一般口演 | 医療データ解析
一般口演13
モデル構築・検証
2021年11月20日(土) 14:10 〜 16:10 D会場 (2号館1階212)

[3-D-1-08] e-Path3層構造と FHIRを擁する医療ロジックの共通化モデルに適した In-Process Clinical Intelligence(IPCI)の設計と実装

*鳥飼 幸太[1]、野口 怜[1]、松山 龍之介[1]、白戸 悠貴[1]、齋藤 勇一郎[1] （1. 群馬大学医学部附属病院システム統合センター）
*Kota Torikai[1], Rei Noguchi[1], Ryunosuke Matsuyama[1], Yuuki Shiroto[1], Yuichiro Saito[1] （1. 群馬大学医学部附属病院システム統合センター）

キーワード：Clinical Path, HL7 FHIR, OSS, e-Path, IoT

医療情報はそれ自身の知識体系が極めて大きく、過去数十年にわたる超長期の蓄積に対し、データ面での整備がLOINC、 SNOMED、 HL7 FHIR等で進みつつある。一方チーム医療の実践においてはデータの「場面ごとの解釈」に相当する医療ロジックが不可欠である。今後クリニカルパスやアラートシステムで医療ロジックの高度な実装が求められるが、個別ベンダーの電子カルテや部門システムに分散しているためプログラムやロジック部分についての共通化についての見通しが得にくい現状がある。また、 e-Path 3層モデルについては今後超長期的なパス集積や共有を可能とする実装が求められる。本研究ではこの状況を踏まえ、現在流通している OSS等の特性を調査し、 Node.js/Node-REDを中核とする RESTfulアーキテクチャを選定し、超長期にわたり持続的な医療ロジックの蓄積、臨床に求められる迅速なフロー編集性やリアルタイム性を充足するアーキテクチャ設計(In-Process Clinical Intelligence: IPCIモデル)について詳述する。併せて、設計したモデルを実際のサーバに実装し、電子カルテデータベースと接続した医療ロジック実装とレスポンス検証を行うとともに、 e-Path 3層構造の実装およびアクセスポイント、 SIPサーバを通じた Push通知機能の実装を行った結果について報告する。

e-Path 3 層構造と FHIR を擁する医療ロジックの共通化モデルに適した In-Process Clinical Intelligence (IPCI) の設計と実装

鳥飼 幸太、野口 怜、松山 龍之介、白戸 悠貴、齋藤 勇一郎
群馬大学医学部附属病院 システム統合センター

Design and implementation of an In-Process Clinical Intelligence (IPCI) suitable for a common medical logic model with e-Path 3-tier structure and FHIR

Kota Torikai, Rei Noguchi, Ryunosuke Matsuyama, Yuki Shiroto, and Yuichiro Saito
System Integration Center, Gunma University Hospital

Abstract: Medical information has an extremely large body of knowledge of its own, and in response to the super-long term accumulation of data over the past several decades, progress is being made on the data side through LOINC, SNOMED, HL7 FHIR, etc. On the other hand, medical logic corresponding to the "site-specific interpretation" of data is essential for the practice of team medicine. On the other hand, medical logic corresponding to the "site-specific interpretation" of the data is indispensable for the practice of team medicine. In the future, advanced implementation of medical logic will be required in clinical pathways and alert systems, but it is difficult to achieve commonality of programs and logic parts because they are distributed in electronic medical records and departmental systems of individual vendors. In addition, the e-Path 3-layer model needs to be implemented to enable ultra-long term path accumulation and sharing. In this research, we investigate the characteristics of currently available OSS, and select a RESTful architecture with Node.js/Node-RED as the core, and design an architecture that satisfies the requirements for long-term sustainable accumulation of medical logic, rapid flow editing, and real-time performance. The architectural design (In-Process Clinical Intelligence: IPCI model) will be described in detail. We also report on the results of implementing the designed model on a real server, implementing the medical logic connected to the electronic medical record database, verifying the response, implementing the e-Path three-layer structure, and implementing the push notification function through the access point and SIP server.

