C. 研究結果

C．1　名称変更と軽微な修正

前年度までのCSX modelに用いていた各要素名またはクラス名は一新した．以下に新旧対照を掲げる（左手が旧，右手が新）：

中核要素

・Substance	infoNode
・Substance@objectID	infoNode@uid
・Conjugator	infoArc
・Conjugator@objectRefID	infoArc@ref
・Relation	arcScope
・Relation@objectID	arcScope@uid

周辺要素

・substance.Code	nodeCode
・substance.Construe	construe

名称変更した事由は，名称の適切化とともに，グラフ構造を意識しながら，より抽象的な名称としておくほうが，思索における混乱を回避しやすく，また本質を的確に表現しうると思われたからである：

・抽象度の高いレベルでのモデル等の設計
・抽象から具象に至る際のobjectの規定

なお前年度の試作実装にて活用されなかった属性Substance@occurrenceは削除した．

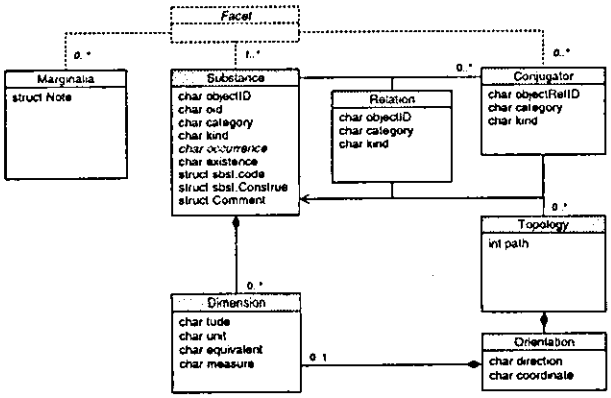
逆に，dimensionにおけるデータ型を明示するために属性dimension@dataTypeを導入した．

またarcScopeにも要素commentを加え，同時にMarginalia/NoteをMarginalia/commentに変更して統一した．

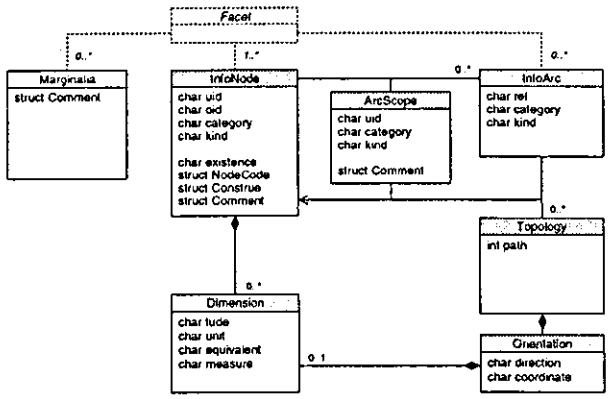
ここに名称変更し，軽微な修正を加えた新しい要素ダイアグラムを今年度の開始点として，本研究を実施した．

よって（C.2）～（C.4）の検討の後，これは更に改変され，そして（C.6）に今期の成果である情報モデルが示されることとなる．

【左図（前年度）】



【右図（今年度開始時点）】



C．2 制約表現能力の付与

本研究を開始した時点のCSX model (B.4) とは，本質的には，事実表現およびパタン表現を為すのみであった．

よって事実やパタンの記述と制約の記述とを分離し・かつ・明示的に扱うようにしたなら，加えて，他の記述表現枠組ではなくCSX modelの枠組みのなかで自己完結的に表現できれば，CSX model 全体としての表現力と明晰性は一挙に向上するとともに，知識表現能力を獲得したことになる．

前年度はその前哨としてCSX modelによる制約表現の可能性に関する詳細な考察を実施した．その結果，CSX model における制約表現記述は以下のうちいずれかを採って記述可能であることが明らかとなった：

・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

今年度は，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する．ただし，制約表現に関する試作実装については，もともと，本研究計画の範囲外である．

C．2．1 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
・関係の基数（多重度）
・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）

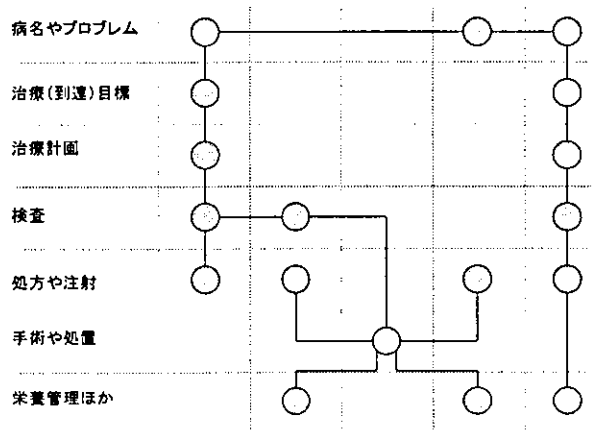
多重グラフにおけるノード間の制約

・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

さて診療情報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る必要がある．

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，今年度は，上記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．2．2 制約類型

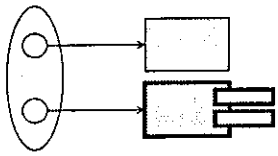
対象範囲に関する制約の類型としては以下のように要約することができよう：

・データ型
・値範囲限定（初期値を含む）
・基数（多重度）
・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

なお以下に記述形式を順次記していく．

C．2．3 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．

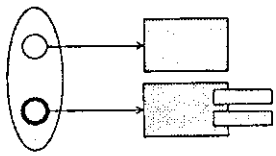
なお dimension[@equivalent]には，例えば“SmallerThan”などの比較叙述詞なども既に用意されている．

C．2．4 基数（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．

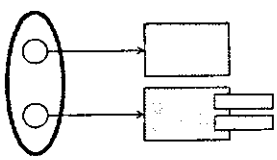


これは定義体 infoNode の context を形成する arcScope において機能することを想定する．この infoArc が指し示す子 infoNode の基数が定義される，と解釈する．

C．2．5 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．

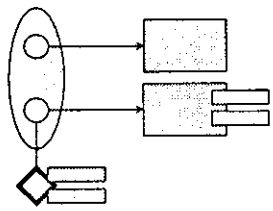
arcScope[@choice="TRUE | 1"]



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が，上記のような値範囲としておく．ただ1以上の正整数 n を指定し，かつ選択肢の要素を n 個以上存在させるなら，記法としての矛盾は生じない．

C．2．6 比較

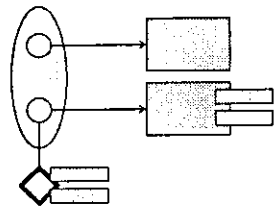
比較または制約 arcScope において，相異なる二つの infoNode 間の物理量または概念量を，infoArc/topology を活用して比較表現する．



