ASP型電子カルテシステムにおける手術・麻酔記録モジュールの設計とその実装

新倉　正三† 　横田　裕介†† 　北岡　有喜††† 　上林 弥彦*††

† 京都大学工学部情報学科　〒 606-8501 京都市左京区吉田本町
†† 京都大学大学院情報学研究科社会情報学専攻　〒 606-8501 京都市左京区吉田本町
††† 国立京都病院医療情報部　〒 612-8555　京都市伏見区深草向畑町 1-1
E-mail: †niikura@db.soc.i.kyoto-u.ac.jp, ††{yyokota,yahiko}@i.kyoto-u.ac.jp,
†††ykitaoka@mbox.kyoto-inet.or.jp

あらまし　現在医療アプリーションとして ASP 型の電子カルテシステムが構築されつつある。本稿では電子カルテシステムにおける手術・麻酔記録を行うモジュールの設計と実装について述べる。医療アプリケーションでは柔軟なデータの追加・変更を可能にすることが重要である一方、入力ミスや改竄を防ぐための仕組みも不可欠である。本システムではすべての入力履歴を保存し容易に参照・復帰可能にすることでこの問題に対処している。また、入力を行う利用者の役割や入力を行う状況の違いに基づいてアクセス制御を行い、誤操作やセキュリティに対処すると同時に様々な立場の利用者が本システムを使用することを可能にしている。

キーワード　医療情報システム, 電子カルテ, 手術記録

Design and Implementation of an Operation Recording Module for an Electronic Health Recording System

Shozo NIIKURA†, Yusuke YOKOTA††, Yuki KITAOKA†††, and Yahiko KAMBAYASHI††

† Dept. of Information Science, Faculty of Engineering, Kyoto University　Yoshidahommachi, Sakyo-ku, Kyoto, 606-8501 Japan
†† Dept. of Social Informatics, Graduate School of Informatics, Kyoto University　Yoshidahommachi, Sakyo-ku, Kyoto, 606-8501 Japan
††† Dept. of Medical Informatics, Kyoto National Hospital　Fushimi-ku, Kyoto, 612-8555 Japan
E-mail: †niikura@db.soc.i.kyoto-u.ac.jp, ††{yyokota,yahiko}@i.kyoto-u.ac.jp,
†††ykitaoka@mbox.kyoto-inet.or.jp

Abstract　We have been developing an Electronic Health Recording system, based on Application Service Provider technology. In this paper, we introduce a new practical module for operation recording. It is important on medical application that operation recording modules can easily handle a variety of data types and access operations (e.g., append and change). In medical applications, erroneous or abnormal data input causes critical problems. In order to solve such problems, we implemented functions to store all input data and logs and enabled medical staffs to make reference to and restore past data. Moreover, we realized a specified access control policy based on users and situations, in order to prevent abnormal data inputs and to keep the system secure.

Key words　medical infomation system, Electronic Health Record, operation recording

1. はじめに

現在、従来型の紙ベースの診療録から電子カルテへの移行が各医療機関において積極的に行なわれている [1]。媒体が紙から

* 2004 年 2 月 6 日逝去

電子データに変わるだけでも、例えば診療録管理において、従来は診療録管理の専門家が十分な管理スペースを用いて管理しなくてはならなかったが、電子カルテの場合それを全てシステム側で管理することができ、管理スペースと管理コストを大幅に削減可能となる。なお日本診療録管理学会の調査によると、我が国の現状は 1994 年の調査 [2]、1999 年の 2 回目の調査とも、大半の病院が診療録管理の専門部門を持っていないという結果が報告されている。他方、アメリカでは診療録管理の有資格者が約 10 万人 (1985 年) おり、単純計算するとどの病院にも平均約 10 名の有資格者が配置され診療録管理にあたっていることになる [1]。

他にもカルテを電子化することで、診療情報のシームレスな共有やデータベースとの連携による応用が可能となる [3]。例えば一つの病院内において、看護士や医師が患者に対して行なった処置を逐一記録し、それを他所にいる医師が任意のタイミングで閲覧することが可能となり、医療事故の未然防止や医師・看護士の正当性を証明する資料となる。さらに、複数の医療機関で電子カルテシステムを共通化し診療録を共有することができれば、医療の連携が可能となり例えば薬の重複処方を避けることができる。またこれを一つの地域内で実践できれば、患者は医療上の各種手続きをシームレスに行なうことが可能となる [4]。従来の医療システムでは、例えば医療機関同士で患者を精密検査などのために行き来させる場合、患者は非常に煩雑な手続きを強いられる。多くの患者たちはこのような手続きを避けるため、近くの診療所ではなく設備の整った病院まで診察を受けに行っていた。このため病院に患者が集中し、病院の外来診療は待ち時間が長く診療時間は僅かとなってしまい、結果として患者への医療サービスが低下してしまった。地域内で医療の連携ができていれば、そのような手続きを簡略化することが可能となり、さらに電子カルテが共通化・共有化されていれば手続きは瞬時に終了する。このように電子カルテシステムを用いることにより、各種手続きはシームレスに行なわれ、病院は高度専門医療に専念でき、診療所では病院での治療を終えた後の患者や一般の外来患者に対し、生活現場に根付いた医療サービスを行なうことができるようになる。また高額の医療機器を無理に診療所内に設置する必要もなくなる。

