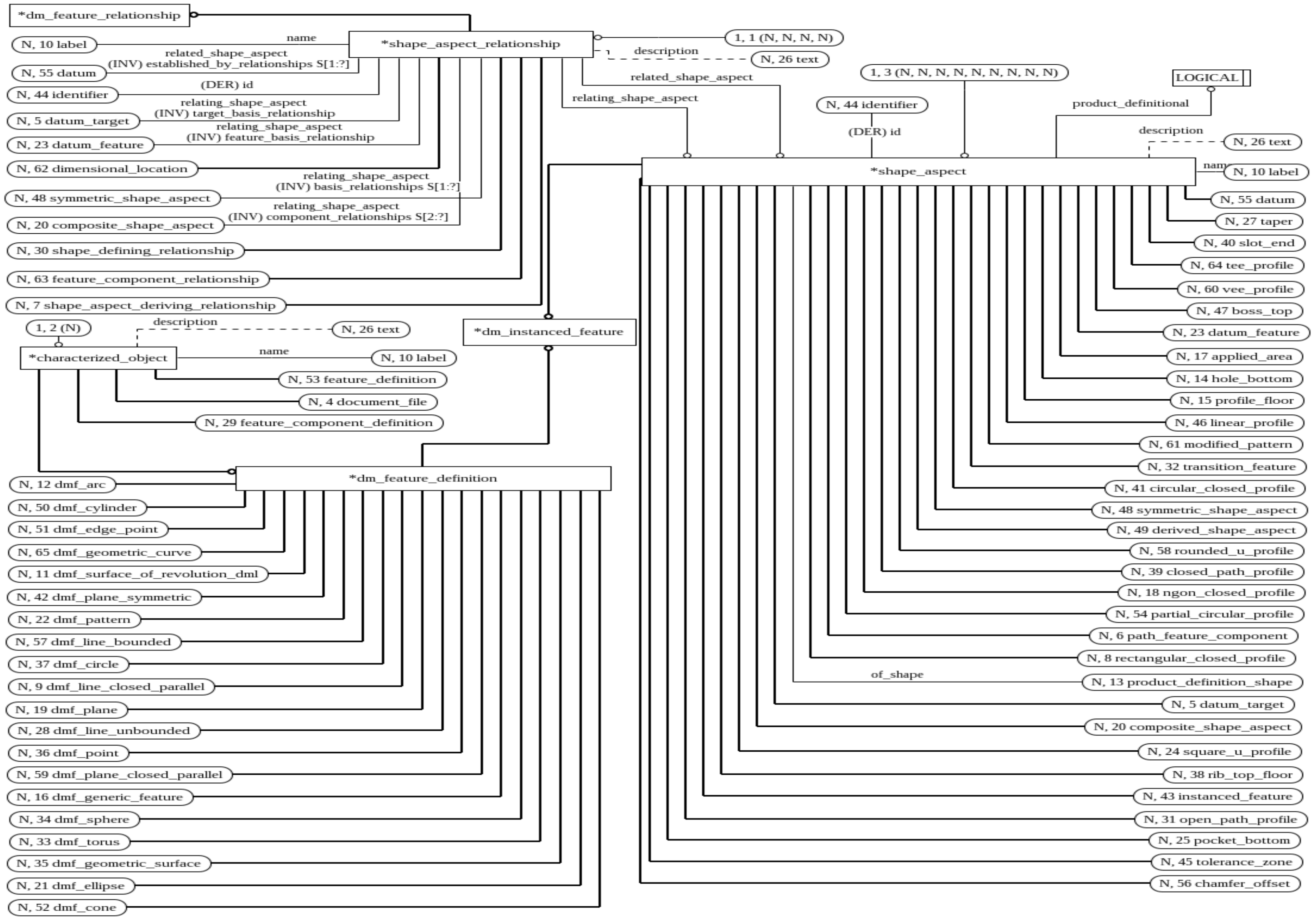
```
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
```

```
 * from STEP Tools, Inc. (www.steptools.com)
 */

FILE_DESCRIPTION(
/* description */ ('Program to generate AP238 p21 file',
'ARM_SCHEMA: ap238_arm_schema'),
/* implementation_level */ '4;1');

FILE_NAME(
/* name */ 'AP238Export',
/* time_stamp */ '2021-04-15T22:56:24-05:00',
/* author */ ('Cristhian Ivan Riano Jaimes','Alberto J. Alvares'),
/* organization */ ('Universidade de Brasilia (UnB)'),
/* preprocessor_version */ 'ST-DEVELOPER v16.11',
/* originating_system */
'JSDAI MULTIPLE Version 4.0.0 (Build 270, 2011-12-15T17:42:49)',
/* authorisation */ 'cristhianivanrj');

FILE_SCHEMA (('INTEGRATED_CNC_SCHEMA'));
ENDSEC;

ANCHOR;
<__CONSTANT UNIT steradian>=#2637;  /* si_unit_and_solid_angle_unit */
<__CONSTANT UNIT radian>=#2636;  /* plane_angle_unit_and_si_unit */
<__CONSTANT UNIT degree>=#2633;  /* conversion_based_unit_and_plane_angle_unit */
<__CONSTANT UNIT millimetre>=#2632;  /* length_unit_and_si_unit */
<__CONSTANT UNIT inch>=#2629;  /* conversion_based_unit_and_length_unit */
<STMOD__GEOMETRIC_CONTEXT__>=#2628;  /* geometric_representation_context_and_global_unit_assigned_context */
<STMOD__PRODUCT_DEFINITION_CONTEXT__>=#16;  /* product_definition_context */
<STMOD__PRODUCT_CONTEXT___>=$;
ENDSEC;

DATA;

/***********************************************
 * Application object: PROJECT (#10)
 * ITS_ID: #10, #11, #12, ['Project JSDAI Export']
 * ITS_WORKPIECES [*]: #10, #13, #19
 * MAIN_WORKPLAN: #10, #14, #15, #221
 */
#10=PRODUCT_DEFINITION('','',#11,#16);
#11=PRODUCT_DEFINITION_FORMATION('','',#12);
#12=MACHINING_PROJECT('Project JSDAI Export','','',(#18));
#13=MACHINING_PROJECT_WORKPIECE_RELATIONSHIP('','','',#10,#19);
#14=PROCESS_PRODUCT_ASSOCIATION('','',#10,#15);
#15=PRODUCT_DEFINITION_PROCESS('machining','',#221,'');
#16=PRODUCT_DEFINITION_CONTEXT('CNC Machining',#17,'manufacturing');
#17=APPLICATION_CONTEXT(
'Application protocol for the exchange of CNC data');
#18=PRODUCT_CONTEXT('CNC Machining',#17,'manufacturing');

/***********************************************
 * Application object: WORKPIECE (#19)
 * REVISION_ID: #19, #20, ['']
 * ITS_ID: #19, #20, #21, ['default workpiece']
 */
#19=PRODUCT_DEFINITION('','',#20,#16);
#20=PRODUCT_DEFINITION_FORMATION('','',#21);
#21=PRODUCT('default workpiece','AP-238','',(#18));