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 順序と共起

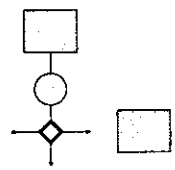
これらは infoArc/topology[@path]にて容易に表現できる．この値範囲は整数である．格納されている値が同値であれば共起と解釈する．



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．

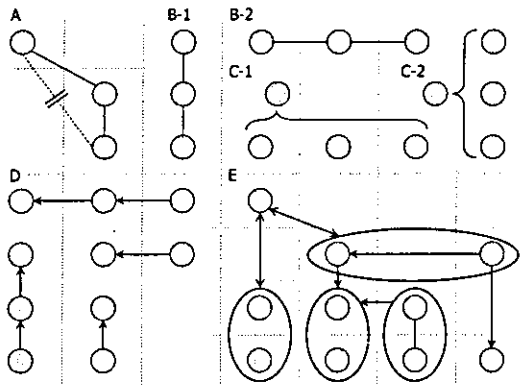
C．2．8 存否

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．



また，これらの表現は，少なくとも暗示的には「限定された場」を要求していることに御留意願いたい．

C.2.8.1 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が，順序や共起と関わる場合には強い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*]"]

C.2.8.2 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

C.2.8.3 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

C.2.8.4 隣接性

隣接性とは，他のinfoNodeとの直接連結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*]"]

C.2.8.5 距離（時間）

時間的な距離は，infoArc/topology配下のorientation/dimensionで表現する．Dimensionは，相対時刻も時間間隔も表現可能である．

なお領域（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よって，結合の可否を表現することとする．

C．2．9 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」ような値を有する包括的クラスとしてのinfoNodeを準備し，それらのinfoNode間にて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，可能である：一つには（a）CSX modelではinfoNode間のvertical relationを表現することができる，いま一つに（b）infoNode[@category and @kind]に格納される属性値自体がcode schemaつまりtaxonomy下に表現されうる，からである．

したがって，ドメイン親和性に関する制約を記述する際に為すべきことは，ドメイン自体を表すスーパークラスinfoNodeに対してinfoNode[@uid]（and/or infoNode[@oid]）を与え，そのようなinfoNode間における制約arcScopeを記述し，かつこれを共通基盤として用いることができればよいこととなる．

C．3 再帰的枠組での粒度依存性問題の解消

C．3．1 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 値の枠組み
・大粒度 の情報塊への 参照枠組み

これを無視したまま定式化を進めると，早晩，管理不能もしくは破綻に至ると思われたため，これら二点にも適切に対応できるようすべく，情報モデルを改変し，同時に情報モデルの内部構造に同型対応順応性を与えることとした．

C．3．2 大域粒度における範疇

さて infoNode は，category ならびに kind という属性を持っており，これらによって，その infoNode が担っている情報が，どのような domain または subdomain に属する（または焦点されている）ものとして扱われているのか，を示すこととしている．なお infoNode/nodeCode は，具体の事物内容を示している．

このような枠組みは，情報粒度が比較的小さい場合には何の障碍もなく機能することは，前年度までに確認している．

しかし infoNode が包含する内容が大域となった場合には，範疇属性の値の枠組み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るからである．

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．従って，後者のような値体系を格納するべく，要素を用意することとした：

・scope[@domain @category @kind]

そして Marginalia を description と改称し，description は以下のように配置することとした（XML/DTD にて示す）：

・description (comment?, scope?)

C．3．3 同型対応順応性の付与

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

なお scope には，その facet または infoNode の形成に関わった（広義の）関与者を示す要素である concern[@concern.refID]を持つこととしている．

C．3．4 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 ID, 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みを入手できたからである．具体的には，

・種々の情報塊の集積が必要な診療文書も容易に構成でき，加えて（C.3.2）や（C.3.3）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 なども facet 等として参照できる

・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

・閲覧参照した過去の診療情報のスナップショットを保管したりできる

ようになった．これらは全て(C.7.4)の末尾に示した diagram と同等である．

さらには（B.14.2）の(3)も解決されたことになる．すなわち SYSTEM は，統括管理すべき情報塊について，それらが発生した順序と時刻に従がって，順序時刻管理“container”に格納（＝参照）できるようになったからである．

斯くの如くに，infoArc が参照すべき情報塊の範囲が広がって infoNode のみではなくなったことから，属性 @objectRefID は @ref と改称することとした．

C．4　その他の追加事項

次の追加を行った：

1) infoNode が画像や音声などの外部ファイルを同定できるよう infoNode@rid を設けた．

2) infoNode の『あり方』を規定した arcScope への参照の利便のために，arcScope の @uid の値を格納すべき infoNode@bearing を設けた．

3) infoNode が複数のコード体系のコードを同時に格納できるために nodoCode の多重度を n とし，同時に nodeCode@priority を設けて，処理優先に関する示唆を与えることとした．

4) description に @uid を加えた（既出）．

三点目によって（B.14.2）の(1)は解決された．また四点目によって XML への直列化は W3C XML Schema のみならず RELAX NG でも行えるようになった．

C．5 Ontology と Meta-modeling

C．5．1 Ontology

オントロジ（ontology）とは凡そ以下の意味として理解されている〔矢澤〕：

α）人工システムを構築する際のビルディングブロックとしての基本概念や語彙の体系
β）概念化のための明示的な規約または記述書
γ）特定の目的のための世界認識に関する合意すなわち共通概念や概念関係
δ）モデルが対象とする世界の概念と，概念間の諸関係を明示する枠組や方法論
ε）エージェントが認識できる形式で記述された意味体系，またはそのような意味体系構築手法による事象の記述

本研究ではδの意味で，この語を用いている．なお ontology には様々な枠組が提案され，またされ続けている現状である．

C．5．2 Meta-modeling

対象世界の情報客体（information object）を整理し，電算機システムに実装しようとする際，様々なモデル化手法が試みられてきた．

近年ではさらに，別個のモデル間で情報交換をする際に，あらかじめ meta-model（generic Class など）自体を記述（meta modeling; meta meta-expression）することで，相互の相同と相違を確認して，mapping やデータ授受の正確さを期する動きがある．