またデータベースにアクセスしてデータ集計を行ない会計処理などに利用することや、データマイニングによる *EBM*(Evidence Based Medicine: 事実に基づく医療) への応用も可能である。ここで evidence とは実際の診療結果から得られた経験的な事実をいい、EBM とはそうした事実をもとに有効性の高い治療方法を選択して医療行為を行なうことをいう。従来は医師の勘や経験に基づき evidence を推測し抽出してきたが、電子カルテの場合には膨大な量のデータの中からデータマイニングによって新しい evidence を発見し、それを次の医療に活かすことが可能となる。

ここで電子カルテが地域医療連携においてより有用に使われるためには、患者の全ての診療情報が電子的に記録されていることが望まれる。なぜなら地域内で電子カルテを共有していながら、治療履歴の一部が参照できないということは、自施設への患者の囲い込みを意味しており、地域医療連携の思想に反している。また、データマイニング時の基礎資料が欠けることになるため、応用範囲を狭めてしまう恐れもある。しかし現状では未実装のモジュールが多数あり、その中の一つに手術・麻酔記録モジュールがある。手術中に使用される薬剤の量は治療費の計算やデータマイニングの際に参照されるので、データベースとの親和性が重要である。また手術中はリアルタイムにデータが入力され、なおかつ複数のユーザからアクセスされることを想定しなくてはならないので、ユーザインタフェースやアクセス制御モデルの設計も重要である。本稿ではこのモジュールの設計と実装について報告する。

2. では、麻酔記録の現状と本稿で実装を行なうモジュールの達成目標を示し、関連研究として電子カルテの応用例である地域医療連携プロジェクトを紹介する。3. では、本モジュールの設計を行ない、4. では、実際の実装報告とユーザインタフェースの概要について説明する。5. では、本稿での成果と今後の方針についてまとめる。

2. 研究の背景と関連研究

手術・麻酔記録モジュールの詳細に入る前に、まず麻酔記録の現状について述べる。麻酔記録は手術中に麻酔医が一人で記入を行ない、患者名や医師名などの手術に関する基本情報や、導入した麻酔の量や時間、血圧や脈拍数などのバイタルサインの時間推移、その他手術中に発生する様々なイベントを逐一麻酔記録用紙に記入していく。麻酔医の仕事はこの記録を行ないながら、患者のバイタルサインの変化に合わせ麻酔の導入量の調節や点滴を行なうなど、患者の状態を終始一貫して管理することである。このため麻酔医のミスは患者の生命を脅かす可能性が高く、手術・麻酔記録モジュールを開発するに際し、麻酔医が手術に集中できる作業環境の構築が肝要である。また、現在の麻酔記録用紙による記録では、患者のバイタルサインの変化を正確に読み取ることがそれほど容易ではない。本モジュールではこのバイタルサインをできる限り麻酔医にとって見やすい形で表示することが求められる。

こうした現状を受けて本モジュールを開発するにあたり、以下のような要求を定義した。まずバイタルサインの変化が時系列的に把握できるように、バイタルサインの各データは折れ線グラフとして表示する。さらにこの折れ線グラフの表示方法を容易に変更するためのインタフェースを用意する。そしてその表示方法をプロファイルとして登録することにより、次回以降の手術でも瞬時に麻酔医の趣向に合った表示方法に変更できるようにする。次に、点滴や麻酔の導入がどの時間帯に行なわれたのかをバイタルサインのグラフと併せて表示する。この情報についてはシンプルな帯状のグラフで表示し、時間帯が判るようにする。また、電子カルテの特徴として情報の共有があり、本モジュールでも共有できるようにする。すなわち手術室の外にいる医師が本モジュールにアクセスし、手術の状況をリアルタイムに閲覧できるようにする。これにより多重のチェック機能が働き、麻酔医のミスや患者の異常の早期発見が期待される。

次に、実際に電子カルテを導入した事例をいくつか紹介し、

本稿で実装を行なう手術・麻酔記録モジュールとの関係について説明する。

まず大橋産科婦人科では古くから WINE [5] という電子カルテが開発されてきた。1989 年に稼働を開始した第 1 世代電子カルテ WINE(WISE & NEAT; 賢く手際の良い医療秘書の実現を目指した開発コード名) は文字だけしか扱えなかった。その後も WINE の開発は大橋医師ら個人によって続けてられ、現在では小規模診療所から中規模病院までをカバーできるほどの本格的かつユニークなシステムに成熟し、診療所などで広く利用されている。

津山中央病院 [6] のように一つの病院・医院だけで独自の電子カルテシステムを導入した事例も多い。このようなシステムは動作が安定しており、病院内で発生する膨大な量のトランザクションを高速に処理することが可能である。またクローズドなシステムであるため、患者の個人情報に対するセキュリティレベルが高く、法的な面での問題点も少ない。反面導入時の単価が高く独立性も高いので、地域医療連携の中で使うのは難しいという側面を持つ。

続いて実際に地域医療連携を行なっている事例を紹介する。九州ではドルフィンプロジェクト [7] のもとで熊本 [8] と宮崎 [9] の二つの地域で地域医療連携が行われている。ドルフィンプロジェクトの特色は、MML(Medical Markup Language) を利用してデータの交換を行なうことと医事会計ソフトとして ORCA(Online Reciept Computer Advantage) を採用していることである。このプロジェクトに参加する各医療機関は院内に MML 対応の電子カルテシステムを構築し、i-Dolphin というセンタサーバと MML を介して電子カルテデータの交換を行ないそれを共有する。また、患者自身が自らのカルテを自宅から Web ブラウザ経由で閲覧することができる。

