C．7 モデルの有用性評価

臨床家・臨床試験専門家・臨床教育家である分担研究者の観点から検証した結果，思考過程モデル，オントロジ・モデル（CSX model），３Cモデル（役割配役立場モデル）はいずれも重要かつ意義深く有用であることが明らかとなった．

Y yes
S support
P preparing
* depends on system design
- not targeted

項　目　細目	事項内容	思考過程モデル	CSXモデル	3Cモデル
1	**Clinical Practice**			
1.1	**患者における問題点の記述**			
1.1.1	症状	Y	P	-
1.1.2	診察所見	Y	P	-
1.1.3	検査所見	Y	P	-
1.1.4	病態の理解解釈に関する正確な記述	Y	P	-
1.1.5	診断病名の正当性に関する記述	Y	P	-
1.1.6	診断病名	Y	Y	-
1.1.7	ゴール設定や介入を制御する主病態（Basso Continuo）の記述	Y	Y	-
1.2	**問題点の変遷とその要因の記述**	Y	Y	-
1.3	**問題点解決に関する記述**			
1.3.1	ゴール設定（例：高血圧患者における脳卒中の予防）	Y	Y	-
1.3.2	代替（あるいは近位）ゴール設定（例：高血圧患者における適切な目標血圧値）	Y	Y	-
1.3.3	代替（あるいは近位）ゴールの妥当性を Basso Continuo による制御の視点からの記述	Y	Y	-
1.3.4	代替（あるいは近位）ゴールの妥当性を 真のゴールとの連関において記述	Y	S	-
1.3.5	ゴールへ至るためのパスの記述（診断計画，治療計画，予防計画など）	Y	S	-
1.3.5+	その妥当性について Basso Continuo による制御の視点からの記述	Y	Y	-
1.3.6	ゴールへのパスの実現性に影響する因子のうち純粋に医学医療学的な事項ではない事項の記述（家庭環境，社会的情勢，患者コンプライアンスなど）	Y	S	-
1.3.7	問題点解決のために実際に行われた医療行為と問題点の連関	Y	Y	-
1.3.7+	当初計画された医療行為と差異がある場合はその要因も含めて	Y	Y	-
1.3.7++	Basso Continuo による制御の視点からの記述	Y	Y	-
1.3.8	実際に行われた医療行為の根拠の記載	Y	Y	-
1.3.8+	Basso Continuo による制御の視点からの記述	Y	Y	-

1.4		**アウトカムに関する記述**			
	1.4.1	得られた（または発生した）アウトカムまたはイベントの記述	Y	Y	-
	1.4.1+	あらかじめ“All or Nothing”アウトカムと設定された近位ゴールと 代替エンドポイントからのアウトカムの併記	Y	Y	-
	1.4.2	設定したゴールとの比較	Y	S	-
	1.4.3	設定したゴールと実際に得られたアウトカムの差異の要因	Y	S	-
1.5		**医療スタッフに関する要件**			
	1.5.1	適切なアクセスを管理できるシステム	-	-	P
	1.5.2	診療スタッフひとりひとりの医療行為決断の根拠の記述	-	-	P
	1.5.3	ジュニアスタッフの決断を適切に制限する枠組．ただしプロポーザルは可能	-	-	P
1.6		**診療情報の蓄積と情報**			
	1.6.1	問題の変遷とのリンク（文脈説明：reasoning)	Y	Y	-
	1.6.2	意義ある解析が可能な診療経過記録の蓄積	-	Y	-
	1.6.3	診療経過記録の追跡と抽出	-	Y	-
	1.6.4	Basso Continuo を軸とした診療スパイラルの追跡と抽出	-	Y	-
	1.6.5	アウトプットの記録と更新	-	Y	S
	1.6.6	記述された情報の一般化	-	P	-
1.7		**地域医療への貢献**			
	1.7.1	地域医療ネットワークの形成	-	*	P
2		**Clinical Research**			
2.1		**前向き（無作為化）介入試験時の診療記録として**			
	2.1.1	説明と同意取得に関する記述	-	-	S
	2.1.2	患者情報に関するプライバシーセキュリティーおよびコンフィデンシャリティ	-	-	Y
	2.1.3	適合基準，除外基準の遵守			-
	2.1.4	無作為に割り付けられた治療の記載			-
	2.1.5	プロトコール治療中止基準の明確な記載			-
	2.1.6	一次，二次エンドポイント，現実の診療行為としてのゴール設定など（降圧薬の臨床試験であればエンドポイントは脳卒中や心筋梗塞，ゴール設定は血圧降下）	Y	P	-
	2.1.7	割り付け治療以外の治療（医療行為）と根拠，問題点とその変遷	Y	Y	-
	2.1.8	割り付け治療以外の治療（医療行為）の質の評価	Y	S	-
	2.1.9	脱落・重篤有害事象の記述	Y	Y	-
	2.1.10	実際のアウトカムの記載	Y	Y	-
2.2		**後ろ向き臨床研究の実現**			-
	2.2.1	アウトカムから後ろ向きに要因を同定可能なモデル構築	Y	Y	-

	2.2.2	医療行為の質の評価 （以下の質の評価とアウトカムの関連を解析できるモデルに基づいたデータ構造）	Y	S	-
		問題点の同定が精確に行われたことを検証きること	Y	S	-
		問題点変遷に関する記述および変遷の妥当性を検証できること	Y	S	-
		ゴール設定は適切に行われたことを検証できること	Y	S	-
		近位ゴール設定が適切に行われたことを検証できること	Y	S	-
		問題点解決のための妥当な診療計画がたてられ，その根拠は明確かつ適切であったことを検証できること	Y	S	-
		実際の診療行為の根拠は明確かつ適切であったことを検証できること	Y	S	-
	2.2.3	真の症例対照研究の実施? 品質評価を含有しプロブレム/介入/ゴール/アウトカムの関連を解析する解析モデルの構築	S	S	-
3		**Clinical Education**			
3.1	3.1.1	ジュニアスタッフによるラウンド時の問診内容，診察所見，検査所見の記述	Y	P	P
	3.1.2	そのティーチングスタッフによるレビュー	Y	Y	P
3.2	3.2.1	ジュニアスタッフによる外来診察時の問診内容，診察所見，検査所見の記述	Y	P	P
	3.2.2	そのティーチングスタッフによるレビュー	Y	Y	P
3.3	3.3.1	ジュニアスタッフによる入院患者，外来患者のゴール設定	Y	Y	P
	3.3.2	そのティーチングスタッフによるレビュー	Y	Y	P
3.4	3.4.1	ジュニアスタッフによる入院患者，外来患者の治療計画の提案	Y	Y	P
	3.4.2	そのティーチングスタッフによるレビュー	Y	Y	P
3.5	3.5.1	ジュニアスタッフによる入院患者，外来患者のアウトカム記述	Y	Y	P
	3.5.2	そのティーチングスタッフによるレビュー	Y	Y	P
3.6	3.6.1	ジュニアスタッフとティーチングスタッフ間の診療に関する思考過程（ゴール設定や根拠の提示，治療計画を含む）の比較	Y	P	-

C．8　グラフ構造表示の設計

C．8．1　必要性

CSX model の応用可能性は，単に病名やプロブレム変遷表現のみに限らず，(A)や(E)に挙げた重要事項に関わる各種情報を表現する際には極めて有効な手法となる．

その際，多重グラフ構造および共起性の制約や許容に関わる表現は必須となるが，Tree pane では，これらを実現できない．

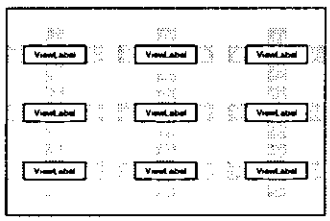
また domain や subdomain を超えて関係が形成される場合がある．ということは「結合の意義とその可否」についても，多様とならざるをえない．よって唯一つの“anchor”属性によって結合可能性が規定されてしまう画面環境とは，本質的に脆弱と云わざるをえないことになる．

これらの点を解決もしくは回避するためには Graph pane が必須となる．

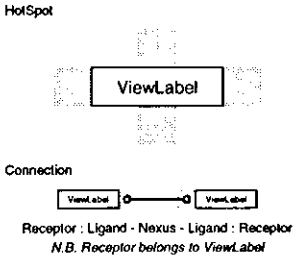
C．8．2　Graph pane

Graph pane は病名/プロブレム変遷のみならず，グラフ構造上にある種々の事物要素とそれらの関係を GUI 上に表示するために考案した Microsoft Windows .NET Framework 2003 上の画面コントロールである．