```

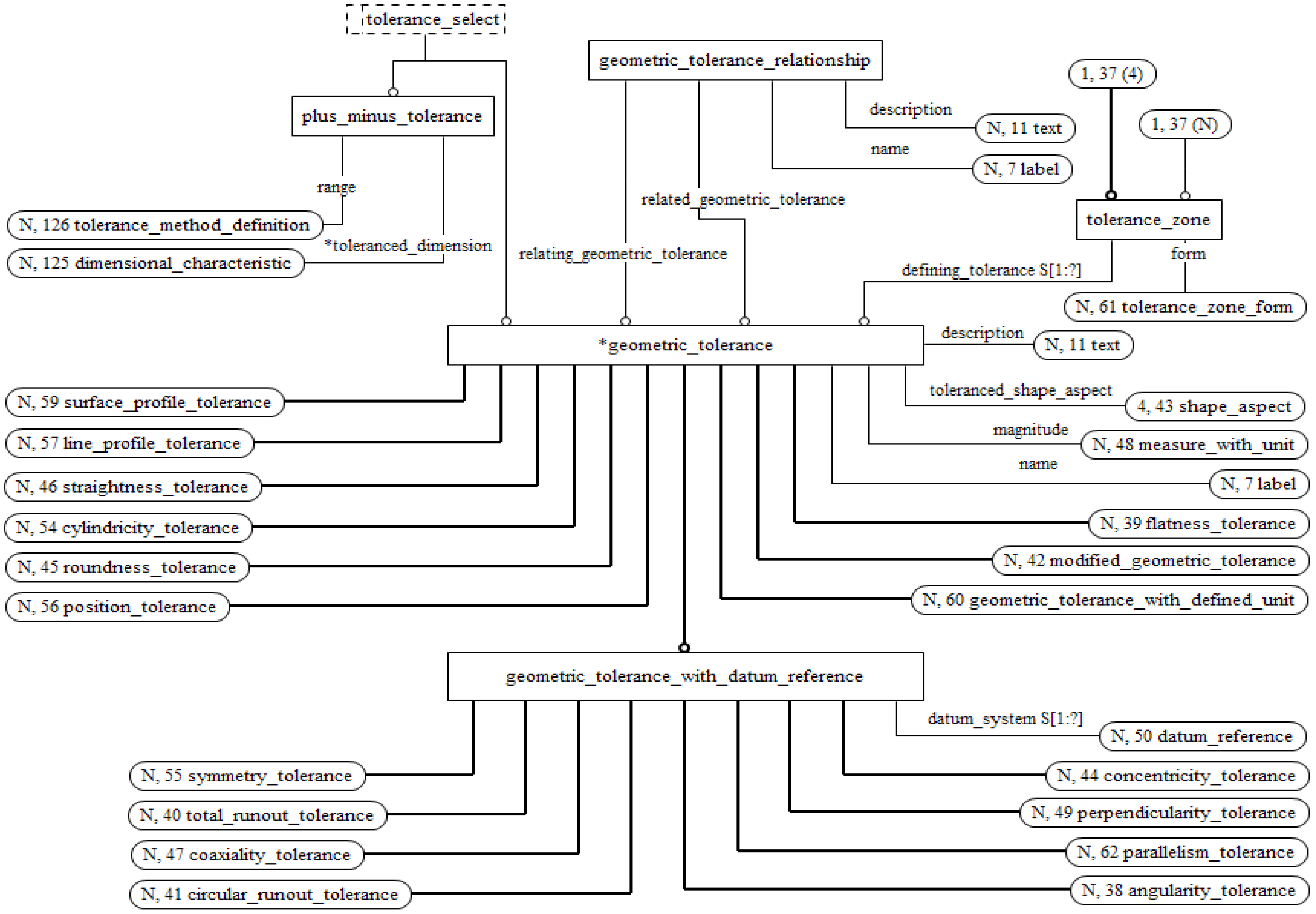
```
/***********************************************
 * Application object: CONST_SPINDLE_SPEED (#22)
 * ROT_SPEED: #22, #23
 */
#22=MACHINING_SPINDLE_SPEED_REPRESENTATION('spindle speed',(#23),#24);
#23=MEASURE_REPRESENTATION_ITEM('rotational speed',NUMERIC_MEASURE(0.),
#33);
#24=(
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNIT_ASSIGNED_CONTEXT((#25,#29,#32))
REPRESENTATION_CONTEXT('INCH DEGREE STERADIAN','3D')
);
#25=(
CONVERSION_BASED_UNIT('inch',#27)
LENGTH_UNIT()
NAMED_UNIT(#26)
);
#26=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.,0.);
#27=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(25.4),#28);
#28=(
LENGTH_UNIT()
NAMED_UNIT(*)
SI_UNIT(.MILLI.,.METRE.)
);
#29=(
CONVERSION_BASED_UNIT('degree',#30)
NAMED_UNIT(#26)
PLANE_ANGLE_UNIT()
);
#30=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(0.01745329252),#31);
#31=(
NAMED_UNIT(*)
SI_UNIT($,.STERADIAN.)
SOLID_ANGLE_UNIT()
);
#32=(
NAMED_UNIT(*)
PLANE_ANGLE_UNIT()
SI_UNIT($,.RADIAN.)
);
#33=DERIVED_UNIT((#34,#37));
#34=DERIVED_UNIT_ELEMENT(#35,1.);
#35=CONTEXT_DEPENDENT_UNIT(#36,'revolution');
#36=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.,0.);
#37=DERIVED_UNIT_ELEMENT(#38,-1.);
#38=(
CONVERSION_BASED_UNIT('',#40)
NAMED_UNIT(#39)
TIME_UNIT()
);
#39=DIMENSIONAL_EXPONENTS(0.,0.,1.,0.,0.,0.,0.);
#40=TIME_MEASURE_WITH_UNIT(TIME_MEASURE(60.),#41);
#41=(
NAMED_UNIT(*)
SI_UNIT($,.SECOND.)
TIME_UNIT()
);

/***********************************************
 * Application object: CONST_SPINDLE_SPEED (#42)
 * ROT_SPEED: #42, #43
 */
#42=MACHINING_SPINDLE_SPEED_REPRESENTATION('spindle speed',(#43),#24);
#43=MEASURE_REPRESENTATION_ITEM('rotational speed',NUMERIC_MEASURE(-1591.),
#33);


/**********************************************
 * Application object: CONST_SPINDLE_SPEED (#44)
 * ROT_SPEED: #44, #45
 */
#44=MACHINING_SPINDLE_SPEED_REPRESENTATION('spindle speed',(#45),#24);
#45=MEASURE_REPRESENTATION_ITEM('rotational speed',NUMERIC_MEASURE(-1591.),
#33);

/**********************************************
 * Application object: CUTTER_LOCATION_TRAJECTORY (#46)
 * BASICCURVE: #46, #47, #48, #49, #50
 * RAPID_SPEED: #46, #51, #52, #53, #54, ['true']
 * ITS_ID: #46, ['start point WS 1 TP 1']
 * ITS_TECHNOLOGY: #46, #55, #182
 */
#46=MACHINING_TOOLPATH('start point WS 1 TP 1',
'cutter location trajectory','','');
#47=ACTION_PROPERTY('basic curve','cutter location trajectory',#46);
#48=ACTION_PROPERTY_REPRESENTATION('','cutter location trajectory',#47,
#49);
#49=REPRESENTATION('',(#50),#24);
#50=POLYLINE('',(#56,#57));
#51=ACTION_PROPERTY('speed profile','rapid',#46);
#52=ACTION_PROPERTY_REPRESENTATION('','rapid',#51,#53);
#53=MACHINING_TOOLPATH_SPEED_PROFILE_REPRESENTATION('',(#54),#24);
#54=DESCRIPTIVE_REPRESENTATION_ITEM('','rapid');
#55=MACHINING_TECHNOLOGY_RELATIONSHIP('','cutter location trajectory',#46,
#182);
#56=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.433070866141732));
#57=CARTESIAN_POINT('',(-0.259842519685039,5.90543307086614,0.433070866141732));
```

C.3 Appendix: EXPRESS G Definitions

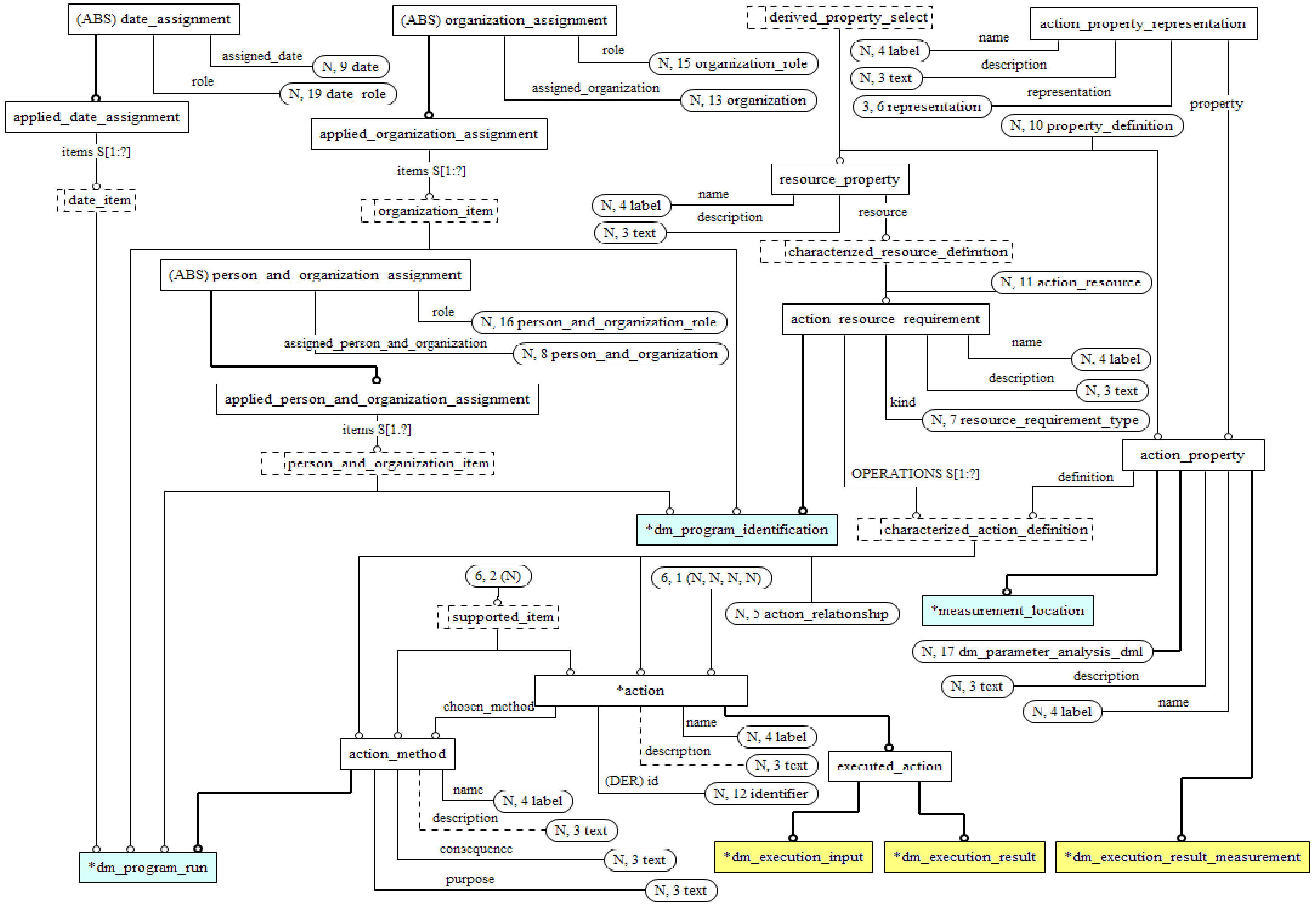
C.3.1 STEP definition in EXPRESS-G for entity `Shape_aspect`