そのような分野（CASE など）では，各レベルの客体は下表左手のように呼ばれることがある．

Layer	Example	**Modeling Layer**	**Handling Object**	*Buddhism Layer*
Meta meta-model	(CDIF) MetaEntity, MetaAttribute (MOF) Class, MofAttribute	Meta Modeling Meta meta-expression	1. meta meta-entity 2. meta meta-relation 3. scope of meta meta-relation	空
Meta model	(UML) generic Class, Attribute, Association (RDB) generic Table, Column, Key		4. Generic i.e. (meta-) information object	蘊 得・非得
Model	(UML) user defined Class, Attribute, Association (RDB) user defined Table, Column, Key -- Doctor, Patient, cure	Modeling Meta expression	5. constraint 6. information object i.e. user defined entity 7. horizontal and vertical relation 8. scope of relation	
User Object -- things -- matters	(UML) Instance (RDB) Record -- Doctor#1, Patient#3, -- Doctor#1.cure.Patient#3	Concretizing (Instantiation) Expression		法

C．5．3 Meta meta-expression

ただ上表左手は適切な分類と用語と云えない．

・各客体は 各モデリング作業においては，各々入出力となっている．
・Meta-modeling レベルでは 各客体を Class と Attribute に明確に分離できるとは限らない．むしろ各客体は，いずれとしても存在しうる．
・Meta-model と Meta meta-model とに配された客体は，上表左手のごとく区分けできるほどに境界明瞭とは云えず，むしろ上表左手は客体というより枠組を与えているに過ぎない．
・にも関わらずそれらを客体と呼び層と呼べば，各客体は各層に所属しつつ“実存している”，という誤解と混乱とを誘発する惧れがある．

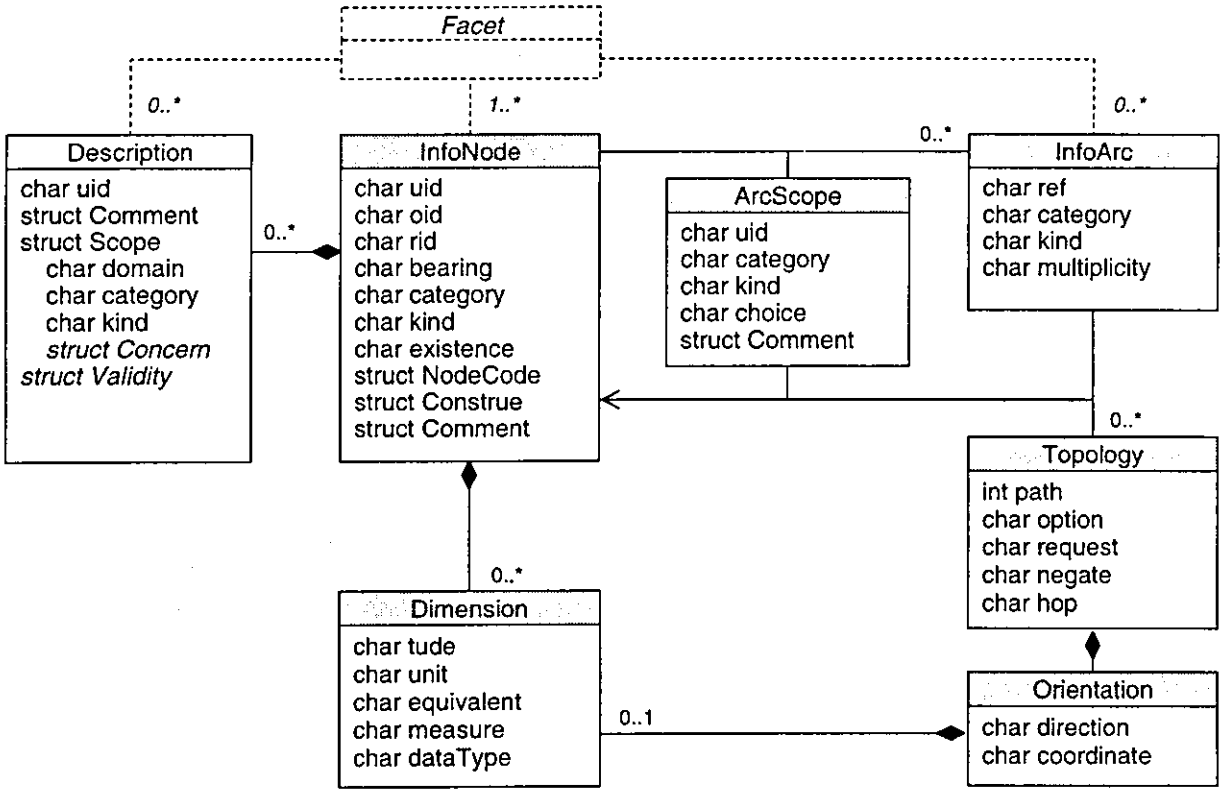
よって本報告書では上表中央の概念を用いることとする．ただし用語については利便と理解支援のため上表左手も用いることがある．

そもそも meta-modeling に関する考察と経験はアナリストやモデラーまたは西欧の専売ではなく，むしろ東洋にこそ深遠なる発想と思索の悠久があったのである（上表右手）．

C．6 情報モデル CSX

C．6．1 CSX model 要素ダイアグラム

前述までの研究成果から下図の情報モデルが得られた．なお各 meta meta-information object の意義，役割，機能の詳細については，本報告書，前年度報告書，ならびに H12-医療-009 の資料を参照願いたい．



なお struct は参照実装では Class として扱っている．また description/scope/concern n は今年度使用しておらず description/infoNode として簡易に代用した．また Validity も使用していない．

C．6．2 CSX model の本質

CSX model とは，ontology でもあり，と同時に meta-modeling 手法（meta meta-expression 環境）でもある（C.5）（D.1.1）（D.1.2）（D.2.5）（D.2.8）．

ただし CASE で云うところのそれとは異なることに御留意願いたい．

そして CSX model は，ontology という paradigm において，さらに meta-modeling，modeling，concretizing に至るまでの各種の情報客体を扱うことが可能な framework を与えているのである：

- Meta meta-information object
  - Meta meta-entity
  - Meta meta-relation
  - Scope of meta meta-relation
- Generic (meta-information object)
  - generic Class
  - generic Attribute
  - generic Association
- Constraint
- Information object (user defined)
  - Class, Attribute, and Instance
  - Horizontal and vertical relation
- Scope of relation

その仕掛けは，まさに情報モデルのデザインに存している．すなわち，

- Ontologyに即しつつ極めて抽象度の高い中核meta meta-information objectを提供：情報素(infoNode)，関係素(infoArc)，関係視座素(arcScope)
- 補足的に，極めて抽象度の高い計量表現要素，計量素（dimension）を提供
- 補足的に，抽象度空間での“位置”関係を表現する要素，位相素（topology）を提供
- 情報素あるいは情報塊が，他と関係を結ぶ際には，その関係を前提する視座意義や視野範囲を必須とするように規定
- 具体事象を表現する情報塊は，この枠組の内で再帰的に構成していくよう規定
- 各要素は，それが何を表現しているのかが決定されるにはmeta-attributeに，コード体系に定座するコードを代入するよう規定