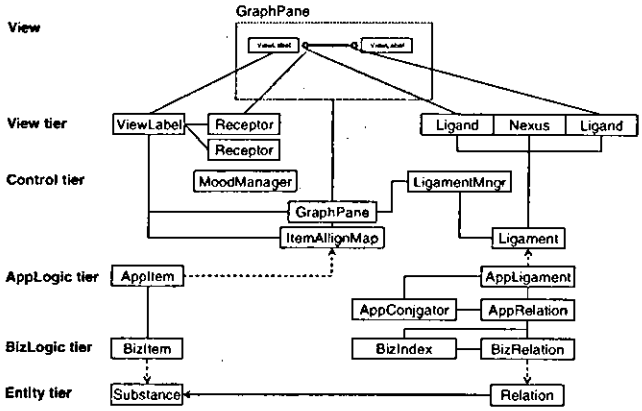
表示項目は pane の内側に用意してこれに表示する仕様とした．Graph pane は関係線のみならず表示項目もその支配下において管理することとなる．表示項目は infoNode のみならず，その子要素や属性をも扱えるとした．



さらに共起や domain 間結合の選択性をも実現可能とするため，関係線と結合とを表す Nexus と Ligand に加えて，infoNode の結合選択性を示す Receptor を用意することとした．



なお Ligand や Nexus は形状や lock や unlock などの諸属性を有しており，programmable あるいは end-user editable とした．



以下に Receptor と Ligand の振舞いとその付帯事項を述べるが，それをサポートするための各種クラスを上図に掲げた．

結合と表示

- 表示項目は ViewLabel に格納され表示される．
- ViewLabel は infoNode[@category @kind]ほか domain における business logic に応じて 0..* の結合選択性（mood）を持つ．
- ViewLabel は当該項目の mood に応じた 0..* の Receptor を持つ．ただし Receptor は GUI には表示されない．
- ViewLabel は上下左右に hotSpot を有する．
- hotSpot に対する GUI operation によって，Receptor が感応する．同時に，Ligand が生成されるか/または既存の対側 Ligand も感応する．
- Receptor は Ligand に対して選択性を有する．より正確には Ligand を介して対側 Receptor の mood に対する選択性を有する．
- hotSpot における GUI 操作によって Ligand と Receptor が通信し，Receptor は Ligand へ mood，connectivity そして ID を渡す．
- Ligand は互いに同側 Receptor の mood を対側 Ligand に伝え，対側 Ligand から伝達された mood と同側 Receptor の mood とを比較して結合可否を判別する．
- 両端 Receptor 間に mood 適合性がある場合のみ Ligand - Nexus - Ligand が survive し，mood 適合性の無い場合には suicide する．なおこの三つ組みを Ligament と呼ぶこととする．
- 生き残った Ligand は hotSpot に配置される．
- 結合と関係線の表示は Ligament によって実現される．

つまり Receptor と Ligand は結合されるものの Receptor 同士は直接的には結合されず，対側 Receptor への結合選択性を示すのみである．

表示管理系

この機能を下支えするために以下を用意する：

・Ligament を統括する LigamentManager
・ViewLabel と LigamentManager とを統括する GraphPane
・ViewLabel での Receptor の発生等を管理する MoodManager

C．8．3 辺と端点と頂点

生成編集された Ligament（Ligand - Nexus - Ligand）情報を Control tier から Entity tier へと伝達する際の留意について述べる．

一つに arcScope[@category and @kind]および infoArc[@category and @kind]を決定する必要がある．

加えて infoArc 数と，Nexus 数・Ligand 数との間の差異を吸収管理する必要がある．この不一致性は，画面構成クラスの編成と・XML Schema 設計との間の，視点の不整合から生じている．つまり頂点管理か，あるいは辺または辺端点の管理か，という差異に拠っている．

これらの整合と翻訳には arcScope の存在が必要条件となる．

arcScope ならびに arcScope[@category]と arcScope[@kind]は，アプリケーションが提供する『場』によって生成・決定される．また，このとき infoArc[@category]も同時に決定されることになる．

というのも GUI operator が「ナニを編集する」という『場』は，将にアプリケーション自身，言い換えれば AppLogic tier に位置する処理群が提供している，からである．

しかし十分条件を満たすには他の管理情報も必要となる．その一つは関係付けられる二つの infoNode の・category と kind で決定される domain や subdomain の同異であり，いま一つは，画面製作者が意図した「GUI 配置における意味・意義」である．

この十分条件によって，Receptor に結合された Ligament は，その意義に応じた arcScope の管理下に置かれて infoArc とされる．また，一つの arcScope 内の infoArc には唯一つの「親」infoArc が存在しうるという整理，換言すれば全ての infoArc[@kind]の決定が為されうるのである．

なお Ligand - Nexus - Ligand と infoArc との mapping を管理するために，AppLogic tier に AppLigament と AppConjugator とを用意した．そして AppConjugator には replica が置かれ，相応と整合に貢献することとした．

AppLogic tier における RelationManager 等は，したがって，以下の二つの役割を担う：

・arcScope と infoArc の管理
・Ligament と BizRelation との間の整合と翻訳

なお，関係の意義付けについて一時的な混沌を許すような『場』を提供する場合には，その意義を GUI operator に尋ねるしかなかろう．このような事例は，変遷や連関をモデル化するツールや，archetype を作成するツールにおいて発生しうる．

通常業務システムでは，arcScope [@category]，arcScope [@kind]，infoArc [@category]は画面製作者の管理下にあって自明であり，infoArc [@kind]は infoNode [@category and @kind]と画面製作者が意図する「GUI 配置における意味・意義」によって決定されうる．

C．9 アーキテクチュアと基幹クラス

前述（C.8）に関わる実装設計ならびに実装は山田が担当した．

C．9．1 論理層

Graph pane が求めるクラス構造は複雑化ゆえ，論理アーキテクチュアを背景したクラス構築とそれらの配置，そして必要なイベントなどを設計した．

なお各論理層（tier）間の相互干渉を回避するため，可及的にイベントを介しての通信またはインタフェイスとして扱える Collection を介してのアクセスとするよう設計した．

論理層は以下の如く5層に分割した：

- View tier
- Control tier
- Application Logic tier
- Business Logic tier
- Entity tier

このデザイン・パタンを採用したのは，変更や拡張を局所化するとともに，下層については再利用を狙ったからである．ただ，このデザイン・パタン自体が有する複雑度の増加，効率の低下，そして伝播性の低下という側面も併せ持つこととなる．

なお論理アーキテクチュアの設計は，本研究の本質に関わるクラスのみに留めている．

View tier

この層はいわゆる view であり Control tier に管理されつつ view 生成する．また GUI 操作の受け取り口でもある．

ViewLabel とその hotSpot ならびに Receptor，そして Ligament (Ligand - Nexus - Ligand) はここに配される．それらの振る舞い等は前述した通りである．したがって基礎的・原則的な結合可否については，この tier 内の messaging のみで完結することになる．

Control tier

この層は View tier とともに boundary もしくは presentation を担う．

LigamentManager，GraphPane，MoodManager が配される．それらの責務の概要は前述した通りである．なお，これより下層の tier との間で状態伝播に関わる役割も果たすこととなる．

Application Logic tier

この層は，実装アプリケーションが提供すべき「場」を構成する責務を担う：

- Solution における限定的な特定の業務
- Data provider のクライアント
- Form と GUI control のクライアント

この層と Business Logic tier との分離により，Business Logic tier を他のアプリケーションで再利用できる可能性を高めることを目的としている．或いは，boundary が変更された場合でも，「場」の形成におけるロジックを再利用できることを目論んだものである．

この層には AppItem，ItemAllignMap，そして AppLigament，AppConjugator，AppRelation，RelationManager が配される．

AppItem は BizItem から生成されるが必ずしも Substance とは限らず，その子エレメントほかアトリビュートであることも可，としている．なお ItemAllignMap は，ViewLabel と AppItem とのマッピングを管理している．

AppLigament は Ligament と相応しているが，RelationManager によって boundary と entity との間の差異が緩衝になっている．GUI 操作で関係が生成変更された場合，AppConjugator や AppRelation の管理，それらのアトリビュート値の管理決定等は RelationManager が担う．

AppItem と AppLigament の編集状況は，必要な場合には，Business Logic tier に伝えられ，対象 domain における business logic の検証を受けることとなる．

Business Logic tier

この層は，対象領域の記述に要する entity を生成するとともに，諸関係について business logic や domain semantics への適合性の検証を実施することを責務とする．さらに将来的には RuleBase や KnowledgeBase にアクセスして censing engine を動作させることを想定している．

したがって上位層へのインタフェイスとなる Collection を生成する際には，事物要素とその諸関係の形式的意味的な整合確認をしたうえで，これを実施することとなる．また前述したとおり，boundary 独立であることを目標としている．

この層には BizItem, BizRelation, BizIndex が配される．なお BizItem とは，実際のところ infoNode の Collection である．

BizIndex とは，多次元空間における infoNode 間の直接的または直近の上下前後左右などの Topology を纏めた要約 Collection である．

