C．2．3 中核要素

中核要素は(C.2.2.2)に挙げた情報素 infoNode, 関係素 infoArc, 関係視座素 arcScope で，その具体の内容は後定義される（C.2.2.3）.

C.2.3.1 infoNode

情報素は情報塊を現す．その範囲は atomic な情報塊から大域粒度の情報塊までとする.

@uid は，当該システム内または想定される対象 instance 内における infoNode に一意に与えられる同定子であり，型は xs:ID とする.

@oid は，infoNode 型（W3C XML complexType）を規定する ISO object identifier であり型は xs:NMTOKEN とする.

@rid は，当該 infoNode が外部参照を指し示す際の URI であり，型は xs:anyURI とする.

@bearing は，当該 infoNode が出現する文脈，もしくは当該 infoNode の出現や在り方を制約する arcScope@uid を格納する．実装上の参照性を向上させる目的で導入した.

@category は，当該 infoNode が（当該 arcScope にて規定される文脈において）存在するべき domain や subdomain を表現する．値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする.

@kind は，@category と基本的に同様なものの @category の値に応じた詳細を示すものとする.

@existence は，当該 infoNode が物理的（＝非抽象）な存在である場合に，その実在状況を表現する.

なお初年度の試作実装にて活用されなかった @occurrence は削除した.

要素 nodeCode

情報素 infoNode が表現すべき具体の事物内容は要素 nodeCode にて規定される.

@CSname, @CScode, @CSver は，それぞれコード体系の名称，コード体系コード，コード体系の版を指定する.

@codename, @code は，それぞれコードに与えられた名称，コードを指定する.

@priority は，情報素 infoNode が複数の nodeCode を持つ場合に，処理優先に関する示唆を与え，型は xs:int とする.

要素 construe

要素 construe は，情報素 infoNode の内容を，特定の名前空間に存在する用語にて説明する際に用いる.

@NSname, @NScode, @NSver は，それぞれ関心領域における名前空間の名称，名前空間のコード，名前空間の版を指定する.

要素 construe は値を持つことができる.

他の要素

情報素はその配下に要素 description と要素 comment を持ちうる.

C.2.3.2 infoArc

関係素は二つの情報素 infoNode の間に関係を結ぶ．その範囲と意義は関係視座素 arcScope が規定する範囲を超えることができないものとする.

@ref は，ただ一つの infoNode, facet あるいはコード体系を指し示す．このような事情から，型は xs:string とする（C.2.6.4）.

@category は，当該 infoArc が（当該 arcScope にて規定される文脈において）指し示す infoNode と指し示される infoNode との関係の意義を表現する．値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする.

@kind は，@category と基本的に同様なものの @category の値に応じた詳細を示すものとする.

@multiplicity は，いわゆる Class（model）を規定する際に多重度を表現し，型は xs:NMTOKEN とする（C.2.7.4）.

関係素は配下に位相素 topology を持ちうる.

C.2.3.3 arcScope

関係視座素は，情報素 infoNode が関係素 infoArc によって関連を結ぶ際の，その視座意義や視野範囲を限定する.

@uid は，当該システム内または想定される対象 instance 内における arcScope に一意に与えられる同定子であり，型は xs:ID とする.

@category は，当該 arcScope が規定する文脈や制約を表現する．値は domain や subdomain に

応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする．

@kind は，@category と基本的に同様なものの @category の値に応じた詳細を示すものとする．

@choice は，いわゆる Class (model) を規定する際に，配下の infoArc が指し示す infoNode に対する選択性を表現する (C.2.7.5)．型は xs:NMTOKEN とする．

関係視座素は配下に要素 comment を持ちうる．

C．2．4 修飾要素

全ての要素は (C.2.3) に示した中核要素のみにて表現することも可能ではある．とはいえ，理解のしやすさや使用頻度にも配慮して修飾要素を用意した (C.2.2.2)：位相素 topology，計量素 dimension．なお，その具体の内容は後定義される (C.2.2.3)．

C.2.4.1 topology

位相素は，関係する情報素の間の物理的または抽象的な“位置関係”を表現する．と同時に，必要に応じて“位置関係”に関する制約を表現する．

抽象的な“位置関係”制約とは，結局のところ概念間の関係の構成に条件や規則を与える，ということである．

位相素は配下に方向素 orientation を持ち，ここに位相内容を示す．

@path は，複数の位相素が存在する場合，その適用順序を規定し，型は xs:int とする．

@option, @request, @negate, @hop は他の情報素 infoNode の存否に関わる制約を表現するが，これらについては後述する (C.2.7)．

要素 orientation

方向素 orientation は，物理的または抽象的な位相内容を示す．

@direction は“位置関係”に係る方向を表現し，値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする．

@coordinate は“位置関係”に係る方向を表現する際に必要に応じて軸面や軸線を表現する．値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする．

C.2.4.2 dimension

計量素は，物理的または抽象的な“計量”内容を表現する．

その様な“計量”内容は通常の Class (model) では attribute として扱われうる内容である．

@tude は“計量”内容を表現し，値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする．

@unit は，@measure に格納される値の単位を必要に応じて規定し，値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする．

@equivalent は，@measure に格納された値自体が直接的に計量素の示すべき値を示しているか否か，その等価性を表現する．値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする．

@measure は“計量”内容の値を格納し，型は xs:normalizedString である．

@dataType は，@measure に格納された値のデータ型を必要に応じて明示する．値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする．ただし，これに含まれる値は W3C XML で規定されるデータ型の範囲内であることを原則とする．

C．2．5 周辺要素

C.2.5.1 description

要素 description は複数の役割を担う，いわば書誌事項の塊である．

@uid は，当該システム内または想定される対象 instance 内における description に一意に与えられる同定子であり，型は xs:ID とする．

よってルート要素 facet 直下の要素は全て@uid を持つことになるので直列化は W3C XML Schema, RELAX NG, object-oriented Class instance あるいは RDBMS でも実施可能である．

なお，情報素 infoNode，関係視座素 arcScope そして要素 description の子要素は各々の親要素と緊く結ばれているので，@uid が必須となる要素は，ルート要素 facet 直下，これらの三種のみである (C.3.5)．

要素 scope

要素 scope は大域粒度の内容を表現するルート要素 facet または情報素 infoNode において，以下の要素属性を与える（C.2.6）.

@domain は，当該 facet または infoNode の関心領域を表現し，値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする.

@category は，当該 facet または infoNode の話題や場面を表現し，値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする.

@kind は，@category と基本的に同様なものの @category の値に応じた詳細を示すものとする. なお scope[@kind] の値については本研究では暫定的である.

要素 scope/concern

要素 scope は大域粒度の内容を表現するルート要素 facet または情報素 infoNode において，その facet または infoNode の形成に関わった（広義の）関与者を指し示す.

@concern.refID は関与者の ID を格納する.

なお，その他の関与者に関する詳細な記述は，本研究では割愛しているが，その片鱗は（C.6.12～13）に垣間見ることができよう.

要素 validity

要素 validity は，記述枠組が交換に供される場合に必要に応じて用いることを想定して設けた. ただし試作参照実装では使用していない.

@conformance は，記述枠組の一致性のレベルを格納する.

@wholeness は，交換等に供されたデータは蓄積記録の全部か部分抽出かを表現する.

他の要素

要素 description も配下に要素 comment を持ちうる.

C.2.5.2 comment

情報素 infoNode，関係視座素 arcScope，要素 description は，必要に応じて要素 comment を持つことができ，instance の構造や処理や判読などを注釈する際に，これを用いる.

@category は，注釈の種別を表現し，値は domain や subdomain に応じて W3C XML dataType facet で値制限することを前提し，型は xs:NMTOKEN とする.

要素 comment は値を持つことができる.