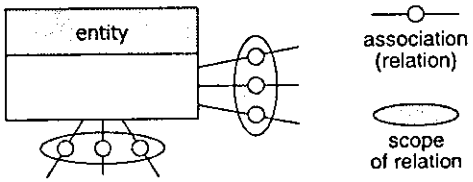
したからである．

極めて抽象度の高いmeta meta-information objectsは，その枠組のなかにおいて，コード体系とくにcode schema（もしくはtaxonomyを明言するコード体系）内に規定されたコードを格納したりnodoCodeで規定されたりして，具体性が付与されClass(model)とされていき，最終的には@uidが附番された時点でinstanceとされる（C.7）（C.11）．

大きな情報塊を構成したり文脈などを表現したりする際には，小さな情報素を順次組み合わせて構築していくことになる．

このとき各々の情報素は互いに関係づけられることになるが，関係を結ぶ際には視座意義や視野範囲などの箍が嵌められることになる．

これら二点によって，データ・ハンドリングにおける解釈の発散や混淆を防ぐとともに，処理効率を損なわないよう留意されている．



そしてmeta-modelingレベルでは，思想的には，ClassとAttributeを厳密に区別せず，何れもmeta meta-entityとしている点で，将に東洋的（仏法的）でもある．

そしてmeta-entityは，meta-relationおよびscope of meta-relationによって関連付けられつつ，具体のClassが構成されていくのである（C.8）（C.9）．

その結果，user objectに至るまで単一の情報モデルで表現することが可能となった．

これは，特定の対象ドメインに関する具体の大規模な情報モデルを構築する際にも，細粒度から大域粒度に至るまでの情報塊を扱いながら，要素数（Class数）自体は全く増加しないことを意味している．

このことはCSX modelの重要な側面を示している：

- CSX modelは小さくロバストでドメイン独立である
- CSX modelにおいて，ドメイン特異性は，代入される属性値，関係視座や関係視野のとりかた，および具体の制約内容によって示される
- CSX modelはドメイン記述においてsingle architecture modelを支援するので，界面にて発生する汚染や混淆を回避する

またCSX modelは，視座意義や視野範囲等において，述語と深層格・もしくは修辞関係をも扱いうることを想定している．したがって各種の静的な関係様相のみならず，所作等に関わる動的な関係様相等も記述可能としている．

さらにCSX modelは軸性を持っていない．より正確に云うなら，a prioriの軸性は規定してはおらず，単にtopologicalな枠組みを提供しているのみである．

ということは逆に，いかなる軸性も必要なだけ生成することができる枠組となっている．軸性というcontextは，scope of meta-relationとmeta-relationとで形作られることになる．

ある軸性における階層構造つまりtaxonomyにおける関係をvertical relationと呼ぶならば，horizontal relationについては，また別個のscope of meta-relationとmeta-relationとで形作られることになる．

C．7 直列化と具体化

CSX model は細粒度から大粒度までの情報塊を，グラフ構造のもとで再帰的に構成することを許容している．また，ある arcScope/infoArc で構成される context に定置された infoNode は，他の context において再利用される（参照される）ことを許容している．

加えて CSX model は関係（infoArc）ならびに関係視座の意義や関係の範囲（arcScope）を，独立した要素として明示的に取り扱っている．

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．

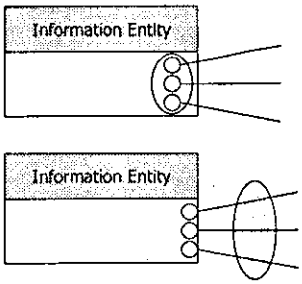
C．7．1 Root element の候補＃1

infoNode を root element に

記述形式としては機能することもあるが，root element を特別扱いするような XML Schema で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，xml 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．このような包含関係は xml の根源構造から生じてくる．



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 に同一の infoNode を定置する必要のある場合，それらの arcScope を同一の infoNode が同時に保有しなければならなくなってしまうからである．

このような形式的な表現状況を，通常，矛盾と呼んでいるのであろう．

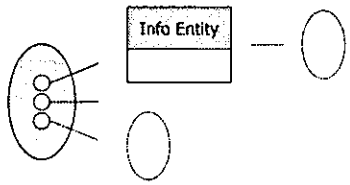
よってこの直列化方式は妥当とは思われない．

C．7．2 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.7.1) 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のかもしれない．

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．7．3 Root element の適任者

Facet を root element に

上記のいずれも 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 を設けることで，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．

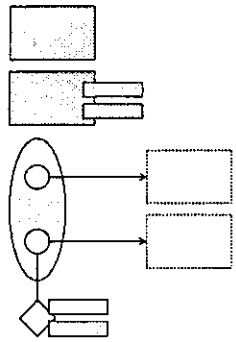
次に xml による直列化に際して上述した破綻の危険を回避する十分条件とは，infoNode も