Entity tier

この層は，Storage に対するアクセスや対象 domain にフレームワーク（本研究では CSX model）を適用するための課題を担う．今回の実装では CSX model に則ったファイルそのものをソースとしている．

Entity tier の構造は CSX model が下支えしていることになる．

event と interface

各 tier で発生する event は，BCE での各層内に留まるものと層間を超えるものとが区別されている．

なお原則として event 名に .FIX とある event は，上位層クラスへの event 発行としている．また boundary では，event を受信するクラスは，その event によって状態遷移するクラスでなく，当該クラスを管理する Manager クラスが受信することがある．

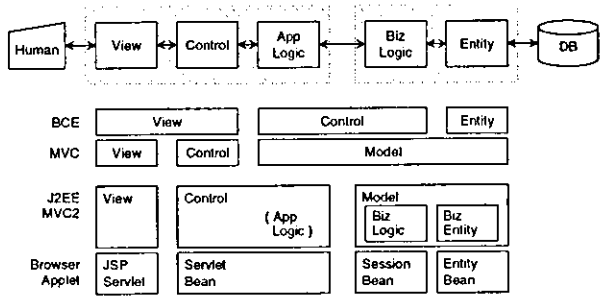
インタフェイスの役割を果たすクラスは，AppItem（と ItemAllignMap）と AppLigament，また BizItem, BizRelation, BizIndex である．

結合強度の表現

現時点では結合強度を格納する属性は，何れのクラスにも用意していない．これがどのような微少構造形式によって具現化されるべきかは論議と検討を要するものの Ligand や Nexus においても表現可能ではある．

C．9．2 アーキテクチュアの比較

上記の5階層モデルと他のデザイン・パタンとを比較してみる．

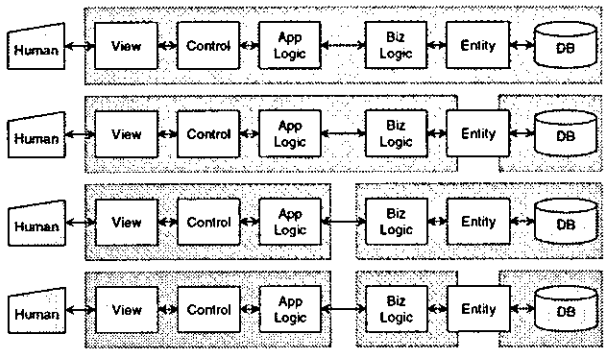


層分割視点は以下の三点に拠っていることが理解できる：

- 想定する deployment の差異，もしくは DB server や Application server に関する2階層または3階層モデルに対する認識
- BizLogic tier の重視の程度
- Client application の製造コストの意識と AppLogic tier の配置

まず三点目については，我々としては，5階層モデルでの Control tier と AppLogic tier の分離は妥当ではないであろう，と考えている．

そのうえで deployment 候補を挙げると下図のようになる．



上段は Stand-alone，他は C/S モデルであり，うち最下段は3階層アーキテクチュアである．下二つはモジュール間 protocol の規定を要求することになる．最下段は，サーバの分散性が向上することになる．

どの deployment を選択するべきかについては実装内容と実装環境に依存するので一概に是非を謂うことはできない．試作アプリケーションは参照実装と位置づけている．したがって5階層モデルの思想を尊重踏襲し，どのようなデザイン・パタンにも対応可能とすることとした．

C．9．3 課題の整理

初年度実装後の課題は以下の四点であった：

A) Task 制御と Initiator
B) Tier 境界を超える event の処理
C) MS Windows 開発環境の要請（画面ありき）
D) MS Windows 開発環境の要請（namespace）

それぞれについて順次，要点と解を述べるが，それらは相互に関連している．

A) Task 制御 および Initiator

全てのプログラムは起動者を要し，また各 tier の全 Class も同様である．

そしてInitiatorは，プログラムを起動させたときに最初に動作し，かつ，各Tierの全Classをすべてinstance化しておかねばならない．

というのも，tier間に不必要な親子関係を持ち込まない，つまり各tierのClassの独立性を保つため，である．

B) Tier境界を超えるeventの処理

各tierのサービスは関数呼出（Method呼出）によって実現されており，これはevent処理も本質的には変らない．

したがってevent受信側やサービス享受側も，必ず，送信側やサービス提供側の情報を持っておく必要がある．しかもこれは，コード記述における単純な形式上の問題ではない．

そこで各tierのClassの「纏まり単位」ごとにClassのInitiatorを準備する必要がある．

各Tier内のClassは一般に，互いに連繋してサービスを提供している．このような「まとまり」に親に相当するClassがある場合，それに担当させる．親の役割の担うClassが無い場合には，「まとまり」に対するManagerを用意して，それに担当させることとする．

C) MS Windows開発環境の要請（画面ありき）

Windowsアプリケーション開発環境では，プログラムに指示を与えることのできる「画面」が『親』の地位を占めることになる．

一方，本研究のTier分割では，View tierのみ画面を持っているが，しかしView tierはControl tierの制御下に存在するべき構成である．

この状況はWindowsアプリケーション開発環境の要請とは矛盾する．よってView tierを起動するmoduleが起動する前に，必要な動作環境を全て整えておく必要がある．

D) MS Windows開発環境の要請（namespace）

Namespaceの解決はDLLを経由して為される．つまりUsingで定義するのみでは機能せず，そのnamespaceを持ったmodule（を含むDLL）は，先立ってcompileされている必要がある．

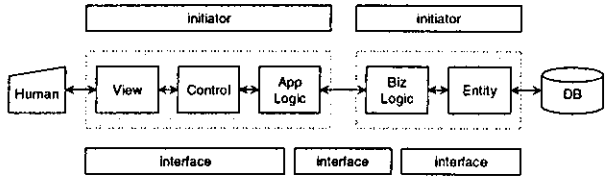
これは『Class間に親子関係を持たせている』ということであって，各tierの独立を妨げる要素となってしまう．これを回避するためには，アプリケーションの最下層にInterfaceを置く必要がある．

すなわち，外部から参照されるClassはInterfaceを継承し，Interfaceに定義されたmethodやpropertyを持たせることとする．

なおClass間メッセージ（EventArg）に関わるClassもClass間で共用となるので，ここに置くこととする．

結論

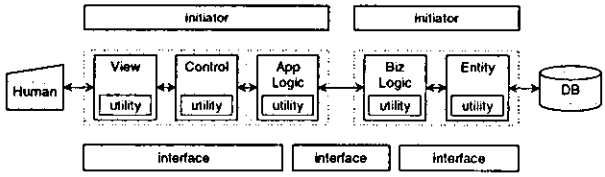
アーキテクチュアは下図の如くした：



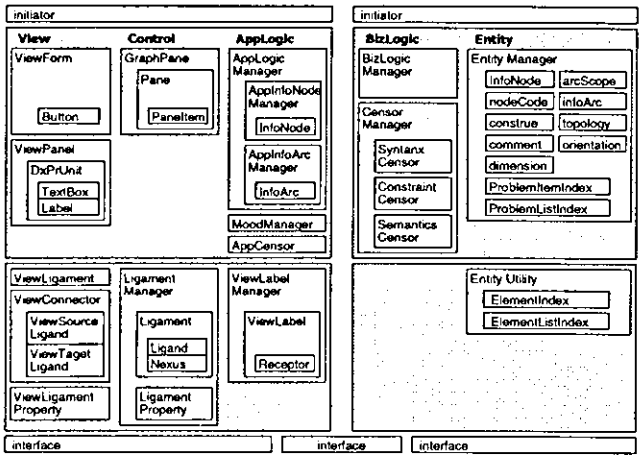
つまり最初に起動されるプログラムが全てのInitiatorに対する責任を負い，かつInterface構造を導入することで，各tierの独立性を保持しつつ伝播と連携とを促進することとした．

C．9．4　基幹実装クラス

さらに各tierにおいて共用性の高いClassはDLLで供給していく構想を踏まえて，Utility Classとして扱うこととした：



そのうえで（C.2.1)(C.8.2）から要請されるClassを以下の如く配置した：



よって参照実装では，この内部Classを基礎として，各アプリケーションを構築していく．

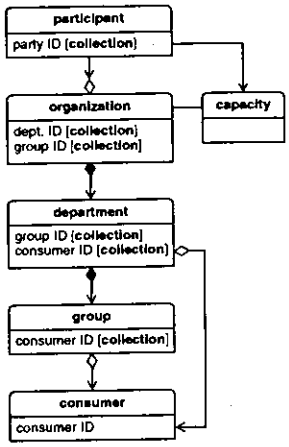
C．１０ 視野範囲限定の実装モデル

実装システムにおける視野範囲限定に関わる戦略と実装は極めて重要である．

C．１０．１ Cascading Focusing

ログインにおける Fleet と患者の選択の際に，役柄配役立場モデルを適用しながら，順送りに視野範囲を制御するようにした．

すなわち（C.6.15）に示した各 Class への ID collection の持たせ方は，集約関係を重視するのではなくて，業務フローにおける参照順序を優先する，という手法で解決している．



