Embedded Pythonが オブジェクトデータベース/ InterOperability/MLの 真価を引き出す

群馬大学医学部附属病院システム統合センター
鳥飼　幸太

最初の問い

開発しやすく、改造しやすく、保守しやすく、人材を集めやすいシステムにするために、
システム環境のデザインに求められることは何でしょうか？

現在の診療録入力

診療を行う時間よりも、情報を探し、
整理し、記載する時間の方が長い？

病棟から電話呼び出し

病室に向かい、
患者に耳を傾ける

診察・診療を行う

病棟や医局まで歩いて戻る

診療録記録のために、
PCを立ち上げる

電子カルテにログインする

患者IDを入力する

検査記録が出ているかを探す

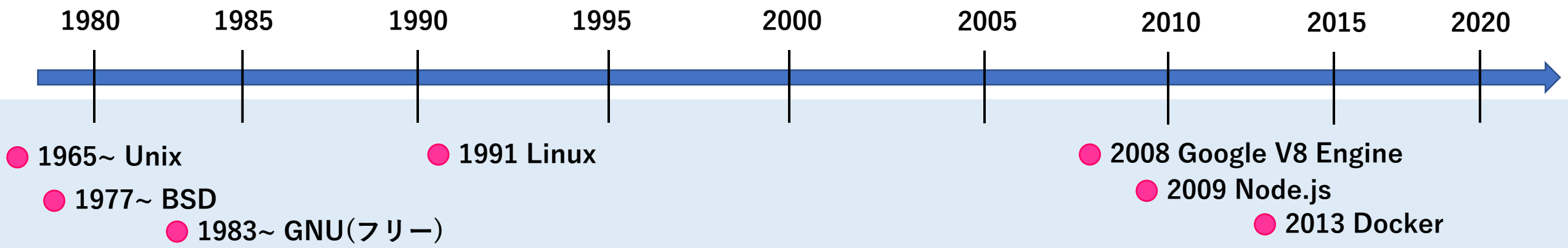
診療録アプリケーション
立ち上げ

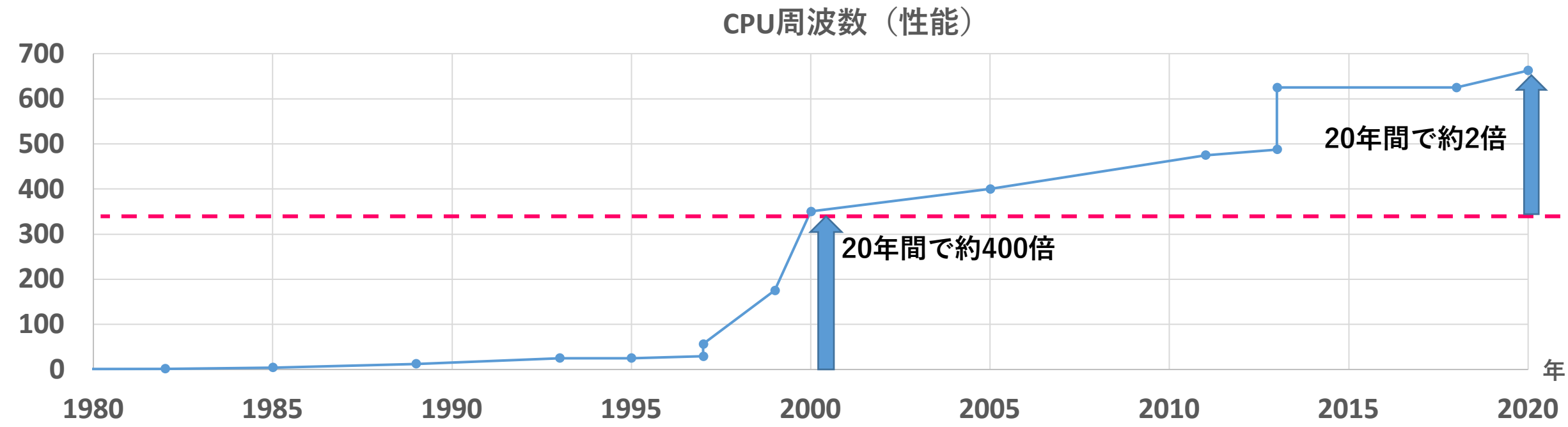
診療録記載

ログオフ

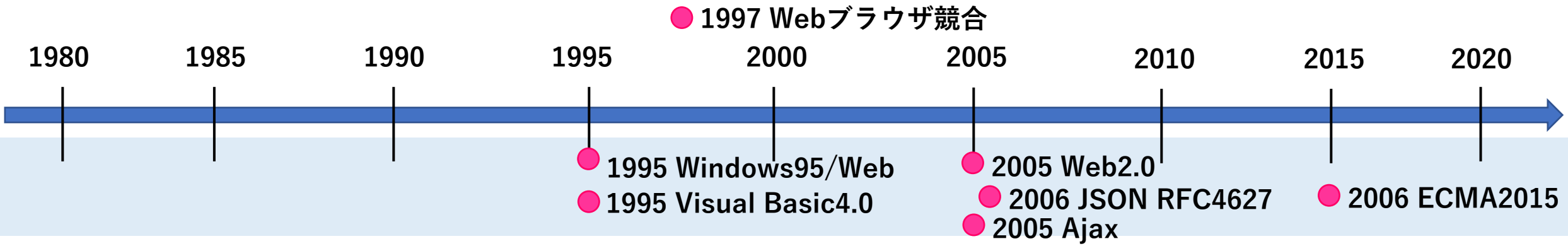
多大な待ち時間
頻回な再入力

標準化と電子化に関する年表(3)





電子カルテ研究開発を通じて、10年考えてきたこと

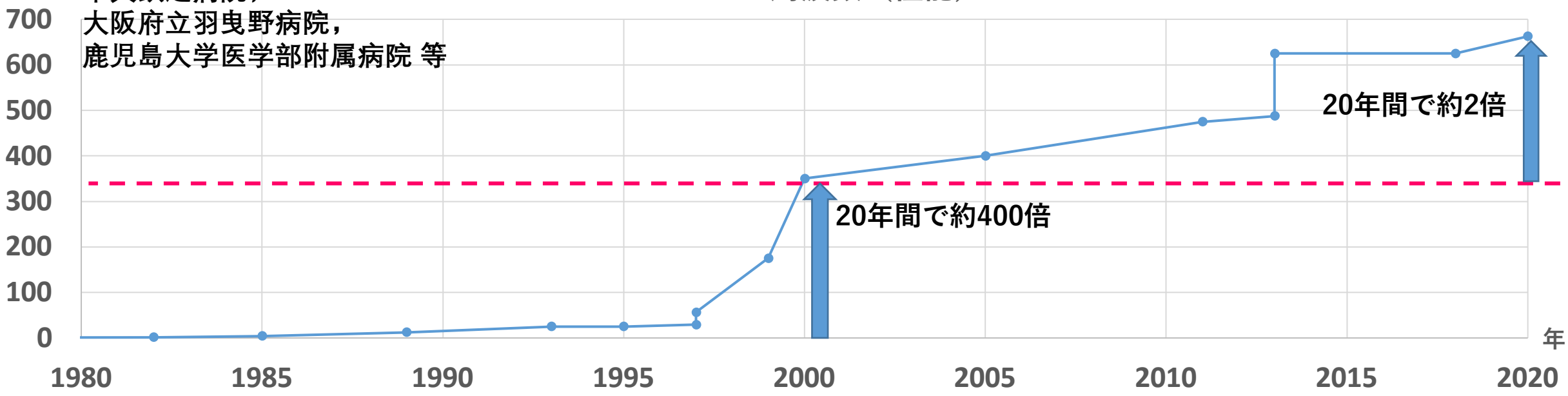


1980 オーダリングシステム
高知医科大学附属病院，
中央鉄道病院，
大阪府立羽曳野病院，
鹿児島大学医学部附属病院 等

1995 電子カルテ
亀田総合病院

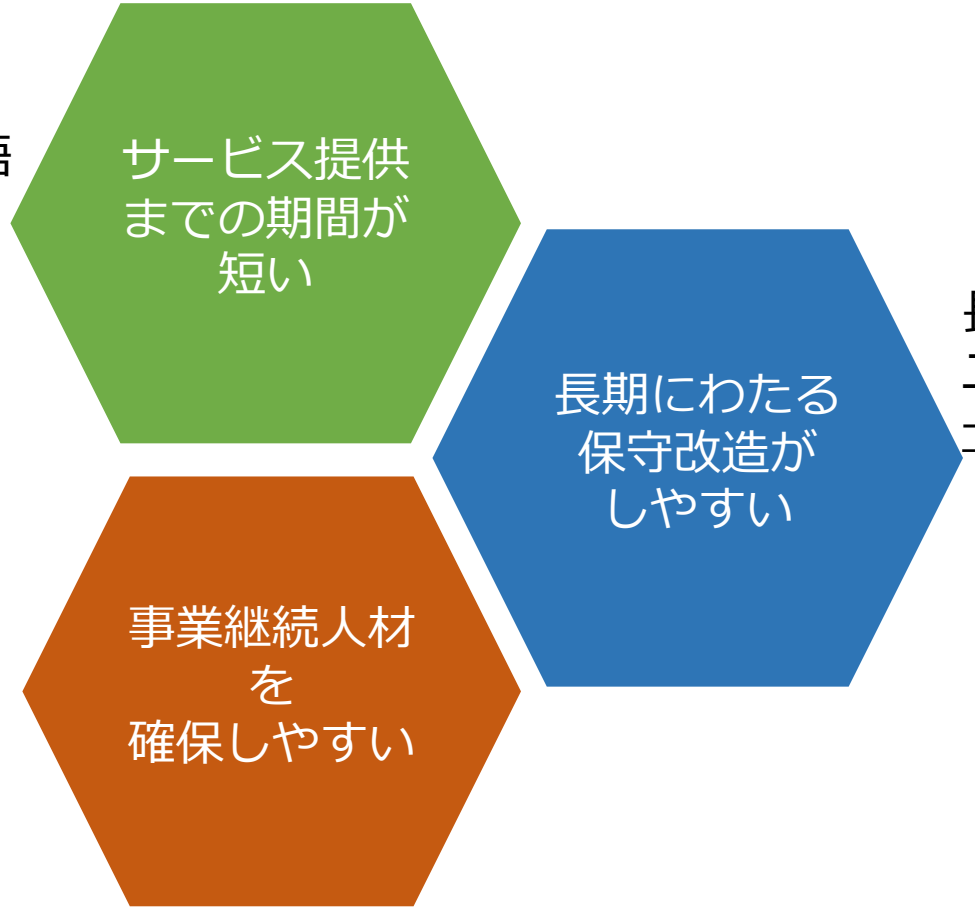
2001 NEC MegaOak NEMR（阪大病院）

2009 群大病院電子カルテ



システム環境に求められる特徴の衝突

ライブラリの充実した言語
新しいサービスに特化した言語
→新規言語？

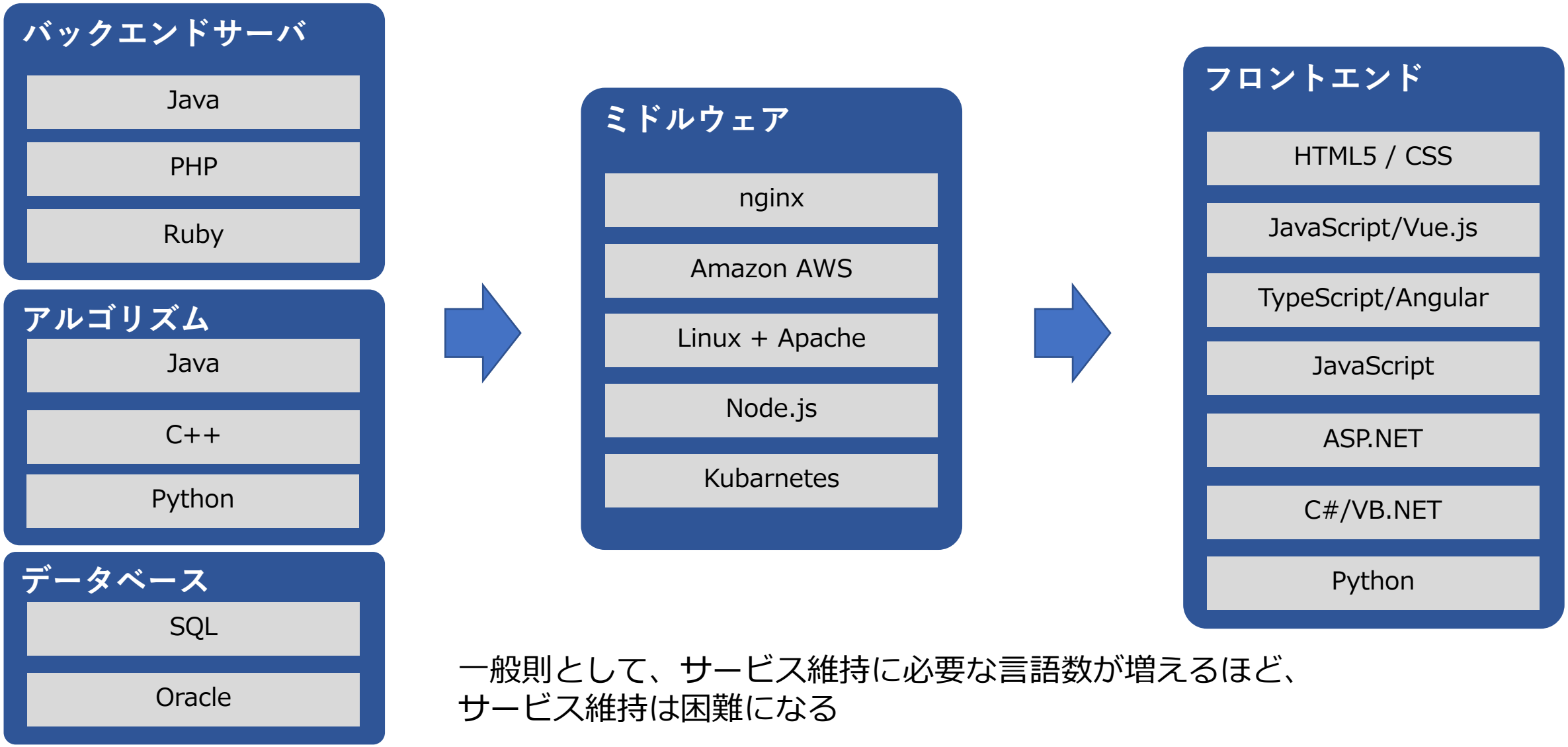


長期的に使われている言語
ユーザー数の多い言語
→古参の言語？

人気のある言語
単純な言語
→独自でない言語？

Web系：バックエンドからフロントエンドまで

- どの箇所に、どのような言語を、幾つ使うかの構成が、アーキテクト設計の要諦



一般則として、サービス維持に必要な言語数が増えるほど、
サービス維持は困難になる

ショートサマリ：データ分析とアルゴリズム記述

- InterSystems IRISは、データ分析とPythonの活用に有用なポイント

データ分析

- 分析目的・仮説の明確化
- データの収集
- データのクレンジング
- データの加工
- データの分析
- 仮説検証と探索的発見

アルゴリズム記述

- 記述に必要な文法が少ない
- オープンソースソフトウェア
- ライブラリの充実
- Git等を通じた版管理
- コード記述の適切な棲み分け
- 分析関連ツールとの連携

