H12－STEP

STEP 適用推進に向けた検討結果の報告

平成 13年 3月

電子商取引推進協議会
普及促進 WG
STEP　SWG

はじめに

製造業の分野において、従来一般に図面等で表現されていた製品情報の企業間情報交換、共有が近年の IT 化の進展にともない大きく変化しようとしている。普及促進 WG STEP-SWGでは、製品情報交換・共有へ IT を活用する立場からデジタル製品情報を授受し活用するための検討を行なった。

まず、デジタル製品情報を授受し活用するための課題全般をまとめた。

併せて、設計情報の生産準備業務への活用を検討し、このための基盤技術としての製品モデルの役割と製造分野での製品モデルの現状をまとめた。

また、STEP を使いこなすための方策として、製品データモデルを、業務に身近なものとして理解していただくために、STEP データモデルを Java で実装し、それを XML 出力する方法を例示することにした。

目次

1　デジタル製品情報を授受し活用するための課題

IT 化の進展にともない、製品情報交換・共有への近年の大きく変化しようとしている。その背景は、次の３つに纏められよう。

（1）設計・加工業務の電子化

1980 年代から製図作業の 2 次元 CAD（Computer Aided Design）化が進み、1990 年代とくに後半から 3 次元 CAD による設計の実用化が始まった。

CAD 以外の製造（CAM：Computer Aided Manufacturing）、解析（CAE: Computer Aided Engineering）、検査（CAT：Computer Aided Testing）等の分野へのコンピュータ支援も急速に進められており、コンピュータ内に作成した製品データに基づく設計･製造の事前検証も行なわれている。

（2）ネットワークでの協業時代への突入

インターネットに代表されるネットワーク環境の充実は、製品情報を関係する企業間でやりとりすることを容易にし、多くのアプリケーションシステムやソフトウェアが相互に製品データを交換する状況を作り出している。

（3）情報を企業資産とみる考え方が広まってきた

蓄積されたデジタル製品データを企業の情報資産として長期にわたり保存･活用することへの関心が高まっている。

一方、製品情報は様々な方法や形式で電子化されており、このデジタル情報を企業間、部門授受し活用するためには、多くの問題が見受けられる。この問題を、IT の活用により解決しようとする時、課題の全般を掴むことが大切となる。

そこで、デジタル製品情報を相互に授受し、活用するための課題を以下の観点から列挙した。

- 設計・製造における、デジタル製品情報の活用の目標
  設計・製造部門の、製品情報のデジタル化して活用することによる期待効果を列挙する。

- 設計・製造における、デジタル製品情報の活用の課題
  設計部門・製造部門の各部門内における、製品情報のデジタル化して活用して行く上での課題を列挙する。
- 部門間におけるデジタル製品情報の授受と活用の課題
  設計部門、製造部門の各部門から見た、部門間におけるデジタル製品情報の授受と活用に関わる課題を５つの視点から列挙する。
  －他事業体の部門との授受における共通課題
  －上流事業体の部門との授受における課題
  －対下流事業体の部門との授受における課題
  －設計部門と製造部門間の授受における課題
  －同一事業体内の調達部門、営業部門との授受における課題
- デジタル製品情報の活用に当たってのシステム関連の課題
  デジタル化された製品情報を作成、変更、交換、共有するためのソフトウェア、ハードウェア、インフラストラクチャに関する課題

以上の内容を図 1-1 に示す。

図 1-1 のその他の欄は、そのいずれにも直接は属しないが、課題として考えておくべきことを列挙した。また、これらの内容を検討した各委員の考える、

- 短期的に解決しなければならないもの
- 長期的に解決しなければならないもの

とコメントを表 1-1 にまとめた。

各委員の所属している、産業・企業・立場により表現の仕方や視点は異なるが、複数の委員より指摘のあったことを抜き出すと以下のようになる。

短期的に解決しなければならないものとしては以下があげられる。

・IGES,STEP にかかわらずデータ流通性の確保と長期保存。（ツールの頑健性、データの品質、管理方法の確立等）

・デジタルデータの保証（セキュリティ、著作権、データ改竄、変換後の正しさの保証等）

・設計・製造における３ＤCAD 活用のレベルアップが必要。

・開発プロセスの改革や新ビジネスモデルの構築までも視野に入れることが必要。

また、長期的に解決しなければならないものとしては以下のものが上げられる。

・3DCAD が構想設計、詳細設計、型設計などを通じて過不足なく使えるような進化が必要。

・データ変換などの余分なことが意識しなくても良いサービスがもっと必要。

・上流事業体は今後の動向や計画を公開し、各下流事業体が対応しやすくする。

・中小製造業の企業向けインフラとして、政府援助のプロジェクトなどは取り組む価値がある。

これらを概観すると、産業、企業規模、業務により、デジタル化の進め方や程度が異なるため、デジタル化された製品情報を、産業内にスムースに流すためには、データ変換にとどまらず、企業間部門間のビジネスプロセスをつなげる観点からデータ交換や共有を取上げることが必要であることが読み取れる。

また、多くの委員から産業全体で考えることが必要との共通の指摘があったのは、デジタル製品情報の保証（セキュリティ、著作権、データ改竄、変換後の正しさの保証等）についてである。

今回は課題の列挙で終わったが、今後具体的な方策を検討するためには、個別のフィールドについての検討・調査が必要となる。

１．設計・製造における、デジタル製品情報活用の目標（期待効果）

①市場を継続、あるいは創造できる製品（ミスのない）を廉価な価格で迅速に提供

（a 設計の場合）

①関連部門との情報共有設計による期間短縮。
－企画と設計のタイムロスの無い情報共有化
－部門～他工場～海外拠点､協力会社
－製品情報（図面、製品構成データ､設変内容）の後工程との授受。
－CAE データの授受。

②仮想試作による上流での機能、品質、コストの作り込み（試作回数の低減）。

③設計開発リードタイム短縮
－３D-CAD,PDM によるコンカレント設計・開発、設計データの再利用

④設計～製造情報（ノウハウ）の蓄積、再利用確立（ナレッジマネジメント）

（b.製造の場合）

①ｸﾞﾛｰﾊﾞﾙに設計情報を利用することによる製造期間短縮。

②各部品の構成情報、生産要件の入れ込み、および設計部門へのフィードバック

③試作／制作リードタイムの短縮
－3D-CAD から試作品／金型などの制作データを自動作成

④３D-CAD から CG による加工／組立作業指示を作成

２．設計・製造における、デジタル製品情報活用の課題

（全体）

①仮想試作検証の方法の確立が必要。
－特に、ｼﾐｭﾚｰｼｮﾝﾍﾞｰｽのツール連携技術確立が必要

②２D-CAD から３次元 CAD への活用進化が進んでいない
－3D モデルと図面の有効な混在利用方法の検討
－図面を無くしたい。そのための阻害要因の洗い出しと対策具体化
－３D モデルの有効利用、有効利用が可能なモデル作りの検討
－３D への移行では、更新ではなく作り変えが必要で、多大な費用が見積もられる。

③管理体制の確立

④ユーザへの啓蒙、教育

（a 設計の場合）

①後工程、試作品評価後の再設計が負担
－3D モデルを作成しても、製品検査用に２D 図面（公差等記した）を作成。

②インフラ設備の価格が高すぎて設備投資が遅れている。
－ナレッジを蓄積、共有化が実現できる具体的な考え方、ツールが不足。

③モデリング技術、ノウハウの蓄積に時間がかかる。

④3D モデルの品質が設計者により異なる。

（b.製造の場合）

①必ずしも有名 CAD が CAM に強いわけではないため、必要な CAM を使用する場合はデータ変換が発生する。

②設備投資資金繰りから設計サイドへの設備投資が先行し、製造サイドへの投資は遅れがち。
－3D ﾃﾞｰﾀの活用出来るﾏｼｰﾝ設備が少なく、又高額(高額の割に機能も不足含めの為投資に踏み切れない。

③操作の煩雑さ･難易さがあり、製造部門として容易に 3D を受入れがたい。

３．部門間におけるデジタル製品情報の授受と活用の課題

（対他事業体共通）

①データの２重持ち、情報伝達のタイムラグ、データ品質の劣化。
②授受異常時の対応（サポート）をどうするか
③著作権／セキュリティ／データ保護などについての問題点あり
④送受における図面データセキュリティと外注先での情報設備の対応の問題あり。

（a 対上流事業体）

①上流事業体の動向に左右される。
－上流事業体での情報の造り込みに対するコスト再配分の考え方が必要。
②紙図面の提出が必須の場合がある。
③メーカサイドで作成したデジタル製品情報を顧客サイドで閲覧が出来ないため、紙による提供が必要。

（b.対下流事業体）

①零細な事業体には 3D モデルに対応する為の体力が乏しい。
→と言っていると、更に世界に負ける
②設備投資は下請け企業側にとっては高負担となっている。
③中小のメーカが多い為、ハイエンドの CAD を使用しているとは限らないため、データ変換が発生。（→シンガポールのように、大学が安い価格で変換サービスを実施するなどの方策が必要）

（c.設計・製造部門間）

①試作品作成後の不良箇所発覚による再作業の発生。
②設計者が作成した３次元モデルだけからでは加工できないため､加工用のモデル作成作業（面の再作成等）が必要。
③図面より CAM 情報を作成が必要。
④設計側としては、寸法指示なし図面の実現を目指すが、製造側での受け入れ態勢の整備が困難な状況。下記要件の解消が必要
－検査をどうするのか。照査・検認作業をどうするのか。－データの保存をどうするのか。－データ管理体制をどうするのか。－生産要件・公差情報をどのように扱うのか。－設計変更の履歴管理をどのようにするのか。
⑤設計部門の CAD とは異なるため、設計変更の際のタイムラグ有り。
⑥設計部門と製造部門での 3D 化実施でﾀｲﾑﾗｸﾞあり。
－3D による先行工程開発準備に対し設計の 2D による先行出図。
－2D 図面での手配の帳票のｾｯﾄ流通が、3D 設計では未確立
→情報が分離し作業が面倒。
⑦詳細部品･詳細形状までの 3D 化が工数増大の為､工程開発に限界が発生。

（d.事業体内の対調達・営業部門等）

①ホスト系システムのため､設計／生産製造部門のシステムとリアルタイムのデータ連携が難しい。
②零細企業の力(ﾉｳﾊｳ･ｺｽﾄ･日程等)に頼った調達が多く、調達部門での 3D 化展開が今後の最大のﾈｯｸになる。

４．デジタル製品情報の活用に当たってのシステム関連の課題

（a.ソフトウェア）

①各システムでの操作性，言葉、定義が異なる。
－開発ﾍﾞﾝﾀﾞｰに対しての共通化、標準化（ある程度）の申し入れが必要（ｸﾞﾛﾊﾞﾙに標準化）

②３D-CAD 修得に時間がかかる。設計者の使う道具になっていない。
－モデル作成／変換へのベンダサポートの強化（ソフトと設計の分る人材要）。
－社内体制の強化(ｿﾌﾄと設計のわかる人材要)。
－３D-CAD 修得とノウハウの蓄積。

③CAD データをいつまでサポートしてくれるのかの不安がある。

④形状データだけでなく製品構成､属性情報の変換が難しい(STEP の進展必要)
－形状データの活用を促進する為の CAD のﾄﾗﾝｽﾚｰﾀの進展も必要。

⑤バージョンアップ毎にトランスレータを買うのは困難。
→各 CAD ｼｽﾃﾑのﾄﾗﾝｽﾚｰﾀは、保守費用(購入費用の 5%程度)の範囲で対応してもらえる事で各種ﾄﾗﾝｽﾚｰﾀ購入企業では問題にはならないが、中小企業でのｺｽﾄ負担として。

⑥ともかく高価。(余計な費用負担)

⑦3D ﾃﾞｰﾀの他部門への活用の為ﾋﾞｭｰﾜ導入時、各 CAD ｼｽﾃﾑ毎に開発思想が異なっている為各 CAD ｼｽﾃﾑ毎のﾋﾞｭｰﾜと変換ｿﾌﾄの導入が必要となっている。また、非公開ｿﾌﾄでは、ﾋﾞｭｰﾜ対応への遅れや不具合対応が出来ない事もある。

（b.インフラ）

①データ管理システム（PDM/ERP など）の導入／構築に多大な費用がかかる。
②Web 技術（JAVA/XML 等）が発展途上のため、上記システムのバージョンアップ費用多大。

５．その他

①デジタル化が負担となっている部門および業務が存在する。業務の中にデジタル化が割り込んできているためで、デジタル化のメリットが出ていない。
今後の大きな課題として、業務プロセスの改革によりデジタル化、標準化、業務の簡素化を行わなければならないと考えている。
－データ変換というあまり創造的でない仕事はしたくないのが本音である。
－デジタル化の問題は文化と組織の壁が依然として存在すること
－ベンダー間の壁の存在

②これから製品情報の活用を行う企業には、導入事例の公表が普及に繋がる。

③IGES，STEP ともになかなか理想どおりに行っていない。その理由をもっと突き臭いレベルで議論することも必要。

－交換した後のモデルの正しさの保証をどうする（検証の仕組みが必要）。

－現状バージョン、パッチ番号まで同じでなければならない（交換ルールが必要）。

図 1-1 デジタル製品情報を授受し活用するための課題 -問題

表 1-1 コメント

短期的に解決しなければならない課題	① IGES,STEP にかかわらずデータ流通性の確保と長期保存。（ツールの頑健性、データの品質、管理方法の確立等） ② データ交換ツールも含めたインフラの整備と活用技術（操作方法の教育も含めた）の確立。 ③企業文化、業務プロセスの変革。	①データ交換がなくならないとして、データ交換処理ツールの玉成も必須。 ②互いのツールの相性を事前に調査し、変換の仕様を合わせるといった作業が必要。 ③その為のｿﾌﾄ及びﾉｳﾊｳを公開し、誰もが良い変換結果を得られる様な仕組みが必要。 ④3D ﾃﾞｰﾀ活用時の検査結果について認証方式に対する各社共通技術(判断基準)の確立。	**項番５（その他）：①②③** **項番４(a)ソフトウエア**： **①**各システムでの操作性、言葉、定義が異なる **④**形状データだけでなく製品構成、属性情報の変換が難しい **⑤**バージョンアップ毎にトランスレータを買うのは困難	・設計における３ＤCAD 活用のﾚﾍﾞﾙｱｯﾌﾟが必要（ｼﾐｭﾚｰｼｮﾝを前提とした商品開発、設計の姿にするために） ・そのために、簡易図面標記の標準化が日本の業界全体に必要 ・データ変換などの余分なことが意識しなくても良いサービスがもっと必要
長期的に解決しなければならない課題	① 製造ノウハウをどのようにして、デジタル情報として保存、検索、活用するか。 ② デジタルデータの保証（セキュリティ、著作権、データ改竄、変換後の正しさの保証等）等法律の面でも必要か。	①ﾃﾞｰﾀ変換等の非効率作業の撤廃。 ②上流事業体は今後の動向や計画を公開し、各下流事業体が対応しやすくする。一対応をすべきかのビジョンを見据える事ができる。 ③CADｼｽﾃﾑ(ｿﾌﾄ･ﾊｰﾄﾞ共)の機能向上と技術向上(使い易さと信頼性向上)。	**項番１：全項目**（特にb項製造の場合の①②③④については、あまり情報がなく、年間計画などで取り組む価値があると思います） **項番３(b)①②③**は、中小製造業の企業向けインフラとして、政府援助のプロジェクトなどをＥＣＯＭで組織して取り組む価値があると考えます（日本電機工業会では、重電ＳCMプロジェクトとして、旧通産省の援助金で配電盤工業会とも組み推進しています）。ただし、STEP-SWG で取り組むためには、もっと議論を掘り下げ、具体的にする必要があると思います。	・3DCAD が構想設計、詳細設計、型設計などを通じて過不足なく使えるような進化が必要 ・異機種間のデータ互換性を保証できるよう、３Ｄカーネルの統合（世界規模で） ・ネットワークを通じて３Ｄデータが容易にﾊﾝﾄﾞﾘﾝｸﾞできる技術（ＸＶＬなど）の進化（ＸＶＬを是非世界標準にする動きが必要）
その他	① デジタル製品情報を活用するメリット及びデメリットを明確にして、成功例だけではなく、失敗例の公表が必要ではないか。それらを参考として、各事業体、各部署におけるデジタル化における戦略、業務プロセスの変革が最優先であると考えます。 ② その中で自部署が必要なインフラの整備（GES,STEP 等のデータ交換ツールも含めて）が必要なのではないか。 ③ 上記を解決するために、現在の技術で何ができるのか、どこまでできるのか、何ができないのかを明確にするようなガイドラインが必要ではないかと思います。また公のガイドラインの作成機関、サポート機関の設立が必要かとも思います。 ④中小の企業や、体力のない企業がデジタル化を推進するためには、上流事業体や先進技術に極端に左右されない技術が必要かと思います。少なくとも 5 年以内は使える技術。普遍的とは言わないまでも、標準化された技術が必要でその技術の有力候補が STEP であると思います。	① 部品製造メーカは、客先の動向に合わせた対応をする為、車両メーカの動向の早期公開が必要。 ② 出来れば業界内でコンセンサスを取り、バージョンの均一化や、バージョンアップ時期を合わせる事が必要。 ③ 各ﾒｰｶの CAD は同じ機種を使用していても、ﾏｲﾅｰ機能の追加や環境の違い等でより専用化されてきており同一 CAD でない為、今後も個別に環境を整え対応する必要がある。 ④ 単にＣＡＤオペレーションを覚えるだけでなく、個別のルールに沿ったデータ作成対応を行う必要がある。		日本企業の優位性がどんどん失われている中で何をすべきか、と言う観点に立つと、今回の課題はまだまだ狭いように思います。もっと世界の動向（ＣＰＣなど）も踏まえての議論が必要な気がします。

表 1-1 コメント（つづき）

短期的に解決しなければならない課題	**項番４(a)**⑥ともかく高価 **項番３(c)**④寸法指示なし図面の実現 **項番３(対他事業体共通)**③著作権／セキュリティ／データ保護	**項番３（対他事業体共通）**③著作権／セキュリティ／データ保護 **項番３(c)**⑤設計部門の CAD とことなるため、設計変更のタイムラグ有り	・新技術の習得と試行 ・インフラの整備 ・全社を上げて、新プロセスの検討プロジェクトの発足	
長期的に解決しなければならない課題	**項番４(ａ)** ④形状データだけでなく製品構成、属性情報の変換が難しい。 ③CAD データをいつまでサポートしてくれるのか不安がある。 **項番３(ｂ)** ①零細な事業体には 3D モデルに対応する為の体力が乏しい。 ②設備投資は下請け企業側にとっては高負担となっている。 ③中小のメーカが多い為、ハイエンドの CAD を使用しているとは限らないため、データ変換が発生	**項番３（ａ）**①上流事業体の動向に左右される可能性がある。 **項番５（その他）**③IGES,STEP ともになかなか理想どおりに行っていない。その理由をもっと泥臭いレベルで議論することも必要。	・企業の e-ビジョン作成 ・長期計画作成	
その他	＜優先順位について＞ こんな所かなと思う部分を示します。あんまり深い根拠、考えではありません。要は、価格が高いため導入が進まず、普及が遅い。設計側は何とか３Ｄ化が出来ても製造側が対応出来ず、結果的に経営者の判断基準であるコストダウンに結び付けられないというジレンマに陥っていると思います。	建設業界は、設計・製造（施行）の基本的な分離という業態である。このため、従来から、設計段階で作成される図面は製品の出来上がりの図であり、施行段階で必要とする図面は作り方の図であるといわれている。このような業務分担が行われている業態の中においては、下流で必要となる情報を上流段階で作り込むことの業界としての合意形成とコスト構造の再配分に対する理解が必要である。 このため、現在の建設業での設計・製造におけるデジタル製品情報活用の範囲は、下流側が使える情報だけを使っているという程度で、積極的に設計・製造間のデジタル情報のあり方を論議し、トータル製品コストの低減という業界にとってのテーマについては、今後という状況と思われる。当面は、設計・施行一環体制で施行可能な企業、そのような案件において、３ＤＣＡＤの利用を含め、生産設計という概念で、従来の設計図と施行図の統合化が進むものと考えられる。	戦術から戦略へ早急に変換すべきです 新技術の見極て導入するかどうかは、企業の戦略の問題である。日本の企業の取り組みは、戦術レベルの導入でとても開発プロセスの改革や新ビジネスモデルの構築まで視野に入れている企業は少ない。 中小企業に対して、人材・資金の援助は、行政の仕事であり、セキュリティー・知的所有権については、業界活動が必要である。	STEP が業務で本当に使われるようになれば、上記課題の少なからぬ部分が解消すると思われる。その意味で未だ STEP の意義は失われていないと思う。しかし、STEP はあまりに理念優先で、特に AP の複雑さは問題であり、規格化に長い時間がかかる上に、利用者側としてはその正しい解釈に多大の労力が必要とされる。 AP の開発に労力をかけるよりも、統合リソースレベルでのマッピングのガイドラインを整備した方が、STEP 普及のためにはむしろ重要ではないかと考える。 **項番４デジタル製品活用に当たってのシステム関連の課題**：STEP こそ、これらの問題の多くを解消するものであったはず

2　設計情報の生産準備業務への活用検討

2.1　はじめに

日本の産業競争力が強い理由としてその製品のコストパフォーマンスの良さが世界中の消費者に高く評価され支持されてきたことがまず挙げられる。すなわち、消費者が欲する性能、デザインの製品が安く提供され、また製品の性能、使いやすさ、更に故障が少ない等の製品の信頼性が消費者の満足度を充分満たした事で世界中の消費者に長い間支持されて来た事にあるといえよう。このような製品はそれを作る人と生産設備から生み出されるが、この製品を作り上げる「ものづくり技術」がすなわち日本の産業競争力の源である。このような「ものづくり技術」は製品企画、製品設計、生産製造、物流の各工程を企業の経営者と従業員がいままでの経験と技術的蓄積とたゆまない努力により作り上げ維持してきたものであるが、昨今の情報化技術の製造業への浸透によりいままでの「ものづくりの方法」を根本から変える新しい「ものづくりの方法」が作り上げられようとしている。

本報告は新しい世紀を迎えるにあたり、このような「新しいものづくり」の取り組みを実現するための、設計情報の生産準備業務への活用検討を行ない、今後の「ものづくり」に対する情報技術からの提言を行なう事を目指すものである。

特に、本報告では日本の代表的な産業分野である機械系産業をその対象としている。また、製品設計、生産設計、製造、流通工程の中で日本の機械系生産システムが得意とする機械製品の「製造・加工」に関わる生産設計を中心に調査分析を行ったものである。

2.2　マニュファクチャリングの情報システムとは

図 2-1は機械系生産システムのプロセスを表現したものである。

製品はマーケッティング部門を中心とした製品計画、製造計画（Ａ１）分析が行なわれ「開発スケジュール」「製品仕様」「生産計画」等が決められ製品開発が着手される。つぎに、製品開発（Ａ２）では既存のモデルをもとに製品仕様を参照して具体的な製品の設計が行なわれる。製品の設計が完了すると製品モデル情報をもとに製造のための設備設計・生産準備（Ａ３）が行なわれる。生産情報が完成すると工場の生産管理（Ａ４）情報をもとにした資材・原料の調達（Ａ５）、製品の製造（Ａ６）が行なわれる。

現在このような一連の機械系生産プロセスを見てみると様々な情報化ツールが存在する。例えば、製品戦略のために過去の情報をデータウェアハウスに蓄積して分析予測す

るマネージメント支援システムや工場の資材調達のＭＲＰシステム、ＥＲＰシステムさらに製品設計分野では３次元ＣＡＤやＰＤＭを中心にしたソリューションが出現して活用され始めている。ところが、製品生産加工のための生産設計・生産準備工程では計算機による製造支援システムとしてＣＡＭシステムが存在するが、情報システムとして見た場合には充分とは言えない。

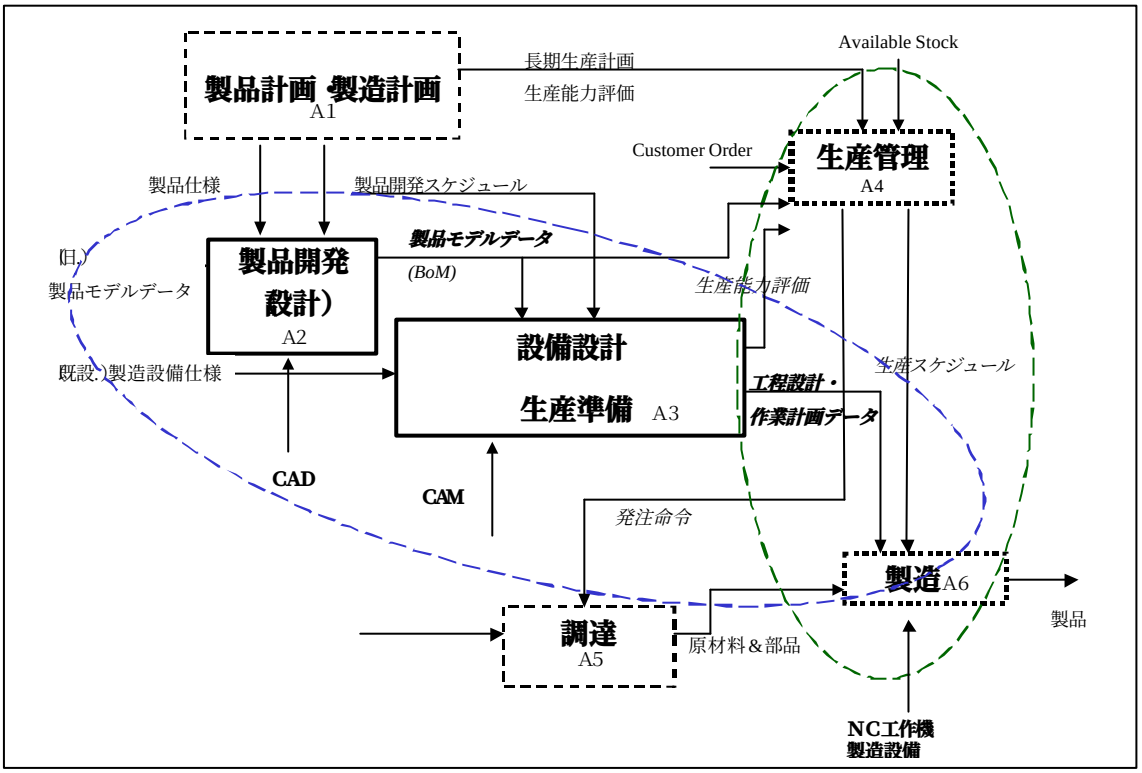


図 2-1 機械系生産プロセスシステム

そもそも「生産設計」とは生産システム便覧によると以下のように定義されている。

“ 設計工程で得られた「製品情報」をもとに「いかに造るか」を決める生産設計を行なう。ここでは大きく分けて、主要な生産工程列を決める工程設計と各工程での人や機械の作業を具体的に決定する作業設計とからなっている。工程設計は製品に対する知識のみでなく、生産工程に関する深い実際的な知識が必要であり、熟練技術を要する難しい仕事であるといわれている。また従来は製品設計と生産設計とは直列的に行なわれていたが、生産に対する要求が厳しくなるに連れて、両者をより融合させて実行しようとする動きが強くなってきている。 ”

生産設計は技術的な難しさと市場にタイミング良く安く登場させるために製品の設計生産効率を上げる事が要求されていて機械系産業の情報化による解決の取り組みが最も注目されている分野である。

2.3　生産設計に対する期待

生産設計に関係する技術者の視点で生産設計分野における現状の課題と情報技術への期待をまとめる。

2.3.1　ユーザの見方

(1) 生産設計技術者（その１）

① 生産技術部門から見たＣＡＤ／ＣＡＭ統合

A. 機械部品における形状特徴と加工

現在の機械部品加工においては、工作機械と工具の進歩とによって、一昔前の常識的な加工方法が最適ではないことも現れている。マシニングセンターの例では、小径の穴をエンドミルでコンタリング穴開け、タップ加工をねじ切りフライス加工、焼き入れ後の材料を切削加工、等がある。これらの生産技術上の変遷にフレキシブルに対応可能なＣＡＤ／ＣＡＭシステムが必要である。このことは、形状特徴と加工方法の分離が本質的に必要なことを意味している。そして、形状特徴と加工方法の間に、人にとって簡単な指示方法が介在する必要性を示唆している。

B. 生産準備作業の実状と自動化への期待

現在のところ、生産準備作業は生産技術部門のスタッフの技術力によるところが大きい。そのため、各技術スタッフの能力を均一化するために、加工ノウハウを「文書」という形で残してはいるが、更新もままならず、なかなか活用されていないというのが実状である。

今後、ＣＡＤ／ＣＡＭ統合により、従来手作業で行なっていた加工順序設定や工具設定が自動化されれば、技術スタッフの能力によらず、最適な加工プロセスが設定できるようになり、製品品質の安定及び生産準備作業期間減少によるリードタイムの短縮が期待される。

(2) 生産設計技術者（その２）

① はじめに

生産設計分野の工程設計のシステム化と生産準備業務自動化に対する課題と今後の技術開発に対する期待を中心にまとめる。

② 工程設計のシステム化

工程設計のシステム化の観点からのニーズを次に示す。

A. ＣＡＤで定義された製品仕様情報から加工特徴の認識

ＣＡＤで作られた形状情報を変換して加工情報にする情報の標準化が必要である。このような標準が工程設計システムの入力に使用できることが現状の業務改善につながると考える。

a) ＣＡＤ、ＣＡＭの種類や組合せに依存しない工程設計システムの構築

企業内では製品や工場ごとに異なるＣＡＤ、ＣＡＭを使用しているケースが多く、個別に工程設計のシステム化に取り組まざるを得なかった。

システムの入力が標準規格で統一されることにより、工程設計システムの共通基盤の構築といった企業規模の取り組みが可能となるであろう。

b) 工程設計システム構築のインフラ整備

従来は生産設計者がＣＡＤモデルに対して加工部位、加工法、公差情報などを付加してＣＡＭモデルを作成していた。このことが工程設計システムを構築する上でのネックの一つとなっていた。

ＣＡＤモデルの形状特徴から加工特徴を自動認識してＣＡＭモデルを効率的に作成できることから、工程設計システム構築の環境が必要であると考える。

B. 標準化された加工特徴を入力として工程設計及び作業設計を自動化

標準化された加工特徴情報を入力として、ＮＣプログラム作成までの自動化が可能であることを期待する。

a) 工程設計及び作業設計の脱技量化、脱属人化

従来、個人の知識や技術に依存することが多かった工程設計及び作業設計のノウハウのシステム化により、経験や熟練度に依存せずに一定の設計品質を得ることを期待する。

b) 工程設計及び作業設計の品質向上

工程設計及び作業設計には複数の解が存在するが、ノウハウのシステム化に

より工程シミュレーションや加工シミュレーションを容易に繰り返し行なうことができる。このため、多くの解を比較・評価してより最適解に近い設計品質を得ることを期待する。

C. 生産資源情報及び加工技術情報の標準化の必要性

生産準備業務の自動化を図る上で生産資源情報及び加工技術情報の標準化が非常に重要であり、そこでの改善が必要である。

a) 生産資源情報の標準化

現状では設備や工具、治具などの生産資源情報は企業や工場、職場ごとに整備されており、その内容やレベルには大きな差があるものと考えられる。これを少なくとも企業、工場のレベルで標準化に取り組む必要がある。

b) 加工技術情報の標準化

加工条件の選択や工具の選択、作業順序の決定などのノウハウについてはＮＣ加工化の進んだ職場ではデータの蓄積がかなり進んでいるものと考えられる。このデータの蓄積が個人に依存せずに進められるような仕組みを作り、標準化の環境を整備する必要がある。

③ まとめ

生産設計のアプリケーションで形状情報から加工特徴を自動作成することが重要である。またこのプロセス間のインターフェースにはシステムに依存しない標準的な情報プロトコルが必要であろう。また体系的な加工特徴の情報化、加工特徴の網羅や、より実部品の工程に近いものへと加工対象モデルを作り上げて行くべきと考える。

また、生産技術情報と加工技術情報の標準化は避けて通れない問題であり、国内外の標準化活動やデファクトスタンダードとなり得るパッケージについて、その動向をウォッチすべきと考える。

(3) 生産設計技術者（その３）

① 課題と期待

自動車業界では、Ｑ（品質向上）、Ｃ（原価削減）、Ｄ（リードタイム短縮）という目的を実現するための重要な手段としてデジタルプロセス化が、急速に進んでいる。これは、ソリッドモデリングがソフトウェア的、ハードウェア的にも充分実用に耐えうることになったこと、欧米の動き、Ｖ－ＣＡＬＳなどの外部的な要因、

さらにバブル経済の苦しい時期を脱して投資意欲が回復してきたことから、この数年、デジタルプロセス化の大波が業界をおそっている。この流れは評価すべきであるが、製品開発の視点からのアプローチであることは否めない。すなわち、最終的には、設計から製造までのものづくりに直結して始めてトータルな成果が得られるというものであるが、今はまだ製造への連携が希薄なことが今後大きな問題となると思われる。

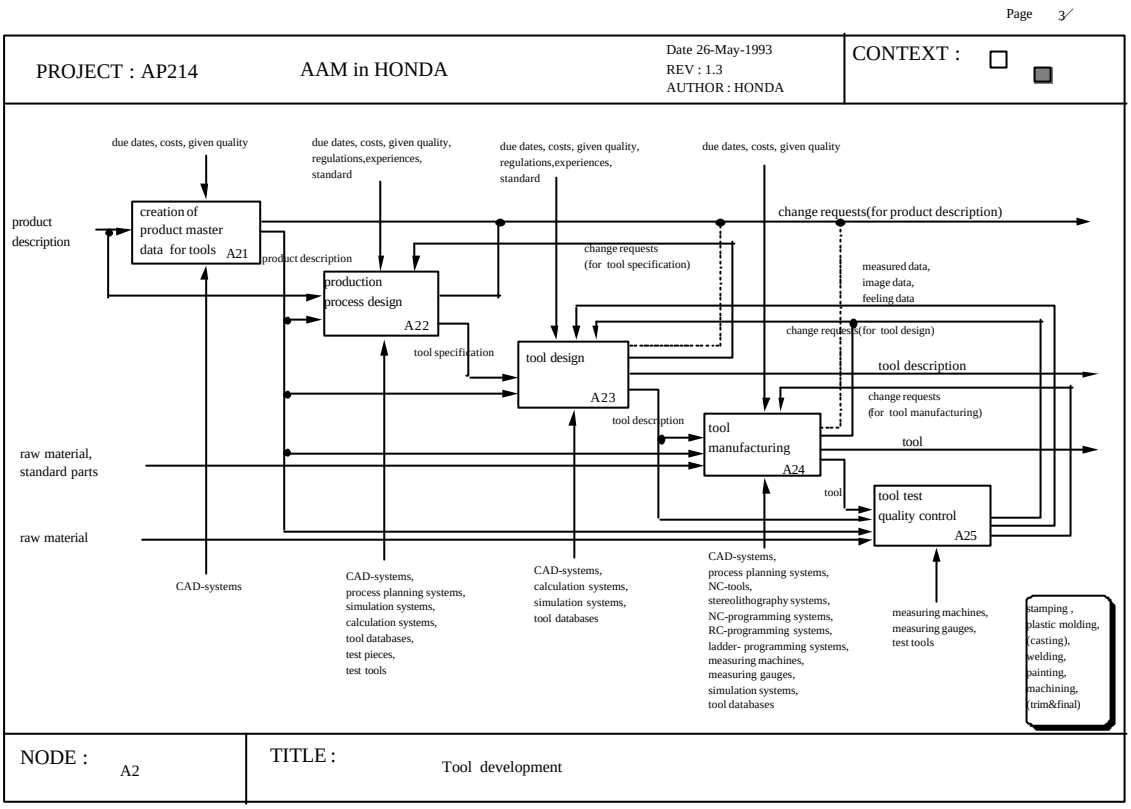


図 2-2 自動車業界におけるトップレベルのAAM

製造との関連での主な課題は、次の通りである。

（a） 設計の段階への生産技術上の検討結果の反映（図 2-2のＡ２３、Ａ２４からＡ２２へのデータの流れ）

（b） 設計意図を含めた製品データの受け取り（図 2-2のＡ２２から後工程へのデータの流れ）

（c） 製品データを受け取った後のＮＣ

（d） 設計意図を含めた製品データの受け取り

（e） 製品データを受け取った後のＮＣ計算（図 2-2のＡ２４の自動化）

（f） ＮＣデータの外作メーカーへのデータ支給

（g） ＮＣマシーン、カッターなどの生産資源データの共有化

（h） 工程設計、作業設計、ＮＣ計算における生産技術情報の蓄積、共有化

以上のカテゴリーのうち優先度の高いものとして（ｃ）、（ｅ）、（ｆ）を解決して、製造との連携の課題について、解決方法の具現化を行なう事は、真の意味のデジタルプロセスを実現する上での重要なことであると考える。

特に、（ｃ）、（ｅ）、（ｆ）を深く分析して掘り下げて、実務データを使った実証を進め、可能性から、実用化をすることが期待される。

さらに、範囲を（ａ）、（ｂ）、（ｄ）まで拡大し、トータルなデジタルプロセスの実現に向けた活動を期待する。

2.3.2 生産設計技術支援システムソリューションベンダーの見方

(1) ベンダー（その 1）

① メリット

A. ２１世紀の製造業の企業イメージとして、

「小グループ」に別れプロジェクト単位にグループが「合目的」に向かって活動し応分負担により成果を分配するようになれば、ファブレス指向により組織分化がすすめば、設計と製造のデータインターフェースとして、必須技術になる。

B. 機械加工分野における標準化の推進は、有効である。

② デメリット

機械加工分野における工程設計、作業設計の効率化を考えた場合、

（a） 加工特徴の自動認識

（b） 生産資源情報

（c） 加工技術情報

（a）の認識率や（b）（c）の情報の持ち方によって生産設計の自動化が左右される。

ベンダーの立場からいえば、現行ＣＡＤ／ＣＡＭの機能拡張のプライオリティーと

して、

- ＳＴＥＰ（ＡＰ２０３）のようなデータ入力インターフェース標準化
- ＳＴＥＰ（ＡＰ２０３）のようなデータ出力インターフェース標準化
- 自動車向けの製品モデルＳＴＥＰ（ＡＰ２１４）への対応
- ユーザーニーズの多いアプリケーションプロトコル対応

の順に対応する方向であろう。

(2)　ベンダー（その２）

ＣＡＤ／ＣＡＭベンダーは、製品設計から生産準備、製造のための工程設計、作業設計、ＮＣデータ作成等、製品製造に関する問題点の解決方法としてＣＡＤ／ＣＡＭシステムをベースとしてソリューションの提供を行なう方向で日々開発を進めている。

さらに、マーケットも自国のみならず、ワールドワイドな展開を考慮してシステム開発を行なっていると言っても過言でない。このような事より、システムの守備範囲は明らかに広くなり、また、ユーザーのニーズも多様化してきて、従来の様な機能的に広い範囲をカバーしているだけのシステムでは満足させる事ができなくなってきている。現在はシステムが広い範囲をカバーし、更に、個々の機能の充実も大きな課題になってきている以上、ＣＡＤ／ＣＡＭベンダーとしては、自社製品だけではユーザーへベストソリューションを提供する事が難しい状況になっているのも事実である。

従って、ＣＡＤ／ＣＡＭベンダーは自社製品の弱点をカバーし、競争力をアップさせる事を目的に、他社製品をサブシステム、もしくは、部品として利用する動きが今後ますます進んでいくと思われる。このような背景にあって、アプリケーション間でもインターフェースとして、ＳＴＥＰのような規格は非常に重要な位置を占めると考える。

ＣＡＤシステムで作成された標準的製品情報から標準化された加工情報へ変換し、更にＮＣデータまで作成するといった一連の流れがシステム化できるという事は重要であり、半自動でもＮＣデータまで作り上げるという一連の流れの技術開発は重要だと考える。

その反面、次のような疑問も出てくる。

製品仕様情報を記述するＡＰ２０３と加工特徴や精度情報などを表わすＡＰ２２４は、ＣＡＤとＣＡＭというような形で明確に分離されるものではないと思われる。

現状のＣＡＤシステムでは、形状情報だけを持つとは限らない。ハイエンドやミッ

ドレンジのソリッドモデラーでは、加工特徴までをフィーチャーとして記述できるのが一般的になっている事から考えても、かなりの情報を上流工程の段階で持つ事になると考える。

このことより、どの範囲までの情報を製品仕様情報（ＡＰ２０３）に持たせるかは、各企業の設計手法や設計側の役割によって、異なるのではないかと思われる。

このような事から推測すると、実際に設計部門と生産準備部門間で運用する場合には、概念的にはＡＰ２２４の仕様に基づいたＡＰ２２４ライクなフォーマットを利用する事が予想される。

いずれにせよ、ＣＡＤ／ＣＡＭベンダーとしては、今後、他社システムとの協調的な路線がトレンドになっていく限り、ＡＰ２２４などをインターフェースとして利用できるという価値は大きく、また、これを共通語として積極的に利用することは間違いないであろう。

2.3.3　研究者の見方

ＩＳＯ１０３０３（ＳＴＥＰ）は製造業における産業技術情報インフラとして規格開発中の産業データ交換の国際規格である。ＳＴＥＰが対象とする産業データ(Industrial Data)は種々の産業分野に及び、データ交換の範囲を飛び越え、各産業分野のプロセスと情報の表現と意味の共通理解にまで広がった。しかし、狭義のＣＡＤデータ交換、ＣＡＤ／ＣＡＭ統合に関する技術開発は終了したわけではなく、依然として未解決の問題を抱えているのが現状である。特に 3 次元ＣＡＤの普及と共に、製品モデル交換に関するＳＴＥＰの有効性に関しては今日的課題であり、問題点の認識とその解決方向については多くの議論がなされている。一方、ＣＡＤ／ＣＡＭ統合化技術に関してはＳＴＥＰ規格においてはＡＰ２２４の提案がなされているが、その実証に関する検証実験例は少ない。

①どの様に提案ＡＰを利用するのか

②提案ＡＰはどの範囲の加工技術を支援するのか

③ＣＡＤシステムとの接続をどの程度可能とするのか

④製品モデルの加工の観点からの解釈の自動化をどの程度可能とするのか

⑤ＣＡＤ／ＣＡＭ統合を実現する上でＡＰ２０３／２２４のみで可能なのか

⑥その信頼性はどの程度あるのか

などの課題を検証する上で実施しなければならない検討課題が山積している。現状、STEP AP203 から AP224 の加工フィーチャを抽出する技術検討を通して、ＣＡＤ／ＣＡＭ統合システムを実現するための基本的方向とＳＴＥＰ／ＡＰ２２４が果たす役割を確認しているが、更に研究開発を進める課題として、

- 生産資源モデルの研究開発
- 加工技術モデルの研究開発

が挙げられる。また、素材料形状を考慮した加工プロセス設計を支援するためには、

- 加工中間形状の表現と加工フィーチャー干渉表現に関する研究開発

が必要となる。これらの研究開発課題は大学等における基礎研究の観点からみても興味ある研究課題であり、今後の研究成果が期待される分野でもある。

2.3.4　まとめ

以上の機械系生産設計に関係する技術者の視点から以下のことが重要な課題であると思われる。

- 製品情報と加工情報の関連付けと自動抽出技術
- 製品情報と加工情報間の標準化
- 加工技術の体系化による加工技術データベースの蓄積

2.4　デジタルマニュファクチャリング

2.4.1　デジタルマニュファクチャリングとは

現在の生産システムは設計分野ではＣＡＤ、生産準備分野ではＣＡＭ、加工分野へのＮＣ工作機、生産管理システムとしてＥＲＰ・ＭＲＰ等のシステムが導入され生産システムの自動化が実現されている。ただし現在のシステムはＣＡＤ／ＰＤＭの様な実装システム毎にその情報が閉じているために設計工程で作成した製品設計情報を直接そのままの状態で工作機械の加工法、加工順番を決める生産設計工程の入力情報とする事ができない。

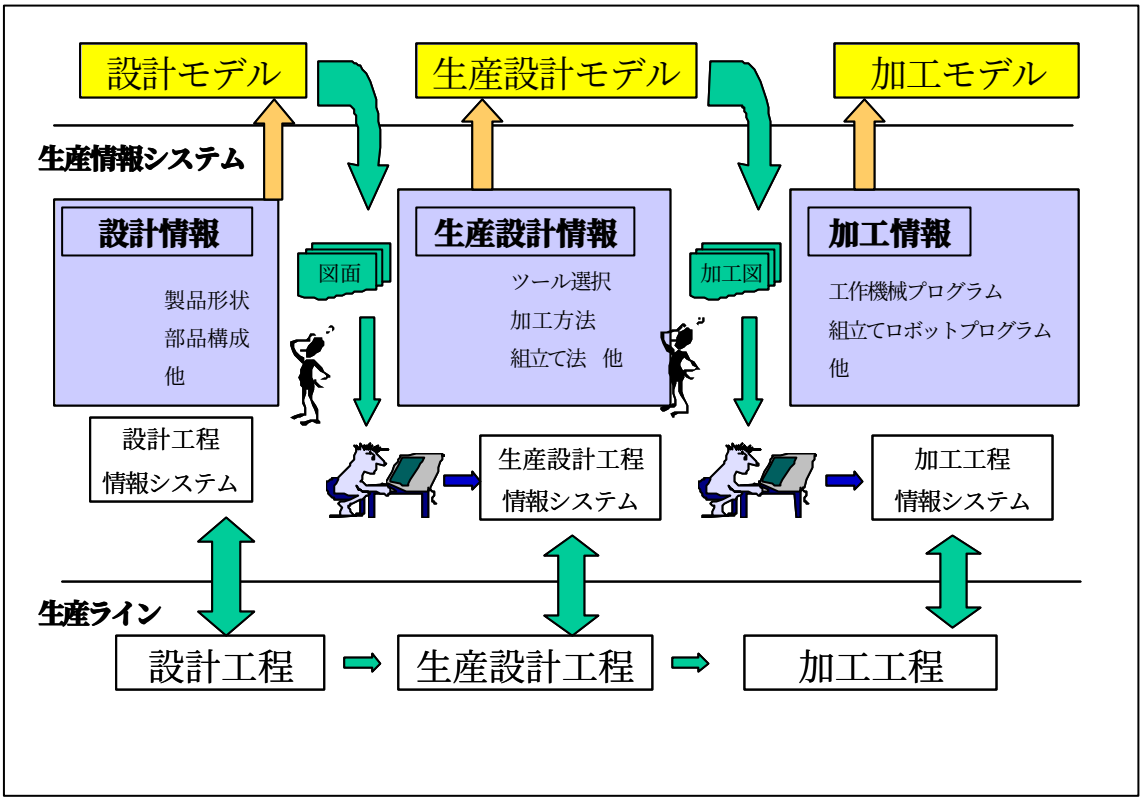


図 2-3 現状の生産システムの情報の流れ

すなわち、現在の生産情報システムは設計から製造までの生産システムの中を情報がシームレスに流れる技術が充分確立していないため、異なったシステム間の情報の受け渡しを行なう際に人間による解釈、再入力作業が発生する。（図 2-3）従って、上流から下流まで工程毎に生産情報が閉じていて情報の流れの非効率性と全生産情報を統一的視点から把握したり最適化する事が難しい。すなわち、工程毎に孤立した情報システムである。（図 2-3）

デジタルマニュファクチャリングは生産プロセスの上流から下流までの全プロセスの中を「ものづくり」のための情報を途切れる事なく連続的に流し、その情報により製品作りを行なおうというものである。このようなデジタルマニュファクチャリングの情報システムを実現するためには上流から下流までの全生産プロセスで統一的に扱える製品モデル表現が必要となる。（図 2-4）