ここで MML と ORCA について補足する。まず世界的な標準の医療情報の記述モデルとして HL7(Health Level 7) [10] [11] といった形式がある。これに対し日本が独自に XML 形式を導入したデータ記述言語が MML(Medical Markup Language) であり、現在もその策定が進められている。これは医療機関の間で診療データを相互に交換するためのものであり、2004 年 2 月現在 Ver 3.0 が公開されている [12]。MML では手術記録情報についての記述もなされているが、本モジュールで扱うような手術中の直接的なデータよりむしろ、患者情報や医師情報、手術概要などの基本情報のみを記述しているだけで、本モジュールでは利用できない。各医療機関が、月に一度保険会社に診療報酬の請求をする際の報告書 (レセプト) を作成するために開発されたのが、レセプトコンピュータであったが、これは高価で維持コストも高かった。そのため日本医師会主導により ORCA プロジェクト [13] を興し、高機能かつ低コストを実現した ORCA [14] を開発した。ORCA はオープンソースで開発され、オンラインで接続されることによりソフト本体やデータベースのマスタ情報を常に最新の状態に保つことができ、維持コストがほとんどかからないのが特徴である。

国立京都病院では伏見医師会と協力し、**１地域１患者１電子カルテ構想**などの診察情報の共有を地域内全域で行なうことによって、地域内の全医療機関を一つの仮想医療機関とみなす**地域医療ユニット構想**を掲げ、全国共通１患者１電子カルテシステム実現のためのモデル作成を試みている [4]。この地域医療ユニットは IT 戦略本部による e-JapanII 基本戦略において先導的取り組み (1) 医療のビジネスモデルとして採り上げられた。電子カルテシステム自体は Apius 社の Apius Ecru [15] というシステムを ASP として導入し、各医療機関では ORCA サーバを配備する。ここで ASP(Aplication Service Provider) とは、利用されるアプリケーションを企業など情報・サービスを共有しようとする一つの団体にとってパブリックな場所に設置し、Web ブラウザなどを通じて利用する形態である。これにより伏見地区内の各医療機関は、院内に電子カルテシステムを構築することなく、少ない導入コストで容易に電子カルテを利用することが可能となる。またこのプロジェクトでは公衆無線 LAN サービスと連携して、かかりつけ医が往診時に電子カルテを見ながら診察を行なうといったモデルも提案しているなど、ASP であるため利用形態の柔軟性も高い。

本稿では産官学連携のもとで、まだどの電子カルテシステムでも実装されていない手術・麻酔記録モジュールを、この Apius Ecru へ組込むことができるよう、Web アプリケーションとして設計しプロトタイプの実装を行なった。

3. 手術・麻酔記録モジュールの設計

3.1 モジュールの概要

現在急速に電子カルテシステムが開発されているが、手術・麻酔記録を行なうためのモジュールは未だ開発されておらず、2. で示したように麻酔医に多大な負担を強いているのが現状である。そこで本稿では電子カルテの開発促進と麻酔医の負担軽減を目的として、手術・麻酔記録モジュール (以下、オペレコと呼ぶ(注1)) を開発することにした。オペレコは Web アプリケーションとして実装された電子カルテ用モジュールで、手術中に発生する麻酔導入などのイベントとバイタルサイン (血圧・脈拍数など) の時間推移を記録し、それをユーザ (主に医者) に見やすい図として表示する。この図は、バイタルサインを折れ線グラフとして、それ以外の麻酔などのデータを帯状のグラフとして表示したもので、アクセスされるたびに動的に生成される。オペレコを利用するユーザには、手術中にデータ入力を行なうことのできる麻酔医 (1 人) と手術中・手術後に関係なく閲覧するだけのユーザ (制限無し) がおり、前者だけが手術中に限りデータの入力を行なうことができることとする。また手術中でも院内にいる上級医や院外の専門医など、手術室の外にいるユーザがアクセスして手術の状況をリアルタイムに確認できる。このようにすることで院内のヒューマンリソースを有効利用できるようになるとともに多重のチェック機能が働くようになり、医療の質の向上につながる。

また電子カルテを利用し、全ての医療行為をウェアハウス上にデジタル保存することでデータマイニングを行なうことが可能となる。その結果、現状の医療の客観的な評価と新たな予

(注1)：OPEration RECOrding Module: オペレコ

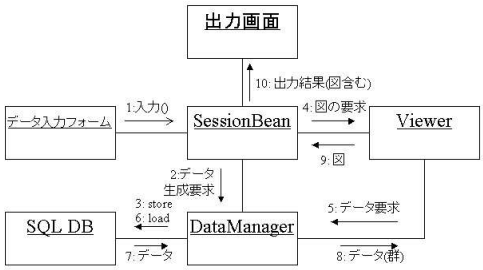

図 1　オペレコ内部でのデータフロー

防・診断・治療・医療経済政策の evidences を創出することが可能となる。オペレコはこの枠組みの中において手術データの生成・格納と、データマイニングなどのデータ処理プログラムからの要求に対応するデータの取出しを行なう。

オペレコは電子カルテにおける一モジュールであり、基本的な手術セッションの管理・コントロールは全てそのシステム側が行なうことを想定している。したがって諸々の電子カルテシステム上でオペレコを利用するためには、手術セッション情報をやりとりするためのインタフェースが必要である。しかしプロトタイプの実装にあたり、本モジュールだけでも手術・麻酔記録を行なうことができるよう、簡単なセッション管理を内部的に実装することにした。