個人的な2大言語：ECMAScriptとPython

- InterSystems IRISは、データ分析とPythonの活用に有用なポイント

ECMAScript

- C言語的文法
- 非同期が標準仕様(Node.js)
- プロトタイプオブジェクト指向
- HTML5(UI)との親和性
- 充実し続けるライブラリ
- Web系グルー言語候補

Python

- Fortran言語/C言語に部分的に類似
- 3.X系から言語が安定
- 必要ならオブジェクト指向で構成可
- 機械学習との親和性
- Git等を通じたライブラリの充実
- バックエンド用グルー言語の候補

個人ユーザーがみるInterSystems製品の構成

- 言語を含むさまざまなサービスの集合体

InterSystems IRIS:
基本的にサーバサイド／ミドルウェアサービス

フロントエンドアプリケーション：
ローカル言語（C++、SQL、.NETなど）

APIアクセス言語：C++、.NETテクノロジー、Node.js、Python、SQLなど

データベース機能
DBRMS
ツリー型データベース

データバス機能
InterOperability

ライブラリ
HTTP/TCP
DICOM
HL7 v2 / v3

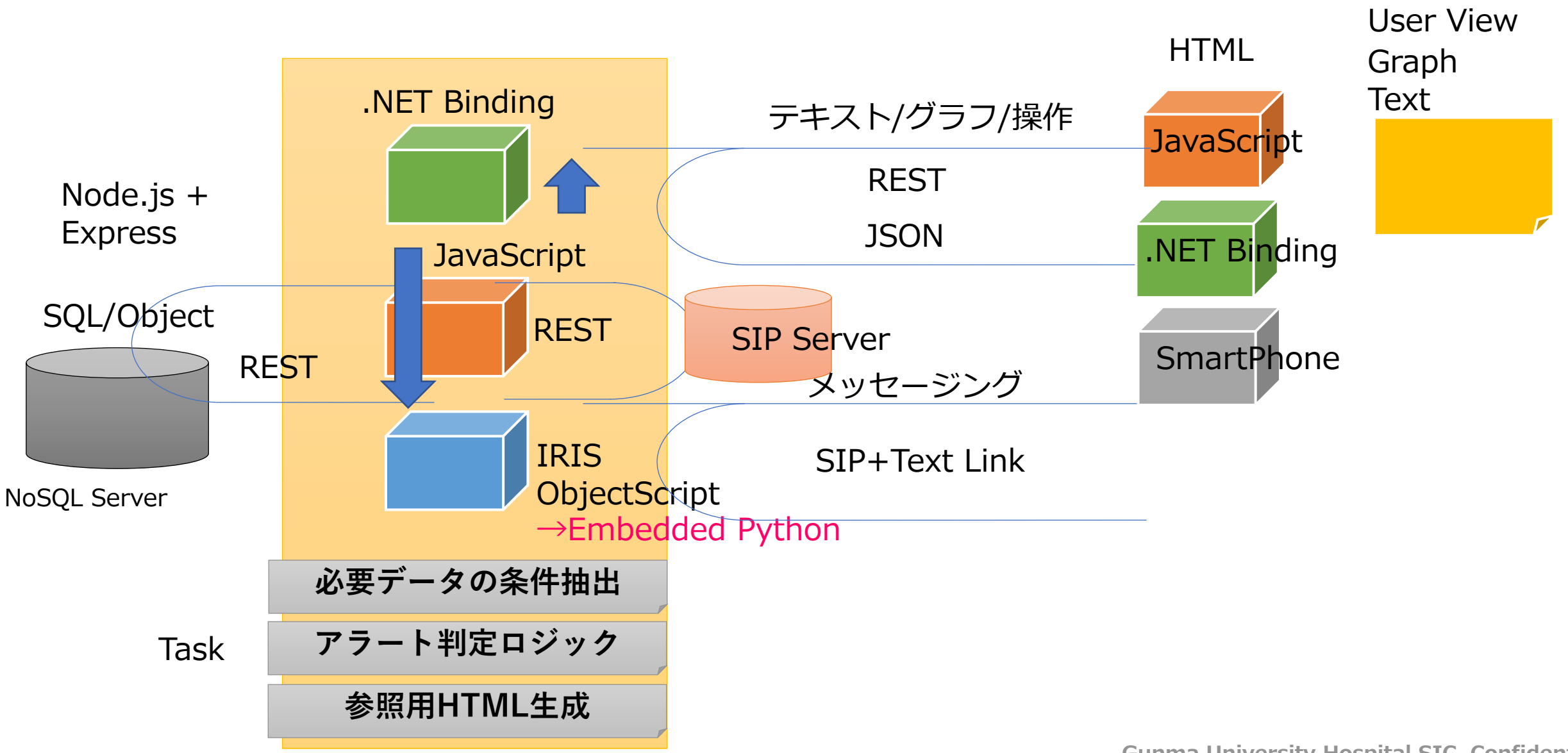
EMR/PHR機能
HealthShare

FHIRサーバ機能
IRIS Health

データベース構築言語：ObjectScript, Embedded Python

物理的実体：B+Treeデータベース、カーネルコードは非公開

サーバ側の中身の話（データ検索、演算、可視化、表示について）



InterSystems製品との関わり方に関する消化不良の話題

医療情報マネージャーの理想：
「作りこんだシステムコードや設定の資産を失うことなく維持発展させたい」

夥しいシステム更新で失敗してきていること：
「アルゴリズムの記述された、ある言語コードを、多言語に移植したくても、翻訳スキルをもち、かつアルゴリズムが理解できるエンジニアは絶滅危惧種」
POSIX思想のように、「あるコードがいつまでも動く」ように構成するならば、それはスクリプト言語であるべきであるし、将来に「翻訳を委託すれば済む」と
絶対に安易に考えてはいけない

UNIX-bashに代わるPOSIX性の言語候補として選定
医療アルゴリズム（Business Process）は
JavaScriptで記述するように設計を決めている

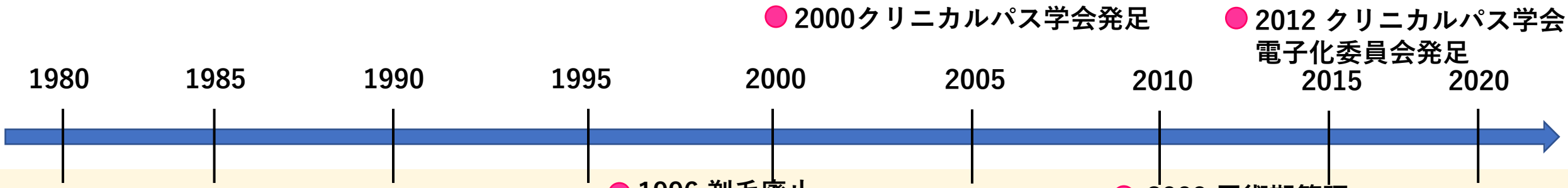
NLP開発機能に関する悩み

実装コード：Python
データの保存先：メモリまたはPythonが読み書きできるファイル
ファイルアクセス速度：低
アルゴリズムの書きやすさ：高

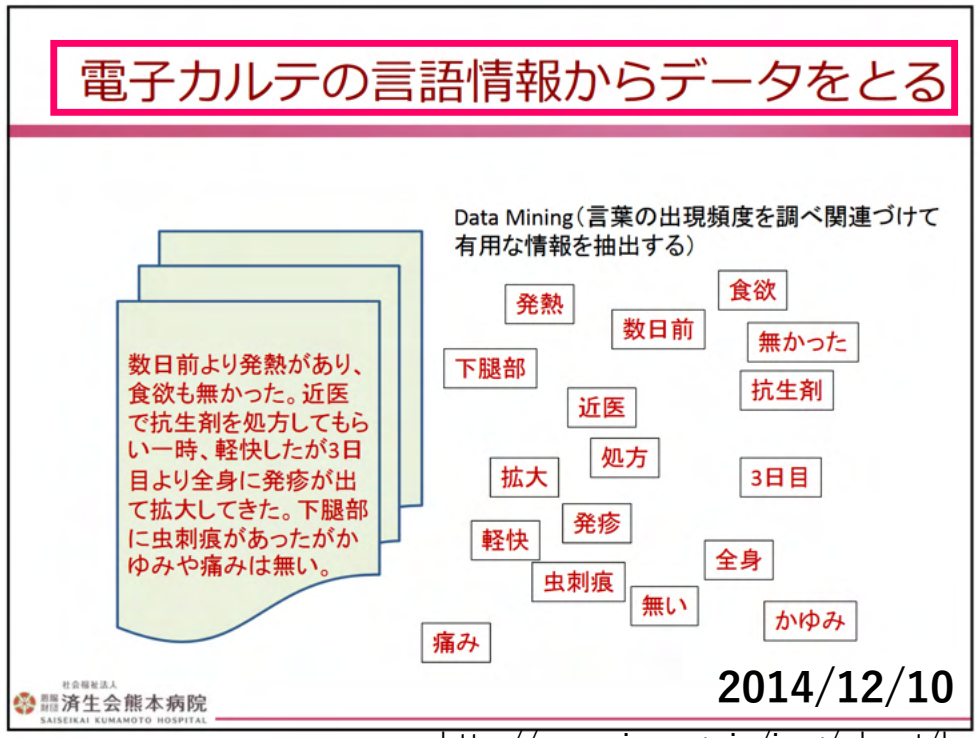
実装コード：ObjectScript
データの保存先：IRISアーキテクチャ（データベース）
ファイルアクセス速度：高
アルゴリズムの書きやすさ：低

悩み：
Python NLPで書き続けると、このまま作りこめば、病院内で運用したい際には「移植」が必要になる：
ObjectScriptでNLPを記述するにはそれなりのスキルが必要

標準化と電子化に関する年表(2)

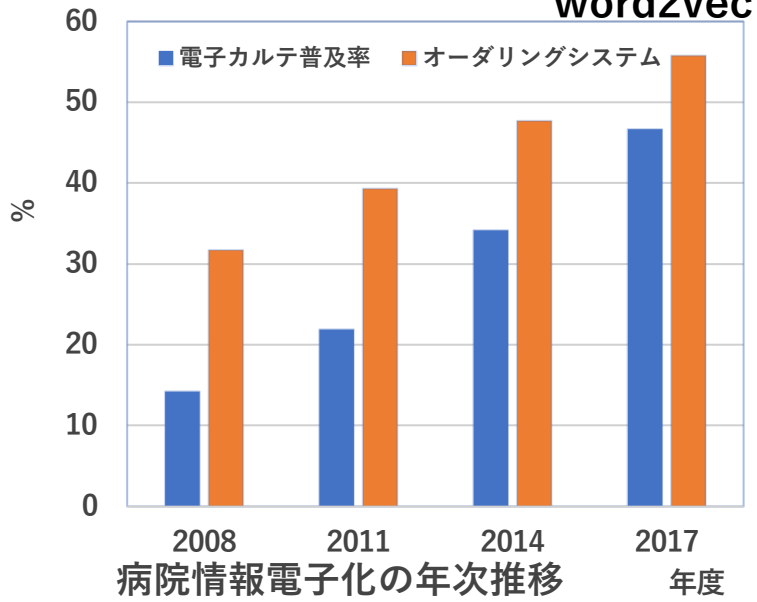


- クリニカルパス運用の成果



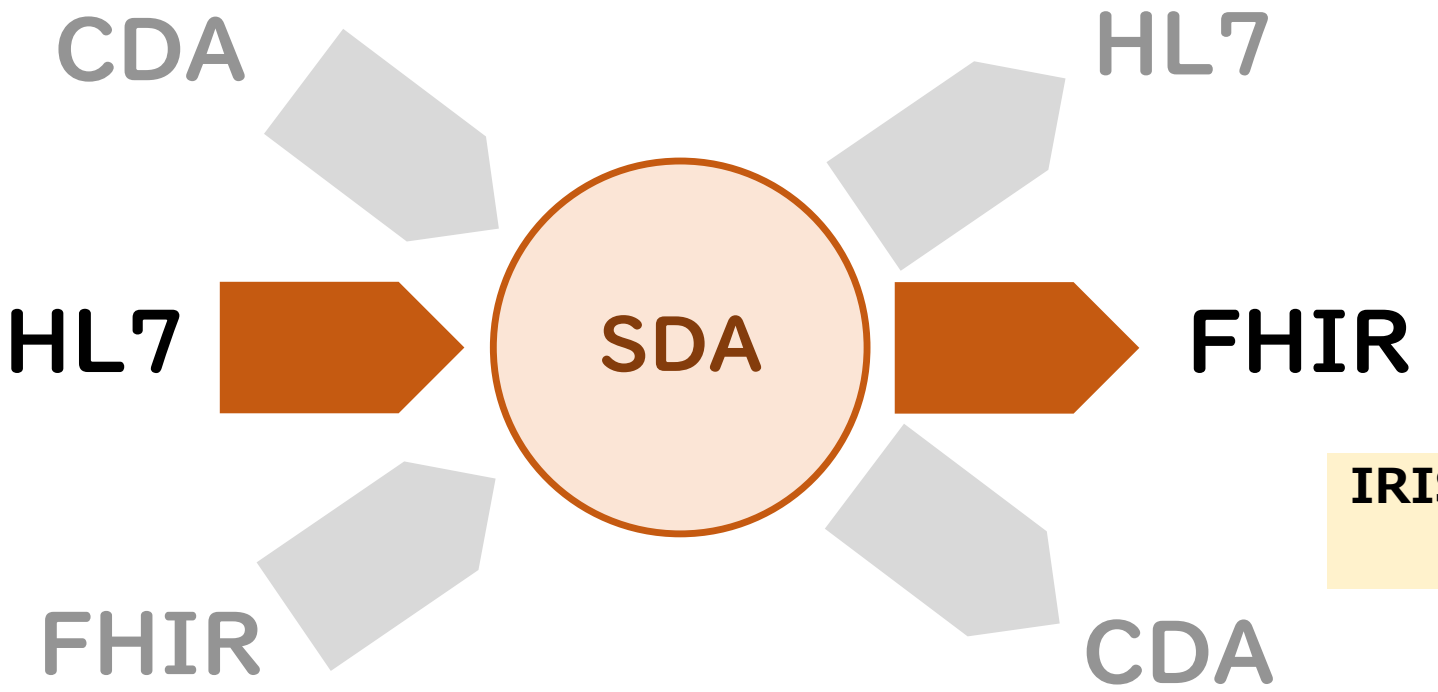
http://www.jscp.gr.jp/img/about/kouen_20141114.pdf