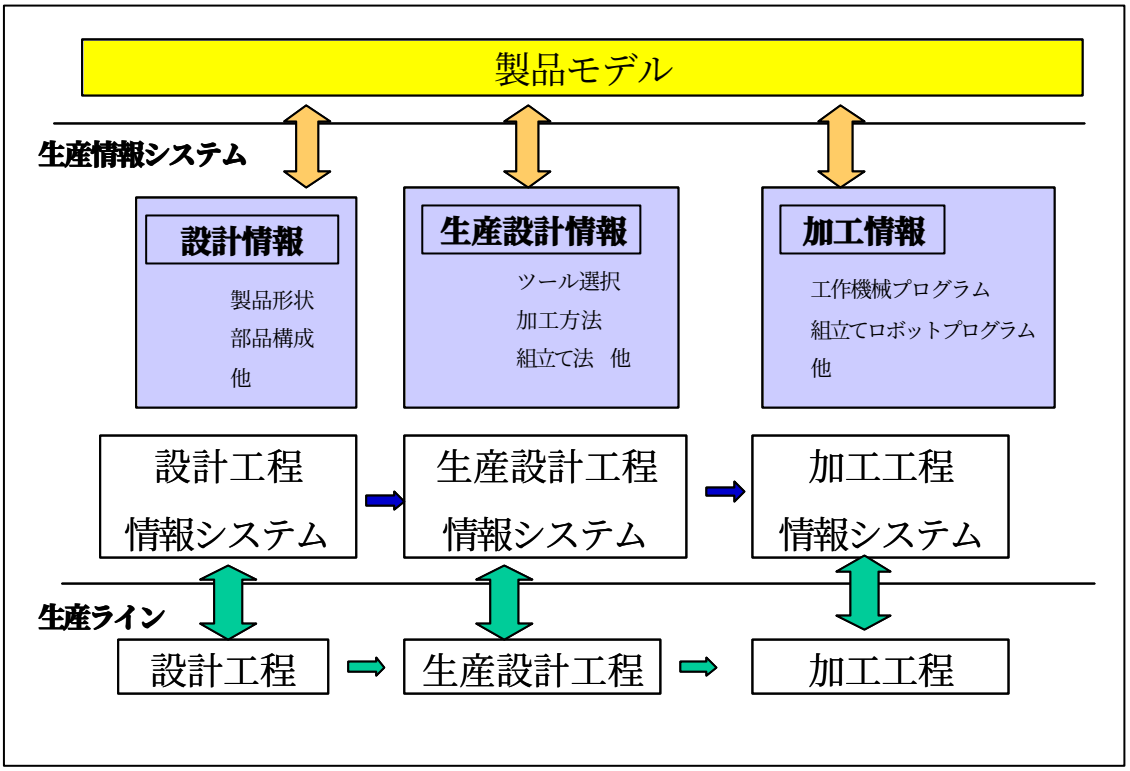


図 2-4　シームレスな生産情報システム

すなわち、設計工程で作成した製品情報を利用し、生産準備工程で生産設計を行ない生産加工情報にできる。そのようなシームレスな情報の流れを実現でき、かつ全生産プロセスで扱える統一的製品モデル表現が必要である。その製品モデルに「ものづくり」の情報をのせて全生産プロセスに情報を流す事でデジタルマニュファクチャリングは実現できる。

このような製品モデル表現は当然実装システムに依存しないデータ構造が必要である。

デジタルマニュファクチャリングを実現するための「製品モデル表現」の情報要件を以下にまとめる。

- 設計から製造までの全生産プロセスで統一的に扱える
- 設計から製造までの全生産プロセスで実装システムに独立なデータモデル

2.4.2　製品モデル情報を活用したデジタルマニュファクチャリング

前節で述べたような製品情報モデルとしてＩＳＯ／ＴＣ１８４／ＳＣ４で開発が進められているＳＴＥＰがある。ＳＴＥＰは当初、異なったＣＡＤシステム間のデータ交換をするための中間ファイル形式として開発に着手された。ところが開発が進むにつれて単なる

ＣＡＤデータのような幾何形状のデータをモデル化するだけでなく製品を作り、運転、保守、廃棄するような製品の全ライフサイクルの製品情報を統一的に表現できるモデルとして見直されその開発が進められてきた。すなわち製品を「設計」、「製造」、「運転」、「廃棄」の各プロセスで活用できる製品モデルである。

このような製品の全ライフサイクルを統一的に扱える製品モデル表現を使えば図 2-4のように各工程毎の情報をそのモデルに入れる事ができて関係する全ての工程の情報を統合して扱う事が可能となる。すなわち上流から下流まで表現可能な単一の製品モデル表現がある場合には、製品モデルデータが生産プロセス情報システムの上流から下流まで流れる中で各工程に必要な情報を入出力しながら「ものづくり」のための情報を流し、それにより生産を行なうことができる。このような製品モデルにより全生産情報を共有する事で統一的な生産情報管理が可能となる。

また、ＳＴＥＰの表現する情報モデルはＥＸＰＲＥＳＳ言語で表現されていて、これによりＳＴＥＰの表現する製品モデル表現は単なるデータ構造だけでなく関係、制約、評価式等を定義する事ができてデータ要素間の関係を表現する事ができる。（図 2-5）このＳＴＥＰの持つ特徴は製品モデルの中に製品仕様を容易に表現する事を意味して他モデル表現手法より優れている面でもある。

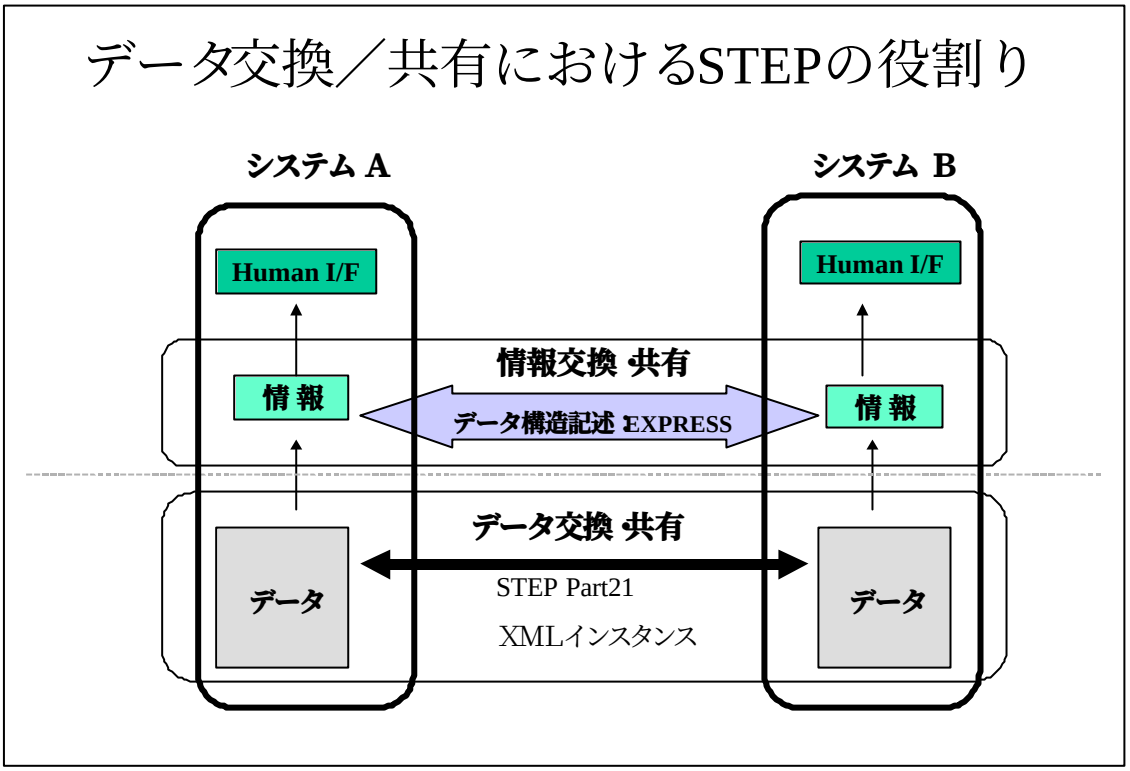


図 2-5 情報交換／共有におけるＳＴＥＰの役割

2.5 ＳＴＥＰを利用した機械系製品の設計から加工工程までの情報モデル

関連するＳＴＥＰの製品モデルは 200 番台のＰＡＲＴに定義されていて、ＡＰ（アプリケーションプロトコル）と呼ばれている。

今回の機械系生産システムを扱う製品情報モデルのＡＰは以下のものがある。

- 設計プロセスでの製品モデル：ＡＰ２０３
- 工程設計プロセスでの製品モデル：ＡＰ２１３　他
- ＮＣ加工プロセスのための製品モデル：ＡＰ２２４
- 加工モデル：ＩＳＯ１４６４９

対象となるＡＰが表現する情報は製品／部品で定義されるデータをキーとして各工程で必要な情報を入出力しながら生産プロセス情報を共有する。

まず機械系生産プロセスで関係する製品モデル表現の概要と特徴をまとめる。

① ＡＰ２０３：Configuration controlled design

設計工程ではこの製品モデル表現を使用して製品設計を実施する事ができる。

ＡＰ２０３は機械部品、組立品を対象とした製品設計のための製品モデル表現であり、

このデータモデルを利用して形態管理された製品の設計結果を表現できる。

このデータモデルは大きく分けて設計作業と、その結果作成された設計モデルを管理する情報と形状、設計仕様などの製品の設計情報が表現できる。

設計管理情報としてその製品設計モデルの設計作業や修正作業情報記録のための設計活動管理（DESIGN_ACTIVITY_CONTROL）、設計結果の承認作業がどの組織のだれによって行なわれたのかを記録する公認（AUTHORIZATION）、設計された部品の採用情報を表現するエフェクティビティ（EFFECTIVITY）が表現できる。

部品（製品）の設計情報は形状、設計仕様が表現できる。形状表現は 3 次元ソリッドモデル表現である Advanced_boundary_representation 他５種類の形状表現で表現可能である。

製品の設計情報としては部品構成情報（BILL_OF_MATERIAL）、材料仕様や加工工程仕様を表現するための設計情報（DESIGN_INFORMATION）、最終的製品の情報表現のための（END_ITEM_IDENTIFICATION）、部品のサプライヤ情報のための供給元管理（SOURCE_CONTROL）がある。

設計工程では設計作業が開始されると製作しようとする製品のコンセプトをもとに主に形状のデザインを行ない形状に加工方法、材料、最終仕上げ状態や部品構成等の設計者がイメージする最終製品仕様を付加していく。また、製品の使用開始や契約先等の製品の管理情報もこの設計工程での製品設計情報である。このような設計がどのような組織の誰により行なわれ、誰にいつ承認されたのかという設計管理情報も重要な情報として設計工程での製品モデルに含まれる。

以下にＡＰ２０３の中で定義されている Uof（情報単位）の概要を示す。

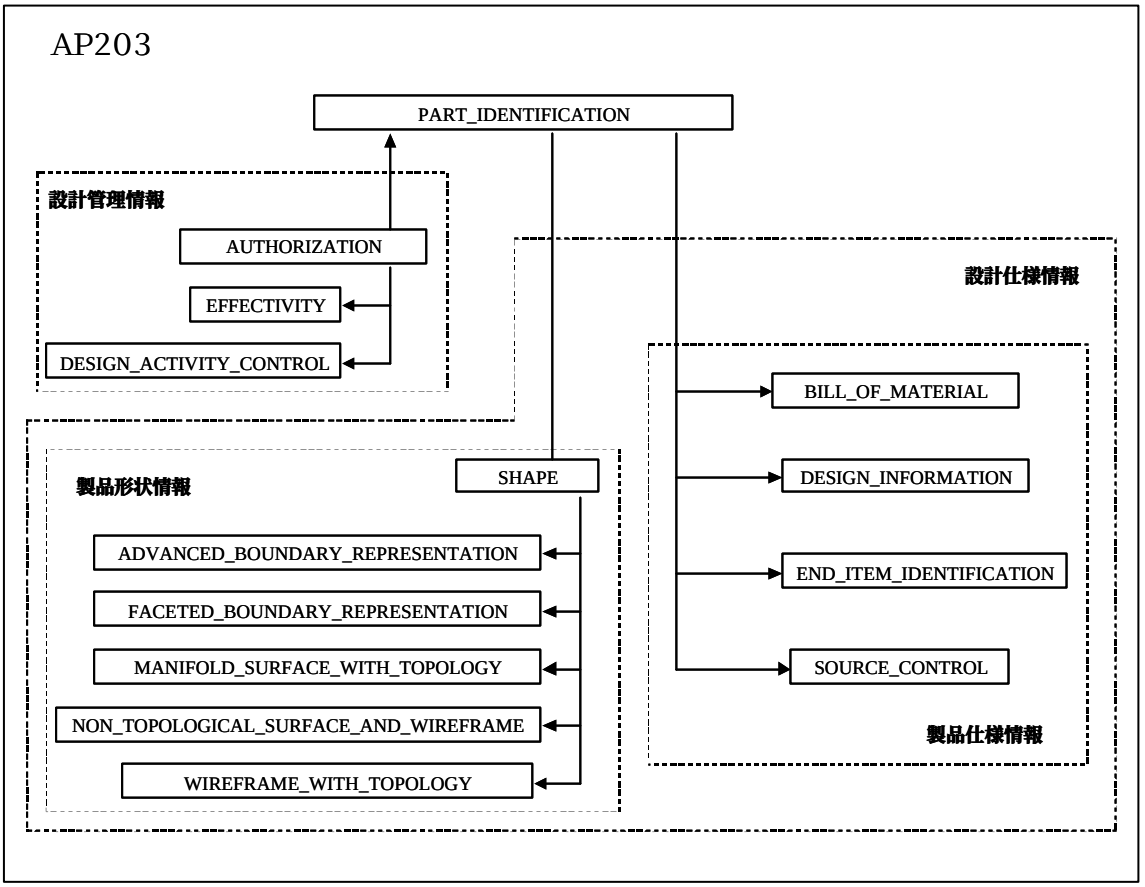


図 2-6 AP203の情報モデル

- Authorization（公認）

  Authorization（公認）は承認部門がある特定の日付、それぞれの承認レベルで製品を承認するためのデータ構造を含んでいる。

  Approval（承認）,Person_organization

- bill_of_material（部品構成表）

  Bill_of_material は製品構成を表わすための階層構造を持った製品または材料のリストを含んでいる。

  Alternate_part（代替部品）,Component_assembly_position,
  Engineering_assembly,Engineering_make_from,
  Engineering_next_higher_assembly,Engineering_promissory_usage,
  Substitute_part

- design_activity_control（設計活動管理）

  Design_activity_control は部品バージョンの履歴を含んでいる。この履歴は初度の設計形態（設計要求事項）と改訂後の設計形態を表わしている。この Design_activity_control は初度部品または改訂作業のための作業指示のデータとなる。

  Change_order,Change_request,Start_order,Start_request,Work_order, Work_request

- design_information（設計情報）

  Design_information は部品識別、形状以外の最終製品の品質要求事項の情報を含む。この情報は設計と製造に関する規格（ Specification）を表わしている。

  Additional_design_information,Design_specification（設計仕様）, Material_specification（素材仕様）,Process_specification（工程仕様）, Specification（仕様）,Surface_finish_specification（仕上げ仕様）, Usage_constraint（使用制約）

- effectivity（エフェクティビティ）

  Effectivity は部品の適用計画に関する情報を含んでいる。

  Planned_date_effectivity,Planned_effectivity,Planned_lot_effectivity, Planned_sequence_effectivity

- end_item_identification（最終品目識別）

  End_item_identification は顧客に引き渡された製品の情報を含んでいる。

  End_item_identification は製品の形態管理の情報を含んでいる。

  Product_configuration, Product_model（製品モデル）

- part_identification（部品識別）

  Part_identification は部品の改訂番号、それぞれの改訂に対する設計要求情報を含んでいる。

  Design_discipline_product_definition, Part（部品）, Part_version

- source_control（供給元管理）

  Source_control は特定の部品を製造するために認定された組織に関する情報を含んでいる。

  Supplier（供給者）, Supplied_part_version

- shape（形状）

  shape は部品形状の幾何表現の定義をする。この Uof では部品の部分形状を異なった幾何表現で示すことができる。

  Geometric_model_representation, Shape,Shape_aspect

- advanced_boundary_representation

  曲面を用いたソリッド表現

- facetted_surface_with_topology

  平面近似ソリッド表現

- manifold_surface_with_topology

  トポロジーを含むマニホールドサーフェース表現

- non_topological_surface_and_wireframe

  トポロジーを含まないワイヤーフレームまたはサーフェース表現

- wireframe_with_topology

  トポロジーを含むワイヤーフレーム表現

② ＡＰ２１３: Numerical control process plans for machined parts.

ＡＰ２１３はＮＣ装置を使用して部品を加工するために工程設計をするための情報モデル表現である。ＡＰ２１３はＮＣ装置を使用して部品を加工するための必要な資材、作業順及び部品形状と作業の関係を記述する命令に関係する情報を表現する。

すなわち部品を加工するためのＮＣプログラミングを作成するために必要な情報、検査情報、ショップフロアー情報等の工程設計のために必要な仕様を表現するための情報である。

生産準備工程では設計工程で作られた部品の設計情報を基にして、加工作業のための材料と加工のための工具やＮＣ工作機械を使い、部品を作成するための加工順番、ＮＣプログラムが作られそれをもとに製品が加工作成される。このような一連のＮＣ工作機械を使い部品を製作するための情報がこのモデルの中の情報として蓄えられる。

ＡＰ２１３は以下の Uof（情報単位）から構成されている。

- Administrative（工程設計管理情報）

  工程設計管理情報とはＮＣ装置の工程設計のために必要な管理情報である。

例えば、部品加工のための工程設計リリース、リビジョン管理がこの情報の中に入る。

Allowed_time,Auxiliary_header_information,Company,Contract,
NC_process_plan_version,Organaization,Part_version,Performance_rate,
Planning_group_member,Process_palan_security,Production_rate,Revision,
Status_authority

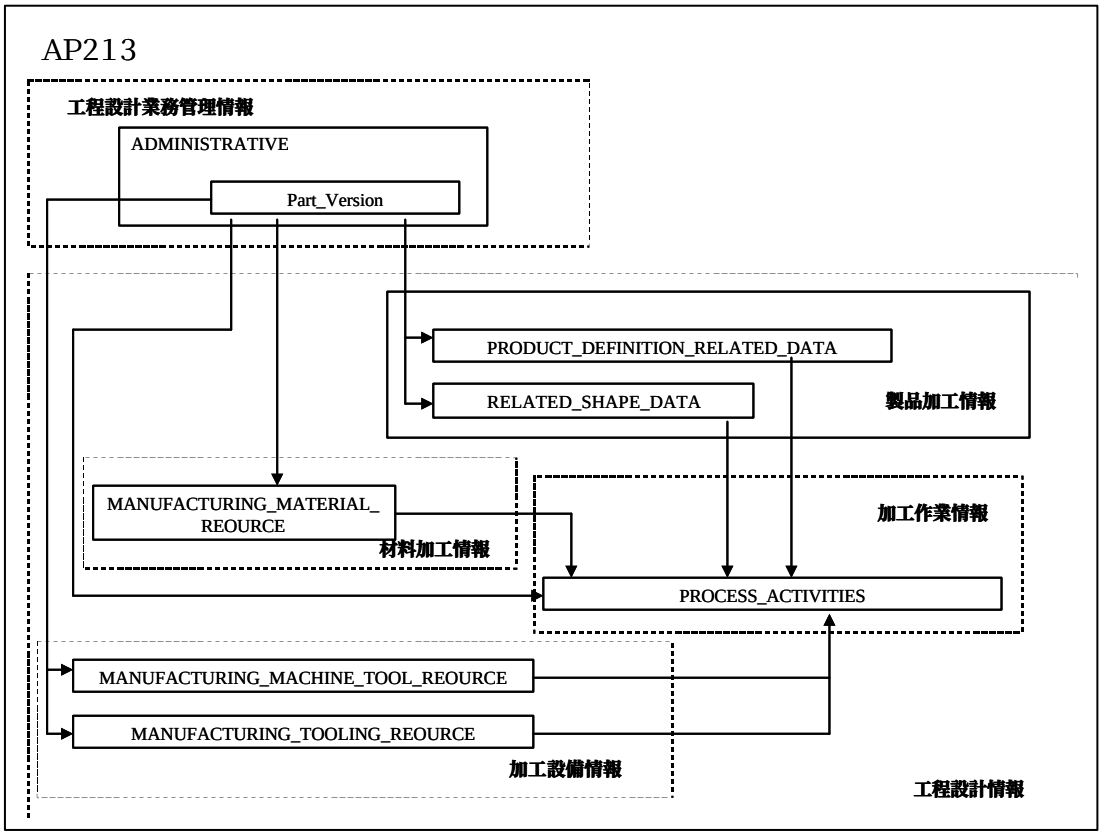


図 2-7 AP213の情報モデル

- manufacturing_machine_tool_resources（加工設備機器資源情報）

この情報は部品、製品の加工のために必要な設備と能力を表わす。たとえばＮＣ装置本体とタレットや回転テーブルなどの付帯機器の情報も含まれる。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Generic_manufacturing_resourec,In_facility_location,Machine,

Machine_parameter,Machine_usage,Numeric_parameter,
Special_capability,String_parameter,Tool_magazine_turret_carousel,
Work_cell, Workstation

- manufacturing_material_resources（**加工材料情報**）

  この情報には部品加工のために必要な材料情報が含まれる。

  この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Material, Material_parameter, Material_specification, Numeric_parameter,
Raw_material_requirement, String_parameter

- manufacturing_tooling_resources（**加工工具情報**）

  この情報には部品加工のための工具、備品に関する情報が含まれる。

  この情報に含まれるアプリケーションオブジェクトには以下のものがある。

Fixture_assembly,Fixture_assembly_element,Fixtuer_contract,
NC_program,Numeric_parameter,String_parameter,Tool_assembly,
Tool_assembly_element,Tool_compensation,Tool_contract,Tool_date,
Tool_parameter

- process_activities（**加工作業情報**）

  この情報には加工作業仕様に関する情報が含まれる。加工作業仕様とは材料を最終部品に加工するための一連の加工作業、検査作業などである。この情報にはＮＣ装置を使い材料を切削する工程、操作、作業が含まれる。

  この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Activity,Activity_group,Alternate_activity,Ancriiary_action, Machine_setup,
Material_removal,Numeric_parameter,Part_fixtuer_mounting,
Part_handling,Part_loading,Part_machine_mounting,Process_parameter,
Special_instruction,String_parameter,Supplement_document,Validation

- product_definition_related_data（**加工製品仕様情報**）

  この情報には部品を製作するために必要なトレランスや図面、イラスト、ＣＡＤによるモデル等の製品設計情報が含まれる。

  この情報に含まれるアプリケーションオブジェクトには以下のものがある。

Drawing, Illustratin, NC_tolerence, Surface_finishi, View_reference

- related_shape_data（加工形状情報）

  この情報には形状に関わる加工作業のための情報が含まれる。部品設計のための形状表現が代表的なものである。

  この情報に含まれるアプリケーションオブジェクトは以下のものがある。

  Advanced_B_rep,FAceted_B_rep,Geometric_model_representation,
  Manifold_surface_with_topology,NC_object_elemet_shape_aspect,
  NC_part_object_element,Non_topological_surface_and_wireframe,
  Wireframe_with_topology

③ ＡＰ２２４：Mechanical Product definition for process planning using machining features

この情報モデルは機械部品を加工するために必要な情報を表現するためのものである。

この情報モデルでは一つの部品、またはアンブリー製品の一つの要素部品を機械加工するためのものである。対象とする機械加工はミーリングまたはターニング装置により行なわれるものとする。この情報モデルは機械加工のために必要な加工特徴をさだめる。　加工特徴は 3 次元のソリッドモデルから導かれる。加工特徴は工作機械を使い材料から形状を作るために必要な情報要素である。この情報モデルは部品製作のための管理情報も表現される。

ミーリングマシンやターニングマシンを使い機械部品を加工する場合には、加工する工程の情報が必要である。この工程における装置や工具の情報と加工材料や製品仕様を参考にして工作機械で材料から削りだすが、設計形状から工作機械を使い形状を削り出すための加工特徴をこの情報モデルは表現する。

以下にこの情報モデルの Uof（情報単位）の概要をまとめる。

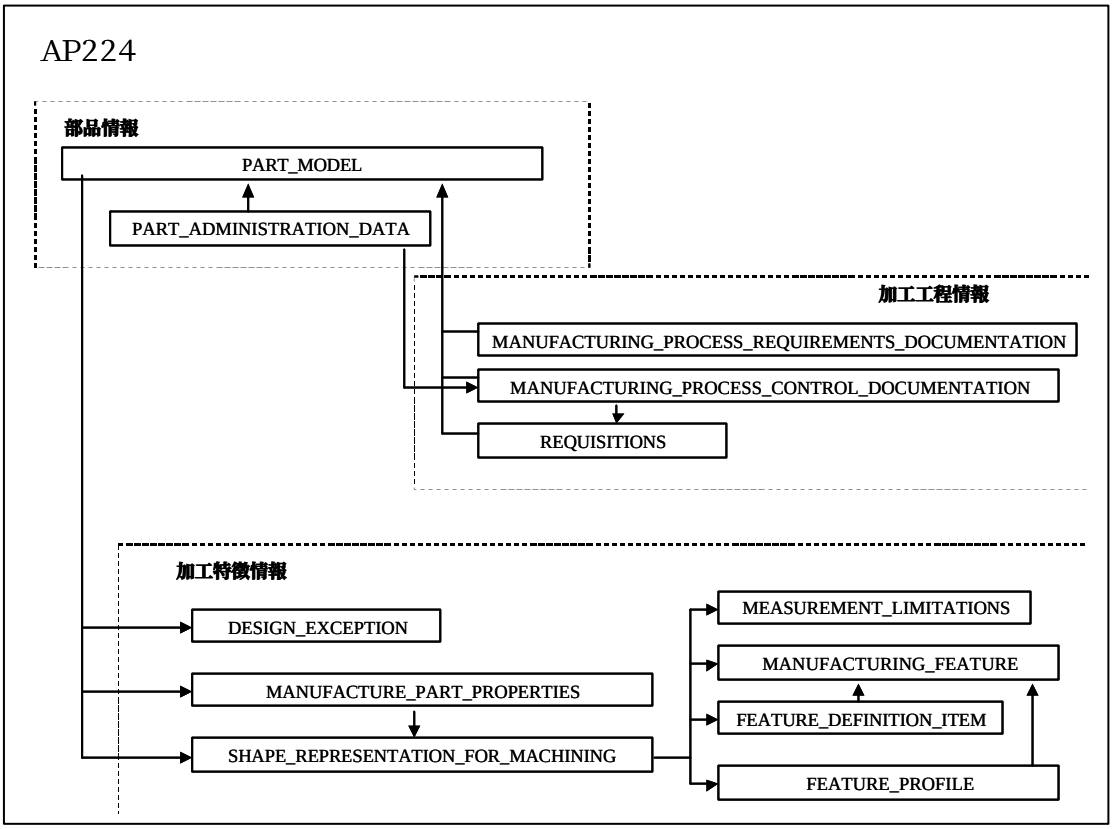


図 2-8 AP224の情報モデル

- Design_exception

  この情報には工程設計段階で発見された問題点、エラーを解決するための情報が含まれている。

  この情報に含まれるアプリケーションオブジェクトは以下のものがある。

  Design_exception_notice,Engineering_change_order,Engineering_change_proposal

- Feature_definition_item

  この情報には加工特徴を定義するために必要な情報が含まれている。この情報は幾何形状と加工特徴の関係を定義する。この情報には形状に関わる加工作業のための情報が含まれる。部品設計のための形状表現が代表的なものである。

  この情報に含まれるアプリケーションオブジェクトは以下のものがある。

  Angle_tapaer,Blind_bottom_condition,Boss_top_condition,Chamfer_angle,Circular_path,Complete_circular_path,Conical_hole_bottom,

Diameter_taper,Directed_taper,First_offset,Flat_hole_bottom,
Flat_slot_end_type,Flat_with_radius_hole_bottom,
Flat_with_taper_hole_bottom,General_path,
General_pocket_bottom_condition,General_profile_floor,
General_rib_top_floor,General_top_condition,Linear_path,
Open_slot_end_type,Partial_area_definition,Partial_circular_path,Path,
Planar_pocket_bottom_condition, Planar_profile_floor, Planar_rib_top_floor,
Planar_top_condition,Pocket_bottom_condition, Profile_floor,
Radiused_slot_end_type,Rib_top_floor,Second_chamfer_offset,Second_offset,
Slot_end_type,Spherical_hole_bottom,Throug_bottom_condition,
Through_pocket_bottom_condition,Through_profile_floor,
Woodruff_slot_end_type

- Feature_profile

この情報には 2 次元形状を定めるに必要な情報が含まれている。この 2 次元形状を掃引することで 3 次元形状は作られる。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Circular_closed_profile,Closed_profile,General_closed_profile,
General_open_profile,Linear_profile,Ngon_profile,Open_profule,
Partial_circular_profile,Profile,Rectangular_closed_profile,
Rounded_U_profile,Square_U_profile,Tee_profile,Vee_profile

- Manufacturing_feature

この情報は部品を加工する際に削り出される材料の形を示す。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Boss,Catalogue_knurl,Catalogue_marking,Catalogue_thread,Chamfer,
Circular_boss,Circular_closed_shape_profile,Circular_cutout,
Circular_offset_pattern,Circular_omit_pattern,Circular_pattern,
Compound_feature,Compound_feature_element,
Compound_feature_relationship,Constant_radius_edge_round,
Constant_radius_fillet,Counterbore_hole,Countersunk_hole,Cutout,
Defined_marking,Defined_thread,Diagonal_knurl,Diamond_knurl,

Edge_round,Fillet,General_boss,General_cutout,General_outside_profile,
General_pattern,General_pocket,General_removal_volume,
General_revolution,General_shape_profile,Groove,Hole,Knurl,
Machining_feature,Manufacturing_feature,Manufacturing_feature_group,
Marking,Multi_axis_feature,Outer_diameter,Outer_diameter_to_shoulder,
Outer_round,Partial_circular_shape_profile,Planar_face,
Pocket,Profile_feature,Protrusion,Recess,Rectangular_boss,
Rectangular_closed_pocket,Rectangular_closed_shape_profile,Rectangular_
offset_pattern,Rectangular_omit_pattern,Rectangular_open_pocket,
Rectangular_open_shape_profile,Rectangular_pattern,Replicate_base,
Replicate_feature,Revolved_feature,Revolved_flat,Revolved_round,Rib_top,
Round_hole,Rounded_end,Shape_profile,Slot,Spherical_cap,
Step,Straight_knurl,Thread,Transition_feature,Turned_knurl.

- manufacturing_part_properties

この情報は部品の特性を表現する。特性は加工のための設備、工程設計を検討する時に利用される。

この Uof **には以下のアプリケーションオブジェクトが定義されている。**

Alternate_material,Descriptive_parameter,Hardness,Material,
Material_property,Numeric_parameter,
Numeric_parameter_with_tolerance,Part_property,Process_property ,
Property,Property_parameter,Surface_property

- Manufacturing_process_control_documentation

この情報は製品加工のための発注情報を与え、加工作業の開始前、作業中、加工完了後に参照される。この情報を基に加工作業条件、使用設備が決められる。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Customer_order,Digital_technical_data_package_work_order,Ordered_part,
Pedigree_creation_orderProject_order,Resource_acquisition_order,
Shop_work_order.

- Manufacturing_process_requirement_documents

この情報は製品の持つ製品特性、製品の使用制約を与えるものである。この情報は加工工程の任意の段階での、部品の特性、加工状態、部品が加工されているときの状態を与える。この情報には企業、顧客、軍、国家、国際標準に関係するものが含まれる。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Specification,Specification_usage_constraint

- Measurement_limitations

この情報は部品加工を行なう時に重要となる形状の寸法または参照部品と加工部品の形状間の寸法関係を定義する。すなわち、加工する場合の許容誤差、公差情報が含まれる。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Angular_dimension_tolerance,

- Part_administration_data

この情報は製品の管理情報が含まれる。この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Approval,Organization,Person,Person_in_organaization

- Part_model

この情報は工程設計に入力するための部品情報と関連する情報を含む。付加的な品質情報やフィードバックされた関係部品のリビジョン情報等が含まれる。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Manufacturing_assembly,Manufacturing_asembly_relationship,
Mating_definition,Mating_definition_relationship,Mating_relationship,
Part,Single_piece_part

- Requisitions

この情報は注文された作業を完遂するために必要な資源の情報を表わす。この情報は加工作業を実施する組織で作られ活用される情報である。

この情報に含まれるアプリケーションオブジェクトは以下のものがある。

Cutting_tool_requisition,Dedicated_fixtue_requisision,
Indirect_stock_requision,Machine_requisition,Material_requisition,

Modular_fixture_requisition, Requisition

- Shape_representation_for_machining

  この情報では作業開始前または完了後の部品の物理的定義が行なわれている。

  この情報は部品特徴をパラメトリックに表現したり、付加的幾何形状、位相定義が行なわれる。

  Base_shape,Block_base_shape,B-rep_model_element,

  B-rep_shape_aspect_representation,

④ **ＣＮＣ**Data Model:**ＩＳＯ１４６４９** Data model for Computerized Numerical Controllers

A. **ＣＮＣデータモデルとは**

CNC Data Model **は、現在、ＩＳＯ ＴＣ１８４ ＳＣ１／ＷＧ７で開発及び規格化が進められている新しいＮＣプログラミング用言語（ＩＳＯ１４６４９）で使用されるモデルの総称である。**

この開発のねらいは、

- **人間に理解しやすい言語とする。**
- **ＮＣコントローラやＮＣ工作機械に依存しない言語とする。**
- **新しい機能に対応できる言語とする。**
- **ＣＡＤ／ＣＡＭと連動できる言語とする。**

など、従来から使用されているＮＣプログラム言語（ＩＳＯ６９８３：通称Ｇコードと呼ばれている言語）の欠点をカバーすると共に、時代にマッチした言語を開発しようとするものである。

図 2-9**にこの**ＣＮＣ Data Model **の概念図を示すが、ここでは、**

（a） Object Oriented Model **に基づく新しいＮＣプログラム言語［本活動での規格化の対象］**

（b） Predefined Function ――― **従来からの言語（ＩＳＯ６９８３）を人間が理解しやすい命令に置き換えたもの**

（c） **ＩＳＯ６９８３ ――― 従来からのＮＣプログラム言語**

の３種類のいずれも使用できるように配慮されている。（一つのプログラムの中で混合して使用することも許されている。） ただし、上記(ｃ)は、当面の暫定的な措置であり、また、(ｂ)も最終的に(ａ)に移行するために、しばらくの間（10 **年程度）**

はその使用を認めようとするものであり、最終のゴールはあくまで(ａ)を全面的に普及させることにある。

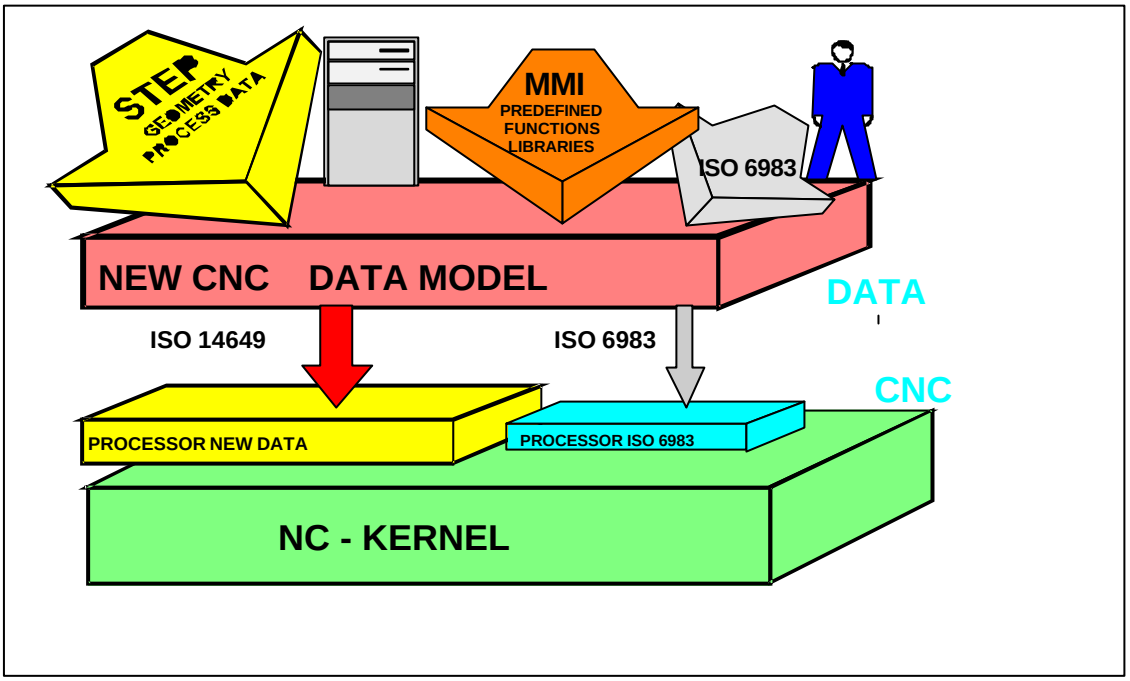


図 2-9 CNCデータモデル概念図

その対象としては、第１ステップとして、2.5 次元及び 3 次元のフライス・穴開け加工を現在実施中であり、第２ステップとして、旋削加工を考えている。

また、この活動の推進は、ヨーロッパ（スイス、ドイツ、イタリア）が中心で行なわれており、それに日本、カナダ、アメリカなどが参加している。なお、日本の窓口としては、日本工作機械工業会が対応している。さらに、技術的な面からは、ヨーロッパのＥＳＰＲＩＴの成果物をベースとしているのが、この活動の特徴と言えよう。

B. オブジェクトオリエンテッドデータモデル

これは、前項でも述べた様に、新しいＮＣプログラム言語用のモデルであり、その構造は、図 2-10、図 2-12に示す様に、

Ａ：Main program、Ｂ：Technology description、Ｃ：Geometry description の３つに大別される。

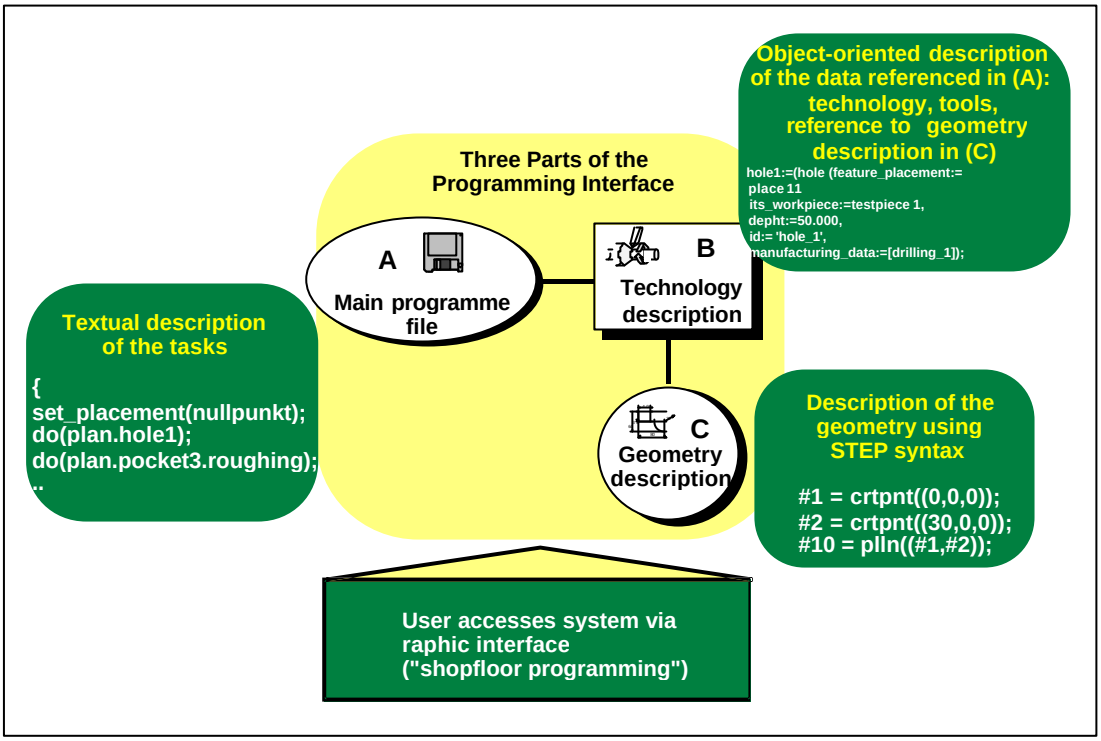


図 2-10　CNCデータモデルの構造

以下では、これらについて、図 2-11に示す例題部品の用のサンプルプログラムも参照しながら、その個別の内容に関して説明を加えることとする。

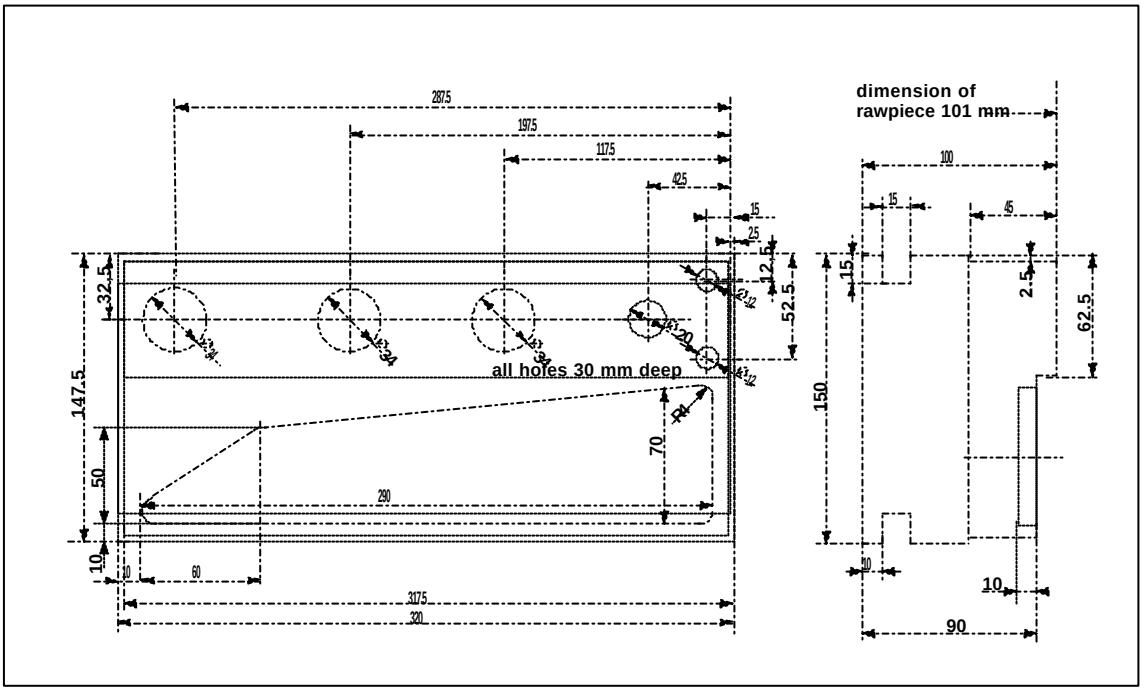


図 2-11 例題部品

a) Main program

ここでは、対象とする部品加工に必要となる“作業”（Workingstep**）の名称のみを、実際の作業順序に従って記述することが、主な役割となる。［ここで記述する“作業”は、後述の** Technology description **の中で定義されているものでなければならない。］**

実際には、加工対象部品の中に存在する各 Manufacturing_feature **毎に、その加工に必要となる作業を洗い出し、それらを部品全体の作業順に並べたものが、** Main program **の中核となる。**

サンプルプログラム「Program File “Examp1.nc”**」の最後の** 2 **行**

```
EXEC PLAN.ROUGH_DRILLING_HOLE1;
EXEC PLAN.DRILLING_HOLE1;
```

がこれに該当する。このプログラムは、例題部品の中の穴（直径 34 **ミリ）一種類しか対象としていないため、作業は、この２つでよいことになる。**

b) Technology description

ここは、加工技術情報の詳細を記述するところであり、Tool Data と Process Data で構成される。

i. Tool Data

サンプルプログラム「Tool Data File “Examp1.too”」では、ＴＯＯＬ３及びＴＯＯＬ５の２本が定義されている。なお、工具仕様は、別途用意されている Tool_sheet のモデルに従って記述されている。

ii. Process Data

ここの記述は、後述する Process Data Model に従って行なわれる。主な内容としては、

- Manufacturing_feature の仕様
- Manufacturing_feature とその加工用作業との関係
- 切削条件（切削速度、送り速度）
- 各作業の詳細情報

が挙げられる。サンプルプログラム「Process Data File “Examp1.dat”」を参照されたい。その最後の部分“ //*** WORKINGSTEPS *** ”で定義さている作業名が、Main Program で記述されている作業名と一致していることを確認いただきたい。

c) Geometry description

これは、ＣＡＤ／ＣＡＭなどで作成された“ ＳＴＥＰ ”形式の幾何情報を、このモデルの中で使用するためのものである。（サンプルプログラムには、この部分は無い。）

勿論、幾何情報の定義は、サンプルプログラムに見られるように、Main Program 及び Technology Descrition の中でも可能である。

以上述べた Object Oriented Data Model による新しいＮＣプログラム言語は、ＣＡＤ／ＣＡＭにより作成されることを前提としている。勿論、マニュアルによる作成も可能ではあるが、大変煩雑になることが予想される。

例題部品のプログラム例（φ34 の穴部分の加工一カ所のみ）

Program File "Examp1.nc"

```
DESCRIPTION
{
        FILENAME = "C:¥DH¥WKS.DIR¥APZ.WPD¥EXAMP1.NC";
        PROGNAME = EXAMP1;
        DATE = "09-30-1996";
        AUTHOR = "WZL";
}
FILES
PLAN := APZFILE(NAME := 'C:¥TESTISO¥INPUT¥EXAMP1.DAT');

CONST
{
PLACEMENT1 = AXIS2_PLACEMENT_3D( LOCATION=CARTESIAN_POINT(COORDINATES=
[200.0000,100.000,10.0000]));
ORIGIN = SET_ORIGIN(LOCATION = PLACEMENT1);
PLACEMENT2 = AXIS2_PLACEMENT_3D( LOCATION = CARTESIAN_POINT(COORDINATES =
[620.0000,450.0000,300.0000]));
WITHDRAWAL_PLANE = ELEMENTARY_SURFACE(POSITION = PLACEMENT2);
}

PROGRAM EXAMP1;     // MAIN PROGRAM
{
EXEC ORIGIN;
EXEC WITHDRAWAL_PLANE;
EXEC PLAN.ROUGH_DRILLING_HOLE1;
EXEC PLAN.DRILLING_HOLE1;
}
```

