STEP-NC Implementation – ARM or AIM?

Allison Barnard Feeney, Tom Kramer, Fred Proctor
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
100 Bureau Drive, Stop 8260
Gaithersburg, MD 20899-8260

Martin Hardwick, David Loffredo
STEP Tools, Inc.
216 River Street
Troy, NY 12180

1 Background

STEP-NC seeks to provide a standard representation of computer numeric control data that uses some or all of the representation methods and architecture developed for ISO 10303, *Product data representation and exchange* (informally known as STEP). STEP-NC is being developed concurrently under two different subcommittees of ISO Technical Committee 184, Industrial automation systems and integration, as two different standards, ISO 14649 *Data model for computerized numerical controllers* and ISO 10303-238 *Application interpreted model for computer numeric controllers*.

The primary difference between the two standards is the degree to which they use the STEP representation methods and technical architecture. STEP includes an information modeling language called EXPRESS that is used to specify information models in STEP. The STEP three-level architecture starts with a requirements model, maps that model to an implementation model based on a set of common data definitions called integrated resources, and has separately designed implementation methods. This architecture is described in more detail in annex A of this paper. Implications of using or not using the full STEP architecture are the main topic of this paper.

ISO TC 184/SC1, Physical device control, is developing ISO 14649, a family of standards that uses the EXPRESS language to define a set of NC domain requirements models in terminology familiar to NC experts. ISO 14649 uses only the EXPRESS language and corresponding implementation methods from the STEP family of standards.

ISO TC 184/SC4, Industrial data, is developing ISO 10303-238, an application protocol in the STEP family of standards. ISO 10303-238 (hereafter AP 238) uses the EXPRESS models in ISO 14649 as the domain requirements model (called an application reference model, or ARM) with a few modifications then maps it to the STEP integrated resources to obtain the implementation model (application interpreted model, or AIM).

At issue is whether it is best, from an implementer's viewpoint, to implement the ISO 14649 models directly (aka ARM implementation), or to implement the requirements from ISO 14649 as mapped into the STEP integrated resources (aka AIM implementation).

This paper first provides a practical comparison of the physical file excerpts that illustrates the obvious differences between the two approaches. Then the value of the STEP approach is discussed in the context of four different model change scenarios. Then considerations more specific to STEP-NC are discussed. Recommendations on best implementation practices are made.

2 Comparison of physical files

In the interest of framing this discussion more concretely, Tom Kramer, a Catholic University guest researcher at NIST, used various software tools (many home-grown) to generate ISO 10303-21 physical file examples corresponding subsets of the ISO 14649 and AP 238 EXPRESS models. Both excerpts represent physical data regarding a 12.7 mm (half-inch) drill. The details of this experiment are provided in Annex B of this document.

2.1 The data

ISO 14649 (ARM) data

```
#22=MILLING_CUTTING_TOOL('DRILL-0.5-2-',#23,(#25),76.2,$,$);
#23=TWIST_DRILL(#24,2,.RIGHT.,.F.,0.);
#24=TOOL_DIMENSION(12.7,31.,0.,55.88,0.,0.,0.);
#25=CUTTING_COMPONENT(0.,#50,$,1000000.,$);
#50=MATERIAL('UNKNOWN','hss',());
```