2006 深層学習の発明　2014 自然言語処理 word2vec



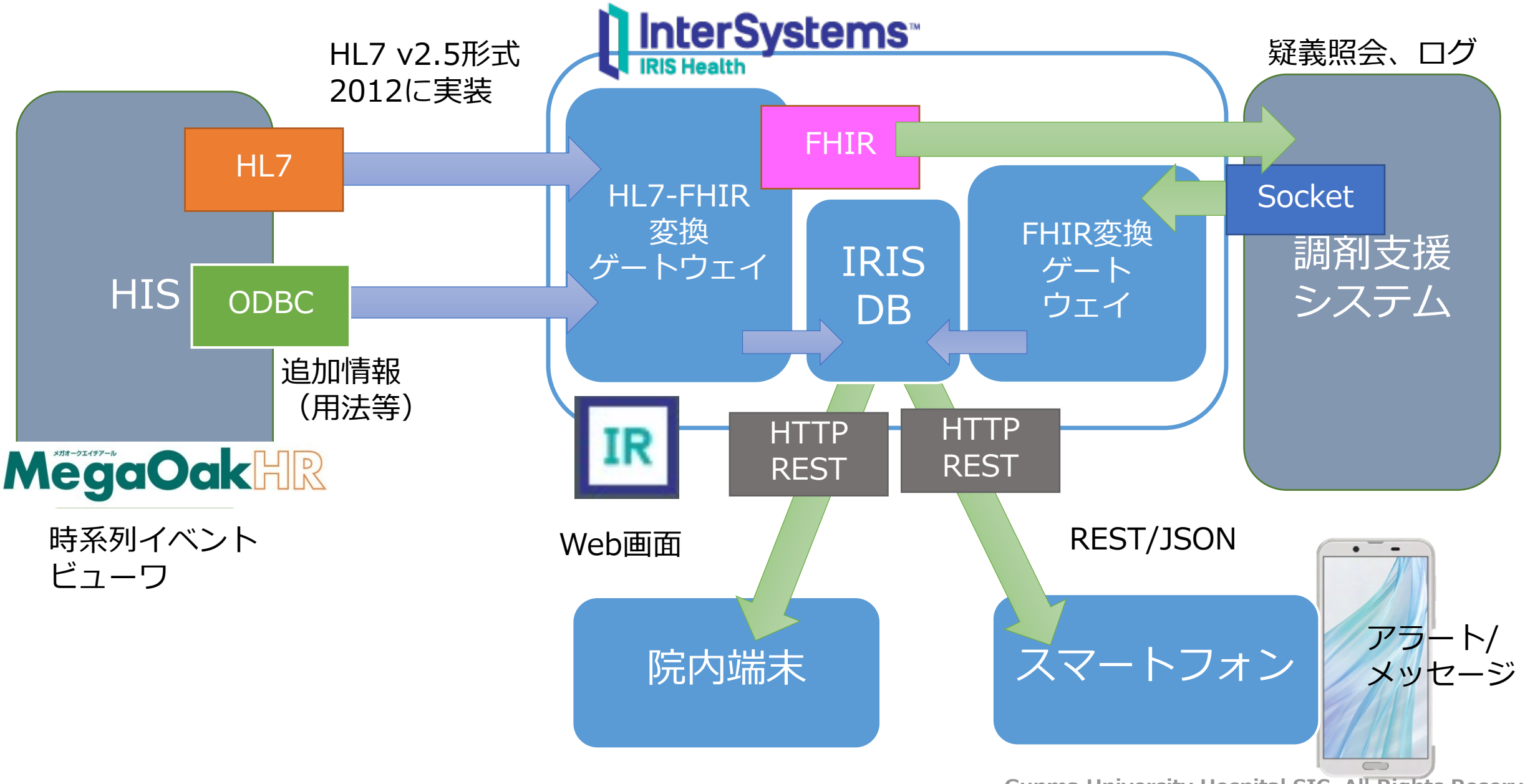
IRISによるデータ変換

- **SDA(インターシステムズ独自のデータ構造)を経由した、HL7からFHIRへのデータ変換**

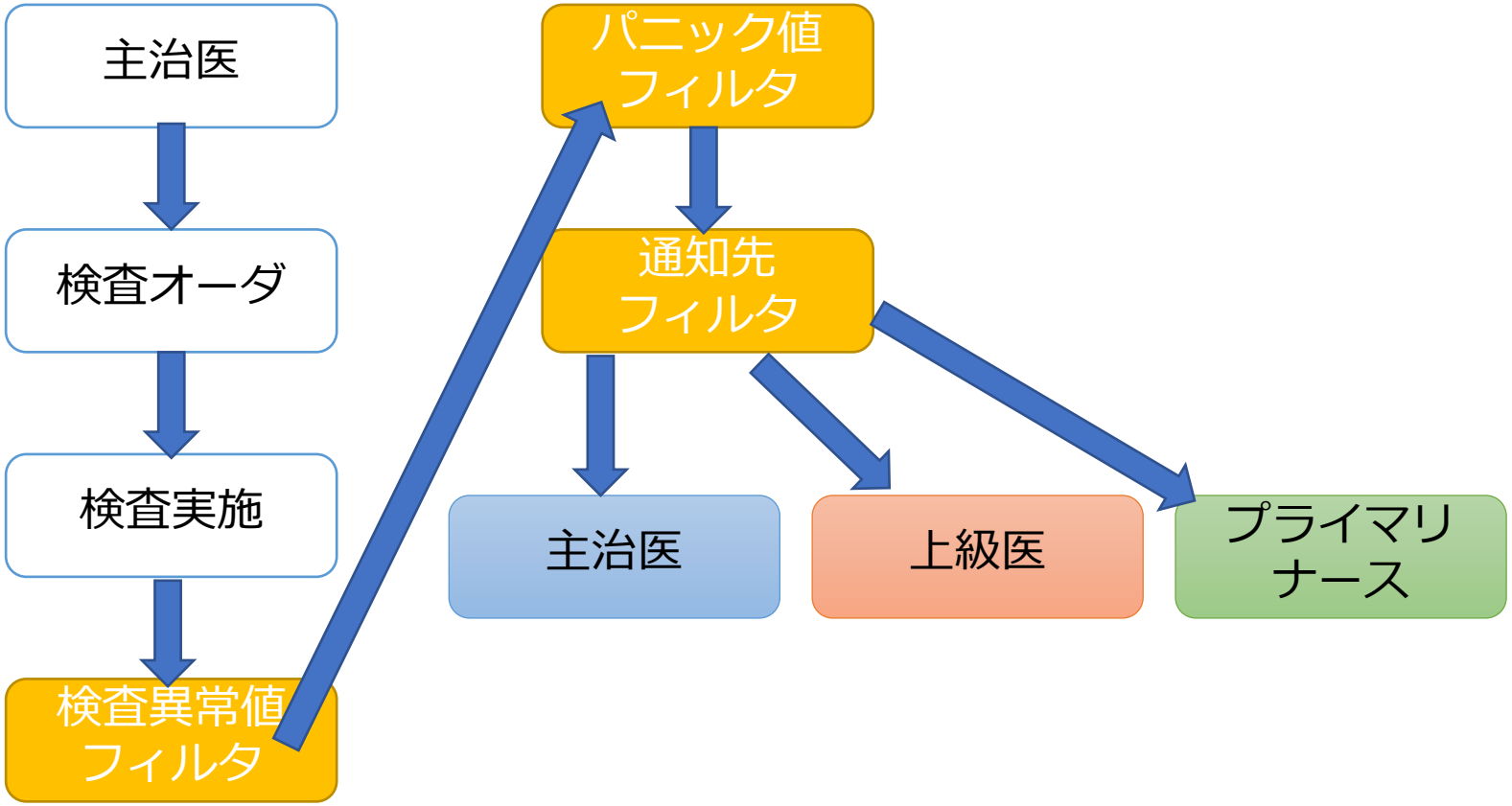


**IRIS for Health 2020.1より
FHIR R4にも対応**

FHIRによる調剤支援システムとの接続図



情報を規則性へ変換する＝フィルタリングすることの重要性



「必要な人に、必要な時に、必要な場所で、必要な内容を、
必要な形式で、必要なデバイスで」

改めて「オブジェクトデータベース」の有用性とは何か

ODB-RDBインピーダンス軽減

アプリケーション構造=JSON
→ツリー型データ構造

DB構造=(およそ)RDB
→2次元テーブル型構造

医療のように時系列データを保存する場合に しばしば困難を生じている

MUMPSは元来MITで電子診療録のために開発された

将来的なFHIR Resourcesの有用な候補

テーブルの継承

例１：検査機器の更新
例２：ガイドラインの変更

同じフィールドに異なる種類のデータを保存しなければならないとき、既存DBでは困難が生じる

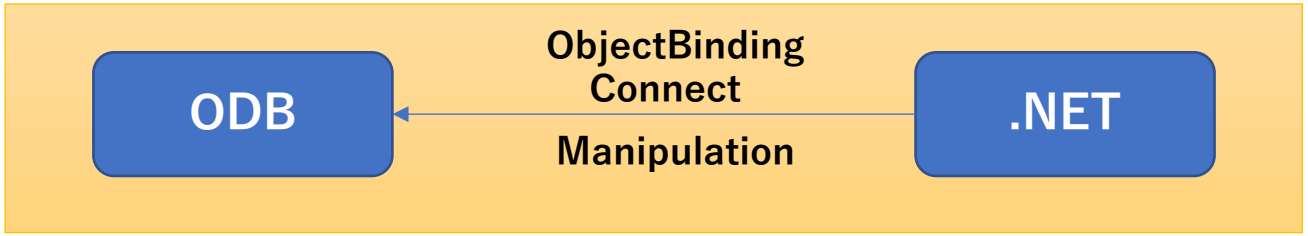
テーブルのポリモーフィズム

例１：検査機器の更新
例２：ガイドラインの変更

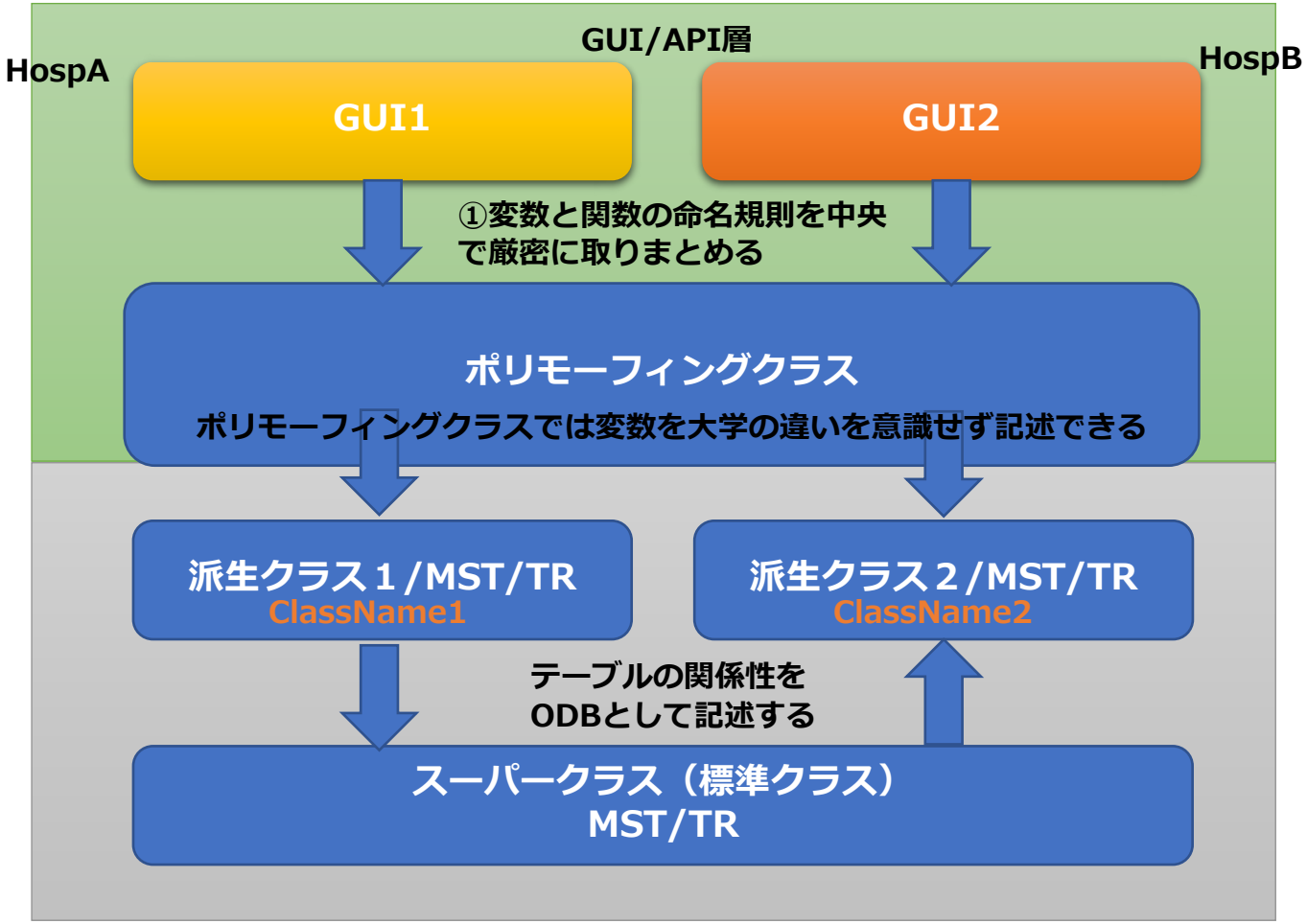
クラスを継承するのと同様に、テーブルを継承する
従前のテーブル利用にコード抵触せず、新しいテーブルの利用を追加できる