Tool Data File "Examp1.too"

```
DESCRIPTION
{
        FILENAME = "EXAMP1.TOO";
        DATE = "09-30-1996";
        AUTHOR = "ST/ETH";
}

CONST
{
// *** TOOLS ***

TOOL3 = TOOL(
        TOOL_TYPE = .GENERAL_TOOL.,
        ID = '7',
        POSITION = 7,
        NUMBER_OF_TEETH = 2,
        DIMENSION = TOOL_DIMENSION(
        DIAMETER = 12.0000,
        EDGE_RADIUS = 0.0000,
     EDGE_CENTER_HORIZONTAL = 0.0000,
        TIP_LENGTH = 0.0000,
        TOOL_TOP_ANGLE = 0.0000,
        TOOL_CIRCUMFERENCE_ANGLE =     0.0000,
        TOOL_LENGTH = 75.0000);
        TECHNOLOGICAL_DATA = TOOL_TECHNOLOGICAL_DATA(
        CUTTING_ANGLE = 0.0000,
        FREE_ANGLE = 0.0000,
        ROTATION_DIRECTION = .CW.,
```

```
        CAN_UNDERCUT = FALSE));
    // DRILL 12MM

  TOOL5 = TOOL(
        TOOL_TYPE = .GENERAL_TOOL.,
        ID = '9',
        POSITION = 9,
        NUMBER_OF_TEETH = 2,
       DIMENSION = TOOL_DIMENSION(
        DIAMETER = 34.0000,
        EDGE_RADIUS = 0.0000,
        EDGE_CENTER_HORIZONTAL = 0.0000,
        TIP_LENGTH = 0.0000,
        TOOL_TOP_ANGLE = 0.0000,
        TOOL_CIRCUMFERENCE_ANGLE = 0.0000,
        TOOL_LENGTH = 75.0000),
        TECHNOLOGICAL_DATA = TOOL_TECHNOLOGICAL_DATA(
        CUTTING_ANGLE = 0.0000,
        FREE_ANGLE = 0.0000,
        ROTATION_DIRECTION = .CW.,
        CAN_UNDERCUT = FALSE));
// DRILL 34MM
}
```

Process Data File "Examp1.dat"

```
DESCRIPTION
{
        FILENAME = "EXAMP1.DAT";
        DATE = "09-30-1996";
        AUTHOR = "BR/WZL";
}

 FILES
        TOOL_DATA := APZFILE(NAME := 'EXAMP1.TOO');

CONST
{
        SECURITY_PLANE1 = ELEMENTARY_SURFACE(POSITION = PLACEMENT1);
        BLOCK = BOX_DIM_TYPE(MIN_VALUES = CARTESIAN_POINT(COORDINATES =
          [-320.0000,-150.0000,-101.0000]),

        MAX_VALUES = CARTESIAN_POINT(COORDINATES = [0.0000,0.0000,0.0000]));
        TEST_WORKPIECE = WORKPIECE(
MANUFACTURING_FEATURES                                                  =
[DRILL_HOLE1,BOHRUNG_3,BOHRUNG_4,BOHRUNG_5,BOHRUNG_6],
        ITS_MATERIAL = 50,
        ITS_GEOMETRY = BLOCK);
              PLACEMENT1 = AXIS2_PLACEMENT_3D( LOCATION = CARTESIAN_POINT(COORDINATES
= [0.0000,0.0000,5.0000]));
        // FOR SECURITY PLANE 1

        PLACEMENT4 = AXIS2_PLACEMENT_3D( LOCATION = CARTESIAN_POINT(COORDINATES =
[-290.0000,-32.5000,
```

```
        -1.0000]));
        // FOR DRILL HOLE 1


  // *** MANUFACTURING FEATURES

  DRILL_HOLE1 = HOLE(
        FEATURE_PLACEMENT = PLACEMENT4,
        ITS_WORKPIECE = TEST_WORKPIECE,
        DEPTH = 30.0000,
        ID = '4',
        MANUFACTURING_DATA = [ROUGH_DRILLING_HOLE1, DRILLING_HOLE1]);


// *** TECHNOLOGY ***

        TECHNO4 = TECHNOLOGY( FEEDRATE = 400.0, SPINDLE = 1273.0);
        TECHNO5 = TECHNOLOGY( FEEDRATE = 400.0,  SPINDLE = 748.0);


// *** WORKINGSTEPS ***
        ROUGH_DRILLING_HOLE1 = HOLE_DRILLING_MANUF_DATA(
       ITS_ID = 'ROUGH_DRILLING_HOLE1',

       ITS_FEATURE=DRILL_HOLE1,
       ITS_SECPLANE=SECURITY_PLANE1,
       ITS_TOOL=TOOL_DATA.TOOL3,
       COOLANT=TRUE,
       ITS_TECHNOLOGY = TECHNO4,
       RETRACT_PLANE = SECURITY_PLANE1);
```

```
DRILLING_HOLE1 = HOLE_DRILLING_MANUF_DATA(
ITS_ID = 'DRILLING_HOLE1',
ITS_FEATURE = DRILL_HOLE_1,
ITS_SECPLANE = SECURITY_PLANE1,
ITS_TOOL = TOOL_DATA.TOOL5,
COOLANT = TRUE,
ITS_TECHNOLOGY = TECHNO5,
RETRACT_PLANE = SECURITY_PLANE
```

C. プロセス データ モデル

Process Data Model は、前述したように、Process Data の記述のためのモデルであり、本規格の中核をなす部分と言える。このモデルは、ＥＸＰＲＥＳＳで表現されているが、以下では、その主要な個所について、ＥＸＰＲＥＳＳ－Ｇなどを用いて、2.5 次元加工を中心に説明することにする。

a) Manufacturing_feature（図 4-13、図 4-14、図 4-15）

Manufacturing_feature の種類とそのデータ構造は、基本的には、ＳＴＥＰ ＡＰ２２４に準拠している。ただし、幾何情報に関しては、ＡＰ２２４は B-Rep Model を基本としているが、このモデルの 2.5 次元部分では、直線、円弧及びその組み合わせ[Contour]で表現することにしている。

b) Workingstep のデータ構造[全体] （図 4-16）

この部分では、2.5 次元加工及び 3 次元自由曲面加工に共通な属性を記述している。

Machine_function ではクーラントやミストの使用を、Technology では切削条件を記述できる。また、Process_model は、例えば、加工中の切削負荷を自動的に検出して、それにより、加工条件を変更させる機能（適応制御）などを使用するためのものである。

c) Workingstep のデータ構造[2.5 次元加工] （図 4-17）

- Approach_retract_strategy は、加工開始前の入口部及び加工終了後の出口部の工具の動きを設定するところである。
- ２．５次元加工用の作業は、まず、

  Milling_manuf_data ――― Ｘ軸及びＹ軸で加工を行なう作業

  Hole_manuf_data ――― Ｚ軸で加工を行なう作業

に大別される。ただし、これは、Manufacturing_feature の定義と対応しているものではない。例えば、Feature が“ Hole ”であっても、Milling_manuf_data を用いて、エンドミルによるコンタリング加工を記述できる。

Milling_manuf_data、Hole_manuf_data ともに、後述する加工用工具動作の分類に従って、必要となる作業を自由に記述することを基本としている。ただし、Milling_manuf_data では、典型的な Feature に関してその加工に必要な標準作業を準備している部分があり、それが Feature_manuf_data である。図 4-18

に、その構造を示す。

d) Generic_manuf_data の構造（図 4-19）

Milling_manuf_data で、自由に作業を記述する場合は、Generic_manuf_data を使用する。

フライス加工の基本作業は、平面加工、側面加工、底と側面の加工及び各々の荒加工、仕上げ加工の６種類である。この Generic_manuf_data も、先に述べた Feature_manuf_data も、tool_motion_data は、この６種類の分類に基づいた図 4-20 に示す Milling_motion_data を参照していることを確認されたい。

また、Milling_motion_data では、加工中の工具動作の種類を、次に述べる Milling_type の中から指定する。

e) Milling_type の種類（図 4-21）

フライス加工用工具動作の種類として、標準としては、７種類を準備している。図 4-22 には、その内の６種類についてそのイメージを表現している。標準以外の動作が必要となる場合には、User_defined_milling を用いる。

f) Hole_manf_data の構造（図 4-22）

- 穴加工用の基本作業は、図 4-22 に示すように、８種類が用意されている。
- その各作業の tool_motion_data は、図 4-23 に示す Hole_manuf_motion_data を参照している。

  穴加工用の工具動作の種類は、標準として、そこに示す５種類を用意している。各 motion_data の属性を設定することにより、工具動作を実現できることになる。
- Drilling_strategy は、穴の入口及び出口部分で、加工条件を変更する時に使用する。
- Boring_motion_data では、寸法計測用の動作も記述できるよう配慮されている。

この Process Data Model は、新しいＮＣプログラム言語用として開発されたものであるが、ＣＡＭで必要される加工技術データベース構築用モデルとしても有効と考えられる。つまり、各 Feature 毎にその加工方法をこのモデルに従って記述しておけば、Feature に対する加工方法を自動的に設定できることになる。ただし、対象とする Feature に対して、登録されている加工方法でよいか

を判定するには、大きさ、被削材質、公差、仕上げ面粗さなど Feature の情報としてこのモデルでは不足しているものを補ってやる必要がある。

D. プリデファインドファンクション

Predefined Fuction は、従来からのＮＣ言語（ＩＳＯ６９８３）を、人間がより理解しやすい言語にしたものと考えればよいものである。つまり、これを使用すれば、従来と全く同じ感覚でＮＣプログラムを作成できるわけである。また、いままで述べてきた新しいＮＣ言語の中で、それと混合して使用できることになっている。Predefined Fuction を混合して使用することで、よりきめの細かい指令を作成することもできることになる。

第１ステップの規格文書では、

- ＩＳＯ６９８３とほぼ同等の機能を持つものを正式規格［Normative］
- 拡張された機能の範囲は、参考資料［Proposal］

として、提案しようとしている。

E. 全体の進捗状況

現在、第１ステップで提案される規格文書の構成は、以下のものを考えている。

- Part1　Overview and Fundamental Principles
- Part10　General Process Data
- Part11　Process data for milling
- Part111　Tool for milling

なお、規格化への段階としては、現在、ＤＩＳ投票段階である。

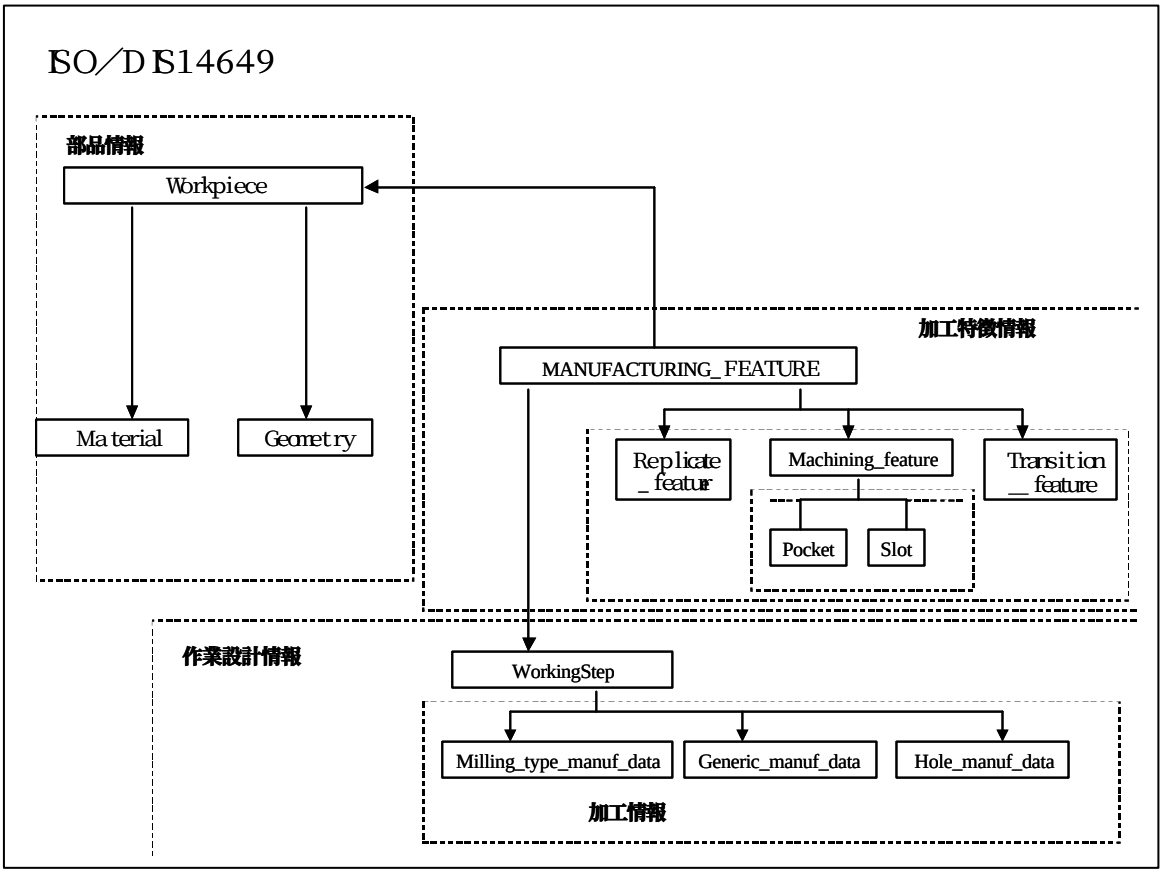
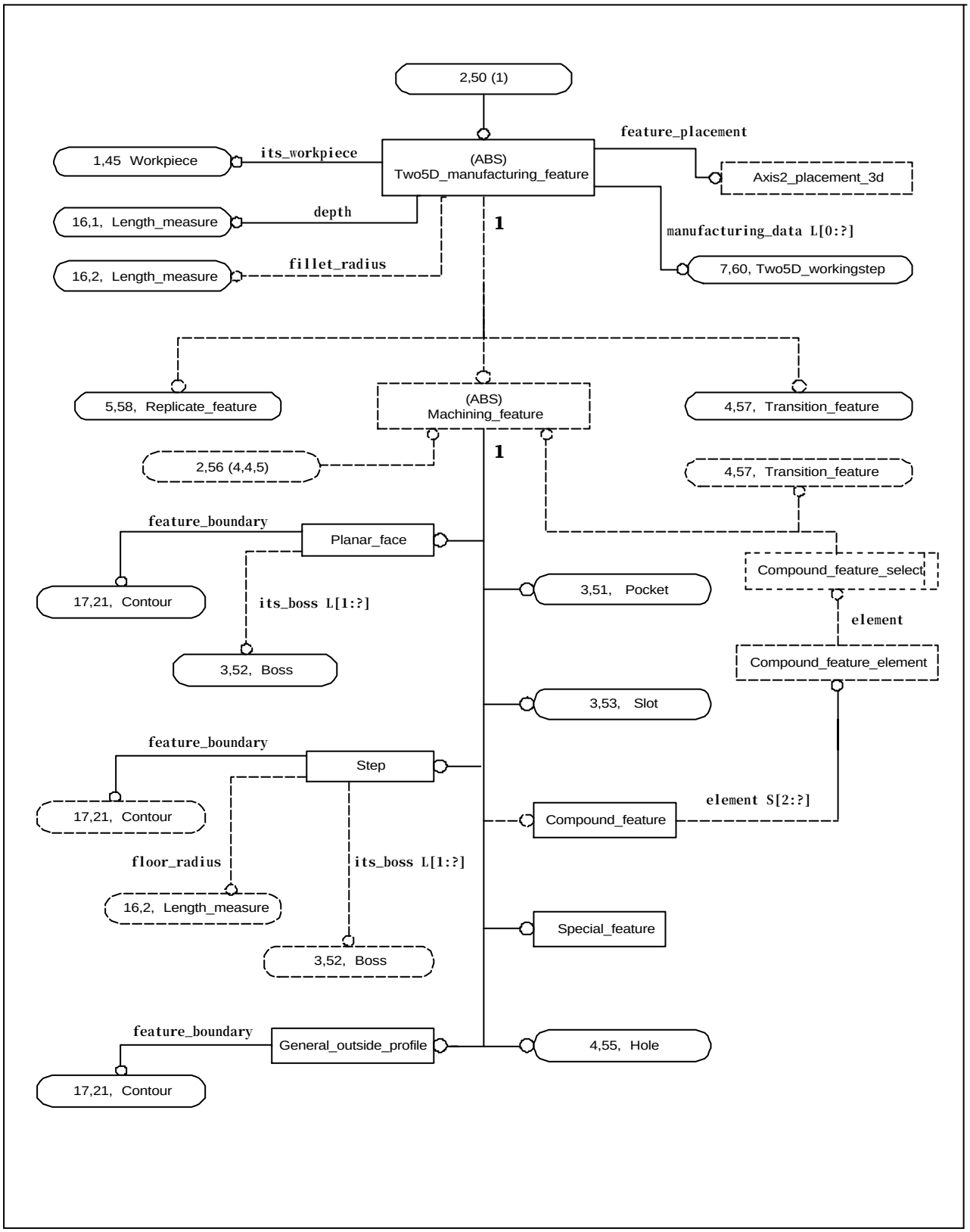
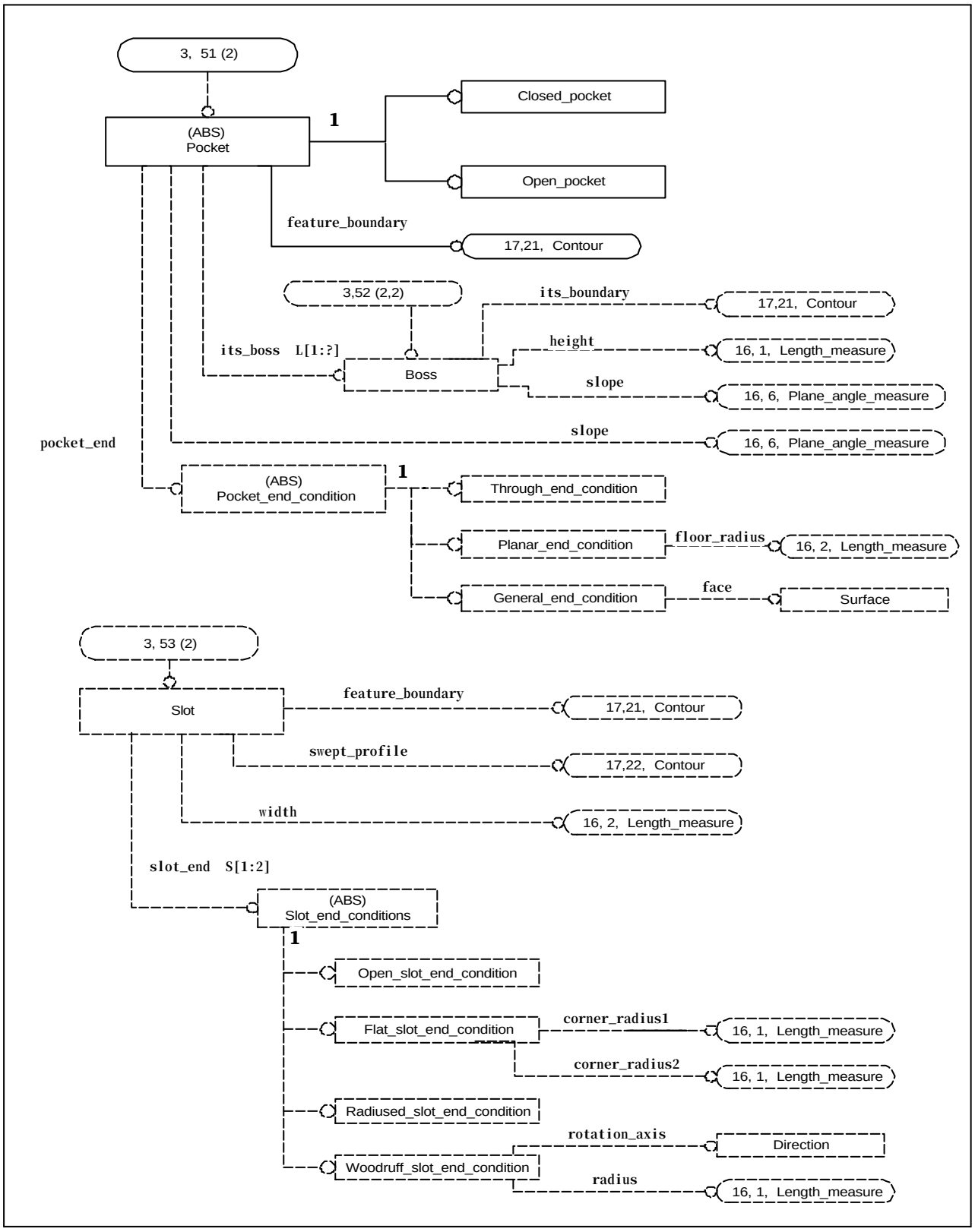


図 2-12 CNCデータモデルの情報モデル



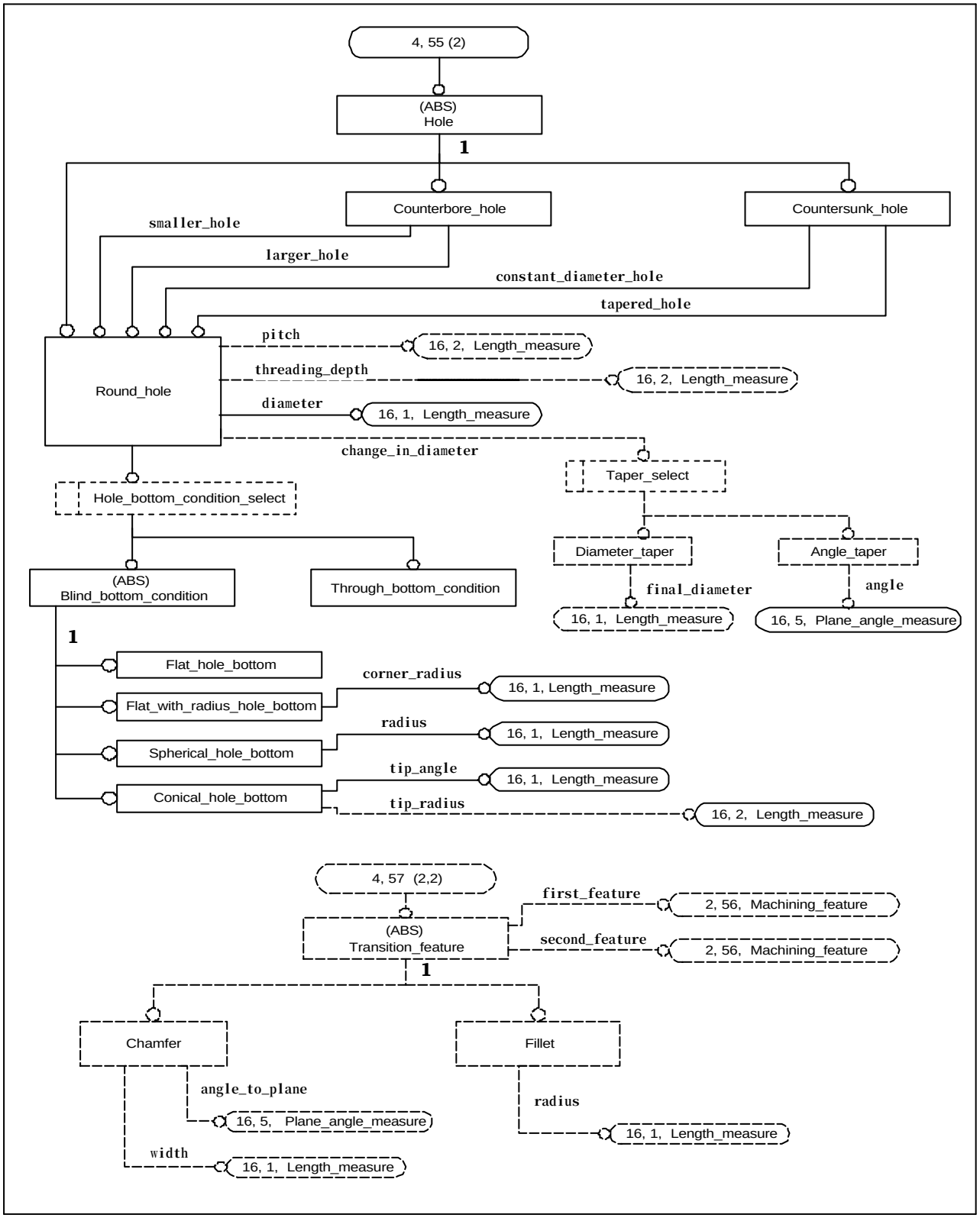
[Two5D_manufacturing_feature]

図 2-13 Manufacturing_feature の種類（2.5 次元）



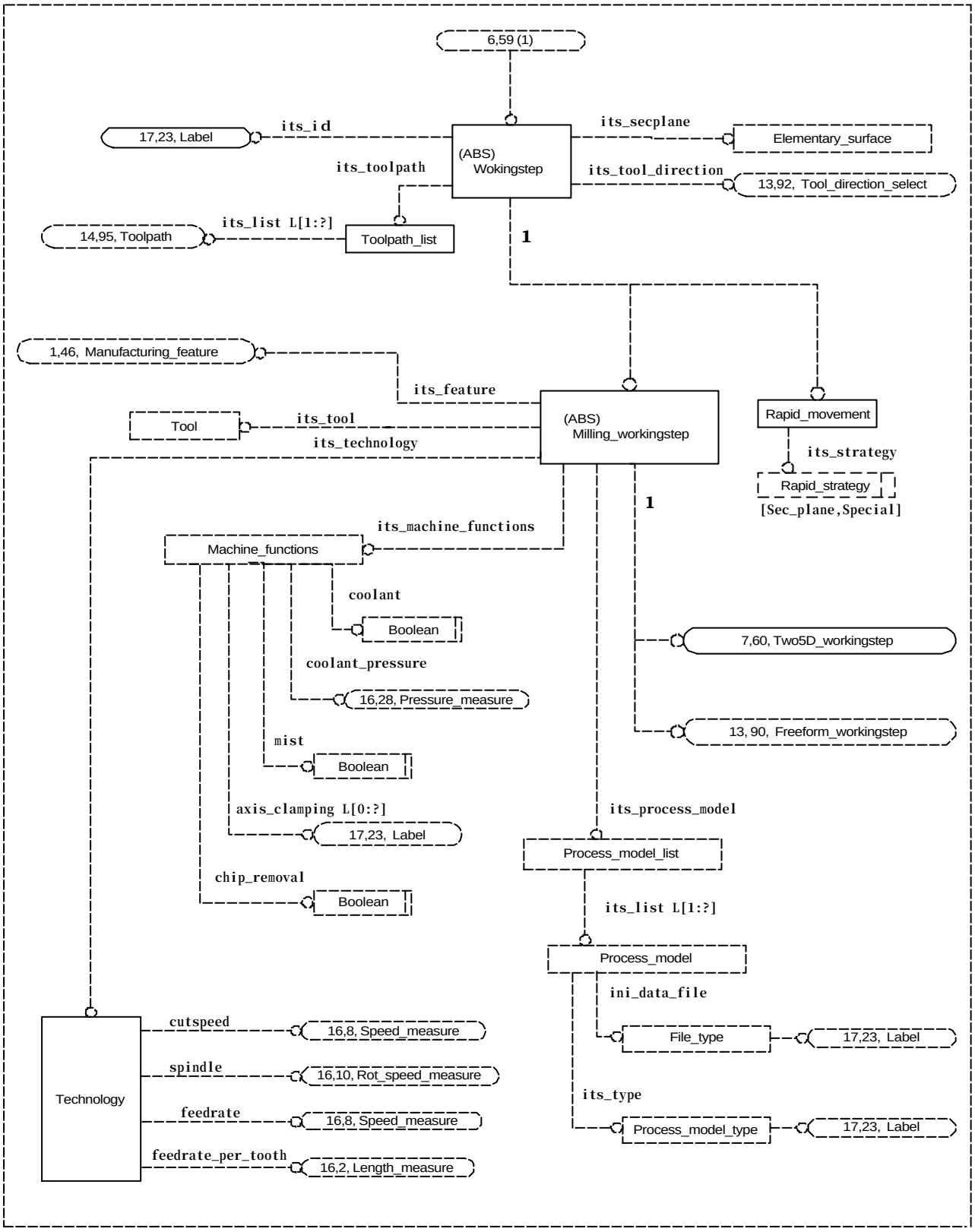
[Pocket, Slot]

図 2-14 Pocket,Slot データ構造



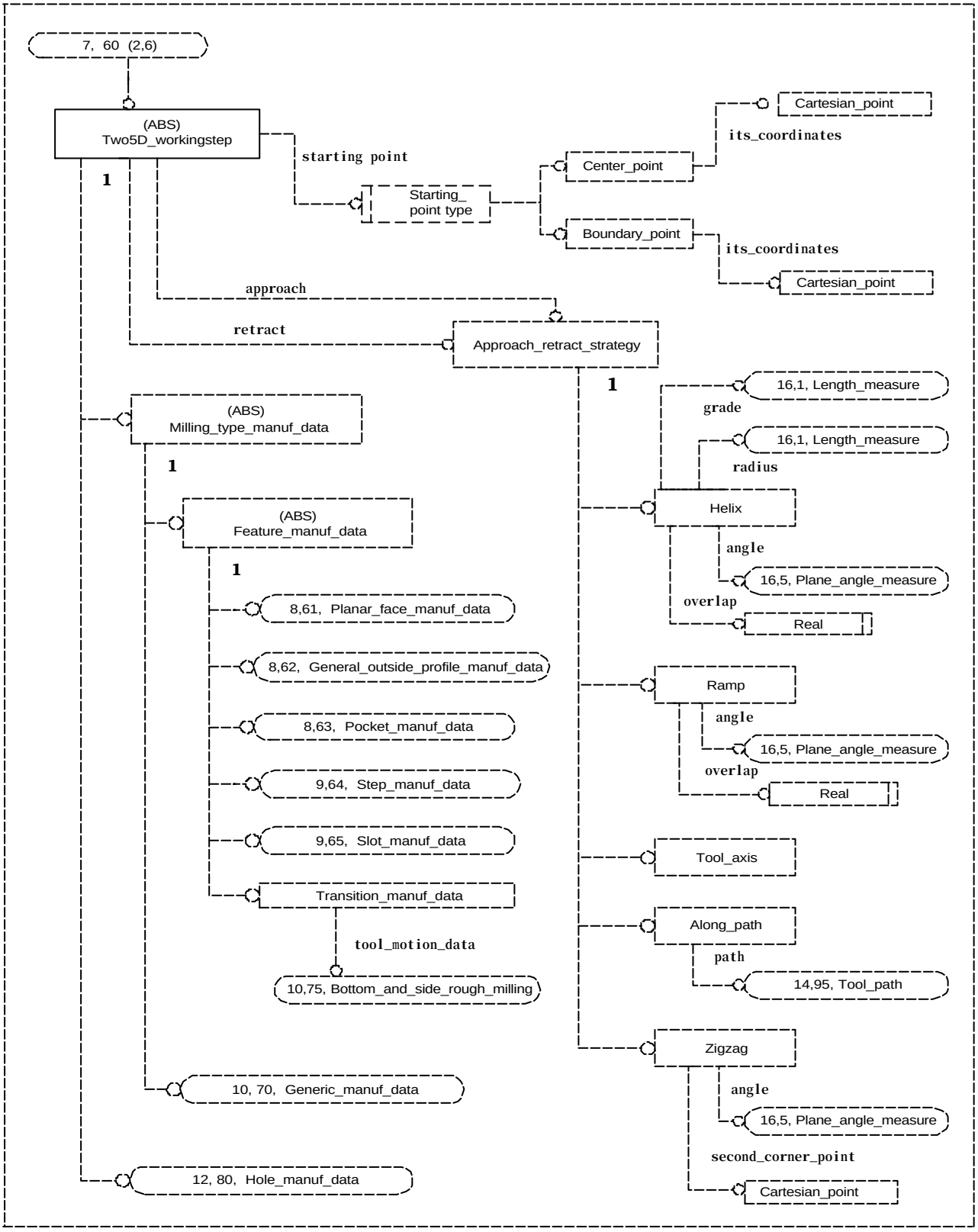
[Hole, Transition_feature]

図 2-15 Hole のデータ構造



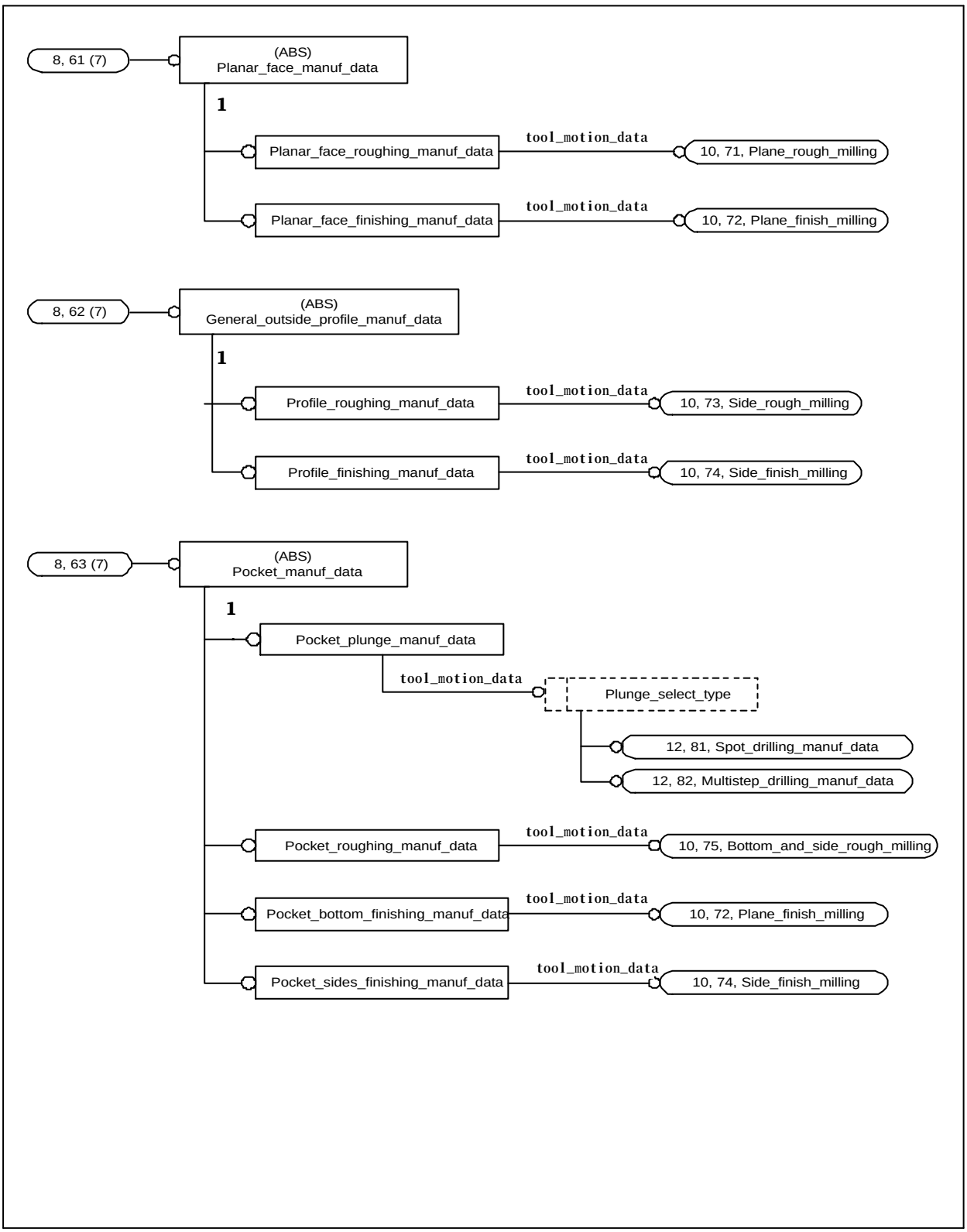
[Milling_workingstep]

図 2-16 Workingstep のデータ構造 全体）



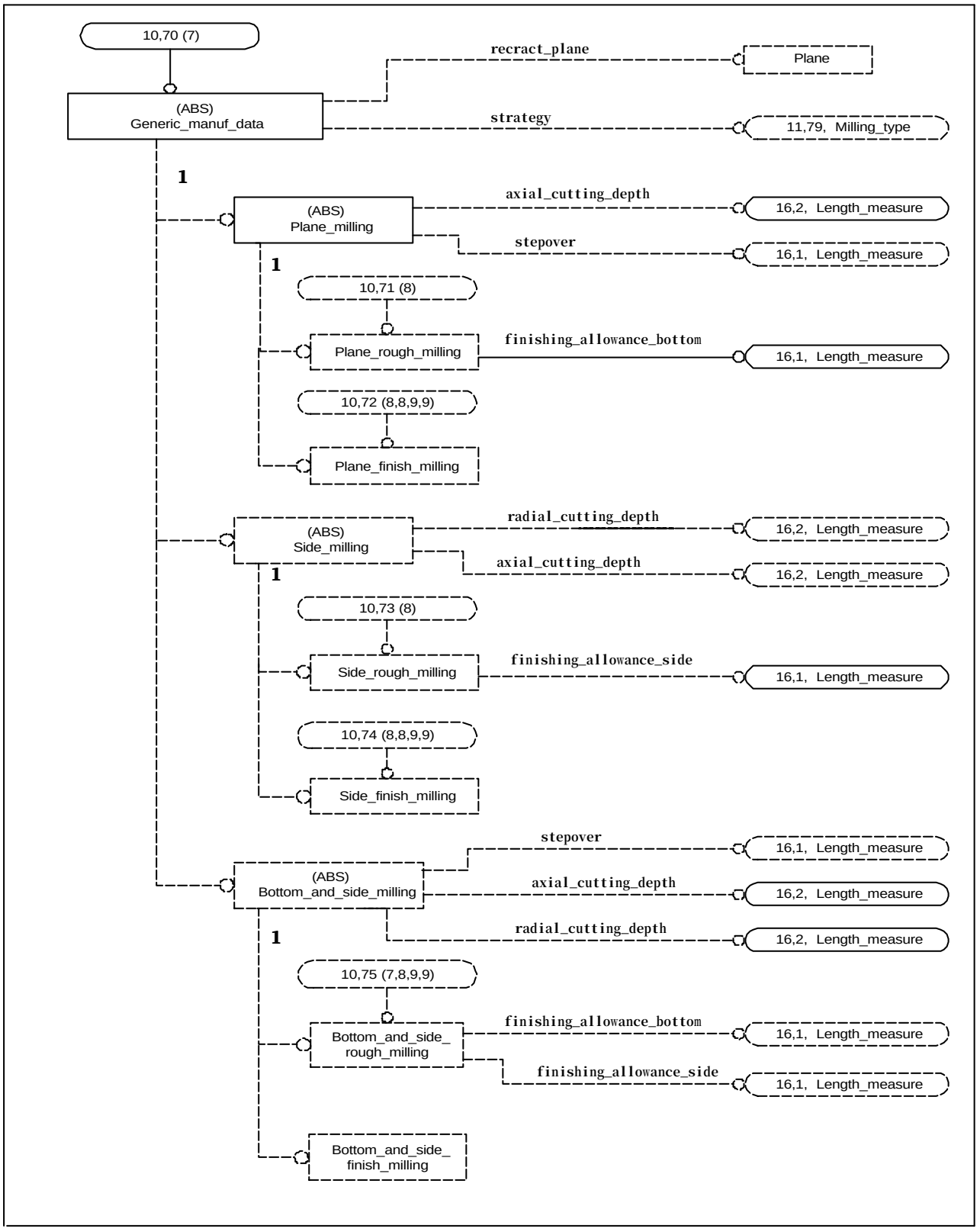
[Two5D_workingstep]

図 2-17 Workingstep のデータ構造（2.5 次元）



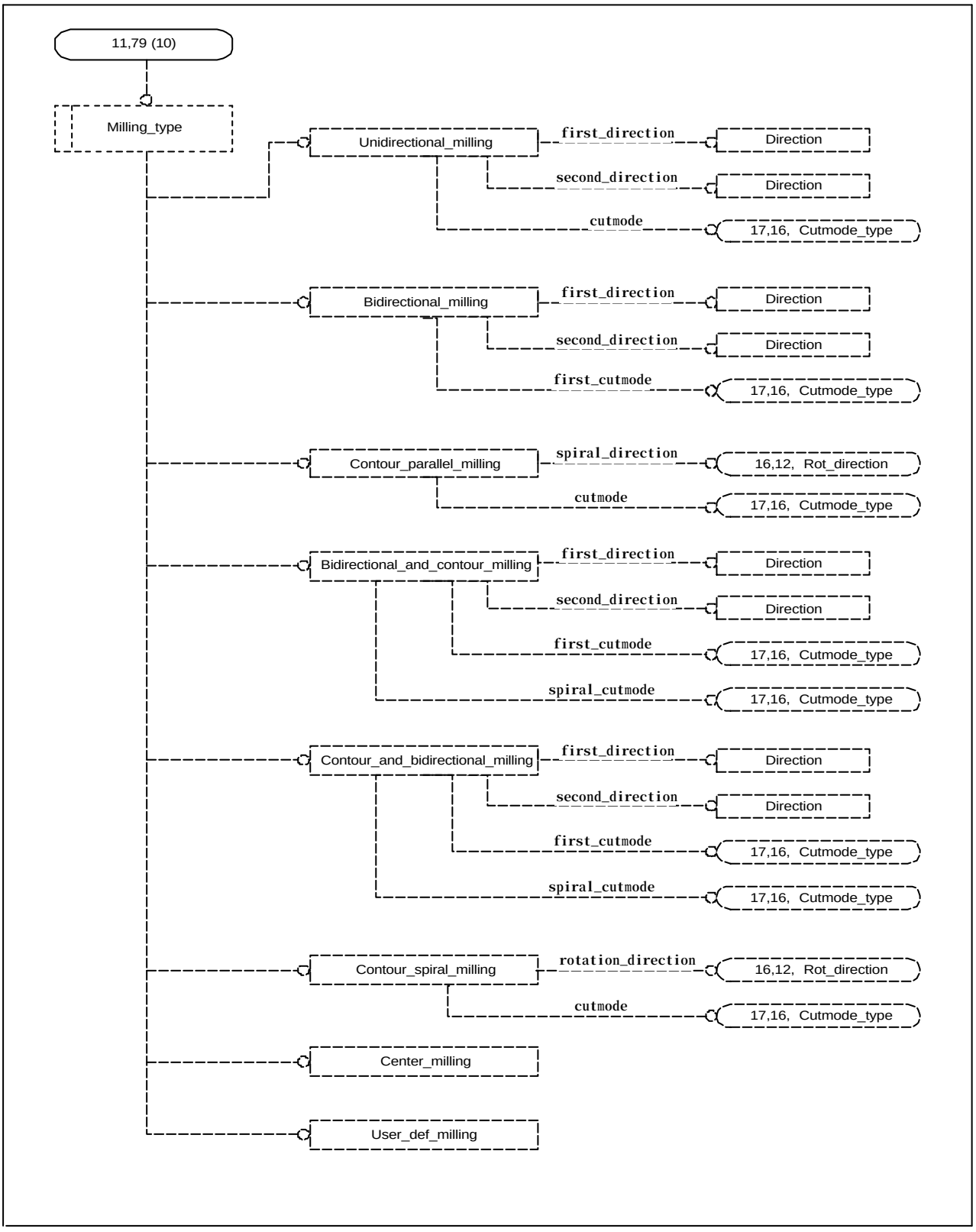
[Planar_face_manuf_data, Outside_profile_manuf_data, Pocket_manuf_data]

図 2-18 Feature_manuf_data のデータ構造の例



[Generic_manuf_data]

図 2-19 Generic_manuf_data（ワイス加工）の構造



[Milling_type]

図 2-20 Milling_type(フライス加工用工具動作)のデータ構造

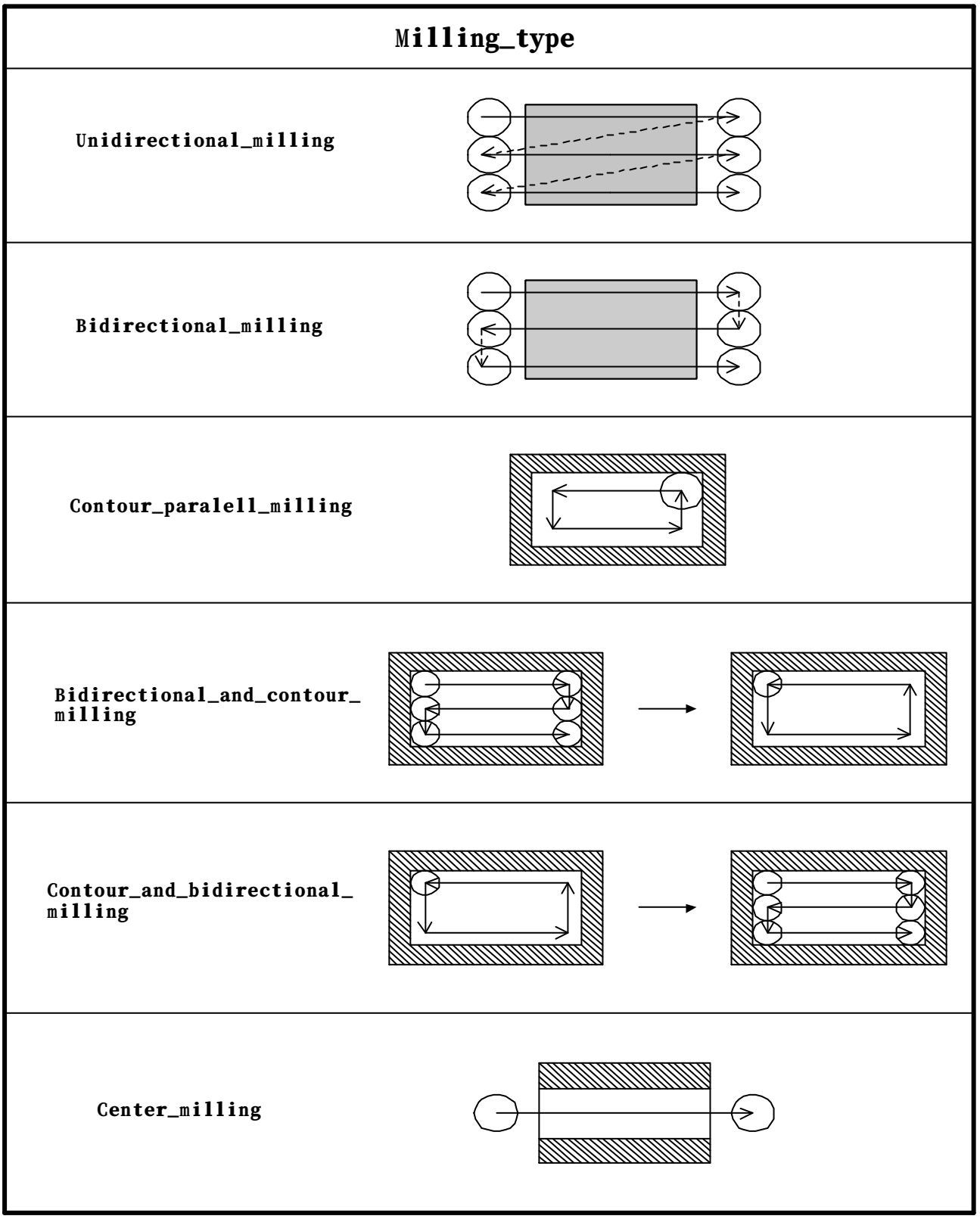
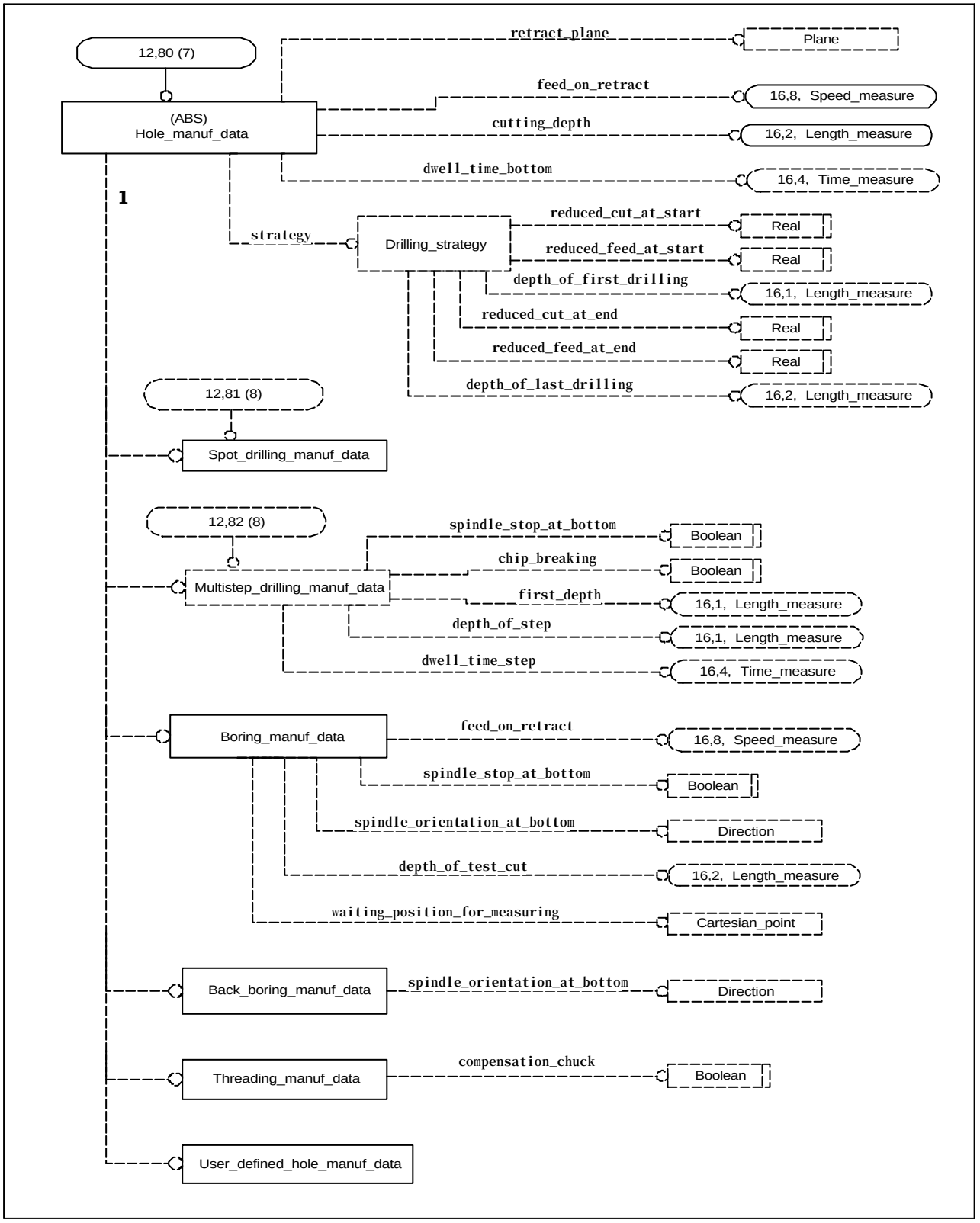