C．１０．２ StructCore

MEDIS-DC 病名コード集と手術処置コード集は，その拠り所の一つとして ICD10，あるいは ICD9-CM や「点数表の解釈」という分類体系を持っている．

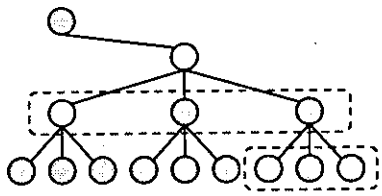
そのようなコード・マスタは，それが依拠する体系の一軸を優先して各コードをツリー表示させるなら，一覧性を保ちつつ視野範囲を的確に限定することが可能となる．

これを実現するための実装 Class である "StructCore" は，初年度に尾藤が作成した．

- 上位階層項目のマスタ ID
- 当該階層項目のマスタ ID
- 階層項目の連番（平板に展開した際の順）
- 階層項目の区分（0：階層，1：項目）
- 階層項目の名称
- 階層項目の深さ（0-n）
- 階層項目の表示順序（その「踊り場」での順）
- 階層項目の直下の子の数
- 上位が指定する直下の階層項目の表示順序
- 深さ-マスタ ID-表示順序

なお最後の二つは，複合インデックスである．

このエンティティは全体として，個々の項目は自らとその親は何者であって "世界" の何処に居り，他者との前後左右の関係は如何なる状況となっているのか，を端的に表現している．



この Class をコード・マスタ毎に付随テーブルとして Caché に格納することで，ツリー表示やリスト表示を高速に実現することができた．

C．１０．３ EntryMenu

その一方で，MEDIS-DC 検査コード集，あるいは薬品コード集は，一軸で規定されるツリー上に項目を並べあげても実用には即さない．

このことは，分類学（systematic）上の軸なる概念は，必ずしも実用に即すわけではないことを示唆している．いずれにせよ，現場に即したメニュそしてメニュ展開が必須となろう．

然るにシステム実装，特に human interface の構成の際に規範となるような階層情報または『項目配置』における参考情報などは，今回，入手することはできなかった．

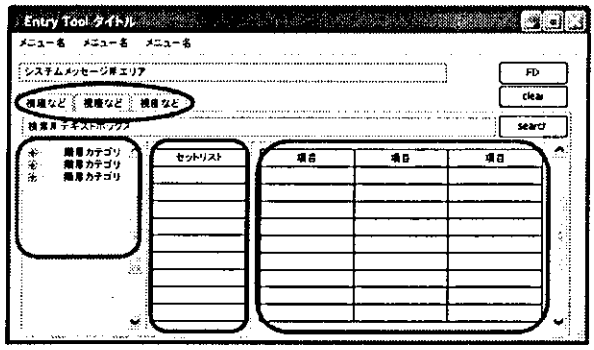
Reference はいわゆる情報モデルのみならず，このようなソースにも与えられたほうが，開発現場には，より実際的であろうと思われる．

ともかくも，なにかしらの一覧表示機構が必要となったので EntryMenu デザインを創ることとした．

C.10.3.1 EntryMenu Design

まずはメニュ構造と展開機構とを，予め想定しておく必要がある．そこで通常用いられている構造と展開を整理しながら一般形と思われるモデルを，将来への発展も見据えつつ構築した．

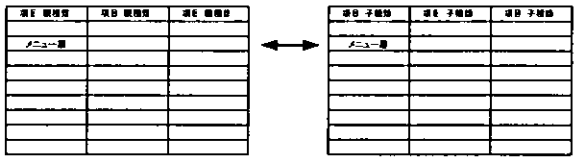
すなわち，Problem oriented または Procedure oriented のような視座が，各人や各診療科部の実情に合致した視野範囲の限定と展開を可能とするようなモデル・デザインである．これを実現する画面例は以下の如くである．



その機能としては次の事項を想定している：

・選択した視座（病名指向や行為指向など）に応じた階層カテゴリの表示
・選択した階層カテゴリに応じたメニュおよびセットリストの展開

このうち前者については将に知識を要求する機能なので実現には準備期間を要するものの不可能ではないし重要ではある．

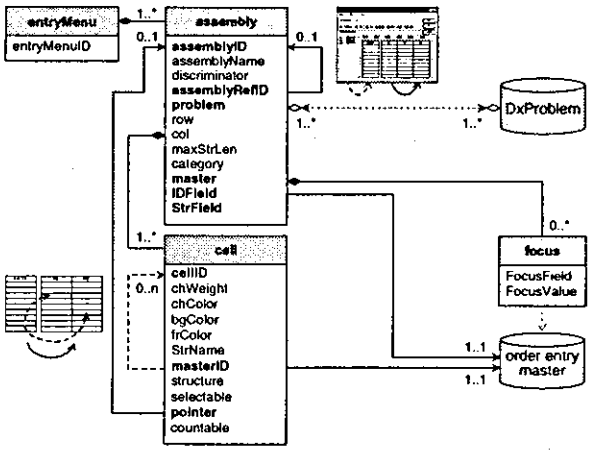


さらに，メニュ項目またはメニュ・タグから，新たなメニュを展開させたり，セット選択にて複数の項目を一括選択したりできる などの，一般的な機能も，当然ながら，想定した．

そしてこれら機能を単に実装するのみならず，(a) 参照実装としてモデル化するとともに，(b) end-user による編集の枠組を構築することに努めた．

C.10.3.2 EntryMenu.class

上記の要件に即して以下のクラス設計した．



クラス assembly は，各『メニュ単位』を表現している．メニュ単位には，通常の選択メニュのほかにセットメニュ（リスト）も含んでいる．セットメニュ（リスト）は，他のメニュ実体を指し示すのでClass diagramとしては自己参照となる．

クラス assembly は対象とするコード・マスタを指し示すと同時に，そのテーブル内の対象とするfieldを指し示す．またメニュの表示上の各種基本属性を規定する（行列の数や幅）．

この条件のもとに，クラス assembly はクラス cell の collection を保持する．

クラス cell は，メニュの表示上の属性を保持する．そして登録可能項目ならコード・マスタ項目を指し示し，セットならエントリ登録可能項目（クラス cell）を指し示し，さらにメニュ展開項目ならクラス assembly を指しすこととした．

なお視座と指向については，そのような知識 ontology が準備されていないため実装の対象外としたが，その実現に要する property は用意しておいた．

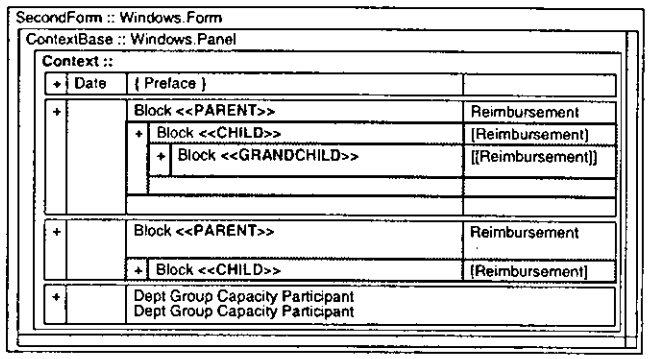
C.10.3.3 EntryMenu.xsd

このクラス設計に即してEntryMenu.xsdを定式化した【総括報告書を参照願いたい】．

C．10．4 Outlined Container

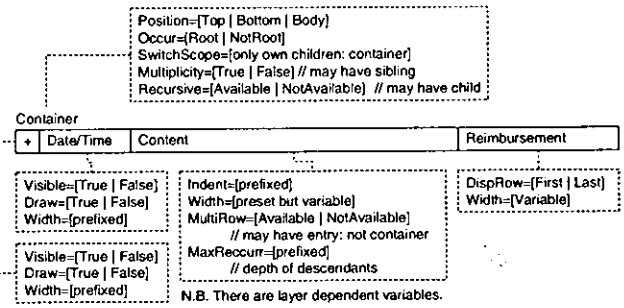
流れや文脈が存在する集積情報において視野範囲を限定する，とは，アウトライン化するということでもある．

よって二号様式画面においてはアウトライン機能を付与した．なお，その際には当然ながら(C.5.3) に記した階層構造にも留意して下図のようにデザインした．



このような階層構造は汎用度の高い module を

再帰的に使用して構成することが可能である．よってその概念モデルを設計した（実装については分担報告書（与那嶺）を参照されたい）．



この実装 Class を container と称する．ただし (C.2.6.4) に記した“container” infoNode とは混同しないよう，御留意願いたい．