例：ある大学で開発された機能を別の大学で実装する場合

GUIから呼ばれるDBクラスを
ClassName2に変更するだけで
動作



大学1
ClassName1



大学2
ClassName2

プロキシクラスを用いた記述　VB.net側

```
        If (exists.HasValue And exists.Value) Then

            Dim contact As User.Main.PersistClassExtension  'プロキシオブジェクトという考え方
            Dim cacheobj As User.Main.PersistClassExtension

            '.NET プロキシ・クラスには、対応する Caché クラスのコレクションの投影が含まれています。このコレクションのタイプ(
            '例えば、Caché のオブジェクトのリストは InterSystems.Data.CacheTypes.ListOfObjects に投影され、Caché の文字列の配
            'これらのコレクションに対しては、.NET の foreach コマンドを使用して、繰り返し処理を実行できます。また、要素を追加

            contact = User.Main.PersistClassExtension.OpenId(cnCache, "32") 'ObjectIDが存在するかどうか

            '既存データの書き込み
            contact.a = 20180114 'データ操作
            contact.b = "InputFromVB.net"
            Dim status As CacheStatus = contact.Save() 'Commitに相当すると考えてよい

            cacheobj = New User.Main.PersistClassExtension(cnCache)
            '新規データのエントリー
            cacheobj.a = 20180415 'データ操作
            cacheobj.b = "NewEntryFromVB.net"
            Dim status2 As CacheStatus = cacheobj.Save() 'Commitに相当すると考えてよい

            Dim stringList As CacheListOfStrings '= cacheobj.Id.Length


            If (status.IsOK <> True) Then
                MsgBox("Error Updating Contact")
            End If
            contact.Close() 'オブジェクトの解放Tryの中で宣言されたオブジェクトはTry構文内のみスコープを持つ
            contact.Dispose() 'プロキシオブジェクトの破棄
        End If
    Catch e As CacheException

    End Try

    Try
```

DBへの書き込み、読み込みをオブジェクトとして記述できる！

→多階層、継承、オーバーロード、
ポリモーフィズムの概念が導入できる！

スーパークラス、派生クラスとObject型の併用によるポリモーフィングの一例
（派生クラスにおける処理の違いに関わらず医療ロジックを記述する方法）

VB.NET側記述（ポリモーフィングクラス）
※医療の上で共通に実施されるワークフロー

```
Imports InterSystems.Data.CacheClient
Imports InterSystems.Data.CacheTypes
1 個の参照
Module Module1
    3 個の参照
    Sub PolyMorphingTest(ByRef o As Object, n As Double) 'Ob

        '接続から記述しているが、必須で
        Dim connectionString As String = "Server = localhost; Port=1972;" +
                "Log File = C:¥VSProject¥NewCarte¥NewCarte¥NewCarte¥bin¥NewCarte2.log; Namespace = USER;" +
                "Password = trki1979; USER ID = torik"
        Dim cnCache As InterSystems.Data.CacheClient.CacheConnection '接続フォーマット
        cnCache = New CacheConnection(connectionString)
        cnCache.Open()

        Dim t As Type = o.GetType 'Objectは常に自分の名前をToStringで表現するが、直接タイプを取り出すこともできる

        'クラスを新規動的生成
        Dim d1 As Object = t.InvokeMember(Nothing,
            System.Reflection.BindingFlags.CreateInstance,
            Nothing, Nothing, New Object() {cnCache}) ' 呼ばれたタイプに合わせて動的に生成するクラスを変更する

        'PolyMorphingで論理演算を記述する
        'PreScriptionElementOver1 ならオーバーライド（大学カスタマイズ1）
        'PreScriptionElementOver2 ならオーバーライド（大学カスタマイズ2）
        'その他ならBaseObjectの関数が呼ばれる（標準版MegaOak）
        '開発フェーズでは派生クラスで製作し、標準版に実装するときにはBaseObjectでもよい
        Dim b As Double = d1.CalcBMIBase(n) '関数名を揃えておく
        MsgBox(o.GetType.ToString & " BMI=" & b.ToString())
        d1.Save() '異なるテーブルに保存されることを確認する Cache'側でリレーションシップを構成しておけば、データを共通に取り出せる

        '----------
        'Dim d1 As Object 'Overloadすべての
```

VB.NET側記述（GUI/API層）

```
'アプリケーション層の記述
'アプリケーション層では、データベースの存在を気にせず記述する
1 個の参照
Sub PolyMorphingTest2()

    Dim d1 As New User.Main.PreScriptionElementOver1() '
    Dim d2 As New User.Main.PrescriptionElementOver2() '
    Dim d3 As New User.Main.PrescriptionElement()

    '医療ワークフローの記述
    'ここがPolymorphing：同じ関数を使っている
    'BMIが施設ごとに異なる場合
    PolyMorphingTest(d1, 30) '実際にはどれかが記述されている
    PolyMorphingTest(d2, 30)
    PolyMorphingTest(d3, 30)
End Sub
```

標準と異なる振る舞いに対して
派生クラスを定義する

呼び出す関数名、引数は完全に同一にできる
→カスタマイズでの変更が最小化される

Object型は異なるクラスのインスタンスを受け入れることができる

クラスインスタンスは自分自身のクラス名を常に保持している
「動的型決定」の実例

派生クラスで呼び出し関数名を統一しておく
またはスーパークラスの関数をオーバーライドする
（GUI設計時に、関数名や変数名が他モジュールと重複していないかをあらかじめチェックする作業が必要）
→全大学マスタの中央管理の必然性を示す

```
/// PrescriptionElememtのオーバーロード
Class User.Main.PreScriptionElementOver1 Extends User.Main.PrescriptionElement
{

Property bmi As %Numeric;

// 派生クラスにおける関数のオーバーロードの例

Method CalcBMIBase(age As %Numeric) As %Numeric [ Language = basic ]
 {
    if age < 10 then
        Return 0.9
    elseif age < 50 then
        Return 1.8
    end if
 }
```

bmiはスーパークラスに記述可

大学AでのBMIの定義

```
/// 別の大学病院のマスタ実装例
Class User.Main.PrescriptionElementOver2 Extends User.Main.PrescriptionElement
{

Property bmi As %Numeric;

// 派生クラスにおける関数のオーバーロードの例

Method CalcBMIBase(age As %Numeric) As %Numeric [ Language = basic ]
 {
    if age < 10 then
        Return 2.5
    elseif age < 45 then
        Return 3.8
    end if
 }
```

ODB側記述

大学BでのBMIの定義

SetとGetによる、大学間の違いを吸収する変数ポリモーフィング実装の一例

VB.NET側記述

```
Imports InterSystems.Data.CacheClient
1 個の参照
Public Class Form5
    0 個の参照
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        '動的生成準備２－２Cache'DBを呼ぶ
        Dim connectionString As String = "Server = localhost; Port=1972;" +
                "Log File = C:¥VSProject¥NewCarte¥NewCarte¥NewCarte¥bin¥NewCarte2.log; Namespace = USER;" +
                "Password = trki1979; USER ID = torik"
        Dim cnCache As InterSystems.Data.CacheClient.CacheConnection '明示的な型宣言
        cnCache = New CacheConnection(connectionString)
        cnCache.Open()

        Dim d As New User.Main.BasicPatientInformationHospA(cnCache)
        d.testValueA = "15.36" 'この数字が基底クラスであるtestValueに反映されることを示す
        d.conversion = 0.89
        Dim s As String
        s = d.testValue
        MsgBox("testValue = " & s)
        'd.Save()
        d.Close()
        d = User.Main.BasicPatientInformationHospA.OpenId(cnCache, "65013")
        d.conversion = 0.89
        d.Close()
        d.Dispose()
    End Sub
End Class
```

SQLが全く記述されていないことに注目

ポリモーフィング上ではある変数に値を代入する以上の意識はしない
（変数testValueAは変数testValueとして完全にオーバーライドしてもよい）
→他大学への展開の際にGUI層のカスタマイズを必要としない

```
/// MegaOakHRの患者基本情報
Class User.Main.BasicPatientInformation Extends User.Main.BaseObject
{

Property PATIENTNO As %String;

Property HBVChecked As %String;

Property HBVCheckedReferLastDate As %String;

Property PatientHeight As %String;

Property PatientWeight As %String;

Property AllergyNotification As %String;

Property testValue As %String;
```

スーパークラスに標準的な値を保持する
→他大学では標準値または派生クラスでの変換のいずれかで実装する

User.Main.BasicPatientInformation.cls + | User.Main.BasicPatientInformationHospA.cls +

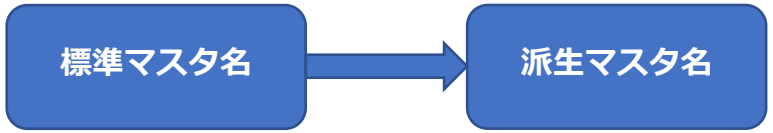
```
/// 患者基本情報の派生クラスA
Class User.Main.BasicPatientInformationHospA Extends User.Main.BasicPatientInformation
{

Property testValueA As %String;

Property conversion As %String;

Method testValueASet(value As %String) As %Status
 {
     //set d = ##class(User.Main.BasicPatientInformationHospA).OpenId($this) //自分自身が呼べるか？
     set ..conversion = "0.74"
     set i%testValueA = value
     set a = value * ..conversion
     set ..testValue = a
     //set status = d.%Save()
     quit $$$OK
 }

Method testValueAGet() As %String
 {
     if i%testValueA = "" {
         quit ..testValue / ..conversion
     }
     else{
         quit i%testValueA
     }
 }
```

ODB側記述

派生クラスでの扱いが異なる場合は
スーパークラスへの変換を記述する
（GUI層でのカスタマイズで吸収する必要がない）

ポリモーフィズム運用によるマスタ、関数の局所的カスタマイズ

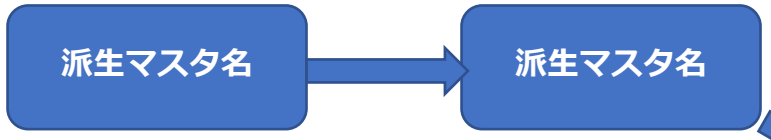
カスタマイズが必要な病院への導入を行う場合



新規に派生マスタを定義

間違いなくカスタマイズされない事例以外は
できるだけ標準マスタを「そのまま」使用しない
全病院間で派生マスタ名は一意になるように運用する
→全国のMegaOak間で一意の名前運用を中央管理する
→地域GUI/APIエンジニアの工数の大幅な削減
→より高い利益率・サービス速度向上を実現

フィールドを追加したい場合



変更なし

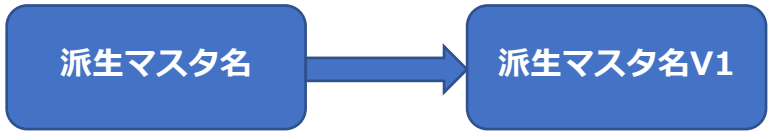
Propertyの追加、Set/Getロジックの変更
関数オーバーライドで対応
（Oracleのような再構成は不要）
API/GUI層なら関数オーバーロードでも対応可

四半期ごとの見直しなど

標準クラスにもPropertyと
関数を追加
→迅速な標準化、機能追加
→市場競争力を高める

大幅な改変を伴う場合

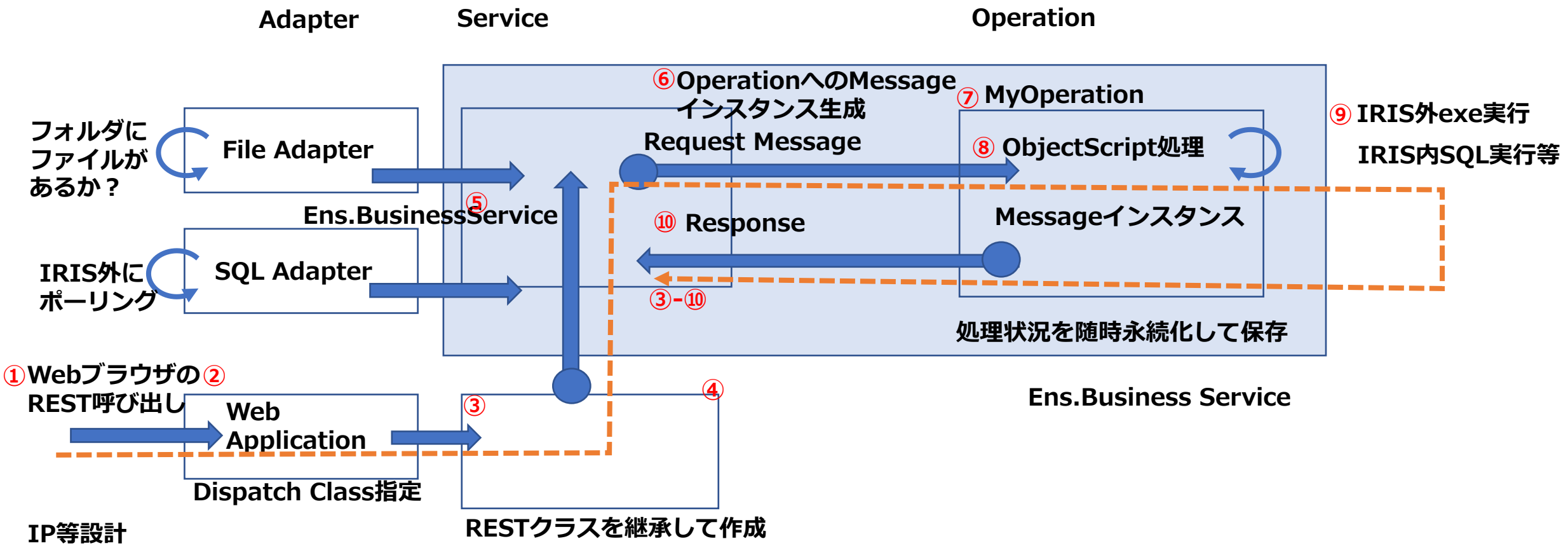
（検査装置の入れ替えに伴うフォーマットの大幅な変更などを例として）



派生クラス生成でバージョン
アップを行う
変数の変換規則を吸収する

GUI/API層でのクラスインスタンス宣言のみ変更
（インスタンスが有効な期間をチェックして動作する
ようにコーディングしておく必要がある）
→テーブルの再定義が不要、カスタマイズに伴う
サービス停止時間の大幅な減少
→ユーザーサービスの大幅な改善