図 2-21 フライス加工用工具動作の種類



[Hole_manuf_data]

図 2-22 Hole_manuf_data の構造

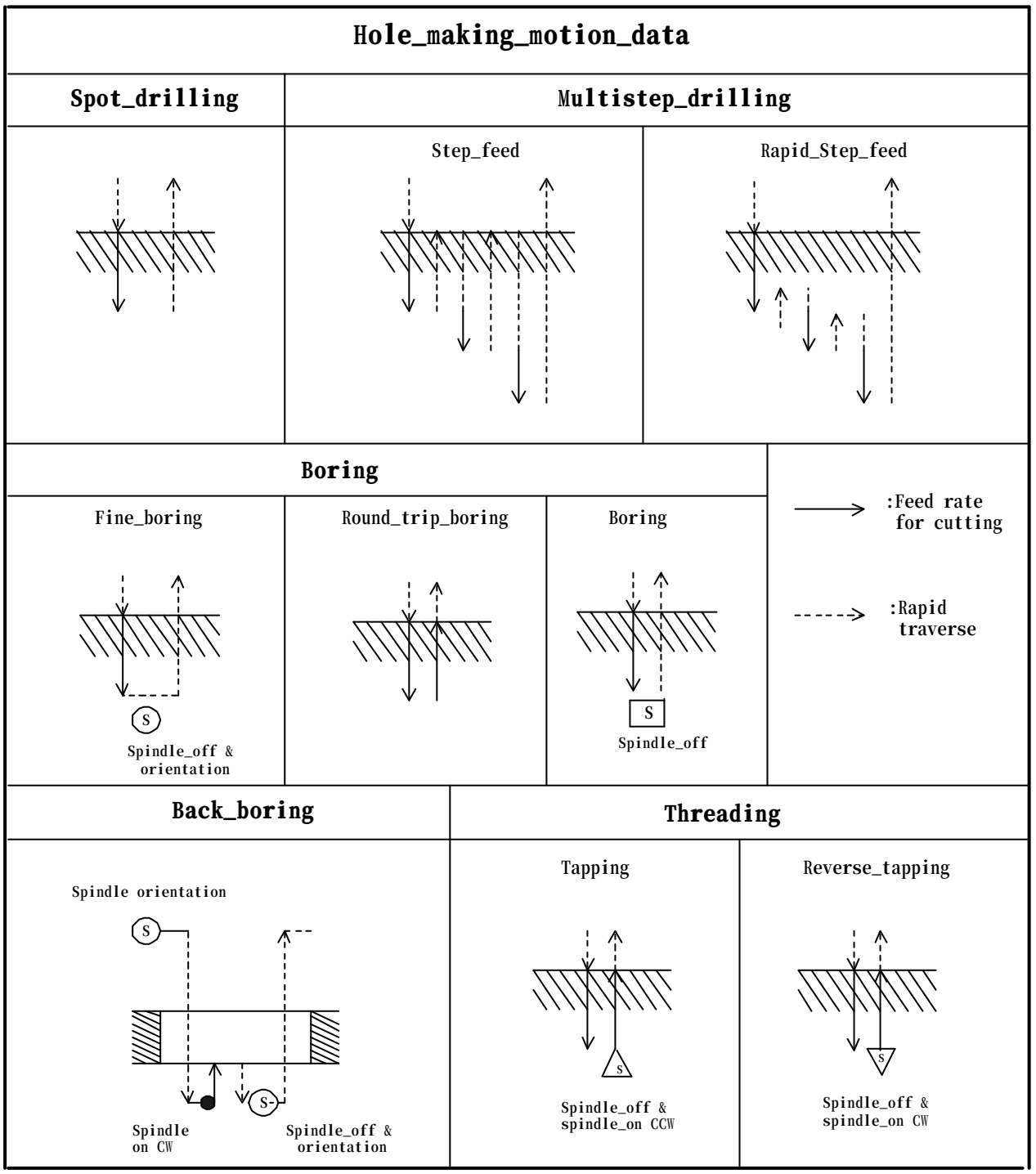


図 2-23 穴加工用工具動作の種類

2.6　ＳＴＥＰを利用した製品モデルデータによるデジタルマニュファクチャリング

前節でＳＴＥＰのＡＰ２０３／ＡＰ２１３／ＡＰ２２４／ＩＳＯ１４６４９による製品モデル表現を概説したが、これらの製品モデルを利用してデジタルマニュファクチャリングのシミュレーションを行なってみる。前節に説明したモデルは全て製品（部品）により情報統一が行なえるので、製品エンティティをキーとした情報モデルの統合を考えると図2-24のように表現できる。

この図で明らかなように製品設計から加工工程までの全ての情報はＳＴＥＰのような製品モデル表現を利用すれば統一的に扱う事ができる。

図 2-24で製品の設計から加工までの情報がどのように作られ統合されるのかを検討してみる。

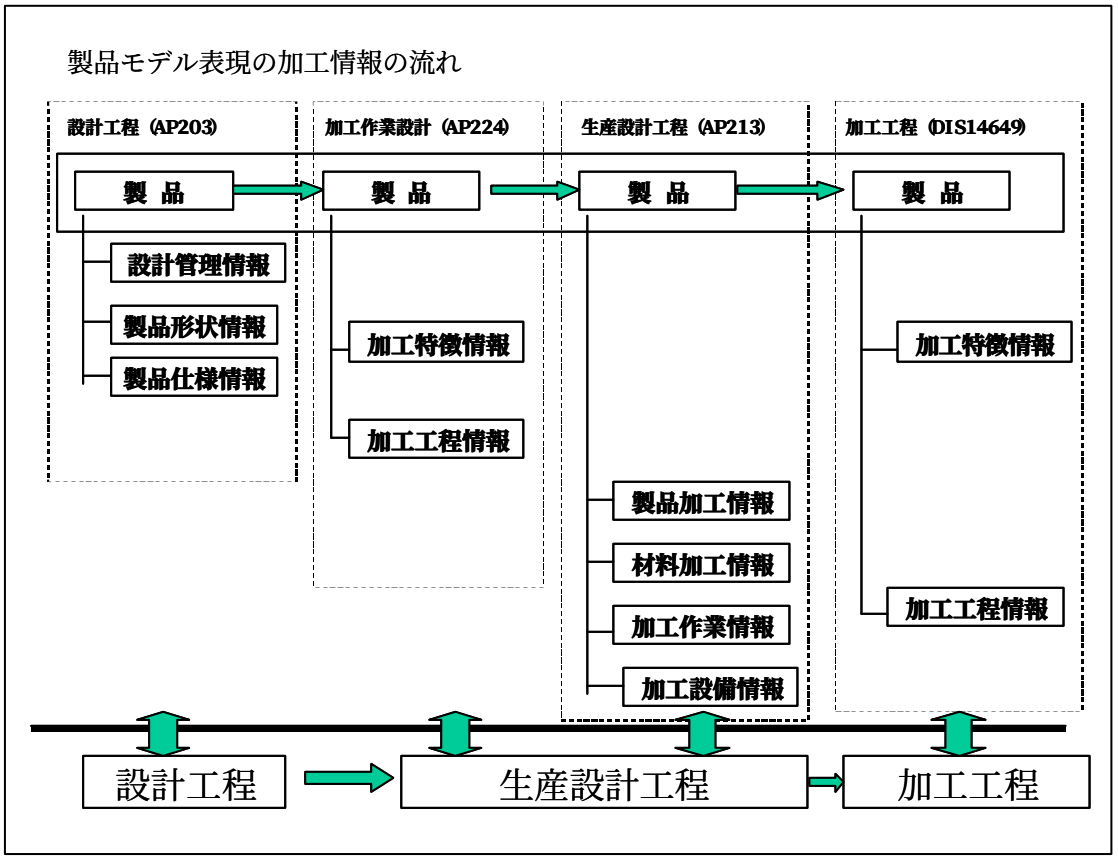


図 2-24 生産プロセスでの加工情報の流れ

(1) 設計工程（図 2-6、図 2-24参照）

設計工程ではＣＡＤを使用して製品の設計を行なう。

製品コンセプトに基づき 3 次元ＣＡＤ上で形状のデザインと部品構成、材料、製品の仕上げに関する仕様を決めて製品設計を行なっていく。設計が完成したら査閲者のチェックと承認を受けて、製品設計モデルをリリースする。この製品モデル情報にはバージョン番号、作成年月日、作成者等の管理情報と形状と製品設計仕様が含まれている。

(2) 加工作業設計工程（図 2-8、図 2-24参照）

製品の完成した設計モデルが出来たら、部品毎のＮＣ工作機で加工するための情報をこの工程では検討する。主な作業は 3 次元形状の幾何情報からＮＣ工作機が加工するための加工特徴の検討を実施していく。デザイン上例外があれば例外処理の情報を参照しながら作業を行なう。

また、加工のため機械設備情報や加工工程での情報も検討して製品モデルに入れていく。

(3) 生産設計工程（図 2-7、図 2-24参照）

以上の工程の情報を基にして実際に部品を加工するための情報を検討する。形状に関わる情報、材料情報、加工する対象設備での工作機械の種類、工具等の加工設備情報を基にして加工作業情報を作り上げていく。

(4) 加工工程（図 2-12、図 2-24参照）

この工程では実際に工作機械を使用し削り出すための情報を作り出す。（２）の工程で作られた加工特徴毎に工作機械の作業を決定していく。WorkingStep は加工単位の作業を表現したものである。また、（３）で検討したツール、設備の情報を参照して作業設計情報は作られる。

以上の様にこのような統一的な製品モデルデータを使用すれば、デジタルマニュファクチャリングは実現できる。特にＳＴＥＰは実装するシステムによらないニュートラルな情報モデルであるので機械生産プロセス情報の統合化は比較的簡単であろう。

2.7　デジタルマニュファクチャリング実現のためのＸＭＬ

ネットワークによる製品モデル共有によるデジタルマニュファクチャリングを検討してみる。インターネットによるマニュファクチャリングデータ共有に関してはＳＴＥＰ__Ｎ

Ｃ他に多くの研究提案が行なわれているが、日本の「新規産業支援型国際標準開発事業：生産プロセスシステム委員会」で議論されている内容を中心にまとめる。

本章はＴＣ１８４／ＳＣ４チャールストン会議（2000 年 10 月）に生産プロセスシステム委員会より提案した「ＩＳＯ ＴＣ１８４／ＳＣ４／WG３ Ｎ９７５：XML-based High Level ＮＣ data Modeling for Manufacturing Data Management」を基にその内容を示す。

(1) ネットワークによるマニュファクチャリングデータ共有の意義

設計と生産は、サプライチェーンのメカニズムの中で、海外との連携も含めて開発のスピードをさらに向上させる必要に迫られている。そのような状況において、製品、品質、検査、訓練といった情報を共有することが必要になっている。また、正確なビジネスの取引のためには、さらに効果的な情報技術が必要とされている。

そのような背景のもとに、以下の点を目的としてＮＣデータモデルを検討した。

- 迅速で仮想的な、プロトタイプ、シミュレーション、製品試験、生産試験の実現
- 最適なマニュファクチャリングツール、知識ベース、製品情報、場所に依存しない訓練方法のより容易な実現

(2) マニュファクチャリングシステムの操作環境

デジタルマニュファクチャリングに必要な環境と要求事項をまとめる。

① コミュニケーションとデータ交換

マニュファクチャリングは製品データ、マニュファクチャリングデータ及びマネージメントデータ交換のためのマルチメディア情報環境を包含し、また、地理的に離れた関係者間のコミュニケーションを可能とする環境も含む。

② データ抽象化

人間とコンピュータシステム間のコミュニケーションのギャップを埋めることが可能な、新しいＷｅｂテクノロジーが必要である。

③ 知識の取得と学習

マニュファクチャリングデータ管理システムは、他のマニュファクチャリングサイトとデータを共有するために、マニュファクチャリングプロセスに関するデータを取得・組織化できるようにすることが必要である。

④　共同診断

リモートでメインテナンスを行なうユーザのために、マルチメディアのツールが必要である。インタラクティブで共有できるツールは、リモートでの診断を可能にする。

以上の実現のために、「Ｗｅｂ／ＸＭＬをベースにしたマニュファクチャリングデータマネージメントシステムのための、ＸＭＬをベースとする高レベルＮＣデータモデル」を提案する。これは、分散するマニュファクチャリングサイトや共同作業者のために、製造設備の性能、操作の品質、向上の生産効率を保証するものである。

(3) マニュファクチャリングデータ管理システムへの要求

機械系製造業のサプライチェーン実現のためには以下の要件を考慮する必要がある。

- 設計と製造に関係する知識の共有
- 製品ライフサイクルに関係する全ての情報を含むこと
- 製品の付加価値を高くし、以下のような予想される問題を解決すること
- 情報の統合と共有の難しさ
- 異なる製品データのシステム化された管理の不足
- 製品のデータ及び情報共有のために、ＷｅｂベースのＰＤＭシステムへ接続可能なこと

(4) マニュファクチャリングデータ管理システムの機能

①　マニュファクチャリング情報

- プロセスプランニングのために、機械製品の定義を行なうためのもの
- プロセスプランニングの部品を定義するために、部品識別、トラッキング、形状表現、材料データを表現するためのもの
- 人間とコンピュータとを結びつけるマニュファクチャリングシェイプを識別するためにマニュファクチャリングフィーチャが必要

②　材料及び管理情報

- プロセスプランニングアクティビティは、必要な設備と材料を確定することができる。管理情報は顧客とサプライヤ両方のトラッキングのために必要とされる。
- 管理情報のトラッキングはプロセスプランを作成するために繰り返す課程の一部分となる。

③ 高レベルＣＮＣデータフォーマット

- ＣＡＭシステムとコンピュータ数値制御（ＣＮＣ）マシンツールとの高レベルインターフェースが必要である。
- このフォーマットには、マニュファクチャリング形状を生成する切削操作を定義するために、マニュファクチャリングフィーチャが含まれる。
- Ｗｅｂ／ＸＭＬをベースとしたマニュファクチャリングデータ管理システム

Ｗｅｂ／ＸＭＬをベースとしたマニュファクチャリングデータ管理システムの概要を、図 2-25に示す。

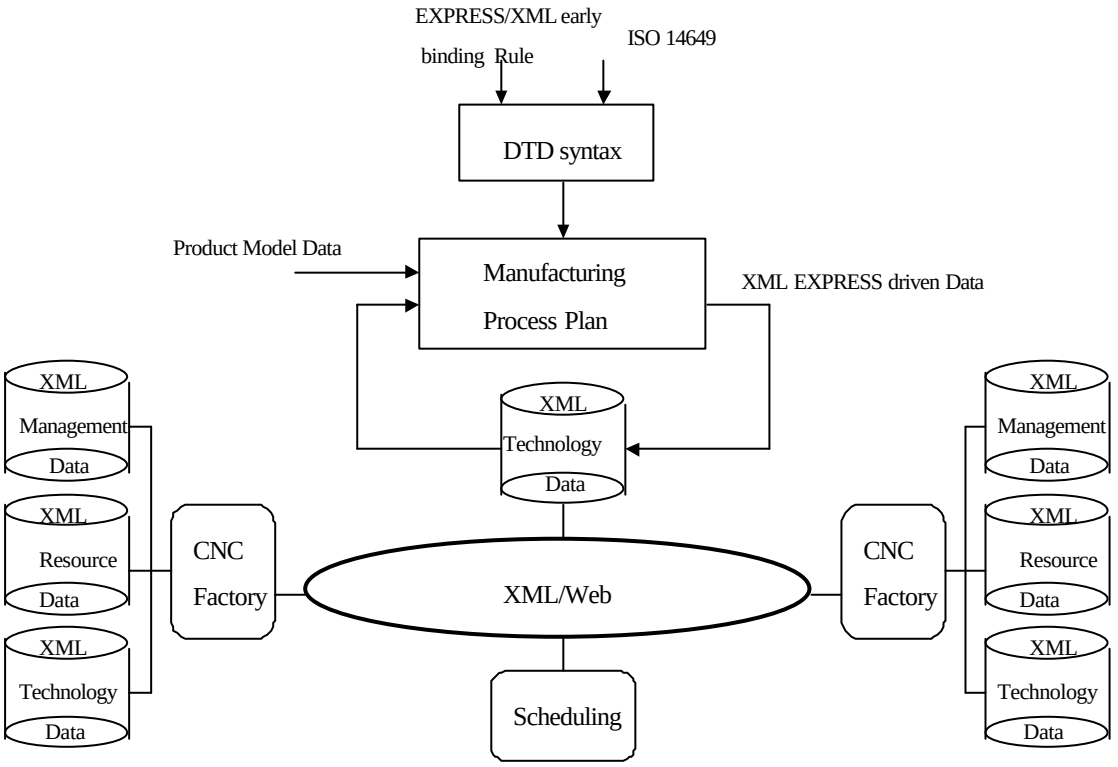


図 2-25 Web／XMLをベースとしたマニュファクチャリングデータ管理システム

(5) 高レベルＮＣデータとその役割

ＩＳＯ１４６４９（CNC Data Model）はＣＡＤ／ＣＡＭとＣＮＣマシンとのデータ変換の新しいモデルとして提案されている。

この規格では、マシンツールモーションよりむしろマシニングプロセスが、オブジェクト指向モデリング（ＳＴＥＰ／ＥＸＰＲＥＳＳが使用されている）を使用して定義されている。

提案されているワーキングステップは高レベルマシニングフィーチャとアソシエイティッドプロセスパラメータに相当している。

この目的を以下に示す。

- ＳＴＥＰを利用してコンピュータが生成した製品データの直接利用をサポートする
- 最新のＣＮＣ装置とＣＡＤ／ＣＡＭシステムに適用できるＮＣ入力データ、ＣＮＣデータモデルのコンパチビリティを確保する
- ＩＳＯ６９８３とＩＳＯ１４６４９の比較：図 2-26にＩＳＯ６９８３とＩＳＯ１４６４９との比較を示す。

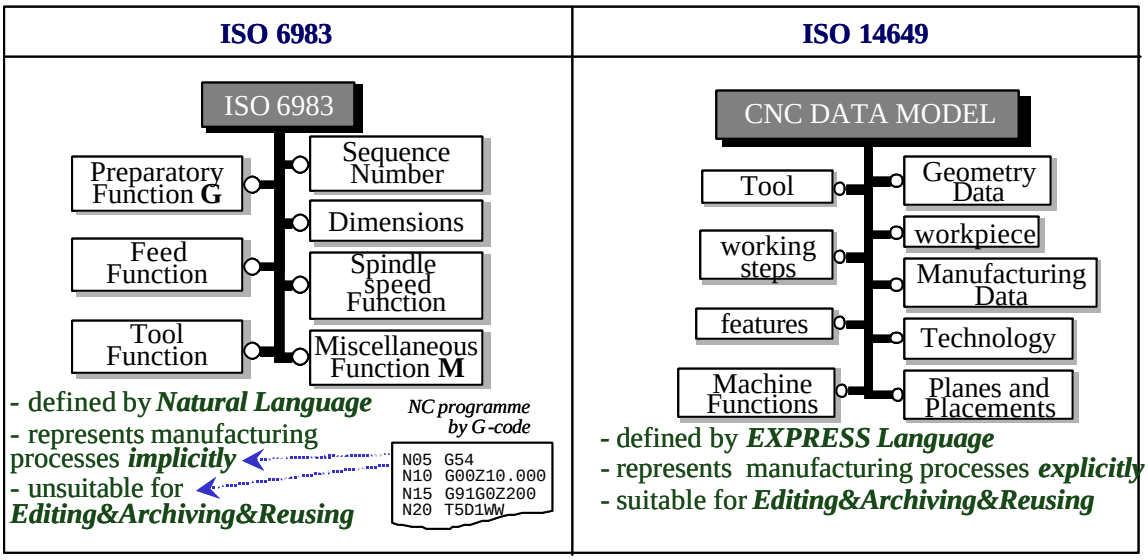


図 2-26　ISO6983とISO14649の比較

ＩＳＯ１４６４９の本質は、むしろデータ交換のためのものではなく、Ｗｅｂ／ＸＭＬにより製品データベースの共有を実現するための基礎にもなり得ると言える。

(6) ＸＭＬ (eXtensible Markup Language)

ＸＭＬは、人間とコンピュータシステムのコミュニケーションを実現するために存在

するギャップを埋めることができるドキュメントテクノロジーである。

またXMLは、ハイパーリンクとオブジェクト指向技術をベースとした有力な情報配信技術である。この機能は、マニュファクチャリングプロセスに必要なデータを作成するにあたって、重複を排除するために有効であると言える。

(7) ＥＸＰＲＥＳＳとXML

ＩＳＯ１４６４９（CNC Data Model）は、ＳＴＥＰ／EXPRESS（ＩＳＯ１０３０３）を利用して開発されている。ＥＸＰＲＥＳＳ言語はエンティティ－アトリビュートモデルを定義するための機能を、完全に用意している。

ＥＸＰＲＥＳＳの定義では、リポジトリに含まれるエンティティは、物理的な表現を持たない、抽象的なデータオブジェクトになる。このことは、ソフトウェア開発者がリポジトリソフトウェアを開発するにあたって、特定の技術やアプローチに縛られなくなることを示している。また、データ交換の問題を、リポジトリ間あるいはＥＸＰＲＥＳＳのリポジトリの間でも、オープンにすることができる。

ＩＳＯ１０３０３では、この問題をＩＳＯ１０３０３ Part21 で定義されているキャラクタベースの文法で解決しようとしている。しかし、Part21 の文法には拡張性がない（ハイパーリンクの仕組みを持っていない）。

(8) ＥＸＰＲＥＳＳか Part 21 か？

以下に示すような理由から、ＥＸＰＲＥＳＳのエンティティインスタンスのキャラクタベースでの交換には、XMLを利用するのが自然であると考えられる。

- XMLは、構造化されたデータオブジェクトを表現することができる、一般的で強靭な文法を定義する。
- XMLは公式ルールに反するドキュメントを検査する機能を提供する。一般的なXMLの評価ソフトを使用して、ＥＸＰＲＥＳＳの制限条件をチェックすることができる可能性がある。
- XMLは拡張性と柔軟性が高く、Part21 では提供できないデータ交換の機能を可能にする。
- XMLはＷｅｂブラウザでサポートされるため、Ｗｅｂブラウザの技術を使用することにより、ＥＸＰＲＥＳＳのエンティティを容易に表示することが可能となる。
- XMLによる表現は、データ交換の中間ファイルとしてだけではなく、製品デ

ータベースの構築及び共有の基礎にもなる。

(9) ＥＸＰＲＥＳＳへのＸＭＬのバインド

ＸＭＬのＥＸＰＲＥＳＳへのアーリーバインドは、以下のような効果が考えられる。

- 表現やモデルのマッピングを、よりユーザにわかりやすい形で提供することができる。
- マッピングの作業量を削減することができる。これにより、プログラミングを容易にし、エラーを少なくすることができる。

提案するマッピングルールを以下に示す。

- タグ名はＥＸＰＲＥＳＳのエンティティ名かアトリビュート名と一致させる。
- タグのエレメントは、アトリビュートバリューと一致させる。
- タグ構造をシンプルにするために、タグアトリビュートにマルチファンクションを導入する。
- リンクによるリファレンスメカニズムは、エンティティインスタンスの最も外側のタグを参照することにより実現させる。

ＥＸＰＲＥＳＳからＸＭＬへのマッピングの基本的な考え方を、図 2-27に示す。

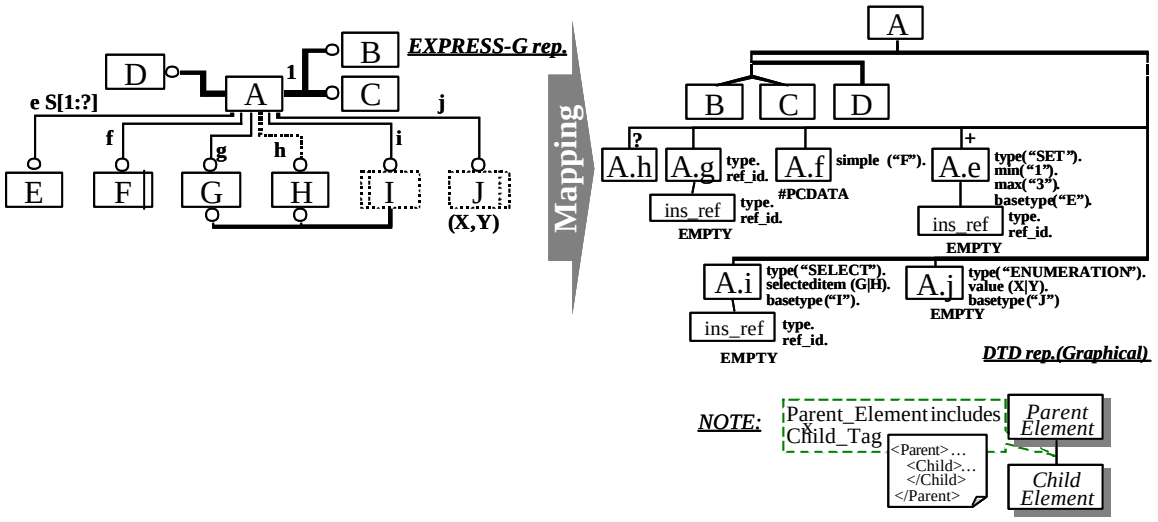


図 2-27 EXPRESSからXMLへのマッピング

(10) ＣＮＣデータモデルからＸＭＬ基準高レベルＮＣデータの生成方法

図 2-28にＥＸＰＲＥＳＳＣＮＣデータモデルからＸＭＬ基準高レベルＮＣデータ

を生成する過程を示す。ＣＮＣデータモデルをベースとしてマニュファクチャリングデータは、ＥＸＰＲＥＳＳスキーマとＥＸＰＲＥＳＳリポジトリから構成されている。ＥＸＰＲＥＳＳスキーマは、提案するマッピングルールを利用することで、ＸＭＬＤＴＤ（Document Type Declaration）にコンバートすることができる。

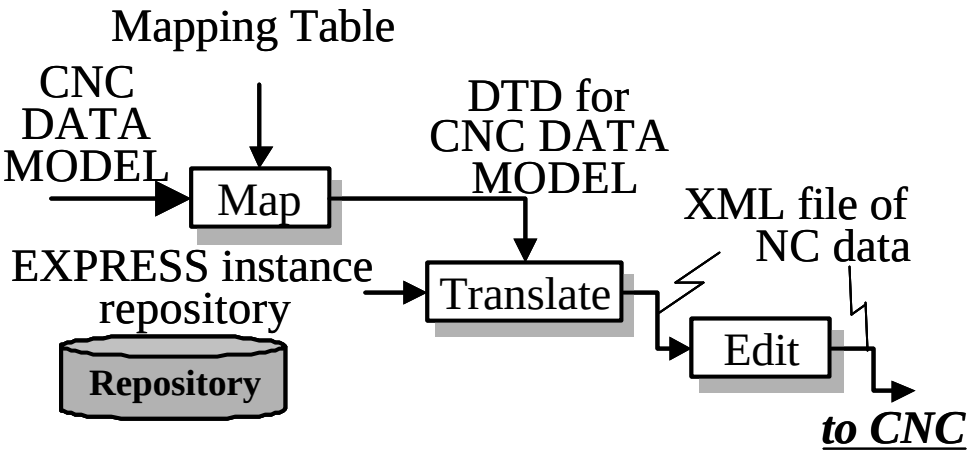


図 2-28 XML基準高レベルNCデータの生成方法

図 2-29にＣＮＣデータモデルのＥＸＰＲＥＳＳスキーマの例を示す。

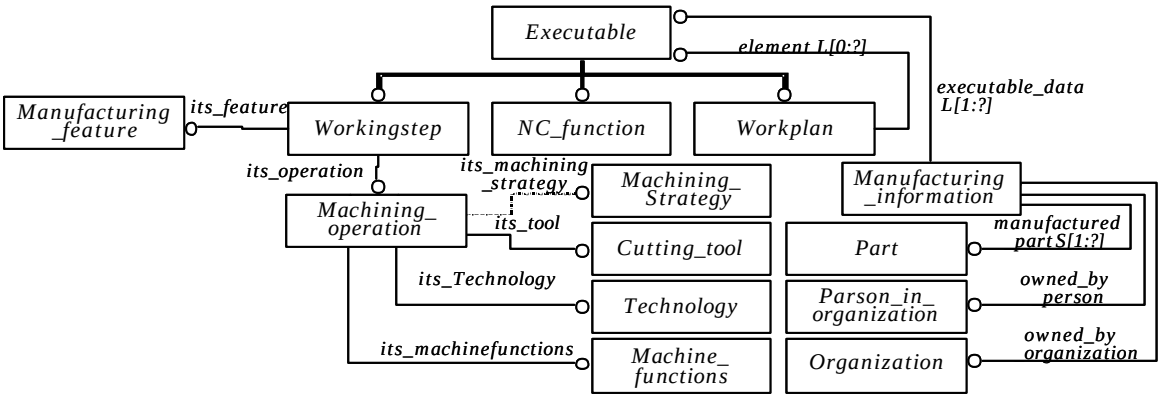


図 2-29 CNCデータモデルのEXPRESSスキーマの例

図 2-30のＣＮＣデータモデルのＥＸＰＲＥＳＳスキーマを、ＸＭＬＤＴＤにコンバートした例を図に示す。

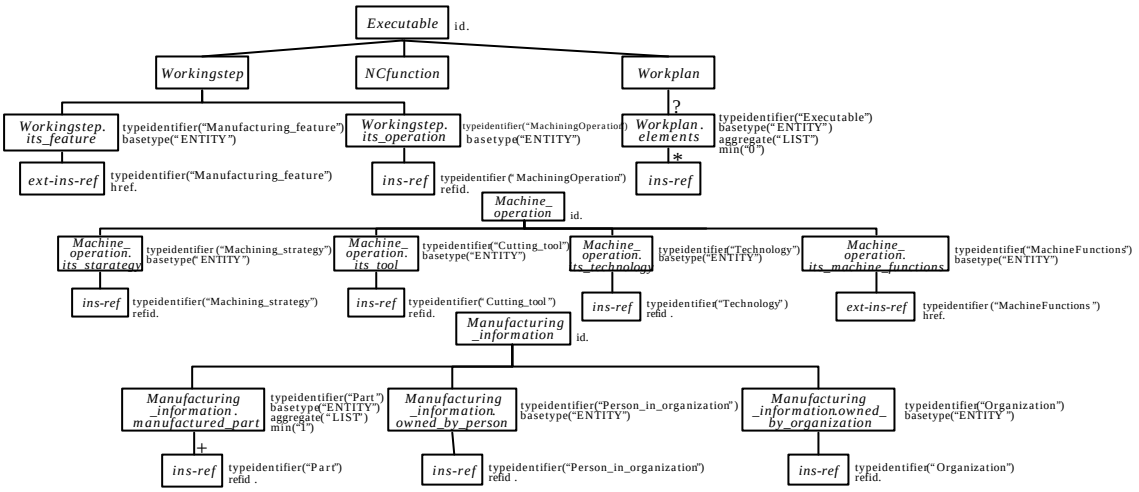


図 2-30 XML DTD にコンバートした例

(11) ＸＭＬ基準高レベルＮＣデータの例

以下のリストにＸＭＬをベースとした高レベルＮＣデータの例を示す。

```
<manufacturing_feature id="SLOT1">
 <manufacturing_feature.id type="SIMPLE">
  <String>SLOT1</String>
 </manufacturing_feature.id>
 <two5D_manufacturing_feature>
  <machining_feature>
   <machining_feature.manufacturing_data type="LIST" min="0"
basetype="two5D_workingstep">
    <ins_ref type="slot_rough_milling" refid="slot1_rough" />
   </machining_feature.manufacturing_data>
   <slot>
```

```
  <slot.course_of_travel>
   <ins_ref type="contour" refid="contour1" />
  </slot.course_of_travel>
   <slot.width>
<ins_ref type="length_measure" refid="length2" /> </slot.width>
  </slot>
 </machining_feature>
 </two5D_manufacturing_feature>
</manufacturing_feature>
<length_measure id="lengh2">
 <length_measure.theoretical_size type="REAL" >
  20.0 </length_measure.theoretical_size>
</length_measure>
<contour id="contour1">
 <contour.segments type="LIST" min="1" basetype="contour_type">
   <ins-ref type="contour_type" refid="poly1" />
 </contour.segments>
</contour>
<contour_type id="poly1">
 <ins_ref type="polyline" refid="polyline1" />
</contour_type>
<workingstep id="slot1_rough">
 <milling_workingstep>
  <two5D_workingstep />
 </milling_workingstep>
</workingstep>
```

(12) ＸＭＬ基準高レベルＮＣデータ編集のユーザインターフェース

図 2-31にＸＭＬをベースとした高レベルＮＣデータを編集する際の、ユーザインターフェースの例を示す。ＸＭＬに対応したＷｅｂブラウザにより、階層構造として表示することができ、検索、編集が可能になる。

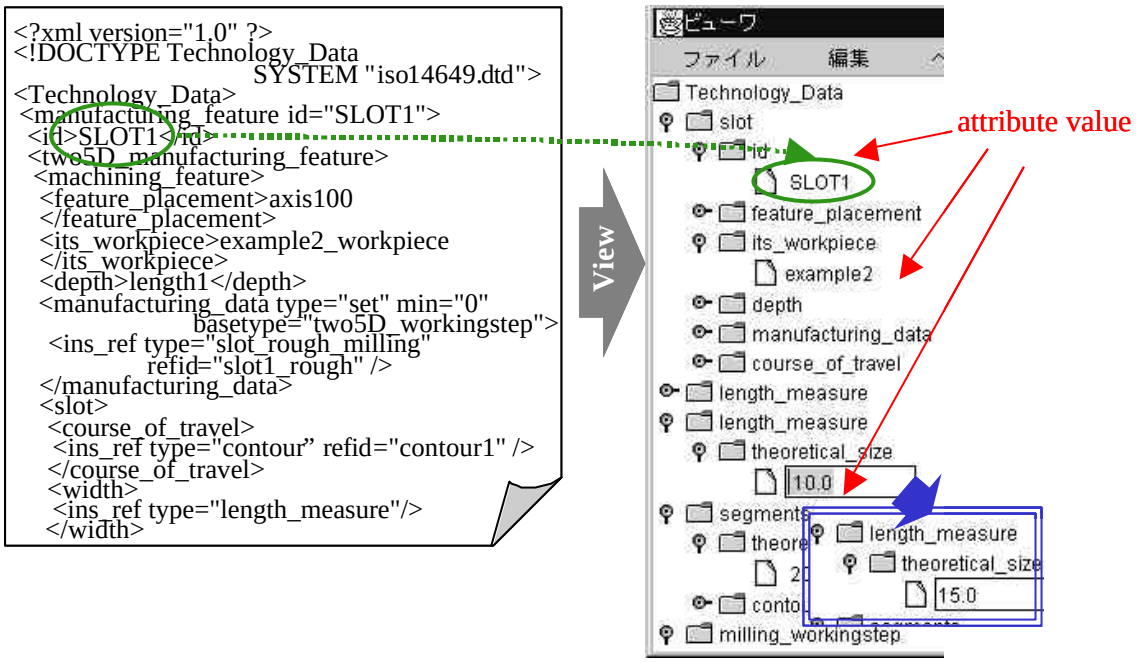


図 2-31 XML 基準高レベル NC データ編集のユーザインターフェース

(13) まとめ

ここでは、ＩＳＯ１４６４９で定義されているＥＸＰＲＥＳＳＣＮＣデータモデルの本質をとらえ、ＥＸＰＲＥＳＳデータモデルの連続的なフォーマットを生成し、その有用性を評価するためにＸＭＬのＥＸＰＲＥＳＳへのアーリーバインドを提案した。これにより、以下のことが言える。

- 製造設備の性能、操作の品質、設備の生産性をリモートで評価するために利用することができるＸＭＬをベースとした高レベルＮＣデータを提案した。
- ＸＭＬをベースとした高レベルＮＣデータは、素早く効果的に操作間のデータを変換し、顧客とパートナーとの、効果的で確実な取引と協力を可能とし、ラピッドプロトタイピング、バーチャルプロトタイピング、シミュレーション、製品試験を実現し、最適なマニュファクチャリングツール、知識ベース、製品情報、場所に依存しないトレーニングを短時間で容易に作成することができる。

2.8　デジタルマニュファクチャリングによる加工作業例

製品モデルで設計から加工作業までを統合したデジタルマニュファクチャリングの作業に関して詳細に検討を行なう。

図 2-32は現状のＣＡＤ／ＣＡＭシステム例で、設計工程のＣＡＤシステムで作られた製品データと加工データを作る生産準備工程のＣＡＭシステムの情報統合ができていない場合の業務例である。

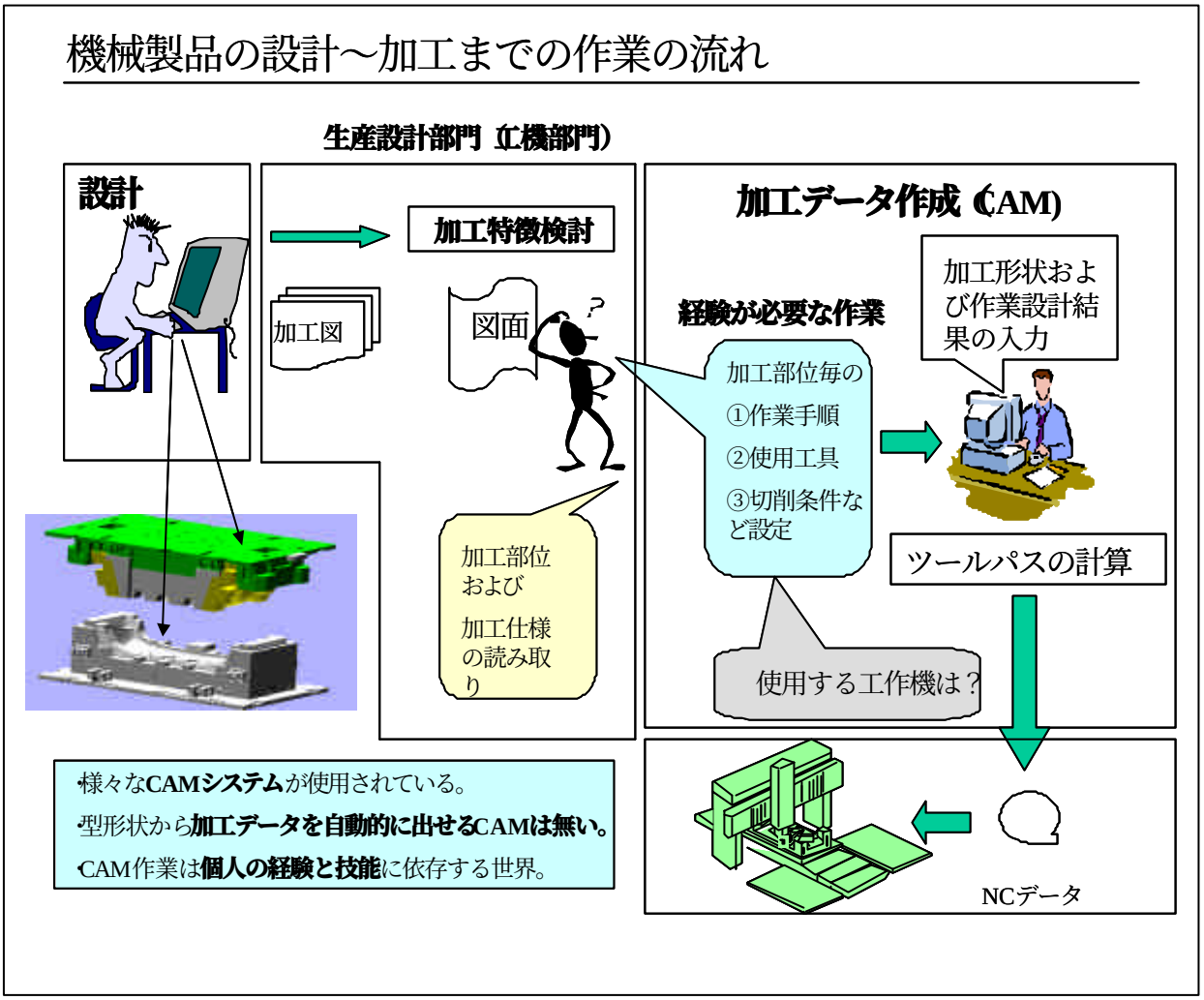


図 2-32 機械部品の設けから加工までの作業の流れ

生産設計技術者は設計で作られた部品の製品情報を、ＮＣ工作機で加工するための加工情報に変換するために、幾何形状、設計仕様を見ながら一つ一つの幾何情報を加工特徴に変換する必要がある。例えば、この穴はスロット、ポケット、ホールのいずれに相当するのか。また加工する場合の工具の選定は、工具の回転数や加工順番に関する条件は、切削油の有無は等、今までの知識と経験を最大限に生かしながら工作機械を使い材料から製品を削り出すための加工情報をＣＡＭに入力する必要がある。この作業は簡単な形状であっても経験と技術が要求される作業である。

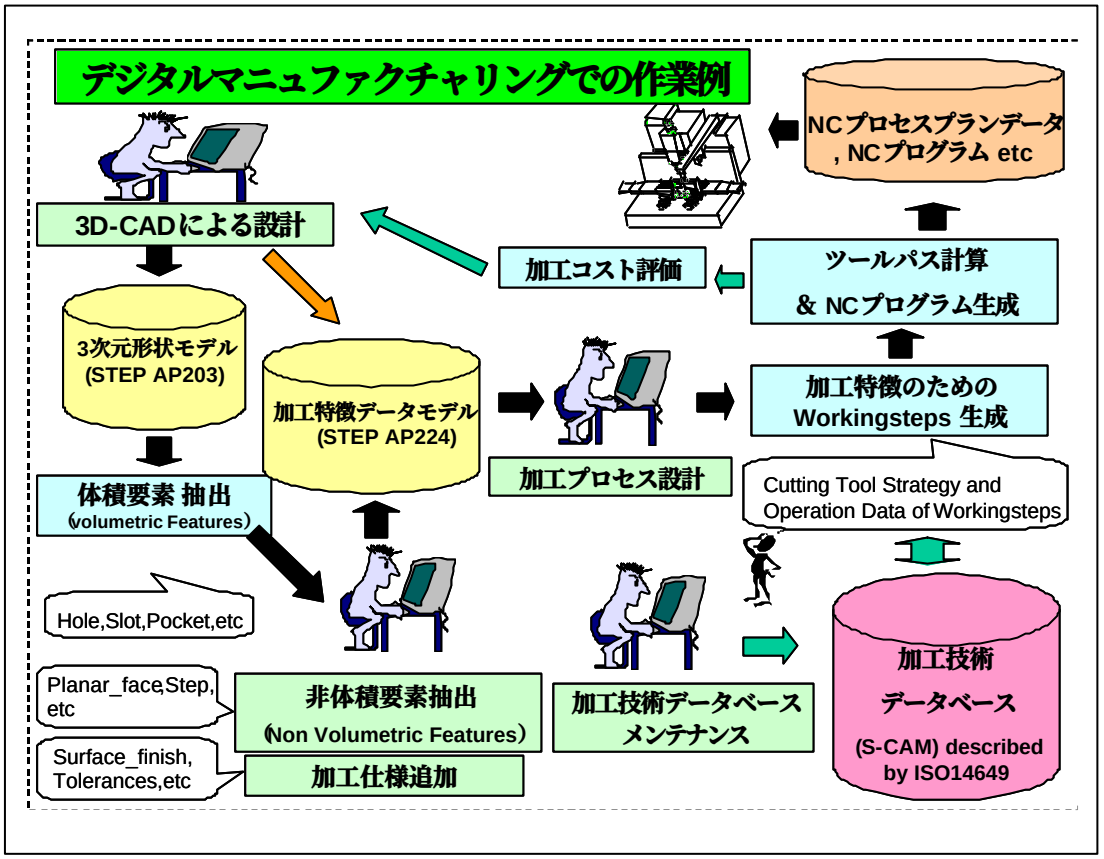


図 2-33 デジタルマニュファクチャリングでの作業例

図 2-33は製品モデルを使いＣＡＤ／ＣＡＭシステムの生産情報の統合を行なった時の例である。

設計工程で作られた３次元ソリッド形状情報と製品仕様を使い加工特徴の認識を行なわせる。

加工特徴の形状からの変換はＡＰ２２４で形状と加工特徴の関係が分類されているから、その情報を基にして、ボス／スロット／ホール等の加工特徴を形状情報から抽出する。また、工作機械の加工法を決定するために、あらかじめ加工特徴毎に準備された工具・加工順・機械作業条件の加工技術データベースを利用して加工特徴毎に作業条件を決定していく。最終的には形状情報と加工特徴からの作業設計情報からツールパスと切削作業条件が決められてＮＣ装置のプロセスプランデータとＮＣプログラムが作成される。