ISO 10303 AP 238 (AIM) data

```
#51=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE (0.01745329252),#495);
#53=(CONVERSION_BASED_UNIT('DEGREES',#51)
 NAMED_UNIT(#430)
 PLANE_ANGLE_UNIT());
#58=SI_UNIT(*,.MILLI.,.METRE.);
#74=MACHINING_TOOL_BODY_REPRESENTATION('drill',(#416,#417, #418),#500);
#78=RESOURCE_PROPERTY_REPRESENTATION('','',#82,#74);
#82=RESOURCE_PROPERTY('tool_body','',#90);
#86=ACTION_RESOURCE_TYPE('cutting tool');
#90=MACHINING_TOOL('TL DRILL-2','12.7 mm TWIST_DRILL', (#133),#86);
#416=(LENGTH_MEASURE_WITH_UNIT()
 MEASURE_REPRESENTATION_ITEM()
 MEASURE_WITH_UNIT(LENGTH_MEASURE(12.7),#58)
 REPRESENTATION_ITEM('diameter'));
#417=(LENGTH_MEASURE_WITH_UNIT()
 MEASURE_REPRESENTATION_ITEM()
 MEASURE_WITH_UNIT(LENGTH_MEASURE(21.184535069175),#58)
 REPRESENTATION_ITEM('flute_length'));
#418=(LENGTH_MEASURE_WITH_UNIT()
 MEASURE_REPRESENTATION_ITEM()
 MEASURE_WITH_UNIT(LENGTH_MEASURE(118.),#53)
 REPRESENTATION_ITEM('included_angle'));
#430=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.);
```

```
#493=(LENGTH_UNIT()
 NAMED_UNIT(*)
 SI_UNIT(.MILLI.,.METRE.));
#495=(NAMED_UNIT(*)
 PLANE_ANGLE_UNIT()
 SI_UNIT($,.RADIAN.));
#499=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(1.E-006),#493,
 'DISTANCE_ACCURACY_VALUE',
 'Maximum model space distance between geometric entities at asserted co
 nnectivities');
#500=(GEOMETRIC_REPRESENTATION_CONTEXT(3)
 GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#499))
 GLOBAL_UNIT_ASSIGNED_CONTEXT((#498,#495,#493))
 REPRESENTATION_CONTEXT('ID1','3D'));
```

2.2 Amount of data

There are 5 lines of ISO 14649 data and 37 lines of AP 238 data. Much of the AP 238 data is simply identifying units (instances presented in boldface font). A fixed set of units is used in ISO 14649, so this data is not necessary when ISO 14649 is used. If you assume that all instances in the AP 238 file that use the same system of units reference the same unit instances, the number of lines of AP 238 data is reduced to 18. For this example, which we assume is typical, AP 238 requires between three to seven times as much data as ISO14649.

In this particular example, the ISO 14649 data set contains more information than the AP 238 data set. The ISO14649 data set gives the following data not given in the AP 238 data set.

- handedness of the drill (the ".RIGHT." in #23).
- the material from which the drill is made (#50).
- the number_of_teeth, i.e. number of flutes (the "2" in #23).

If the AP 238 data set included this information, it would be significantly longer. AP 238 can represent all of this data; the generating system simply chose not to provide it.

File size is likely to be of only minor importance in most cases, but is likely to be significant in a few cases where there are large amounts of information. If there are 3 terabytes of ARM data, there may be 10 to 20 terabytes of AIM data representing the same information. If storage for some amount of information in an ARM data set costs $100,000, storage for that amount in AIM form may cost $300,000 to $700,000. This will occur in some cases and will be important in those cases. The cases where the difference is between $10 and $70 will be much more frequent, but will not be much cause for concern.

2.3 Programming complexity

Suppose that one wants to determine the diameter of the drill used in the one and only drilling operation. The ISO14649 data is arranged in a tree with #22 at the root. Although it is not shown above, #22 is referenced by the drilling operation that uses the drill. All of the data about the drill can be obtained by tracing down the tree three or fewer levels. The C++ code to obtain the drill diameter could be written in a single line. The AP 238 data is not accessible starting from any

single object and tracing down a tree. Different data must be obtained by different methods. And the different methods may be very cumbersome. It is unlikely that the equivalent operation on AP 238 data could be implemented in fewer than twelve lines of code. Annex B outlines the steps that must be taken to retrieve the drill diameter from the AP 238 data structures.

This example is presumed to be indicative of AP 238. All in all, it appears that data access sections of application programs based on AP 238 will be many times as long as equivalent sections based on ISO 14649.

An AIM further obscures semantics through the use of what are termed "complex instances," or objects of multiple type. These are used in the AP 238 data file (seven of the eight objects that are written on two or more lines) to indicate relationships that are modeled as simple attributes in ISO14649. These multiple type objects are required by various WHERE rules of AP 238. The fact that a simple attribute cannot be modeled in a simple way is due to the generic design of the STEP integrated resources. It contributes to large file size and programming complexity, but also serves as insulation from changes to the requirements model, or ARM.

The number of lines of code required to build an application is important because, in most cases, the number of lines of code translates directly into the cost and time required to build a system. However, most developers will use libraries (free or commercial) that will perform common tasks and insulate them from the more tedious tasks, thereby reducing the number of lines of code that must be written by hand. In the end, the most important code deals with the other aspects of the application.

Programming against the AIM is much more complex than programming against an ARM. Programmers should rely on reusable code modules to the degree possible.