InterOperability詳説

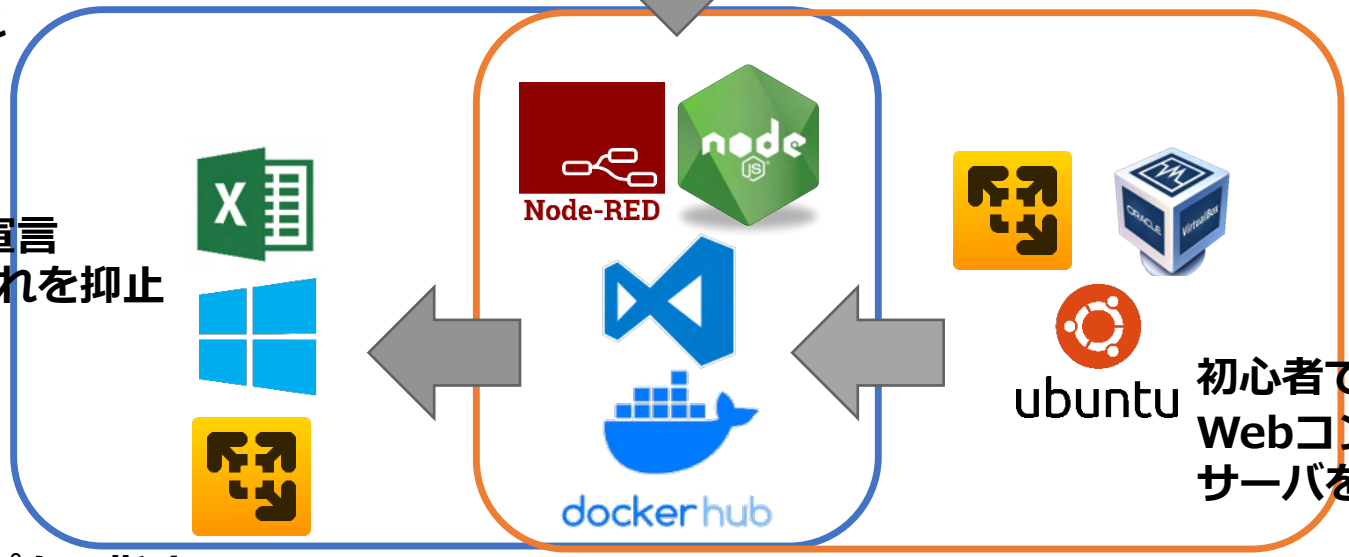


考察：永続化対象と利用ユーザ維持

Androidは現時点では必ずしも成功した
開発プラットフォームとは言えない

Linux上で「ツールとして」Windowsを使う例はまだ少ないが、永続化視点では意味がある
LTS以降の「載せ替え」が課題→dockerまで抽象化？

データよりもExcel関数/UI/VBA資産がWindowsで大きな位置を占めている？
→仮想化して永続化の流れ



永続的なバージョンの宣言
OS変更時のユーザー離れを抑止

初心者でも理解しやすいGUIの獲得
Webコンピューティングでは多数のサーバを必要とする→ライセンス費問題

Windowsは常に
端末内でのワンストップを目指す
仮想化の脅威

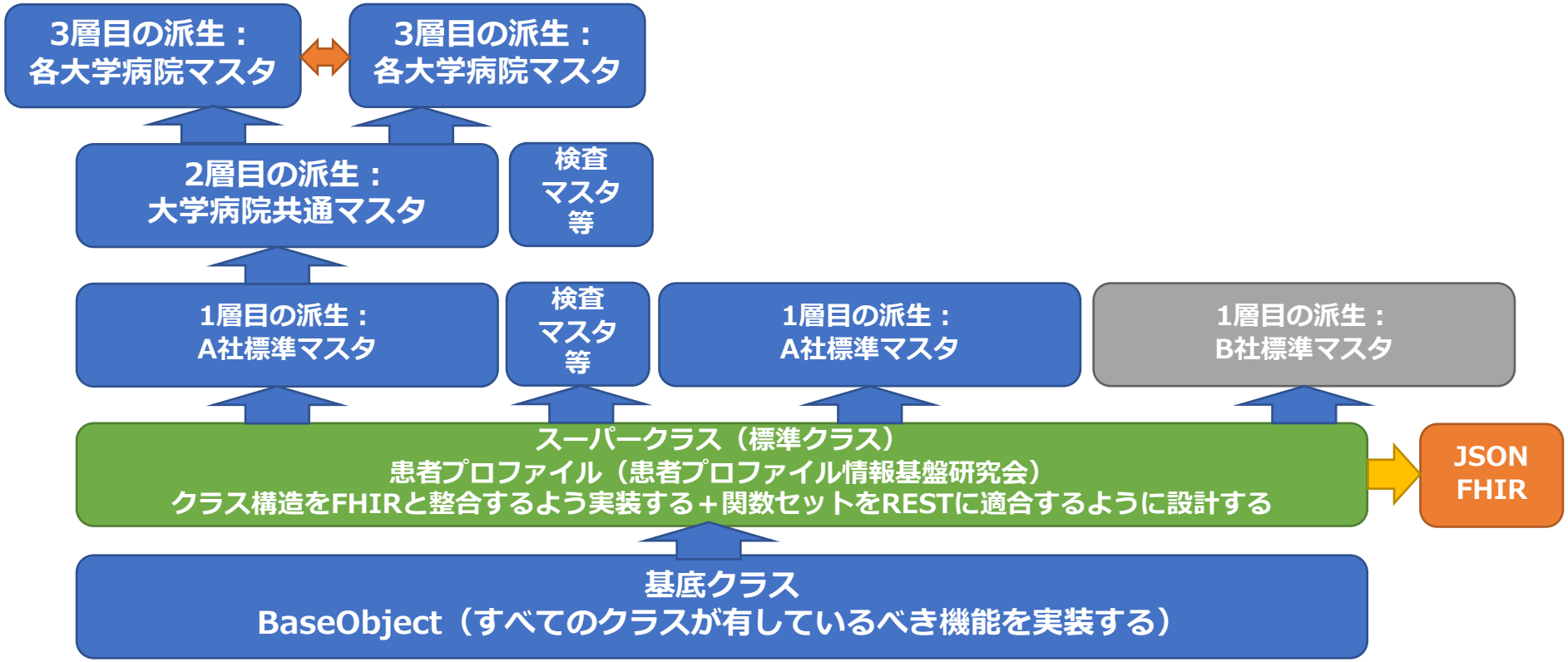
pip/npmなどのコマンドインストーラが
再び浸透してきた

Macはハードウェアと一体の
デバイスとして領域を拡充
「使いやすいUnix」または
「ツール」または
「HTMLアプリの土台」

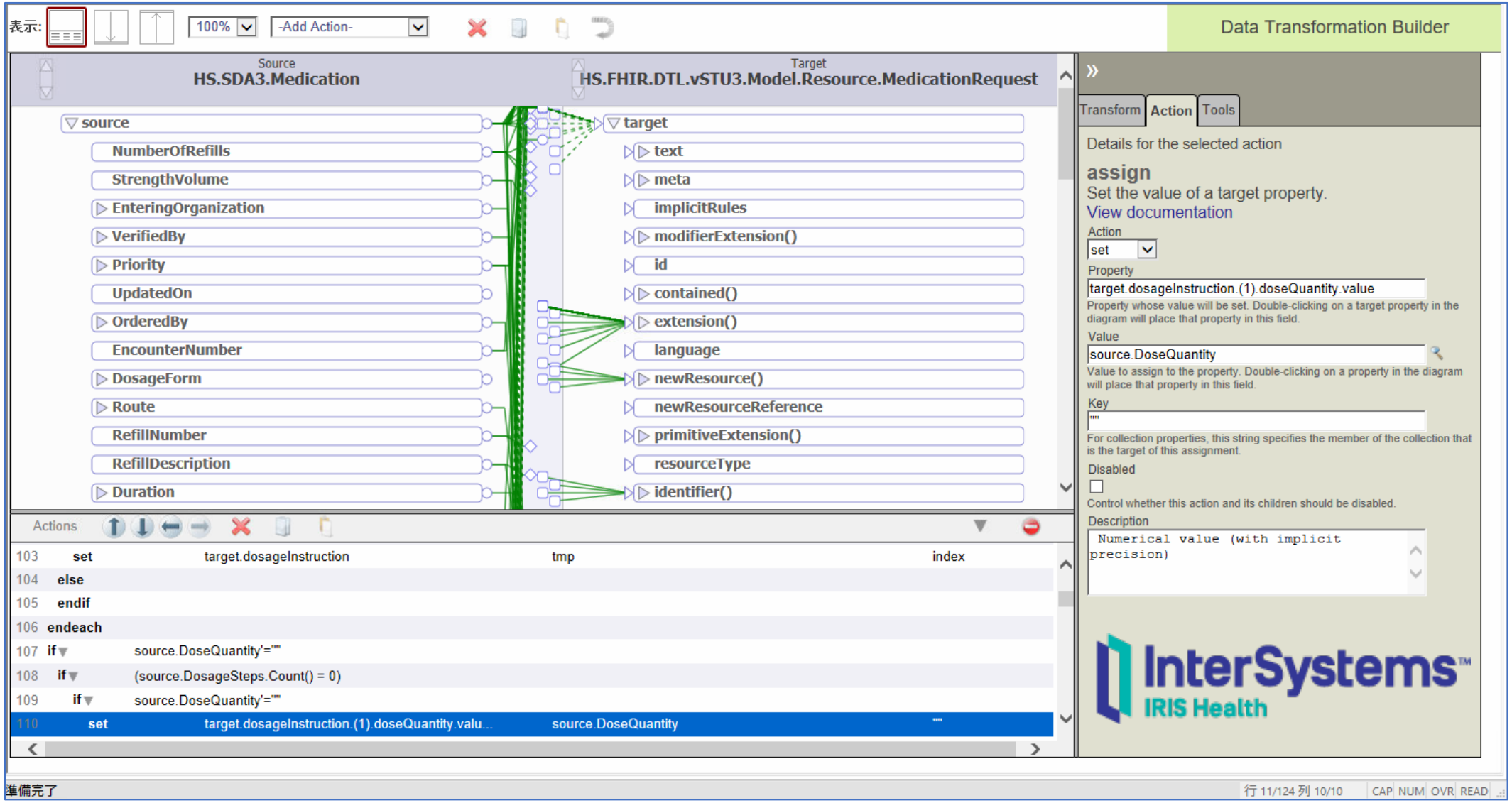
ECMAScriptはHTML/OSSと相まって
「次世代POSIX」の地位を確立

クラウドによる「囲い込み」
スケーリング用途での拡大
医療情報に適するか要検討

デザイン検討：患者基本情報（患者プロファイル）におけるODB設計と実装例

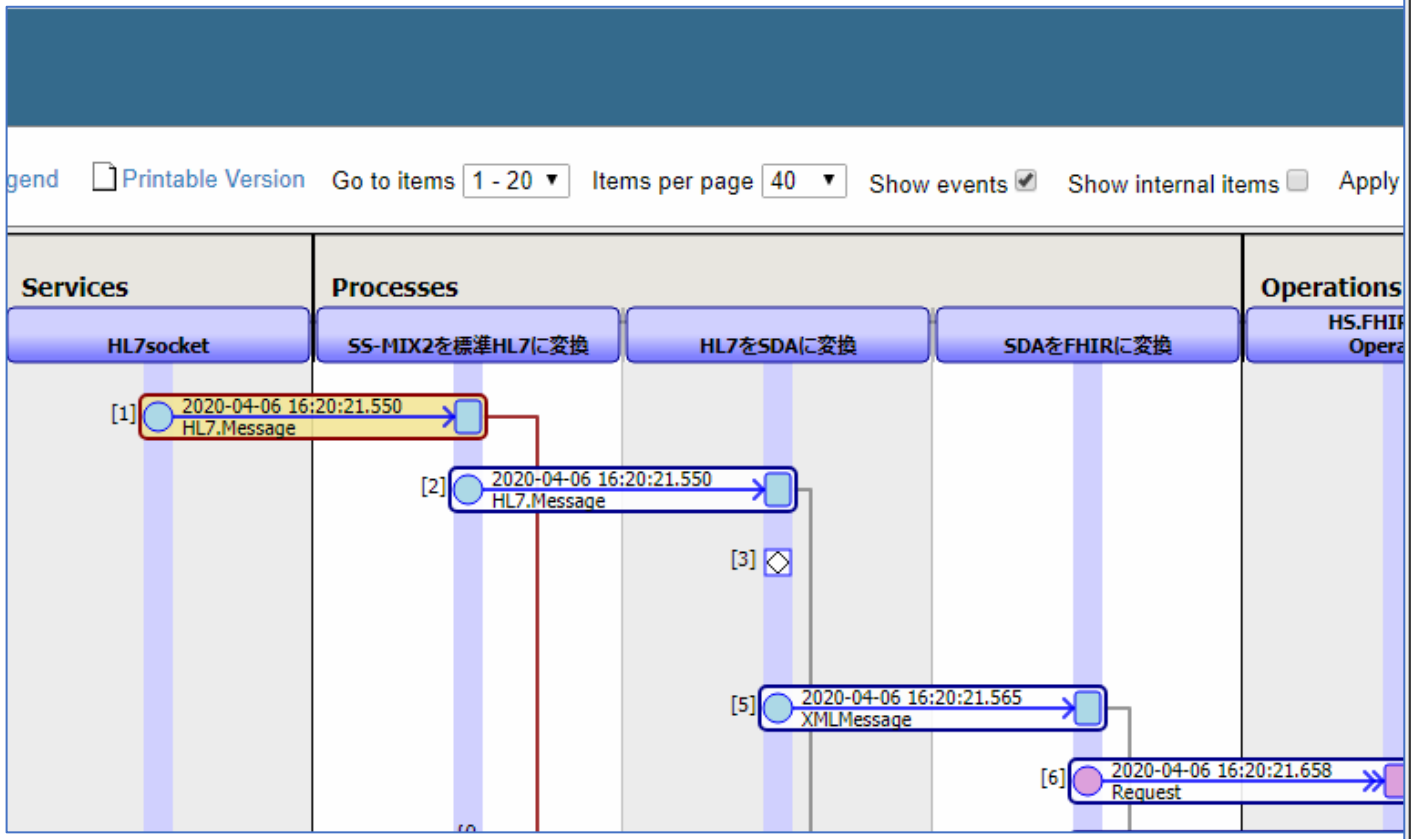


データ変換画面(SDA⇒FHIR)