また加工特徴自動認識と加工技術データベースは加工特徴の標準化ができなければ機械毎に設定し直す必要があり標準化が重要なキーになる。

(1) 生産設計作業と情報の流れ

生産設計から加工までの業務分析を行なうと図 2-34のようになる。

ここでは設計工程で作られた製品モデル情報を入力して、工程設計を行ないＮＣプログラムを作成して材料からＮＣ工作機械を使い最終製品を削り出す作業である。

主な工程は４つの業務プロセスから構成されている。

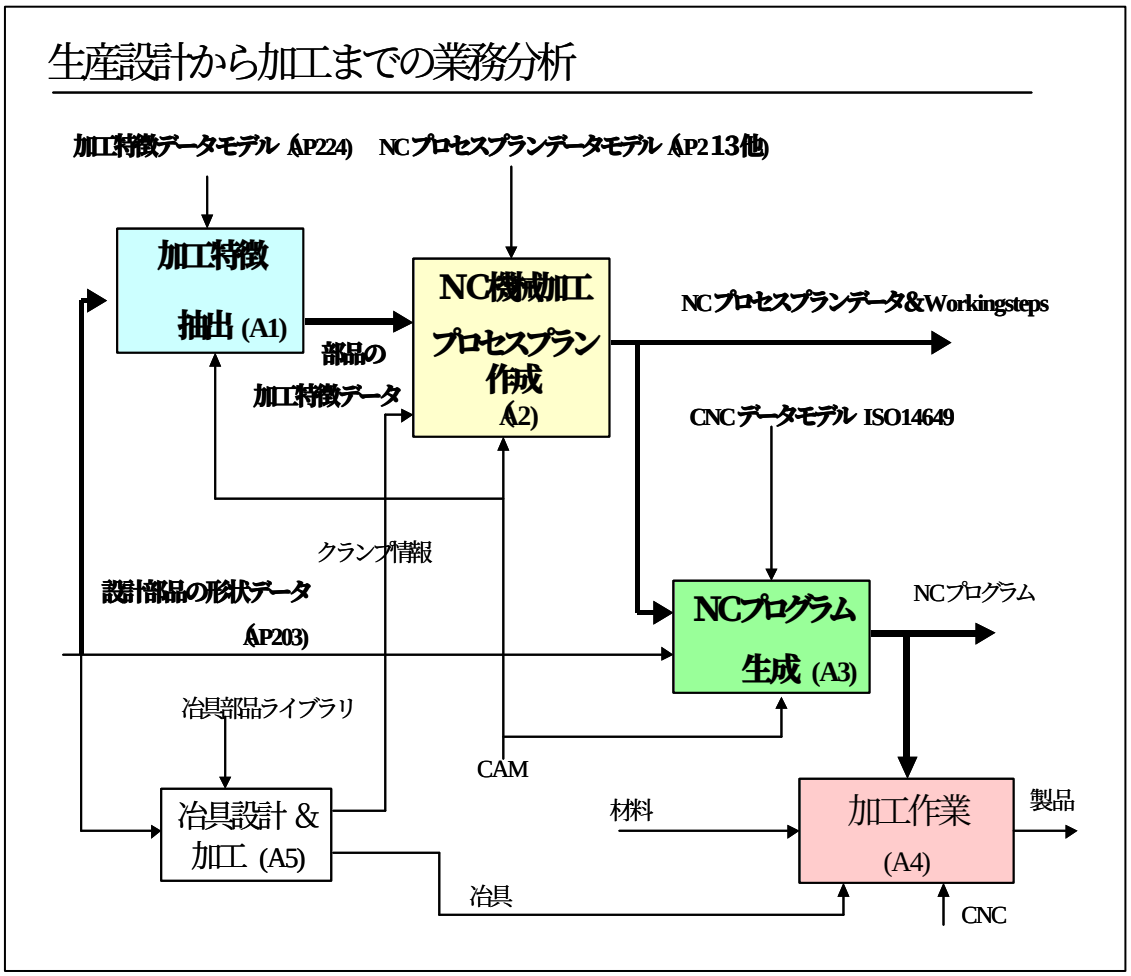


図 2-34 生産設けから加工までの業務分析

- 加工特徴の抽出（Ａ１）：設計工程で作られた製品モデルの「形状情報」から標準の加工特徴を参照して、製品モデルにボス／スロット／ホール等の「加工特徴」を付加していくプロセス。
- 加工のための工程設計（Ａ２）：加工特徴毎に加工のための工程設計を行なう。

この工程を分析すると図のようになる。ここでは加工特徴毎に分類され（図 2-36）、加工のための工程設計を行ない（図 2-37）、加工工程毎に工作機械の加工動作を決めて行く（図 2-38）、一つの工具による一つの加工動作は Workingsteps と定義されている。すなわち、Ａ２４では工程毎の Workingsteps を決定するものである。

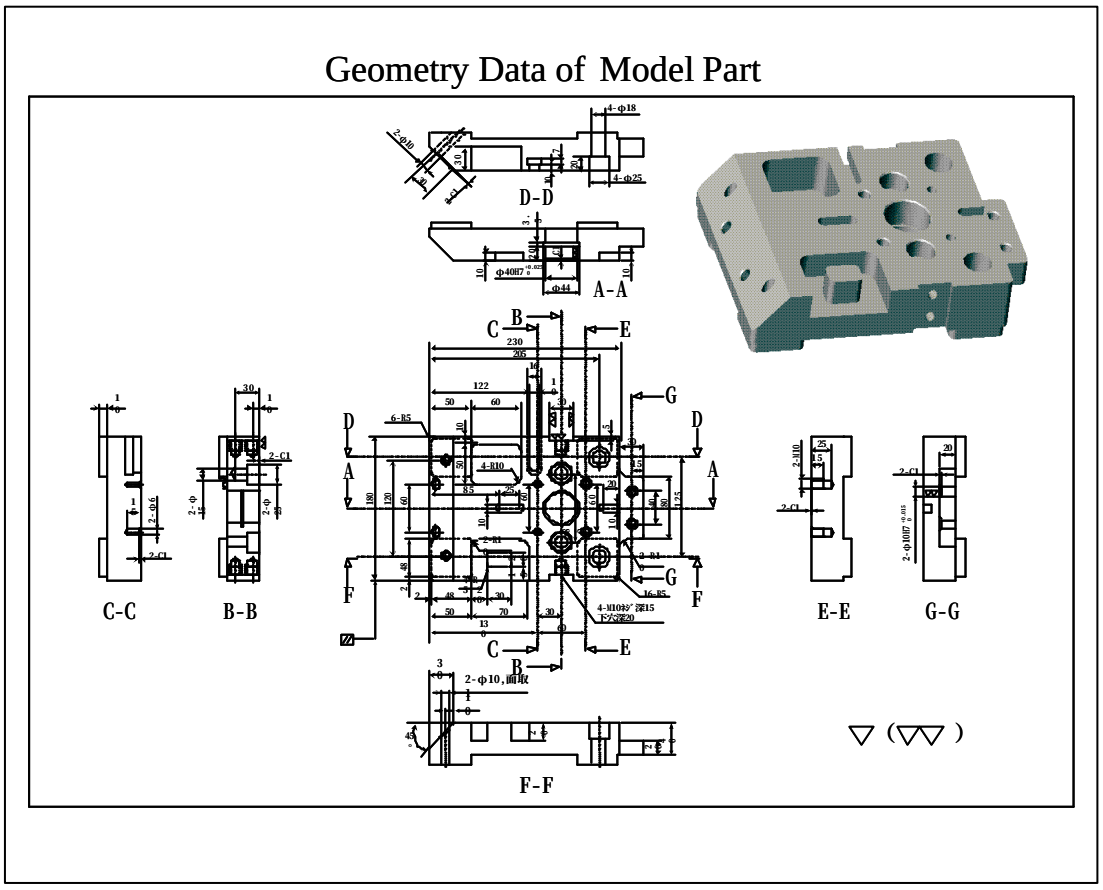


図 2-35 加工対象例

- ＮＣプログラムの生成（Ａ３）：Workingsteps 毎のＮＣ工作機械のＮＣプログラムを生成する。
- 部品の機械加工（Ａ４）：ＮＣプログラムで実際に加工する。

(2) 加工特徴抽出の例

ここでは、加工特徴抽出の仕組みを詳細に例示する。図 2-35は設計工程で作られた

部品の設計データである。この設計データをＡＰ２２４の加工特徴で分類すると図2-36のようになる。この例では加工の際の部品取り付け方向もあわせて決めていて、Ａ／Ｂ／Ｃ／Ｄ／Ｅが抽出された加工特徴の方向である。各加工特徴はラベル付けされて、ＡＰ２２４の加工特徴を参考にした分類が行なわれている。例えば[Hole_1]は方向がＡの２つの Countersunk_hole から形成されている。

[Hole_3]では方向がＡで２つの加工特徴から構成された Compound_feature であり、その加工特徴は Round_hole と Counterbore_hole である。

このように、部品形状幾何情報からＡＰ２２４を参考にして加工特徴を抽出する事ができる。

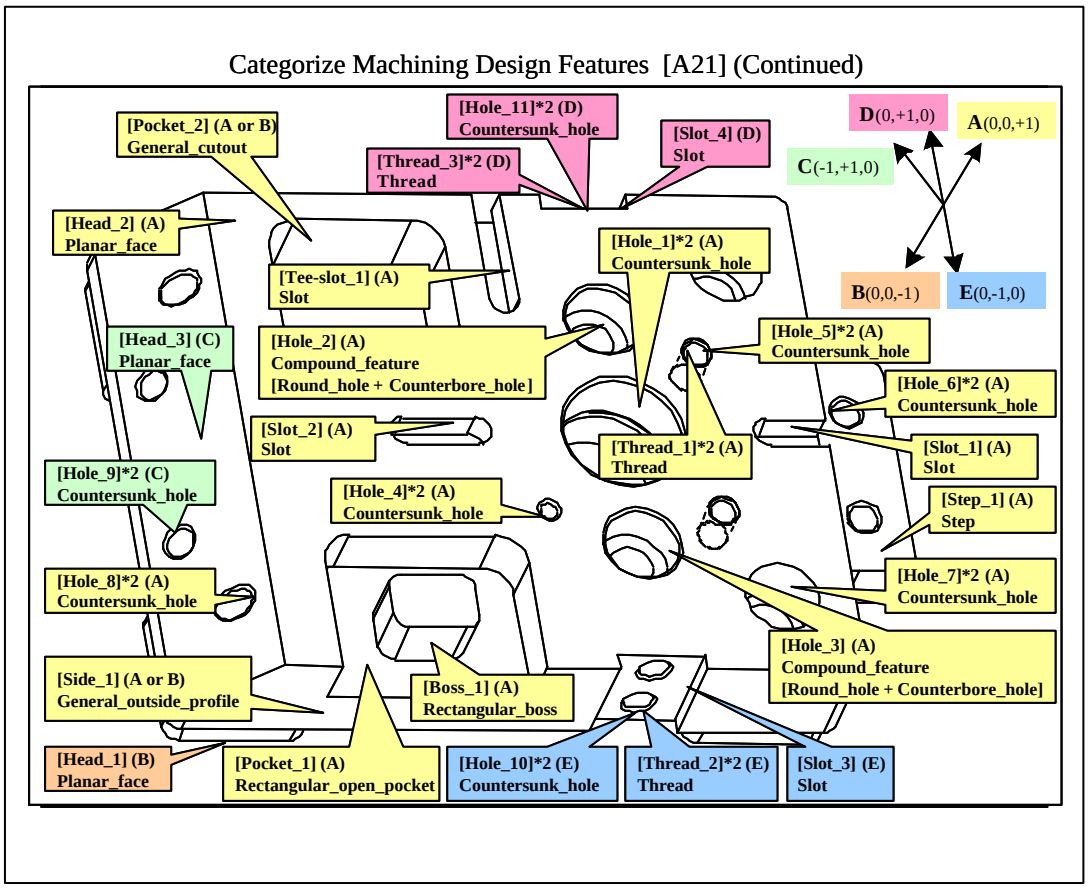


図 2-36 加工特徴

(3) 加工工程設計の例

加工工程設計は４つの工程に分解される。

- Create Strategy of Process Plan
- Modify Machining Design Feature
- Assign Machining Design Feature to Each Process
- Generate Machining Process Feature

ここでは、材料のクランプ、（２）で抽出した加工特徴を工作機械で作業するための加工特徴の変換、各加工特徴の加工順を定める。

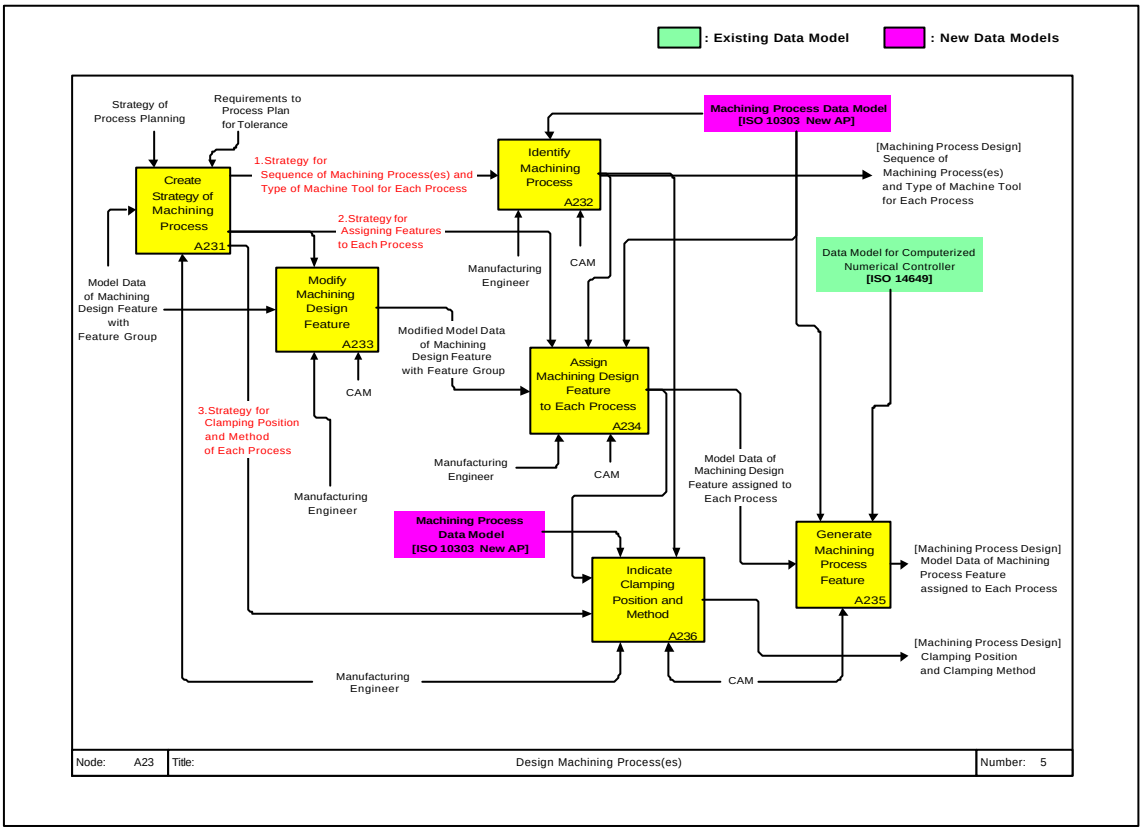


図 2-37 加工プロセス

(4) Workingstep の生成

この工程も４つの作業工程に別れる。

- Determine Actual Machine Tool for Each Process
- Design Workingsteps and Generate Cutting Tool Strategy for Feature of

Each Process

- Determine Sequence of Workingsteps for Each Process
- Determine Operation Data of Workingsteps for Each Process

この工程では（３）で得られた加工特徴毎にＮＣ装置の加工作業である Workingsteps **を決定する。（図** 2-37**、図** 2-38**参照）**

Workingsteps **は作業タイプ**(Operation Type)**、切削法**（Cutting Tool Strategy）**、工具**(Tool Body)**等の工具の動作**(Tool Path Strategy)**を決める事で一つの加工作業が決まる。**

例えば、Head_2 **でラベル付けされた加工特徴は** Head_2_rough,Head_2_Finish **の２つの** Workingsteps **より構成される。**

Head_2_Rough **でラベル付けされた** Workingsteps **は工具は** Facemill **を使い** Plane_rough_milling **作業を行なう。**

このように、加工特徴毎に切削するための Workingsteps **をこの工程では決定する。**

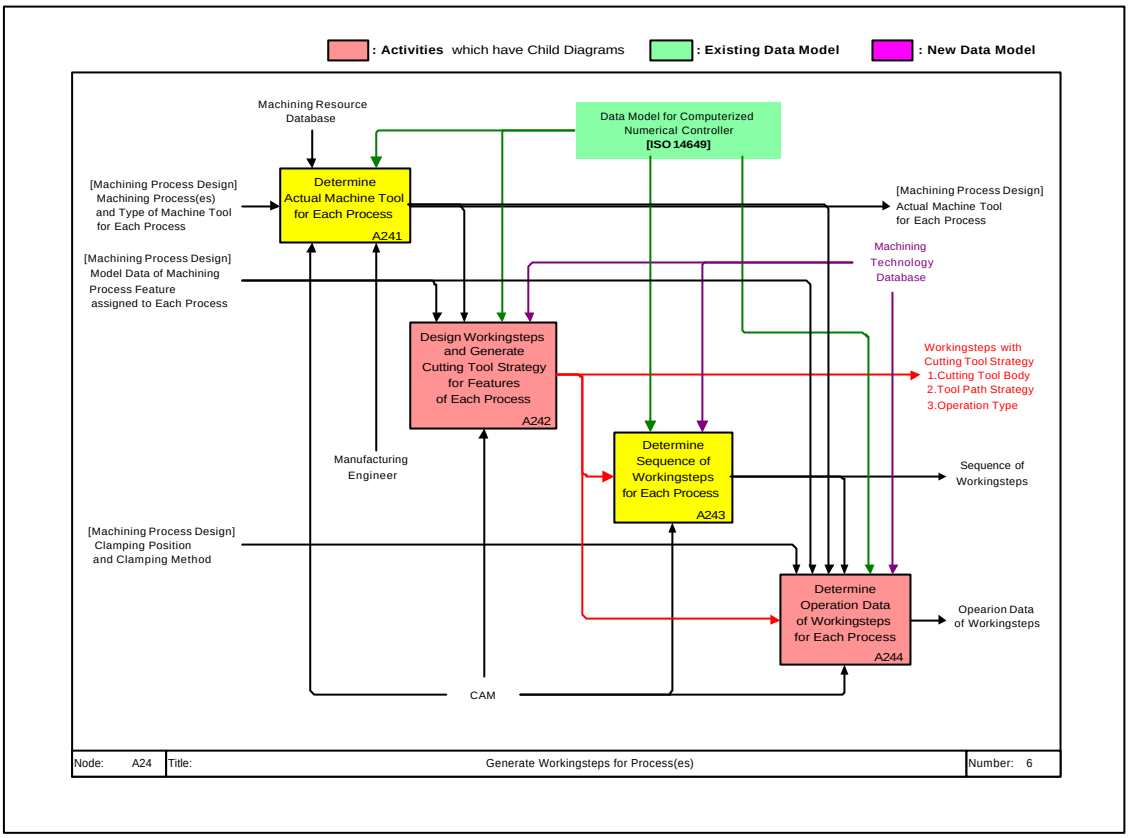


図 2-38 Workingstep の生成

以上具体的な製品モデルデータを加工するための情報の流れと加工動作に関して細かく例示した。ここでは、情報の加工されていく様子とこのための必要なデータの標準化に関して検討する。

まず、設計工程で作られた製品モデルデータは様々なＣＡＤシステムで作られるためにＣＡＤ固有のデータ形式であれば次の工程に入力する際に変換が必要となる。従って、設計された製品モデル表現はＳＴＥＰのようなニュートラルフォーマット形式である必要がある。

次に、幾何形状から加工特徴を抽出する際にも標準的な幾何形状と加工特徴の対応関係が必要になる。従ってＡＰ２２４のような加工特徴の標準化は必要である。加工プロセスの工程設計も設備毎にそのモデルが異なれば特定な設備に対しての工程設計に制限されるために標準化は重要な課題である。また、加工特徴に対する Workingsteps を導く際にも標準的な加工技術データベースが必要となる。

以上のように製品モデルによる設計から加工工程までの情報統合のためにはＳＴＥＰのような統一が取れた標準的なデータモデルが必要となる。

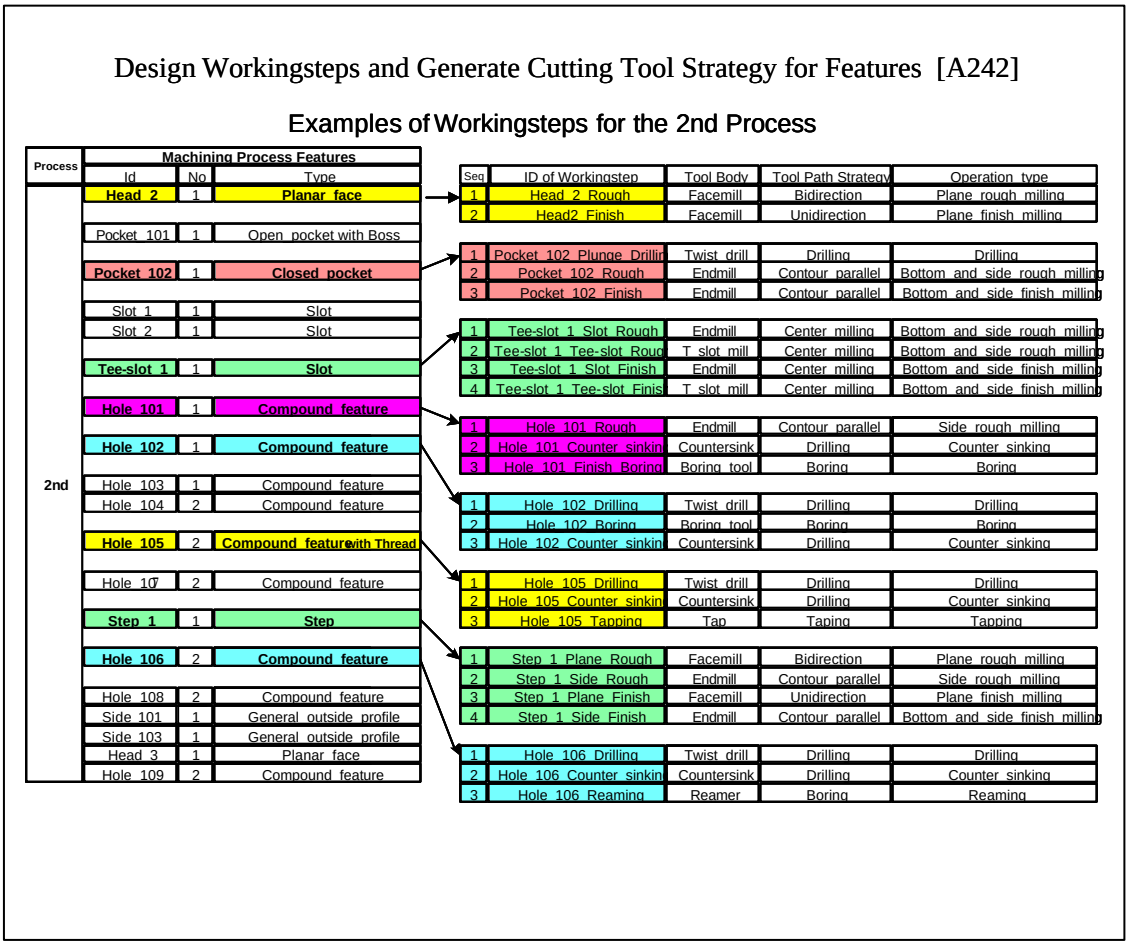


図 2-39 導出された加工作業

2.9 デジタルマニュファクチャリング実現へ向けた海外の取り組み

海外ではデジタルマニュファクチャリング実現に向けて２つの代表的なプロジェクトが取り組まれている。

- 軍の備品を長期間、低コストで供給させる事を目的としたＲＡＭＰプロジェクト
- デジタルマニュファクチャリングを実現してＧコードの世界から決別する、及びインターネット上でマニュファクチャリングデータを共有して世界的な分散生産プロセスシステムを実現しようというＳＴＥＰ＿ＮＣプロジェクト

ここではＳＴＥＰ＿ＮＣの取り組みに関して調査報告を行なう。

(1) ＳＴＥＰ＿ＮＣとは

このプロジェクトの公式の名前は the Model Driven Intelligent Control of Manufacturing であるが、簡単に「スーパーモデルプロジェクト」と呼ばれている。資金は米国標準技術院（ＮＩＳＴ）、米国商務省の Technology Administration から提供されて、1999 年 10 月に Advanced Technology Project として 290 万ドルかけて正式に立ち上げられた。このプロジェクトへの参加は、メーカー、ソフトウェアベンダー、マシンコントロール製造業者、政府及び軍、さらにはニューヨークのハドソンバレーから参加した中小企業から構成されている。

プロジェクトの開発期間として ３年で計画されている。初年度の目標は、３種類のマニュファクチャリングフィーチャを含むＳＴＥＰ部品モデルとＳＴＥＰ＿ＮＣデータベースを構築して、そのデータベースを使用して工作機コントローラを運転することである。２年目の目標は、ＳＴＥＰ＿ＮＣフライス盤によって定義される加工フィーチャのすべてを含むデータベースを構築して、そのデータベースを使用してＳＴＥＰ＿ＮＣテスト部品を製造する。３年目の目標はターニング、グラインディングあるいは放電加工など別の機械加工の過程に関するデータベースを構築することである。

このＳＴＥＰ＿ＮＣプロジェクトでは 3 次元製品モデルから工作機械の加工プログラムに直接変換する技術とインターネットを介した製品モデルデータの利用技術の開発を目指している。これは「どこでも」製品モデルデータへアクセスを許可され、充分整備された工作機械がある工場であればインターネット上の製品モデルから直接部品が製作できることを意味している。すなわち、このような世界中からアクセスできるインターネットを利用したデジタルマニュファクチャリングシステムが完成すれば、デジタル製品モデルは普遍的な「ＮＣ部品」として利用できるようになり、世界的な部品製作と供給チェーンが確立でき、必要な時に世界中の工場で同じ品質の機械部品を加工して入手することができるようになる。

ＳＴＥＰ＿ＮＣの実現する世界として以下のようなシナリオが準備されている。

「工作機械の入力はあまり遠くない将来、入力に必要なマシンツールはウェブページだけになるであろう。そのことで、部品加工作業は次のように変わるであろう。

まず、工作機にＰＣベースのＣＮＣへウェブブラウザを呼び出す。そして一つのウェブサイトを開く。ホームページのメニューから、一つの部品データベースを選択す

る。

そこでは、ワークピースの 3 次元イメージができあがっている。タスクバーのアイコンをクリックして、ポップアップウィンドウでパラメーターとデフォルト値をチェックして確認後に工作機械のＣＹＣＬＥＳＴＡＲＴボタンをクリックする。スピンドルモータはうなり音をあげて始動し軸が動き始め、冷却剤がほとばしり、瞬時完成したチップはすぐに作業台上で弾み始めるだろう。

現在進行中の努力の効果により、数年先にはこのシナリオのように、ほとんどのショップがこのような工作機械を動かすことになるだろう。ＮＣプログラムは、古めかしいものになる。全ての工作機コントローラが必要とするものは、ウェブページの 3 次元イメージによって表現されるデジタル製品モデルになるだろう。」

このＳＴＥＰ＿ＮＣでは、ＩＳＯ１４６４９ＣＮＣデータモデルを使用する。このモデルでは工作機械をＧコード無しに制御する事を目指すものである。カッティングツールの動かし方は、製品モデルデータベースの中に全て含まれる。今までのようにツールパスデータを、別々のファイルとして作成する必要はなくなると考えられる。ツールパスは、製品モデルに基づいてＣＮＣ自身で生成されることになるだろう。それは、ポストプロセッサが必要なくなるということを意味する。データは実行のためにＣＮＣの中のマシンにより、フォーマットされる。そして、製品モデルが変化しないので、いつでもどこでも必要な時に、機械加工のハードコピーを得ることができるようになる。

またＳＴＥＰ＿ＮＣの製品モデルはＳＴＥＰによる製品モデルを前提としているから、今までのＣＡＤシステム固有のレガシーな製品モデルデータのようにＣＡＤシステムライフサイクルに制約される事なく、その製品モデルの利用は半永久的に少なくとも製品と同じ程度の長期間活用ができて、製品ライフサイクルを充分サポートできるものとなることを意味する。

例えば、航空宇宙の部品の典型的な寿命は２５年である。この間コンピュータ技術における変化や工作機技術における進歩も、工作機入力として製品モデルのユーザビリティに影響しないと考えられる。

(2) ＳＴＥＰ＿ＮＣを実現する技術について

ＳＴＥＰ＿ＮＣを実現する技術は以下の 3 点である。

- ＳＴＥＰ製品モデル（ＳＴＥＰによる製品モデル表現）

- ＣＮＣデータモデル
- ＸＭＬによる製品データ共有

それぞれの技術的概要とＳＴＥＰ__ＮＣ実現のための課題をまとめる。

① ＳＴＥＰ製品モデル

ＳＴＥＰは 製品モデル情報を表現して交換／共有する方法に関するＩＳＯ規格（ＩＳＯ１０３０３）であり製品に関するすべての不可欠の情報をやりとりすることができるように設計されている。（図 2-40参照）

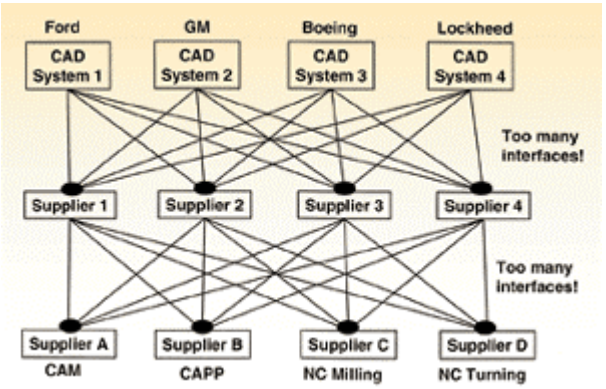


図 2-40 製品情報ネットワーク (複数のCADとサプライヤ)

1984 年に国際標準作成団体であるＩＳＯの主催のもとに Product Data Exchange Specification が設立された。目標はコンピュータで解釈可能な製品データモデルを作成するための方法を定義することであった。これらのモデルは、製品仕様のデータが製品ライフサイクル中で変化しない方法を使用してデータ交換と製品データの共有ができる事を目指した。この製品データモデルをカバーする世界規格がＳＴＥＰとして知られるようになっている。開発着手から 15 年間、様々なワーキンググループと委員会が製品データモデルの規格を開発するために定期的に会合をもってきた。

現在ではＣＡＤによる製品モデル仕様に関してＳＴＥＰ規格はＣＡＤデータ交換の手法として知られているＩＧＥＳの機能を超えるレベルまで開発されてきたので、ＩＧＥＳに代わって今後はＳＴＥＰが公式にその役割が位置づけられている。

特に、現在では大手及び中小のＣＡＤベンダーは、ほとんどの自社の最新ＣＡＤ製品に、ＳＴＥＰトランスレータを装備している。また、これらのＳＴＥＰトランスレータは、コンフォーマンスとインターオペラビリティについてテストが行われている。中には有効に機能しないトランスレータもあったが改善されて来た。ＳＴＥＰ構成課程の革新的な特徴の１つは、ＳＴＥＰ対応システムが意図されたとおりに機能するか保証する手順が、初期のうちから準備してあったことである。この方法により開発は遅くなったかもしれないが、結局はうまくいっているように見える。要するに、ＳＴＥＰは現在では設計段階の製品モデルデータ開発のために活用されている。レンセラ大学 Martin Hardwick 教授は、現在 100 万以上のＳＴＥＰ利用可能なＣＡＤステーションが世界中で利用されていると報告している。

② 「スーパーモデル」と「ＳＴＥＰ＿ＮＣ」

スーパーモデルは「ワーキングステップ」、すなわちＣＮＣ工作機で実行される特定の加工作業のライブラリを定義する。（図 2-41参照）

ＳＴＥＰの製品モデル表現とコンセプトを保つための「ワーキングステップ」の概念は製品モデルをＣＮＣが実行する加工動作単位として製品モデルに組み入れられた一般的な概念である。これらの記述は特定のプログラム形式やコードにリンクされるものではない。しかしワーキングステップは伝統的なＭコードとＧコードによって表される機械加工コマンドとほぼ同等の機能を持つ。

スーパーモデルは、工作機入力として製品モデルを使用するために基礎となり得るデータモデルである。スーパーモデルは、機械加工情報の完全なデータベースが構築される必要がある。そのデータベースには、部品を切削するための工作機コントローラにはどのような能力が存在しなければならないかが含まれる。

このように「スーパーモデル」は、ジオメトリのような形状を表わす設計情報とフォームフィーチャー（穴、スロット、輪郭など）のような加工をするためのマニュファクチャリングプランニング情報を含み、ツールの選択や固定具の位置など加工方針の情報が追加されたものである。

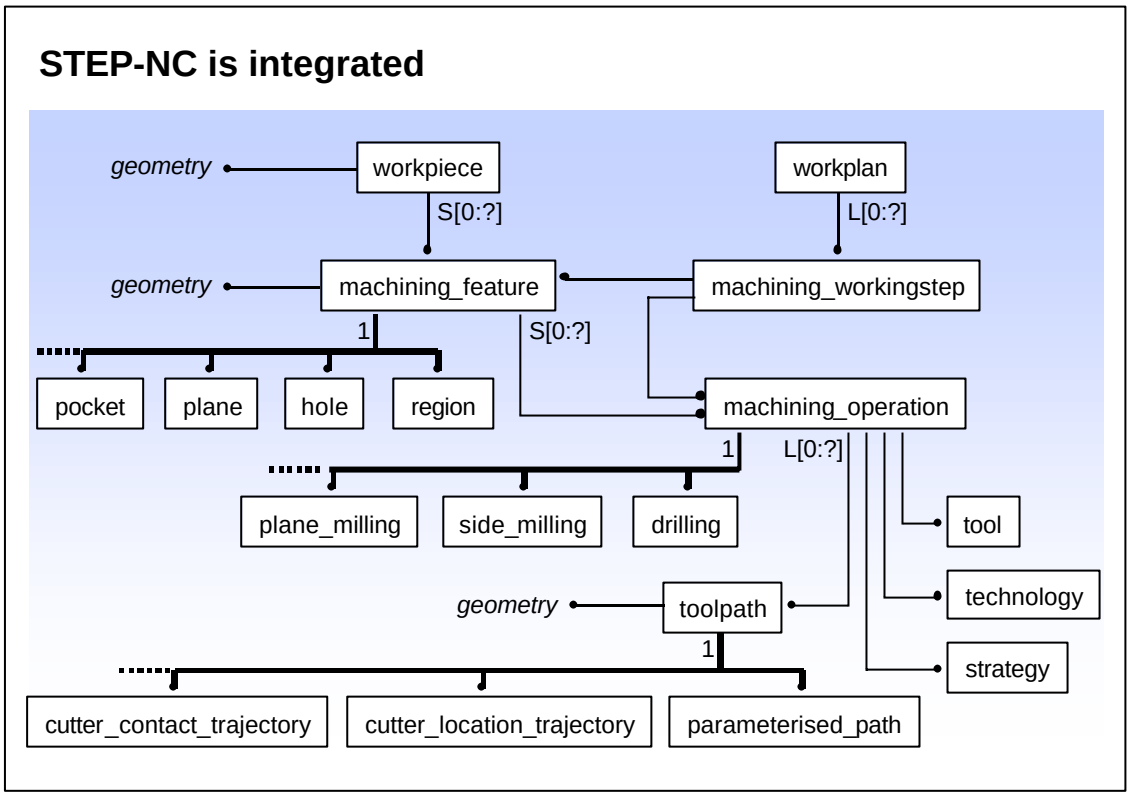


図 2-41 CNCデータモデル

スーパーモデルでは加工対象の部品(workpiece)を加工特徴(machining_feature)に分解して各加工特徴毎に加工動作（machining_workingstep）を定める。ワーキングステップは工作機械の加工作業の動作定義（machining_operation）と対応付けられている。すなわち、工具(tool)、穴あけ(drilling)や平面切削作業(plane_milling)などの工作機械の加工動作、加工仕様、工具の動き（toolpath）を定める。

The Super Model Database スーパーモデルプロジェクトは、ワークピースのＳＴＥＰ製品モデル表現とＳＴＥＰＮＣによって定義された加工情報を集める必要がある。製品モデルの製品幾何情報は、ＡＰ２０３のような一つのＳＴＥＰアプリケーションプロトコルにより定義することができる。

加工特徴は、プログラムの一員であるハネウェルＦＭ&Ｔによって開発されるフィーチャ自動認識システムから抽出される。FBMach Process Planning System（FBMach はfeature-based machiningの短縮形)と呼ばれるこのソフトウェアは、ＳＴＥＰ製品モデルのジオメトリを読んで、穴などの加工フィーチャ(ポケット、ス

ロットなど）に自動的に翻訳することができる。FBMach システムの実用的開発は、2000 年の終わりまでに期待されている。2000 年 11 月にはスーパーモデルテスト部品への応用のデモに FBMach システムが組込んでデモがなされた。

STEP Tools 社は、workingsteps、workingstep-methods、workingstep actions、及びマシン フィーチャをマッチングするためのテーブルをセットアップしている。

このスーパーモデルデータベースは、データベースを構造化するために、同社の ST Repository 製品データ管理ソフトウェアの最新バージョンに適合させている。各リポジトリは、ＣＡＤ、ＣＡＭエンジニアによって利用されているツールと、ジオメトリ、フィーチャ、ワーキングステップをやりとりする標準的なインターフェースを使用している。

ＳＴＥＰの規格は、特定の産業に必要とされる製品モデルのために、カスタマイズされた様々な「アプリケーションプロトコル」とよばれる製品データモデルの集合が準備されている。これらのアプリケーションプロトコルは共通の幾何学、トポロジー、公差、関係、属性、アセンブリ、構成及び他の製品特性をカバーしたリソースモデルにより、共通のフレームワークの上に作られている。アプリケーションプロトコルは必要に応じて、 新しい製品モデルとして加えることができる。

現在、スーパーモデルに関連する製品情報をカバーするために、ＳＴＥＰのアプリケーションプロトコルへ追加が行なわれようとしている。これが「ＳＴＥＰ＿ＮＣ」である。ＳＴＥＰ＿ＮＣは、シナリオで述べた世界を実現するための基礎技術を形成する。ＳＴＥＰ＿ＮＣ製品モデルデータを、ダイレクトな工作機入力として使用可能にする開発は、既に着手されている。

2000 年 5 月に、テスト部品の製品モデルについて加工情報を追加するために必要とされるデータのプロトタイプが示された。プロジェクトの後半では、この「スーパーモデル」が使用できる工作機コントローラが開発される計画である。現在テスト部品のデータモデルとしてミリング・ワークピースが準備されている。ターニングとグラインディング用のものも準備中である。5 月に示された「スーパーモデル」の例では、機械加工方法、ツールパス・プランニング、ツール選択に関する情報をネットワーク経由で共有するために、ＸＭＬを使用したグローバルな e-manufacturing に必要とされるデータベースを生成することまで実証された。

③　ＸＭＬによる製品データ共有

スーパーモデルデータベースへのＳＴＥＰ Tools 社のアプローチの主要な部分は、そのインターフェースにおけるＸＭＬの使用である。ＸＭＬ（eXtensible Markup Language）は、インターネットを介してデータだけではなく、情報を交換することができるタグで構造化された言語である。ＸＭＬは、データベースを読むソフトウェアアプリケーションが、「タグ」を解釈することでデータベースに格納されている情報のタイプが何であるのかを確認できる。

ＨＴＭＬ（Hyper Text Markup Language）は、どんなインターネットブラウザでもテキストを表示できるようにするWWWのための言語である。ＸＭＬはそれ以上のレベル機能とインターオペラビリティを提供する。ＳＴＥＰを実装するためのＸＭＬ規格は完成に近づいている。この規格は、製品モデルのすべてのデータが同じ方法により「タグ付けされること」を確実にするだろう。

スーパーモデルのために、ＸＭＬは、加工法、ツールパス、及びツール選択情報を、ジオメトリ、フィーチャ及びマシニング ステップにリンクする便利な方法をデータベースに提供する。

適切なタグがあるデータを整理することによって、例えば、ドリルであけられる穴として決められたジオメトリは、ラフ取りリング、ボーリング、カウンターボーリングといった段階に作業命令にリンクされる。また、それぞれの段階は、ワークピースの材料、表面処理の必要条件などなどのように抜粋され、速度や供給の表に対応されることになる。ＸＭＬはデータが正しく分類されるためにタグを提供する。

ＸＭＬにより、インターネットのネットワークにつながれているＣＮＣが、製品モデルデータベースから部品を機械加工するために必要な情報を見つけることができる。

2000 年 5 月の Industrial Review Board ミーティングでは、STEP Tools 社は穴、スロット及びポケットを含む３つの異なったタイプのマシンフィーチャに必要な情報をリンクして、ＸＭＬトランザクションがそのテスト部品に関するデータベースを作り上げるためにどのように使用されたのかを示した。

(3) ＳＴＥＰ＿ＮＣの実証

① Down To The ＣＮＣ

Super Model プロジェクトの目標は、製品モデルを使用して、マシニングセンターでＳＴＥＰ＿ＮＣデータにより表現された加工法にしたがって部品を切削するこ

とができるか示すことである。（図 2-42）

この実証段階で深くかかわるメンバーのひとつが、ペンシルバニアの Electro-Mechanical Integrators、Inc.（ＥＭＩ）of Franconia である。ＥＭＩの技術者は、ＳＴＥＰ＿ＮＣデータを受け入れることを可能にするブリッジポート制御装置の新しいソフトウェアを作成している。

図 2-42 ブリッジポートV3XTミリングマシン

- ブリッジポートＶ３ＸＴミリングマシンは、スーパーモデルデータベースの情報を利用して部品を機械加工することができる。

ＥＭＩではＳＴＥＰ＿ＮＣを実行するためのカスタマイズ コマンドパーサと、コマンド インターポレータを開発している。この開発ではリアルタイムで製品モデルの情報を解釈するための新しいＣＮＣプロトコルを作成している。このソフトウェアは、軸の移動を決定して指定されたツールを選択し、コマンドを発行するために必要とするデータを引き出すことになる。これにより、Ｇコードは不要になる。

このコンセプトは、インターネットを介してデータを受けたり保存したりするサーバにＣＮＣがネットワークで接続可能であることを意味している。

ＥＭＩはテスト部品において、３つの選択した製造品のフィーチャを、実際に切断するデモンストレーションを行なうことを予定している。

更にＣＮＣがＷｅｂ上の製品データベースにアクセスし、機械加工するためのフィーチャを探し出し、次にコマンドを発生さて穴をあけ、スロットを研磨して、ポ

ケットを機械加工するところを次の目標として定めている。

STEP Tools 社はローレンスリバモア国立研究所と一緒に活動しており、そこでは、ＳＴＥＰ＿ＮＣインタフェースがＯＭＡＣ(Open Modular Architecture Control)プロジェクトのために開発されている。ＳＴＥＰ＿ＮＣの追加テストとデモンストレーションは、ペンシルバニアのスクラントンにある General Dynamics Land Systems による生産機械加工施設で行なわれることが予定されている。Jet Propulsion Laboratory のパイロットプロジェクトもまた、資金獲得のための提案計画中である。

(4) ＳＴＥＰ＿ＮＣプロジェクトの将来ビジョン

ＳＴＥＰ＿ＮＣの完成は作業現場から、経験と技能を要求されるＮＣプログラムを作成する中間的なステップが排除されることを意味する。

経験と技能の必要な製造技術者の煩雑な作業とは、例えば次のような作業を意味するであろう。

ＮＣプログラム作成の中間的な工程の大部分は製品データの変更が必要となった場合は、最初から、部品形状を再入力して加工条件へ変換して、再度工程設計を行なう必要があった。また、新しい工程設計にあわせて形状を処理するようにツールパスも再生成しなければならなかった。更にツールパスファイルは、工作機と制御装置との組み合わせの要求条件に合うように、後処理する必要があった。作業現場では実際の部品加工にあわせて後処理されたファイルをしばしば編集する必要があった。要するに、部品が金属から実際に削り出すまでに、１つの設計部品情報から百以上の加工に関する検討をする必要があった。

ＳＴＥＰとＳＴＥＰ＿ＮＣは 製品加工の世界から生産現場の技術者に技能を要するこのような煩雑な作業から開放することが可能になった。

工作機械による加工業務から技能と経験を必要でなくすることはＧコードプログラミングを時代遅れにする。ＣＡＭソフトウェアの進歩は、Ｇコードをますます少なく、プログラマとマシンオペレータにプログラミングテクニックを不要にすると考えられる。

但し、製造技術者は本来の製造現場の最適な製造に関わる知識と判断がさらに要求されるであろう。すなわち、どのようなマシンツールのためのプロセスデータが用意されるだろうが、重要な加工作業現場の工作機械のローカルパラメーターは製造技術

者の判断が必要である。

最適なカッティングツールを選択して、供給、スピードなどの設定は、作業現場の工作機で行なわれることになるからである。

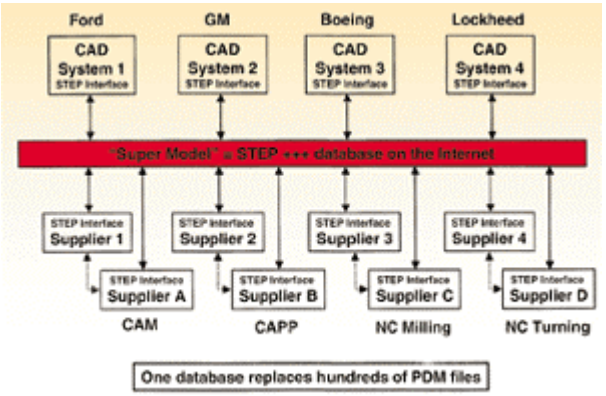


図 2-43 STEP_NC情報システム

スーパーモデルは、何百の製品モデル管理データファイルを、一つのデータベースに置き換えて、インターネットで共有することを可能にする。製品モデルのデータ交換やフォーマット変換のために必要とされたトランスレータ、エディタ、ポストプロセッサは、必要がなくなる。

加工フィーチャを認識して現場の工作機械の作業条件に最適化されたパラメーターを入力したら、後はＣＡＭ機能による知的なコントローラの領域になる。このソフトウェアは適切なパラメーターがＣＮＣに設定された後に、部品を作るのに必要な動きを生成する。

このＣＡＭ機能は将来のすべてのＣＮＣに必要なコンポーネントとなるであろう。

ＳＴＥＰ＿ＮＣの開発は単なる工作機械の機能強化と作業概念の変化をおこすのでは無く、製造そのものに変革を与える事になると予想される。そして、現状のビジョンでは、ＣＮＣ工作機は更に以前より重要になると考える。

デジタル製品モデルデータベースは、設計と工学の視点から見ると製品をただ一つのデータベースでデザインと製造を管理することができることを意味する。

製品の間、法人の間、アプリケーションの間で、製品データを共有することができ

る。インターネットにより、グローバルで仮想的にこのデータの共有を瞬時にできるようになる。

　また、このコンセプトは今後ＣＡＤとＣＡＭが異なった関係を持つことを意味する。

　ＳＴＥＰを用いた製品モデルは、設計者と製造技術者間で新たな高度の協調作業と新しい可能性の展開を与えることになるものと期待される。

3 設計・加工情報共有の実装検討

3.1 はじめに

現在、工業製品のサプライチェーンにおいては、設計・加工情報を再利用、共有することにより生産性を向上させることが求められており、その実現には設計・加工情報を伝達することが不可欠である。