2.4 Permissiveness of AIM

Because it is a design goal of STEP to be extensible, it is possible for an AIM to refer to constructs outside the scope of the ARM. However, such inclusions would not be considered in conformance with the AP, as the AP scope is limited to the requirements standardized in the AP ARM. As the requirements for STEP-NC grow, this flexibility will allow support of new requirements in future versions of AP 238. This characteristic of an AIM is illustrated by the string "flute_length" in instance #417 of the AIM The ISO 14649 ARM does not have an attribute named "flute_length," but the AP 238 software allowed it to be written into the AIM data anyway.[1] In this case, we have identified an error in the software that wrote out the AIM data file. It is important to note that not adhering strictly to the standardized requirements would threaten the portability of AP 238 programs. Portability should be one of the most important criteria for STEP-NC.

[1] ISO 14649 has an attribute named "cutting_edge_length" that has the same meaning as "flute_length." In the ARM data, this is the "55.88" in #24. "cutting_edge_length" should have appeared in the AIM data where "flute_length" appears.

2.5 An AIM ⇒ ARM Translator?

ARM data is easier for software developers to deal with than AIM data. Why not build an AIM to ARM translator? This would only be written once, could be made publicly available as open source software, and allow AIM data to be the medium of exchange while keeping software developers happy. There are several problems with this approach, and a better solution is underway.

The problems stem from the fact that AIM data contains more information than ARM data. Specifically, the AIM connects features and workingsteps to STEP boundary-representation geometry, connections that are lost when converting to the ARM. The AIM also allows the units of items to change, from Metric to Imperial for example, and it models tolerances in the same way as AP-203 edition 2, AP-219 and AP-224.

An alternative to a translator would be the availability of a software library implementing ARM-like data access code (such as `the_hole->diameter()`) for AIM data. The implementer would not have to go through complex procedures for data access. Reverse functions could be built into a file writer, so the application programmer could create an AIM Part 21 file from in-core ARM data structures. If this software existed, no ARM files would need to be written. STEP Tools' STEP Indexing (STIX) software does this; presumably other STEP vendors will follow suit.

3 Usefulness of AIM in protecting legacy data and systems

The previous clause illustrates some of the costs of implementing the STEP-NC AIM. This clause makes the argument that the STEP-NC AIM allows for growth without disturbing existing applications, and that benefit is worth its cost, under most circumstances.

The separation of the implementation-level model from the requirements model in the STEP architecture isolates the data from certain types of changes, making it possible to change the requirements yet produce upwardly-compatible AIM models and thus continue to support data corresponding to older versions of the AP. There is going to be a second version of STEP-NC and probably a third version as well.

This is one of the most compelling arguments supporting the use of the STEP architecture. However, it is often the last consideration mentioned. When discussions on the relative merits of AIM and ARM implementation start, attention is focused on the obvious costs discussed in the previous clause, although they have the least impact in the long term. The most important question is often ignored: "How am I going to make sure that I can put my data in a safe place and still use it several years from now?"

Here are four scenarios starting with extensions or revisions of an ARM or AIM, and the effects that result. These scenarios are all intended to be realistic, but vary in likelihood.
In all cases, data access source code would need to be regenerated (and recompiled). COTS tools are available that will produce this new source code automatically in less than an hour for a large EXPRESS model.

3.1 Scenario 1: Extending the ARM Data Model

In this scenario, an ARM revision occurs that only adds new classes and subclasses. Here, ARM-based systems and AIM-based systems have the same reactions to the change, except that AIM-based systems have a slight edge because the old executable can probably input new data (but not do much with it). This scenario is likely to occur given that shops run old software for much longer than you would expect. Provided that the extra information is not of interest to them, their tools would still continue to run when they feed them new data.

Scenario 1	ARM implementation	AIM implementation
Executable	The old executable will handle new data that does not include instances of new classes or subclasses. The old executable will not handle new data that includes instances of new classes or subclasses; it will probably reject such data at the front end of processing. The new executable will handle all old data and new data.	The old executable will handle new data that does not include instances of new classes or subclasses. The old executable will not deal well with new data that includes instances of new classes or subclasses. It may accept such data at the front end of processing (since the data is all integrated resources data at that point), but it will probably not be able to do much useful with the data. The new executable will handle all old data and new data.
Application Source Code	New application source code will be required only to handle instances of new classes and subclasses. Application code that handles old classes and subclasses only will need no change	New application source code will be required only to handle instances of new classes and subclasses. Application code that handles old classes and subclasses only will need no change.