*dm_feature_relationship
*shape_aspect_relationship
N, 10 label
name
related_shape_aspect
(INV) established_by_relationships S[1:?]
N, 55 datum
(DER) id
N, 44 identifier
relating_shape_aspect
(INV) target_basis_relationship
N, 5 datum_target
relating_shape_aspect
(INV) feature_basis_relationship
N, 23 datum_feature
N, 62 dimensional_location
relating_shape_aspect
(INV) basis_relationships S[1:?]
N, 48 symmetric_shape_aspect
relating_shape_aspect
(INV) component_relationships S[2:?]
N, 20 composite_shape_aspect
N, 30 shape_defining_relationship
N, 63 feature_component_relationship
N, 7 shape_aspect_deriving_relationship
1, 1 (N, N, N, N)
description
N, 26 text
related_shape_aspect
relating_shape_aspect
1, 3 (N, N, N, N, N, N, N, N, N)
LOGICAL
N, 44 identifier
(DER) id
product_definitional
description
N, 26 text
*shape_aspect
name
N, 10 label
N, 55 datum
N, 27 taper
N, 40 slot_end
N, 64 tee_profile
N, 60 vee_profile
N, 47 boss_top
N, 23 datum_feature
N, 17 applied_area
N, 14 hole_bottom
N, 15 profile_floor
N, 46 linear_profile
N, 61 modified_pattern
N, 32 transition_feature
N, 41 circular_closed_profile
N, 48 symmetric_shape_aspect
N, 49 derived_shape_aspect
N, 58 rounded_u_profile
N, 39 closed_path_profile
N, 18 ngon_closed_profile
N, 54 partial_circular_profile
N, 6 path_feature_component
N, 8 rectangular_closed_profile
of_shape
N, 13 product_definition_shape
N, 5 datum_target
N, 20 composite_shape_aspect
N, 24 square_u_profile
N, 38 rib_top_floor
N, 43 instanced_feature
N, 31 open_path_profile
N, 25 pocket_bottom
N, 45 tolerance_zone
N, 56 chamfer_offset
1, 2 (N)
description
N, 26 text
*dm_instanced_feature
*characterized_object
name
N, 10 label
N, 53 feature_definition
N, 4 document_file
N, 29 feature_component_definition
*dm_feature_definition
N, 12 dmf_arc
N, 50 dmf_cylinder
N, 51 dmf_edge_point
N, 65 dmf_geometric_curve
N, 11 dmf_surface_of_revolution_dml
N, 42 dmf_plane_symmetric
N, 22 dmf_pattern
N, 57 dmf_line_bounded
N, 37 dmf_circle
N, 9 dmf_line_closed_parallel
N, 19 dmf_plane
N, 28 dmf_line_unbounded
N, 36 dmf_point
N, 59 dmf_plane_closed_parallel
N, 16 dmf_generic_feature
N, 34 dmf_sphere
N, 33 dmf_torus
N, 35 dmf_geometric_surface
N, 21 dmf_ellipse
N, 52 dmf_cone

C.3.1.1 STEP definition in EXPRESS-G for entity `Geometric_Tolerances`

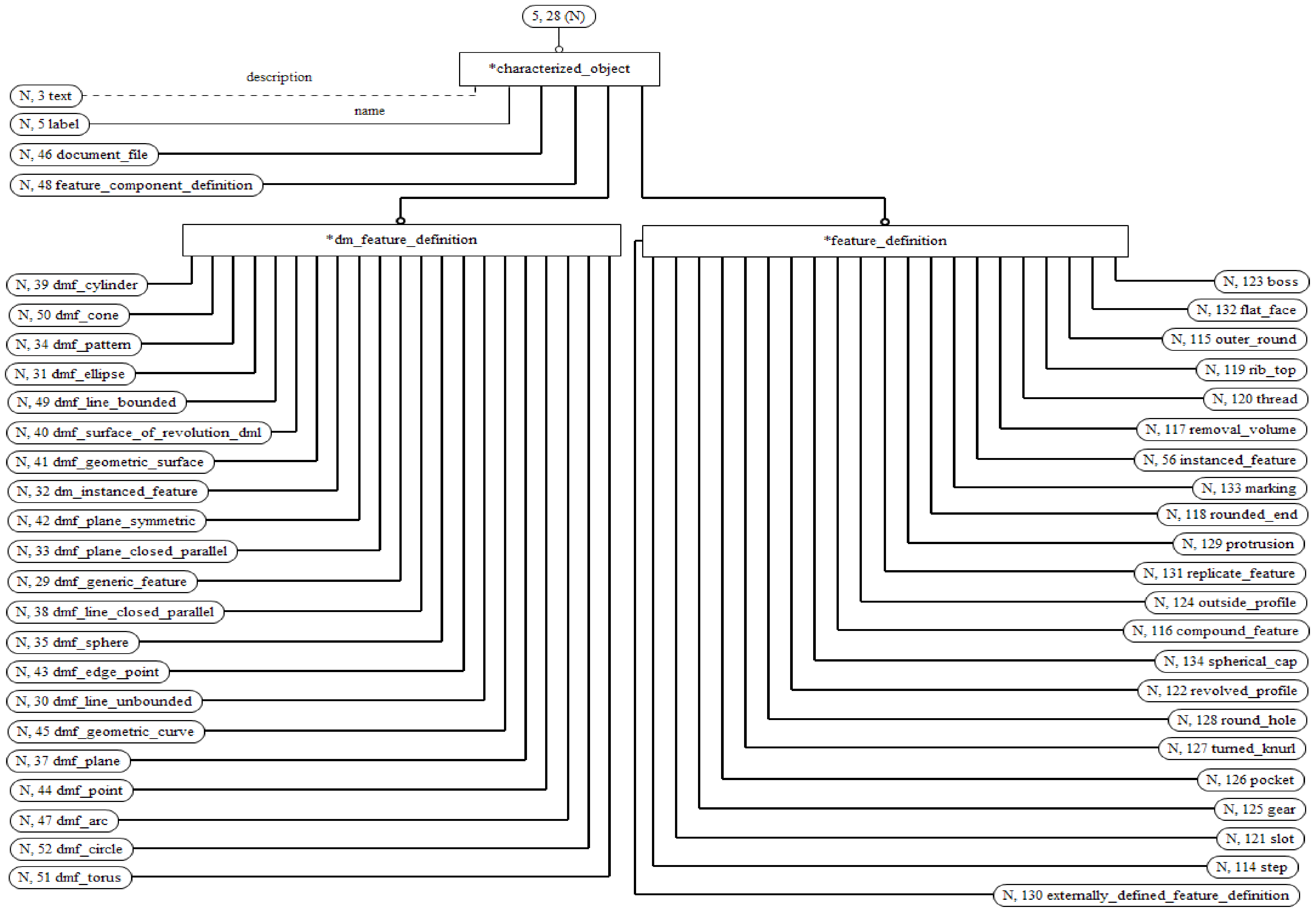

tolerance_select
geometric_tolerance_relationship
1, 37 (4)
1, 37 (N)
plus_minus_tolerance
description
N, 11 text
name
N, 7 label
range
related_geometric_tolerance
tolerance_zone
N, 126 tolerance_method_definition
*toleranced_dimension
N, 125 dimensional_characteristic
relating_geometric_tolerance
form
defining_tolerance S[1:?]
N, 61 tolerance_zone_form
*geometric_tolerance
description
N, 11 text
toleranced_shape_aspect
4, 43 shape_aspect
magnitude
N, 48 measure_with_unit
name
N, 7 label
N, 59 surface_profile_tolerance
N, 57 line_profile_tolerance
N, 46 straightness_tolerance
N, 54 cylindricity_tolerance
N, 45 roundness_tolerance
N, 56 position_tolerance
N, 39 flatness_tolerance
N, 42 modified_geometric_tolerance
N, 60 geometric_tolerance_with_defined_unit
geometric_tolerance_with_datum_reference
datum_system S[1:?]
N, 50 datum_reference
N, 55 symmetry_tolerance
N, 40 total_runout_tolerance
N, 47 coaxiality_tolerance
N, 41 circular_runout_tolerance
N, 44 concentricity_tolerance
N, 49 perpendicularity_tolerance
N, 62 parallelism_tolerance
N, 38 angularity_tolerance