3.2　モジュールのアーキテクチャ

まずシステム全体のアーキテクチャについて説明する。想定しているシステムは ASP 型の電子カルテシステムとし、基本的な手術に関係する情報、すなわち執刀医や麻酔医、患者、手術概要等は電子カルテシステムが取り扱うことを想定している。オペレコは、この電子カルテシステムより手術セッション情報を受け取り、手術・麻酔記録を行なうように設計されている。しかし、現時点では本モジュールの導入先の電子カルテシステムが確定していないため、プロトタイプを作成するにあたり手術セッションのオープンとクローズだけは行なえるようにした。

続いて図 1 にオペレコ内部でのデータフローを示す。

この図において SessionBean, DataManager, Viewer は以下のような役割を果たす。

SessionBean はセッション情報を保持するとともに、DataManager や Viewer との通信・制御を行なう、オペレコの本体部分に相当する。また、ユーザ側からの入力の窓口もこの SessionBean が担っている。

DataManager は手術中の被記録データを全て核データ (3.4 参照) として格納・管理し、必要に応じてデータベースとの通信や Viewer へのデータの提供を行なう。この際格納されている全てのデータが核データであることが保証されているため、統一された操作でデータの受け渡しが可能となっている。DataManager は原則として 1 回のオペレコセッションにおいて 1 つだけ生成されるものとする。

Viewer はオペレコの出力用メインモジュールで図の生成を行なう。強調表示や表示すべき時間区間などのカスタマイズ情報を保有し、SessionBean からの要求に応えて図を生成する。このカスタマイズ情報はプロファイルとして登録できるものとし、状況やユーザの嗜好に応じて選択できるものとする。Viewer も原則として 1 回のオペレコセッションにおいて 1 つだけ生成されるものとするが、複数あっても特に何らかの依存関係を破壊するものではない。

ここで図 1 の状態遷移を追いながらその動作を説明する。

（1） ユーザはデータ入力フォームから手術中に発生したイベントを送信する。リクエストを受け取った SessionBean はその入力が適正かどうかをチェックし、適正であれば受理して (2) へ進む。非受理の場合にはエラー画面を出力し入力のやり直しを指示する。(図 1 には記述していない)

（2） SessionBean は (1) のリクエストを解析し、DataManager にデータの生成を指示する。

（3） DataManager は (2) の要求に応じてデータを生成し SQL DB へそのデータを格納する。

（4） SessionBean は無事にデータが生成・格納されたことを確認したら、Viewer に対して図の生成を要求する。

（5） Viewer は (4) の要求に応じて、図を生成するのに必要となるデータを DataManager に対して要求する。

（6） DataManager は (5) の要求に応じて、現在保持しているデータを確認し不足しているデータがあれば SQL DB へデータの取り出し要求を送る。

（7） SQL DB が (6) の要求に応じデータを返す。

（8） DataManager は (5) で要求されたデータを Viewer に渡す。

（9） Viewer は (4) で要求された図を生成し、生成された図のファイル名を SessionBean に返す。

（10） SessionBean は、(9) で受け取ったファイル名から <img> タグを生成し、出力画面に html コードを出力する。

手術に関する基本情報を受け渡ししたり、データ処理モジュールに大量のデータを渡しするためには、オペレコにそれを行なうためのインタフェースを用意する必要がある。このようなインタフェースは SessionBean と直接に通信を行ない、セッションのオープン・クローズやデータの受け渡しを行なう。

3.3　アクセス制御モデル

オペレコの利用にあたり利用者の役割とその状況の違いに基づいたアクセス制御を行なう必要性がある。まずオペレコの利用者について整理すると、

（1） 麻酔医などの記録者 (RW)

（2） 一般の医師や上級医などの閲覧者 (RO)

（3） データマイニングなどのプログラム (DM)

の 3 つに分類される。各利用者毎の利用状況について整理すると、RW がオペレコを用いて手術を記録するのは原則として手術中だけであり、DM がオペレコからデータを取り出すのは手術後が主である。RO は手術中・手術後に関係なくアクセスすることになる。

ここでは利用者の種別毎にアクセス制御モデルを考えることにする。まず RW は原則として全てのアクセス (保存・読込) を可能とし、同時に開くことが可能な RW セッションは高々 1 セッションとする。これにより同時アクセスによるデータ書込の衝突を未然に防ぐことができる。次に RO は原則として読込

のみ可能とし、コメントデータの書込だけはできるようにする。これにより上級医や専門医のアドバイスなどが手術中に即座に手術室に伝達されることが期待される(注2)とともに、手術後でも一般の医師などが特に注目すべき事項に対してコメントを記述することができるようになる。複数のユーザからのアクセスを可能にするため、同時に開くことのできるセッション数に対する制限は特に設けないことにする。最後の DM については、データを高速に読込みそれをプログラムに渡すことができればよい。したがってオペレコの主要機能の一つである図の生成も不要である。セッション数に対する制限は特に設けないことにする。

次にセッションの管理において意味を持つものとして、手術開始・手術終了・中断・再開・破棄・閲覧開始・閲覧終了という７つの操作がある。これらの操作 (op と表記する) の概要を以下に示す。ここで (1)〜(5) の操作については原則として RW だけが実行できる。RO と DM は図の生成をするかどうかが異なるだけなので、両者をほぼ同等とみなし (6),(7) の操作だけでセッションの管理を行なう。