3.2 Scenario 2: Revising the ARM Data Model

In this scenario, an ARM revision occurs that adds new classes and subclasses, moves attributes up or down a class hierarchy, and changes the order of attributes in entities. An AIM-based system would have a substantial advantage in this scenario. The AIM was designed to avoid these types of problems by adding some indirection to the ARM data through property relationships, standard mechanisms for describing measures, units, shapes, and other common components.

This scenario has been the case with the development of the majority of AP revisions. The AP teams fine-tune their ARMs. The mappings in the AP act as a buffer, so information can move around some in the ARM while appearing in the same place in the AIM.

Scenario 2	ARM implementation	AIM implementation
Executable	The old executable will not work with new data that contains instances of changed or new classes. The new executable will not work with old data that contains instances of	The old executable will probably work with new data that contains instances of changed old classes. The old executable will probably input but not deal well with instances of

	changed old classes.	new classes and subclasses. The new executable will probably work with all old data, even if it contains instances of changed old classes.
Application Source Code	New application source code will be required to handle instances of new classes and subclasses. Old application code that handles instances of old classes and subclasses only will not need to change for rearranged attribute order, but will need (probably minor) changes for rearranged hierarchies.	New application source code will be required to handle instances of new classes and subclasses. Old application code that handles instances of old classes and subclasses only will not need to change.

3.3 Scenario 3: Extending the AIM Integrated Model

In this scenario, the AIM is extended while the ARM is unchanged. At first glance this scenario seems nonsensical, since the AIM would only be extended to reflect extensions to the ARM. It is important to note, however, that an AIM extension is really the addition of other STEP resources into the data exchanged with the CNC.

Scenario 3	**ARM implementation**	**AIM implementation**
Executable	N/A	The old executable would either report an error when it encountered data from other STEP resources, or could be configured to ignore this data for forward compatibility. The new executable will work with all the schemas.
Application Source Code	N/A	Source code would need to be added to handle the other STEP resources; presumably this already exists but not necessarily freely

3.4 Scenario 4: Revising the AIM Integrated Model

In this scenario, an AIM revision is made without changing the ARM. This would occur if the STEP integrated resources changed, or if it were decided that a new AIM could be better than the old one. While an ARM-based system is much preferable in this scenario, this scenario is the least likely to occur. STEP has produced second editions of most of the integrated resource parts, and the key design constraint on the update was backwards compatibility.

Scenario 4	**ARM implementation**	**AIM implementation**
Executable	N/A	The old executable will probably not work with new data. The new executable will probably not work with old data.
Application Source Code	N/A	Large amounts of application source code will likely have to be rewritten.

3.5 Scenario Comparison

Scenario 1, extending the ARM, is the most likely. It has already happened several times, with the addition of the turning and EDM parts. AIM-based systems have a distinct advantage here. Scenario 2, revising the ARM, would impact both ARM- and AIM-based systems for the same reasons and would necessitate substantial source code rework. AIM systems are at a disadvantage here simply due to the relative difficulty in dealing with AIM data as discussed in Section 2. Because of the impact to both ARM- and AIM-based systems, Scenario 2 is unlikely. Scenario 3 shows the real benefits of combining STEP and NC: data from the rest of the STEP world can be included with less effort than any existing data exchange solution. Scenario 4, in which the AIM is revised to reflect changes in the STEP Integrated Resources, is unlikely for the simple fact that such changes adversely affect all of STEP.

3.6 Requirements for Change

The STEP-NC AIM leverages previous developments in STEP that defined models for 3D geometry and manufacturing features. The STEP-NC model must anticipate future initiatives will use its resources in new models. For example, a model that allows feedback to be sent from the machine tool about its capabilities to the enterprise design systems may soon be under development. A model for inspection is already under development.

Future machine tools will perform many different kinds of operations including, milling, turning, EDM, robotics and material handling and inspection. One machine may perform multiple kinds of operations. This will be particularly true for micro-machines where NC machining will play a large role in creating the miniature planes, spacecraft and medical devices of the future. This is another requirement for extensibility.

The STEP-NC standard for milling is available today. There will be others for milling, turning, bending, contouring, inspection, robotics and all other processes that require digital manufacturing data. All these processes may be supported by future extensions to STEP-NC.