C.3.2 STEP definition in EXPRESS-G for entity `dm_execution_result`



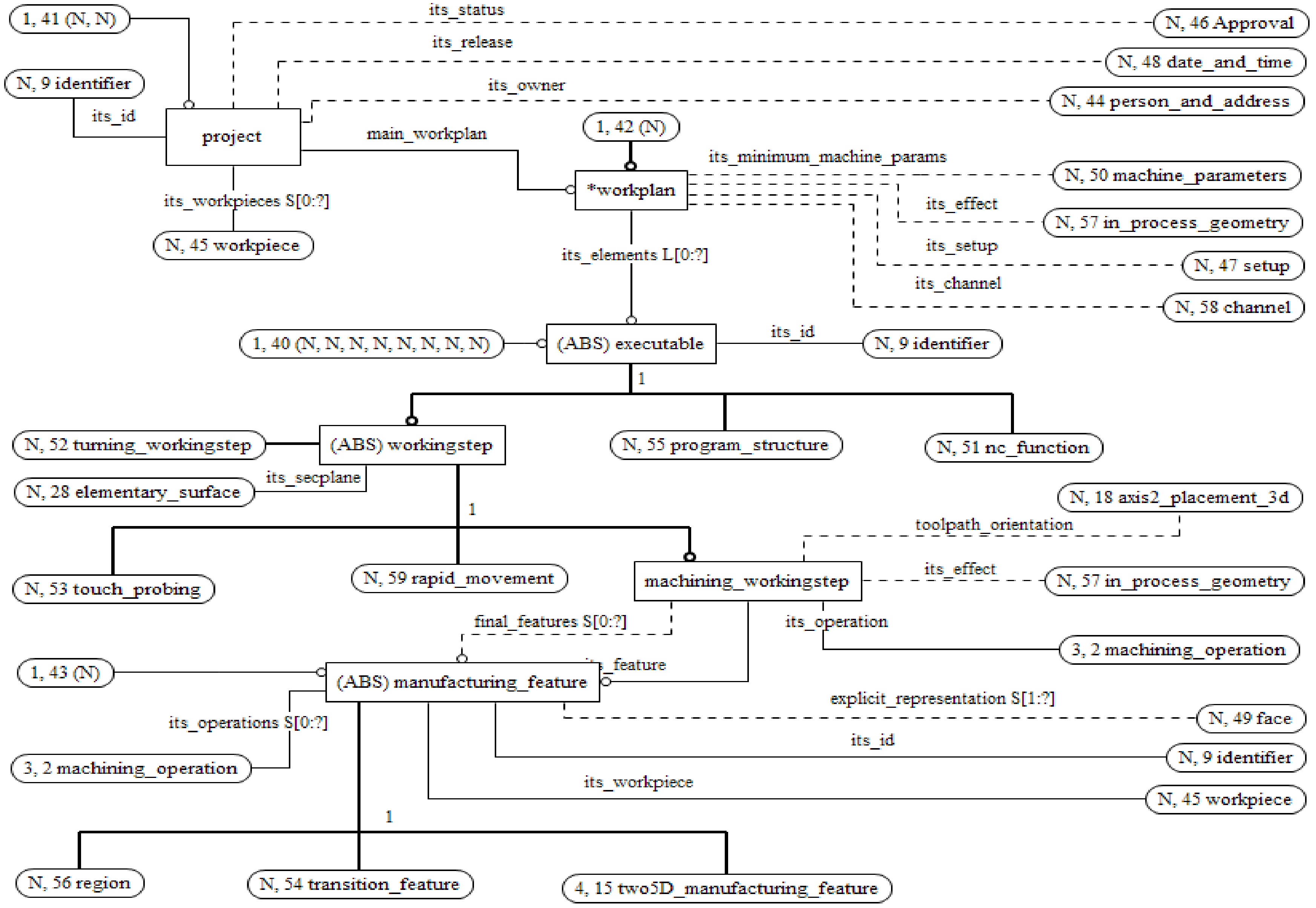
(ABS) date_assignment
assigned_date
N, 9 date
role
N, 19 date_role
applied_date_assignment
items S[1:?]
date_item
(ABS) organization_assignment
role
N, 15 organization_role
assigned_organization
N, 13 organization
applied_organization_assignment
items S[1:?]
organization_item
derived_property_select
action_property_representation
name
N, 4 label
description
N, 3 text
representation
3, 6 representation
property
N, 10 property_definition
resource_property
name
N, 4 label
description
N, 3 text
resource
characterized_resource_definition
N, 11 action_resource
(ABS) person_and_organization_assignment
role
N, 16 person_and_organization_role
assigned_person_and_organization
N, 8 person_and_organization
applied_person_and_organization_assignment
items S[1:?]
person_and_organization_item
action_resource_requirement
name
N, 4 label
description
N, 3 text
kind
N, 7 resource_requirement_type
OPERATIONS S[1:?]
definition
action_property
characterized_action_definition
*dm_program_identification
6, 2 (N)
6, 1 (N, N, N, N)
supported_item
N, 5 action_relationship
*measurement_location
N, 17 dm_parameter_analysis_dml
description
N, 3 text
name
N, 4 label
*action
chosen_method
action_method
name
N, 4 label
description
N, 3 text
(DER) id
N, 12 identifier
executed_action
name
N, 4 label
description
N, 3 text
consequence
N, 3 text
purpose
N, 3 text
*dm_program_run
*dm_execution_input
*dm_execution_result
*dm_execution_result_measurement

C.3.3 STEP definition in EXPRESS-G for entity `Feature`

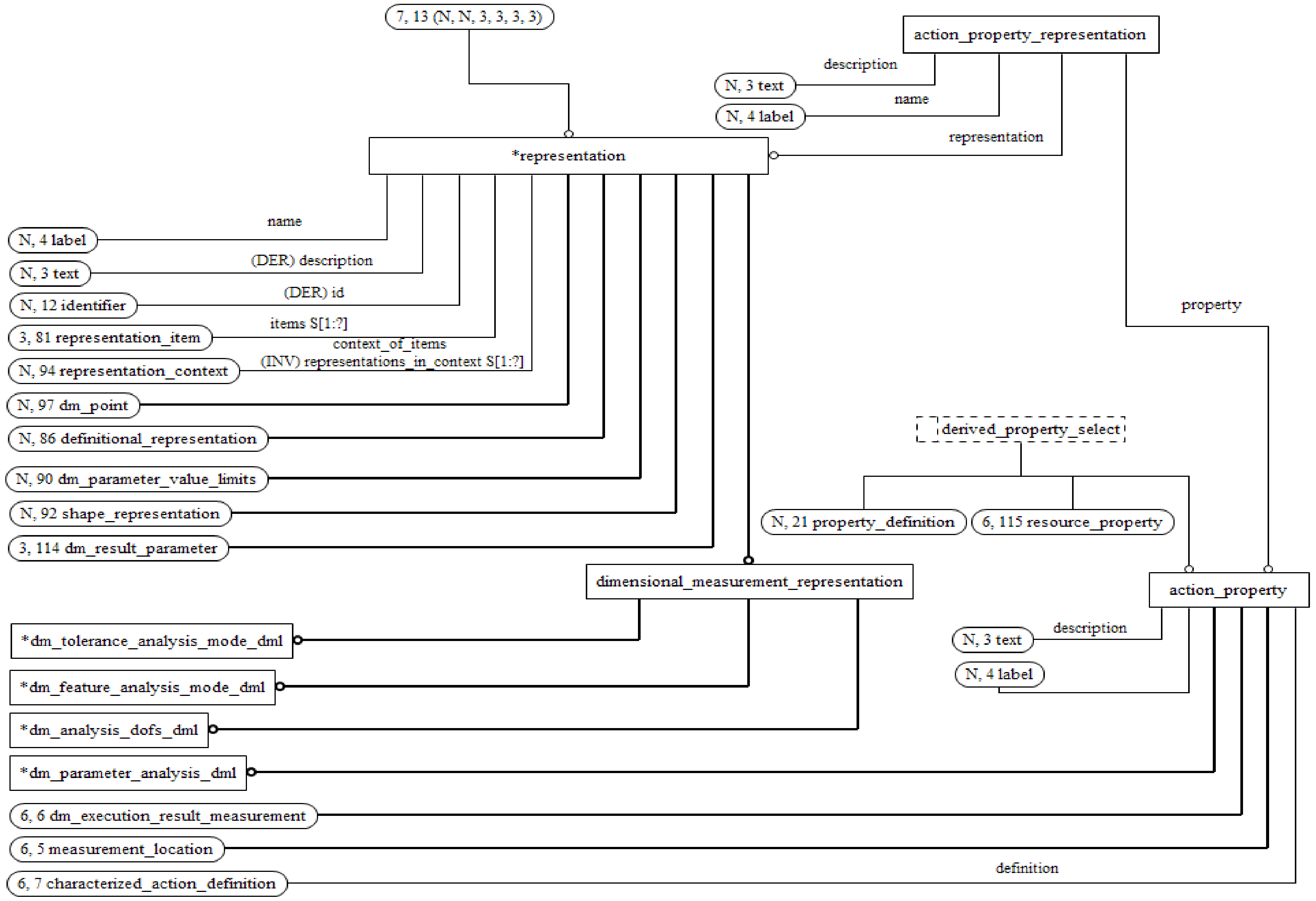


5, 28 (N)
*characterized_object
description
name
N, 3 text
N, 5 label
N, 46 document_file
N, 48 feature_component_definition
*dm_feature_definition
*feature_definition
N, 39 dmf_cylinder
N, 50 dmf_cone
N, 34 dmf_pattern
N, 31 dmf_ellipse
N, 49 dmf_line_bounded
N, 40 dmf_surface_of_revolution_dml
N, 41 dmf_geometric_surface
N, 32 dm_instanced_feature
N, 42 dmf_plane_symmetric
N, 33 dmf_plane_closed_parallel
N, 29 dmf_generic_feature
N, 38 dmf_line_closed_parallel
N, 35 dmf_sphere
N, 43 dmf_edge_point
N, 30 dmf_line_unbounded
N, 45 dmf_geometric_curve
N, 37 dmf_plane
N, 44 dmf_point
N, 47 dmf_arc
N, 52 dmf_circle
N, 51 dmf_torus
N, 123 boss
N, 132 flat_face
N, 115 outer_round
N, 119 rib_top
N, 120 thread
N, 117 removal_volume
N, 56 instanced_feature
N, 133 marking
N, 118 rounded_end
N, 129 protrusion
N, 131 replicate_feature
N, 124 outside_profile
N, 116 compound_feature
N, 134 spherical_cap
N, 122 revolved_profile
N, 128 round_hole
N, 127 turned_knurl
N, 126 pocket
N, 125 gear
N, 121 slot
N, 114 step
N, 130 externally_defined_feature_definition