とはいえ両者は，参照実装においては相補的に利用され機能している．

なお property に格納される値は“独立”ではなく，むしろ

・クラス container の種別に依存
・クラス container が位置する階層深さに依存

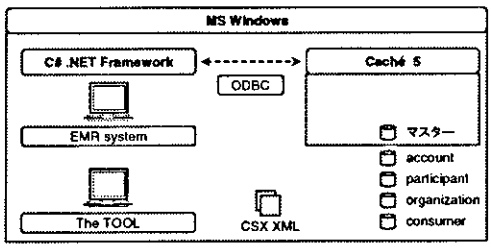
している．あるいは，言い方を変えれば，そのような要素を property して，他の要素は基本骨格として再帰性を促したわけである．

C．11　参照実装の俯瞰

実装アーキテクチュアと基幹クラスは山田が，病名 Composer は尾藤，ほかのクライアント・アプリケーションは与那嶺と大嶺が，サーバの構築は山本（ソリトン）が実装し，設計等は廣瀬と協同して行った．

C．11．1　Stand-alone 構成

Stand-alone 版は，方法（B.13）に記した開発環境で，下図のように構成した．



Stand-alone 版はコード・マスタのみ Caché に格納している．その他の各ファイルを配置したディレクトリ構成は以下の通りである（左手がディレクトリ名，右手がファイル名）：

crt	certification.xml
csm	consumerDir.xml
stf	participantDir.xml
org	organization.xml
dat	PT####_PrDx.xml PT####_PrMa.xml PT####_PMLk.xml
ext	PT####_########_##.jpg

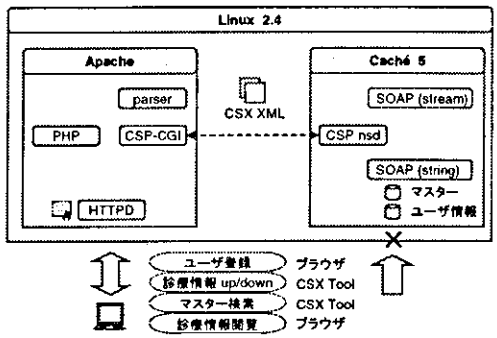
各ファイル名は（C.6.15）に提示した Class と相応している．

敢えて各ディレクトリに分割配置した理由は，今後に C/S 版とした際にも，整理と見通しとを良くしておくためである．

つまり各ファイルは，それぞれ別個のサーバ，もしくは partition に管理される可能性が高く，同時に，それが妥当な管理であろうと思われるからである．

C．11．2　Client/Server 構成

C/S 版のシステム構成は以下のようにした．



すなわち，Caché の直接アクセスは許可せず，CSP-CGI を介して，CSP::SOAP サービスを利用することとした．Stand-alone 版と同様，マスタ検索は Caché のデータベースを検索し，診療経過ファイルは OS 保管することとした．

図中で「ユーザ情報」と記されている内容は，実際には組織情報も含まれている．つまり（C.11.1）で提示した構成のうち crt，csm，stf，org に含まれるファイルを Caché に格納したわけである．

ただし ext は，C/S 版実装では対象外とした．というのもサーバに対して upload されたバイナリファイルを無防備に扱うリスクは，回避すべきだからである．

予めこのように分割しておいたことで，Client 側での改変は入出力「向き」変更と理解することが容易となった．

勿論それだけではなく，そもそも（C.9.4）に示したアーキテクチュアとしておいたことが，改変を容易にした．

そして最終的には（C.11.1）と（C.11.2）との hybrid 版とした．

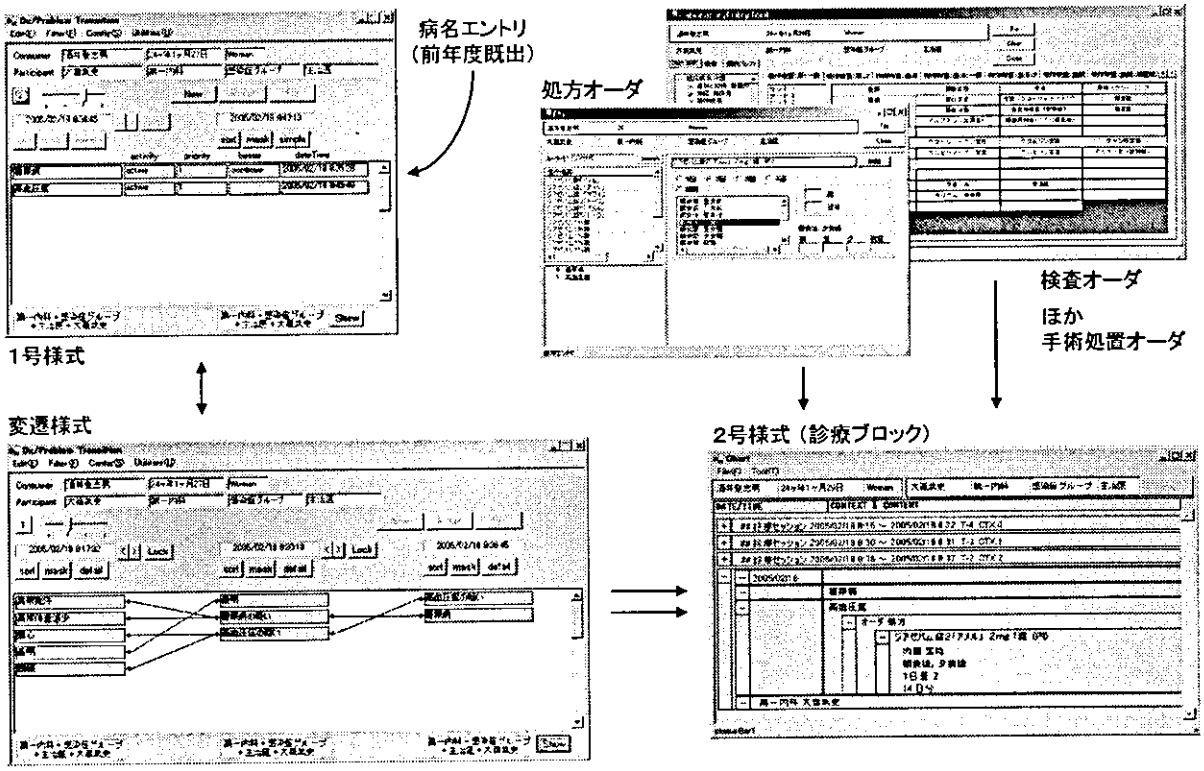
C．11．3 画面と機能の概括

C.11.3.1 診療システム

診療システムは以下のようなイメージであり，またそのような流れにて用いることとした．

1) まず，所属と立場を明らかにしてログインする．

2) 患者一覧から患者を選択すると一号様式画面（Dx/Problem Transition）が展開される．
この画面は，病名やプロブレムの変遷を閲覧編集するためにも用いられる．



3) 必要ならば病名エントリツール（Dx/Problem Composer）で病名を構築して登録する．

4) 一号様式画面での編集を終えたなら，一号様式画面を確定する（FIX ボタン）．

5) 診療対象とする病名を二号様式画面（Chart）に送信して二号様式画面に診療ブロックを形成する．

6) 検査オーダツール（LaboTest EntryTool）や処方オーダツール（Prescription EntryTool），あるいは手術処置エントリツール（Procedure/Operation EntryTool）で必要な項目を選択して，二号様式画面の診療ブロックに送信する．