C．2．6 再帰性と粒度依存性

C.2.6.1 粒度依存性

CSX model は，CSX model を用いて表現・記述する情報は全て，再帰的に構成することを前提している．

ただ細粒度の情報塊から大域粒度の情報塊を再帰的に構築していく際に，情報塊が求める「属性の値の枠組」や「情報塊への参照枠組」が変容していくことに気づかされる：

・大域 粒度における範疇属性の 値の枠組
・大粒度 の情報塊への 参照枠組

C.2.6.2 大域粒度における範疇

情報素 infoNode が属する（または焦点される）domain や subdomain は@category と@kind にて示されるが，具体の事物内容は nodeCode にて規定される．

このような枠組は，粒度が比較的小さい場合には何の障碍もなく機能する．しかし infoNode が包含する内容が大域となった場合には，category や kind という範疇属性に格納される値の枠組自体が変容してくる．

たとえば親 infoNode を紹介状，その子または孫 infoNode を病名や検査項目名としてみよう．子や孫の infoNode[@category @kind]を埋めることは容易だが，親 infoNode は文書であり，様々な domain や subdomain の infoNode を集積している．

そのような場合，@category や@kind は，ある事物の階層的範疇を示しているのではなくて，むしろ集積された情報塊の，その話題や場面を示すように変容していくからである．

このような異質な視座を持つ体系の値を同一の属性に格納することは，妥当とは思われない．よって description/scope を用意した．

C.2.6.3 同型対応順応性

一方，facet に包絡される infoNode と arcScope は，その全体を infoNode としても構築できるので，infoNode の配下にも description を配することとした．

・facet/description/scope
・facet/infoNode/description/scope

C.2.6.4 大粒度情報塊の参照

様々な情報塊を CSX model で構築していくと，種々の infoNode や facet が蓄積していくことになる．

このとき facet を infoNode に構成しなおして参照することも可能ではあるが，その様な構成コストを節減すべき場合もある．したがって，次の仕様を持つ“container” infoNode を用意することとした：

・arcScope による視座意義や視野範囲による限定の 枠（箍） を出ない
・属性に uid，category，kind を持つ
・infoNode を参照できる
・facet を参照できる
・container 自身を参照できる
・参照実体が複数存在する場合には，その発生発現の順や時刻を記述しうる

これらの表現と構成の要件は全て，現状の infoNode，arcScope，infoArc にて実現可能である．唯一必要なことは，各属性に格納すべき適切な値を決定することのみである．

このような“container” infoNode を導入した利点は小さくない．というのも情報塊の粒度や大域性に関わらず，一様にして，情報塊を効率的に構成するための情報塊参照枠組を入手でできたからである．具体的には，

・種々の情報塊の集積が必要な診療文書でも，それらを容易に構成でき，加えて（C.2.6.2）や（C.2.6.3）の支援によって，その視野や場面や話題を明示できる．

・関与者のリストを保持する directory などを参照することもできる．

・アプリケーションの個々の構成部品に視野を与えることができる．

・閲覧した診療情報の“スナップショット”も保管できる．

ようになった．これらは全て(C.3.4)の末尾に示した diagram と同等の構成となる．

また実装システムでは，システムが統括責任を負うべき情報塊の発生順序と時刻について，“container”を利用して管理することも可能となった．

C．2．7 制約表現

本研究を開始した時点のCSX model (B.8) とは，本質的には事実記述であって，付随的にパタン表現も為しうるのみであった．

事実やパタンの記述と制約の記述とを分離し，かつ明示的に扱えるようにし，しかも他の記述枠組ではなく自己完結的に表現できたなら，CSX model の表現力と明晰性は向上して，知識表現能力を獲得することになる．そのうえ (B.6.6) からの要請もある．

初年度は前哨としてCSX model による制約表現可能性に関して詳細に考察したところ，以下のうちいずれかの手法によって，CSX model での制約表現が可能なことを明らかにした：

- XML simpleType facet による値の限定
- 定義体 infoNode と制約体 infoNode を準備し，制約 arcScope と infoArc で両者を連結
- 制約 arcScope，その配下の infoArc，さらにその配下の topology の，属性とその値によるグラフ構造内での関係の規定

第二年度は，その後の考察ならびに考案を踏まえつつ[医療情報学 23S：800-801,2003]，実際に情報モデルにそれらを反映した．ただし制約表現に関する試作実装はもともと対象範囲外である．

C.2.7.1 対象事項

制約表現の対象となりうる事項は次のように分類できよう：

情報塊の構造や値

- データ型
- 初期値と値範囲
- 関係の多重度（基数）
- 結合性（親和性）と選択性

比較や条件など

- 比較
- 条件（含意と連言）

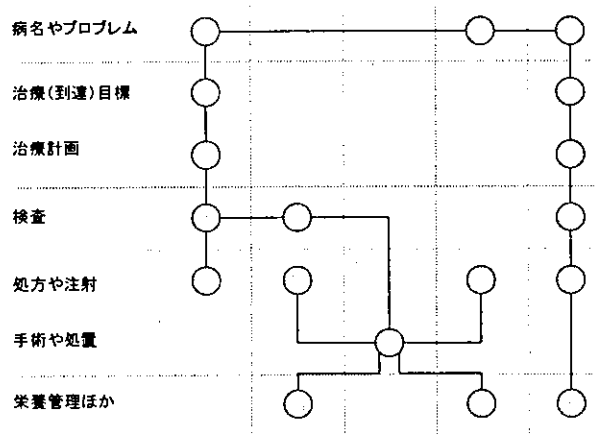
多重グラフにおけるノード間の制約

- 径の形成
  - 存否の要求
  - 選択と取捨自由度
  - 隣接性と(区間)距離
  - 順序性と共起性

さて業務事象の扱いには時制軸が必須であることから，扱うべきは，時制に関する単方向性有向グラフである．ただ制約表現については，前方参照も後方参照も許容する．また時制は，間隔尺度として扱いにも留意すべきであろう．

次に，グラフ内の node が存在している domain や subdomain（例：病名プロブレム subdomain や機器機材 subdomain など）を弁別することとした．

これら二点を要約すると，多領域かつ離散時刻区間に存在する多重グラフにおける制約，ということになる．



なお含意や連言を含む条件や状態の扱いについては，第二年度は，上図の如き多重グラフにて扱いうる範囲を超えないこととした．また処理規則の記述も，対象範囲外とする．

この範囲において，まず制約類型を洗い出し，記述形式を定式化していくこととした．

C.2.7.2 制約類型

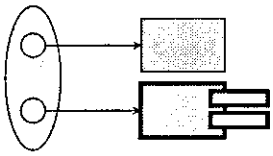
対象範囲（C.2.7.1）は以下のように要約することができよう：

- データ型
- 値範囲限定（初期値を含む）
- 多重度（基数）
- 選択
- 比較
- 順序と共起
- 存在要求，存在否定，存在許容
  - 取捨自由度
  - 依存
  - 排他
  - 隣接性と時間距離
- ドメイン（サブドメイン）親和性

次に個々の記述形式について順次記していく．

C.2.7.3 データ型と値範囲

データ型，ならびに初期値あるいは極値や正常範囲などの値範囲を表現する．



まず定義体 infoNode を用意し，加えて，制約内容を記述した制約体 infoNode を用意する．そして定義体と制約体とを制約 arcScope にて連結する．

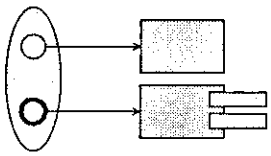
なお dimension[@equivalent]には，たとえば“SmallerThan”などの比較叙述詞なども既に用意されている．

C.2.7.4 多重度

基数を規定するために属性 multiplicity を，infoArc に加える．

infoArc[@multiplicity="min..max"]

この値範囲は 0 と正整数であり min <= max であることを要する．

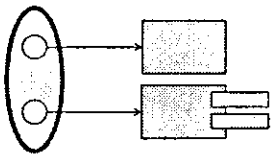


これは定義体 infoNode の context を形成する arcScope において機能することを想定する．この infoArc が指し示す子 infoNode の基数が定義される，と解釈する．

C.2.7.5 選択

選択は，与えられた infoNode 集合から，特定の infoNode を採用しなければならないことを要求する．新たな属性 choice を，arcScope に加える．