HL7

InterSystems IRIS Health



Header | Body | Contents

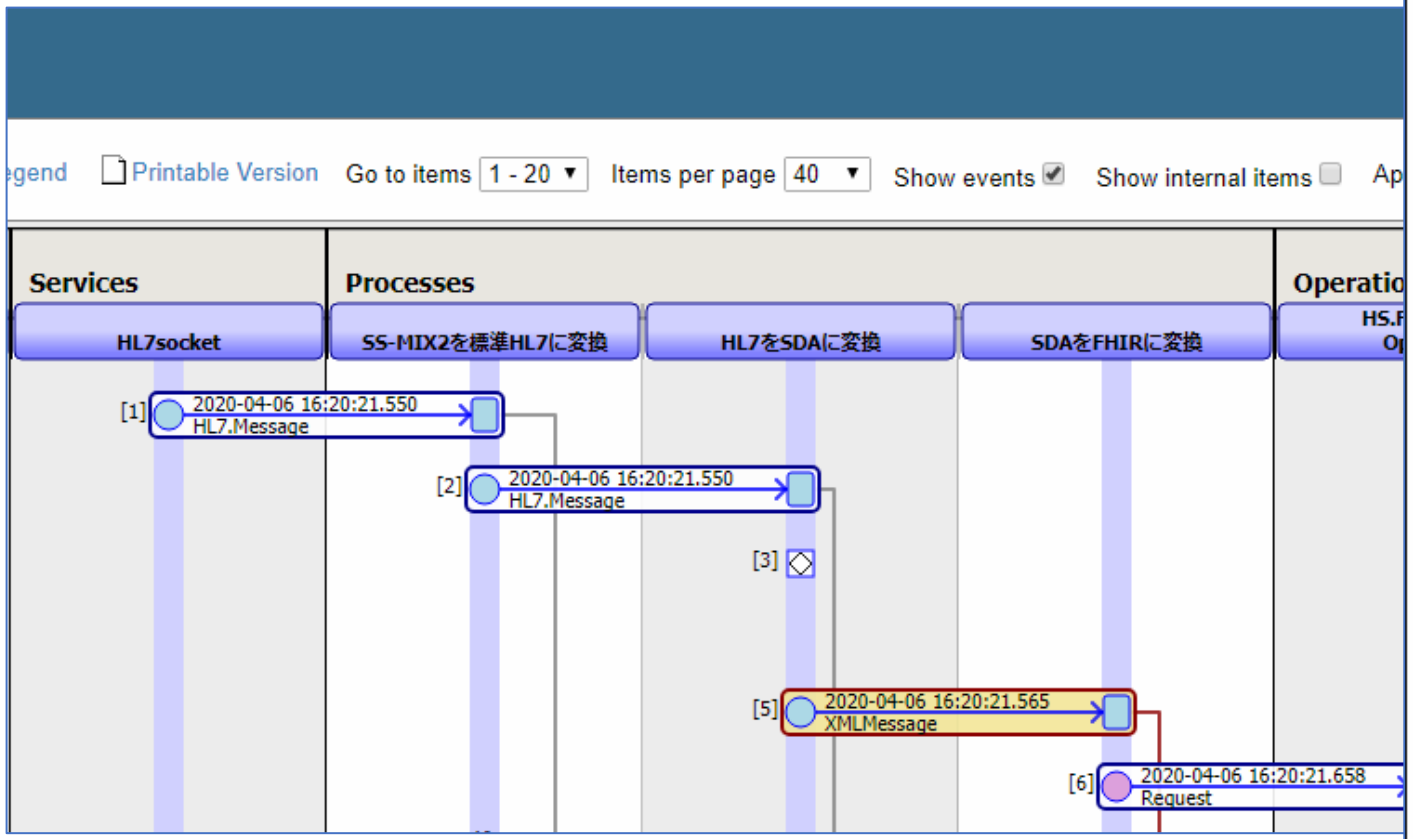
View Full Contents | View Raw Contents

HL7 **OMP_O09** Message - Id = **1039**, DocType = '**YUYAMA:OMP_O09**', Message Type '**薬剤/処置オーダ**', **19** Segments

1	MSH | ^~\& | PC-ORDERING/AD | HIS | GW | GW | 20200322163813 |
2	ZGW | 80006006 | 20200322180000 | OMP-01 ^ 処方オーダ ^ L | 003200
3	PID | 0001 | | 80006006 | | テスト　ユヤマＩＦ ^ ^ ^ ^ ^
4	ZIN | 01 ^ 協会けんぽ ^ L
5	ORC | NW | 003200051030710 | | 1 | | | | | 2020032216381
6	RXE | | | | | | | | | | | | | | | 006550 |
7	TQ1 | 1 | 2 ^ T | 3N123 & 分3　朝・昼・夕食後 & L | | | 2 | 202
8	RXO | 186400 ^ 結合型エストロゲン ^ MDCHOT ^ 186300 ^ プレドニン錠
9	RXR | PO | | | | |
10	ORC | NW | 003200051030710 | | 2 | | | | | 2020032216381
11	RXE | | | | | | | | | | | | | | | 006550 |
12	TQ1 | 1 | 2 ^ T | 2A13 & 分2　朝・夕食直後 & L | | | 2 | 202003
13	RXO | 186400 ^ 結合型エストロゲン ^ MDCHOT ^ 186300 ^ プレドニン錠
14	RXR | PO | | | | |
15	ORC | NW | 003200051030710 | | 3 | | | | | 2020032216381
16	RXE | | | | | | | | | | | | | | | 006550 |
17	TQ1 | 1 | 2 ^ T | 1V1 & 分1　朝食前 & L | | | 2 | 2020032218 |
18	RXO | 186400 ^ 結合型エストロゲン ^ MDCHOT ^ 186300 ^ プレドニン錠
19	RXR | PO | | | | |

HL7からSDAへの変換

InterSystems IRIS Health



Header | Body | Contents

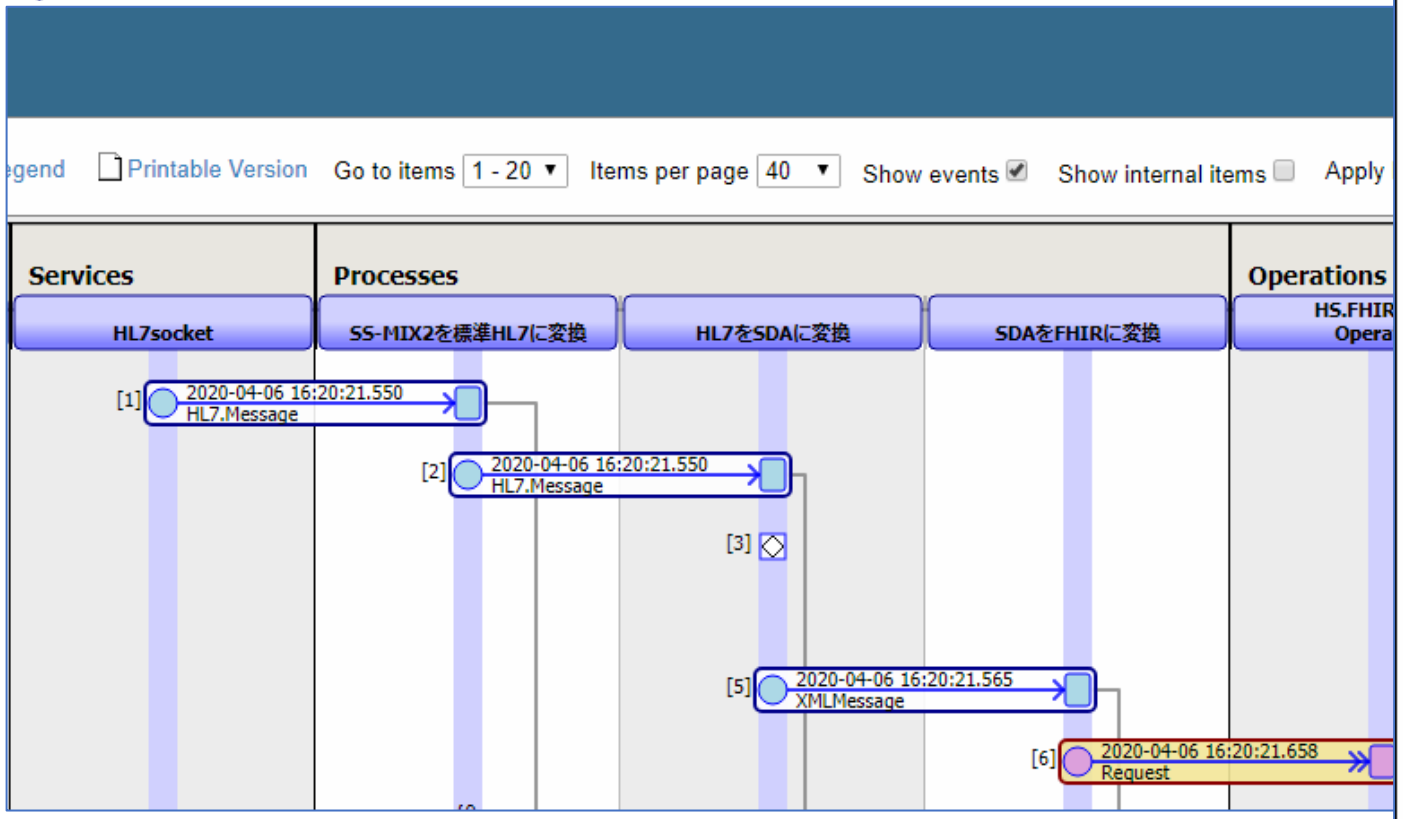
View Full Contents | View Raw Contents

Expand All

```
<?xml version="1.0" ?>
<!-- type: SSMIX2toFHIR.Message.XMLMessage  id: 410 -->
<XMLMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:s="http://www.w3.org/2001/XMLSchema">
    <Name>SDAStream</Name>
    <ContentStream><![CDATA[<?xml version="1.0"
    encoding="UTF-16"?>
    <Container
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:isc="http://extension-
    functions.intersystems.com">
    <EventDescription>OMP_O09</EventDescription>
    <SendingFacility>HIS</SendingFacility>
    <Patient>
    <Aliases>
    <Name>
    <Extension>
    <RepresentationCode>IDE</RepresentationCode>
    </Extension>
    <FamilyName>テスト　ユヤマＩＦ</FamilyName>
    <Type>Legal</Type>
    </Name>
    <Name>
    <Extension>
    <RepresentationCode>SYL</RepresentationCode>
    </Extension>
    <FamilyName>テスト ユヤマIF</FamilyName>
    <Type>Legal</Type>
    </Name>
    </Aliases>
    <PatientNumbers>
    <PatientNumber>
```

SDAからFHIRへの変換

InterSystems™
IRIS Health



Header | Body | Contents

View Full Contents | View Raw Contents

Expand All

```
<?xml version="1.0" ?>
<!-- type: HS.Message.FHIR.Request  id: 920 -->
<Request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:s="http://www.w3.org/2001/XMLSchema">
    <TimestampUTC>2020-04-06T07:20:21Z</TimestampUTC>
    <ContentType>application/json+fhir</ContentType>
    <Payload>{
      "resourceType": "Bundle",
      "entry": [
        {
          "fullUrl": "Patient/49515690-f5fc-431e-8343-
d03c856065e6",
          "request": {
            "method": "PUT",
            "url": "Patient/49515690-f5fc-431e-8343-
d03c856065e6"
          },
          "resource": {
            "resourceType": "Patient",
            "birthDate": "1998-01-01",
            "gender": "male",
            "id": "49515690-f5fc-431e-8343-d03c856065e6",
            "identifier": [
              {
                "assigner": {
                  "reference": "Organization/ef583fb7-
4864-4b66-b13d-98c625c2ebb3"
                },
                "value": "80006006"
              }
            ]
```

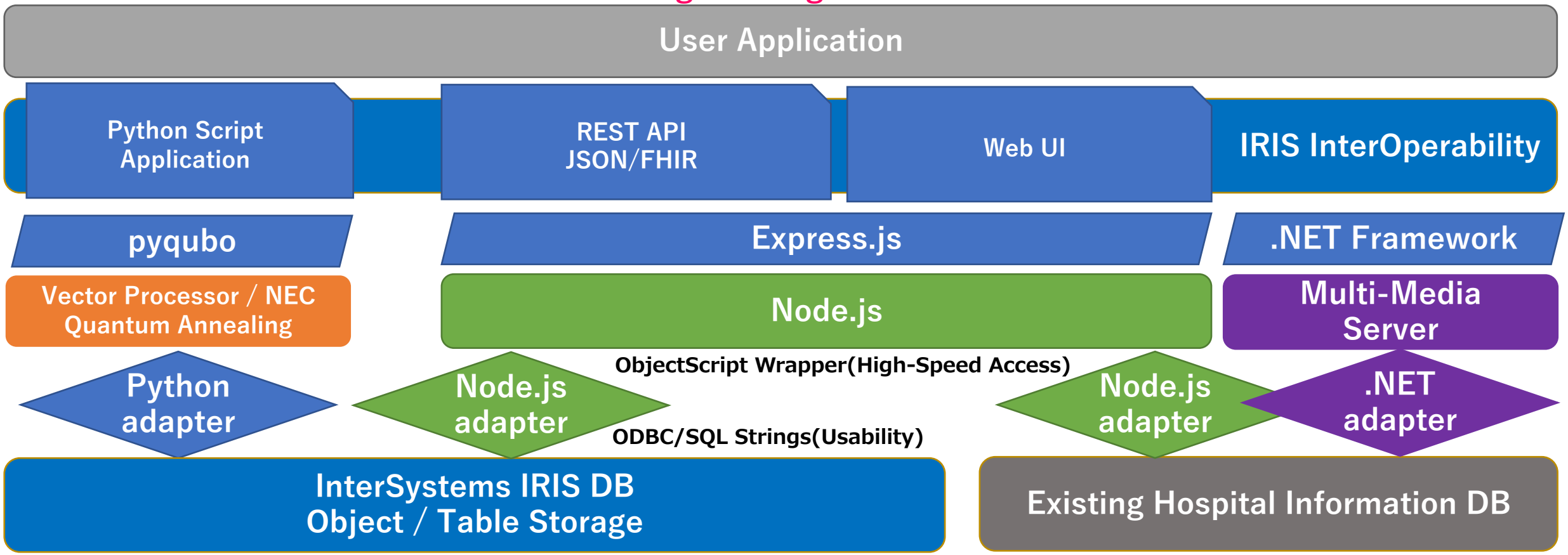
Architecture Design For Clinical Intelligence