（1） **手術開始** op を受けてオペレコは新しい RW セッションを生成し、それに伴うデータを生成する。

（2） **手術終了** op により DataManager 内の全データをデータベースに書き出すなどの後処理を実行し、RW セッションを終了する。一度手術終了 op を実行してしまうと、もうその手術に RW としてアクセスすることはできない。したがって、手術の継続を行ないたい場合には、次に述べる中断・再開 op を利用する。

（3） **中断** op は手術終了 op とは異なり後処理を一切行なわず、RW セッションを一時中断する。次に述べる再開 op を実行することで、再び RW としてその手術にアクセスすることが可能である。

（4） **再開** op は中断 op で中断された RW セッションを再開するための操作である。手術の実情を踏まえ記録を行なう医師の変更を許容する。

（5） **破棄** op は開かれている RW セッションで記録した内容を全て破棄するための操作である。医療用のアプリケーションであることを考えるとあまり意味がないが、患者コードの入力ミスにより無効な RW セッションが開かれるなどの事故を想定して用意した。

（6） **閲覧開始** op は新しい RO セッションを生成する。RO セッションでは、データベースに格納されているデータを読み出し DataManager 上に状態を再構築する。最新のデータは常にデータベース上に置かれているものとし、アクセスのたびにデータベースに問い合わせを行なう。RO セッションでは原則として読み出し専用で新しいデータを生成しないようにするが、「コメント」データだけは記録できるようにする。オペレコでは常に一意に定まるデータ ID を発行することが可能であるため、これによる衝突の心配は無い。

(注2)：手術室では最低でも 5 分に 1 度はバイタルサインの入力が行なわれるため、最悪の場合でもそのタイミングで伝達されることになる。

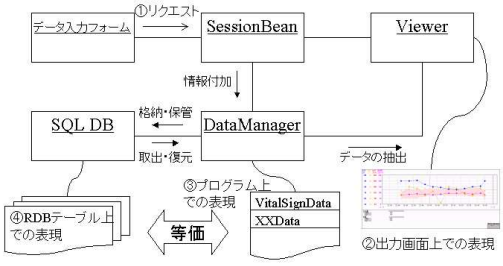

図 2　４つのデータ表現

（7） **閲覧終了** op は RO セッションを終了する。

3.4　データ構造

まずデータ構造について述べる前に、オペレコで扱われるデータの表現方法について述べる。一つのデータは次の４つの表現によって表わされる。

（1） 入力フォーム上での表現 (データ入力フォームから送信されるリクエスト)

（2） 出力画面上での表現 (Viewer によって出力される図)

（3） プログラム上での表現 (DataManager に格納されるデータ)

（4） RDB 上での表現 (PostgreSQL 上のレコード)

図 2 に４つの表現の概要を示す。各表現はそれぞれ異なるが、少なくとも (3) と (4) は等価でなければならない。具体的には、(1) の内容 (1 レコード分) を正確に反映しさらに情報を追加してプログラム内部で表現したものが (3) であり、(2) の図も (3) から抽出されたデータを元に生成される。すなわち、(1) と (2) は (3) の一部分でしかない。オペレコは手術中・手術後のどちらのタイミングでもアクセスされ、表示させる条件が同等なら常に同じ図を生成しなくてはならない。しかし DataManager 中の情報は各セッションごとに固有のものであり、これを他のセッションの DataManager に複製する手段が必要となる。そこで (3) の内容を全て (4) の形式で表現し、各セッションでは (4) の表現を介して同じ情報を共有することにした。これにより DataManager の内容がリアルタイムに変化していく手術中でも、閲覧者は常に最新の情報を確認することができるようになる。本節では (3) の DataManager に格納されるデータの表現方法について述べ、(4) の表現への変換が容易であることを確認する。

手術・麻酔記録の中で記録されることが想定されるデータは、バイタルサインのような (時刻, (値列)) の形式で与えられる**単一データ**と、点滴など (時刻 1, 時刻 2, (値列)) の形式で与えられる**区間データ**とがある。また主要な記録対象である麻酔については、上述の単一データと区間データの両方を利用する可能性がある。そこで我々は**双方向リストモデル**を採り入れ、(LinkPrev, LinkNext, 時刻, (値列)) という形式で与えられるデータ構造を構築した。このデータ構造を採用したデータのことを本稿では**核データ**と呼ぶ。

このデータ構造を採用することで、区間データは２つの互いに対になる核データによって内部的に表現することができる。

またあるデータに対する修正用のデータを一つの核データとして表現することで、対象となるデータを変更することなしにデータの修正や削除を行なうことができる。ここでは修正データの適用に関する概要を述べるのみとし、3.5 で例を用いて詳しく説明する。単一データ X に対して、修正データ Y,Z を適用すると図 3 のように元データに対して修正データが直列に最後尾に追加されていく。