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．7．4 直列化要件の整理

以上の検討を纏めると，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．



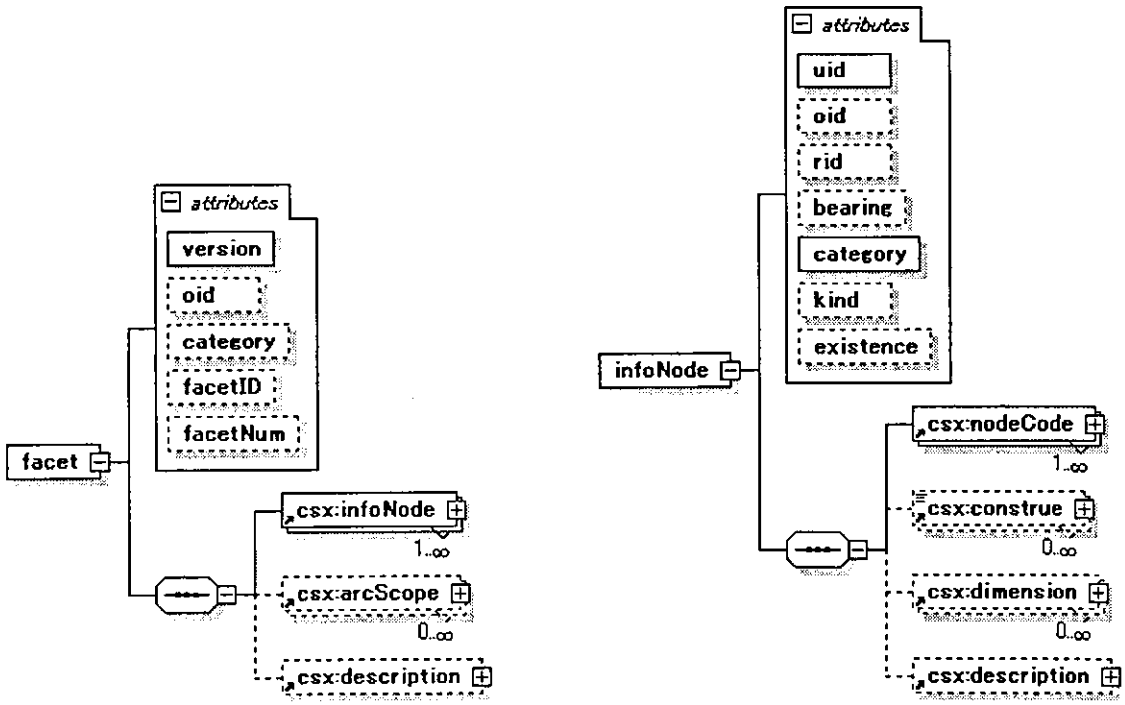
なおFacetは，より大きな局面においては，「そのFacetが包含表現している情報内容自体」が，別のcontextに定置される，すなわち別のcontextを形成するarcScopeのinfoArcによって参照されるべきこととなる（C.3.4）．

C．7．5 XMLダイアグラム

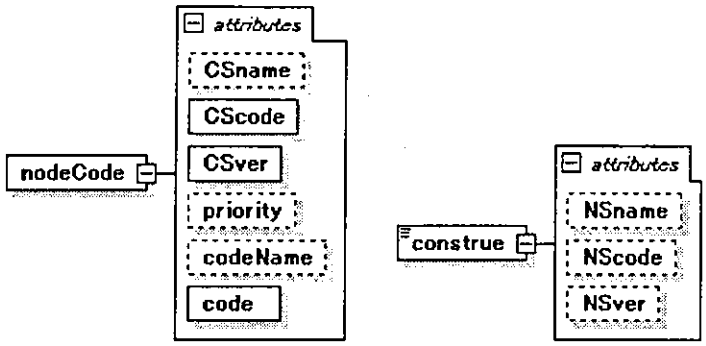
直列化はW3C XML Schemaで行うことを原則とするがRELAX NGによる直列化も許容した．選択肢を狭めないためである．ちなみに試作アプリケーションではRELAX NGに拠る要素出現パタンとなっている．なおxsdについては【資料】を参照願いたい．