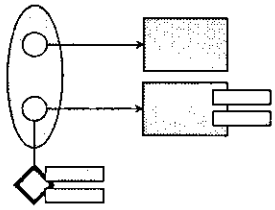
arcScope[@choice="TRUE | 1"]



選択は通常，可選択個数が一であることが要求される．しかし実世界では可選択数が一以上の事象も多いので，上記のような値範囲とした．なお 1 以上の正整数 n を指定し，かつ選択肢の要素を n 個以上存在させているなら，記法自体としては整合破綻することはない．

C.2.7.6 比較

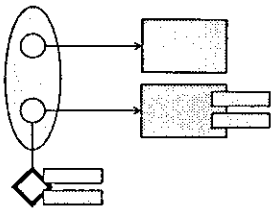
比較または制約 arcScope において，相異なる二つの infoNode 間の物理量または概念量を，infoArc/topology を活用して比較表現する．



なお，何について，どのように，どれ程，に関する記述は，topology/orientation ならびに topology/dimension において表現する．また orientation[@direction]には“SmallerThan”などの比較叙述詞なども既に用意されている．

C.2.7.7 順序と共起

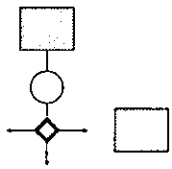
これらは infoArc/topology[@path]にて容易に表現できる．この値範囲は整数である．格納されている値が同値であれば共起と解釈する．



これは制約 arcScope において機能することを想定する．なお取捨自由度や存否要求等も infoArc/topology に記述され，これらは連携して機能するよう解釈される必要がある．

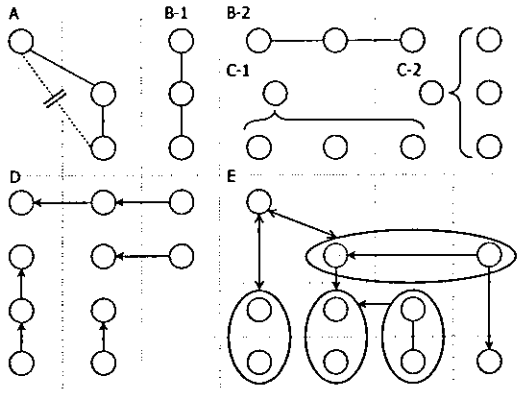
C.2.7.8 存否

存否要求は制約 arcScope において機能し，順序や共起とともに infoArc/topology に記述され，これらは連携して機能するよう解釈される必要がある．



なお依存と排他における多重性や多段性は，infoNode の連（run）や組（combination）にて

表現することを前提とする．



これらの表現は，暗示的には「限定された場」を現す infoNode を要求していることに御留意願いたい．

取捨自由度：存否許容

取捨自由度とは，存否許容を表現する．これのみが記されるだけでは重要な意義を為さない．しかし順序や共起と関わる場合は条件効果を発揮することがある．新たな属性 option を，topology に加える．

topology[@option="self | [*infoNode@uid*]"]

依存：存在要求

依存とは，他の infoNode の存在要求である．新たな属性 request を，topology に加える．

topology[@request="self | [*infoNode@uid*]"]

依存（存在要求）には以下の種別がありうる：

・多段依存
・共起元依存
・複数元依存
・多重元依存

排他：存在否定

排他とは，他の infoNode の存在否定である．新たな属性 negate を，topology に加える．

topology[@negate="self | [*infoNode@uid*]"]

排他（存在否定）には以下の種別がありうる：

・多段排他
・共起元排他
・複数元排他
・多重元排他

隣接性

隣接性とは，他の infoNode との直接結合の拒否の主張である．新たな属性 hop を，topology に加える．

topology[@hop="[*infoNode@uid*]"]

距離（時間）

時間的な距離は，infoArc/topology 配下の orientation/dimension で表現することができる．計量素 dimension は，相対時刻も時間間隔も表現可能である．

ただ領域（domain や subdomain）は名目尺度であるため距離概念の表現は困難である．よって次に述べるドメイン親和性にて，結合の可否を表現することとする．

C.2.7.9 ドメイン親和性

個々の infoNode が属する domain や subdomain に応じながら，infoNode 間での連結親和性または拒否性を規定する．

個々の infoNode が属する domain や subdomain は infoNode[@category and @kind]で規定されている．したがってドメイン親和性は「そのような値を持つ」つまりドメイン自体を表すスーパークラス infoNode を準備し，それらの間で結合制約を記述すればよいことになる．

よってスーパークラスの概念と記述可能性が必要となるが，これは容易に可能である：CSX model は (a) infoNode 間の vertical relation を表現でき，また (b) 範疇子 @category や @kind の値自体が code schema つまり taxonomy 下に表現されうる，からである．

C．3　直列化について

CSX model は，関係（infoArc）や関係の視座意義や視野範囲（arcScope）を独立要素として明示的に扱うとしたうえで，arcScope/infoArc によって構成された一つの context に定置された情報素 infoNode は他の context でも再利用される（参照される）ことを許容し，この繰り返しによって細粒度から大粒度までの情報塊を再帰的に構成することを許容している．

これらのことから XML Schema による直列化に際しては特段の留意が必要となった．すなわち，root element には何を置くべきか，という設問の解決である．

以下，検討の過程と結論に至った事由を記す．

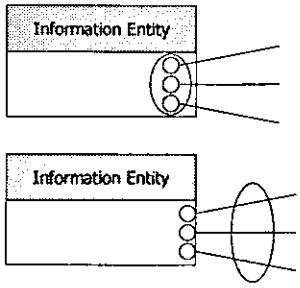
C．3．1　Root element の候補＃1

infoNode を root element に

記述形式としては機能することもあるが，ただ root element を特別扱いする schema においては，再帰参照時に障碍が発生しうる．

ほか，形式的または扱いの面で厄介と思われる事項は，node tree が極めて深くなりうること，しかもその深い木構造において，実体 infoNode と参照 infoNode とがモザイク状に分散しうること，である．

意味論と形式論の双方に関わる事項としては，infoNode つまり情報塊は，関係視座（視野）や関係を包含することは妥当や否や，という問に直面することになる．この包含関係は node tree という根源的な表現構造から生じている．



もし此れを是とするならば，情報塊は基本的に，自らが包含する関係視座（視野）や関係を認知している（しうる）ことを主張することとなるが，これは不自然もしくは不可能であろう．

相反する複数の context 内に同一 infoNode が存在する場合，相反する arcScope を同一の infoNode が同時に保有することになるからである．そして，このような形式的な表現状況は矛盾，と呼ばれることが通常である．

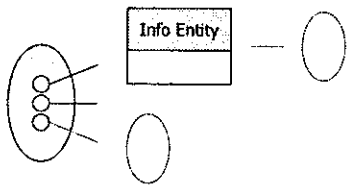
よってこの直列化方式は妥当とは思われない．

C．3．2　Root element の候補＃2

arcScope を root element に

意味論的には，関係視座や関係視野が存在しなければ如何なる情報塊も存在しえない（意味を持たない）という主張は，特定の哲学的な立場によっては，妥当と云えるだろう．

しかし現状の CSX model を直列化する場合は，すぐに破綻してしまうことになる．というのも (a) infoNode は arcScope を参照する属性を持っておらず，かつ，(b) arcScope は arcScope を参照する属性を持っていないからである．



もちろん御都合主義的に，それらの属性を追加することも容易ではある．

ただ前者については，(C.3.1) にて記した問と同じ問を孕むことになる．後者は，もし冒頭の主張を是とするならば，その哲学的な意義を，一般論として定義できるかもしれない．

しかしながら arcScope を root element とするのが妥当であると認めるには (a) と (b) とが同時に成立する必要があるのである．

よってこの直列化方式は妥当とは思われない．

C．3．3　Root element の適任者

Facet を root element に

上記候補のいずれも root element に採用することができないのであれば，新たな要素を用意する必要がある．よってこれを facet とした．

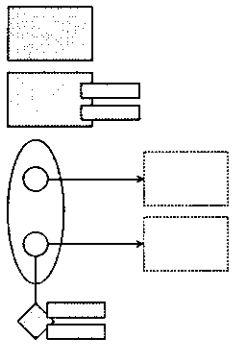
Facet を設けたならば，infoNode，arcScope，infoArc を必要に応じて包み，離合集散させるための必要条件を得た．