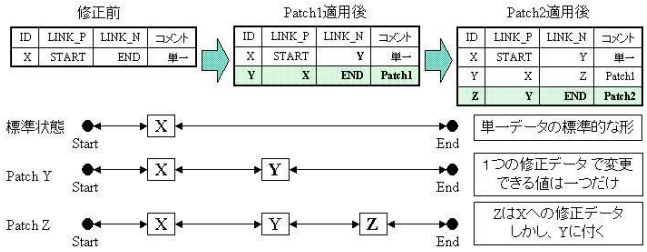

図 3　単一データの修正

また、区間データ X1〜X2 については図 4 のように X1,X2 のそれぞれについて修正データが直列に付き、それが結合される。

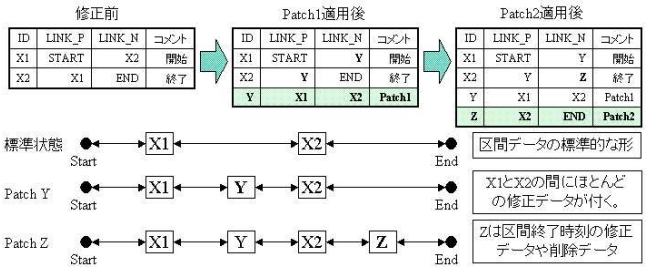

図 4　区間データの修正

次に、情報量の多い点滴や麻酔のデータ表現について述べる。点滴や麻酔などは複数の成分が混合されており、そのそれぞれについて量や速度が異なる。これを一つのデータとして扱うとデータの構造が複雑になり、データベースへの格納や取出しが困難になってしまう。そこでこの問題を解消するために、これらのデータを各成分ごとに区間データで表現することにした。このように分離することでデータ構造の単純性が維持され、各データの形状がほぼ同型であるためデータベースとの親和性が向上し、データマイニングなどのデータ処理が行ないやすくなる。また、点滴や麻酔はある一定の速度で数時間導入される可能性があり、この情報を区間データでより適切に表現するため、区間データの不完全な状態を許容することとする。すなわち図 4 の場合、X1 のみを生成した状態を許容し、点滴や麻酔の導入が終了した時点で改めて X2 を生成する。ここで区間データの内部構造について少し言及する。区間データでは X1 に区間開始時刻とその区間の間に実施されるほとんど全ての情報が記述され、X2 には区間終了時刻だけが記述される。したがって X2 に対して付される修正データ Z は、その区間データが有効か無効かを記述する削除／復帰情報か区間終了時刻の修正情報のみを記述する。ある麻酔の導入量 (速度) を変更する場合、オペレコはその変更のあった時刻で区間データを閉じ (X2 の生成)、新たに区間データを開く (X1 の生成) ことでその状況を記述する。ただし、この不完全な状態は RW セッション終了時までに解消される必要があり、RW セッションが何らかの形で終了した際には、後処理として区間終了データを自動的に付すことを想定している。なお本稿では点滴や麻酔の中断・再開をオペレコ側で関知する必要性はないと判断し、またデータ構造を単純に保つため、その都度区間データを生成することで対応する。

以上から、オペレコ上で表現される全てのデータは単一ないし複数の核データによって構成可能であることが確認できた。また、ひとつひとつの核データはそのまま RDB 上のレコードとして容易に表現可能であるため、本節の初めで述べた (3) プログラム上での表現と (4)RDB 上での表現との等価性が満たされているといえる。

3.5　データの具体例とモジュールの動作

オペレコで発生するデータは全て 3.4 で述べた核データによって記述される。**バイタルサイン**の 1 つのデータは例えば以下のような表現によって記述される。ただし、RDate, IDate はそれぞれオペレコが受信した時刻、値として入力された時刻を表す。

元データ

ID:10001, Type:VitalSign, LinkPrev:START, LinkNext:END,
RDate:[2003/12/23 13:52:51], IDate:[2003/12/23 13:50:00],
血圧 (上):131, 血圧 (下):78, 脈拍数:83, 体温:36.2, SaO2:92.0

オペレコはこの情報を例えば以下のように出力する。ただし <> で囲まれる部分は全ての核データにおいて保持されることが保証されているデータであり、これを info データと呼ぶ。

オペレコ上の表現

<"10001" DATA_VitalSign: LinkPrev: -1 LinkNext: -2
[2003/12/23 13:52:51]> [2003/12/23 13:50:00]
bpH:131 bpL:78 pul:83 tem:36.2 SaO2:92.0

LinkPrev と LinkNext は双方向リストにおいて連結される前と後ろのデータの ID を表す。核データで基本的な単一データを表現する場合、LinkPrev, LinkNext にはそれぞれ START と END を表す値が与えられる。このデータに対して修正が行われたとき LinkNext は新しく生成される修正データの ID に書き換えられ、修正データの LinkPrev には修正対象の ID がセットされる (図 3, 4 参照)。修正データの形式は非常に単純で、((info), DataType, Field, Value) で与えられる。このように単純な構造にすることで複雑なデータの修正も可能になった。ここで DataType はバイタルサインなどのデータの種別を表し、Field は血圧などのデータ項目を表す。したがって一つのデータに対して複数の修正を行なう場合には、修正データを複数生成して、それを順に双方向リストにより連結することで実現される。

例えば、上述のデータを以下のように書き換える場合を考える。

修正内容

IDate:[2003/12/23 13:45:00], 血圧 (上):140

書換え対象は２つのフィールドなので、このデータは以下のように記述される。

修正後のデータ

<”10001” DATA_VitalSign: LinkPrev: -1 LinkNext: 10002
[2003/12/23 13:52:51]> [2003/12/23 13:50:00]
bpH:131 bpL:78 pul:83 tem:36.2 SaO2:92.0
<”10002” DATA_Patch: LinkPrev: 10001 LinkNext: 10003
[2003/12/23 13:53:35]> Type:VitalSign
Field:IDate Value:[2003/12/23 13:45:00]
<”10003” DATA_Patch: LinkPrev: 10002 LinkNext: -2
[2003/12/23 13:53:50]> Type:VitalSign
Field:bpH Value:140