4 Benefits of STEP integration

A primary goal of STEP is to facilitate interoperability among implementations of different STEP application protocols. This is accomplished through consistent mapping of similar requirements to the generic building block data structures in the integrated resources. In an AIM implementation, the implementation may leverage data that is available in other STEP AP formats due to the effort made to integrate and harmonize STEP APs. Integration is another benefit of STEP that offsets the cost of the additional lines of code and size of data files.

There are three models that must be supported by STEP-NC: the machining model developed by ISO 14649, the tolerance model developed by the aerospace industry (for support of on-machine inspection), and the PDM model developed by ISO 10303 (for integration of the machine into the wider enterprise). The STEP integration method allows the three object models to be integrated into one application protocol.

The association to AP 224 features and AP 203 geometry is retained in the STEP-NC data, allowing the set-up to be changed. Without these associations you get a STEP-NC program that can only be run for one machine set-up, a fundamental issue with RS274D.

Many enterprise applications need to read and write STEP-NC data not just CNC controllers. These can include planning applications, analysis applications, CAD applications and CAM applications. Each application may only need to read and write a subset of the data. The STEP integration architecture allows the different perspectives of these applications to be supported within the one AP 238 model as shown below. For example, a CAD application can read the AP 203 subset of AP 238, a CAM application can read the AP 224 subset and an inspection application can read the AP 219 superset.

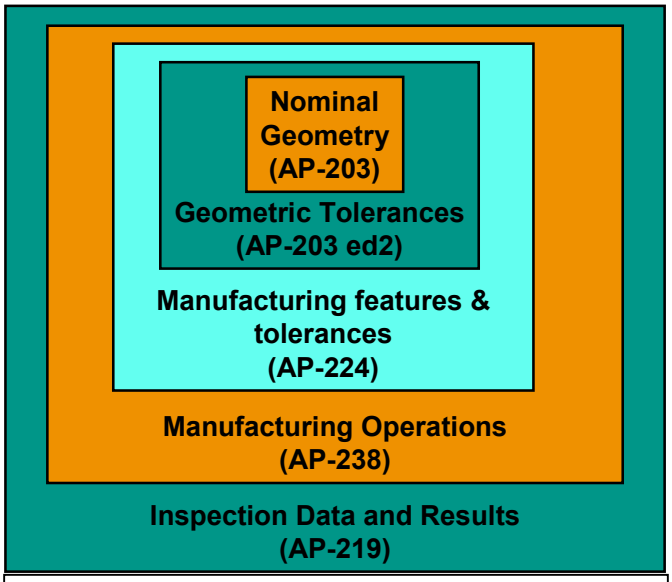

Figure 1 Integrated STEP data specifications

5 What is the difference between 14649 and AP-238?

The AP-238 model encodes the 14649 model in the AIM so why is there a difference between the two models? Because by integrating, AP-238 makes it possible to insert data connections linking the 14649 manufacturing data to the design data defined by AP-224 and AP-203. This is a key difference between the AIM and the ARM.

In AP-238, integration allows the manufacturing data described by 14649 to be linked to the design intent information described by AP-224 and AP-203. This means that the reason for each manufacturing choice is documented in the data so if manufacturing conditions change then appropriate new manufacturing data can be generated from the original design intent information.

So far the Super Model project has documented two scenarios where this makes a practical difference.

The first scenario is in the computation of speed and feed data. Using AP-238 it is not necessary for speed and feed data to be computed until the very last stage of manufacturing because the design tolerances and surface finishes required for the product are in the AP-224 and AP-203 data and connected to the 14649 data. Therefore intelligent software on the CNC can generate the best speed and feed for the current operating conditions of the tool, and the NC programming software that generates the STEP-NC data does not need to know how to set them. This allows the NC programmer to save time and it allows the AP-238 STEP-NC programs to be more portable.

The second scenario is related to the first. Another STEP-NC attribute that does not have to be set immediately in AP-238 is the orientation of the part on the machine. In 14649 this information is crucial because each operation is defined using a profile that defines the boundary of the material volume to be removed in the XY plane and a depth for removal. The origin of the XY plane must be correct because for the part to be machined correctly.

In AP-238 there is more flexibility because the AP-224 features are axis and orientation independent. Therefore, the connection to the AP-224 feature provided by AP-238 allows the axis and orientation of each removal operation to be recomputed if it needs to be changed for a new set-up or machine. This is a crucial capability and the only way to remove post-processors from manufacturing because if your STEP-NC files are set-up specific then very little has been gained over RS274D.