次に node tree を前提した schema による直列化に際して上述した破綻を回避する十分条件とは，infoNode も arcScope も，互いに互いを包まないことである．したがって infoArc は infoNode を pointer で参照するのみ，とする．

C．3．4　直列化要件の整理

以上を纏めると，ツリー構造を前提する枠組においてCSX modelを直列化する場合には，次図のように扱うべきことになる．

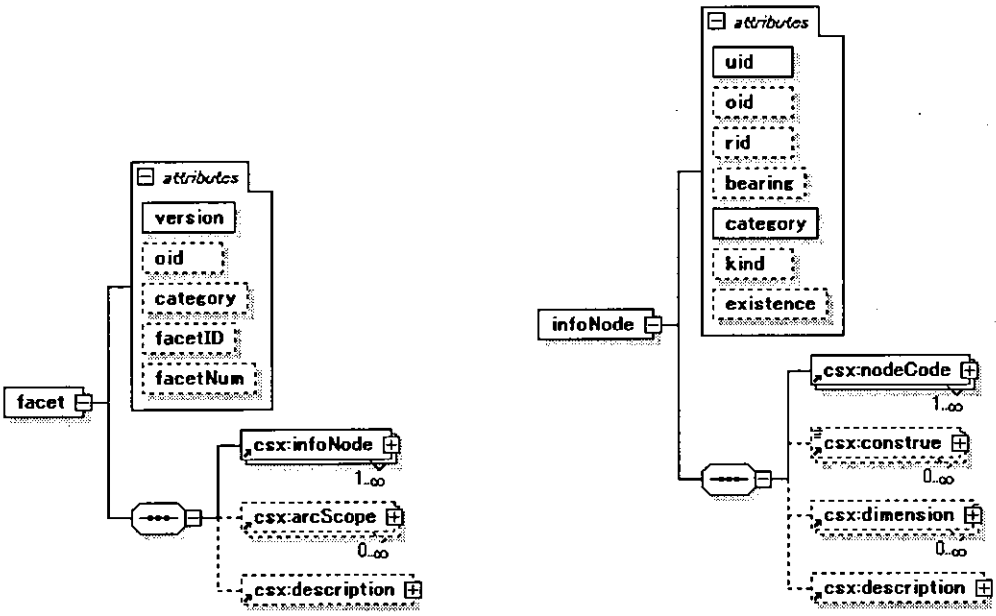
図中では，長方形はinfoNode，小円はinfoArc，楕円はarcScope，菱形はtopologyならびにorientation，小長方形はdimensionを表している．



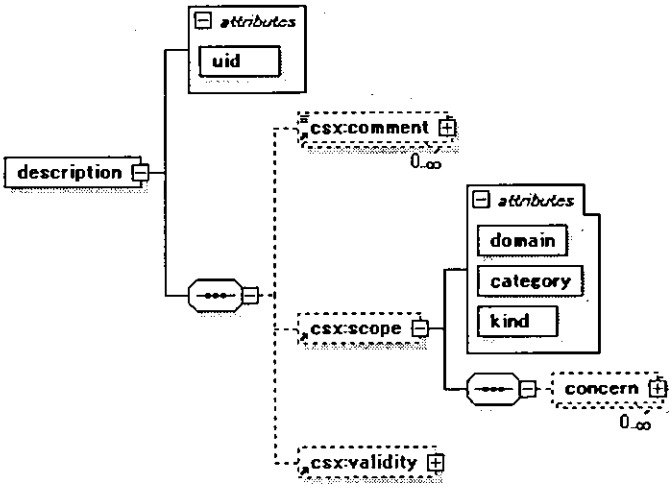
C．3．5　XMLダイアグラム

直列化はW3C XML Schemaで行うことを原則とするがRELAX NGによる直列化も許容した．ちなみに試作アプリケーションではRELAX NGに拠る要素出現パタンとなっている（資料1）．

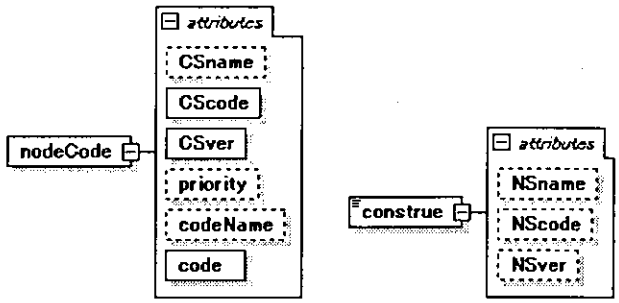
facetならびにinfoNode



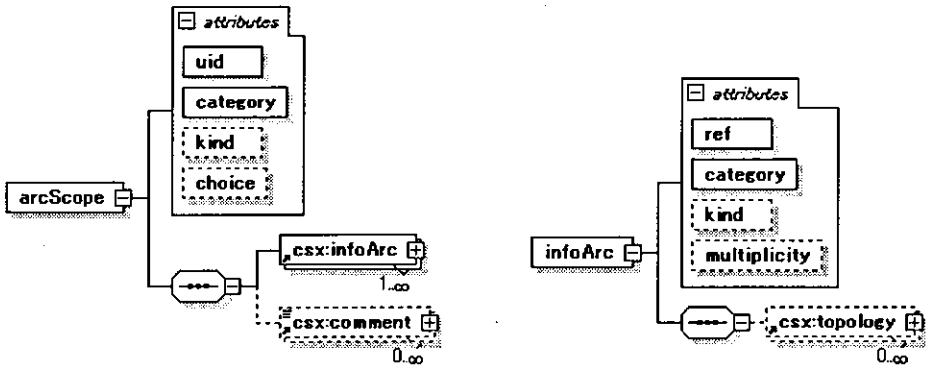
descriptionとscope



infoNode 配下の nodeCode と construe

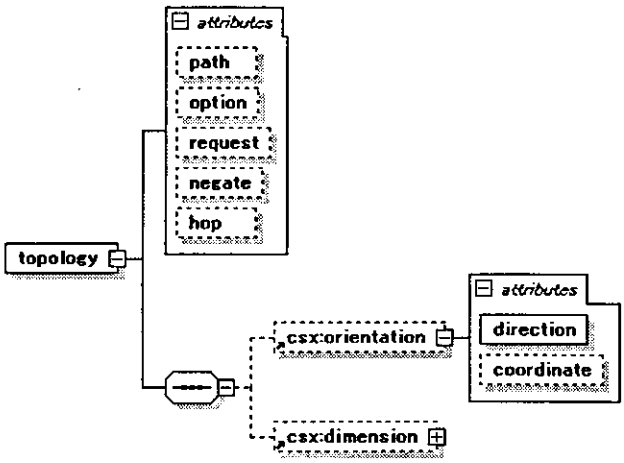


arcScope と infoArc



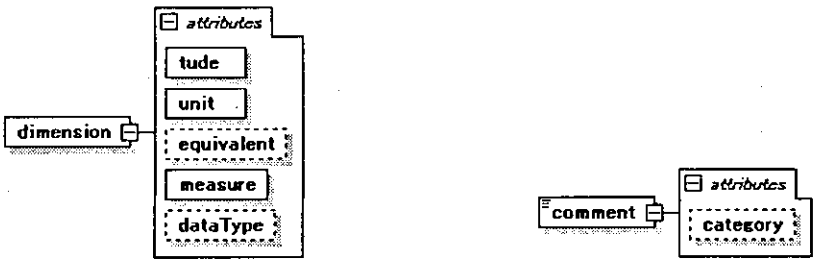
topology と orientation

orientation と dimension は一括して topology 配下に置いている.



dimension と comment

これらは必要に応じて親 element のなかで用いられる.