以下にXMLダイアグラムを示す（Altova xml spy 2005からの出力）．

Facetならびに infoNode

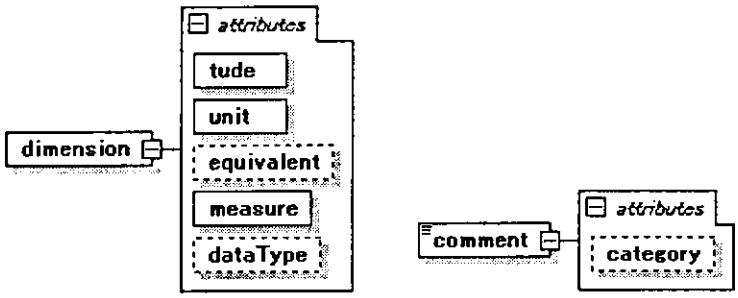


infoNode 配下の nodeCode と construe



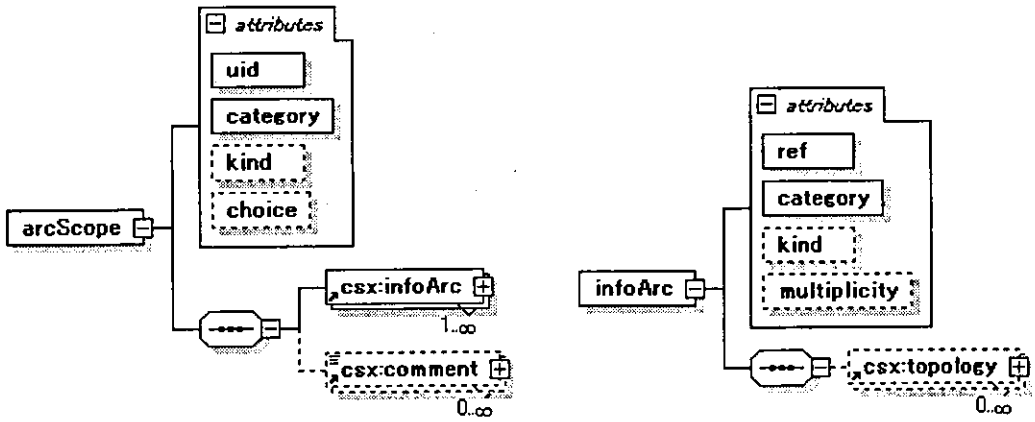
dimension と comment

これらは必要に応じて，親 element のなかで用いられる．



arcScope と infoArc

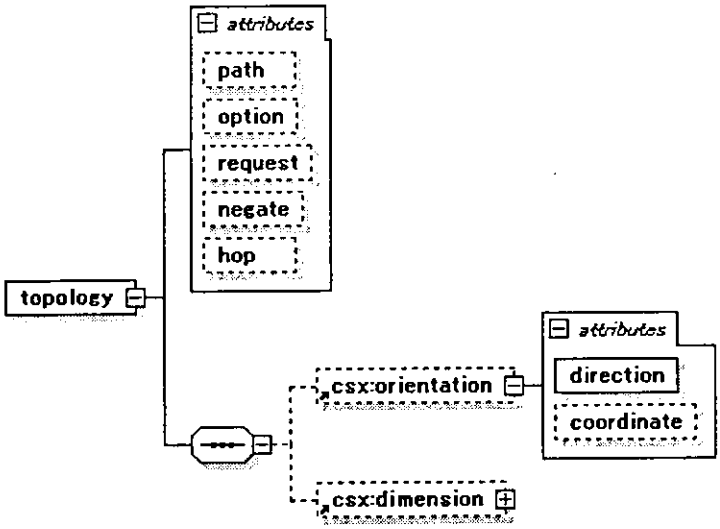
infoArc は arcScope 内に閉じられている．



topology と orientation と dimension

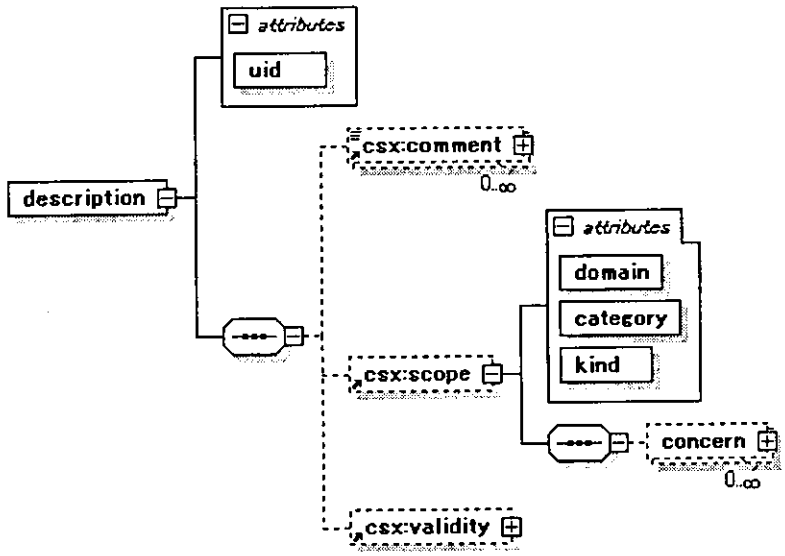
infoNode 間の関係に，抽象的または物理的な“位置関係”の詳細を与える．

なお orientation と dimension は，一括して topology 配下に置いている．



description と scope

facet または infoNode の配下に配される．



C．7．6 属性値の整理

前述までの成果による改変から，属性値の値範囲について再整理を施した【資料】．その際，EDR corpus の階層化範疇体系ならびに意味関係分類や，GDA（Global Document Annotation：大域文書修飾）で用いられているタグの意味関係分類をも参照した．

なお資料では W3C XML Schema によって提示したものの，記述形式は基本的に任意ではある．

C．8 病名プロブレム変遷の構成

C．8．1 病名構成の原型

CSX model による病名構成は容易であり，既に前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，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）．

前年度の成果概況は(B.13.2)の通りであるが，以下には改訂状況を踏まえて註釈する．

まず病名は四要素：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 と infoArc によって表現するとした．

そして病名の諸属性は，その構成体の諸属性を示す子エレメント 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.6.2).*

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．8．2 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，また妥当であると思われる．

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 内に包含されることとなった．

C．8．3 Stage と Intervention

病名プロブレム変遷と病名診療行為連関とを記述するとは，結局のところ，診療経過を記述することを意味している．

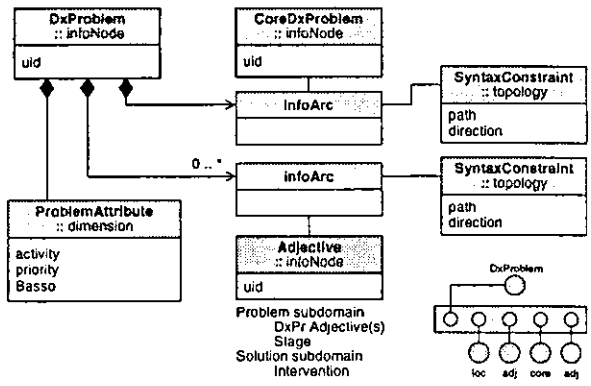
しかも臨床医師に対して試作システムの試用を依頼するとは，実診療ではないのであるから，結局，診療経過モデルの試作を依頼することを意味している．

前年度の終期から今年度の初期にかけて，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．

なおこれらは construe ではなくて infoNode に格納して arcScope で関連付ける構成が本来の姿ではある（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．

しかし今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，前年度に構築済みの「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．8．4 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り，既に前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ている）．前年度の成果概況は（B.13.2）の通りであるが，以下には改訂状況を踏まえて註釈する．

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 との関係を，arcScope と infoArc とを用いて表現することになる．

C．8．5 Assessment, Goal, Plan

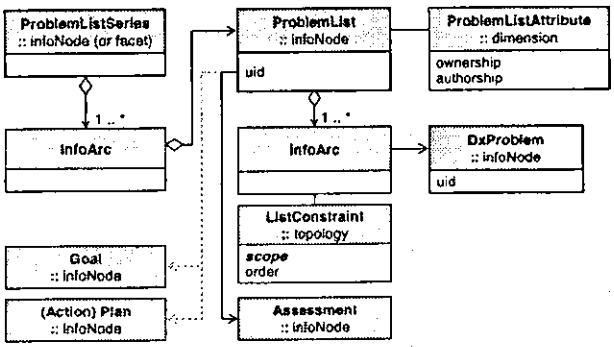
さて，プロブレムリストについても，前年度の終期から今年度の初期にかけて，対象予定の臨床医師に preliminary な調査を行ったところ以下のような知見を得た：

・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.2）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．

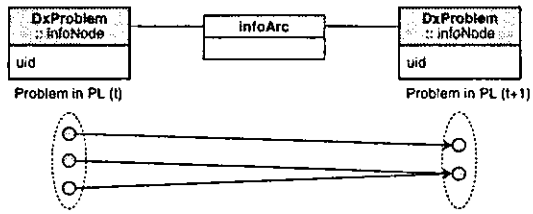
よって Goal および Plan を，問題(形成)空間に存する ProblemList の配下に包むことは，本来，妥当ではない．



しかし今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，前年度に構築済みの「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．8．6 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り，既に前年度に完了している．前年度の成果状況は（B.13.2）の通りであるが，以下に概説する．



変遷表現の範囲は（B.1）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．

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

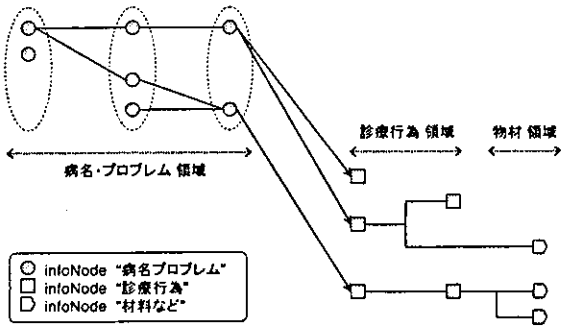
時刻 t-1 で分岐があり時刻 t で合流がある例で考えると，一番目では記述困難となり，二番目では時刻 t-1 における分岐情報が暗黙化される．逆も同類の問題を生じることとなる．よって，三番目を採択した．