Aiming Outcome:

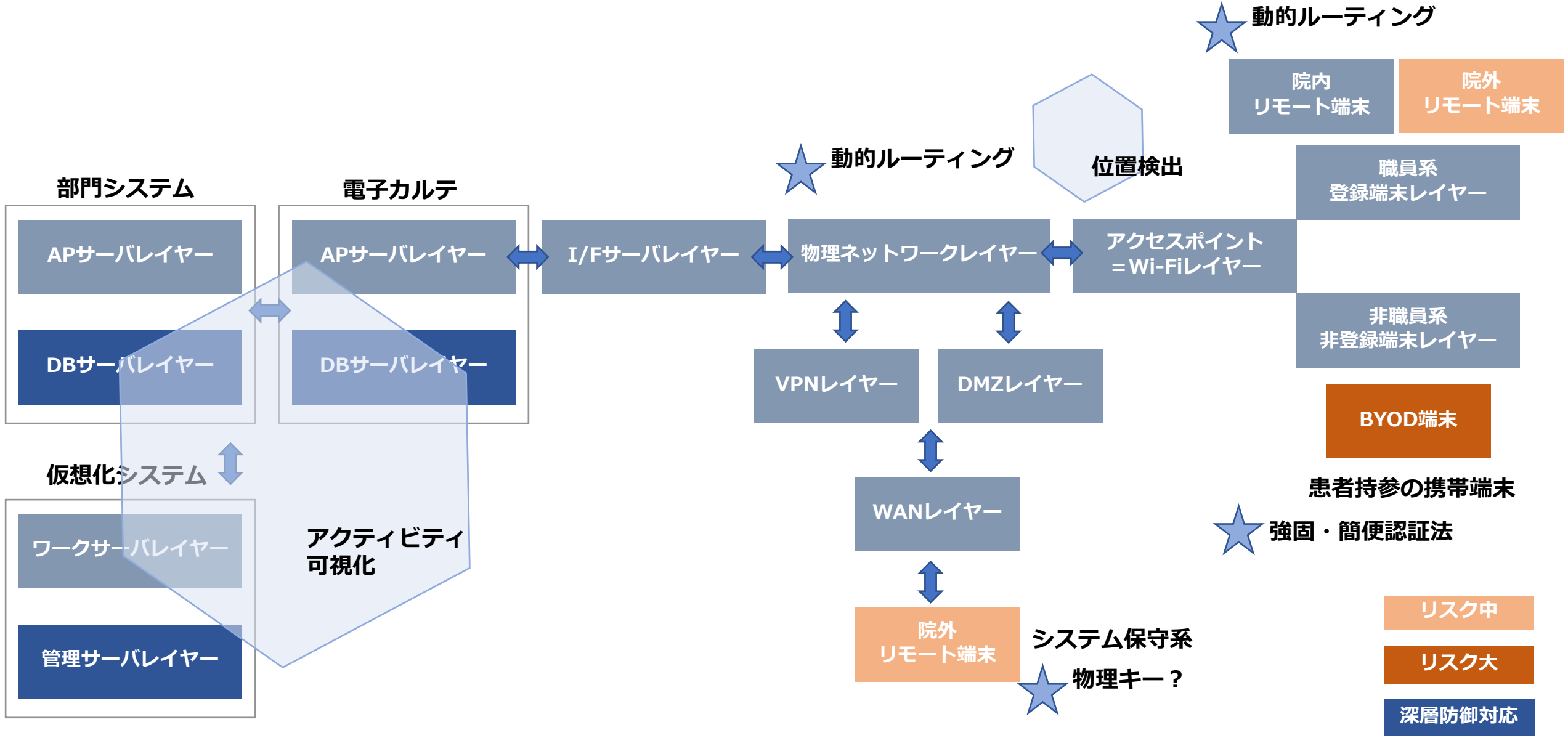
Combination optimization For Faster Scheduling

Adapt FHIR Connection & Prototype Object-Oriented Programming

Multi-Device Application Tablet/SmartPhone



ネットワークセキュリティ向上とサービス向上両立を目指した構造の考え方



ベンダー側視点の「医療ソフトウェア」のライフサイクルと経済性

医療ITが有する素因

- **医療の現状は「半分市場、半分福祉」？**
- **ITベンダーが収益を高くできるのは、「ソフトウェアは容易に複製できる」から**
- **＜医療ノウハウ＞の詰まったコードを「棄てざるを得ない」事情がベンダーの収益性の障壁になってきた**

病院で利用されるベンダー製ソフトウェアの始まりから終わりまで

「試してくれる」医療者とコンタクトする機会の獲得
大学病院：医療情報部
診療科
民間は手探り
凄い苦労とコスト

医療者にITベンダーがヒアリング
言語が通じ合わない壁
医療知識のベース＋
ITスキルのベース
凄い苦労とコスト

テストソフトの作成
開発コスト
＜医療ノウハウ＞
捻出元が必要
優れた
アーキテクトが
寿命を決める
多くはボツ、歩留不良

「注文の多い料理店」
＜医療ノウハウ＞

機能追加
接続変更

バグ改修
「非常に高い信頼性」
→改修が強く忌避される理由

実地検証
医療現場IT能力
凄い苦労とコスト

病院が
「ソフトに診療を合わせず、
診療にソフトを合わせる」
よう要求

他サイト拡販

OS等
環境変化
追従の移植
使えなくなる関数やコード
を探して移植するコストが
ユーザーに理解されない
開発主の離職

貴重なヒアリングや
医療現場知見が棄てられる

ローカル
バージョン
保守困難
→終売

事業継続

移植困難
→終売
貴重なヒアリングや
医療現場知見が棄てられる
知見が積みあがらず
同じ事の繰り返し
開発者が楽にならない
コスト回収が困難に
→医療IT最大の問題

知的労働としても、費用としても、期間としても多額の投資
（電子カルテでは＞20年）

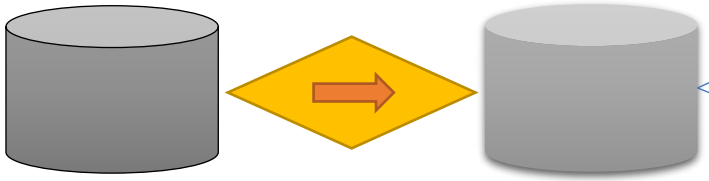
ソフトウェアの収益性

多くの電子カルテベンダー、PACSベンダーなどが、Web化に際して古いプラットフォーム（＋ノウハウ資源）を棄てた
新しいプラットフォームのアーキテクトは「標準化対応するには複雑独自に作りこみ過ぎた」状態？

病院側視点の「電子化データ」利活用の目的とシステム構造

医療ITが
有する素因

- **病院経営の改善**
  - 収入増加（診療報酬・加算の獲得支援）
  - 不要支出の削減（診療中のプロセスの無駄、廃棄率の高い物品機材などの可視化）
  - 合理的な人事・システム再配置
- **病院診療での活用**
  - 「定期診療帳票」の発行
  - Learning Health System (LHS)へのアプローチ
  - （大学病院であれば）研究開発

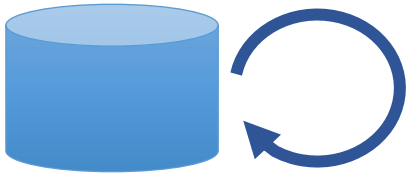


電子カルテ

Data Transfer Logic(DTL)	Data Ware House (DWH)
ソフトウェア費用	ソフトウェア費用
コード作成費	コード作成費
コード改造費	コード改造費

アルゴリズムをベンダー製独自コードで
記述すると、移植困難→ベンダーロック誘発

支払・収入

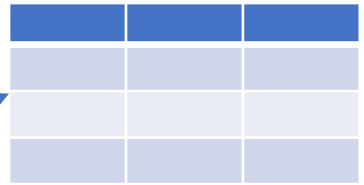


ベンダーソフト
新機能を基本機能と
してキャッチアップ

短期利用（更新計画不要）であれば導入は比較的容易
長期利用を考える場合は長期保守リスク検討が不可欠

この問題に加えて、「病院内APIの超長期保守」の問題、
「医療レベル向上の要求対応」の問題が生じる

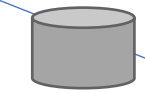
ベンダーロックを起こしやすい方法



←多くの病院が選択している側

DWHソフトが備えているビューを利用する
（事務職などでコーディングしない場合）

ベンダーロックを起こしにくい方法



HTMLサーバ

←今後望まれる側
→病院ごとにプログラマ雇用が必要

Webサーバなどを介してビューを作成する
（別途コーディングエンジニアが常駐）

市場では既に「有能なプログラマの争奪戦」が起きている
病院への問い：
機械学習やWebで高給が支払われているプログラマに、
魅力ある職場であることをアピールするには、
どんな魅力を掲げればよいか？

データ分析に係る本質的な特徴

分析し始めてしばらくすると、「こんなデータを予め測定しておけばよかった」と気が付く

特徴１：データ分析では、
「最初にデータを揃えた時から、追加データの発生を意識しておく必要がある」

特徴２：データ分析では、
「追加データの発生に伴い、データベースの頻回な変更を企図する必要がある」

データ分析に係る本質的な特徴

分析し始めてもう少しすると、「このような分析をすべきであった」と気が付く

特徴３：データ分析では、
「分析方法の洗練に対応できる快適な操作性・高速なデータプロセス速度が不可欠」

特徴４：データ分析では、
「データ抽出・分析アルゴリズムのスキル人材確保・頻回な編集作業の計画が不可欠」

InterSystems製品との関わり方に関する消化不良の話題

医療情報マネージャーの理想：
「作りこんだシステムコードや設定の資産を失うことなく維持発展させたい」

IRIS InterOperability → JavaScript ECMA2015-

夥しいシステム更新で失敗してきていること：
「アルゴリズムの記述された、ある言語コードを、多言語に移植したくても、翻訳スキルをもち、かつアルゴリズムが理解できるエンジニアは絶滅危惧種」
POSIX思想のように、「あるコードがいつまでも動く」ように構成するならば、それはスクリプト言語であるべきであるし、将来に「翻訳を委託すれば済む」と絶対に安易に考えてはいけない
また
「システム内で使用される開発言語の総数が多すぎると、マネジメントできなくなる」
このため、システム内使用言語の数を減らすことがシステムの永続性を強化するために極めて重要

UNIX-bashに代わるPOSIX性の言語候補として選定
医療アルゴリズム（Business Process）は
JavaScriptで記述するように設計を決めている

NLP開発機能に関する悩み

開発ベースで堀田さん/大平さんと検討中　　バージョンアップに伴って商用実装される

Python NLP → IRIS NLP

実装コード：Python
データの保存先：メモリまたはPythonが読み書きできるファイル
ファイルアクセス速度：低
アルゴリズムの書きやすさ：高

実装コード：ObjectScript
データの保存先：IRISアーキテクチャ（データベース）
ファイルアクセス速度：高
アルゴリズムの書きやすさ：低

悩み：
NLPアルゴリズムを記述しやすいのはPythonであるが、Python NLPで書き続けると、このまま作りこめば、病院内で運用したい際には「移植」が必要になる：
といって、ObjectScriptでNLPを記述するにはそれなりのスキルが必要で、アルゴリズム資産を構成するのに不安が大きい。またPythonで書き続けると、研究発表としては名が売れるが、そのアルゴリズムの実態をIRIS製品に反映できないと、長期的にInterSystemsの利益にならないし、サポートなども受けられず、自分の書いたコードが広まり、医療業界に根付いていかない