C．4　病名プロブレム変遷の構成

C．4．1　病名構成の原型

CSX model による病名構成は容易であり，既に初年度に完了している（ただし現段階では標榜診療科や保険情報との関連は割愛した．また，病名/プロブレムの階層構造については試作実装を躊躇している）．

まず病名/プロブレムは四要素：部位，前置修飾語，根幹病名，後置修飾語，から構成されるとした．

そして各要素を infoNode として定義しておき，次に病名全体を表す構成体 infoNode を用意し，この構成体は各要素から組立てられることを arcScope/infoArc で表現するとした．

そして病名の諸属性は，その構成体の諸属性を示す子要素 dimension にて表現される．

> *dimension も，それ自身が何を表すのか，は meta-attribute @tude の値で決定され meta model となり，@unit や @dataType が指定されて model (Attribute) となる．*
> *そして user object 段階では，@measure に具体値が挿入されるのである (C.2.4.2).*

病名を構成する要素 infoNode の並び順の表現方法には複数の選択肢がありうる：

- infoArc/topology[@path]
- infoArc/topology/orientation[@direction]
- infoArc/topology/dimension

CSX model における表現の明晰さと解析解釈の容易さとを考慮して一番目を正規採用することとした．

C．4．2　Basso Continuo

変遷モデル (B.2) では病名/プロブレムの属性として rank を設けていたが，これは Basso Continuo と呼び換えることとした．その概念は，その病態が，診療目標の設定や介入計画に対して通奏低音の如く影響する，そういう病名/プロブレムである．

蓄積された診療情報の追跡性を考慮する場合，このような病態の存在を記録する必要があると判断されたからである．

C．4．3　MEDIS-DC 病名への対応

ただ MEDIS-DC 病名集では，交換コード書式に関する指針を表明している．それに依拠すると，根幹病名と修飾語とを別々の infoNode としないほうが扱いも容易であり，妥当と思われる．よって参照実装では，そのようにした．これに伴って，修飾語の並び順も nodeCode 内に包含した．

C．4．4　Stage と Intervention

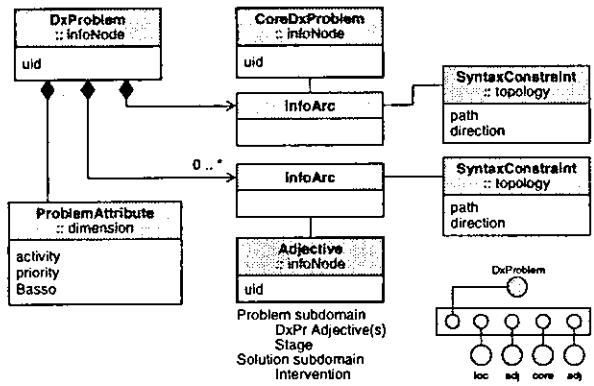
臨床医師に対して試作システムの試用を依頼するとは，実診療ではないのであるから，結局，診療経過モデルの試作を依頼することを意味している．

そのような試作システムを使う際に必要な事項は何か，について，対象予定の臨床医師に preliminary な調査を行ったところ以下のような回答を得た：

- MEDIS-DC 病名集には含まれていない専門的な病期や重症度などを記載できないと，有意義な診療経過モデルを構築できない．
- 病名または病名の構成要素ではないものの，介入（＝医療行為）の状況や介入の予定あるいは結果を記載できないと，有意義な診療経過モデルを構築できない．

臨床現場のこれらの意見を踏まえて，参照実装においては，通常の意味での病名または保険傷病名に加えて，Stage を格納する construe と，Intervention を格納する construe を用意し，各々，MEDIS-DC 病名集では表現できない病期や重症度，あるいは介入状況等を格納することとした．

ただし本来の姿は，これらは infoNode に格納して arcScope/infoArc で関連付けて構成するべきではある（下図）．



そしてまた，このように構成したほうが，構成要素が属すべき subdomain に関する混淆または汚染を，自然に回避することができる．

しかし第二年度は効率的に実装作業を遂行するよう求められていることから，初年度に構築済みの「病名 Composer」モジュールの改変を最小限とするべく，上述したように construe を用いることとした．

C．4．5　修飾語の扱い

現時点の MEDIS-DC 病名集分類では，修飾語の種類の区分分類が計画されているものの未完だが，試作アプリケーション用に仮分類した：

01. イベント
02. 介入・介入関連
05. 経過や病期や急性慢性など
06. 発現の時期や周期など
11. 性別
12. 年齢時期
15. 上下左右高低頭尾腹背
16. 形状・方向・部分・非正常物
17. 基数
18. 解剖
19. 人工物
21. 型
22. 発現状態や重畳など
31. 重症度や程度
32. 病因論的
81. プロブレム定義
82. 型や程度など
84. 経過や病期
85. 妊娠の週および月
86. 胎齢・週齢
91. 介入・介入関連
99. 削除対象

なお50以前が前置修飾語で80以降は後置修飾語である．

C．4．6　プロブレムリストの原型

CSX model によるプロブレムリストの構成は容易であった（ただし現段階では標榜診療科や保険情報との関連は割愛している）．

まずプロブレムリストを infoNode として定義する．そして当該プロブレムリストに含まれる病名 infoNode との arcScope を定義する．

病名と同様に，プロブレムリストの諸属性は，要素 infoNode の属性でなく，その子要素の dimension にて表現される．

C．4．7　Assessment, Goal, Plan

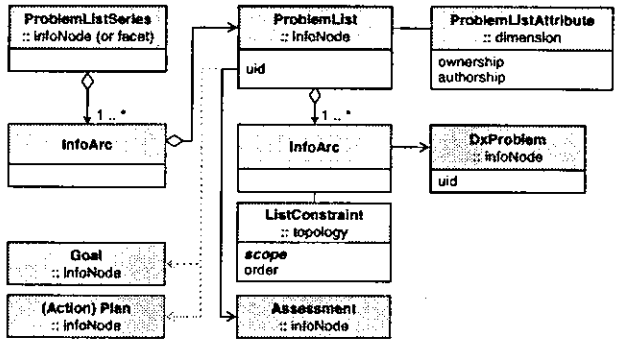
プロブレムリストについても対象予定の臨床医師に preliminary な調査を行ったところ次のような知見を得た：

・有意義な診療経過モデルを構築するには，評価（Assessment），診療目標（Goal），診療計画（Plan; Action Plan）の記述は必要である．
・特に評価そして診療目標の設定は必須である．

臨床現場のこれらの意見を踏まえて，参照実装においては，これら三者も付加することとした．

ただ（B.3）に依拠すると，三者のうち評価は問題(形成)空間に存している．しかし他の二者が属する subdomain は，各々，目標空間と解決空間である．

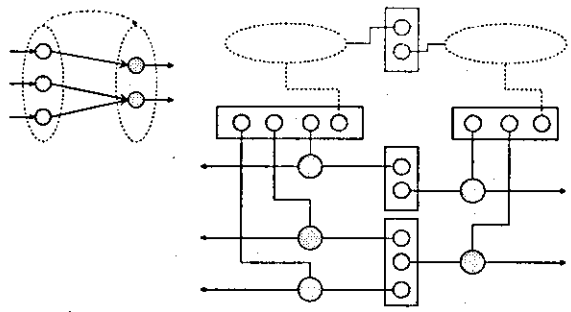
よって Goal および Plan を，問題(形成)空間に存する ProblemList の配下に包むことは，本来，妥当ではない．



しかし第二年度は効率的に実装作業を遂行するよう求められていることから，初年度に構築済みの「病名 Composer」モジュールの改変を最小限とするべく三者は全て，ProblemList の配下に配置することとした．