C．9　加療過程の構成

C．9．1　設計範囲

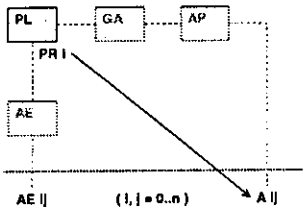
C.9.1.1　連関に関する概念設計

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.9.1.2　思考過程モデルとの対比

今年度の設計実装の範囲は（B.2）ならびに（B.3）で記している（下図のPL：Problem Listには PR：Problem が包含されている）．これを基に，（B.6.2）を参照しつつ書き換えると以下のようになる．



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』で現実に実施された各 A ij と相応した PR i との間にのみ，短絡的に連関を形成する

となる．なお，『診療セッション』については後述する．

C.9.1.3　短絡性

つまり思考過程ならびに診療経過の全過程を考慮対象とするならば明らかに短絡があって，その短絡の範囲内における『経過過程』の捕捉（capturing）モデルであり，また実装である．

とはいえ，紙媒体の診療記録においては，そのような捕捉を効率的に実施することは困難であった．現状の HIS は IT システムであるにも関わらず，garbage-in, garbage-out と揶揄されているが如く，このような連関を実装した HIS は稀有である．そのうえ，いわゆる EBM に資すべき evidence 発見研究においても同様であった．

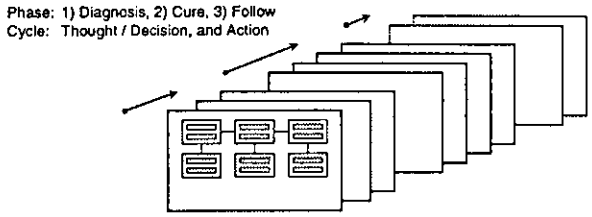
いずれにせよ，前述の短絡性に配慮しながら，診療行為履歴の構成を考案することとする．

C．9．2　診療単位と連関支援構造

C.9.2.1　診療セッション

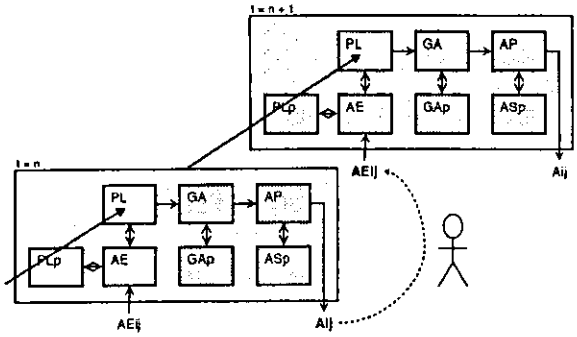
初診から転帰に至るまでの一般的な診療過程は，先ず三つの相（フェーズ）に弁別できる：診断相，加療相，継随相．

各相では何回かの診察と加療が行われる．その際，（B.2）に示した思考と加療のサイクルが1回まわることになる．これを診療セッションと呼ぶことにする．



診療セッションの連なりから診療フェーズが構成される際，各々は緊密に関連しているのは当然だが，診療セッションつまり思考と加療の回転の軸は PL となっている．

すなわち診療過程とは PL を回転軸とした螺旋構造を成している（診療スパイラル）のである．

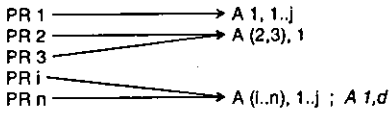


そして AE：Assessment and Evaluation（評価）と GA：Goal（目標）が補足的に機能しており，診療の方向性，あるいはその修正，そして診療停止の条件などが設定されている（はずである）．

このような全体の流れにおいて（C.9.1.2）に示した短絡を構成することになる．

C.9.2.2 診療ブロックとエイリアス

さて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.2) 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．

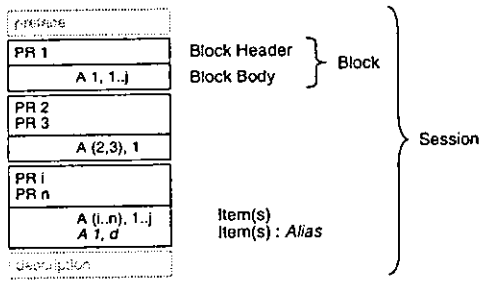
しかし本研究の範囲内での設計実装は短絡を前提しているので，上記の多価関係は，その様な短絡枠組みのなかにおいて，むしろ可及的に精確に記述される必要がある．

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.6.2）に図示したデザインと同値となる．

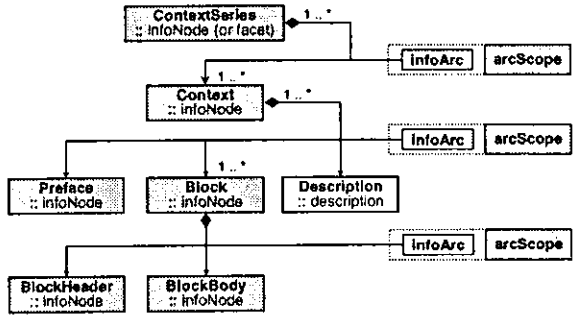
なお A ij には，子を持って機能すべき情報塊（＝括り）が存在することを許容する．例えばオーダ塊がこれに相当する．括り情報塊は "container" infoNode として扱うことになる．

C．9．3 クラス化

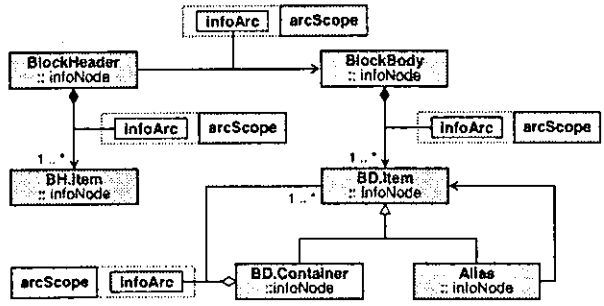
上記の結果を基にしつつ CSX model から必要な Class を生成していくと，BlockHeader および BlockBody については次図の如くなった．

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

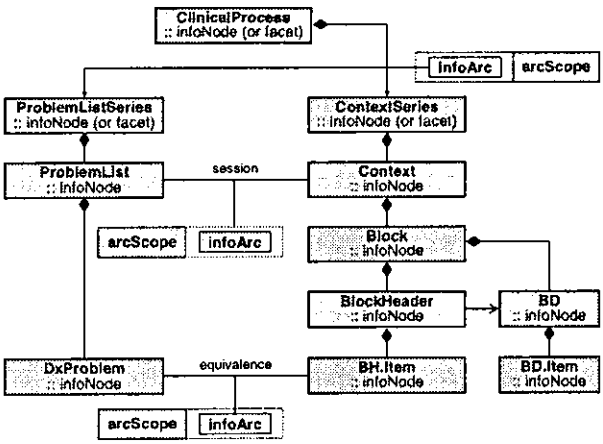
ここで，この関係（BH:->BD）の意義は，他のそれとは異なっていることに御留意願いたい．すなわち，この関係は特定の加療行為の事由や根拠等を示しているのである．