7) 確定すれば（FIX ボタン），病名変遷ならびに病名診療行為連関が確定し，XML document として保存される．

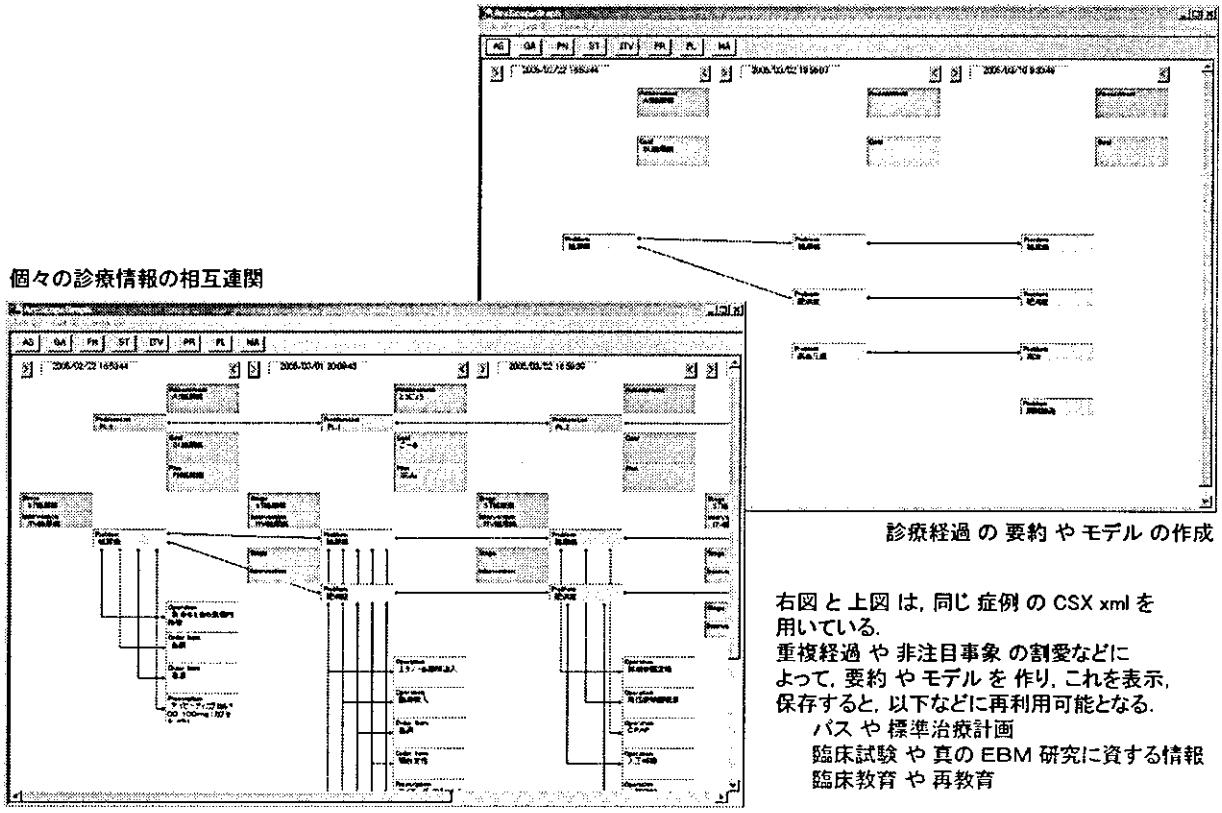
このシステムは Stand-alone 版と C/S 版とを作成した．

C.11.3.2　生成と抽出要約のツール

先の診療システムから出力された診療履歴ファイルを読み込んで，診療過程の要約をビジュアルに作成したり，あるいはCSX model に即した様々な情報塊を作成したりするためのツール TheTOOL である．

なお自然言語を処理するのではなく，意味を担った情報塊 node を探索したり取捨選択したりして，情報塊 node が織り成すグラフ構造に“要約”を施す，つまり臨床医学の観点から重要なノードとそれらの関連を抽出編集するためのツールである．

下図では，左下の前方画面に「診療履歴ファイルを読み込んだ直後の表示例」を，また右上の後方には「抽出と要約後の表示例」を示している．



このアプリケーションはStand-alone版のみである．

C．11．4 機能一覧

C.11.4.1 診療アプリケーション

本研究では以下の参照実装を行った：

Welcome
- ログイン機能
- 所属組織自動絞込機能
- 診療科部選択機能
- 診療グループ選択機能
- 立場選択機能

Patient List
- 選択可能患者一覧機能（自動絞込）
- 患者属性一覧機能
- 所属組織変更機能
- 立場変更機能
- 選択可能患者再検索機能
- 患者選択機能
- 患者情報転送機能（→ Platform）
- 職員情報転送機能（→ Platform）
- アプリ終了機能

Platform
- 患者情報受信機能
- 職員情報受信機能
- 患者属性表示機能
- 職員属性表示機能
- 診療セッション中止機能
- 診療セッション suspend 機能
- 診療セッション resume 機能（自動処理）
- 患者情報転送機能（→Transition/Chart）
- 職員情報転送機能（→Transition/Chart）

Dx/Problem Transition
- 患者情報受信機能
- 職員情報受信機能
- 患者属性表示機能
- 職員属性表示機能
- 一号様式病名欄表示機能
- 新規プロブレムリスト構築機能
  - 病名自動複写機能
  - 変遷自動形成機能
- Dx/Pr Composer 呼出機能
- 構築病名受信機能
- 病名プロブレム一覧機能
- 病名プロブレム変遷一覧機能
- 変遷表示 pane 数切替機能
- 変遷表示スライド機能（連動）
- 変遷表示スライド機能（非連動）
- 変遷表示スライド機能（ロック）
- 病名属性編集機能
  Activity, Priority, Basso
- 病名属性編集機能
- 変遷編集機能
- 変遷属性編集機能
- 評価入力編集機能
- 評価表示機能
- 評価複写機能（隣接表示から）
- 目標入力編集機能
- 目標表示機能
- 目標複写機能（隣接表示から）
- 計画入力編集機能
- 計画表示機能
- 計画複写機能（隣接表示から）
- 病名プロブレムリスト変遷履歴記録機能
- 病名転送機能（→ Chart）
  - 診療ブロック選択機能
  - 診療ブロックヘッダ追加機能

Dx/Problem Composer
- 病名一覧＋選択機能
- 病名修飾語一覧＋選択機能
- 病名修飾語検索機能
- 病名検索機能
- 病名修飾語検索機能
- 病名プロブレム構築機能
- 構築病名転送機能（→ Transition）

Chart
- 患者情報受信機能
- 職員情報受信機能
- 患者属性表示機能
- 職員属性表示機能
- 加療行為履歴一覧機能
- 加療行為履歴アウトライン機能
- 病名受信機能
- 診療ブロック形成機能
- 各 EntryTool 呼出機能
  - 患者情報受信機能（→ Lab/Px）
  - 職員情報受信機能（→ Lab/Px）
- 検査オーダ項目受信機能
- 処方オーダ項目受信機能
- 手術処置エントリ項目受信機能
- 受信項目の削除機能
- 外部ファイル登録機能
- 登録した外部画像ファイルの表示機能
- 外部ファイル登録の削除機能
- 加療行為と病名との関連付け機能
- 加療行為履歴記録機能

LaboTest EntryTool
- 病名一覧機能（ツリー）
- セット表示機能〔選択機能は現状未実装〕
- 検査項目メニュ表示機能

検査項目選択機能
検査項目検索表示選択機能（3軸,AND/OR）
選択項目転送機能（→ Chart）
診療ブロック表示機能
メニュ&セットのユーザ設定機能

Prescription EntryTool
薬品検索＋一覧機能
薬品検索選択機能
服薬指示設定機能
定時：一日量，均等・不均等割付
頓用：一回量
処用時期
処用日数
選択設定項目転送機能（→ Chart）
診療ブロック表示機能
診療ブロック選択機能

Procedure/Operation EntryTool
処置手術項目一覧機能（K/J,ICD9-CM）
処置手術項目検索機能
処置手術項目選択機能
頻用項目ユーザ設定機能

本報告書（C.9）から窺い知れる様に，実装の難易度や参照実装の重要性は，画面数や機能数といった単純な計量値から推し量れるものではない．

むしろ（C.2）の如き極めて抽象度が高い情報モデルを用い，（C.9）の如き論理構成を忠実に実現した，そのこと自体が意義深い．

本研究における試作診療システムの主要画面は9，主要機能は106を数え，一般的な外来診療も模倣できる環境を提供しえた．短期の開発期間と限られた予算で此処までを達成できたことは，システム構築においても何かしら示唆するものが含まれているように思われる．

なお本システムは試作ゆえに，市販電子診療録システムに付随する細かな機能等は割愛しており，本研究の成果を試し評価するに最小限の機能となっている．

C.11.4.2 生成と抽出要約のツール

前述とは別に特殊なツールも開発した．これは，元々は，CSX modelに基づいたinstanceをxmlにて直列化しつつ生成・保存・編集するためのツールであった．加えて，グラフ構造の視覚化機能も付加していた．

そこで，この機能を流用しつつ，前述した診療システムから出力された診療履歴xmlファイルから，診療過程の要約を表示・閲覧・保存する機能を持つツールとした．

TheTOOL
CSX Ontological XMLに基づく
XML instanceの読込機能と保存機能

XmlInstance
XML instanceの表示機能

EDIT
root elementの諸属性の表示
infoNodeの生成と削除
infoNodeの諸属性の生成編集削除
infoArcの生成と削除
infoArcの諸属性の生成編集削除
arcScopeの生成と削除
arcScopeの諸属性の生成編集削除
topologyの生成と削除
topologyの諸属性の生成編集削除
orientationの生成と削除
orientationの諸属性の生成編集削除
dimensionの生成と削除
dimensionの諸属性の生成編集削除
上記編集削除中の相互連関