C．4．8　病名/プロブレムの変遷

CSX model による病名/プロブレム変遷の定式化も比較的容易である．



変遷表現の範囲は（B.2）に記した『変遷記述に必要な述語群』を『変遷関係』に置換した．

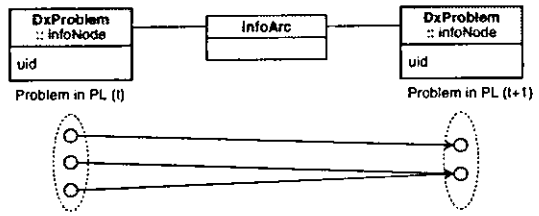
病名/プロブレム変遷の記述形式を規定する際

には，二つの選択肢を採りうる：

・プロブレム変遷のみ記述する
・プロブレムとリストの変遷の双方を併記する

CSX model における表現の明晰さと解析解釈の容易さとを考慮して，後者を採用した．

そして変遷関係は，￥変遷 arcScope における変遷 infoArc にて表現することとなる．



次に，変遷表現における語彙選択においては，その視点を確定しておく必要がある．視点決定には幾つかの選択肢がありうる：

・時刻 t-1 プロブレムリストの元から 時刻 t プロブレムリストの元への変遷関係を，時刻 t-1 側から見て infoArc[@kind]の値を決定する
・時刻 t プロブレムリストの元から 時刻 t-1 プロブレムリストの元への変遷関係を，時刻 t 側から見て infoArc[@kind]の値を決定する
・個々の元における変遷関係の出入状態によって infoArc[@kind]の値を決定する

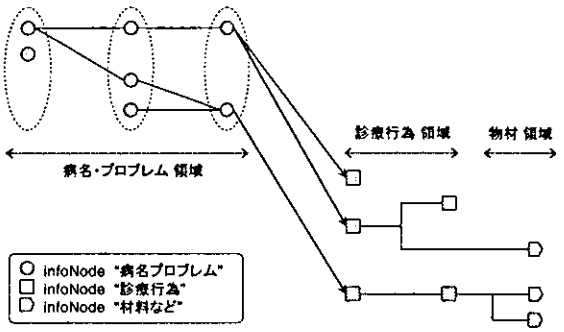
時刻 t-1 で分岐があり時刻 t で合流がある例で考えると，一番目では記述困難となり，二番目では時刻 t-1 における分岐情報が暗黙化される．逆も同類の問題を生じることとなる．よって，三番目を採択した．

C．5　病名診療行為連関の構成

C．5．1　設計範囲

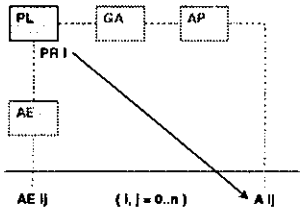
C.5.1.1　連関に関する概念設計

病名変遷と診療行為連関の形成は，下図に示す概念モデルにて表現可能である［医療情報学23S：962-966,2003］：



C.5.1.2　思考過程モデルとの対比

設計実装の範囲は（B.3）に記している（下図のPL：Problem Listには PR：Problemが包含されている）．これを基に（B.6.3～4）を参照しつつ書き換えると以下のようになる．



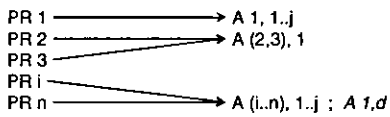
C.5.1.3　短絡性

上図に則って設計実装範囲を再定義するなら，各診療セッションにおいてPLに含まれているPR iについて，その診療セッションにて現実に実施された各A ijと相応したPR iとの間のみに短絡的に連関を形成する，となる．

思考過程ならびに診療経過の全過程を考慮の対象とするとき，本研究での試作実装は，このような短絡の範囲内での『経過過程』の捕捉（capturing）モデルである．これについても配慮しながら診療行為履歴の構成を考案した．

C．5．2　診療単位と連関支援構造

一つの診療セッションにおいて，幾つかのPR i（i=1..n）に対してA ij：Medical Action（i,j=0..n；加療行為）が実施される．これを言い換えれば，PR i と A ij との関係は多価（multivalent）である：



この多価性は，（B.3）に示したような，全ての思考と加療のプロセスが表現される場合には，個々の要素間の諸関係を辿ることで明示的に説明可能となろう．

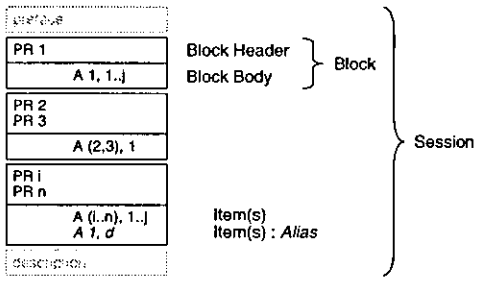
しかし本研究の範囲内での設計実装は短絡を前提しているので，上記の多価関係は，その様な短絡枠組のなかにおいて，むしろ可及的に精確に記述される必要がある．

さらには診療スパイラルでの多元写像を意識するとき，三つの視点を保持しつつモデル設計される必要がある：

- Human interfaceにおいてもITシステムの内部処理においても，関連付けのコストが極少化されうること
- 関係関連記述を容易とするために，投射元が一塊化されること
- 入射元は事実に即しつつ網羅されること

そこで先ず（i）投射元をクラスタ化し，次に（ii）必要な場合には入射元にaliasを導入し，（iii）双方の関連付けにはツリー構造を前提して親 node から子 node を投射することとし，（iv）その構成をそのままデータベースに格納する，という戦略とした．

これを前図に適用すると下図を構成できよう：



一つの診療セッションは一つ以上の「診療ブロック」から構成され，診療ブロックではBlockHeaderからBlockBody（の個々要素）への投射関係が記述された，とする．なおBlockHeaderでは必要に応じてPR iのクラスタ化を許容し，BlockBodyには必要に応じて他のBlockBodyに存在するA ijをaliasとして配下要素に包含する，という構成である．これは（B.6.3）に示したデザインと同値となる．

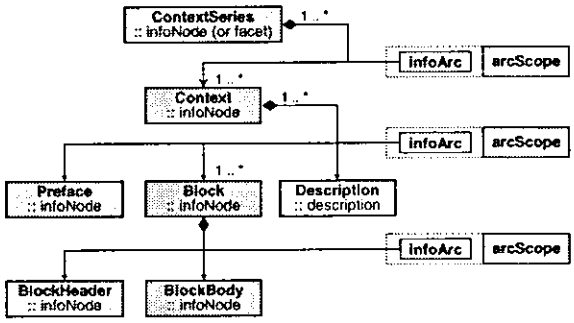
なおAijには，子を持って機能すべき情報塊（＝括り）が存在することを許容する．例えばオーダ塊がこれに相当する．括り情報塊は“container”infoNodeとして扱うことになる．

C．5．3　クラス化

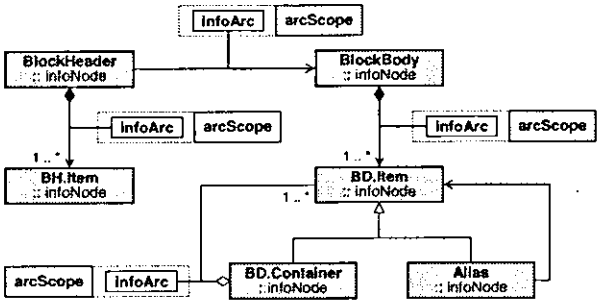
上記の結果を基にしつつCSX modelから必要なClassを生成していくと，BlockHeaderおよびBlockBodyについては次図の如くなった．

ここで，クラスContextは診療セッションのうちProblem subdomainを除いた情報塊である．またクラスContextSeriesは診療プロセスからProblem subdomainを除いたもの，つまりはContextの連から構成される情報塊である．

なおinfoArcは関係Classとして機能している．ただしarcScopeの規定する視座意義や視野範囲を超えることはできないことを模式的に示している．



次にBlockHeaderとBlockBodyは，それぞれ，複数のBlockHeaderItemまたはBlockBodyItemを格納でき，またBlockBodyContainerはBlockBodyItemを再帰的に格納可能であり，Aliasは（他の）BlockBodyItemを指し示している状況を下図に表現する．