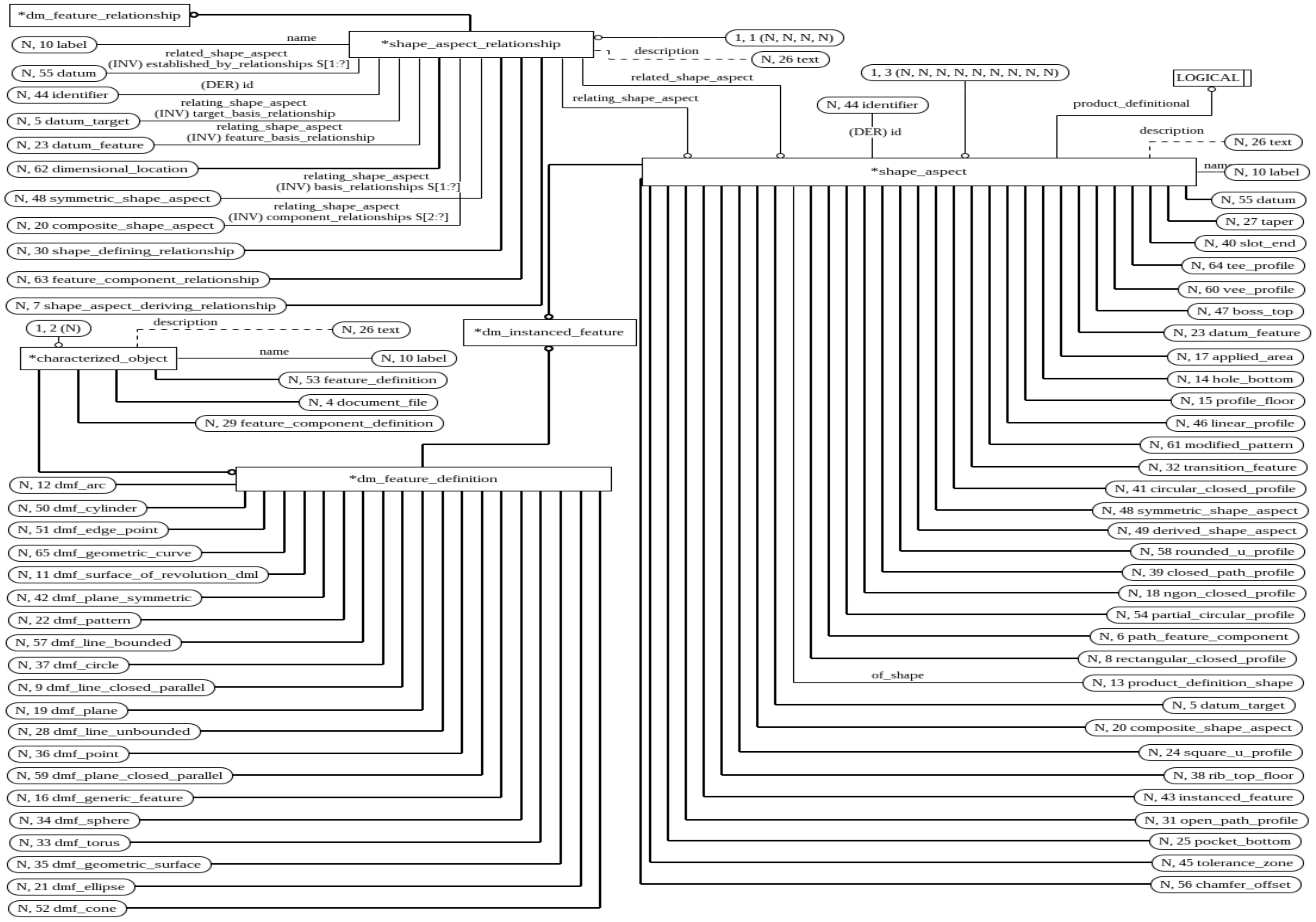
C.3.4 STEP definition in EXPRESS-G for entity `Project`


1, 41 (N, N)
its_status
N, 46 Approval
its_release
N, 48 date_and_time
N, 9 identifier
its_owner
N, 44 person_and_address
its_id
project
main_workplan
1, 42 (N)
*workplan
its_minimum_machine_params
N, 50 machine_parameters
its_effect
N, 57 in_process_geometry
its_setup
N, 47 setup
its_channel
N, 58 channel
its_workpieces S[0:?]
N, 45 workpiece
its_elements L[0:?]
1, 40 (N, N, N, N, N, N, N, N)
(ABS) executable
its_id
N, 9 identifier
1
N, 52 turning_workingstep
(ABS) workingstep
N, 55 program_structure
N, 51 nc_function
its_secplane
N, 28 elementary_surface
N, 18 axis2_placement_3d
1
toolpath_orientation
N, 53 touch_probing
N, 59 rapid_movement
machining_workingstep
its_effect
N, 57 in_process_geometry
final_features S[0:?]
its_operation
3, 2 machining_operation
1, 43 (N)
its_feature
(ABS) manufacturing_feature
explicit_representation S[1:?]
N, 49 face
its_operations S[0:?]
its_id
N, 9 identifier
3, 2 machining_operation
its_workpiece
N, 45 workpiece
1
N, 56 region
N, 54 transition_feature
4, 15 two5D_manufacturing_feature

C.3.5 STEP definition in EXPRESS-G for entity `dimensional_measurement`


7, 13 (N, N, 3, 3, 3, 3)
action_property_representation
description
N, 3 text
name
N, 4 label
representation
*representation
name
N, 4 label
(DER) description
N, 3 text
(DER) id
N, 12 identifier
items S[1:?]
3, 81 representation_item
context_of_items
(INV) representations_in_context S[1:?]
N, 94 representation_context
N, 97 dm_point
N, 86 definitional_representation
N, 90 dm_parameter_value_limits
N, 92 shape_representation
3, 114 dm_result_parameter
property
derived_property_select
N, 21 property_definition
6, 115 resource_property
dimensional_measurement_representation
action_property
*dm_tolerance_analysis_mode_dml
*dm_feature_analysis_mode_dml
*dm_analysis_dofs_dml
*dm_parameter_analysis_dml
description
N, 3 text
N, 4 label
6, 6 dm_execution_result_measurement
6, 5 measurement_location
definition
6, 7 characterized_action_definition

C.3.6 STEP definition in EXPRESS-G for entity `dm_feature`


*dm_feature_relationship
N, 10 label
name
*shape_aspect_relationship
1, 1 (N, N, N, N)
description
N, 26 text
related_shape_aspect
(INV) established_by_relationships S[1:?]
N, 55 datum
(DER) id
N, 44 identifier
relating_shape_aspect
(INV) target_basis_relationship
N, 5 datum_target
relating_shape_aspect
(INV) feature_basis_relationship
N, 23 datum_feature
N, 62 dimensional_location
relating_shape_aspect
(INV) basis_relationships S[1:?]
N, 48 symmetric_shape_aspect
relating_shape_aspect
(INV) component_relationships S[2:?]
N, 20 composite_shape_aspect
N, 30 shape_defining_relationship
N, 63 feature_component_relationship
N, 7 shape_aspect_deriving_relationship
related_shape_aspect
relating_shape_aspect
1, 3 (N, N, N, N, N, N, N, N, N)
LOGICAL
N, 44 identifier
(DER) id
product_definitional
description
N, 26 text
*shape_aspect
name
N, 10 label
N, 55 datum
N, 27 taper
N, 40 slot_end
N, 64 tee_profile
N, 60 vee_profile
N, 47 boss_top
N, 23 datum_feature
N, 17 applied_area
N, 14 hole_bottom
N, 15 profile_floor
N, 46 linear_profile
N, 61 modified_pattern
N, 32 transition_feature
N, 41 circular_closed_profile
N, 48 symmetric_shape_aspect
N, 49 derived_shape_aspect
N, 58 rounded_u_profile
N, 39 closed_path_profile
N, 18 ngon_closed_profile
N, 54 partial_circular_profile
N, 6 path_feature_component
N, 8 rectangular_closed_profile
of_shape
N, 13 product_definition_shape
N, 5 datum_target
N, 20 composite_shape_aspect
N, 24 square_u_profile
N, 38 rib_top_floor
N, 43 instanced_feature
N, 31 open_path_profile
N, 25 pocket_bottom
N, 45 tolerance_zone
N, 56 chamfer_offset
1, 2 (N)
description
N, 26 text
*dm_instanced_feature
*characterized_object
name
N, 10 label
N, 53 feature_definition
N, 4 document_file
N, 29 feature_component_definition
*dm_feature_definition
N, 12 dmf_arc
N, 50 dmf_cylinder
N, 51 dmf_edge_point
N, 65 dmf_geometric_curve
N, 11 dmf_surface_of_revolution_dml
N, 42 dmf_plane_symmetric
N, 22 dmf_pattern
N, 57 dmf_line_bounded
N, 37 dmf_circle
N, 9 dmf_line_closed_parallel
N, 19 dmf_plane
N, 28 dmf_line_unbounded
N, 36 dmf_point
N, 59 dmf_plane_closed_parallel
N, 16 dmf_generic_feature
N, 34 dmf_sphere
N, 33 dmf_torus
N, 35 dmf_geometric_surface
N, 21 dmf_ellipse
N, 52 dmf_cone