また、インターネットに代表される、安価な通信インフラの拡充は、設計・加工情報を電子的に交換・共有するための環境を促進している。

そこで、設計・生産現場における情報の共有の実装に焦点をあて、そのための方策を検討することとした。

設計・加工情報の交換、共有を踏まえた電子的伝達のための有効な手段として、ＳＴＥＰに代表される製品モデルがある。しかしながら、製品モデルは業務で使用するアプリケーションソフトウェアに内蔵されて使われるため、一般にアプリケーション開発者以外の方々にはなじみが薄く、如何にこれを資産として生かすか考えにくい面がある。設計・生産現場における情報の共有のための手段として、ＸＭＬ、Ｗｅｂ技術と結合したＳＴＥＰの使い方を検討することを通して、データモデルを業務に身近なものとして理解し、使いこなしていただくための提案を行なう。

3.1.1 設計・加工情報活用の目的

設計から生産準備、製造へと、一般に製品開発の下流に行くほど、地域的に分散した形で仕事が行なわれる傾向にある。このため、設計・加工情報のタイムリーな取得や、変更情報の迅速な伝達は、生産性の向上のための不可欠な要件となる。

そのためには、設計・加工の情報を電子的に表現し、伝達を行う必要がある。

ここでは、ＳＴＥＰに代表される製品情報モデルを実装し、ＸＭＬで表現して伝達する方策を検討する。これにより、分散環境において、設計・加工情報をＷｅｂにより共有することができる。

3.1.2 設計・加工情報を共有するための必要条件

設計・加工情報を共有するための条件としては、

- 情報を伝達・交換・共有するためのネットワークとリポジトリの存在

- 設計・加工情報伝達のための標準モデルの存在
- 設計・加工情報を扱うアプリケーションの存在

が必要になる。

ネットワークに関わる条件としては、

- 独自のネットの構築による共有
- WWWによる共有

が現実解としてあるが、ここでは、開かれた場における情報の伝達に有用なWWWによる共有について検討する。

3.2 設計・加工情報を扱うための標準モデル

製造業の分野において、従来一般に図面等で表現されていた製品情報企業間情報交換、共有が近年のＩＴ化の進展にともない大きく変化しようとしている。

一方、製品情報は様々な方法や形式で電子化されており、異なったアプリケーションシステム間、異なったベンダーのソフトウェア間、あるいは異なったバージョン間のデータのやりとりは困難である。企業間の情報交換においては、この問題が顕在化しやすく、このため企業間情報連携の壁となってきている。

製品情報は、それが扱われるライフサイクルの広がりと、ＩＴ技術の変化の激しさを考えると、様々なやり方での電子化が考えられるため、現状のシステムに基づいて共通モデルを定めることは困難である。

この問題を最終的に解決するための製品記述規格としてＳＴＥＰ（STandard for the Exchange of Product model data）が開発されている。ＳＴＥＰでは、システムに注目してそのインターフェースを定義するのではなく、業務の活動モデルを作成し、情報モデルを作り上げる過程を踏むことにより、業務の核となる情報を取り込んでいる。1994 年にはリリース１が発行され、実証実験や産業での実用性検討を踏まえて実用化へ踏み出している。

そこで、本検討では設計・加工情報を扱うための標準モデルとして、ＳＴＥＰを取上げ検討する。

3.3 文書情報を扱うための標準モデル

情報の一般的な表現方法として文書情報があるが、近年文書情報を扱うための方法が急速に進展している。

3.3.1　ＳＧＭＬ文書

文書を電子的に扱うための有力な規格として、1986 年に国際標準化機構によってＩＳＯ８８７９として制定された文書記述言語ＳＧＭＬ(ＩＳＯ／ＩＥＣ８８７９－１９８６)がある。我が国においても、1992 年にＪＩＳＸ４１５１として制定されている。

ＳＧＭＬは、文書にマークを付加する際のマーク付けの方法、文書構造を表現するための文書型定義を記述するための言語として規定されている。

ＳＧＭＬで記述された文書をＳＧＭＬ文書と言い、ＳＧＭＬ宣言、文書型定義、文書実現値より構成される。

ＳＧＭＬ宣言には、使用する文字セットやＳＧＭＬ文書の処理系に要求する機能などを書く。通常、同じシステム環境でＳＧＭＬ文書を蓄積しておく場合、文書の一つ一つにＳＧＭＬ宣言を付ける必要はない。

文書型定義（Document Type Definition）は、文書を構成する要素（element）と、それらの要素間の関係、要素の属性を宣言する。

文書実現値は、実際の文書の要素に文書型定義の規則に則りマーク付けをしたものである。

```
<!DOCTYPE memo [
<!ELEMENT memo    -- (title,section,date,content)>
<!ELEMENT title    -O (#PCDATA)>
<!ELEMENT section  -O (#PCDATA)>
<!ELEMENT date     -O (#PCDATA)>
<!ELEMENT content  -O (#PCDATA)>
]>
<memo>
<title>年末調整の件</title>
<section>会計課 鈴木</section>
<date>期限11月20日（金）</date>
<content>「給与所得者の保険料控除申告書」を送付致しました。書類を熟読の上、期限までに会計課にご提出宜しくお願い致します。</content>
</memo>
```

図 3-1 SGML 文書の例

図 3-1にＳＧＭＬ文書の例を示す。

図 3-1で<!DOCTYPE memo[…]>の部分が文書型定義である。

文書型定義は、文書実現値の前に記述する方法と、外部ファイルを参照する方法がある。文書実現値の前に記述する方法の場合は要素宣言などを［ … ］の部分に書く。

文書型定義の要素宣言は" <!ELEMENT"ではじまり、要素名、タグの省略指定、続いて要素の階層関係と下位要素の出現順序、出現回数を指定し">"で終わる。ここに書かれる要素名が文書実現値の<…>（開始タグと呼ぶ）や</…>（終了タグと呼ぶ）の中に書かれる名前になる。

図 3-1の 2 行目で要素名"memo"は、下位要素 title,section,date,content から構成され、出現順は title,section,date,content の順であることを示している。3 行目から 6 行目の要素については、下位要素はなく文字データが入るので、ＳＧＭＬの予約語 #PCDATA を記述している。

"--"とか"-O"はタグの省略指定で、"-"は省略不可、"O"は省略可を示す。左側が開始タグの省略可否を、右側が終了タグのそれを示す。

ＳＧＭＬ文書は、テキスト形式で記述し、文書の物理的体裁から論理構造を独立に表現していることから以下の特徴がある。

- 文書の構造を明示的に指示できる。
- 文書構造を手がかりにしたデータの管理、データベース化、文書断片の部品化ができる。
- 文書構造を手がかりにした高度の情報検索ができる。

従って、使用する文書型定義を、文書を取り交わす企業間で共通に定めることにより文書の交換・流通・保存が容易になる。

3.3.2 ＳＧＭＬの適用分野

ＳＧＭＬを採用して電子化するのに適した文書の種類には以下のようなものがある。

- 構造が明確な文書
- 保守作業が頻繁にあり、改訂履歴管理が必要な文書
- 標準化の対象としたい文書
- 同一の文書を色々な媒体で出版したい文書
- 長期保存・長期利用が必要な文書

- 企業間、異システム間での文書の交換・流通が必要な文書
- 製品のライフサイクルの色々な場面で、再利用したい文書

これらに該当する代表的な文書には、技術文書や法務文書が考えられる。

ＳＧＭＬを用い、本稿の最初に述べたような要件で電子文書を実際に運用するためには、ＳＧＭＬ文書支援システムが必要となる。

ＳＧＭＬ文書支援システムにおいて、文書の作成のためには、定めた文書型定義に基づくＳＧＭＬ文書が容易に作成できること。既存文書をＳＧＭＬ文書に変換できることなどが求められる。

文書の閲覧のためには、地理的・時間的に離れた場所でも容易に読むことができること、簡単に体裁付けて表示できること、既定の体裁で印刷ができることなどが求められる。

文書の管理のためには、文書の搬入、搬出の管理、履歴管理ができること、変更に関連して対応する文書が漏れなく変更できること、安全保護対策が考慮されていることが求められる。

現在、ＳＧＭＬ文書支援システムが、必ずしもこれらの全ての要件を満たすわけではないが、これにより、紙の削減、文書の再利用、文書の一元管理、文書の標準化、長期にわたる保管などが実現されつつある。

3.3.3　ＸＭＬの登場

近年のインターネットの普及と相携えて、文書を電子的に扱うためにのＨＴＭＬやＸＭＬなどの新タイプのマークアップ言語が登場した。

ＨＴＭＬ(Hypertext Markup Language)は、ＷＷＷ（World Wide Web）で、多数または不特定多数の相手に情報を公開、伝達する手段として、便利に使えることが認められている。ＨＴＭＬでは主として、その要素を表示するときの書式やレイアウトを制御するためにマーク付けを行なう。このため、情報を共有し自動処理し保存し再利用するための言語としては、制約が多く強力ではない。

この要求に応えるためにＸＭＬ(eXtensible Markup Language; 拡張可能マーク付け言語) が開発中である。ＸＭＬはＷＷＷに対応できるように簡潔・明快・軽量にしたＳＧＭＬのサブセットと形容すべきものである。ＸＭＬ基本部分は 1998 年 2 月Ｗ３Ｃ (World Wide Web Consortium) によって勧告された。

ＸＭＬ文書は、ＳＧＭＬと同様に文書の部分構造を、その前後を"タグ"でマーク付けして

"要素"とし、文書全体を要素の積み重ねにすることによって、その構造を明らかにする。

また、ＳＧＭＬと同様に、要素の種類を使用者が定義できるため、様々なタイプの文書や、数値データの羅列にも有効に対処することができる。

ＳＧＭＬとの違いは、開始タグと終了タグの対応がとれており親子関係にある要素のタグが必ず入れ子になっていることであり、このような条件を満足したＸＭＬ文書(ウェルフォームドＸＭＬ文書、well-formed ＸＭＬ document)は、文書型定義を必ずしも必要としないことである。ウエルフォームドＸＭＬ文書は、分析ソフトウェアによって全体を要素の木として認識することができるため、検索･取り出しをしたり、特定の部分を読み取ってデータベースに取り込むなど、任意のプログラム処理を施すことができる。したがって、伝票のようにデータ中心のものから、一般のビジネス文書、設計図書、保守記録、医療カルテ、法務文書など、多様な文書に対する応用が期待されている。

図 3-2にウエルフォームドＸＭＬ文書の例を示す。

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<memo>
<title>年末調整の件</title>
<section>会計課 鈴木</section>
<date>期限11月20日（金）</date>
<content>「給与所得者の保険料控除申告書」を送付致しました。書類を熟読の上、期限までに会計課にご提出宜しくお願い致します。</content>
</memo>
```

図 3-2 XML 文書の例

3.4　Ｗｅｂ上における設計・加工情報の取り扱い方法

前章にみたように、各種の構造化された情報を表現したり、Ｗｅｂ上で情報を取り扱うにあたっては、ＸＭＬが有力な方法であることが理解できる。

ＸＭＬの利点をもう一度整理すると以下が挙げられる

- データが構造化されるのでアプリケーションからのアクセスがしやすい。
- データの意味とデータ自身を区別可能
- ファイル間参照もＨＴＭＬのように指定可能
- Internet Explore5.0 や Netrscape5.0 のブラウザでもＸＭＬ表示をサポート
- その他ＸＳＬなどのスタイルシート言語などの多くのサポートが容易されている。

このため、ＳＴＥＰ規格においてもＥＸＰＲＥＳＳスキーマとデータの交換実装規格として Part28 の検討を鋭意進めている。

Part28 については別途調査することとし、ここでは設計・加工情報のＸＭＬ表現についての基本的な範囲の検討を行なう。

3.5　ＳＴＥＰモデルを実装しＸＭＬで表現するための準備

本論では、設計・加工情報をＷｅｂで共有するための方策として設計・加工情報をＳＴＥＰモデルで表現し、それをＸＭＬ出力方法について検討する。

設計・加工情報をＳＴＥＰで表現すること自体は、ＸＭＬ表現とは、別個の問題であり、設計・加工情報をＸＭＬで表現することにのみ着目するのであれば、必ずしも検討しなくても良いことがらである。しかしながら、設計・加工情報の送出、表示・加工のみならず、情報の追加・更新を行なおうとするとそのためのアプリケーションがアクセスするデータ構造が必要となる。このときそのデータ構造をＳＴＥＰのモデルで表し、その出力表現としてＸＭＬを使用するならば、より適用性の高いシステムを構築できることとなる。

このシステムの実装方式を検討する準備として、以下では

- ＳＴＥＰ実装の基礎
- Java 言語の基礎

を説明する。

3.5.1　ＳＴＥＰ実装の基礎

図 3-3にＳＴＥＰ規格体系を示す。

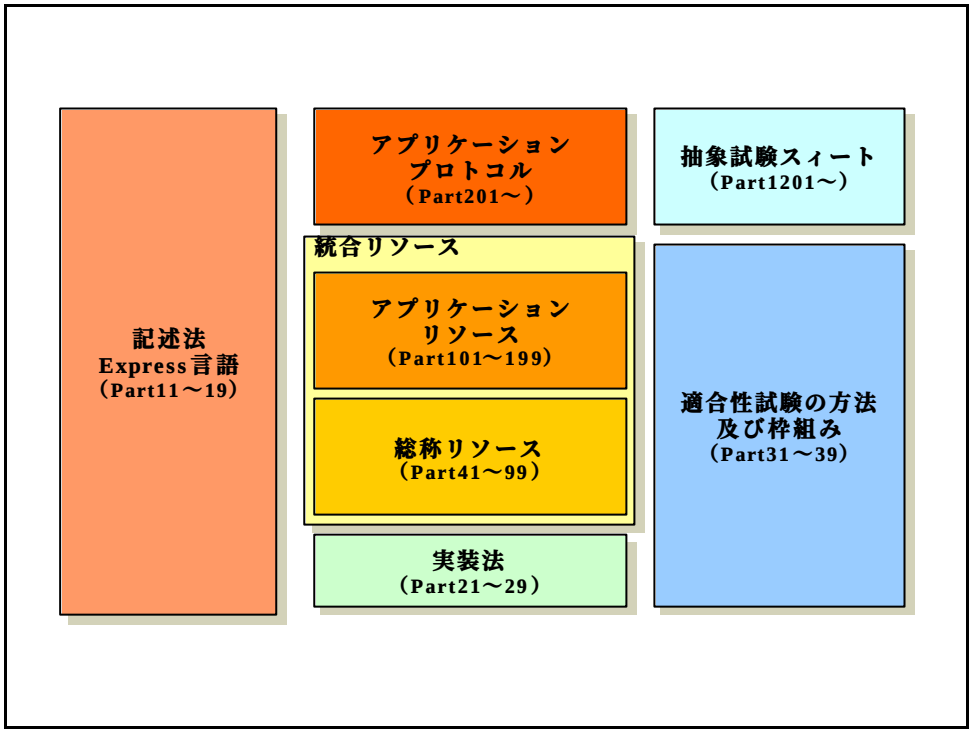


図 3-3　STEP規格体系

ＳＴＥＰ規格は一つの規格から造られているのではなく、さまざまな産業の製品モデルを実現するための、膨大なパート（part）と呼ばれる規格群により構成されている。各パートは以下の７つの組にまとめられる。

- アプリケーションプロトコル
- 統合リソース：総称リソース
- 統合リソース：アプリケーションリソース
- 記述法
- 実装法
- 適合性試験：適合性試験方法論及び枠組み
- 適合性試験：抽象試験スイート

アプリケーションプロトコルは、個々の応用分野の要求に従い、実際に交換される一群

のデータ表現を規定する。ＳＴＥＰの利用者は、自分に適切なアプリケーションプロトコルを選択して、これを実装したアプリケーションプログラムを使用することになる。

統合リソースは、製品モデルデータ記述に必要な基礎的なデータ資源をまとめたものである。統合リソースは、どの応用分野でも共通に利用できるデータ資源である汎用的な総称リソースと、個々の応用分野ごとの一般性の高いデータ資源であるアプリケーションリソースに分けられる。総称リソースは、その間で相互に参照しあっており、アプリケーションリソースは総称リソースを参照している。

記述法には、製品データの記述にあたり整合性を確保しかつ曖昧さを排除するためのＥＸＰＲＥＳＳという名のデータ仕様記述言語が用意されている。この言語は、人に理解可能であると同時にデータ処理系の作成に役立つよう、計算機処理可能になっている。アプリケーションプロトコルや統合リソースは、このＥＸＰＲＥＳＳ言語で記述されている。

実装法は、アプリケーションプロトコル実装のために、ＥＸＰＲＥＳＳ記述された規格を、同様に形式言語記述された実装形式に展開するための方法を定義している。

実装法のパートとしては物理ファイル交換構造（ＩＳＯ１０３０３－２１）や標準データアクセスインターフェース（ＩＳＯ１０３０３－２２）がある。

適合性試験は、適合性試験方法論及び枠組みと抽象試験スイートの２つでカバーされる。「適合性試験方法論及び枠組み」は、適合性試験他の試験のための枠組みを提供する。「抽象試験スイート」は、良いプロセッサの開発を可能にし、問題のない交換への期待を促進する。

(1) ＳＴＥＰ実装法の概要

規格として定められている規格群の中でＳＴＥＰを実装する際に特に関係してくる規格としては、記述言語に関する規格である Part11、ＳＴＥＰ交換ファイルの規格を定めた Part21 がある。ここでは上記が関連する規格書の概要を紹介する。（本節の多くの例は Part11/Part21 の規格書から引用している）

(2) Part11 の概要

本項では Part11 からＥＸＰＲＥＳＳの主要機能を紹介する。ＥＸＰＲＥＳＳの主要な機能は、次の宣言によって得られる。

- エンティティ
- 型

- **スキーマ**
- **定数**
- **関数・手続**
- **大域ルール**

以下に概説する。

① エンティティ

仕様書の中では「共通の特性によって定義された情報のクラス」と定義されている概念であり、製品モデルを構成する最小の単位である。具体的にはエンティティ名、エンティティを構成する属性（attribute）の組で定義される。

（例）実数の座標を持つ 3 次元の点の定義。

```
ENTITY point;
  x : REAL;
  y : REAL;
  z : REAL;
END_ENTITY;
```

図 3-4にエンティティの構成を示す。

図 3-4 エンティティの構成

② 属性

エンティティの属性は、エンティティの特色、特性を表わす。属性の宣言はエンティティと、属性によって参照されるデータ型との間の関係を定める。属性は、明示属性、誘導属性（DERIVE）、逆属性（INVERSE）の３種類が定められている。

③ 局所ルール

局所ルールは、エンティティインスタンスの定義域に関する制限を与える。局所ルールは、一意性ルール（UNIQUE）、定義域ルール（WHERE）がある。

④ 下位型及び上位型

ＥＸＰＲＥＳＳではエンティティ間で上位型（supertype entity）、下位型（subtype entity）の定義ができる。下位型は上位型の持つ全ての性質（定義、属性、制約条件）を引き継ぐ。以下の継承関係を定義できる。

- ＯＮＥＯＦ継承

  下位のエンティティ間で性質を共有しない排他的な関係があるもの。

  （例）多くの種類の花（flower）があるが、どの花も同時に複数の種類に属す

る事はできない。

```
TYPE color = ENUMERATION OF (white,red,pink,yellow,bule);
END_TYPE;

ENTITY flower;
 ABSTRACT SUPERTYPE OF ( ONEOF(rose,violet,lily,chery));
 flower_color ; color;
END_ENTITY;

ENTITY rose;
 SUBTYPE OF (flower);
 rose_attr:STRING;
END_ENTITY;

ENTITY  violet；
 SUBTYPE OF（flower）;
 violet_attr:STRING;
END_ENTITY；
```

- **ＡＮＤＯＲ継承**

下位型のエンティティが互いに排他的で無い場合。

（例）人（person）**は夜間に学校に通う従業員（**employee）**かもしれないので、人は同時に従業員であり、かつ学生（**student）**であってもよい。**

```
ENTITY person;
 SUPERTYPE OF (employee ANDOR student);
 name : STRING;
  age  : INTEGER;
 address : STRING;
```

```
END_ENTITY;

ENTITY employee;
 SUBTYPE OF (person);
 employee_attr:STRING;
END_ENTITY;

ENTITY student;
 SUBTYPE OF (PERSON);
 student_attr:STRING;
END_ENTITY;
```

- **ＡＮＤ継承**

上位型を完全に分類する方法が複数ある事を示すとき、対象を 2 以上の分類法によるＯＮＥＯＦ継承の組み合わせで表現する場合に利用される。

（例）人（person）は、男性（male）または女性（female）に分類できて、かつ国民（citizen）または外国人（alien）にも分類できる。

```
ENTITY person;
 SUPERTYPE OF ( ONE OF(male,female) AND ONE
OF(citizen,alien));
 name : STRING;
 age  : INTEGER;
 address : STRING;
END_ENTITY;

ENTITY male;
 SUBTYPE OF (person);
 male_attr:STRING;
```

```
END_ENTITY;

ENTITY female;
 SUBTYPE OF (person)；
 female_attr:STRING;
END_ENTITY;

ENTITY citizen;
 SUBTYPE OF (person);
 citizen_attr:STRING;
END_ENTITY;

ENTITY alien;
 SUBTYPE OF (person);
 alien_attr:STRING;
END_ENTITY;
```

- **多重継承**

一つのエンティティが複数の上位エンティティを継承する場合、すなわち、列挙された全ての上位型エンティティの性質を継承する必要がある。

（例）

```
ENTITY nurbs_curve;
 SUBTYPE OF
(b_spline_curve_with_knots,rational_b_spline_curve);
END_ENTITY;
```

⑤ 型

ＥＸＰＲＥＳＳでは様々な製品モデルを表現するために、様々なデータ型を持っている。データ型は、単純データ型、集合体データ型、名前付きデータ型、構成デ

ータ型、及び一般化データ型が定義されている。

- **単純データ型**

  数値型（NUMBER）、**実数型**（REAL）、**整数型**（INTEGER）、**文字列型**（STRING）、**ブール型**（BOOLEAN）、**論理型**（LOGICAL）、**及び** 2 **進データ型**（BINARY）**からなる。**

（例）個人データの表現。

```
ENTITY person;
  name: STRING;
   age : INTEGER;
   address : STRING;
END_ENTITY;
```

- **集合体データ型**

  配列（ARRY）、**リスト**（LIST）、**バック**（BAG）、**セット**（SET）**のデータ型が定義されている。**

（例）バック、セットの定義例

```
a_bag_of_point : BAG OF point;
a_set_of_point : SET OF point;
```

- **名前付きデータ型**

  エンティティ宣言で定めるエンティティとＴＹＰＥ宣言で定める定義型からなる。

（例）point **による線の定義例（エンティティ宣言）**

```
ENTITY line;
  p0 : point;
   p1 : point;
```

END_ENTITY;

（例）構造物の測定単位を示す例（TYPE 宣言）

TYPE volume = REAL;

END_TYPE;

ENTITY structure;

bulk : volume;

END_ENTITY;

- **構成データ型**

列挙型（ENUMERATION）とセレクト型（SELECT）がある。

（例）車の動きを定義する。

TYPE car_move = ENUMERATION OF (forword,back,left,right);

END_TYPE;

（例）異なる型を使い、ある役割を表わす場合にセレクト型を使用することができる。

TYPE attachement = SELECT(nail,screw) ;

END_TYPE;

ENTITY nail;

length : REAL ;

head_area : REAL ;

END_ENTITY;

ENTITY screw ;

length : REAL;

pitch : REAL;

END_ENTITY;

- **一般化データ型**

  ある他のデータ型を一般化するのに指定する。

⑥ スキーマ

関連するエンティティ及び他のデータ型宣言の集まりはスキーマ（SCHEMA）としてまとめられる。

図 3-5にスキーマの構成を示す。

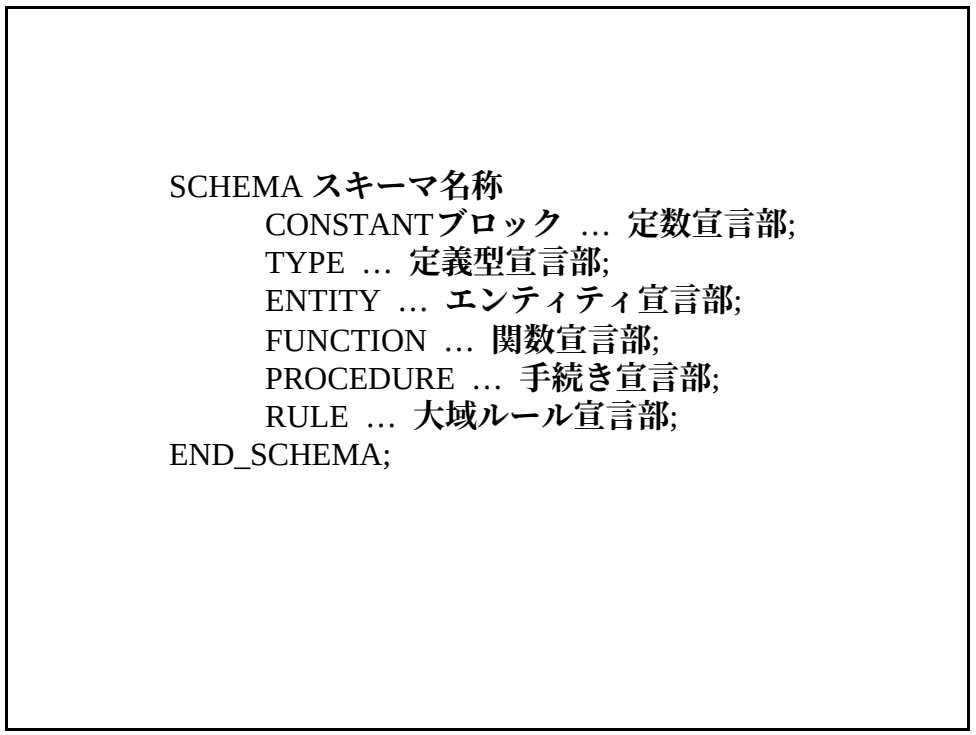


図 3-5 スキーマの構成

⑦ 定数

ＥＸＰＲＥＳＳでは名前付き定数宣言を使用できる。

（例）

```
CONSTANT
  thousand : NUMBER := 1000;
  milion : NUMBER := thousand**2 ;
```

END_CONSTANT;

⑧ 関数・手続

ＥＸＰＲＥＳＳでは関数、手続によりアルゴリズムを表現できる。

関数呼び出しは式であり特定のデータ型を返す。手続は実行文であり戻値を持たない。手続の仮引数にＶＡＲ：を指定すると、引数のもつエンティティインスタンスや値が変化したとき呼び出し側に反映できる。

実行文はアルゴリズムを記述するための具体的な処理を記述する。ＥＸＰＲＥＳＳで定義されている実行文は、空文、ALIAS 文、代入文、CASE 文、複合文、ESCAPE 文、IF 文、手続呼出し文、REPEAT 文、RETRN 文、SKIP 文がある。

⑨ 大域ルール

大域ルールはスキーマの有効範囲で、一つ以上のエンティティデータ型に適用される制約を定義する。

(3) Part21 の概要

Part21（Clear text encoding of the exchange structure）はＪＩＳでは「交換構造のクリアテキスト符号化」と訳されているＳＴＥＰ交換ファイルの仕様を定めた規格であり、Part11 と同様に最初のＩＳとして発行された実装法に関する規格である。

この規格ではＥＸＰＲＥＳＳ言語から交換構造（ファイル）の構文へのマッピング規則が定められており、ＥＸＰＲＥＳＳスキーマからアスキー表現のＳＴＥＰファイルへ展開されるので、これにより異なるシステム間での製品データを交換することができる。

ＳＴＥＰ交換ファイルは ISO10303-21;の文字列で始まり、END-ISO10303-21;の文字列で終わる ASCII 文字（大文字）により定義されたファイルであり、ファイルの構造はヘッダー節とデータ節により構成されている。

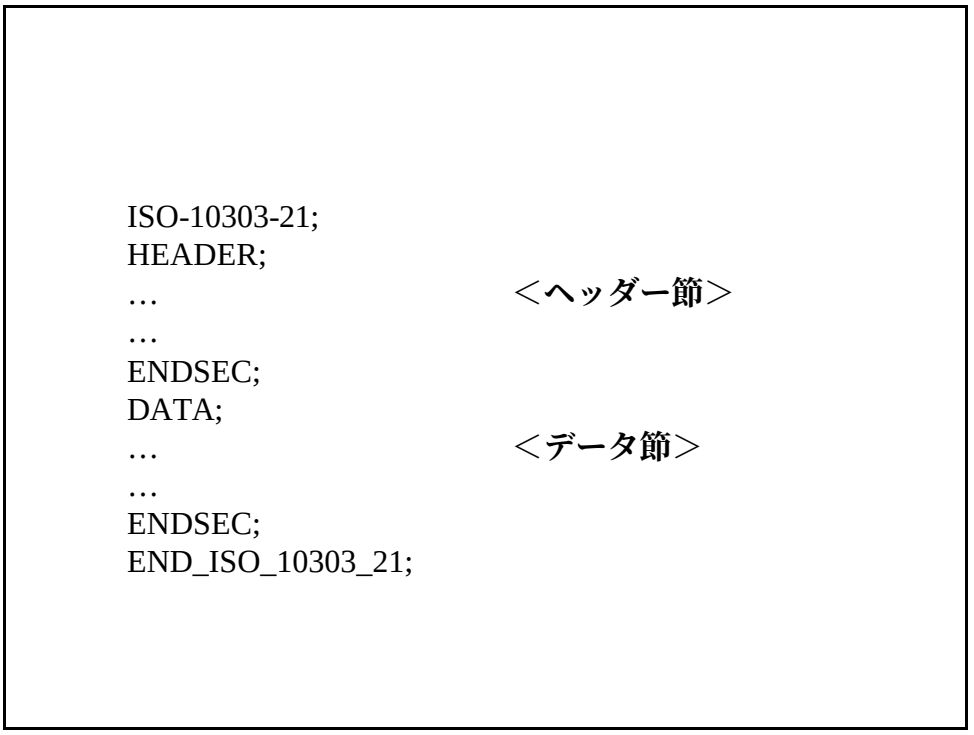


図 3-6　STEP交換ファイルの構造

①　ヘッダー節

ヘッダー節ではファイル名、作成日、作成者、所属、作成システム、スキーマ名、実装レベルなどのヘッダー情報が記述されている。

②　データ節

ＥＸＰＲＥＳＳのデータ構造が写像されたインスタンスデータが並ぶ、定義されている書式を以下に示す。

#自然数　=　エンティティ名（属性 1、属性 2、…）；

③　マッピング規則

- **単純データ型**

  属性定義の順番に記述する。

（例）単純データ型のマッピング例

ENTITY widget;

i1:INTEGER;

```
    bn:BINARY;
    s1:STRING(3);
    s2:STRING;
    l :LOGICAL;
    b :BOOLEAN;
    r1:REAL(4) ;
    r2:REAL;
  END_ENTITY;
```

以下の様にマッピングされる。

```
#10 = WIDGET(10,"123FBD",'ABC','XYZABCDEFG',.T.,.F.,5.000,5.4321) ;
```

- **集合体データ型**

  LIST、ARRAY 等の集合体データ型は"(")"で要素を括って、((要素 1)、(要素 2)、(要素 3))の様に表現する。なおオプショナル属性で値を持たないものは$で表現する。

（例）LIST の例

```
  ENTITY widget;
    att1 : LIST[0:?] OF INTEGER;
    att2 : LIST[1:?] OF INTEGER;
    att3 : OPTIONAL LIST[1:?] OF INTEGER ;
    att4 : REAL;
  END_ENTITY;
```

は以下の様にマッピングされる。

```
  #20 = WIDGET((),(10,20,30),$,12.5);
```

（一番目の要素は empty list (list with zero element)）

（例）ARRAY の例

```
ENTITY widget；
  att1:ARRAY [-1:3] OF INTEGER;
  att2:ARRAY [1:5] OF OPTIONAL INTEGER ;
  att3:ARRAY [1:2] OF ARRAY [1:3] OF INTEGER ;
END_ENTITY;

#30 = WIDGET((1,2,3,4,5),(1,$,3,4,5),((1,2,3),(10,20,30)));
```

（例）SET の例

```
ENTITY widget;
   a：SET OF INTEGER ;
END_ENTITY;

#40 = WIDGET((1,2,3)) ;
#41 = WIDGET((0,$,2));
#42 = WIDGET((0,0,3));
```

（例）BAG の例

```
ENTITY widget;
   b：BAG OF INTEGER;
END_ENTITY;

#50 = WIDGET((1,2,3));
#51 = WIDGET((10,11,12,$,14,15,16));
```

- **名前付きデータ型**

（例）エンティティの例

```
ENTITY part ;
```

```
  att1：LIST [0,3] OF REAL;
  size: REAL ;
END_ENTITY;

ENTITY widget;
  id : NUMBER;
  pp : part;
END_ENTITY;
```

参照されるエンティティは#番号で識別する。

```
#60 = PART((1.1,2.2,3.3),100.23) ;
#61 = WIDGET(10,#60);
```

（例）定義型（TYPE）の例

```
TYPE
  type1 = INTEGER;
END_TYPE;

TYPE
  type2＝ LIST [1:2] OF REAL;
END_TYPE;

ENTITY widget;
  att : REAL;
  att1 : type1;
  att2 : type2;
END_ENTITY;

#70 = WIDGET(10.5,3,(11.1,22,2)) ;
```

- **列挙型**

ENUMERATION **は該当するインスタンスの実データがマッピングされる。**

```
TYPE
 color= ENUMERATION OF (red,white,green,blue);
END_TYPE;

ENTITY widget;
 p_color : color;
END_ENTITY ;
```

```
#80 = WIDGET(.RED.) ;
```

- **セレクト型**

セレクト型の型名が特定できない場合には、セレクト型名付きで写像する。型名が特定できる場合には定義元のマッピングを使う。

（例）

```
TYPE
 size = SELECT(area,radius);
END_TYPE;

TYPE
 area : REAL ;
END_TYPE;

TYPE
 radius : REAL ;
END_TYPE;

ENTITY circle;
```

```
  ll : size;
END_ENTITY;
```

値 30.3 **の** AREA **で** CIRCLE **は表わされている**。

```
#90 = CIRCLE(AREA(30.3));
```

- **誘導属性**

誘導属性（DERIVED **属性**）**はＳＴＥＰ交換ファイルにはマッピングされない**。

（例）

```
ENTITY yy;
  y1：REAL:
  y2：REAL;
  y3：REAL;
END_ENTITY;

ENTITY xx;
  x1：yy ;
  x2：yy ;
  x3：yy ;
  DERIVE
  att1：REAL:=ff_function(x1,x2);
  att2：REAL:=ll_function(x2,x3);
END_ENTITY;

#100 = YY(10.1,20.2,30.3) ;
#101 = YY(100.1,200.2,300.3);
#102 = YY(100.1,2000.2,3000.3) ;
#103 = XX(#100,#101,#102) ;
```

- **継承関係**

  継承関係のマッピングは内部写像（internal mapping）あるいは外部写像（external mapping）により表現される。

  －内部写像

  内部写像は、上位型が特定できるもので、上位型の属性を継承して次のように表わす。

  #番号＝エンティティ名（属性 1、属性 2、…）　；

  －外部写像

  外部写像は上位型が特定できない時に使用する写像であり、上位型の型名を含めて次のように記述する。

  #番号＝（AA('XYZ')BB(10,20.5)CC(#10)）

- **その他**

  －誘導属性により再定義された属性

  上位型の属性を下位型の誘導属性を用いて再定義している時は、上位型の属性を"*"で表わす。

  －定義域ルール、逆属性、大域ルール、CONSTANT、コメント

  マッピングしない。

3.5.2　Java 言語の基礎

(1) Java の特徴

Java は、一般的に「インターネットのようなオープンな分散コンピューティング環境でアニメーションやインタラクティブな操作を簡単に実現できる安全性の高いオブジェクト思考のプログラミング言語」として認識されている。

Java がサポートしている主なオブジェクト指向技術を以下に示す。

- **モジュール化と情報隠蔽を可能にするインスタンス変数とメソッドを一緒にまとめるカプセル化機能**
- **新しいクラスとその振舞いは既存のクラスで定義できる継承機能**
- **メッセージが複数のオブジェクトに送られたときでも、オブジェクトの性質によって決定されるポリモーフィズム機能**

(2) パッケージ及びパッケージ間参照

Java ではパッケージと呼ばれる集合にクラスを集めることを許している。パッケージは、ネストレベルなどによって作業を系統立て、他の人によって提供されるライブラリコードからその仕事を分離するのに用いられる。

パッケージの中で直接宣言できるものは、クラスとインターフェースである。パッケージ宣言の形式を以下の図 3-7に示す。

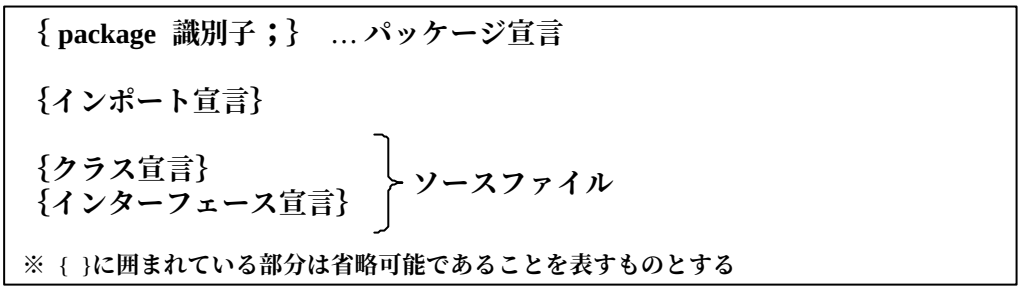


図 3-7 パッケージ宣言の形式

パッケージ宣言は省略可能であり、省略可能されたときはデフォルトパッケージとよばれる無名のパッケージの中に自動的に組み込まれる。

また外部で宣言されたパッケージ内の項目を、インポート宣言によって使用可能にすることができる。インポート宣言にはパッケージにおける完全名(クラスやインターフェース名まで)を指定する方法と、パッケージ自体を指定する方法の２種類があり、そのインポート宣言の形式を以下の図 3-8に示す。

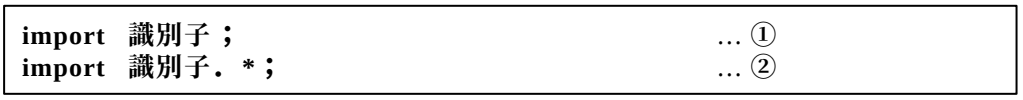


図 3-8 インポート宣言の形式

図 3-8の①のようにした場合、その識別子で表される１つのクラスやインターフェースがインポートされ、②のようにした場合はその識別子のパッケージに含まれる全

てのパッケージ、クラス・インターフェースをインポートする。これは、後に説明するクラスや、メソッド、変数を参照する際にフルパスを入力せず、短縮形で参照することを可能にするためのものである。但し、他のパッケージから参照しない場合や、直接プログラム内で完全名を用いて参照する場合はインポート宣言は省略可能である。

(3) データ型

Java は、強く型付けられた言語であり、全ての変数がデータ型をもたなければならない。型は、変数が適用し得る値または式から導出される値と、それらの値に作用できる演算を限定し、その演算の意味を決定するものである。変数は、型とその識別子という２つのコンポーネントから成っている。

データ型は大きく基本型、参照型の２つに分類される。

- 基本型

  Java の基本型には、byte、short、int、long、char、float、double の８種類がある。整数はすべて 10 進数・16 進数・8 進数を扱うことができる。

- 参照型

  Java の参照型には３種類あり、それらはクラス型、インターフェース型、配列型である。Java でクラスとはオブジェクトの構造と機能を規定するための型であり、インターフェースとは、そのインターフェースを実装するクラスで継承したいメソッド名と定数を宣言するための型である。

- 配列型

  Java では関連する一連のデータを配列オブジェクトに格納して、それらに同じように操作することを可能にする。その要素を定義するデータ型は、基本型でも参照型でもよい。以下の図 3-9に配列の宣言形式を示す。

{修飾子} データ型名 識別子 [] ;
※ {} に囲まれている部分は省略可能であることを表すものとする

図 3-9 配列の宣言の形式

(4) インターフェース

インターフェースとは、多重継承、つまり２つ以上のクラスからクラスを派生することが Java で禁止されているために、メソッドの名前と定数フィールドを多重継承するのに用いられる型のことを言う。よって、インターフェースクラスはインスタンス化されることはない。

また、インターフェース自身も互いに継承することで派生することが可能である。

インターフェースの宣言形式を以下の図 3-10に示す。

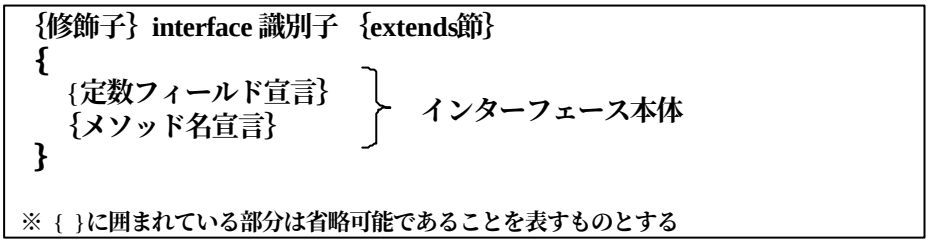


図 3-10 インターフェースの宣言の形式

インターフェースに指定できる修飾子は public と abstract の２種類であるが、元々インターフェースは実装部分を提供しないので明示しなくても abstract 宣言はされているといえる。

インターフェースはユーザー自身が定義するか、ある別のインターフェースを拡張することで得られ、拡張される側のインターフェースをスーパーインターフェース、拡張して得られるクラスをサブインターフェースと呼ぶ。Java はインターフェースには多重継承を許している。extends 節とは、継承するスーパーインターフェースを指定するもので、複数ある場合はカンマで区切って並べる。また、特にスーパーインターフェースを必要としない場合は省略することが可能である。

(5) クラス

クラス型はデータ型の一つとして位置づけられている。クラスの宣言形式を以下の図 3-11に示す。

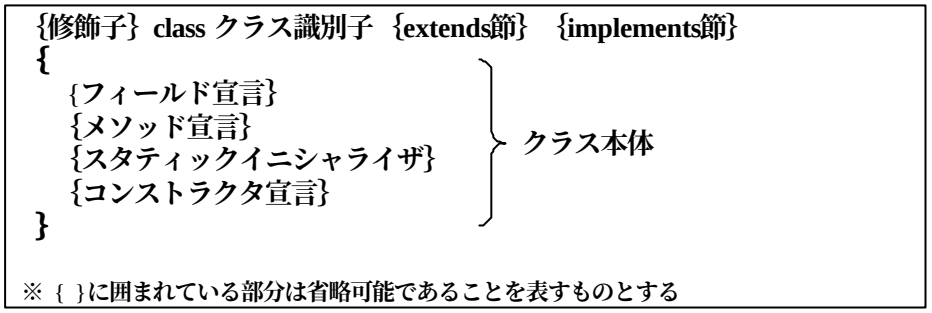


図 3-11 クラスの宣言の形式

クラス宣言において、修飾子とはクラスにスコープや継承に関する制約を加える役割を持っている。クラスに指定できる修飾子は public、abstract、final の３種類ある。但し修飾子は省略可能である。

クラスはある別のクラスを拡張することで得られ、拡張される側のクラスをスーパークラス、拡張して得られるクラスをサブクラスと呼ぶ。Java は単一継承のみをサポートしているのですべてのクラスはスーパークラスを唯一持っている。extends 節とは、継承するスーパークラスを指定するもので、特にスーパークラスを必要としない場合は省略することが可能であり、その場合はスーパークラスとして Java API 中の java.lang.Object クラスをスーパークラスに指定したとみなされる。implements 節とは、実装するインターフェースを指定するもので、複数指定可能であり、その場合はカンマで区切ってその識別子を並べる。特にインターフェースを実装しない場合は書かない。

- フィールド

  フィールドとは、インスタンス変数のことを言い、メソッドと合わせてクラスのメンバと呼ばれる。メンバの名前の scope はそのメンバの属するクラスが宣言されている部分全体である。またクラスのメンバとは、宣言されたメンバに加えスーパークラスから継承したメンバも含まれる。

  フィールド宣言の形式を以下の図 3-12に示す。

```
{修飾子} フィールドのもつ型  フィールド識別子；
※｛ ｝に囲まれている部分は省略可能であることを表すものとする
```

図 3-12 フィールドの宣言の形式

フィールド宣言において、修飾子は、public、protected、final、private、static、等があり、省略してもよいが組み合せて用いることもできる。修飾子に続いてそのフィールドの型を宣言し、その後にそのフィールドの識別子を書く。

- メソッド

メソッド宣言はメソッドの呼び出し式によって呼び出されるようなコードを記述するものである。メソッド宣言の形式を以下の図 3-13に示す。