6 Consequences of the 14649 limits

The technical limits of the ISO 14649 architecture seriously impact the business benefits of STEP-NC.

AP 238 enables the manufacture of parts on any machine that has the required capabilities. All the information required to make the part is in the data. *However, the ISO 14649 method does not allow a STEP-NC file to contain proper tolerance and PDM information.*

AP 238 can communicate all the information required to manufacture a product between engineers, across the supply chain from design to planning to manufacturing. *However the ISO 14649 model only supports one kind of exchange from CAM to CNC and does not recognize that many other kinds of applications may want to read all or some of this data*.

The USA Super Model project has predicted STEP-NC will make the task of path planning 35% faster because less information has to be defined and because 3D feature recognition can be used to accelerate the definition. The project has also estimated that small to mid-sized machining jobs will complete 50% faster because the information in a STEP-NC file allows a CNC to do full safety checking and to compute the optimal speeds and feeds for the cutting tools automatically. Therefore, high speed and five axis machines will be used more often for small to mid sized job lots. The Manufacturing Road Map has estimated that by value such job lots represent 75% of manufacturing. *The 14649 model supports the requirements of CNC vendors for easier shopfloor programming. For faster CAM planning, CAM systems must be able to read the data as well as write it so that they can re-plan a plan when necessary and 14649 does not*

support this case because it losses the original CAD data used to make the first plan. Better machines can only be used for small job lots if full software checking is available and this can only be done if integrated, design, inspection and manufacturing information is passed into the control. For faster machining it must be possible to fully simulate and check the program on the CNC control using software. If full software checking is not possible, then such checking will continue to be the responsibility of the operator.

The USA Super Model Project has tried to exchange a 14649 file with the STEP-NC IMS project in Germany. The Super Model project could not create a legal 14649 file for transmission to Germany because in the USA we did not know the characteristics of the tool that would be used to machine the file. Therefore, we were unable to set the correct speed and feed information or select the right cutting tools. We could send a file without this information to Germany, but using 14649 we could not communicate sufficient design intent information to allow them to set the feed and speed data over there.

The experiment shows that 14649 is sufficient to allow data exchange only when the CAM engineer knows enough about the machine tool to set the cutting parameter data. This will most often be the case when the CAM and CNC systems are on the same shop floor and will rarely be the case for International data exchange.

7 Summary

AP 238 AIM files are more complex, less human-readable, and significantly larger than ISO 14649 files. Programming against the AP 238 AIM is more complex and requires more lines of code than programming against ISO 14649 models. These costs are worthwhile for those who have requirements for long-term extensibility and integration with current and future STEP applications.

AP 238 leverages benefits of the STEP architecture; it is integrated with other application models.

AP 238 carries all the required information from CAD to CAM to CNC. ISO 14649 loses the 3D model information. Consequently, ISO 14649 offers less automated checking for collisions, less automated compilation and optimization, and less portability between machines and systems.

8 Recommendations

1. Implement the AIM in the cases where long-term data retention, future modification and extension of the implementation are important.

2. To compensate for verbosity of AIM data, have a preprocessor scan for duplicate entities, compact files by removing whitespace, and consider incremental updates to large files.

3. Use public or commercial code libraries where possible to focus programming efforts on the application-specific detail that are most important.

Annex A

STEP architecture primer

The architecture of STEP is designed to support the development of standards for product data exchange and product data sharing. The architecture is governed by the following concepts: (1) the scope of what is standardized and what is conformance tested is set at the level of "an application," (2) application requirements are based on a model of a business activity, (3) application requirements are standardized using natural language (with an accompanying data model), and (4) a mapping is specified defining how the application requirements are satisfied using a STEP data model.

Figure A.1 provides an overview of the primary components of the STEP architecture. There are two distinct types of data specifications.

- **context-independent** specifications, called integrated resources, are the building blocks of the standard.
- **application-context-dependent** exchange specifications, called application protocols, are developed to satisfy clearly defined industrial information requirements. These are the parts of STEP that are implemented.