Keywords: Clinical Path, HL7 FHIR, OSS, e-Path, IoT

1. 緒論

本研究は医療情報における「診療ロジック」(Clinical Logic: 以下 CL)の実装上での永続化を目的とする活動のひとつとして位置づけられる。

クリニカルパス(Clinical Path、以下 CP)は CL としての側面を有し、既に全国で活用されている。CP が抱える課題として、ある施設で作成した有益なクリニカルパスが別の施設で導入できない点が挙げられる。この課題に取り組む目的で 2018 年に e-Path 事業が AMED によって採択され、2021 年 3 月に事業が完了した[1]。E-Path 事業の成果として、異なるベンダーにおいて CP の互換性を担保するための 3 層モデルが考案されている。この3層モデルは outcome-measure をひとつのユニットとし、このユニットを結合、多層化することで CP フローを構成する。一方、病院の外来、入院、検査、投薬などを単位とする診療ワークフロー（Clinical Work Flow: 以下 CWF）のロジック化検討が UML モデルとして提案されている。さらに、近年診療内に導入が進められているアラート機能（Clinical Alert Function: 以下 CAF）や診療支援の形態である Clinical Decision Support System (以下 CDSS)も、アトミックな要素に分解することで「診療ロジック」の範疇に含まれると考えられる。診療現場および医療事故防止専門委員会等での医療情報に関する要望調査では、オーダーに伴った薬剤／検査アラートの充実や、これらを組み合わせたクリニカルパスにおけるアラート機能の要望が挙げられている。一方、CP、CWF、CAF はいずれも同質の「診療ロジック」を含むが、仮に計算機上への実装を個別の計算機言語やフレームワークによるプログラムで記述した場合、機能連携の実装に必要な工数が増大し、リファクタ性を含む保守性が悪化する。診療ロジック共通化についての考察は医療情報学では決して目新しいものではないが、実装アーキテクチャについては医療情報システムの部門専業化や高度化に伴い、ベンダーに負う比率が高く、共通化を通じた CL 永続化に対する課題解決が求められている。

近年、HL7 協会[2]、NeXEHRS 研究会[3]等で日本版の標準規格策定が進められている HL7 FHIR は、これまで蓄積されてきた診療情報ラベル運用規則を踏襲しながら、トランザクションにおけるツリー型データ構造を JavaSctipt Object Notation (以下 JSON)構造にて記述でき、2 次元テーブル型構造とツリー型構造の両方を取り扱える特徴を具備するに至っている。プログラミング言語自身の課題としては、オブジェクト指向言語(Object-Oriented Programming, 以下 OOP)における基底クラスの肥大化抑制とリファクタ性の向上が必要とされている。同様に、データ間の関係性を記述する SQL 言語で記述される RDBMS についても、正規化と相反するリファクタ性が存在し、実行速度とデータの直交性を重視した結果高度な正規化によってリファクタ性を減少させていると考えられる。

2. 目的

医療ロジックの実装が永続性を失う一因として、計算機上でハードウェアに依存する環境設定の継承困難性が挙げら

れる。近年登場した仮想化技術は計算機における基盤システム自身の継承性の向上に貢献している。本研究は前述の技術群について、CL の実装上の永続化要請を充足しうるアーキテクチャを検討することを目的とする。また設計したアーキテクチャをもとに実装するリファクタ性の検証ならびに CP、CWF、CAF、CDSS が必要とする条件を充足するかを検証することを目的とする。

3. 方法

方法を以下の手順に示す。

3.1 実装に用いる構成要素の検討

システムを構成する要素数が少ないほど、機構を理解する労力が削減され、保守性と永続性に貢献する。さらに、システム構成要素により、診療現場で求められる高いリファクタ性を充足させる必要がある。上述の観点より実装に用いる言語、プログラムの実行形式ならびにフレームワークは可能な限り少ない数であるべきと考え検討を行った。

プログラムの実行形式においては、コンパイラ型とスクリプト型が存在する(図 1)。コンパイラ型で作成された実行プログラムはハードウェアや OS の特定の版に依存するため、スクリプト言語が優先すると考えられる。スクリプト言語のうち、Web 型のサーバー/クライアント運用を行う場合、サーバサイドとクライアントサイドで用いる言語について検討を行なった。言語比較には PHP、Java および Node.js(JavaScript)[4]を検討した。サーバー言語(例:PHP/JavaScript や Java/JavaScript)に対し、Node.js の拡張された JavaScript 文法でハードウェアアクセス等を可能とする手法は、サーバー/クライアント運用における使用言語数の削減に寄与すると考えられる。Node.js におけるプログラミング言語においては、2015 年に策定された ECMAScript2015 より、プロトタイプベースオブジェクト指向(Prototype-based Object-Oriented Programming, 以下 P-OOP)が導入されている。P-OOP の「データ実体が定義となる」仕様はデータフロー型プログラミング(Data Flow Programming、以下 DFP)の基盤として、リファクタ性ならびに並走するワークフローの記述に適していると推測される。フレームワークについては、フロープログラミングを提供する Node-RED[5]の実装に合わせ Express を採用した。

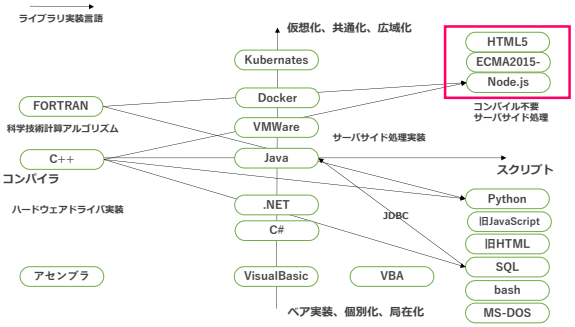

図 1　IPCI におけるプログラミング言語の選定

3.2 アーキテクチャ設計

医療情報システムのライフサイクルを維持継続的なものとするには、物理的に劣化し、五年を超える保守が困難となっているハードウェアの交換に際して移行工数が少なくなるよう設計に取り入れる必要がある。ハードウェア自身の仮想化技術として、VMWare、VirtualBox、Hyper-V などが存在し、Linux ベースでの仮想化コンテナ技術として Docker が存在する。図 2 にサーバーサービスにおけるモダン化箇所について示す。今後、医療情報システムの各サービスが要素化する、すなわちマイクロサービス化するに伴い、有償の仮想化技術の場合、維持コストが継承の限界を定める要因となるリスクが生じる。そこで本研究では継承性を重視する目的で、Docker を採用し、Docker 上 Linux 環境にてデータベースを運用する方針とした。Docker には FHIR サーバ機能を有するデータベースのプログラム箇所を搭載し、Node.js は OS 上に実装しても JavaScript ライブラリの永続性に支障がないこと、Docker ではネットワークポートの割り当て規則が追加され保守複雑化を避けることとした。