```
{修飾子} 戻り値の型  識別子(仮引数の型  仮引数名）{throws節}
{
        ／／ 実装部分
}
※｛ ｝に囲まれている部分は省略可能であることを表すものとする
```

図 3-13 メソッドの宣言の形式

メソッドの宣言はまず、修飾子を宣言し、続いてそのメソッドの戻り値のデータ型がある場合はそのデータ型を、ない場合には void を記す。その後メソッドの識別子を書き、次は（ ）の中に仮引数となる変数を、型、仮引数名の順に記す。ただし、複数の仮引数をとる場合は、この型と名前のセットをカンマで区切って並べる。その後、実装時に発生する例外処理のための throw 節を記す。

メソッドに指定できる修飾子としては、public、protected、final、private、abstract、static がある。

- コンストラクタとイニシャライザ

コンストラクタとは、新規のクラスのインスタンスを初期化するのに使われ、

スタティックイニシャライザとはそのクラスが最初にロードされる際、クラスの初期化を助けるのに用いられる、実行可能なブロックのことをいう。

(6) ＡＰＩ（Application Programming Interface）

Java の言語仕様の予約語とされているのは、データ型や修飾子、分岐やループを司る実行文など、ごくわずかな基本的な事柄のみを記述するための用語に過ぎない。それだけでは不十分な表現方法を補助するために、開発環境ＪＤＫ（Java Development Kit）中にライブラリとして、様々なメソッドを含んだクラスやインターフェースがパッケージで用意されている。

3.6 ＳＴＥＰモデルを実装しＸＭＬで出力するための方式の提案

3.6.1 基本的考え

本項では、ＳＴＥＰモデルを実装しＸＭＬで出力を表現するための方式の提案を行なう。

ＳＴＥＰツールについては既にいくつかのものが出ており、ＳＴＥＰモデルからＸＭＬを出力するものも商品化されようが、ここではＳＴＥＰ規格を使いこなし実装するための筋道の理解を目的とする。提案する方式は、各種の制約のため汎用的ではなく、性能的にも充分なものではないが、アプリケーションソフトウエア内におけるＳＴＥＰの役割や動きを理解するためには役立つと考えている。

基本的なシステムの構造は、図 3-14のようになる。

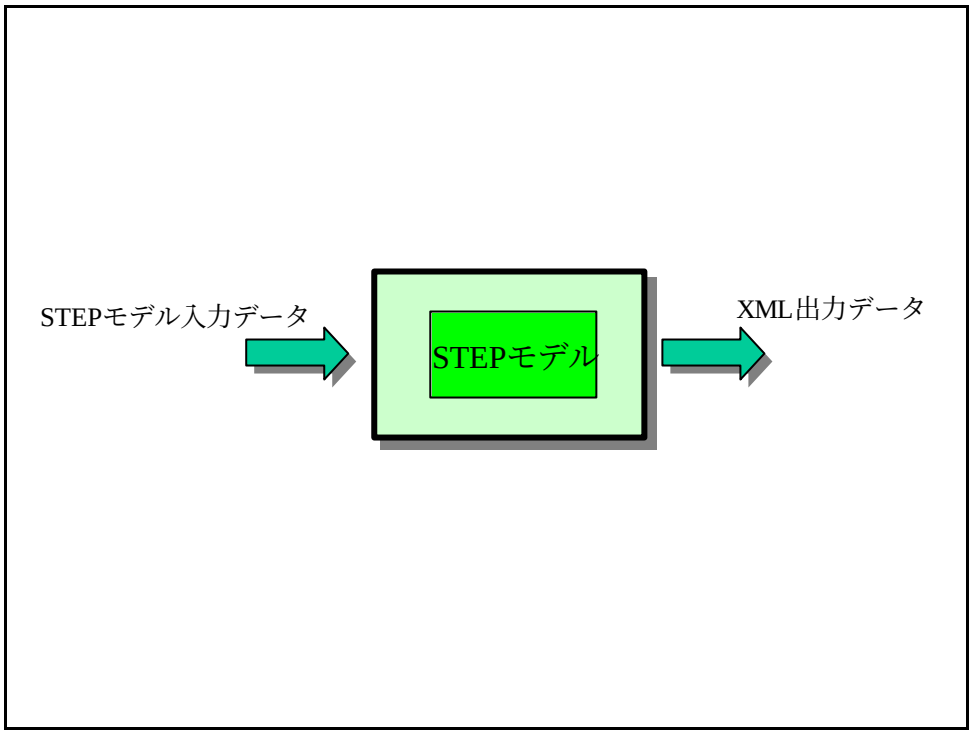


図 3-14 基本的なシステムの構造

図中の「ＳＴＥＰモデル」を表現するデータ構造としては、考えている設計・加工情報を対照とするアプリケーションプロトコルを採用する。

規格としてのＳＴＥＰは、設計・加工情報の論理モデルを規定するものであって、物理モデルを規定するものではないが、論理モデル自体を物理モデルとして扱うことにより、ＳＴＥＰの働きを目に見える形で理解することができる。これにより、プログラムがシンプルに作成できて、実際のシステム構築のプロトタイプとして役立つことも期待される。

ＳＴＥＰモデルの入力データとしては、Part21 の項で見たように、ＳＴＥＰではエンティティを Part21 形式へマッピングする規則が定義しているので、Part21 形式データが適当である。

ここで、Part21 形式データが、Java 言語を使用したプログラムにおける、コンストラクタ呼び出し列に似ていることに気がつく。すなわち、ＳＴＥＰモデル側でアプリケーションプロトコルの各エンティティ等を Java のクラスの形で用意しておき、Part21 形式データをＪava クラスのコンストラクタ呼び出し列のプログラムに変換して実行すれば、入力データから直ちにＳＴＥＰモデルを生成することができることが判る。

3.6.2　提案方式の構成

前節の基本構想を元に、図 3-15に示す構成のシステムを提案する。

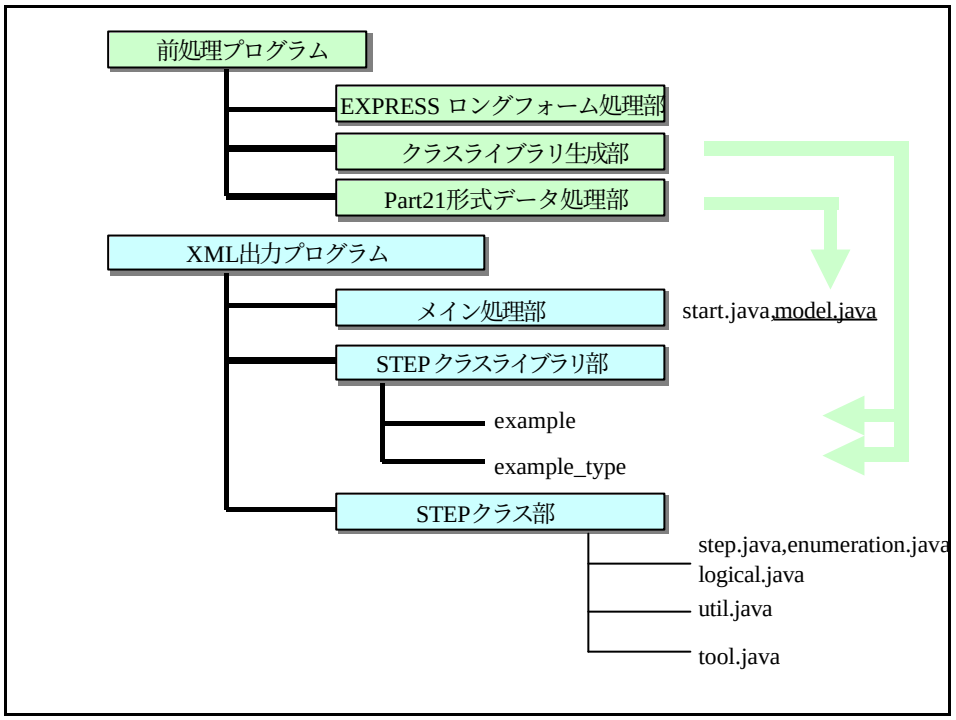


図 3-15 提案方式の構成と処理の流れ

前処理プログラムは３つの機能をもっている。

対照とするアプリケーションプロトコルのロングフォームを入力し、ＥＸＰＲＥＳＳロングフォーム処理部を実行すると、前処理プログラム内部にその内容をテーブル化する。

ＥＸＰＲＥＳＳロングフォーム処理部終了後、クラスライブラリ生成部を実行すると、このテーブルに基づき、当該アプリケーションプロトコルの各エンティティに対応する Java クラスとインターフェースを生成する。

ＥＸＰＲＥＳＳロングフォーム処理部終了後、Part21 形式データを読み込ませて Part21 形式データ処理部を実行すると、Part21 形式データに対応する Java プログラム（model.java プログラム）を生成する。

ＸＭＬ出力プログラムは、メイン処理部、ＳＴＥＰクラスライブラリ部、ＳＴＥＰクラ

ス部の３部から構成される Java プログラムである。

メイン処理部は、メインメソッドを持つ start.java プログラムと前処理プログラムの Part21 形式データ処理部が生成した model.java プログラムからなる。

ＳＴＥＰクラスライブラリ部は、アプリケーションプロトコルのエンティティ、型宣言文等から前処理プログラムのクラスライブラリ生成部が作成した Java クラス、インターフェースより構成される。生成されたクラス群、インターフェース群はアプリケーションプロトコルのスキーマ名（図 3-15では"example"とした）を元にしたパッケージ名（example, example_type）でそれぞれパッケージ化される。

ＳＴＥＰクラス部はＸＭＬ出力プログラムに作りつけの部分である。

step.java は STEP クラスライブラリの各クラスに対応するインスタンスを管理するためのクラスである。

enumeration.java は列挙型の宣言に対応するためのクラスである。

logical.java は論理型宣言に対応するクラスである。

util.java には、アプリケーションプロトコルのエンティティの誘導属性で参照される関数をメソッド群として用意しておく。

tool.java には、ＳＴＥＰモデルのＸＭＬ出力などを行なうメソッドを用意しておく。

3.6.3　ＳＴＥＰモデルの内部管理

model.java プログラムを実行すると、多数のインスタンスが生成される。個別のインスタンスは定義した変数で参照できるが、特定の要件を満たすインスタンスの探索や、一括してＳＴＥＰモデル内のインスタンスをＸＭＬ変換しようとするときの扱いは不便である。

そこでこれらのインスタンスを生成順にリストでつなぐことにする。これにより、すべてのインスタンスを漏れなく処理することができる。この管理を行うために"step"という名前のクラスを用意する。この step クラスは生成されると、直前に生成された step クラスに結びつけられる。この方式を図 3-16に示す。

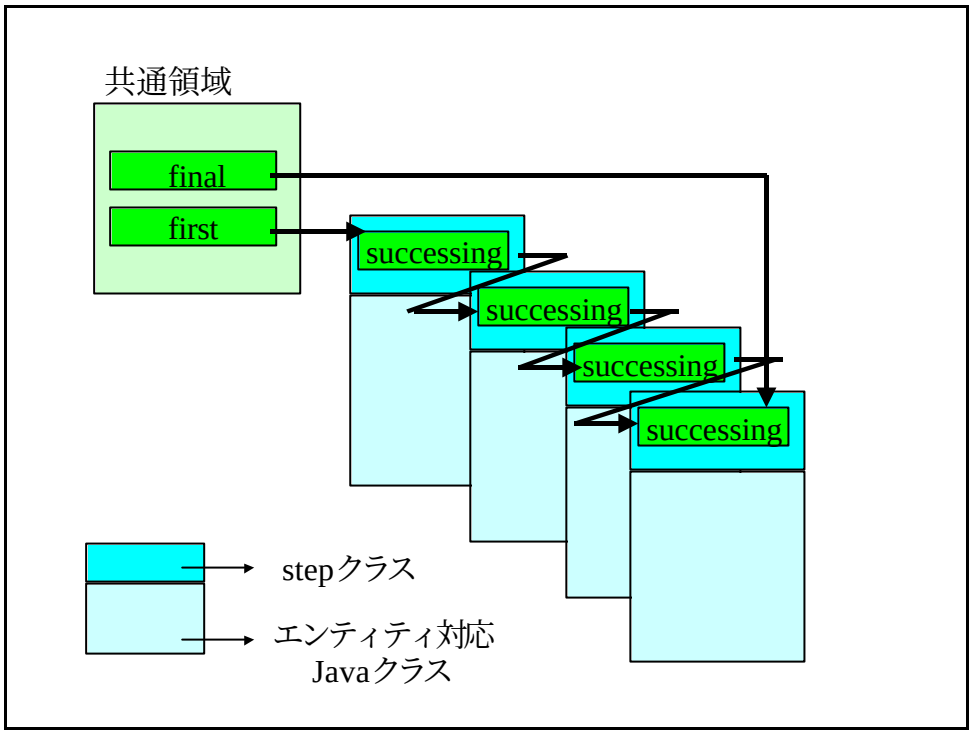


図 3-16 ＳＴＥＰモデルの内部管理の方式

STEP クラスライブラリ部の各クラス（エンティティに対応するクラス）を step クラスの下位型として定義することにより、各クラスのインスタンスは生成時に結びつけられる。

さらに、Java のＡＰＩを用いてＥＸＰＲＥＳＳ言語の汎用関数である USEDIN 関数（参照元を取得する関数）、TYPEOF（データ型を取得する関数）に対応するメソッドを step クラスに用意し、エンティティを表すクラス間を辿れるようにすることで、ＸＭＬ出力だけでなく、スキーマ間の変換へも適用が可能になる。

3.6.4 EXPRESS ロング フォームからの Java クラス生成

先の節でＥＸＰＲＥＳＳ言語と Java 言語の言語レベルでの対応を検討した。この結果、仕様記述言語として見る場合、ＥＸＰＲＥＳＳに比べて Java 言語には足りない機能があることが認められる。

また、一方でＥＸＰＲＥＳＳ言語と Part21 言語のマッピング規則を検討した。

現実のデータ交換が Part21 形式で行なわれているということは、実装について考える場合は、Java 言語でＳＴＥＰの Part21 形式データを自由に取り扱えるレベルをサポートす

れば、実用上ほとんど問題無いことを示唆している。

そこで、Part21 形式を介して表現されたエンティティデータを、対応 Java クラスで出来るだけ近いデータ型で受け止めるという方針で対応を以下に検討した。

- スキーマ

  Java の package 文で対応する。

- インターフェース仕様

  ＥＸＰＲＥＳＳロング フォーム には現れないのでサポートしない。

- エンティティ

  Java のクラスに対応させる。

  －エンティティの明示属性

  明示属性は Java のクラスのフィールドに対応させる。

  －オプショナル属性

  明示属性と区別しない（区別できていない）。データ値が未定義の場合は 0 ないし null 値を代入する。

  －誘導属性

  誘導属性により再定義された属性をサポートする。ここで使用する関数は事前に用意する。

  －逆属性

  Part21 形式データにはマッピングされないのでサポートしない。

  －抽象上位型

  抽象上位型（ABSTRUCT SUPERTYPE）は Java の修飾子 abstract で置換える。

  －下位型

  自クラスがどのクラスの下位型（SUBTYPE）であるかは extends 節に上位型を書くことにより表わす。２重継承の場合は、サブタイプリストの最初のエンティティについては extend 節を使用して継承し、２番目のエンティティについては、その継承を上位に辿って、上位エンティティの全ての明示属性を自クラスのフィールド変数として展開する。２番目のエンティティについては、２番目のエンティティ名に“_”を付加した型名のインターフェース型を定義し、implements 文で実装して使用する。例 4 では空の interface 文を定義しているが、必要な変数アクセスをメソッドとしてここに宣言して使用する。

－ONEOF 継承

SUBTYPE の項で記述した内容により暗にこの機能は継承時に果たされている。

－AND,ANDOR 継承

この結果は、Part21 形式データでは外部写像の形で現れる。外部写像は ANDOR 継承から導出される複合エンティティと考えられる。これらには実用上は良く使われるパターンがあるので、それをロングフォームにあらかじめ独立エンティティとして追加しておき、対応する Java クラスの生成や Part21 形式データの Java 変換に使用することにする。

外部写像を定義するエンティティを定義するために、本報告だけの便宜的なものであるが CONSTRUCT 文を仮に定義し、外部写像を構成するエンティティの名前を CONSTRUCT 文にアルファベット順に宣言する。（例５参照）

このエンティティの継承関係については妥当なパスを検討し、SUBTYPE OF の項に適切な上位型を宣言するのが良い。つまり、単一継承を持つ新 Entity を再定義することになる。

- NUMBER 型

  Java の double 型に対応させる。

- REAL 型

  Java の double 型に対応させる。

  実数の精度はサポートしていない。

- INTEGER 型

  Java の int 型に対応させる。

- STRING 型

  Java の String 型に対応させる。

  文字列の長さはサポートしていない。

- BINARY 型

  Java には対応型がなく、サポートしない。

- LOGICAL 型

  Java には UNKNOWN がないため、対応するクラスがないので事前に Java クラスを用意する。

- BOOLEAN 型

  Java の boolean 型に対応させる。

- ARRAY・LIST・SET・BAG 型

  いずれも、配列に対応させる。このときサイズの大きさを明示宣言しない。

  配列の初期化宣言は、コンストラクタ呼び出し時に行なう。

- 定義型（TYPE）

  定義データ型は基底型を用いて新しい型を作成し、名前を付ける。

  ー基底型が REAL,INTEGER,STRING 型の場合はそれぞれ Java の double,int,String 型に直接対応させる。

  ー基底型が列挙型（ENUMERATION）の場合は、事前に用意する enumeration クラスを継承した対応する Java クラスを作成する。

  ー基底型がセレクト型（SELECT）の場合は、Java の空のインターフェース型（型名としてセレクト型名を使用）に対応させる。セレクト型のリストに宣言されたエンティティに対応する Java クラスの implement 節にインターフェース型名を記述する。

- 関数・手続

  必要な関数･手続はＳＴＥＰクラス部の util.java のメソッドとして登録しておく。

- 大域ルール・局所ルール

  Part21 形式データにはマッピングされないのでサポートしない。

- 定数

  Part21 形式データにはマッピングされないのでサポートしない。

- 総称・一般化集合体・被集合体

  Java の Object 型で対応する。

実際のクラス生成例を以下に示す。

（例 1）エンティティの例

<ＥＸＰＲＥＳＳ表現>

```
ENTITY point;
 x:REAL;
```

```
  y:REAL;
  z:REAL;
END_ENTITY;
```

<Java 表現>

```
package example;
import step.*;
import example_type.*;
public class point extends step{
// field: point
    public double x ;
    public double y ;
    public double z ;
    public point(){  }
   public point(double arg1,double arg2,double arg3){
         x=arg1;
        y=arg2;
        z=arg3;
     }
}
```

（例 2）属性にセレクト型のあるエンティティの例

<**ＥＸＰＲＥＳＳ表現**>

```
TYPE attachement = SELECT(nail,screw) ;
END_TYPE;

ENTITY nail;
  length:REAL;
  head_area:REAL;
```

```
END_ENTITY;
ENTITY screw ;
 length:REAL;
 pitch:REAL;
END_ENTITY;
```

<Java 表現>

```
package example;
import step.*;
import example_type.*;
public class nail extends step implements attachement{
// field: nail
    public double length ;
    public double head_area ;
    public nail(){ }
    public nail(double arg1,double arg2){
        length=arg1;
        head_area=arg2;
     }
}

package example;
import step.*;
import example_type.*;
public class screw extends step implements attachement{
// field: screw
    public double length ;
    public double pitch ;
    public screw(){ }
    public screw(double arg1,double arg2){
```

```
        length=arg1;
        pitch=arg2;
      }
}
```

```
package example_type;
public interface attachement{}
```

（例 3）単一継承のあるエンティティの例

この例では属性に列挙型が使われている。

<ＥＸＰＲＥＳＳ表現>

```
TYPE color = ENUMERATION OF (white,red,pink,yellow,bule);
END_TYPE;

ENTITY flower;
  ABSTRACT SUPERTYPE OF ( ONEOF(rose,violet,lily,chery));
  flower_color:color;
END_ENTITY;

ENTITY rose;
  SUBTYPE OF (flower);
  rose_attr:STRING;
END_ENTITY;

ENTITY violet；
  SUBTYPE OF (flower);
  violet_attr:STRING;
END_ENTITY;
```

<Java 表現>

```
package example;
import step.*;
import example_type.*;
abstract public class flower extends step{
// field: flower
    public color flower_color ;
    public flower(){  }
    public flower(color arg1){
        flower_color=arg1;
      }
}

package example;
import step.*;
import example_type.*;
public class rose extends flower{
// field: rose
    public String rose_attr ;
    public rose(){  }
    public rose(color arg1,String arg2){
        super(arg1);
        rose_attr=arg2;
      }
}
package example;
import step.*;
import example_type.*;
public class violet extends flower{
// field: violet ;
```

```
    public String violet_attr ;
    public violet(){ }
    public violet(color arg1,String arg2){
        super(arg1);
        violet_attr=arg2;
      }
}

package example_type;
import step.*;
public class color extends enumeration{
  public color(String arg1){
    super(arg1);
    super.enume=".white.red.pink.yellow.bule.";
  }
}
```

（例 4）２重継承のあるエンティティの例

この例では属性で配列が使われている。

<ＥＸＰＲＥＳＳ表現>

```
ENTITY b_spline_curve;
  degree              : INTEGER;
  control_points_list : LIST [2:?] OF point;
  curve_form          : STRING;
  closed_curve        : LOGICAL;
  self_intersect      : LOGICAL;
END_ENTITY;

ENTITY b_spline_curve_with_knots;
```

```
 SUBTYPE OF (b_spline_curve);
 knot_multiplicities : LIST [2:?] OF INTEGER;
 knots              : LIST [2:?] OF REAL;
 knot_spec          : STRING;
END_ENTITY;

ENTITY rational_b_spline_curve;
 SUBTYPE OF (b_spline_curve);
 weights_data : LIST [2:?] OF REAL;
END_ENTITY;

ENTITY nurbs_curve;
 SUBTYPE OF (b_spline_curve_with_knots,rational_b_spline_curve);
END_ENTITY;
```

<Java 表現>

```
package example;
import step.*;
import example_type.*;
public class b_spline_curve extends step{
// field: b_spline_curve
    public int degree ;
    public point [] control_points_list ;
    public String curve_form ;
    public logical closed_curve ;
    public logical self_intersect ;
    public b_spline_curve(){  }
    public b_spline_curve(int arg1,point[] arg2,String arg3,logical arg4,logical arg5){
        degree=arg1;
       control_points_list=arg2;
```

```
        curve_form=arg3;
        closed_curve=arg4;
        self_intersect=arg5;
      }
  }

  package example;
  import step.*;
  import example_type.*;
  public class b_spline_curve_with_knots extends b_spline_curve{
  // field: b_spline_curve_with_knots
     public int [] knot_multiplicities ;
     public double [] knots ;
     public String knot_spec ;
     public b_spline_curve_with_knots(){ }
     public b_spline_curve_with_knots(int arg1,point[] arg2,String arg3,logical
arg4,logical arg5,int[] arg6,double[] arg7,String arg8){
        super(arg1,arg2,arg3,arg4,arg5);
        knot_multiplicities=arg6;
        knots=arg7;
        knot_spec=arg8;
      }
  }

  package example;
  import step.*;
  import example_type.*;
  public class rational_b_spline_curve extends b_spline_curve implements
rational_b_spline_curve_{
  // field: rational_b_spline_curve
```

```
    public double [] weights_data ;
    public rational_b_spline_curve(){ }
    public rational_b_spline_curve(int arg1,point[] arg2,String arg3,logical arg4,logical arg5,double[] arg6){
      super(arg1,arg2,arg3,arg4,arg5);
       weights_data=arg6;
     }
  }

  package example;
  import step.*;
  import example_type.*;
  public class nurbs_curve extends b_spline_curve_with_knots implements rational_b_spline_curve_{
  // field: rational_b_spline_curve
    public double [] weights_data ;
  // field: nurbs_curve
    public nurbs_curve(){ }
    public nurbs_curve(int arg1,point[] arg2,String arg3,logical arg4,logical arg5,int[] arg6,double[] arg7,String arg8,double[] arg9){
      super(arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8);
       weights_data=arg9;
     }
  }

  package example_type;
  public interface rational_b_spline_curve_{}
```

（例 5）ANDOR 継承のあるエンティティの例

<ＥＸＰＲＥＳＳ表現>

```
ENTITY person;
 SUPERTYPE OF (employee ANDOR student);
 name:STRING;
 age:INTEGER;
 address:STRING;
END_ENTITY;

ENTITY employee;
 SUBTYPE OF (person);
 employee_attr:STRING;
END_ENTITY;

ENTITY student;
 SUBTYPE OF (person);
 student_attr:STRING;
END_ENTITY;

ENTITY employee_and_student;
 SUBTYPE OF (person)
 CONSTRUCT OF (employee,
               person,
               student);
END_ENTITY;
```

<Java 表現>

```
package example;
import step.*;
import example_type.*;
public class person extends step{
// field: person
```

```
    public String name ;
    public int age ;
    public String address ;
    public person(){ }
    public person(String arg1,int arg2,String arg3){
        name=arg1;
        age=arg2;
        address=arg3;
     }
}

package example;
import step.*;
import example_type.*;
public class employee extends person{
// field: employee
    public String employee_attr ;
    public employee(){ }
    public employee(String arg1,int arg2,String arg3,String arg4){
            super(arg1,arg2,arg3);
        employee_attr=arg4;
     }
}

package example;
import step.*;
import example_type.*;
public class student extends person{
// field: student
    public String student_attr ;
```

```
    public student(){ }
    public student(String arg1,int arg2,String arg3,String arg4){
        super(arg1,arg2,arg3);
        student_attr=arg4;
      }
}

package example;
import step.*;
import example_type.*;
public class employee_and_student extends person{
// field: employee
    public String employee_attr ;
// field: student
    public String student_attr ;
    public employee_and_student(){ }
    public employee_and_student(String arg1,String arg2,int arg3,String
arg4,String arg5){
        super(arg2,arg3,arg4);
        employee_attr=arg1;
          student_attr=arg5;
      }
}
```

3.6.5 Part21 形式データの Java プログラム変換

Part21 形式データを Java のコンストラクタに変換するにあたっての対応を以下に検討した。

- ヘッダー節は Java のコメントにする。
- データ節の各文は、Java のコンストラクタ呼び出しに変換する。

すなわち、

#自然数　=　エンティティ名（属性 1、属性 2、…）；

を

クラス名　e 自然数　=　new クラス名（引数 1、引数 2、…）；

に変換する。

Part21 形式データでは、データ節の中での各文の出現順序に制約はないが、Java プログラムでは、参照されるインスタンスは事前に記述されている必要があるので、各文を実行可能な順番に並べ直す必要があることに注意が必要である。

以下に前節で取り上げた例がどのように Java のコンストラクタに対応つけられるかを示す。

（例 1）単純データ型のマッピング例

```
ENTITY widget;
 i1:INTEGER;
 s2:STRING;
 l :LOGICAL;
 b :BOOLEAN;
 r2:REAL;
END_ENTITY;
```

<Part21 形式表現>

```
#10 = WIDGET(10,'XYZABCDEFG',.T.,.F.,5.4321) ;
```

<Java 表現>

```
widget e10=new widget(10,"xyzabcdefg"new logical(".t."),false,5.4321) ;
```

（例 2）LIST の例

```
ENTITY widget;
 att1 : LIST[0:?] OF INTEGER;
 att2 : LIST[1:?] OF INTEGER;
 att3 : OPTIONAL LIST [1:?] OF INTEGER ;
```

```
  att4 : REAL;
END_ENTITY;
```

<Part21 形式表現>

```
#20 = WIDGET((),(10,20,30),$,12.5);
```

<Java 表現>

```
widget e20=new widget(new int[]{},new int[]{10,20,30},null,12.5);
```

（例 3）ARRAY の例

```
ENTITY widget;
  att1 ： ARRAY [-1:3] OF INTEGER;
  att2 ： ARRAY [1:5] OF OPTIONAL INTEGER ;
  att3 ： ARRAY [1:2] OF ARRAY [1:3] OF INTEGER ;
END_ENTITY;
```

<Part21 形式表現>

```
#30 = WIDGET((1,2,3,4,5),(1,$,3,4,5),((1,2,3),(10,20,30)));
```

<Java 表現>

```
widget e30=new widget(new int[]{1,2,3,4,5},new int[]{1,0,3,4,5},new int[][]{{1,2,3},{10,20,30}});
```

（注）個々の配列要素についてのオプショナル指定はサポートしていないので、個別に値（この場合は 0）を与える必要がある。

（例 4）SET の例

```
ENTITY widget;
```

```
 A : SET OF INTEGER ;
END_ENTITY;
```

<Part21 形式表現>

```
#40 = WIDGET((1,2,3)) ;
#41 = WIDGET((0,0,2));
#42 = WIDGET((0,0,3));
```

<Java 表現>

```
widget e40=new widget(new int[]{1,2,3});
widget e41=new widget(new int[]{0,0,2});
widget e42=new widget(new int[]{0,0,3});
```

（例 5）BAG の例

```
ENTITY widget;
 b : BAG OF INTEGER;
END_ENTITY;
```

<Part21 形式表現>

```
#50 = WIDGET((1,2,3));
#51 = WIDGET((10,11,12,0,14,15,16));
```

<Java 表現>

```
widget e50=new widget(new int[]{1,2,3});
widget e51=new widget(new int[]{10,11,12,0,14,15,16});
```

（例 6）エンティティの例

```
ENTITY part ;
```

```
 att1： LIST [0:3] OF REAL;
 size: REAL ;
END_ENTITY;

ENTITY widget;
 id : NUMBER;
 pp : part;
END_ENTITY;
```

<Part21 形式表現>

```
#60 = PART((1.1,2.2,3.3),100.23) ;
#61 = WIDGET(10,#60);
```

<Java 表現>

```
part e60=new part(new double[]{1.1,2.2,3.3},100.23);
widget e61=new widget(10,e60);
```

（例 7）定義型（TYPE）の例

```
TYPE
 type1 = INTEGER;
END_TYPE;

TYPE
 type2=LIST [1:2] OF REAL;
END_TYPE;

ENTITY widget;
 att : REAL;
 att1:type1;
```

```
  att2:type2;
END_ENTITY;
```

<Part21 形式表現>

```
#70 = WIDGET(10.5,3,(11.1,22.2)) ;
```

<Java 表現>

```
widget e70=new widget(10.5,3,new double[]{11.1,22,2});
```

（例 8）列挙型の例

```
TYPE
  color= ENUMERATION OF (red,white,green,blue);
END_TYPE;

ENTITY widget;
  p_color : color;
END_ENTITY ;
```

<Part21 形式表現>

```
#80 =  WIDGET(.RED.) ;
```

<Java 表現>

```
widget e80=new widget(new color(".red."));
```

（例 9）セレクト型の例

```
TYPE
  size = SELECT(area,radius);
END_TYPE;
```

```
TYPE
 area : REAL;
END_TYPE;

TYPE
 radius : REAL;
END_TYPE;

ENTITY circle;
 ll : size;
END_ENTITY;
```

<Part21 形式表現>

値 30.3 の AREA で CIRCLE は表わされている。

```
#90 = CIRCLE(AREA(30.3));
```

<Java 表現>

```
circle e90=new circle(new Double(30.3));
```

3.6.6 出力する XML 文書の設計

ここでは、文書型定義を伴わないウェルフォームドXML文書作成のためのXML文書設計を説明する。

ウェルフォームドXML文書作成のためにはXML宣言とXMLデータが必要である。

- XML宣言

  XML宣言では、XMLの規格のバージョンと文字コードを指定する。

  XML規格のバージョンは現在 1.0 であるので、version 属性で 1.0 を指定する。

  文字コードはシフトＪＩＳコードを使用するので、encoding 属性で"Shift-JIS"を指定する。

従って

<?xml version="1.0" encoding="Shift_JIS" ?>

と記述する。

- ＸＭＬデータ

ＸＭＬデータは XML の基本単位である要素からなる。

要素は開始タグで始まり、内容（例えば文字データ）、終了タグと続く。開始タグと終了タグが必ず対になっていることが必要である。開始タグと終了タグに書かれる名前を要素名という。開始タグと終了タグの間の内容を持たないものを空要素と呼び＜要素名／＞と記述する。

－ルート要素

ＸＭＬデータ内の要素の親となる要素をルート要素と呼び必ず一つ必要である。ここでは、その要素名を Model とする。

－階層構造の定義

Model は複数のエンティティから構成され、各エンティティはその複数の属性から構成されるので図 3-17に示す階層構造を持たせることとする。

ＸＭＬでは、兄弟関係をあらわすときは要素を並列に並べて記述し、親子関係は要素の中に要素を置いて、入れ子構造を作り記述する。

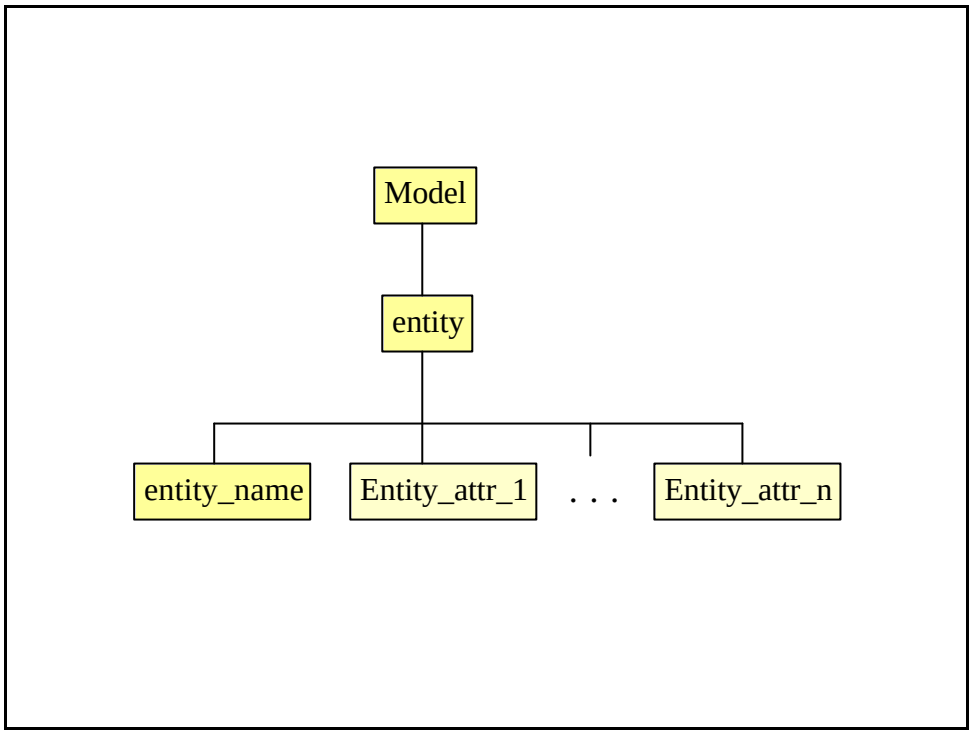


図 3-17 Model の階層構造

（例）application_context **エンティティ**

```
<entity>
 <entity_name>application_context</entity_name>
 <application>configuration controlled 3d designs of mechanical parts and assemblies</application>
 </entity>
```

－要素属性の定義

エンティティの属性の中には、他のエンティティのインスタンスを参照するものがあるので、各エンティティインスタンスを識別する仕掛けが必要になる。そこでエンティティ要素にエンティティインスタンスの名前を属性として与えることとする。属性は対象となるエンティティの要素名の後に、属性名と属性値を等号で結んで与える。また、属性値はダブルクオーテーションマークで囲む。

（例）application_context エンティティを参照する application_protocol_definition エンティティ

この例ではエンティティ要素に属性名 id の属性値を与えて、エンティティの参照関係を表現する。

```
- <entity id="e10">
  <entity_name>application_context</entity_name>
  <application>configuration controlled 3d designs of mechanical parts and assemblies</application>
  </entity>
- <entity id="e20">
  <entity_name>application_protocol_definition</entity_name>
  <status>international standard</status>

<application_interpreted_model_schema_name>config_control_design</application_interpreted_model_schema_name>
  <application_protocol_year>1994</application_protocol_year>
  <application>e10</application>
  </entity>
```

3.6.7　ＳＴＥＰモデルからのＸＭＬ出力

Java では、インスタンスのクラス名はもちろんのこと、フィールド名、フィールドの型、そのインスタンスにおけるフィールドの値をＡＰＩ機能を使用して取り出すことができる。すなわち、エンティティが持つ、明示属性に関する情報を、Java クラスに変換した後も、その Java クラス自体が保持していることになるため、実装が非常に容易になる。

このＡＰＩを利用すると、ＳＴＥＰモデルからのＸＭＬ出力は、ＳＴＥＰモデルの内部管理で紹介したチェインをたどりながらインスタンスの各フィールドの持つ名称と値を前節の形式で出力することで実現できる。

3.7　ＸＭＬ出力プログラムの実行

ここでは、図 3-18のような立方体ソリッドモデルを例題としてＸＭＬ形式で出力するプログラムの実行手順を説明する。

立方体モデルは、ＡＰ２０３（形態管理された設計）に基づき表わすことにする。

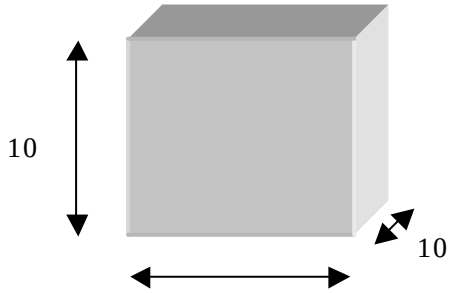


図 3-18 立方体ソリッドモデル

手順は次の 3 段階で行なう。

①立方体モデルを定義するＡＰ２０３準拠の Part21 形式データを用意する。

②Part21 データ処理部により、model.java プログラムを生成する。

③ＸＭＬ出力プログラムに②の model.java を組み込んで実行し、ウェルフォームドＸＭＬ文書を出力して Internet Explorer で表示する。

①のＡＰ２０３に準拠した Part21 形式データは、市販ＣＡＤソフトのＳＴＥＰ出力機能を使用すれば手に入れることができる。ここでは、参考文献（1）にあるものを使用する。

用意した Part21 形式データをＡＰ２０３に適合させた上で作業を進めたい場合は、NIST Expresso を利用するとよい。

Expresso は作成した Part21 形式データを検証するために使用することができる。 このプログラムはＥＸＰＲＥＳＳスキーマを内部表現（ＬＩＳＰに基づいた）に翻訳し、クラス定義、ＥＸＰＲＥＳＳタイプのための関数及びルールを作成して、この表現をロードする。ＥＸＰＲＥＳＳに対応する Part21 形式データをパート 21 ファイルからロードするか、あるいはエディタ画面で作成すると、NIST Expresso はＥＸＰＲＥＳＳスキーマの中で規定された制約でデータを検証する。

NIST Expresso に関する情報は下記のＵＲＬで得ることができる。

http://www.mel.nist.gov/msidstaff/denno/nist-expresso.html

3.7.1　立方体ソリッドモデルの Part21 形式データ

本項では、立方体ソリッドモデルの Part21 形式データを示す。

(1) Part21 形式データの内容

- アプリケーションコンテキスト関連

```
#10 = APPLICATION_CONTEXT('Configuration controlled 3D designs');
#20 = APPLICATION_PROTOCOL_DEFINITION('international standard','c
onfig_control_design', 1994, #10);
#30 = MECHANICAL_CONTEXT(' ', #10, 'mechanical');
#40 = DESIGN_CONTEXT(' ', #10, 'design');
```

- 製品定義関連

product を含む製品定義関連のデータは次の通りである。ここで、product データは mechanical_context データを、product_definition_formation_with_specified_source は design_context を参照している。

```
#50 = PRODUCT('Cube1', ' ', ' ', (#30));
#60 = PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURC
E('Cube1.1', ' '   , #50, .NOT_KNOWN.);
#70 = PRODUCT_DEFINITION(' ', ' ', #60, #40);
#80 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#50));
#90 = PRODUCT_DEFINITION_SHAPE(' ', ' ', #70);
```

- 機密区分関係

product_definition_formation_with_specified_source データに対して、機密区分を設定する。機密区分レベルは、'unclassified'とする。

```
#100 = SECURITY_CLASSIFICATION_LEVEL('unclassified');
#110 = SECURITY_CLASSIFICATION(' ', ' ', #100);
```

#120 = CC_DESIGN_SECURITY_CLASSIFICATION(#110, (#60));

- **日付と時間**

  日付と時間は、複数のデータに対してある役割を持って割り当てられる。

#130 = COORDINATED_UNIVERSAL_TIME_OFFSET(9, 0, .AHEAD.);

#140 = LOCAL_TIME(15, 15, 3.0E+001, #130);

#150 = CALENDAR_DATE(1998, 17, 2);

#160 = DATE_AND_TIME(#150, #140);

- **機密区分への時間の割り当て**

  security_classification **データに対して、**date_and_time **データを割り当てる。役割は、**'classification_date'**と設定する。**

#170 = DATE_TIME_ROLE('classification_date');

#180 = CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#160, #170, (#110));

- **製品定義への時間の割り当て**

  product_definition **に時間を割り当てる。役割は、**'creation_date'**と設定する。**

#190 = DATE_TIME_ROLE('creation_date');

#200 = CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#160, #190, (#70))

- **人と組織の定義**

  人と組織は、複数のデータにある役割を持って割り当てられる。

#210 = PERSON('12345', 'Yamada', 'Taro', $, $, $);

#220 = ORGANIZATION('67890', 'KGT Inc.', ' ');

#230 = PERSON_AND_ORGANIZATION(#210, #220);

- **製品への人と組織の役割の割り当て**

  product **データに** person_and_organaization **データを割り当てる。役割は、**'design_owner'**とする。**

#240 = PERSON_AND_ORGANIZATION_ROLE('design_owner');

#250 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#230, #240, (#50));

- **製品定義バージョンと製品定義への人と組織の役割の割り当て**

product_definition_formation_with_specified_source データと product_definition データに person_and_organization データを割り当てる。役割は'creator'とする。

#260 = PERSON_AND_ORGANIZATION_ROLE('creator');

#270 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#230, #260, (#60, #70));

- 製品定義バージョンへの人と組織の役割の割り当て

  product_definition_formation_with_specified_source データに person_and_organization データを割り当てる。役割は上記と異なり、'design_supplier'とする。

  #280 = PERSON_AND_ORGANIZATION_ROLE('design_supplier');

  #290 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#230, #280, (#60));

- 機密区分への人と組織の役割の割り当て

  security_classification データに、person_and_organization データを割り当て、'classification_officer'という役割を与える。

  #300 = PERSON_AND_ORGANIZATION_ROLE('classification_officer');

  #310 = CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#230, #300, (#110));

- 承認関連

  product_definition_formation_with_specified_source 、 product_definition 、security_classification の各データに、approval データを割り当てる。

  #320 = APPROVAL_STATUS('not_yet_approved');

  #330 = APPROVAL(#320, ' ');

  #340 = APPROVAL_DATE_TIME(#160, #330);

  #350 = APPROVAL_ROLE('approver');

  #360 = APPROVAL_PERSON_ORGANIZATION(#230, #330, #350);

  #370 = CC_DESIGN_APPROVAL(#330, (#110, #60, #70));

- length_unit データ作成

  length_unit を SI 単位系で定義する。

#380 = (GEOMETRIC_REPRESENTATION_CONTEXT(3) GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#450)) GLOBAL_UNIT_ASSIGNED_CONTEXT((#390, #400, #440)) REPRESENTATION_CONTEXT('', ''));

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

#400 = (NAMED_UNIT(*) PLANE_ANGLE_UNIT() SI_UNIT($, .RADIAN.));

#440 = (NAMED_UNIT(*) SI_UNIT($, .STERADIAN.) SOLID_ANGLE_UNIT());

#450 = UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(1.0E-004), #390, 'MODEL_ACCURACY', 'Maximum Tolerance applied to Model');

- **頂点**

#460 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 0.0E+000));

#470 = VERTEX_POINT('V1', #460);

#480 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 0.0E+000));

#490 = VERTEX_POINT('V2', #480);

#500 = CARTESIAN_POINT('', (1.0E+001, 1.0E+001, 0.0E+000));

#510 = VERTEX_POINT('V3', #500);

#520 = CARTESIAN_POINT('', (0.0E+000, 1.0E+001, 0.0E+000));

#530 = VERTEX_POINT('V4', #520);

#540 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 1.0E+001));

#550 = VERTEX_POINT('V5', #540);

#560 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 1.0E+001));

#570 = VERTEX_POINT('V6', #560);

#580 = CARTESIAN_POINT('', (1.0E+001, 1.0E+001, 1.0E+001));

#590 = VERTEX_POINT('V7', #580);

#600 = CARTESIAN_POINT('', (0.0E+000, 1.0E+001, 1.0E+001));

#610 = VERTEX_POINT('V8', #600);

- **稜線**

#620 = DIRECTION('', (1.0E+001, 0.0E+000, 0.0E+000));

#630 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 0.0E+000));

#640 = VECTOR('', #620, 1.0E+001);

#650 = LINE('L1', #630, #640);

#660 = EDGE_CURVE('E1', #470, #490, #650, .T.);

#670 = DIRECTION('', (0.0E+000, 1.0E+001, 0.0E+000));

#680 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 0.0E+000));

#690 = VECTOR('', #670, 1.0E+001);

#700 = LINE('L2', #680, #690);

#710 = EDGE_CURVE('E2', #490, #510, #700, .T.);

#720 = DIRECTION('', (1.0E+001, 0.0E+000, 0.0E+000));

#730 = CARTESIAN_POINT('', (0.0E+000, 1.0E+001, 0.0E+000));

#740 = VECTOR('', #720, 1.0E+001);

#750 = LINE('L3', #730, #740);

#760 = EDGE_CURVE('E3', #530, #510, #750, .T.);

#770 = DIRECTION('', (0.0E+000, 1.0E+001, 0.0E+000));

#780 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 0.0E+000));

#790 = VECTOR('', #770, 1.0E+001);

#800 = LINE('L4', #780, #790);

#810 = EDGE_CURVE('E4', #470, #530, #800, .T.);

#820 = DIRECTION('', (0.0E+000, 0.0E+000, 1.0E+001));

#830 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 0.0E+000));

#840 = VECTOR('', #820, 1.0E+001);

#850 = LINE('L5', #830, #840);

#860 = EDGE_CURVE('E5', #470, #550, #850, .T.);

#870 = DIRECTION('', (0.0E+000, 0.0E+000, 1.0E+001));

#880 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 0.0E+000));

#890 = VECTOR('', #870, 1.0E+001);

#900 = LINE('L6', #880, #890);

#910 = EDGE_CURVE('E6', #490, #570, #900, .T.);

#920 = DIRECTION('', (0.0E+000, 0.0E+000, 1.0E+001));

#930 = CARTESIAN_POINT('', (1.0E+001, 1.0E+001, 0.0E+000));

#940 = VECTOR('', #920, 1.0E+001);

#950 = LINE('L7', #930, #940);

#960 = EDGE_CURVE('E7', #510, #590, #950, .T.);

#970 = DIRECTION('', (0.0E+000, 0.0E+000, 1.0E+001));

#980 = CARTESIAN_POINT('', (0.0E+000, 1.0E+001, 0.0E+000));

#990 = VECTOR('', #970, 1.0E+001);

#1000 = LINE('L8', #980, #990);

#1010 = EDGE_CURVE('E8', #530, #610, #1000, .T.);

#1020 = DIRECTION('', (1.0E+001, 0.0E+000, 0.0E+000));

#1030 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 1.0E+001));

#1040 = VECTOR('', #1020, 1.0E+001);

#1050 = LINE('L9', #1030, #1040);

#1060 = EDGE_CURVE('E9', #550, #570, #1050, .T.);

#1070 = DIRECTION('', (0.0E+000, 1.0E+001, 0.0E+000));

#1080 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 1.0E+001));

#1090 = VECTOR('', #1070, 1.0E+001);

#1100 = LINE('L10', #1080, #1090);

#1110 = EDGE_CURVE('E10', #570, #590, #1100, .T.);

#1120 = DIRECTION('', (1.0E+001, 0.0E+000, 0.0E+000));

#1130 = CARTESIAN_POINT('', (0.0E+000, 1.0E+001, 1.0E+001));

#1140 = VECTOR('', #1120, 1.0E+001);

#1150 = LINE('L11', #1130, #1140);

#1160 = EDGE_CURVE('E11', #610, #590, #1150, .T.);

#1170 = DIRECTION('', (0.0E+000, 1.0E+001, 0.0E+000));

#1180 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 1.0E+001));

#1190 = VECTOR('', #1170, 1.0E+001);

#1200 = LINE('L12', #1180, #1190);

#1210 = EDGE_CURVE('E12', #550, #610, #1200, .T.);

- **シェル**

#1220 = CLOSED_SHELL('CS', (#1340, #1460, #1580, #1700, #1820, #1940));

- **面**

#1230 = EDGE_LOOP('EL1', (#1240, #1250, #1260, #1270));

#1240 = ORIENTED_EDGE('OE11', *, *, #660, .F.);

#1250 = ORIENTED_EDGE('OE12', *, *, #810, .T.);

#1260 = ORIENTED_EDGE('OE13', *, *, #760, .T.);

#1270 = ORIENTED_EDGE('OE14', *, *, #710, .F.);

#1280 = FACE_OUTER_BOUND('FOB1', #1230, .T.);

#1290 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 0.0E+000));

#1300 = DIRECTION('', (0.0E+000, 0.0E+000, -1.0E+002));

#1310 = DIRECTION('', (-1.0E+001, 0.0E+000, 0.0E+000));

#1320 = AXIS2_PLACEMENT_3D('', #1290, #1300, #1310);

#1330 = PLANE('P1', #1320);

#1340 = ADVANCED_FACE('AF1', (#1280), #1330, .T.);

#1350 = EDGE_LOOP('EL2', (#1360, #1370, #1380, #1390));

#1360 = ORIENTED_EDGE('OE21', *, *, #660, .T.);

#1370 = ORIENTED_EDGE('OE22', *, *, #910, .T.);

#1380 = ORIENTED_EDGE('OE23', *, *, #1060, .F.);

#1390 = ORIENTED_EDGE('OE24', *, *, #860, .F.);

#1400 = FACE_OUTER_BOUND('FOB2', #1350, .T.);

#1410 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 0.0E+000));

#1420 = DIRECTION('', (0.0E+000, -1.0E+002, 0.0E+000));

#1430 = DIRECTION('', (1.0E+001, 0.0E+000, 0.0E+000));

#1440 = AXIS2_PLACEMENT_3D('', #1410, #1420, #1430);

#1450 = PLANE('P2', #1440);

#1460 = ADVANCED_FACE('AF2', (#1400), #1450, .T.);

#1470 = EDGE_LOOP('EL3', (#1480, #1490, #1500, #1510));

#1480 = ORIENTED_EDGE('OE31', *, *, #710, .T.);

#1490 = ORIENTED_EDGE('OE32', *, *, #960, .T.);

#1500 = ORIENTED_EDGE('OE33', *, *, #1110, .F.);

#1510 = ORIENTED_EDGE('OE34', *, *, #910, .F.);

#1520 = FACE_OUTER_BOUND('FOB3', #1470, .T.);

#1530 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 0.0E+000));

#1540 = DIRECTION('', (1.0E+002, 0.0E+000, 0.0E+000));

#1550 = DIRECTION('', (0.0E+000, 1.0E+001, 0.0E+000));

#1560 = AXIS2_PLACEMENT_3D('', #1530, #1540, #1550);

#1570 = PLANE('P3', #1560);

#1580 = ADVANCED_FACE('AF3', (#1520), #1570, .T.);

#1590 = EDGE_LOOP('EL4', (#1600, #1610, #1620, #1630));

#1600 = ORIENTED_EDGE('OE41', *, *, #760, .F.);

#1610 = ORIENTED_EDGE('OE42', *, *, #1010, .T.);

#1620 = ORIENTED_EDGE('OE43', *, *, #1160, .T.);

#1630 = ORIENTED_EDGE('OE44', *, *, #960, .F.);

#1640 = FACE_OUTER_BOUND('FOB4', #1590, .T.);

#1650 = CARTESIAN_POINT('', (1.0E+001, 1.0E+001, 0.0E+000));

#1660 = DIRECTION('', (0.0E+000, 1.0E+002, 0.0E+000));

#1670 = DIRECTION('', (-1.0E+001, 0.0E+000, 0.0E+000));

#1680 = AXIS2_PLACEMENT_3D('', #1650, #1660, #1670);

#1690 = PLANE('P4', #1680);

#1700 = ADVANCED_FACE('AF4', (#1640), #1690, .T.);

#1710 = EDGE_LOOP('EL5', (#1720, #1730, #1740, #1750));

#1720 = ORIENTED_EDGE('OE51', *, *, #810, .F.);

#1730 = ORIENTED_EDGE('OE52', *, *, #860, .T.);

#1740 = ORIENTED_EDGE('OE53', *, *, #1210, .T.);

#1750 = ORIENTED_EDGE('OE54', *, *, #1010, .F.);

#1760 = FACE_OUTER_BOUND('FOB5', #1710, .T.);

#1770 = CARTESIAN_POINT('', (0.0E+000, 1.0E+001, 0.0E+000));

#1780 = DIRECTION('', (-1.0E+002, 0.0E+000, 0.0E+000));

#1790 = DIRECTION('', (0.0E+000, -1.0E+001, 0.0E+000));

#1800 = AXIS2_PLACEMENT_3D('', #1770, #1780, #1790);

#1810 = PLANE('P5', #1800);

#1820 = ADVANCED_FACE('AF5', (#1760), #1810, .T.);

#1830 = EDGE_LOOP('EL6', (#1840, #1850, #1860, #1870));

#1840 = ORIENTED_EDGE('OE61', *, *, #1060, .T.);

#1850 = ORIENTED_EDGE('OE62', *, *, #1110, .T.);

#1860 = ORIENTED_EDGE('OE63', *, *, #1160, .F.);

#1870 = ORIENTED_EDGE('OE64', *, *, #1210, .F.);

#1880 = FACE_OUTER_BOUND('FOB6', #1830, .T.);

#1890 = CARTESIAN_POINT('', (0.0E+000, 0.0E+000, 1.0E+001));

#1900 = DIRECTION('', (0.0E+000, 0.0E+000, 1.0E+002));

#1910 = DIRECTION('', (1.0E+001, 0.0E+000, 0.0E+000));

#1920 = AXIS2_PLACEMENT_3D('', #1890, #1900, #1910);

#1930 = PLANE('P6', #1920);

#1940 = ADVANCED_FACE('AF6', (#1880), #1930, .T.);

- **ソリッド**

#1950 = MANIFOLD_SOLID_BREP('MSB', #1220);

#1960 = ADVANCED_BREP_SHAPE_REPRESENTATION('ABSR', (#1950), #380);

#1970 = SHAPE_DEFINITION_REPRESENTATION(#90, #1960);

(2) **追加した** ENTITY

この Part21 **形式データは** AND OR **継承を含むため、以下にしめす** ENTITY **を** AP203 **ロングフォームに追加して処理を行う。**