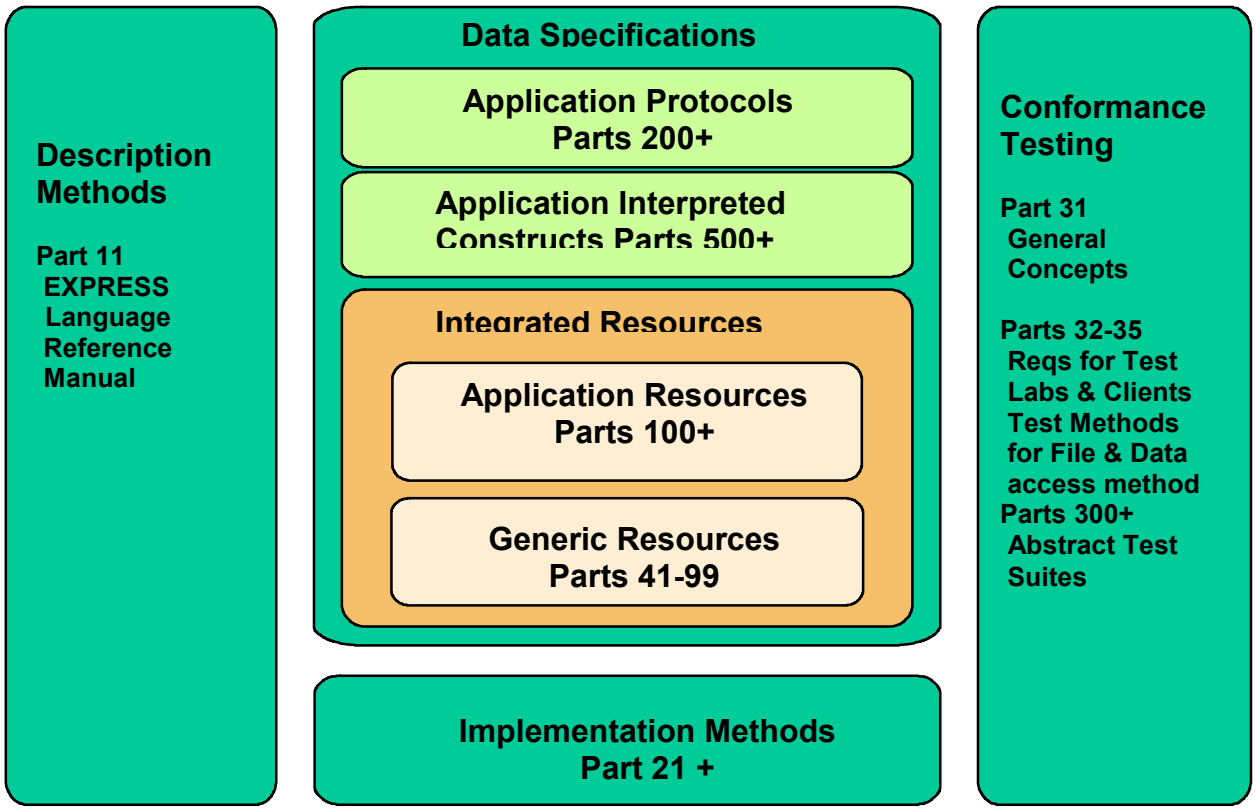

Figure A.1 STEP architecture

The other components of the architecture provide the infrastructure of STEP.[2]

The integrated resources are a consensus semantic structure that spans product data usage and application for all industries. The integrated resources are a single, consistent, semantically non-redundant model. The integrated resources were designed with several principles in mind, primarily reusability and extensibility. The model is partitioned into small schemas, each with a unique functionality or purpose. This partitioning facilitates management of the model and isolates specific functionality. This enables the model to be extended when additional industrial requirements are identified. The constructs of the integrated resources are of two types: (1) specialized data models with strong semantics, and (2) general data models with weak semantics. If the semantics of the application match those of one of the specialized models, that specialized model is used. If not, the semantics of the application are represented using the general data models with detailed constraints embodying the application semantics. The structure of the IRs is such that there is a "vocabulary" and a "grammar" that is shared by all APs and enables

[2] **Description methods** include the SC4-developed information modeling language EXPRESS. Their use supports computer-sensible standards specifications.

Implementation methods include physical file exchange format and data access language. Data definition is separated from the exchange format and the data access language.

Conformance testing methods provide a built-in basis for assessing conformance of implementations. SC4 standardizes abstract test suites as well as the testing methodology and framework. Standardized test suites promote repeatability and consistency of testing.

implementations of different APs to read files created by implementations of other Pas according to the degree that they represent common requirements.