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．9．4 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．9．5 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.9.5.1 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.9.5.2 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．

よって少なくとも今年度においては，修辞関係による表現は避けるよう努めた．

C.9.5.3 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．10 Unique Identifierの設計

本来 @uid は，特定のアプリケーションまたはシステムが（同時に）対象としている instance 群において一意でありさえすれば任意である．

しかし（B.10）の前提のもと，参照実装を開発する際には，xml document の視認は避けようもない．よって human readability を向上させるために，infoNode，arcScope，ならびに root element 直下に存在する description の uid は以下のように設計した．

つまり，各 object の種別を表現する 2 or 3 letters code と，それが出現する順序番とを組合せて附番するスキーマである．

なお業務システム実装では，本来，システム内において一意性を保証しうる番号を活用するべきは当然である．

C．10．1 infoNode, description

ある infoNode が初めて生成される xml ファイル内における当該 infoNode が出現する位置を，順次表現するために以下を用いた：

```
PL:   ProblemList
PR:   Diagnosis or Problem
AS:   Assessment
GA:   Goal〔略語：GL も許容〕
AP:   (Action) Plan〔略語：PN も許容〕
---
CTX:  Context
P:    Preface
B:    Block
HD:   BlockHeader
BD:   BlockBody
CTN:  Container
ITM:  Item
D:    Directory
---
DS:   Description
PPT:  Participant
---
##:   sequence number
```

なお病名プロブレム変遷では PL.0 が先頭で，加療行為経過では CTX.0 を先頭とする．

連番は，PL および CTX では，ProblemListSeries または ContextSeries における出現順とし，其々のファイル内においては，同項目の出現順とする．ただし，親項目を超えた連番とはならないこととした．

C．10．2 arcScope

各 arcScope の uid 値には prefix を用いることとし，これに引き続く文字列は，その arcScope において，親，場，あるいは類する意義を持つ infoNode の uid 値とすることを原則とした：

```
RL:   Scope of Relation
```

同時に，関係する infoNode の @bearing には，将にその arcScope@uid の値が必要なだけ挿入されることとした．

この機構により xml の判読性は格段に向上した．なお，前者は一般的な関係に用い，後者は変遷のみに用いることとした．

以下の書式は変遷を記述する際に用いた．前者は ProblemList の変遷，後者は DxProblem の変遷を司ることとした：

```
TRS:  TRS.PL.##.**
      TRS.PR.##.**.@@
```

上述において##と**はプロブレムリスト番号を表す．また@@は変遷関係の連番である．その直列化例は（C.11.9）に記してある．

以下の書式は，病名変遷 xml と加療履歴 xml とを結合する際に用いた．前者は session 結合，後者は DxProblem と BlockHeaderItem との同値性を宣言するために用いた：

```
Lk:   Lk.##.PL.CTX
      Lk.##.PR.BH.**.@@
```

上述において##は診療セッション番号を表す．また**は Block 番号を，@@は BlockHeaderItem 番号を表すこととした（C.11.10）．

C．10．3 その他

以下は準備したが実際には使用していない：

```
C1F:  Chart (first form)
C2F:  Chart (second form)
STG:  Stage
ITV:  Intervention
```

C．11 構成要素の直列化

情報モデル，およびその直列化における要素間包含関係については（C.6）（C.7.5）を御参照願いたい．なおdataTypeはW3C XML Schemaのデータ型に依拠している．

C．11．1 病名

前年度から引き継いだ課題である（B.14.2）の(1)の事項は，（C.4）の（3）で解決済みゆえ，これに基づいて以下を記す：

```
<csx:nodeCode
 Cascade="JPMHLW-Dx"
 Aspersion="2.2"
 code="EH0K+H280"
 codename="1型糖尿病性白内障"
 priority="1" />

<csx:nodeCode
 Cascade="ICD10"
 Aspersion="10"
 code="E103"
 codename="1型糖尿病・眼合併症あり"/>

<csx:nodeCode
 Cascade="ICD10"
 Aspersion="10"
 code="H280"
 codename="他に分類される疾患での白内障
 ほか・糖尿病性"/>
```

また「詳細専門情報語の修飾付加機能（専門的な病期や重症度，介入状況）」の要求については，今年度の試作アプリケーションにおいては(C.8.3)で説明した簡易な手法で都合することとした：

```
<csx:construe
 NSname="重傷度/評価型"
 NScode="Stage"
 NSver="CSX0.96.9">[要素値]< csx:construe />
<csx:construe
 NSname="処置/介入"
 NScode="Intervention"
 NSver="CSX0.96.9">[要素値]< csx:construe />
```

繰り返すがStageやInterventionに関してのこの様なモデリングと直列化は，CSXにおける正規ではない．

その他のクラスattributeについては，以下の通りである：

```
<csx:dimension
 tude="activity"
 unit="anonymous"
 equivalent="Equal"
 measure="active"
 dataType="string" />

<csx:dimension
 tude="priority"
 unit="anonymous"
 equivalent="Equal"
 measure="1"
 dataType="string" />

<csx:dimension
 tude="Basso"
 unit="anonymous"
 equivalent="Equal"
 measure="continuo"
 dataType="string" />

<csx:dimension
 tude="dateTime"
 unit="anonymous"
 equivalent="Equal"
 measure="2005/02/08 11:56:41"
 dataType="datetime" />

<csx:construe
 NSname="重傷度/評価型"
 NScode="Stage"
 NSver="CSX0.96.9"> [内容] </csx:construe>
<csx:construe
 NSname="処置/介入"
 NScode="Intervention"
 NSver="CSX0.96.9"> [内容] </csx:construe>
```

なおBassoについては後述もしくは分担研究報告書（植田）にて記す．

C．11．2 プロブレムリスト

まずプロブレムリスト全体のinfoNodeを作る：

```
<csx:infoNode
 uid="PL.0"
 category="Problem"
 kind="ProblemList"
 bearing="RL.PL.0 TRS.PL.0.1">

<csx:nodeCode
 CScode="PARCEL"
 CSver="0.96.9"
 codeName="ProblemList"
 code="ProblemList" />
```