```
ENTITY REPRESENTATION_CONTEXT1
 SUBTYPE OF (GEOMETRIC_REPRESENTATION_CONTEXT)
 CONSTRUCT OF (GEOMETRIC_REPRESENTATION_CONTEXT,
          GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT,
          GLOBAL_UNIT_ASSIGNED_CONTEXT,
          REPRESENTATION_CONTEXT);
END_ENTITY;
```

```
ENTITY NAMED_UNIT1
 SUBTYPE OF (NAMED_UNIT)
 CONSTRUCT OF (LENGTH_UNIT,
              NAMED_UNIT,
              SI_UNIT);
END_ENTITY;

ENTITY NAMED_UNIT2
 SUBTYPE OF (NAMED_UNIT)
 CONSTRUCT OF (NAMED_UNIT,
             PLANE_ANGLE_UNIT,
              SI_UNIT);
END_ENTITY;

ENTITY NAMED_UNIT3
 SUBTYPE OF (NAMED_UNIT)
 CONSTRUCT OF (NAMED_UNIT,
              SI_UNIT,
            SOLID_ANGLE_UNIT);
END_ENTITY;

ENTITY NAMED_UNIT4
 SUBTYPE OF (NAMED_UNIT)
 CONSTRUCT OF (CONVERSION_BASED_UNIT,
              NAMED_UNIT,
            PLANE_ANGLE_UNIT);
END_ENTITY;
```

3.7.2 model プログラム

立方体モデルを記述した Part21 形式データを Part21 形式データ処理部に掛けて、model.java プログラムを生成する。

この model.java プログラムを以下に示す。

プログラムとして実行するために、Part21 形式データと比べて、e380、e1350、e1470、e1590、e1710、e1830、e1220 については順番が入れ替わっていることが確認できる。

なお original というメソッドは、内部で管理されているインスタンスの番号から、Part21 形式データの#番号への変換テーブルである。

```
package main;
import step.*;
import config_control_design.*;
import config_control_design_type.*;
public class model {
public static void expression() {
//ISO-10303-21;
//HEADER;
//DATA;
application_context e10=new application_context("configuration controlled 3d designs of mechanical parts and assemblies");
application_protocol_definition e20=new application_protocol_definition("international standard","config_control_design",1994,e10);
mechanical_context e30=new mechanical_context(" ",e10,"mechanical");
design_context e40=new design_context(" ",e10,"design");
product e50=new product("cube1"," "," ",new product_context[]{e30});
product_definition_formation_with_specified_source e60=new product_definition_formation_with_specified_source("cube1.1"," ",e50,new source(".not_known."));
product_definition e70=new product_definition(" "," ",e60,e40);
```

```
   product_related_product_category                                    e80=new
product_related_product_category("detail",null,new product[]{e50});
   product_definition_shape e90=new product_definition_shape(" "," ",e70);
   security_classification_level e100=new security_classification_level("unclassified");
   security_classification e110=new security_classification(" "," ",e100);
   cc_design_security_classification                                   e120=new
cc_design_security_classification(e110,new classified_item[]{e60});
   coordinated_universal_time_offset                                   e130=new
coordinated_universal_time_offset(9,0,new ahead_or_behind(".ahead."));
   local_time e140=new local_time(20,15,3.0e+001,e130);
   calendar_date e150=new calendar_date(1998,17,2);
   date_and_time e160=new date_and_time(e150,e140);
   date_time_role e170=new date_time_role("classification_date");
   cc_design_date_and_time_assignment                                  e180=new
cc_design_date_and_time_assignment(e160,e170,new date_time_item[]{e110});
   date_time_role e190=new date_time_role("creation_date");
   cc_design_date_and_time_assignment                                  e200=new
cc_design_date_and_time_assignment(e160,e190,new date_time_item[]{e70});
   person e210=new person("12345","yamada","taro",null,null,null);
   organization e220=new organization("67890","kgt inc."," ");
   person_and_organization e230=new person_and_organization(e210,e220);
   person_and_organization_role                                        e240=new
person_and_organization_role("design_owner");
   cc_design_person_and_organization_assignment                        e250=new
cc_design_person_and_organization_assignment(e230,e240,new
person_organization_item[]{e50});
   person_and_organization_role e260=new person_and_organization_role("creator");
   cc_design_person_and_organization_assignment                        e270=new
cc_design_person_and_organization_assignment(e230,e260,new
person_organization_item[]{e60,e70});
```

```
   person_and_organization_role                                    e280=new
person_and_organization_role("design_supplier");
   cc_design_person_and_organization_assignment                    e290=new
cc_design_person_and_organization_assignment(e230,e280,new
person_organization_item[]{e60});
   person_and_organization_role                                    e300=new
person_and_organization_role("classification_officer");
   cc_design_person_and_organization_assignment                    e310=new
cc_design_person_and_organization_assignment(e230,e300,new
person_organization_item[]{e110});
   approval_status e320=new approval_status("not_yet_approved");
   approval e330=new approval(e320," ");
   approval_date_time e340=new approval_date_time(e160,e330);
   approval_role e350=new approval_role("approver");
   approval_person_organization                                    e360=new
approval_person_organization(e230,e330,e350);
   cc_design_approval e370=new cc_design_approval(e330,new
approved_item[]{e110,e60,e70});
   named_unit1 e390=new named_unit1(null,new si_prefix(".milli."),new
si_unit_name(".metre."));
   named_unit2 e400=new named_unit2(null,null,new si_unit_name(".radian."));
   named_unit3 e440=new named_unit3(null,null,new si_unit_name(".steradian."));
   uncertainty_measure_with_unit e450=new uncertainty_measure_with_unit(new
Double(1.0e-004),e390,"model_accuracy","maximum tolerance applied to model");
   representation_context1 e380=new representation_context1(3,new
uncertainty_measure_with_unit[]{e450},new unit[]{e390,e400,e440},"","");
   cartesian_point e460=new cartesian_point("",new
double[]{0.0e+000,0.0e+000,0.0e+000});
   vertex_point e470=new vertex_point("v1",e460);
   cartesian_point e480=new cartesian_point("",new
```

```
double[]{1.0e+001,0.0e+000,0.0e+000});
   vertex_point e490=new vertex_point("v2",e480);
   cartesian_point                e500=new                cartesian_point("",new
double[]{1.0e+001,1.0e+001,0.0e+000});
   vertex_point e510=new vertex_point("v3",e500);
   cartesian_point                e520=new                cartesian_point("",new
double[]{0.0e+000,1.0e+001,0.0e+000});
   vertex_point e530=new vertex_point("v4",e520);
   cartesian_point                e540=new                cartesian_point("",new
double[]{0.0e+000,0.0e+000,1.0e+001});
   vertex_point e550=new vertex_point("v5",e540);
   cartesian_point                e560=new                cartesian_point("",new
double[]{1.0e+001,0.0e+000,1.0e+001});
   vertex_point e570=new vertex_point("v6",e560);
   cartesian_point                e580=new                cartesian_point("",new
double[]{1.0e+001,1.0e+001,1.0e+001});
   vertex_point e590=new vertex_point("v7",e580);
   cartesian_point                e600=new                cartesian_point("",new
double[]{0.0e+000,1.0e+001,1.0e+001});
   vertex_point e610=new vertex_point("v8",e600);
   direction e620=new direction("",new double[]{1.0e+001,0.0e+000,0.0e+000});
   cartesian_point                e630=new                cartesian_point("",new
double[]{0.0e+000,0.0e+000,0.0e+000});
   vector e640=new vector("",e620,1.0e+001);
   line e650=new line("l1",e630,e640);
   edge_curve e660=new edge_curve("e1",e470,e490,e650,true);
   direction e670=new direction("",new double[]{0.0e+000,1.0e+001,0.0e+000});
   cartesian_point                e680=new                cartesian_point("",new
double[]{1.0e+001,0.0e+000,0.0e+000});
   vector e690=new vector("",e670,1.0e+001);
```

```
line e700=new line("l2",e680,e690);
edge_curve e710=new edge_curve("e2",e490,e510,e700,true);
direction e720=new direction("",new double[]{1.0e+001,0.0e+000,0.0e+000});
cartesian_point e730=new cartesian_point("",new double[]{0.0e+000,1.0e+001,0.0e+000});
vector e740=new vector("",e720,1.0e+001);
line e750=new line("l3",e730,e740);
edge_curve e760=new edge_curve("e3",e530,e510,e750,true);
direction e770=new direction("",new double[]{0.0e+000,1.0e+001,0.0e+000});
cartesian_point e780=new cartesian_point("",new double[]{0.0e+000,0.0e+000,0.0e+000});
vector e790=new vector("",e770,1.0e+001);
line e800=new line("l4",e780,e790);
edge_curve e810=new edge_curve("e4",e470,e530,e800,true);
direction e820=new direction("",new double[]{0.0e+000,0.0e+000,1.0e+001});
cartesian_point e830=new cartesian_point("",new double[]{0.0e+000,0.0e+000,0.0e+000});
vector e840=new vector("",e820,1.0e+001);
line e850=new line("l5",e830,e840);
edge_curve e860=new edge_curve("e5",e470,e550,e850,true);
direction e870=new direction("",new double[]{0.0e+000,0.0e+000,1.0e+001});
cartesian_point e880=new cartesian_point("",new double[]{1.0e+001,0.0e+000,0.0e+000});
vector e890=new vector("",e870,1.0e+001);
line e900=new line("l6",e880,e890);
edge_curve e910=new edge_curve("e6",e490,e570,e900,true);
direction e920=new direction("",new double[]{0.0e+000,0.0e+000,1.0e+001});
cartesian_point e930=new cartesian_point("",new double[]{1.0e+001,1.0e+001,0.0e+000});
vector e940=new vector("",e920,1.0e+001);
```

```
line e950=new line("l7",e930,e940);
edge_curve e960=new edge_curve("e7",e510,e590,e950,true);
direction e970=new direction("",new double[]{0.0e+000,0.0e+000,1.0e+001});
cartesian_point e980=new cartesian_point("",new double[]{0.0e+000,1.0e+001,0.0e+000});
vector e990=new vector("",e970,1.0e+001);
line e1000=new line("l8",e980,e990);
edge_curve e1010=new edge_curve("e8",e530,e610,e1000,true);
direction e1020=new direction("",new double[]{1.0e+001,0.0e+000,0.0e+000});
cartesian_point e1030=new cartesian_point("",new double[]{0.0e+000,0.0e+000,1.0e+001});
vector e1040=new vector("",e1020,1.0e+001);
line e1050=new line("l9",e1030,e1040);
edge_curve e1060=new edge_curve("e9",e550,e570,e1050,true);
direction e1070=new direction("",new double[]{0.0e+000,1.0e+001,0.0e+000});
cartesian_point e1080=new cartesian_point("",new double[]{1.0e+001,0.0e+000,1.0e+001});
vector e1090=new vector("",e1070,1.0e+001);
line e1100=new line("l10",e1080,e1090);
edge_curve e1110=new edge_curve("e10",e570,e590,e1100,true);
direction e1120=new direction("",new double[]{1.0e+001,0.0e+000,0.0e+000});
cartesian_point e1130=new cartesian_point("",new double[]{0.0e+000,1.0e+001,1.0e+001});
vector e1140=new vector("",e1120,1.0e+001);
line e1150=new line("l11",e1130,e1140);
edge_curve e1160=new edge_curve("e11",e610,e590,e1150,true);
direction e1170=new direction("",new double[]{0.0e+000,1.0e+001,0.0e+000});
cartesian_point e1180=new cartesian_point("",new double[]{0.0e+000,0.0e+000,1.0e+001});
vector e1190=new vector("",e1170,1.0e+001);
```

line e1200=new line("l12",e1180,e1190);

edge_curve e1210=new edge_curve("e12",e550,e610,e1200,true);

oriented_edge e1240=new oriented_edge("oe11",null,null,e660,false);

oriented_edge e1250=new oriented_edge("oe12",null,null,e810,true);

oriented_edge e1260=new oriented_edge("oe13",null,null,e760,true);

oriented_edge e1270=new oriented_edge("oe14",null,null,e710,false);

edge_loop e1230=new edge_loop("el1",new oriented_edge[]{e1240,e1250,e1260,e1270});

face_outer_bound e1280=new face_outer_bound("fob1",e1230,true);

cartesian_point e1290=new cartesian_point("",new double[]{1.0e+001,0.0e+000,0.0e+000});

direction e1300=new direction("",new double[]{0.0e+000,0.0e+000,-1.0e+002});

direction e1310=new direction("",new double[]{-1.0e+001,0.0e+000,0.0e+000});

axis2_placement_3d e1320=new axis2_placement_3d("",e1290,e1300,e1310);

plane e1330=new plane("p1",e1320);

advanced_face e1340=new advanced_face("af1",new face_bound[]{e1280},e1330,true);

oriented_edge e1360=new oriented_edge("oe21",null,null,e660,true);

oriented_edge e1370=new oriented_edge("oe22",null,null,e910,true);

oriented_edge e1380=new oriented_edge("oe23",null,null,e1060,false);

oriented_edge e1390=new oriented_edge("oe24",null,null,e860,false);

edge_loop e1350=new edge_loop("el2",new oriented_edge[]{e1360,e1370,e1380,e1390});

face_outer_bound e1400=new face_outer_bound("fob2",e1350,true);

cartesian_point e1410=new cartesian_point("",new double[]{0.0e+000,0.0e+000,0.0e+000});

direction e1420=new direction("",new double[]{0.0e+000,-1.0e+002,0.0e+000});

direction e1430=new direction("",new double[]{1.0e+001,0.0e+000,0.0e+000});

axis2_placement_3d e1440=new axis2_placement_3d("",e1410,e1420,e1430);

plane e1450=new plane("p2",e1440);

advanced_face e1460=new advanced_face("af2",new face_bound[]{e1400},e1450,true);

```
oriented_edge e1480=new oriented_edge("oe31",null,null,e710,true);
oriented_edge e1490=new oriented_edge("oe32",null,null,e960,true);
oriented_edge e1500=new oriented_edge("oe33",null,null,e1110,false);
oriented_edge e1510=new oriented_edge("oe34",null,null,e910,false);
edge_loop e1470=new edge_loop("el3",new oriented_edge[]{e1480,e1490,e1500,e1510});
face_outer_bound e1520=new face_outer_bound("fob3",e1470,true);
cartesian_point e1530=new cartesian_point("",new double[]{1.0e+001,0.0e+000,0.0e+000});
direction e1540=new direction("",new double[]{1.0e+002,0.0e+000,0.0e+000});
direction e1550=new direction("",new double[]{0.0e+000,1.0e+001,0.0e+000});
axis2_placement_3d e1560=new axis2_placement_3d("",e1530,e1540,e1550);
plane e1570=new plane("p3",e1560);
advanced_face e1580=new advanced_face("af3",new face_bound[]{e1520},e1570,true);
oriented_edge e1600=new oriented_edge("oe41",null,null,e760,false);
oriented_edge e1610=new oriented_edge("oe42",null,null,e1010,true);
oriented_edge e1620=new oriented_edge("oe43",null,null,e1160,true);
oriented_edge e1630=new oriented_edge("oe44",null,null,e960,false);
edge_loop e1590=new edge_loop("el4",new oriented_edge[]{e1600,e1610,e1620,e1630});
face_outer_bound e1640=new face_outer_bound("fob4",e1590,true);
cartesian_point e1650=new cartesian_point("",new double[]{1.0e+001,1.0e+001,0.0e+000});
direction e1660=new direction("",new double[]{0.0e+000,1.0e+002,0.0e+000});
direction e1670=new direction("",new double[]{-1.0e+001,0.0e+000,0.0e+000});
axis2_placement_3d e1680=new axis2_placement_3d("",e1650,e1660,e1670);
plane e1690=new plane("p4",e1680);
advanced_face e1700=new advanced_face("af4",new face_bound[]{e1640},e1690,true);
oriented_edge e1720=new oriented_edge("oe51",null,null,e810,false);
oriented_edge e1730=new oriented_edge("oe52",null,null,e860,true);
```

oriented_edge e1740=new oriented_edge("oe53",null,null,e1210,true);

oriented_edge e1750=new oriented_edge("oe54",null,null,e1010,false);

edge_loop e1710=new edge_loop("el5",new oriented_edge[]{e1720,e1730,e1740,e1750});

face_outer_bound e1760=new face_outer_bound("fob5",e1710,true);

cartesian_point e1770=new cartesian_point("",new double[]{0.0e+000,1.0e+001,0.0e+000});

direction e1780=new direction("",new double[]{-1.0e+002,0.0e+000,0.0e+000});

direction e1790=new direction("",new double[]{0.0e+000,-1.0e+001,0.0e+000});

axis2_placement_3d e1800=new axis2_placement_3d("",e1770,e1780,e1790);

plane e1810=new plane("p5",e1800);

advanced_face e1820=new advanced_face("af5",new face_bound[]{e1760},e1810,true);

oriented_edge e1840=new oriented_edge("oe61",null,null,e1060,true);

oriented_edge e1850=new oriented_edge("oe62",null,null,e1110,true);

oriented_edge e1860=new oriented_edge("oe63",null,null,e1160,false);

oriented_edge e1870=new oriented_edge("oe64",null,null,e1210,false);

edge_loop e1830=new edge_loop("el6",new oriented_edge[]{e1840,e1850,e1860,e1870});

face_outer_bound e1880=new face_outer_bound("fob6",e1830,true);

cartesian_point e1890=new cartesian_point("",new double[]{0.0e+000,0.0e+000,1.0e+001});

direction e1900=new direction("",new double[]{0.0e+000,0.0e+000,1.0e+002});

direction e1910=new direction("",new double[]{1.0e+001,0.0e+000,0.0e+000});

axis2_placement_3d e1920=new axis2_placement_3d("",e1890,e1900,e1910);

plane e1930=new plane("p6",e1920);

advanced_face e1940=new advanced_face("af6",new face_bound[]{e1880},e1930,true);

closed_shell e1220=new closed_shell("cs",new face[]{e1340,e1460,e1580,e1700,e1820,e1940});

manifold_solid_brep e1950=new manifold_solid_brep("msb",e1220);

advanced_brep_shape_representation e1960=new

```
advanced_brep_shape_representation("absr",new representation_item[]{e1950},e380);
   shape_definition_representation                                    e1970=new
shape_definition_representation(e90,e1960);
   //ENDSEC;
   //END-//ISO-10303-21;
   tool.xml();
    }
   public static int original(int arg1) {
       int[]                                                              vlist=new
int[]{10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190,200,210,220,
230,240,250,260,270,280,290,300,310,320,330,340,350,360,370,390,400,440,450,380,46
0,470,480,490,500,510,520,530,540,550,560,570,580,590,600,610,620,630,640,650,660,6
70,680,690,700,710,720,730,740,750,760,770,780,790,800,810,820,830,840,850,860,870,
880,890,900,910,920,930,940,950,960,970,980,990,1000,1010,1020,1030,1040,1050,106
0,1070,1080,1090,1100,1110,1120,1130,1140,1150,1160,1170,1180,1190,1200,1210,124
0,1250,1260,1270,1230,1280,1290,1300,1310,1320,1330,1340,1360,1370,1380,1390,135
0,1400,1410,1420,1430,1440,1450,1460,1480,1490,1500,1510,1470,1520,1530,1540,155
0,1560,1570,1580,1600,1610,1620,1630,1590,1640,1650,1660,1670,1680,1690,1700,172
0,1730,1740,1750,1710,1760,1770,1780,1790,1800,1810,1820,1840,1850,1860,1870,183
0,1880,1890,1900,1910,1920,1930,1940,1220,1950,1960,1970,0,0};
        return vlist[arg1];
    }
   }
```

3.7.3 XMLリスト

model.java **プログラムを** XML **形式出力プログラムのメイン処理部に組み込み実行することにより、ウェルフォームド** XML **文書を出力できる。**

このため、model.java **プログラムの中で**

tool.xml();

を呼び出しをしている。

出力リストを見ると、要素名によりエンティティとインスタンス値が明確で読みやすいことが見て取れる。一方出力に要する行数は Part21 形式では 190 行であったのに対して、XMLの形式では 1170 行となり、容量的には大幅に大きいことが見てとれる。

<?xml version="1.0" encoding="Shift_JIS" ?>
- <Model>
- <entity id="e10">
<entity_name>application_context</entity_name>
<application>configuration controlled 3d designs of mechanical parts and assemblies</application>
</entity>
- <entity id="e20">
<entity_name>application_protocol_definition</entity_name>
<status>international standard</status>
<application_interpreted_model_schema_name>config_control_design</application_interpreted_model_schema_name>
<application_protocol_year>1994</application_protocol_year>
<application>e10</application>
</entity>
- <entity id="e30">
<entity_name>mechanical_context</entity_name>
<name />
<frame_of_reference>e10</frame_of_reference>
<discipline_type>mechanical</discipline_type>
</entity>
- <entity id="e40">
<entity_name>design_context</entity_name>
<name />
<frame_of_reference>e10</frame_of_reference>

<life_cycle_stage>design</life_cycle_stage>
</entity>
- <entity id="e50">
<entity_name>product</entity_name>
<id>cube1</id>
<name />
<description />
<frame_of_reference>e30</frame_of_reference>
</entity>
- <entity id="e60">
<entity_name>product_definition_formation_with_specified_source</entity_name>
<id>cube1.1</id>
<description />
<of_product>e50</of_product>
<make_or_buy>.not_known.</make_or_buy>
</entity>
- <entity id="e70">
<entity_name>product_definition</entity_name>
<id />
<description />
<formation>e60</formation>
<frame_of_reference>e40</frame_of_reference>
</entity>
- <entity id="e80">
<entity_name>product_related_product_category</entity_name>
<name>detail</name>
<description>null</description>
<products>e50</products>
</entity>
- <entity id="e90">

```
<entity_name>product_definition_shape</entity_name>
<name />
<description />
<definition>e70</definition>
</entity>
- <entity id="e100">
<entity_name>security_classification_level</entity_name>
<name>unclassified</name>
</entity>
- <entity id="e110">
<entity_name>security_classification</entity_name>
<name />
<purpose />
<security_level>e100</security_level>
</entity>
- <entity id="e120">
<entity_name>cc_design_security_classification</entity_name>
<assigned_security_classification>e110</assigned_security_classification>
<items>e60</items>
</entity>
- <entity id="e130">
<entity_name>coordinated_universal_time_offset</entity_name>
<hour_offset>9</hour_offset>
<minute_offset>0</minute_offset>
<sense>.ahead.</sense>
</entity>
- <entity id="e140">
<entity_name>local_time</entity_name>
<hour_component>20</hour_component>
<minute_component>15</minute_component>
```

<second_component>30.0</second_component>
<zone>e130</zone>
</entity>
- <entity id="e150">
<entity_name>calendar_date</entity_name>
<year_component>1998</year_component>
<day_component>17</day_component>
<month_component>2</month_component>
</entity>
- <entity id="e160">
<entity_name>date_and_time</entity_name>
<date_component>e150</date_component>
<time_component>e140</time_component>
</entity>
- <entity id="e170">
<entity_name>date_time_role</entity_name>
<name>classification_date</name>
</entity>
- <entity id="e180">
<entity_name>cc_design_date_and_time_assignment</entity_name>
<assigned_date_and_time>e160</assigned_date_and_time>
<role>e170</role>
<items>e110</items>
</entity>
- <entity id="e190">
<entity_name>date_time_role</entity_name>
<name>creation_date</name>
</entity>
- <entity id="e200">
<entity_name>cc_design_date_and_time_assignment</entity_name>

<assigned_date_and_time>e160</assigned_date_and_time>
<role>e190</role>
<items>e70</items>
</entity>
- <entity id="e210">
<entity_name>person</entity_name>
<id>12345</id>
<last_name>yamada</last_name>
<first_name>taro</first_name>
</entity>
- <entity id="e220">
<entity_name>organization</entity_name>
<id>67890</id>
<name>kgt inc.</name>
<description />
</entity>
- <entity id="e230">
<entity_name>person_and_organization</entity_name>
<the_person>e210</the_person>
<the_organization>e220</the_organization>
</entity>
- <entity id="e240">
<entity_name>person_and_organization_role</entity_name>
<name>design_owner</name>
</entity>
- <entity id="e250">
<entity_name>cc_design_person_and_organization_assignment</entity_name>
<assigned_person_and_organization>e230</assigned_person_and_organization>
<role>e240</role>
<items>e50</items>

```
  </entity>
- <entity id="e260">
  <entity_name>person_and_organization_role</entity_name>
  <name>creator</name>
  </entity>
- <entity id="e270">
  <entity_name>cc_design_person_and_organization_assignment</entity_name>
  <assigned_person_and_organization>e230</assigned_person_and_organization>
  <role>e260</role>
  <items>e60,e70</items>
  </entity>
- <entity id="e280">
  <entity_name>person_and_organization_role</entity_name>
  <name>design_supplier</name>
  </entity>
- <entity id="e290">
  <entity_name>cc_design_person_and_organization_assignment</entity_name>
  <assigned_person_and_organization>e230</assigned_person_and_organization>
  <role>e280</role>
  <items>e60</items>
  </entity>
- <entity id="e300">
  <entity_name>person_and_organization_role</entity_name>
  <name>classification_officer</name>
   </entity>
- <entity id="e310">
  <entity_name>cc_design_person_and_organization_assignment</entity_name>
  <assigned_person_and_organization>e230</assigned_person_and_organization>
  <role>e300</role>
  <items>e110</items>
```

```
  </entity>
- <entity id="e320">
  <entity_name>approval_status</entity_name>
  <name>not_yet_approved</name>
  </entity>
- <entity id="e330">
  <entity_name>approval</entity_name>
  <status>e320</status>
  <level />
  </entity>
- <entity id="e340">
  <entity_name>approval_date_time</entity_name>
  <date_time>e160</date_time>
  <dated_approval>e330</dated_approval>
  </entity>
- <entity id="e350">
  <entity_name>approval_role</entity_name>
  <role>approver</role>
  </entity>
- <entity id="e360">
  <entity_name>approval_person_organization</entity_name>
  <person_organization>e230</person_organization>
  <authorized_approval>e330</authorized_approval>
  <role>e350</role>
  </entity>
- <entity id="e370">
  <entity_name>cc_design_approval</entity_name>
  <assigned_approval>e330</assigned_approval>
  <items>e110,e60,e70</items>
  </entity>
```

```
- <entity id="e390">
  <entity_name>named_unit1</entity_name>
  <dimensions>null</dimensions>
  <prefix>.milli.</prefix>
  <name>.metre.</name>
  </entity>
- <entity id="e400">
  <entity_name>named_unit2</entity_name>
  <dimensions>null</dimensions>
  <prefix>..</prefix>
  <name>.radian.</name>
  </entity>
- <entity id="e440">
  <entity_name>named_unit3</entity_name>
  <dimensions>null</dimensions>
  <prefix>..</prefix>
  <name>.steradian.</name>
  </entity>
- <entity id="e450">
  <entity_name>uncertainty_measure_with_unit</entity_name>
  <value_component>1.0E-4</value_component>
  <unit_component>e390</unit_component>
  <name>model_accuracy</name>
  <description>maximum tolerance applied to model</description>
  </entity>
- <entity id="e380">
  <entity_name>representation_context1</entity_name>
  <context_identifier />
  <context_type />
  <coordinate_space_dimension>3</coordinate_space_dimension>
```

```
<uncertainty>e450</uncertainty>
<units>e390,e400,e440</units>
</entity>
- <entity id="e460">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>0.0,0.0,0.0</coordinates>
</entity>
- <entity id="e470">
<entity_name>vertex_point</entity_name>
<name>v1</name>
<vertex_geometry>e460</vertex_geometry>
</entity>
- <entity id="e480">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>10.0,0.0,0.0</coordinates>
</entity>
- <entity id="e490">
<entity_name>vertex_point</entity_name>
<name>v2</name>
<vertex_geometry>e480</vertex_geometry>
</entity>
- <entity id="e500">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>10.0,10.0,0.0</coordinates>
</entity>
- <entity id="e510">
<entity_name>vertex_point</entity_name>
```

<name>v3</name>
<vertex_geometry>e500</vertex_geometry>
</entity>
- <entity id="e520">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>0.0,10.0,0.0</coordinates>
</entity>
- <entity id="e530">
<entity_name>vertex_point</entity_name>
<name>v4</name>
<vertex_geometry>e520</vertex_geometry>
</entity>
- <entity id="e540">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>0.0,0.0,10.0</coordinates>
</entity>
- <entity id="e550">
<entity_name>vertex_point</entity_name>
<name>v5</name>
<vertex_geometry>e540</vertex_geometry>
</entity>
- <entity id="e560">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>10.0,0.0,10.0</coordinates>
</entity>
- <entity id="e570">
<entity_name>vertex_point</entity_name>

<name>v6</name>
<vertex_geometry>e560</vertex_geometry>
</entity>
- <entity id="e580">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>10.0,10.0,10.0</coordinates>
</entity>
- <entity id="e590">
<entity_name>vertex_point</entity_name>
<name>v7</name>
<vertex_geometry>e580</vertex_geometry>
</entity>
- <entity id="e600">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>0.0,10.0,10.0</coordinates>
</entity>
- <entity id="e610">
<entity_name>vertex_point</entity_name>
<name>v8</name>
<vertex_geometry>e600</vertex_geometry>
</entity>
- <entity id="e620">
<entity_name>direction</entity_name>
<name />
<direction_ratios>10.0,0.0,0.0</direction_ratios>
</entity>
- <entity id="e630">
<entity_name>cartesian_point</entity_name>

<name />
<coordinates>0.0,0.0,0.0</coordinates>
</entity>
- <entity id="e640">
<entity_name>vector</entity_name>
<name />
<orientation>e620</orientation>
<magnitude>10.0</magnitude>
</entity>
- <entity id="e650">
<entity_name>line</entity_name>
<name>l1</name>
<pnt>e630</pnt>
<dir>e640</dir>
</entity>
- <entity id="e660">
<entity_name>edge_curve</entity_name>
<name>e1</name>
<edge_start>e470</edge_start>
<edge_end>e490</edge_end>
<edge_geometry>e650</edge_geometry>
<same_sense>true</same_sense>
</entity>
- <entity id="e670">
<entity_name>direction</entity_name>
<name />
<direction_ratios>0.0,10.0,0.0</direction_ratios>
</entity>
- <entity id="e680">
<entity_name>cartesian_point</entity_name>

<name />
<coordinates>10.0,0.0,0.0</coordinates>
</entity>
- <entity id="e690">
<entity_name>vector</entity_name>
<name />
<orientation>e670</orientation>
<magnitude>10.0</magnitude>
</entity>
- <entity id="e700">
<entity_name>line</entity_name>
<name>l2</name>
<pnt>e680</pnt>
<dir>e690</dir>
</entity>
- <entity id="e710">
<entity_name>edge_curve</entity_name>
<name>e2</name>
<edge_start>e490</edge_start>
<edge_end>e510</edge_end>
<edge_geometry>e700</edge_geometry>
<same_sense>true</same_sense>
</entity>
- <entity id="e720">
<entity_name>direction</entity_name>
<name />
<direction_ratios>10.0,0.0,0.0</direction_ratios>
</entity>
- <entity id="e730">
<entity_name>cartesian_point</entity_name>

<name />
<coordinates>0.0,10.0,0.0</coordinates>
</entity>
- <entity id="e740">
<entity_name>vector</entity_name>
<name />
<orientation>e720</orientation>
<magnitude>10.0</magnitude>
</entity>
- <entity id="e750">
<entity_name>line</entity_name>
<name>l3</name>
<pnt>e730</pnt>
<dir>e740</dir>
</entity>
- <entity id="e760">
<entity_name>edge_curve</entity_name>
<name>e3</name>
<edge_start>e530</edge_start>
<edge_end>e510</edge_end>
<edge_geometry>e750</edge_geometry>
<same_sense>true</same_sense>
</entity>
- <entity id="e770">
<entity_name>direction</entity_name>
<name />
<direction_ratios>0.0,10.0,0.0</direction_ratios>
</entity>
- <entity id="e780">
<entity_name>cartesian_point</entity_name>

<name />
<coordinates>0.0,0.0,0.0</coordinates>
</entity>
- <entity id="e790">
<entity_name>vector</entity_name>
<name />
<orientation>e770</orientation>
<magnitude>10.0</magnitude>
</entity>
- <entity id="e800">
<entity_name>line</entity_name>
<name>l4</name>
<pnt>e780</pnt>
<dir>e790</dir>
</entity>
- <entity id="e810">
<entity_name>edge_curve</entity_name>
<name>e4</name>
<edge_start>e470</edge_start>
<edge_end>e530</edge_end>
<edge_geometry>e800</edge_geometry>
<same_sense>true</same_sense>
</entity>

★この間 <entity id="e820">**から**<entity id="e1010">**まで省略**。

- <entity id="e1020">
<entity_name>direction</entity_name>
<name />
<direction_ratios>10.0,0.0,0.0</direction_ratios>

```
  </entity>
- <entity id="e1030">
  <entity_name>cartesian_point</entity_name>
  <name />
  <coordinates>0.0,0.0,10.0</coordinates>
  </entity>
- <entity id="e1040">
  <entity_name>vector</entity_name>
  <name />
  <orientation>e1020</orientation>
  <magnitude>10.0</magnitude>
  </entity>
- <entity id="e1050">
  <entity_name>line</entity_name>
  <name>l9</name>
  <pnt>e1030</pnt>
  <dir>e1040</dir>
  </entity>
- <entity id="e1060">
  <entity_name>edge_curve</entity_name>
  <name>e9</name>
  <edge_start>e550</edge_start>
  <edge_end>e570</edge_end>
  <edge_geometry>e1050</edge_geometry>
  <same_sense>true</same_sense>
  </entity>
- <entity id="e1070">
  <entity_name>direction</entity_name>
  <name />
  <direction_ratios>0.0,10.0,0.0</direction_ratios>
```

```
  </entity>
- <entity id="e1080">
  <entity_name>cartesian_point</entity_name>
  <name />
  <coordinates>10.0,0.0,10.0</coordinates>
  </entity>
- <entity id="e1090">
  <entity_name>vector</entity_name>
  <name />
  <orientation>e1070</orientation>
  <magnitude>10.0</magnitude>
  </entity>
- <entity id="e1100">
  <entity_name>line</entity_name>
  <name>l10</name>
  <pnt>e1080</pnt>
  <dir>e1090</dir>
  </entity>
- <entity id="e1110">
  <entity_name>edge_curve</entity_name>
  <name>e10</name>
  <edge_start>e570</edge_start>
  <edge_end>e590</edge_end>
  <edge_geometry>e1100</edge_geometry>
  <same_sense>true</same_sense>
  </entity>
- <entity id="e1120">
  <entity_name>direction</entity_name>
  <name />
  <direction_ratios>10.0,0.0,0.0</direction_ratios>
```

```
  </entity>
- <entity id="e1130">
  <entity_name>cartesian_point</entity_name>
  <name />
  <coordinates>0.0,10.0,10.0</coordinates>
  </entity>
- <entity id="e1140">
  <entity_name>vector</entity_name>
  <name />
  <orientation>e1120</orientation>
  <magnitude>10.0</magnitude>
  </entity>
- <entity id="e1150">
  <entity_name>line</entity_name>
   <name>l11</name>
  <pnt>e1130</pnt>
  <dir>e1140</dir>
  </entity>
- <entity id="e1160">
  <entity_name>edge_curve</entity_name>
  <name>e11</name>
  <edge_start>e610</edge_start>
  <edge_end>e590</edge_end>
  <edge_geometry>e1150</edge_geometry>
  <same_sense>true</same_sense>
  </entity>
- <entity id="e1170">
  <entity_name>direction</entity_name>
  <name />
  <direction_ratios>0.0,10.0,0.0</direction_ratios>
```

```
  </entity>
- <entity id="e1180">
  <entity_name>cartesian_point</entity_name>
  <n ame />
  <coordinates>0.0,0.0,10.0</coordinates>
  </entity>
- <entity id="e1190">
  <entity_name>vector</entity_name>
  <name />
  <orientation>e1170</orientation>
  <magnitude>10.0</magnitude>
  </entity>
- <entity id="e1200">
  <entity_name>line</entity_name>
  <name>l12</name>
  <pnt>e1180</pnt>
  <dir>e1190</dir>
  </entity>
- <entity id="e1210">
  <entity_name>edge_curve</entity_name>
  <name>e12</name>
  <edge_start>e550</edge_start>
  <edge_end>e610</edge_end>
  <edge_geometry>e1200</edge_geometry>
  <same_sense>true</same_sense>
  </entity>
- <entity id="e1240">
  <entity_name>oriented_edge</entity_name>
  <name>oe11</name>
  <edge_start>e490</edge_start>
```

<edge_end>e470</edge_end>
<edge_element>e660</edge_element>
<orientation>false</orientation>
</entity>
- <entity id="e1250">
<entity_name>oriented_edge</entity_name>
<name>oe12</name>
<edge_start>e470</edge_start>
<edge_end>e530</edge_end>
<edge_element>e810</edge_element>
<orientation>true</orientation>
</entity>
- <entity id="e1260">
<entity_name>oriented_edge</entity_name>
<name>oe13</name>
<edge_start>e530</edge_start>
<edge_end>e510</edge_end>
<edge_element>e760</edge_element>
<orientation>true</orientation>
</entity>
- <entity id="e1270">
<entity_name>oriented_edge</entity_name>
<name>oe14</name>
<edge_start>e510</edge_start>
<edge_end>e490</edge_end>
<edge_element>e710</edge_element>
<orientation>false</orientation>
</entity>
- <entity id="e1230">
<entity_name>edge_loop</entity_name>

<name>el1</name>
<edge_list>e1240,e1250,e1260,e1270</edge_list>
</entity>
- <entity id="e1280">
<entity_name>face_outer_bound</entity_name>
<name>fob1</name>
<bound>e1230</bound>
<orientation>true</orientation>
</entity>
- <entity id="e1290">
<entity_name>cartesian_point</entity_name>
<name />
<coordinates>10.0,0.0,0.0</coordinates>
</entity>
- <entity id="e1300">
<entity_name>direction</entity_name>
<name />
<direction_ratios>0.0,0.0,-100.0</direction_ratios>
</entity>
- <entity id="e1310">
<entity_name>direction</entity_name>
<name />
<direction_ratios>-10.0,0.0,0.0</direction_ratios>
</entity>
- <entity id="e1320">
<entity_name>axis2_placement_3d</entity_name>
<name />
<location>e1290</location>
<axis>e1300</axis>
<ref_direction>e1310</ref_direction>

</entity>
- <entity id="e1330">
<entity_name>plane</entity_name>
<name>p1</name>
<position>e1320</position>
</entity>
- <entity id="e1340">
<entity_name>advanced_face</entity_name>
<name>af1</name>
<bounds>e1280</bounds>
<face_geometry>e1330</face_geometry>
<same_sense>true</same_sense>
</entity>

★この間 <entity id="e1350">**から**<entity id="e1820">**まで省略**。

- <entity id="e1840">
<entity_name>oriented_edge</entity_name>
<name>oe61</name>
<edge_start>e550</edge_start>
<edge_end>e570</edge_end>
<edge_element>e1060</edge_element>
<orientation>true</orientation>
</entity>
- <entity id="e1850">
<entity_name>oriented_edge</entity_name>
<name>oe62</name>
<edge_start>e570</edge_start>
<edge_end>e590</edge_end>
<edge_element>e1110</edge_element>

<orientation>true</orientation>
</entity>
- <entity id="e1860">
<entity_name>oriented_edge</entity_name>
<name>oe63</name>
<edge_start>e590</edge_start>
<edge_end>e610</edge_end>
<edge_element>e1160</edge_element>
<orientation>false</orientation>
</entity>
- <entity id="e1870">
<entity_name>oriented_edge</entity_name>
<name>oe64</name>
<edge_start>e610</edge_start>
<edge_end>e550</edge_end>
<edge_element>e1210</edge_element>
<orientation>false</orientation>
</entity>
- <entity id="e1830">
<entity_name>edge_loop</entity_name>
<name>el6</name>
<edge_list>e1840,e1850,e1860,e1870</edge_list>
</entity>
- <entity id="e1880">
<entity_name>face_outer_bound</entity_name>
<name>fob6</name>
<bound>e1830</bound>
<orientation>true</orientation>
</entity>
- <entity id="e1890">

<entity_name>cartesian_point</entity_name>
<name />
<coordinates>0.0,0.0,10.0</coordinates>
</entity>
- <entity id="e1900">
<entity_name>direction</entity_name>
<name />
<direction_ratios>0.0,0.0,100.0</direction_ratios>
</entity>
- <entity id="e1910">
<entity_name>direction</entity_name>
<name />
<direction_ratios>10.0,0.0,0.0</direction_ratios>
</entity>
- <entity id="e1920">
<entity_name>axis2_placement_3d</entity_name>
<name />
<location>e1890</location>
<axis>e1900</axis>
<ref_direction>e1910</ref_direction>
</entity>
- <entity id="e1930">
<entity_name>plane</entity_name>
<name>p6</name>
<position>e1920</position>
</entity>
- <entity id="e1940">
<entity_name>advanced_face</entity_name>
<name>af6</name>
<bounds>e1880</bounds>

<face_geometry>e1930</face_geometry>
<same_sense>true</same_sense>
</entity>
- <entity id="e1220">
<entity_name>closed_shell</entity_name>
<name>cs</name>
<cfs_faces>e1340,e1460,e1580,e1700,e1820,e1940</cfs_faces>
</entity>
- <entity id="e1950">
<entity_name>manifold_solid_brep</entity_name>
<name>msb</name>
<outer>e1220</outer>
</entity>
- <entity id="e1960">
<entity_name>advanced_brep_shape_representation</entity_name>
<name>absr</name>
<items>e1950</items>
<context_of_items>e380</context_of_items>
</entity>
- <entity id="e1970">
<entity_name>shape_definition_representation</entity_name>
<definition>e90</definition>
<used_representation>e1960</used_representation>
</entity>
</Model>

3.8　　参考文献

（1）ＳＴＥＰ実装ガイド、日本情報処理開発協会、1998

（2）ＳＴＥＰ実用化に向けての調査研究報告書、日本情報処理開発協会、1998

（3）製品モデル表現とその利用技術、日本規格協会、1995

（4）最新ＸＭＬがわかる、技術評論社、2000

（5）ＳＴＥＰ適用推進に向けた検討結果の報告、日本情報処理開発協会、2000

参加メンバー

＜委員＞

河井　弘之	カルソニックカンセイ株式会社(部品工業会)
山本　一人	株式会社ケイ・ジー・ティー
小林　公博	財団法人建設業振興基金
坂井　佐千穂	セイコーエプソン株式会社
吉田　和三	日本アイ・ビー・エム株式会社
烏田　正志	日本アビオニクス株式会社
石見　正和	財団法人日本建設情報総合センター
野口　憲	社団法人日本航空宇宙工業会
対比地　幹雄	日本電気株式会社
廣　喜充	株式会社日立製作所
飯塚　正治	富士重工業株式会社
坂田　豊	株式会社富士通九州ｼｽﾃﾑｴﾝｼﾞﾆｱﾘﾝｸﾞ
佐々木　安夫	富士電機株式会社

＜事務局＞

千田　雅彦	電子商取引推進協議会
上田　高廣	電子商取引推進協議会
竹内　斎之郎	電子商取引推進協議会
調　敏行	電子商取引推進協議会

平成13年3月発行
発行：電子商取引推進協議会
東京都江東区青海2－45
タイム24ビル10階
Tel　03-5500-3600
E-mail　info@ecom.or.jp