D.4 Appendix: Neutral Exchange File for Dimensional Inspection

D.4.1 DMIS File

```
DMISMN/'model2_inspection_setup_1',05.2
$$
$$ Generated by dmis52_pm2.tcl Version: 12.0.20170316 at Apr. 16, 2021 21:47:14
$$
UNITS/MM,ANGDEC,TEMPF
PN(PROGRAM_HEADER_STATEMENT_ID)=PARTID/'model2_inspection_setup_1'
PR(PROGRAM_HEADER_STATEMENT_RV)=PARTRV/'120'
S(TP20_STD_A-5003-0040-01-A_A90_B0)=SNSDEF/PROBE,INDEX,POL,90.,0.0,0.0,1.,0.0,137.27273,3.,SPHERE
S(TP20_STD_A-5003-0040-01-A_A0_B0)=SNSDEF/PROBE,INDEX,POL,0.0,0.0,0.0,0.0,1.,137.27273,3.,SPHERE
S(TP20_STD_A-5003-0040-01-A_A90_B90)=SNSDEF/PROBE,INDEX,POL,90.,90.,-1.,0.0,0.0,137.27273,3.,SPHERE
D(MCS)=DATSET/MCS
F(PLANE1)=FEAT/PLANE,CART,75.,-10.,5.,0.0,1.,0.0
F(PLANE1_BND1)=FEAT/PLANE,CART,150.,-10.,0.0,0.0,0.0,1.
F(PLANE1_BND2)=FEAT/PLANE,CART,0.0,-10.,0.0,1.,0.0,0.0
F(PLANE1_BND3)=FEAT/PLANE,CART,0.0,-10.,10.,0.0,0.0,-1.
F(PLANE1_BND4)=FEAT/PLANE,CART,150.,-10.,10.,-1.,0.0,0.0
BOUND/F(PLANE1)$
,F(PLANE1_BND1)$
,F(PLANE1_BND2)$
,F(PLANE1_BND3)$
,F(PLANE1_BND4)
F(PLANE2)=FEAT/PLANE,CART,90.,51.72201,10.,0.0,0.0,1.
F(PLANE2_BND1)=FEAT/PLANE,CART,180.,155.,10.,1.,0.0,0.0
F(PLANE2_BND2)=FEAT/PLANE,CART,180.,-51.55598,10.,0.0,-1.,0.0
F(PLANE2_BND3)=FEAT/PLANE,CART,0.0,-51.55598,10.,-1.,0.0,0.0
F(PLANE2_BND4)=FEAT/PLANE,CART,0.0,155.,10.,0.0,1.,0.0
BOUND/F(PLANE2)$
,F(PLANE2_BND1)$
,F(PLANE2_BND2)$
,F(PLANE2_BND3)$
,F(PLANE2_BND4)
F(PLANE3)=FEAT/PLANE,CART,150.,72.5,5.,-1.,0.0,0.0
F(PLANE3_BND1)=FEAT/PLANE,CART,150.,155.,10.,0.0,-1.,0.0
F(PLANE3_BND2)=FEAT/PLANE,CART,150.,155.,0.0,0.0,0.0,1.
F(PLANE3_BND3)=FEAT/PLANE,CART,150.,-10.,0.0,0.0,1.,0.0
F(PLANE3_BND4)=FEAT/PLANE,CART,150.,-10.,10.,0.0,0.0,-1.
BOUND/F(PLANE3)$
,F(PLANE3_BND1)$
,F(PLANE3_BND2)$
,F(PLANE3_BND3)$
,F(PLANE3_BND4)
F(PLANE4)=FEAT/PLANE,CART,75.,72.5,0.0,0.0,0.0,1.
F(PLANE4_BND1)=FEAT/PLANE,CART,0.0,155.,0.0,1.,0.0,0.0
F(PLANE4_BND2)=FEAT/PLANE,CART,0.0,-10.,0.0,0.0,1.,0.0
F(PLANE4_BND3)=FEAT/PLANE,CART,150.,-10.,0.0,-1.,0.0,0.0
F(PLANE4_BND4)=FEAT/PLANE,CART,150.,155.,0.0,0.0,-1.,0.0
BOUND/F(PLANE4)$
,F(PLANE4_BND1)$
,F(PLANE4_BND2)$
,F(PLANE4_BND3)$
,F(PLANE4_BND4)
```

```
SNSLCT/SA(TP20_STD_A-5003-0040-01-A_A0_B0)
$$ Number of PTMEAS = 15
MEAS/PLANE,F(PLANE2),3
GOTO/162.,-30.90038,40.
SNSET/APPRCH,6.5
SNSET/RETRCT,7.5
PTMEAS/CART,162.,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,141.42857,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,120.85714,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,100.28571,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,79.71429,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,59.14286,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,38.57143,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,18.,-30.90038,10.,0.0,0.0,1.
PTMEAS/CART,162.,-7.29399,10.,0.0,0.0,1.
PTMEAS/CART,162.,16.31241,10.,0.0,0.0,1.
PTMEAS/CART,162.,39.91881,10.,0.0,0.0,1.
PTMEAS/CART,162.,63.52521,10.,0.0,0.0,1.
PTMEAS/CART,162.,87.13161,10.,0.0,0.0,1.
PTMEAS/CART,162.,110.738,10.,0.0,0.0,1.
PTMEAS/CART,162.,134.3444,10.,0.0,0.0,1.
GOTO/162.,134.3444,40.
ENDMES
$$ Number of PTMEAS = 64
MEAS/PLANE,F(PLANE3),3
GOTO/120.,138.5,1.
PTMEAS/CART,150.,138.5,1.75,-1.,0.0,0.0
PTMEAS/CART,150.,138.5,2.14286,-1.,0.0,0.0
PTMEAS/CART,150.,138.5,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,138.5,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,138.5,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,138.5,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,138.5,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,138.5,9.,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,9.,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,2.14286,-1.,0.0,0.0
PTMEAS/CART,150.,119.64286,1.75,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,1.75,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,2.14286,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,100.78571,9.,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,9.,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,2.14286,-1.,0.0,0.0
PTMEAS/CART,150.,81.92857,1.75,-1.,0.0,0.0
PTMEAS/CART,150.,63.07143,1.75,-1.,0.0,0.0
PTMEAS/CART,150.,63.07143,2.14286,-1.,0.0,0.0
```

```
PTMEAS/CART,150.,63.07143,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,63.07143,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,63.07143,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,63.07143,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,63.07143,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,63.07143,9.,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,9.,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,2.14286,-1.,0.0,0.0
PTMEAS/CART,150.,44.21429,1.75,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,1.75,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,2.14286,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,25.35714,9.,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,9.,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,7.85714,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,6.71429,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,5.57143,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,4.42857,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,3.28571,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,2.14286,-1.,0.0,0.0
PTMEAS/CART,150.,6.5,1.75,-1.,0.0,0.0
GOTO/120.,6.5,1.
ENDMES
DATDEF/FA(PLANE1),DAT(B)
DATDEF/FA(PLANE2),DAT(A)
T(PERPENDICULARIDAD1)=TOL/PERP,.5,DAT(B),XAXIS
OUTPUT/FA(PLANE3),TA(PERPENDICULARIDAD1)
T(PLANITUD1)=TOL/FLAT,.3
OUTPUT/FA(PLANE4),TA(PLANITUD1)
T(PARALELISMO1)=TOL/PARLEL,.5,DAT(A),ZAXIS
OUTPUT/FA(PLANE4),TA(PARALELISMO1)
ENDFIL
```