InfoNodeGraph
infoNodeの子要素の状況表示
infoNodeの子要素のグラフ表示

ArcScopeGraph
病名変遷と診療行為連関のグラフ表示
診療セッション単位の送り（順逆）
変遷関係線の再描画
不要nodeの非表示・再表示
必要nodeの表示ロック
不要nodeの非表示ロック
node探索の設定
包絡するnode種別
探索方向（時間的に順・逆）
隣接するinfoNodeの属性に
依存したnode探索停止条件
node探索開始nodeの設定
node探索
その編集結果の表示と保存

このツールは，本研究班のなかではTheTOOLと読んでいる．

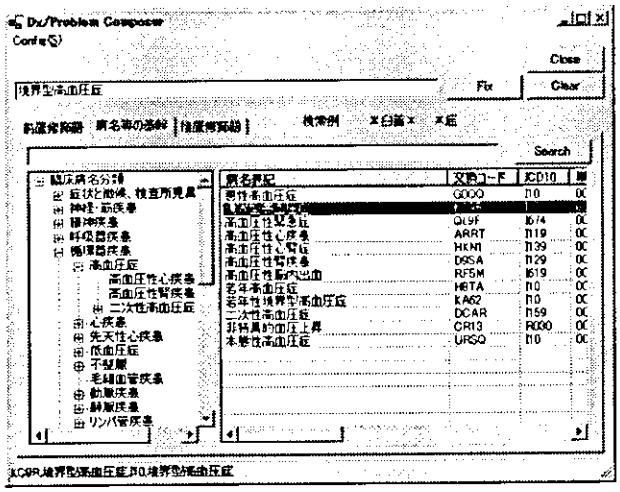
TheTOOLは，臨床にも臨床研究にも臨床教育においても，極めて有用なツールとなりえよう．