4. 手術・麻酔記録モジュールの実装

4.1 システム構成

サーバ側は以下のように構成した。

- OS: Debian GNU/Linux 3.0 woody
- DB: PostgreSQL 7.2.1
- JSP Engine: Tomcat 4.0.3
- Java: Sun JDK 1.4.1
- 使用言語
  - Java(システム・本体部分)
  - JSP(JavaBeans との連携)
  - HTML, JavaScript(フロントエンド)

クライアント側については、以下のようなシステム構成を想定している。

- ブラウザ: Netscape 4.x など Mozilla 系
  - 要 JavaScript・フレーム対応
- 表示: XGA サイズの液晶ディスプレイ
- ユーザ: 手術室にいる麻酔医とその手術を監督する上級医、外部にいる専門医など

オペレコは Web アプリケーションとして様々なクライアントからアクセスされる可能性があるので、アプレットなどブラウザや OS に対する依存性の高い技術は極力使わないように配慮した。ただし十分な利便性を簡易に実現するために JavaScript だけは導入した。JavaScript はブラウザの種類やバージョンによって利用できる命令が異なるが、今でも利用者の多い Netscape 4 系統の仕様の範囲内で記述し、最新の Mozilla や (Windows 版)Internet Explorer 5.5 以降でも正しく動作するように仕様を確認しながら記述した。

4.2 データクラスの実装

まず図 5 にオペレコで利用される主なデータクラスとその関係をクラス図として示す。図 5 において、クラス間を結ぶ矢印は継承関係を表わし、矢印の先が s スーパークラスである。また先端が菱形 (◆) の線は has-a 関係を表わし、◆の付いている方が持つ側である。線の横に書かれた数字は、対応関係を表わしており、1-1 なら「１つのオブジェクトが１つのオブジェクトを持つ」ことを表わし、1-N なら「１つのオブジェクトが複数のオブジェクトを持つ」ことを表わしている。

ここで図中右側の DataManager, SessionBean, Viewer は

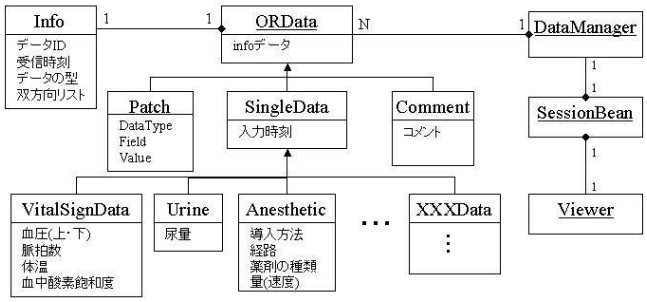

図 5　各データクラス間の関係

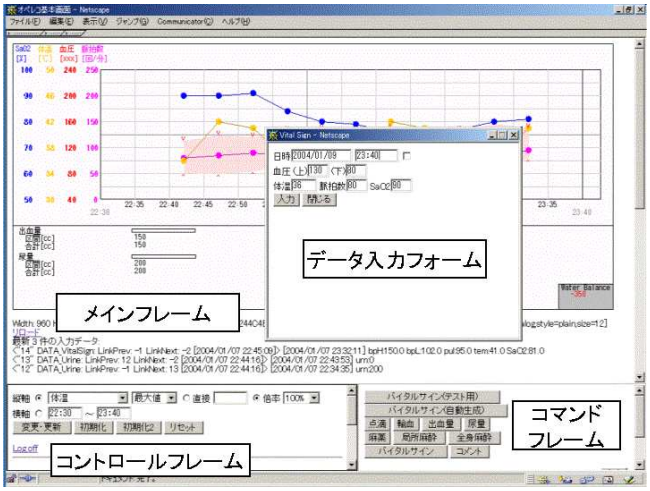

図 6　オペレコの基本画面

3.2 節にて説明したクラスであり、全体の制御を行なっている。残りのクラスは 3.4 節で述べたデータクラスである。

ORData は 3.4 節の核データを表わすクラスで全てのデータクラスのスーパークラスである。*Info* は 3.5 節で述べた info データを表わすクラスで、特に双方向リストの実現とそれに伴うデータの ID の管理を行なう。*Patch* は修正データを表わす。3.5 節で述べたように非常に単純な構造である。*SingleData* は基本的なデータクラスのスーパークラスであり、図 5 に見るように ORData に入力時刻を追加しただけのクラスである。*VitalSignData* 他 SingleData のサブクラスについては、各データ毎に必要となる値を追加しただけのクラスである。

4.3 ユーザインタフェース

本稿ではオペレコのプロトタイプを Web アプリケーションとして実装した。ここではそのユーザインタフェースについて説明する。

オペレコにアクセスすると、まずログイン画面が表示される。ログインするとユーザのプロファイルが読み込まれ、RW セッションを開始し(注3)基本画面を出力する (図 6)。基本画面は以下のように３つのフレームに分割される。メインフレーム (画面・上) ではオペレコが生成する画像を常に表示する。コントロールフレーム (画面・左下) ではフォームが静的に配置されて

(注3)：今回のプロトタイプでは RW セッションのみ実装を行ない、他のユーザからの同時アクセス（RO セッション）には対応していない。

おり、メインフレーム上の表示方法のコントロールを行なう。コマンドフレーム (画面・右下) では各種コマンドボタンが配置される。このボタンを選択するとデータ入力用の小ウィンドウが生成され、そこからデータを入力する。これらのコマンドが実行されると即座にメインフレームの画像が再描画される。

コントロール用コマンドは、バイタルサインの表示方法や時間軸のスケールなど、ユーザの好みが強く反映されることが予想されるため、任意のタイミングでその設定値をプロファイルとして保存・読出ができるようにした。また、ユーザアカウント毎にデフォルトのプロファイルを指定できるようにもなっている。