BlockHeaderItemは，BlockHeaderとBlockBodyに仲介されつつBlockBodyItemを指示している．

関係の意義

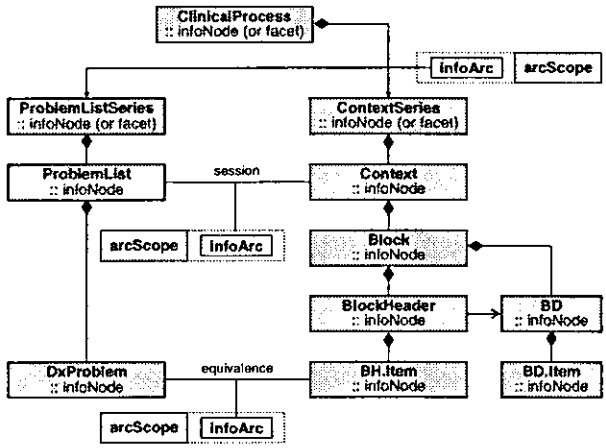
ここで，この関係 BH :-> BD の意義は，他のそれとは異なっていることに御留意願いたい．すなわち，この関係は特定の加療行為の事由や根拠等を示しているのである．

その一方，ここで挙げたその他の関係は，情報塊の構成あるいは包含を意味している．

このような意義の差はarcScopeとinfoArcのmeta-attributeであるcategoryやkindの値で表現されることになる．

C．5．4　相応性と連関

さらに，DxProblemとBlockHeaderItemの同値性を明示することに加えて，Problem subdomainとSolution subdomainとを連関させるために，下図の関係を構成することとした．



個々の関係を下方から上方へと説明していくと，下の関係はDxProblemとBlockHeaderItemとの同値性を表現し，中の関係は診療セッション全体を構成し，上の関係は診療スパイラルの全体を構成していることになる．

C．5．5　連関の記述について

さて病名-加療行為の連関，つまり特定の加療行為の事由や根拠の表現手法について述べる．

可能性のある手法として，直ぐに三種を挙げることができよう．それぞれを検討した．

C.5.5.1　論理関係

含意（implication：⇒）によって示すことができよう．ただ記号論理のみでは他の状況等を表現しやすいとは云い難い．

よって他の状況を付加する必要のない場合にのみ用いるか，あるいは後述する深層格と併用しながら利用することとした．

C.5.5.2 修辞関係

表現語彙を豊富に有することから記述は容易となるものの，逆に記述「揺れ」が生じやすく，揺れが存在したなら機械解釈の際のコストは増大する危険性を孕むことになる．

よって修辞関係での表現は避けるよう努めた．

C.5.5.3 述語と深層格

深層格表現は，適度な付帯状況，すなわち動作主格や対象格や道具格そのほかを付帯しつつ，理由格にて事由や根拠を明言できる点で有利である．

CSX model では，述語は arcScope で，また格は infoScope で表現できることから CSX model への馴染みも良く，また自然である．よって当面，これを活用することとした．

ただし，以下の点に留意せねばならない：

- Fillmore の流れを汲む格範疇は所作を主体とした分類のため，補格や属格の表現には無力である．
- ある context において，各変項の深層格を同定するには相当の人的コストを要する．
- 修辞関係との境界は曖昧である．

第一点は H12-医療-009 での分担研究の成果を継承しつつ，これを流用した．

第二点は，本研究の参照実装に必要となる事項（＝試作範囲）についてのみ，主任研究者が，その作業を行うこととした．なお人的コストの投入については，実装設計において唯一回のみ実施すれば良いことである．

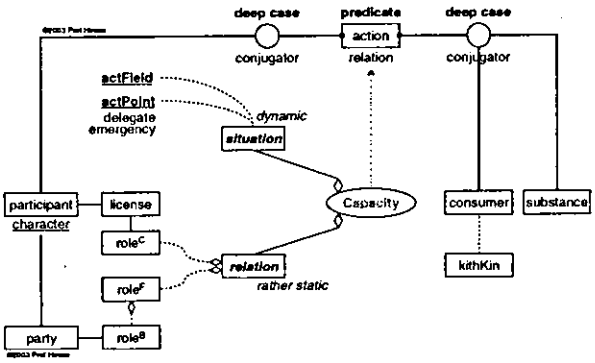
第三点については，本研究の範囲を超えるため，ここまでの検討で終えることとした．

C．6　役柄配役立場モデル

C．6．1　要約

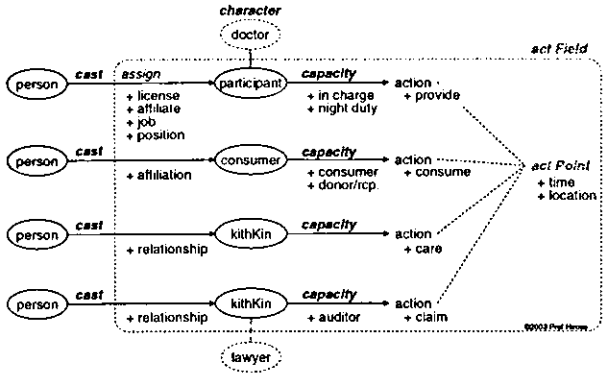
要約すれば次のようになる．

・権限を管理するには，先ずは，サービス提供者（Participant）の権限根拠を管理する必要がある．
・権限根拠には，所属機関の所属部署や国家資格などといった静的な事象に基づく役割のほかに，現場において動的に発生する事象に基づく動的な役割も，厳に存在している．
・サービス消費者（Consumer）にアクセスする際には，上述した権限根拠は「立場（Capacity）」として要約しつつ捉えることができる．
・正しい権限付与とは，権限根拠の集約体である立場に対して 必要十分な権限が与えられる，ことを謂う．



・各関係者（Participant，Consumer，kithKin）は，ある特定の所作（action）のなかで，ある場（actPoint）に統合される．
・その際，各々の役割関係あるいは意味関係は，述語と深層格（deep case）とで表現する．これは情報モデル（C.2.2）と相応している．
・個人は様々な役柄（Character）を持つことができ，そのうえで更に，「場」に応じた立場を執っている．
・してみるとヒトは或る「文脈場（actField）」に存在する（すべき・するであろう）何らかの役柄に配役（Cast）されつつ，出番を待機していることになる〔待機時間 0 の場合もある〕．

このような状況は次図のように模式化できる．よってこのような概念モデルを役柄配役立場モデル（Cast-Character-Capacity model：3C model）と称することとする．



C．6．2　行為点の形成

全構成の中心的な単位は行為点（actPoint）である．

actPoint は時刻，場所，関係者から成る．関係者は通常複数となる．場合によっては場所も複数となることがある．ただしこの場合でも，システム管理時刻およびシステムが管理するべき actPoint 連番は，一つの actPoint ごとに唯一である．

actPoint は PKI（Public Key Infrastructure）における点管理の思想に沿うよう設計した．

C．6．3　行為場の形成

次に行為場（actField）を構成する．actField は actPoint の連である．

すなわち，一人の患者の一連の診療に関わる actPoint の連なりである．よって actField は，少なくとも暗黙的には目的や解決すべき問題をもって時間推移のなかで綴じられることになる．ゆえに（B.3）との強い相応性がある．

なお個々の actPoint の関係者（登場人物）は種々であり全 actField を通じて一致するのはむしろ稀であろう．場所も同様だろう．

C．6．4　行為者

行為者（Actor）は，具体のヒト（Person）または組織単位（Party）である．

Person は，後述する「役柄」に「配役」される．よって後述する「実施者」とは相異なる．

Party は，後述する「組織単位縦列」構成要素となりうる．

なお現時点では，Party は Character に cast されない，と考えることとする．というのも 3C model の主眼とは，Person が執行すべき行為を支える権限に関する根拠の管理にある，からで

ある．

C．6．5 役柄

役柄（Character）は三つしか用意していない：

・実施者（Participant）
・消費者（Consumer）
・親類縁者知人など（kithKin）

Participantはhealth service提供者である．Consumerはhealth serviceの消費者である．kithKinとはconsumerの親類縁者知人などである．

なおCharacter自体は三種のみとしているものの範疇子属性によって拡張は可能である．ただそれよりも，各々が執りうる「立場」は様々であるし，権限管理においては立場のほうが重要である．

C．6．6 配役

Personは一つ以上のCharacterに配役（cast）されうる．

Personは Characterに castされなければ，actPoint（場面：scene）には登場することができない，とする．ただし actFieldには予め登録されていることがある(多い)．