F.5 Appendix: Support Codes for the Calculation of Manufacturing Errors

F.5.1 Code to Get the Deflection Angle

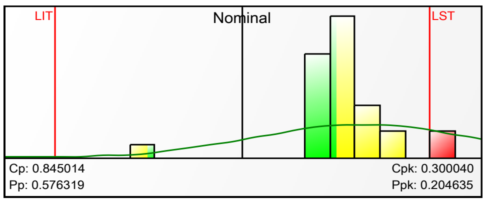
```

close all;
load Vars;
nameFeature=Dx_Tolerancia_DSTNCE;
figure(1)
plot(nameFeature)
xlabel('Observation')
ylabel('Distance (mm)')
title('Historical Production Data')

figure(2)
histfit(nameFeature) % Plot histogram with normal distribution fit
format shortg
xlabel('Distance (mm)')
ylabel('Frequency (counts)')
title('Distance Histogram')

pd = fitdist(nameFeature,'normal') % Fit normal distribution to data

x1 = Y1
y1 = nameFeature
format long
b1 = x1\y1

X1 = [ones(length(x1),1) x1];
b1 = X1\y1

figure(3)
yCalc1 = X1*b1;
scatter(x1,y1)
hold on
plot(x1,yCalc1,'--')
xlabel('Obs')
ylabel('Valor')
title('Linear Regression Relation Between Valor & Obs')
grid on
legend('Data','Slope','Slope & Intercept','Location','best');

Rsq1 = 1 - sum((y1 - yCalc1).^2)/sum((y1 - mean(y1)).^2)

Tet1= atan(b1(2,1));
Teta1 = rad2deg(Tet1)
```

G.6 Appendix: Measurement Results

G.6.1 Measurement Results Acquired Via CMMs

Variable Global Report

Caract: Dx_Tolerância-DSTNCE



Nominal: 170.500000

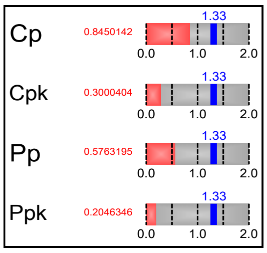
XD Bar: 171.144928

S(total): 0.5783829

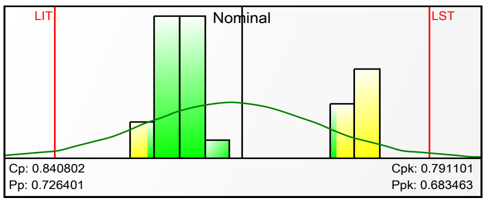
Máx: 173.102000

Mín: 170.016000

Relação De Defeito: 7



Caract: Dy_Tolerância-DSTNCE



Nominal: 170.500000

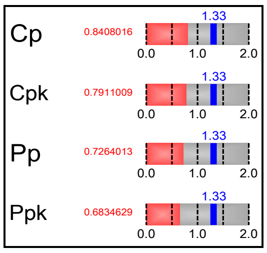
XD Bar: 170.440888

S(total): 0.4588832

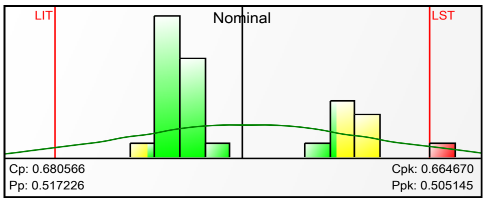
Máx: 171.224000

Mín: 170.003000

Relação De Defeito: (



Caract: Ly_1 Length Toleranc-DSTNCE



Nominal: 170.500000

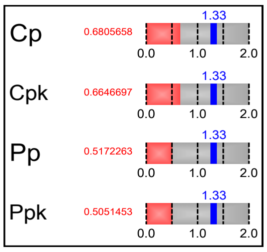
XD Bar: 170.523357

S(total): 0.6444633

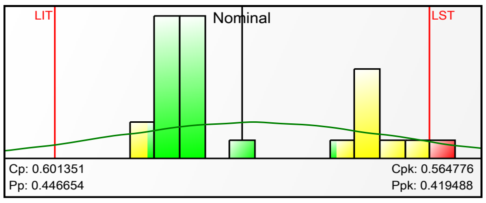
Máx: 173.026000

Mín: 170.021000

Relação De Defeito: 3



Caract: Ly_2 Lenght_Toleran-DSTNCE



Nominal: 170.500000

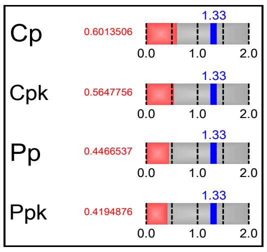
XD Bar: 170.560821

S(total): 0.7462903

Máx: 173.453000

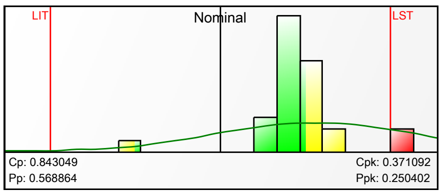
Mín: 169.908000

Relação De Defeito: 3



Variable Global Report

Caract: Lx_1 Lenght_Toleran-DSTNCE



Nominal: 170.500000

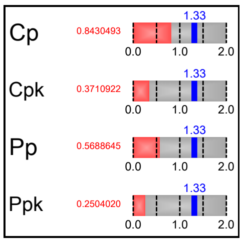
XD Bar: 171.05982

S(total): 0.5859626

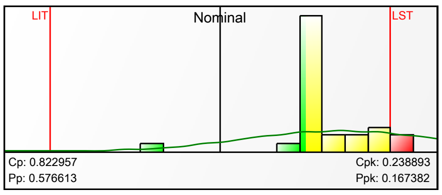
Máx: 173.126000

Mín: 169.967000

Relação De Defeito:



Caract: Lx_2 Lenght_Toleran-DSTNCE



Nominal: 170.500000

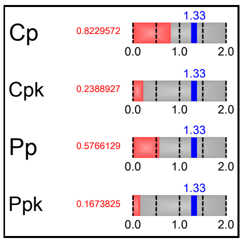
XD Bar: 171.20971

S(total): 0.5780885

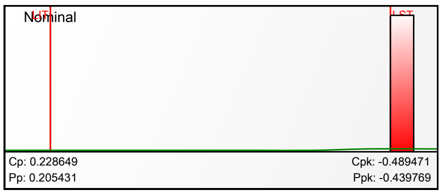
Máx: 173.077000

Mín: 170.054000

Relação De Defeito:



Caract: Y_Paralelismo-LINE-PARLEL



Nominal: 0.000000

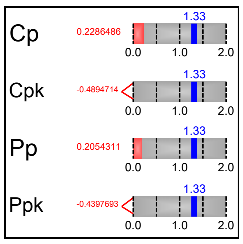
XD Bar: 0.2070357

S(total): 0.0811302

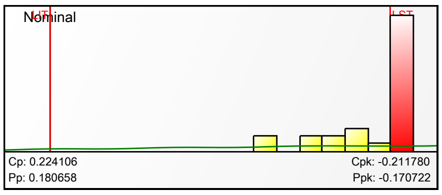
Máx: 0.394000

Mín: 0.108000

Relação De Defeito:



Caract: X_Paralelismo-LINE-PARLEL



Nominal: 0.000000

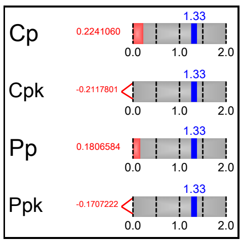
XD Bar: 0.1472500

S(total): 0.0922551

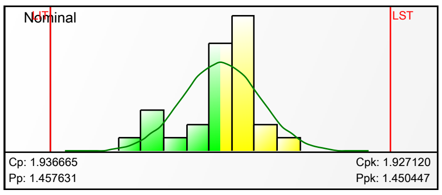
Máx: 0.392000

Mín: 0.063000

Relação De Defeito:



Caract: X1-Y1 Perpendiculari-LINE



Nominal: 0.000000

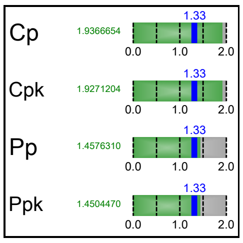
XD Bar: 0.5024643

S(total): 0.1143408

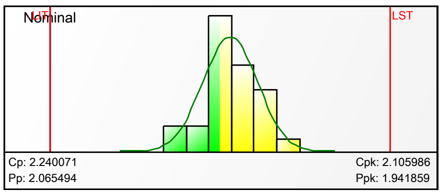
Máx: 0.702000

Mín: 0.232000

Relação De Defeito: (



Caract: Y1-X2 Perpendiculari-LINE



Nominal: 0.000000

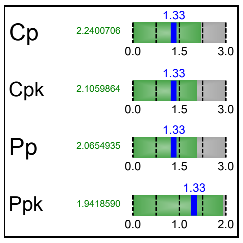
XD Bar: 0.5299286

S(total): 0.0806910

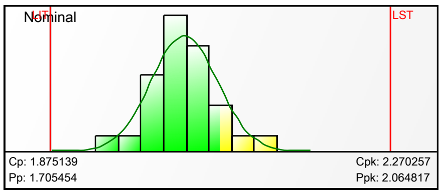
Máx: 0.691000

Mín: 0.337000

Relação De Defeito: (



Caract: X2-Y2 Perpendiculari-LINE



Nominal: 0.000000

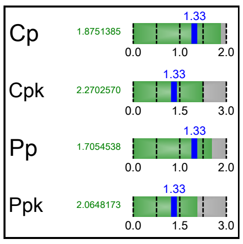
XD Bar: 0.3946429

S(total): 0.0977257

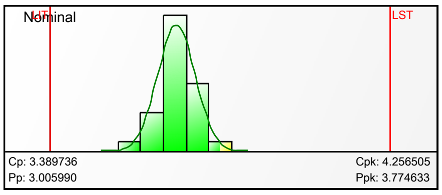
Máx: 0.603000

Mín: 0.178000

Relação De Defeito: (



Caract: Y2-X1 Perpendiculari-LINE



Nominal: 0.000000

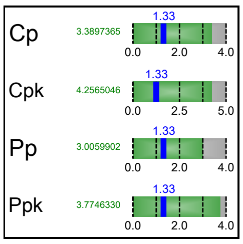
XD Bar: 0.3721481

S(total): 0.0554448

Máx: 0.512000

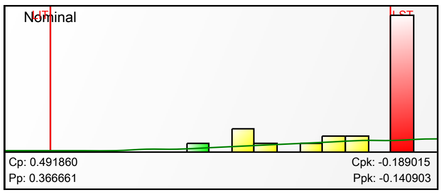
Mín: 0.260000

Relação De Defeito: (



Variable Global Report

Caract: Ly1_Retilineidade-LINE-STRGHT



Nominal: 0.000000

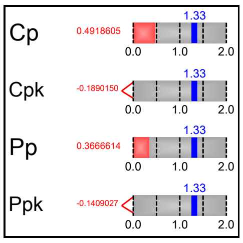
XD Bar: 0.0596071

S(total): 0.0227276

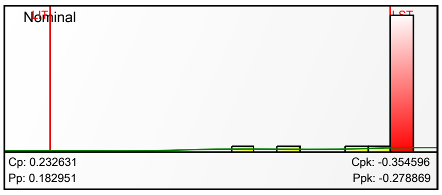
Máx: 0.117000

Mín: 0.023000

Relação De Defeito:



Caract: Ly2_Retilineidade-LINE-STRGHT



Nominal: 0.000000

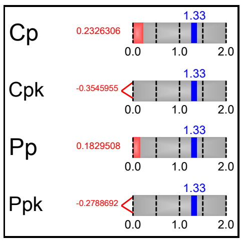
XD Bar: 0.0881071

S(total): 0.0455496

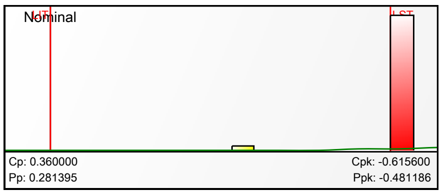
Máx: 0.269000

Mín: 0.029000

Relação De Defeito:



Caract: Lx_1_Retilineidade-LINE-STRGHT



Nominal: 0.000000

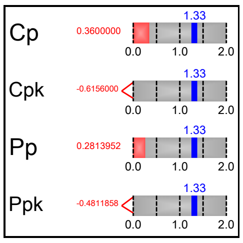
XD Bar: 0.0927500

S(total): 0.0296143

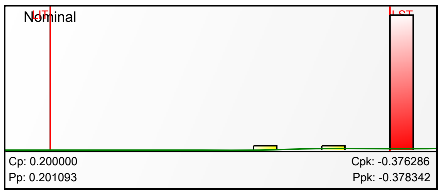
Máx: 0.153000

Mín: 0.027000

Relação De Defeito:



Caract: Lx_2 Retilineidade-LINE-STRGHT



Nominal: 0.000000

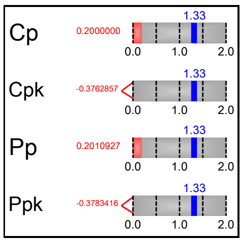
XD Bar: 0.0970357

S(total): 0.0414403

Máx: 0.208000

Mín: 0.032000

Relação De Defeito:



ANNEXES

I.1 Annex: Data Schema EXPRESS

I.1.1 shape_tolerance_schema Schema EXPRESS

This annex presents the EXPRESS language code for the `shape_tolerance_schema` scheme.

```

SCHEMA shape_tolerance_schema;
REFERENCE FROM product_property_definition_schema
    (shape_aspect,
     shape_aspect_relationship);
REFERENCE FROM measure_schema
   (derive_dimensional_exponents,
    dimensional_exponents,
    measure_with_unit,
    measure_value);
REFERENCE FROM representation_schema
    (representation);
REFERENCE FROM support_resource_schema
    (label,
     text);
REFERENCE FROM shape_aspect_definition_schema
    (datum_reference,
     limit_condition);
REFERENCE FROM shape_dimension_schema
    (dimensional_characteristic,
     dimensional_location);

TYPE tolerance_method_definition = SELECT
  (tolerance_value,
   limits_and_fits);
END_TYPE;

TYPE  shape_tolerance_select = SELECT
     (geometric_tolerance,
      plus_minus_tolerance);
END_TYPE;

ENTITY dimension_related_tolerance_zone_element;
  related_dimension : dimensional_location;
  related_element   : tolerance_zone_definition;
END_ENTITY;

ENTITY geometric_tolerance;
  name                      : label;
  description               : text;
  magnitude                 : measure_with_unit;
  toleranced_shape_aspect : shape_aspect;
WHERE
  WR1: ('NUMBER' IN TYPEOF
       (magnitude\measure_with_unit.value_component)) AND
       (magnitude\measure_with_unit.value_component >= 0.0);
END_ENTITY;

ENTITY geometric_tolerance_relationship;
                                  name                         :label;
                                  description                  :text;
```

```
                                      relating_geometric_tolerance : geometric_tolerance;
                                      related_geometric_tolerance  : geometric_tolerance;
END_ENTITY;

ENTITY geometric_tolerance_with_datum_reference
   SUBTYPE OF (geometric_tolerance);
   datum_system : SET [1:?] OF datum_reference;
END_ENTITY;

ENTITY geometric_tolerance_with_defined_unit
  SUBTYPE OF (geometric_tolerance);
  unit_size : measure_with_unit;
WHERE
  WR1: ('NUMBER' IN TYPEOF
       (unit_size\measure_with_unit.value_component)) AND
       (unit_size\measure_with_unit.value_component > 0.0);
END_ENTITY;

ENTITY modified_geometric_tolerance
  SUBTYPE OF (geometric_tolerance);
  modifier : limit_condition;
END_ENTITY;

ENTITY projected_zone_definition
  SUBTYPE OF (tolerance_zone_definition);
  projection_end   : shape_aspect;
  projected_length : measure_with_unit;
WHERE
  WR1: ('NUMBER' IN TYPEOF
       (projected_length\measure_with_unit.value_component)) AND
       (projected_length\measure_with_unit.value_component > 0.0);
  WR2: (derive_dimensional_exponents
       (projected_length\measure_with_unit.unit_component)=
        dimensional_exponents(1,0,0,0,0,0,0));
END_ENTITY;

ENTITY runout_zone_definition
  SUBTYPE OF (tolerance_zone_definition);
  orientation  : runout_zone_orientation;
END_ENTITY;

ENTITY runout_zone_orientation;
  angle : measure_with_unit;
END_ENTITY;

ENTITY runout_zone_orientation_reference_direction
  SUBTYPE OF (runout_zone_orientation);
  orientation_defining_relationship: shape_aspect_relationship;
END_ENTITY;

ENTITY statistical_distribution_for_tolerance
  SUBTYPE OF (representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF\representation.items |
       NOT ('QUALIFIED_MEASURE_SCHEMA.MEASURE_REPRESENTATION_ITEM'
       IN TYPEOF (item)))) = 0;
END_ENTITY;

ENTITY tolerance_with_statistical_distribution;
   associated_tolerance : shape_tolerance_select;
   tolerance_allocation : statistical_distribution_for_tolerance;
END_ENTITY;

ENTITY tolerance_zone
  SUBTYPE OF (shape_aspect);
  defining_tolerance : SET [1:?] OF geometric_tolerance;
  form               : tolerance_zone_form;
END_ENTITY;

ENTITY tolerance_zone_form;
  name : label;
END_ENTITY;

ENTITY tolerance_zone_definition
  SUPERTYPE OF (ONEOF (projected_zone_definition,
                       runout_zone_definition));
  zone : tolerance_zone;
  boundaries: SET [1:?] OF shape_aspect;
END_ENTITY;

ENTITY limits_and_fits;
  form_variance   : label;
  zone_variance   : label;
  grade           : label;
  source          : text;
END_ENTITY;

ENTITY  plus_minus_tolerance;
  range                : tolerance_method_definition;
  toleranced_dimension : dimensional_characteristic;
UNIQUE
  UR1: toleranced_dimension;
END_ENTITY;

ENTITY tolerance_value;
   lower_bound : measure_with_unit;
   upper_bound : measure_with_unit;
DERIVE
    ubvc  : REAL := upper_bound\measure_with_unit.value_component;
    lbvc  : REAL := lower_bound\measure_with_unit.value_component;
WHERE
   WR1: ubvc > lbvc;
   WR2: upper_bound\measure_with_unit.unit_component =
        lower_bound\measure_with_unit.unit_component;
END_ENTITY;

END_SCHEMA;   -- End of shape_tolerance_schema
```