次に描画される図について説明する。図 6 のメインフレームに折れ線グラフが描かれているが、このグラフの横軸が時刻を表し、縦軸がバイタルサインのそれぞれの項目の値を表す。血圧だけは上と下の値の関係が重要なので、血圧の上と下の折れ線グラフで囲まれる部分には特に色を塗っている。バイタルサインの表示は、コントロールフレームからの縦軸・横軸のスケール変更要求によって制御される。バイタルサインの下には尿量や点滴、麻酔などの使用量が数値として表示され、時間区間を帯状にして表現している。麻酔や点滴などは、そのデータが入力されたときに初めて図中にその項目・成分が描画される。ここで 3.4 節において述べた、麻酔や点滴などの区間データが不完全な状態にあったときの処理について述べる。まずデータ入力フォームから麻酔導入命令を受け取り、区間開始データのみを生成する (不完全な状態)。そして Viewer が図を生成する際、不完全な状態にある区間データを継続状態と判断し、描画時刻まで帯を伸ばす。区間終了データはデータ入力フォームからの入力か、RW セッションの終了によって生成される。

長時間におよぶ手術の場合は**ウォーターバランス**のチェックも重要である。ウォーターバランスとは手術中に出入りする水分量の合計値で、手術中の患者の状態を把握するための重要な指標の一つである。オペレコでは出血量や尿量、輸血、点滴などをもとにウォーターバランスを計算し、図中右下に表示する。

最後にコメントデータとデータの修正方法について述べる。メインフレームに表示される図の任意の点をマウスでクリックすると、オペレコにとってその点から近いと判断されるデータに対するデータ修正用の小ウィンドウが生成される。この小ウィンドウは、データ修正の他にコメントデータを入力することができるようになっており、コメントデータを入力すると図中に印が付き、その印をクリックすることでコメントを表示するようになっている。

5. まとめと今後の課題

本稿では ASP 型の電子カルテシステム上で動作する手術・麻酔記録モジュールについての設計と実装を行なった。またその中で双方向リストをベースにしたデータ構造を導入することで、麻酔や点滴などの複雑なデータの表現も簡単に行なえるようになった。一つ一つのデータの記述が単純なためデータベースとの親和性が高く、データ集計やデータマイニングなどのデータ処理に適した構造であるといえ、医療分野でだけでなく広く応用することが可能であり、データベース技術の実際的な利用分野として有望である。本モジュールの設計と実装により、麻酔医の手術中の負担の軽減とそれに伴う手術の質的な向上が期待されるとともに、これまで実装されてこなかった他の電子カルテ用モジュールの開発も促進されることが期待される。

今後はまずユーザインタフェースにおいてまだ完成していない部分の実装を行なう。具体的には、コメントデータと修正データの入力インタフェースと 3.3 節で述べた複数同時アクセスの制御の部分である。そして GUI の評価を行ない、より使いやすい GUI を模索するとともに、実際に稼働している電子カルテシステムと手術セッション情報の通信を行なうためのインタフェースを実装する。これにより本モジュールを電子カルテシステムへ組込むことが可能となり、医療の現場で利用されることが期待される。また、電子カルテシステムによっては XML を用いたデータ交換を行なうものもある。本稿ではデータマイニングなどのデータ処理を前提としており、パフォーマンス重視の立場から XML 形式のデータを利用せず、他モジュールへ直接データを渡すことを想定していた。しかし今後はそのような電子カルテシステムとのデータの交換も視野に入れ、XML 形式へのエクスポートも行なう予定でいる。

文　献

[1] 里村 洋一 (編), “改訂版 電子カルテが医療を変える,” 日経 BP 社, 東京, 2003.

[2] 日本診療録管理学会編, “我が国の病院における診療管理の現状,” 考古堂書店, 1995.

[3] 電子カルテ研究会, “新版 電子カルテってどんなもの？,” 中山書店, 2002.

[4] 北岡 有喜, “患者本位の医療を目指した地域医療情報ネットワークの構築,” 京都高度情報化推進協議会平成１４年度総会, 2002.

[5] 大橋 克洋, “WINE Project,” http://www.ocean.shinagawa.tokyo.jp/WINE/index.html .

[6] 宮島 孝直, “津山中央病院電子カルテシステムの実際,” http://www.seagaia.org/sg2003/ms/miyashima.html , 2003.

[7] “ドルフィンプロジェクト,” http://www.kuh.kumamoto-u.ac.jp/dolphin/ .

[8] 吉原博幸, “Dolphin Project 熊本地域での展開,” http://www.seagaia.org/sg2003/ms/yoshi/yoshi.html , 2003.

[9] 荒木 賢二, “地域医療情報の共有・活用を目的とした宮崎健康福祉ネットワーク (はにわネット),” http://www.seagaia.org/sg2003/ms/araki/araki.html , 2003.

[10] “HL7.org,” http://www.hl7.org/library/standards_non1.htm .

[11] “HL7 について,” http://www.threeweb.ad.jp/˜ hassan/face10/hl7/ .

[12] MedXML コンソーシアム, “Medical Markup Language(MML) Version3.0 規格書,” 2003.

[13] “ORCA プロジェクト,” http://www.orca.med.or.jp/ .

[14] “日医標準レセプトソフト,” http://www.jma-receipt.jp/ .

[15] Apius, “Apius Ecru,” http://www.apius.com/apius/index.htm .