C．１２　参照実装（診療プラットフォーム）

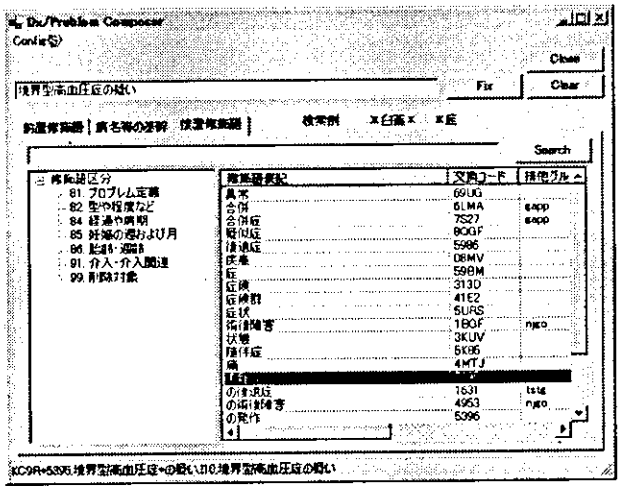
C．１２．１　病名 Composer

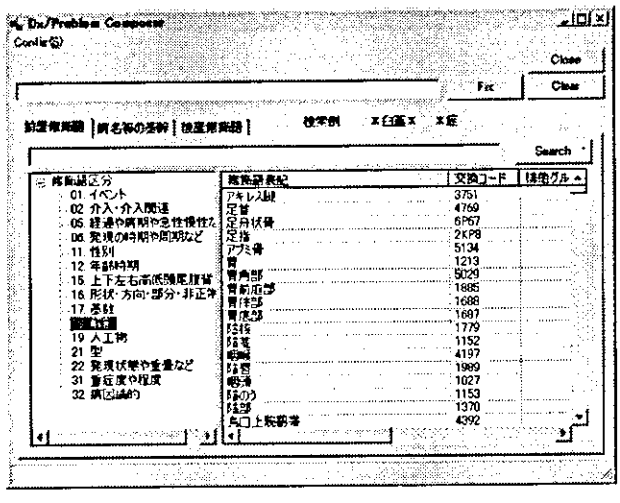
画面例を列挙する（実装は尾藤）．

C.12.1.1　MEDIS-DC 根幹病名



C.12.1.2　MEDIS-DC 病名修飾語

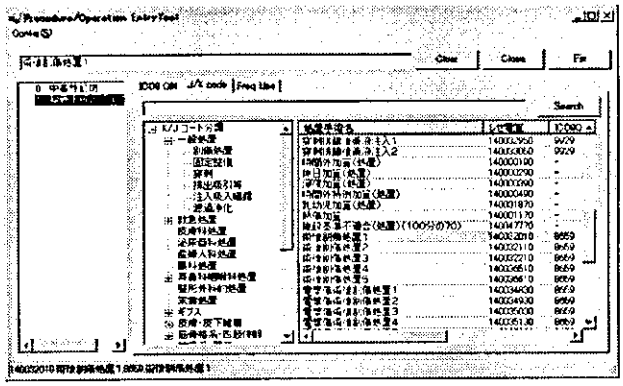




C．１２．２　手術処置 EntryTool

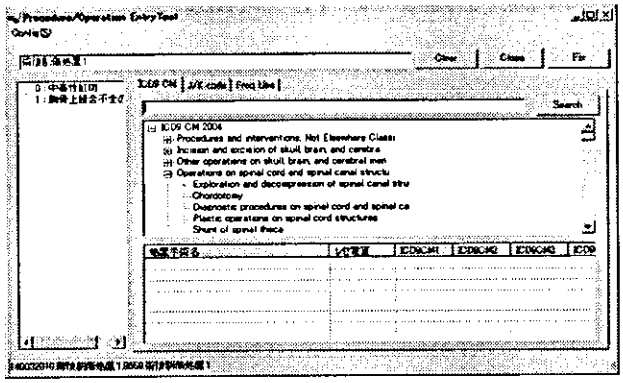
画面例を列挙する（実装は廣瀬）．

C.12.2.1　K/J code



病名 Composer の skeleton を用い(C.10.2)を活用した．

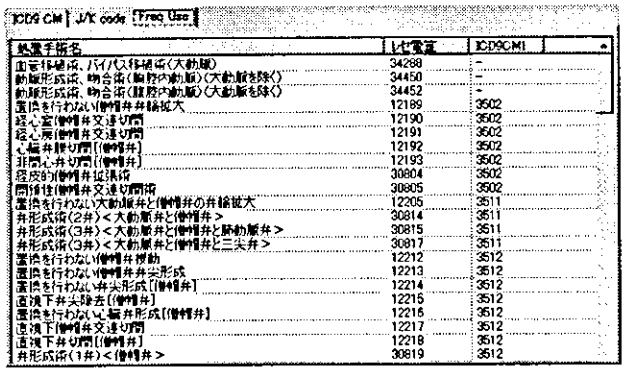
C.12.2.2　ICD9-CM



病名 Composer の skeleton を用い(C.10.2)を活用した．

C.12.2.3　FrequentUse

簡易なユーザ設定ファイルを作り，これを読み込ませて実現した．

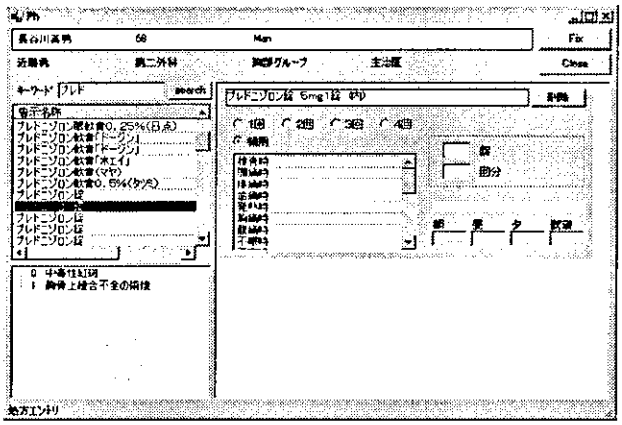


C．12．3　処方 OrderTool

画面例を列挙する（実装は大嶺）．

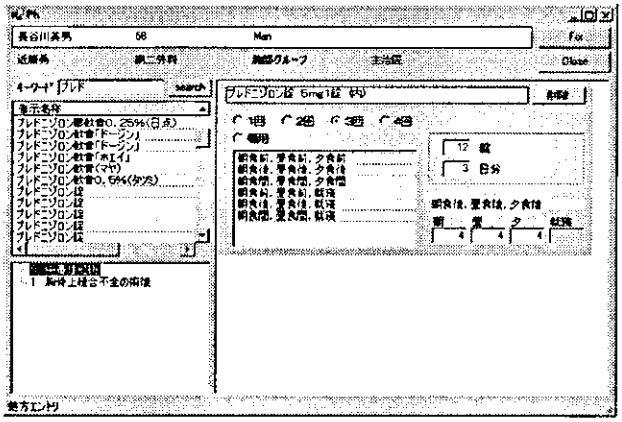
C.12.3.1　Search

画面左手上方で検索し，検索結果をリスト表示させ，その中から選択する．



C.12.3.2　Instruction

画面右手で，定時/頓用を選び，用量等を決定する．



C.12.3.3　Targeting

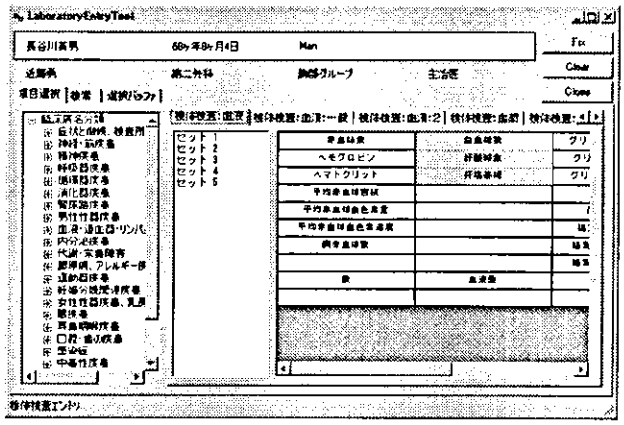
その後，画面左手下方で，この処方を格納するべき診療ブロックを選択する．

C．12．4　検査 OrderTool

画面例を列挙する（実装は大嶺）．

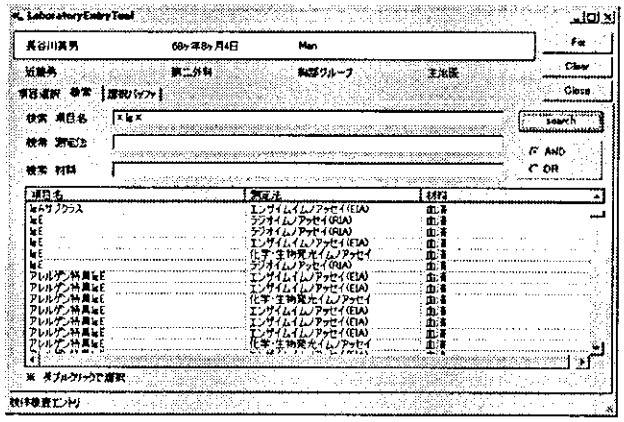
C.12.4.1　Menu

メニュ画面は（C.10.3）に則って作製した．

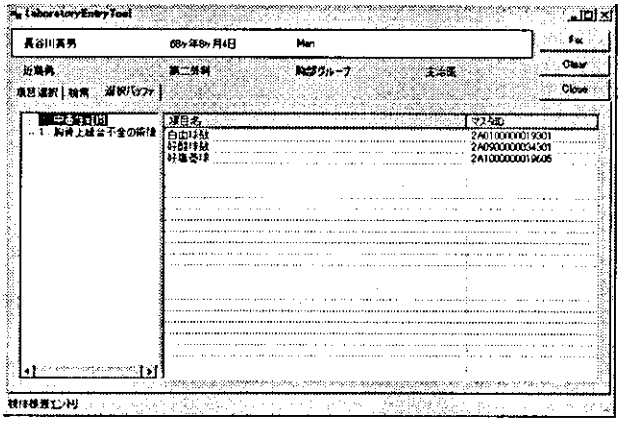


C.12.4.2　Search

メニュに存在しない項目は，3軸の and/or にて検索可能とした．



C.12.4.3　Targeting

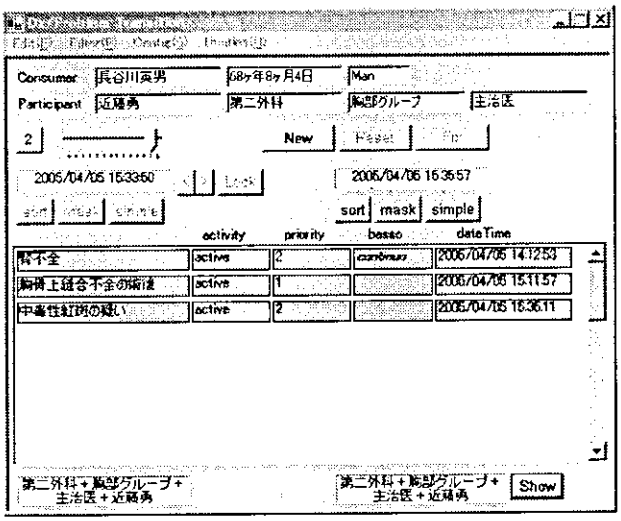


C．12．5 病名変遷 Editor

画面例を列挙する（実装は尾藤，与那嶺；基幹クラスは山田）．

C.12.5.1 Single pane

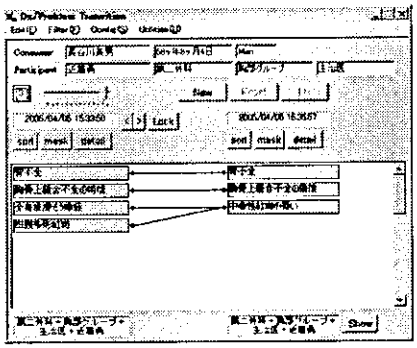
通常の一号様式保険傷病名欄と大差ない．

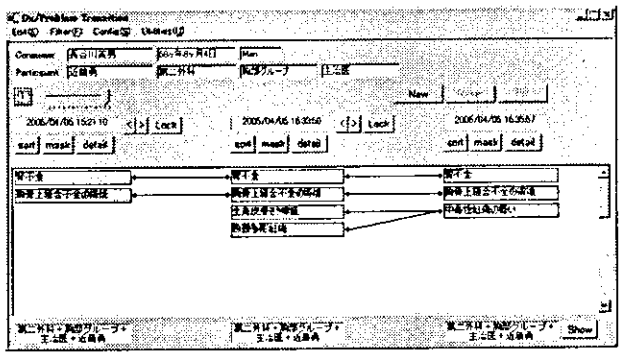


ただactivity, priority, bassoを設定できるようになっている．

C.12.5.2 Multi pane

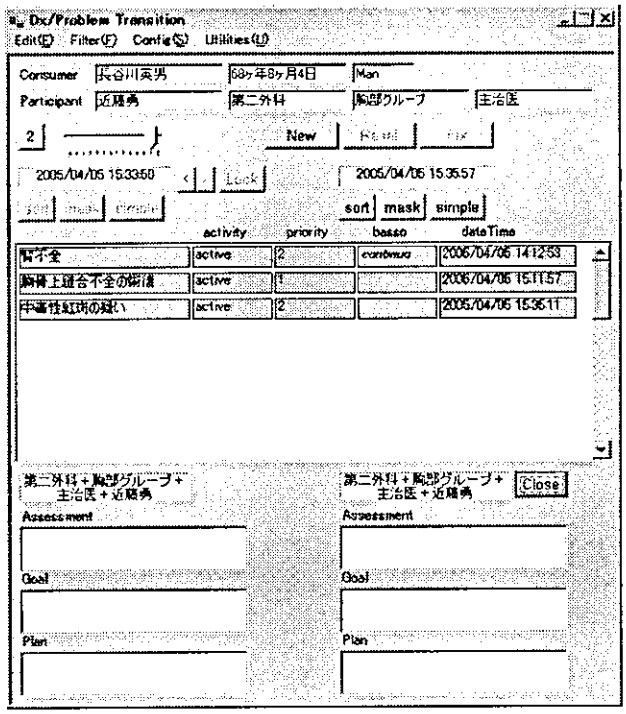
Multi-paneにすると，病名変遷が閲覧編集可能となる．





C.12.5.3 AS, GL, PN

Assessment, Goal, Planも入力可能とした．

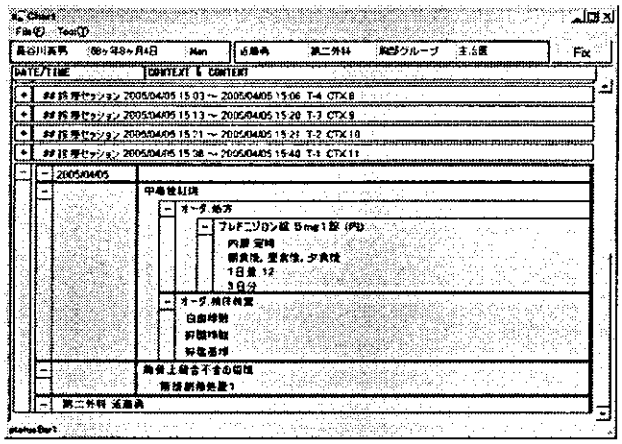


C．１２．６　加療履歴と病名連関

画面例を列挙する（実装は大嶺；基幹クラスは山田）．

C.12.6.1　診療ブロックの構成例

前述（C.12.2～4）の入力結果は以下のように表示される．



これで病名と加療行為が関連付けられた．

C.12.6.2　アウトライン化の構成例

必要に応じて，加療行為履歴をアウトライン化できる（C.10.4）．