C．6．7 立場の適用

Characterは，actPointで立場（Capacity）が適用される・または選択する．Characterは，あるactPointにおいてあるCapacityのもとに行為（Action）を実施（act）する，とする．

言い換えればCharacterは，Characterであるのみでは何らActionを actできないし，そもそも当該actPointでのCapacityが前提されないCharacterが actPointに登場しても無意味ゆえ，actPointには登場できない，あるいは，登場しないのである．

C．6．8 組織単位と役割

組織単位（Party）とは特定の役割（Role）を担った同定可能かつ機能的なヒトの塊である．Partyは，それが存する界において一つ以上のRoleを持っている．

例： 医療機関，診療科部，診療グループ，Taskforce，委員会など．

Partyには，Partyの Roleに即したCharacterが求められる．必要とされているCharacterを確保するためにPersonが castされ，同時に，そのPartyに配属（assign）される．

C．6．9 組織単位縦列

組織単位縦列（Fleet）は，多段のPartyにて構成された複合Partyである．あるいは，組織単位縦列におけるある境界Partyまでの Party列である．

通常，祖先Partyの Roleは子孫Partyへ継承される．継承結果としての Roleは，Fleetの Roleに集約されることになる．

Fleetは実装システムにおけるデータの扱いを容易とするために設けられた．

C．6．10 根拠の継承と集約

Personは，Personが持つ資格や技能証明などの属性に応じたCharacterに castされ，そのCharacterは特定の Partyや Fleetに assignされて所属する Partyや Fleetの Roleを引き継ぎ，Character（Participant）の Capacityは特定の actPointにおける・その場のsituationに応じた Roleをも集約したうえで決定される．

このような事情から，Capacityはけっして静的ではありえないのである．

C．6．11 権限獲得と付与対象

Characterは特定の場において特定のCapacityを『執る』ことで初めて特定の Actionを actするための権限（privilege）を『獲る』ことができるのである．

言い換えれば権限とは Capacityに基づきつつCapacityに対して与えられるのである．

権限付与の対象が Personまたは Participantであると考えるのは完全な誤解もしくは思慮不足であり，「である」故の権限付与ではなく「立場に基づく責務を遂行する」故の権限付与なのである．

C．6．12 モデル内の要素

3C modelを構成する要素を列挙する：

・行為者
・役柄（Character）
　・サービス実施者（Participant）
　・サービス消費者（Consumer）
　・親類縁者知人など（kithKin）
・組織単位（Party）
・組織単位縦列（Fleet）

・配役（Cast）
・行為場（actField）
・組織で規定される役割（roleInHosp）
・場において要求される役割（roleToPatient）
・権限根拠（PrivilegeBasis）
・立場（Capacity）
・行為点（actPoint）
・所作（action）
・権限（Privilege）

つまり 3C model を完全に実装するには，先ずこれらの要素を全てクラス化する必要がある．

そのうえで対象業務に要する付帯Classを準備し，さらに業務の利便に考慮した機能を付加していく必要がある．

上記の要素に認証Certificationを加えただけでも，大雑把に見積もっても以下の如くなる．

・actField（actPoint+,Fleet+）
・actPoint（actCharacter+, Location+, Time）
・actCharacter（Character,Capacity）
・Capacity（PrivilegeBasis）
・PrivilegeBasis（License*,Role+,..）
・Role（roleInHosp+,roleToPatient*）
・Directory（Character+,Fleet+,Party+）
・Character（Person,Fleet）
・Fleet（Party+）
・Pearson（License*,Insurance*,..）
・Certification（Person,key,pass）

C．6．13　必要な拡張

加えて，たとえばCSX modelによる記述枠組を交換しようとした場合には以下のClassが必要となる．

・Case（actField+ | actPoint+）
・Holder（actPoint+）
・Concerned（actPoint+）

簡約すれば，クラスConcernedは現場における所作への直接的な関わり，クラスHolderとは文書の作成や認証や発行への関わり，クラスCaseとは当該事例に関する関与者等の広がり，を表現することになる．

これだけの量の試作実装を，本研究の主主題と同時に実施することは不可能なので，参照実装では，範囲を絞って成果の一部を適用することとした．

C．6．14　試作アプリでの対象範囲

応用した要素や考え方は以下とした：

1）Fleetの構成
2）FleetへのConsumerの登録
3）FleetへのParticipantの所属
4）Capacityの宣言
5）Capacityに応じたアクセス可能範囲の制限

とはいえ，この機能範囲を示すのみでも，

・3C modelを適用しても操作負荷は増大しない
・立場に拠って付与される権限が変化しうる

ことの実感による理解には十分と思われる．

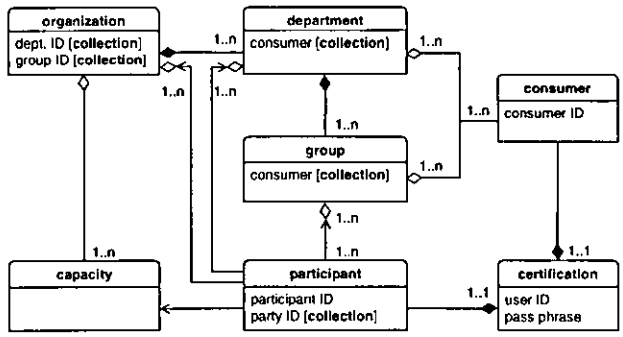
C．6．15　実装クラス

実装したClassは以下の通りである：

・certification
・participantDir
・consumerDir
・organizationDir（organization, capacity）
・party（department, group）

いずれもSchemaはxsdまたはwell-formed xmlにて記されており構造は単純である．とはいえ3C modelに即して，次のような特徴がある：

・certificationに登録されているPersonは，ParticipantにもConsumerにもなりうる．
・Partyの再帰構成によってFleetを構築する．当然ながら種別（organization, department, group）の階層秩序性は保持している．



なおClassへのcollectionの持たせかたは，参照実装「診療システム」での絞り込み効率に配慮しての方策である（C10.1）．

C．6．16　登録ルール

上記ほか，以下の小さなルールを用意した：

・UserAが作ったpaticipantAのpartyIDには，UserAの操作により，UserAの作ったhospitalA, departmentA, groupAのうち，各々，自身より下位のpartyIDのみをcollectできる．
・UserAの作ったconsumerAは，UserAの操作に

より,UserAの作ったfleetAにcollectできる.
・UserAの作ったconsumerAは，UserAの操作により,UserBが作ったfleetBにcollectできる.
・UserBは，fleetBにcollectされているconsumerAを，そのfleetBにおけるconsumer collectionから削除できる.

このルール実装によって，簡易ではあるものの，場の形成を実現させた.

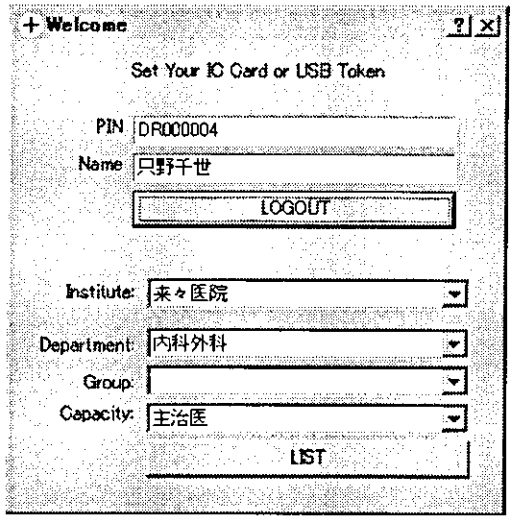
よってConsumerの紹介など地域連携をも模倣できるようになっている.

C．6．17 ログインと記録

参照実装へのログインの際,end-userは以下を尋ねられることになる：

・account
・password
・organization
・department
・group (optional)
・capacity

そして後三者はParticipantの氏名と共に記録されることになる.



Capacityは，ログインの際または患者選択の際に決定されなければならないが，妄りにend-userの負荷を増加させることはない.