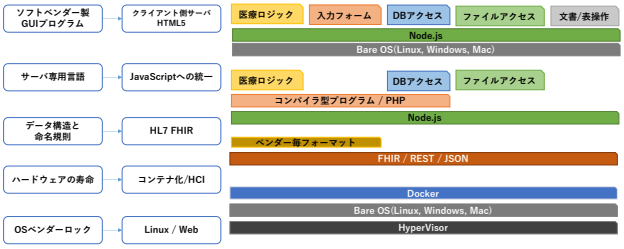

図 2　サーバ構成要素における Web/仮想化

3.3 実装

図 3 に IPCI とにおける医療ロジックの独立化について示す。Application1、Application2 は院内における医療情報のソフトウェアを模式的に表す。データベースアクセスを擁するアプリケーションでは医療ロジックをプログラム内に有する。IPCI はアプリケーションにおいて医療ロジックを参照する箇所を REST API として非同期アクセスにより与条件における当該状況から医療ロジックの演算結果を戻り値として出力する。IPCI 自身はユーザーからの入力ダイアログや結果出力を行えるよう HTML5 におけるユーザーインターフェースを擁する。また IPCI は独立したデータベースアクセス機能を有し、複数の医療データベースにアクセスする。

IPCI コンセプトにより設計したアーキテクチャに基づき、AMD Opteron 16 コア、128GB メモリ、512GB SDD 上に IPCI サーバを構築し、本院における電子カルテシステム(NEC MegaOakHR)への ODBC アクセスならびに院内 Web サーバとしての表示を行うことができた。また院内ワークフローにおいては「電子カルテと業務革新」[6]に記載された UML 対応

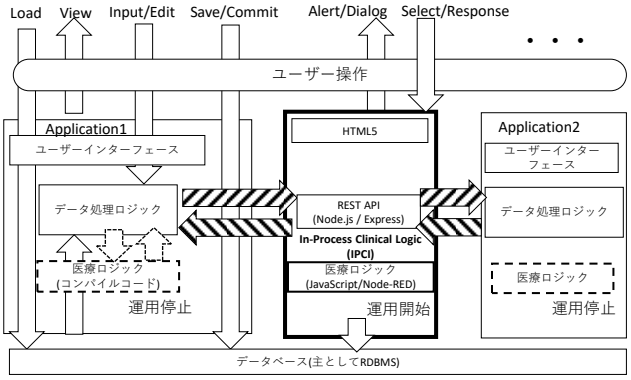

図 3　IPCI による医療ロジックの独立化

図に相当するフローロジックのすべてを Node-RED によって作成し、院内ワークフローの表現が Node-RED にて実装できることを確認した。さらに、e-Path における Outcome、Assessment、Task に相当するオブジェクト層を Node-RED におけるフロープログラミングならびに FHIR フォーマットにて記述できることを確認した。

4. 考察

IPCI アーキテクチャにおいて医療ロジックを記述する Node-RED は IoT ならびに REST API 要素をノード要素として組み込むことが可能であり、これら IoT ノードからの入力をトリガーとしたロジック実行を行えるなどの利点を有する。Node-RED は OSS の 1 種であり、またその記述言語は ECMAScript に準拠しているため、医療ロジックを記述した場合の永続性の観点からも利点があると考えられる。課題としては、JavaScript がスクリプト言語であり、コンパイラ言語と比較して実行速度が遅いため、高周波数の CPU を搭載したサーバーおよび広帯域の情報ネットワークによって IPCI サーバの性能が大きく左右されるため、既存システムと比較して情報インフラの増強が必須であることが挙げられる。この傾向はアジャイル性を有するスクリプト型言語である Python においても同様であり、コード資源の再利用性を高める手段として定着しており、院内Web化と同一の対策が今後広く進むものと推測される。

5. 結論

本研究では CP、CWF、CAF を共通に搭載する「医療ロジック」のプラットフォームとして、Linux、Docker, Nose.js/Express、Node-RED を構成要素とする In-Process Clinical Intelligence (IPCI)のアーキテクチャ設計ならびに実装を行った。院内ワークフローの UML 表現ならびに FHIR フォーマットに対応する e-Path3 層構造の実装テストを行い、必要機能を表現できることを確認した。

参考文献

1) https://e-path.jp/
2) http://www.hl7.jp/
3) https://nexehrs.jp/
4) https://nodejs.org/ja/
5) https://nodered.org/
6) 電子カルテと業務革新、飯田修平, 成松亮, 2008.