An application protocol represents a measurable and shareable subset of STEP capability that is expressed in the terminology of a specific industry or discipline. An AP uses application domain terminology, hence it is understandable to potential users of the AP. The AP is a measurable subset because it specifies what data structures are allowed in an exchange file and the constraints on that data. STEP reconciles the requirements for a generic, reusable vocabulary with the semantic richness of industrial applications by defining a particular application context and explaining the meaning and usage of standard data constructs (i.e., product data) within that domain.

An application protocol specifies:

- The context within which the AP is intended to be used.
- The set of user's information requirements the AP intends to meet.
- The exchange specification that is the selection of building blocks from the IRs, including specialized semantics and additional constraints on the generic constructs.[3]

There are 4 major components of an AP.

- The **AAM** partially describes the context of the AP. It describes the activities that may be supported by the AP, and serves as a requirements gathering and scoping tool.
- The **ARM** also partially describes the context. It serves as both a requirements discovery mechanism and as a requirements specification mechanism.
- The **AIM** has several functions. It specifies the subset of the IR "vocabulary" used with the AP. It specializes the semantics of the generic data structure and adds constraints, thus providing the exchange specification for the AP.
- The **mapping table** specifies the relationship between the ARM and the AIM. It contains rules and constraints specify the usage of the IRs for the AP. The combination of the AIM and mapping table completely specify the usage of the generic structures for the AP.

STEP incorporates the rich semantics and constraints of individual application domains while maintaining a limited, generic, reusable set of data constructs as an exchange mechanism.

[3] **Context**: A narrow, well-defined domain and a focus on a specific interface or interfaces will enable the precise identification of information requirements.
Information requirements: The specification of a collection of concepts and their meaning that are required to perform an activity defines the functionality of an application protocol.
Exchange specification: The application protocol specifies the subset of the integrated resources, constraints and specializations needed to meet the requirements.

Annex B

Physical file comparison details

B.1 Data generation method

Both sets of data were generated automatically. The ISO 14649 data has not been modified. The ISO 10303-238 has data has been modified (1) to compact it by eliminating redundant copies of #58 and #495 and unnecessary newlines so that the two data sets are at roughly the same level of compaction, and (2) to get the drill data to match the ISO 14649 data.

The generation path of both sets of data starts with a Part 21 file that corresponds to the ARM of ISO 10303-224. This hand-written file describes a block with a one feature, a 12.7 millimeter hole in the top.

The file containing the ISO 14649 data was produced by:

1. Inputting the ISO 10303-224 ARM file to FBICS and having it output an ALPS Part 21 workstation-level process plan and associated data.
2. Inputting the ALPS plan and associated data to the alps2stepnc translator and having it output an ISO 14649 Part 21 file.

The file containing the AP 238 data was produced by:

1. Inputting the ARM 224 file to FBICS and having it output a Parasolid file.
2. Inputting the Parasolid file to the STEP Tools Parasolid-to-AP 203 service and having it output an AP 203 file.
3. Inputting the AP 203 file to the STEP Tools ST-Plan system and having it output an AP 238 plan file. This step was performed by Martin Hardwick.

B.2 Complexity of programming against an AIM

To find the diameter of the drill used in the drilling operation in the AP 238 data, one might perform the following steps. Reference is made to the instances to help follow the procedure. The drilling operation is #133.

1. Make a list of all instances of machining_tool in the file.
2. Sort through the list found in step 1 to find the machining_tool that is used in the drilling operation. [result: #90]
3. Make a list of all instances of resource_property in the file.
4. Sort through the list found in step 3 to find the one whose name is "tool body" and whose resource is the machining_tool found in step 2. [result: #82]
5. Make a list of all the resource_property_representations in the file.
6. Sort through the list found in step 5 to find the one whose property is the resource_property found in step 4. [result: #78]
7. Take the representation (which should be a machining_tool_body_representation) of the resource_property_representation found in step 6 and extract from it the items, which is a list of representation_items.

8. Sort through the list found in step 7 to find the item whose name is "diameter". [result: #416]
9. The item found in step 8 should be a simultaneous instantiation of several different entity types, one of which is measure_with_unit. The diameter is the value_component of the measure_with_unit portion of the item.

It is unlikely that the procedure described above could be implemented in fewer than twelve lines of code. If other properties of the tool were required, however, only steps 8 and 9 of the procedure would need to be repeated.