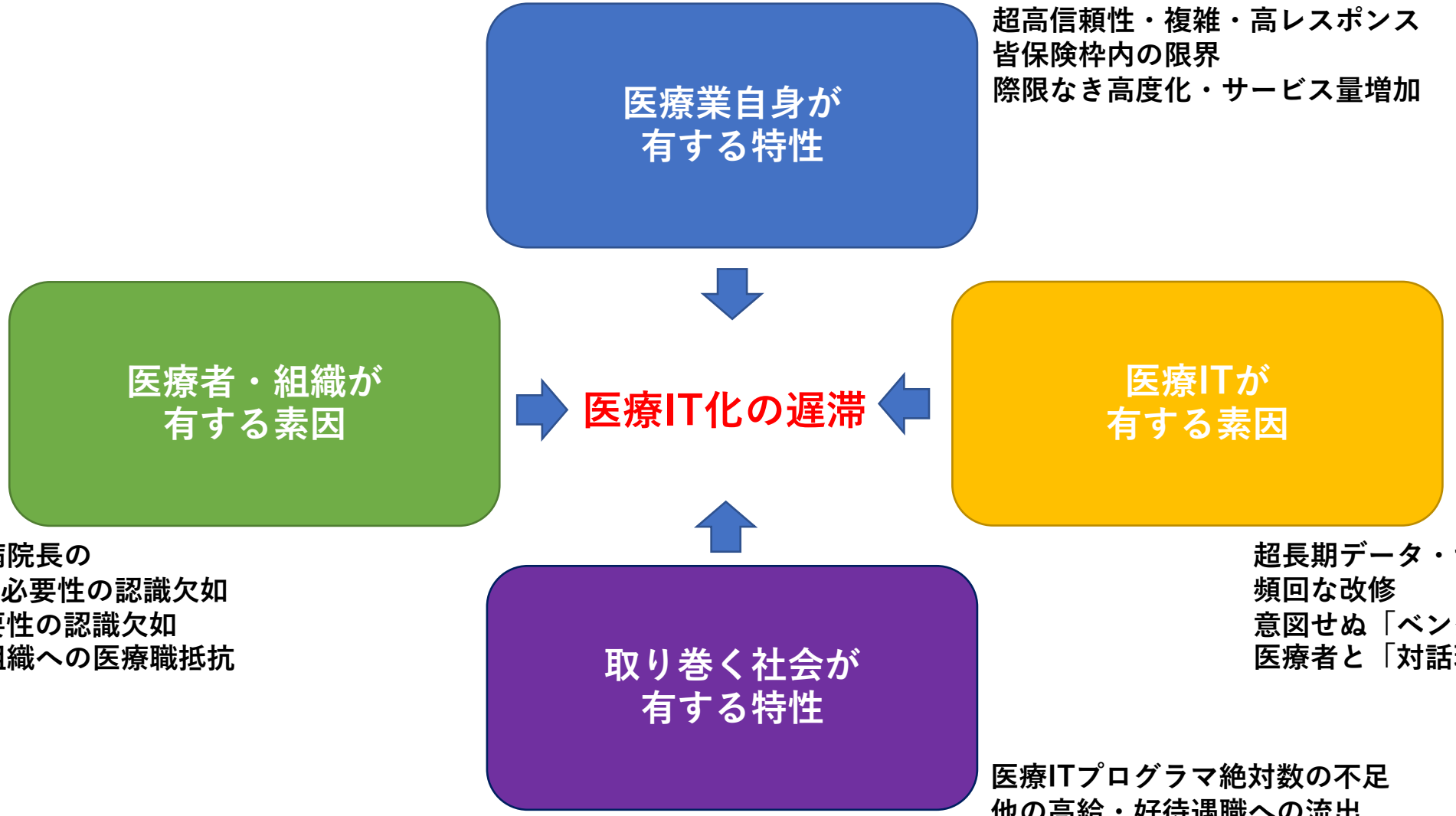
相談したいこと：
群大病院では着手段階からPOSIX性を重視した開発を意識しているために、システム永続性や実装時のスムーズさを維持できているが、NLPの開発を、ソフトウェアの販売まで見据えて行うためには、どのような開発構成を設計すればよいか？

最初の問い

- **なぜ日本の医療ITは、他の産業ITに比べて進歩しにくいのか？**

医療ITの進歩が遅滞する要因の整理

- **これらの素因がさらに交絡することで、医療IT化が進まない要因に**



なぜか、インターシステムズさんのチュートリアルが難しい…

- **使い始めるのに時間がかかる**
- **なぜかコンソール操作で説明する**
- **想定ユーザー＝データベース管理者？**
- **…ということで、「アプリケーションユーザー」から見た、IRISの使い方を紹介します**

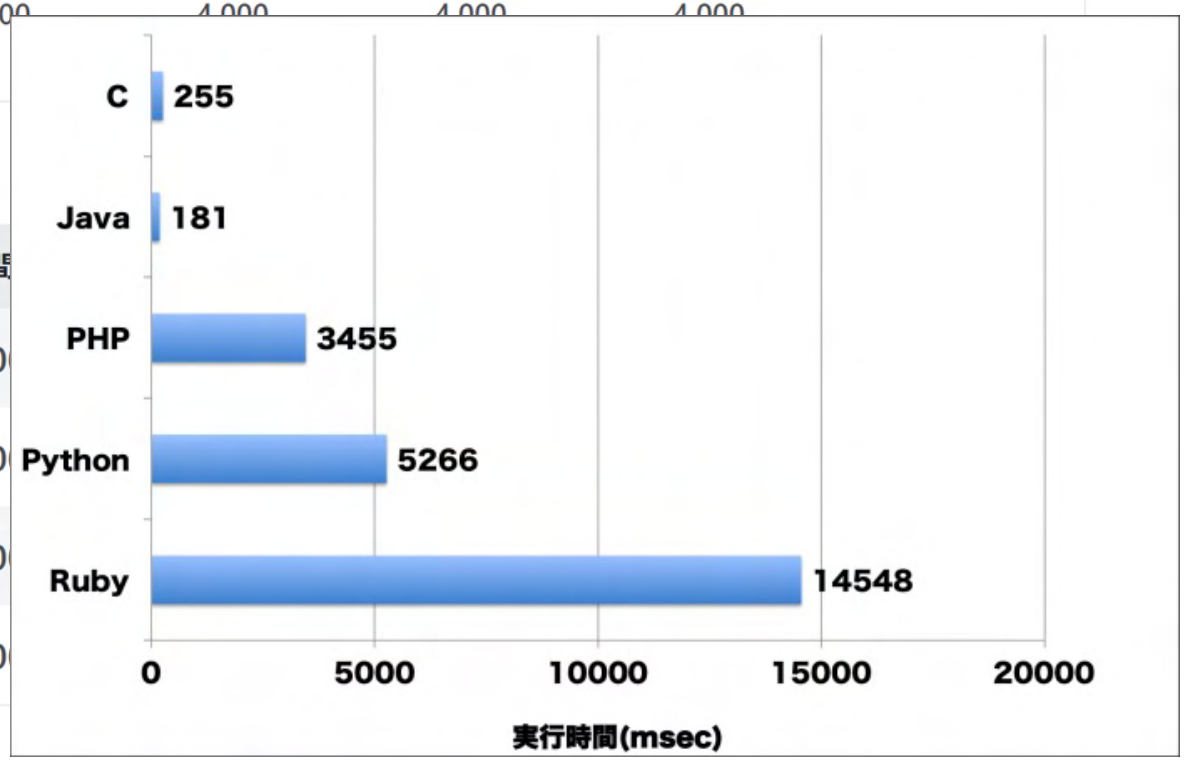
クラウド時代は、改めて「アルゴリズムの良さ」を求めている

Azure ・ クラウド上の「仮想マシン」のディスク価格

ディスク サイズ (GiB)	4	8	16	32	64	128	256	512	1,024-65,536 (1 TiB ずつ増加)
最大 IOPS	1,200	2,400	4,800	9,600	19,200	38,400	76,800	130,000	
最大スループット (MB/秒)	300	600	1,200	2,400	4,000	4,000	4,000	4,000	

*ディスクを構成する方法の詳細については、Ultra Disk のドキュメント ページをご覧ください。

Ultra ディスク構成	単位	1 時間
ディスク容量 (GiB)	GiB	$0.0
プロビジョニング済み IOPS	IOPS	$0.0
プロビジョニング スループット (MB/秒)	MBps	$0.0
プロビジョニング済み vCPU 予約料金*	vCPU	$0.0



https://web.archive.org/web/20140802192155/http://maplesystems.co.jp/blog/all/programming/2015

オブジェクトデータベース：M言語からObjectScriptまで

- **MGH（マサチューセッツ総合病院）で、電子カルテを機能させるために作られて55年**

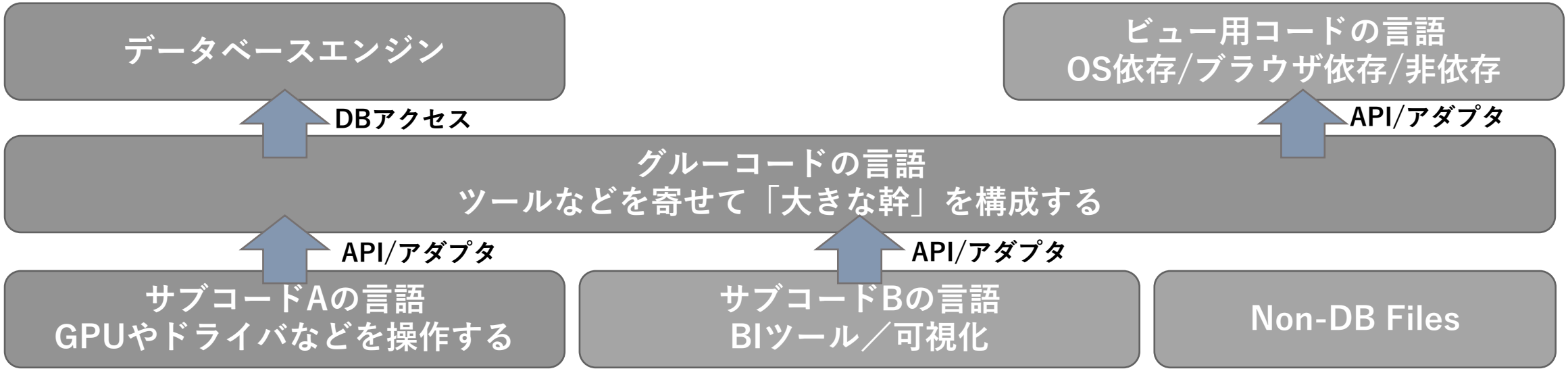
M言語は標準言語です
国際標準言語ISO/IEC 11756 International technology－Programming language-MUMPS），日本工業規格標準言語（プログラミング言語MUMPS JIS X 301l-1995），米連邦標準言語（American National Standard for Information System－Programming Languages－MUMPS ANSI/MDC X11.1-1977，1984，1990），米連邦情報処理標準言語（政府調達標準：Federal Information Process-ing Standard：FIPS PUB 125-1）などの 標準規格プログラミング言語です。

コードの「機能」と特徴

- **サブコードが「継続的利用」に不可欠な機能を有する場合、ユーザーからみたソフトウェアの「価値」は「サブコードの基幹業務での必要性と長期稼働性」によって大きく左右される**

グルー言語の特徴では「アダプタ」の充実度と、
市場に出てくる新製品のキャッチアップが重要
グルー言語は「交換が極めて困難」→慎重を期した選定が不可欠

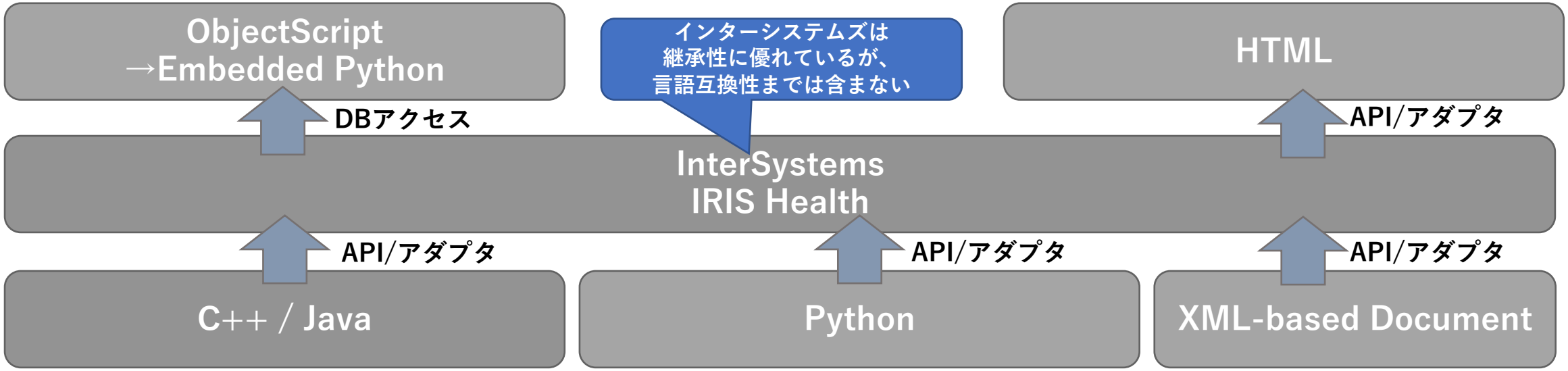
アーキテクチャ設計で考えること：
ビューを提供するOSの検討（OS依存を選ぶか、非依存を選ぶか）
市場で雇用できるエンジニアの技術力と待遇の検討
「長く使える」ビューの構成と検討
デバイスの調達のしやすさ



アーキテクチャ設計で考えること：
サブコードの言語の持続性と、市場から開発者を確保できる見通しは？
市場で開発者が少ない場合、自社に長く留まってもらう配慮ができるか？
最悪の場合、改修不能、移植不能となった場合の事業継続をどうするか？
１：ソフトの寿命と思って終売としてよいか？
２：将来的により簡便なソフトを用いて再開発するよう検討するか？

コードの「機能」と特徴

- **ソフト系ベンダーのビジネスモデル例２： InterSystemsの（おそらく）考えるプラットフォーム構想**



バスとなるミドルウェアは、「接続数の多さ」自身がソフトの価値であるが、
現実に接続数を増やすと、ミドルウェアは「替えが利かない」急所になる
海外の大病院に比べると日本の病院システム集約規模は小さく、
ライセンスが比較的高価なIRISは実装の工夫が必要

最大の利点：Python言語で、アプリケーションもIRIS-DB操作も可能に

内容	C++	VisualBasic	ObjectScript	Python
変数宣言	int i;	Dim i as Integer	Class内定義 Property i as %Numeric; ClassMethod内定義 #dim i as %Numeric	宣言不要

関数、引数、戻り値	（定義：戻り値なし）
	void fct(int i, int *j){
	}
	（定義：戻り値あり）
	int fct(int i, int *j){
	return nInt;
	}
	
	使用(戻り値なし)
	cls1::.fct(i, &j);
	使用（戻り値あり）
	k = cls1.fct(i, &j);
		（定義：戻り値なし）
		Sub fct(byval i as integer,
		byref j as integer)
		End Sub
		（定義：戻り値あり）
		Function fct(byval i as integer,
		byref j as integer) As Integer
		Return nInt
		End Function
		使用（戻り値なし）
		cls1.fct(i, j)
		使用（戻り値あり）
		k = cls1.fct(i, j)
			定義
			ClassMethod fct(i as %Numeric, byref
			j as %Numeric) as %Numeric
			{ quit nInt}
			
			使用（戻り値なし）
			Do ##class(“cls1”).fct(i, .j)
			使用（戻り値あり）
			Set k = ##class(“cls1”).fct(i, .j)
			
			または(動的な関数名決定要請の場合)
			$CLASSMETHOD(“cls1”, “fct”,i, .j)
				定義
				class cls1:
				def fct(i, j=0):
				return nInt
				#タブで関数内記述
				
				使用
				k = cls1.fct(i)

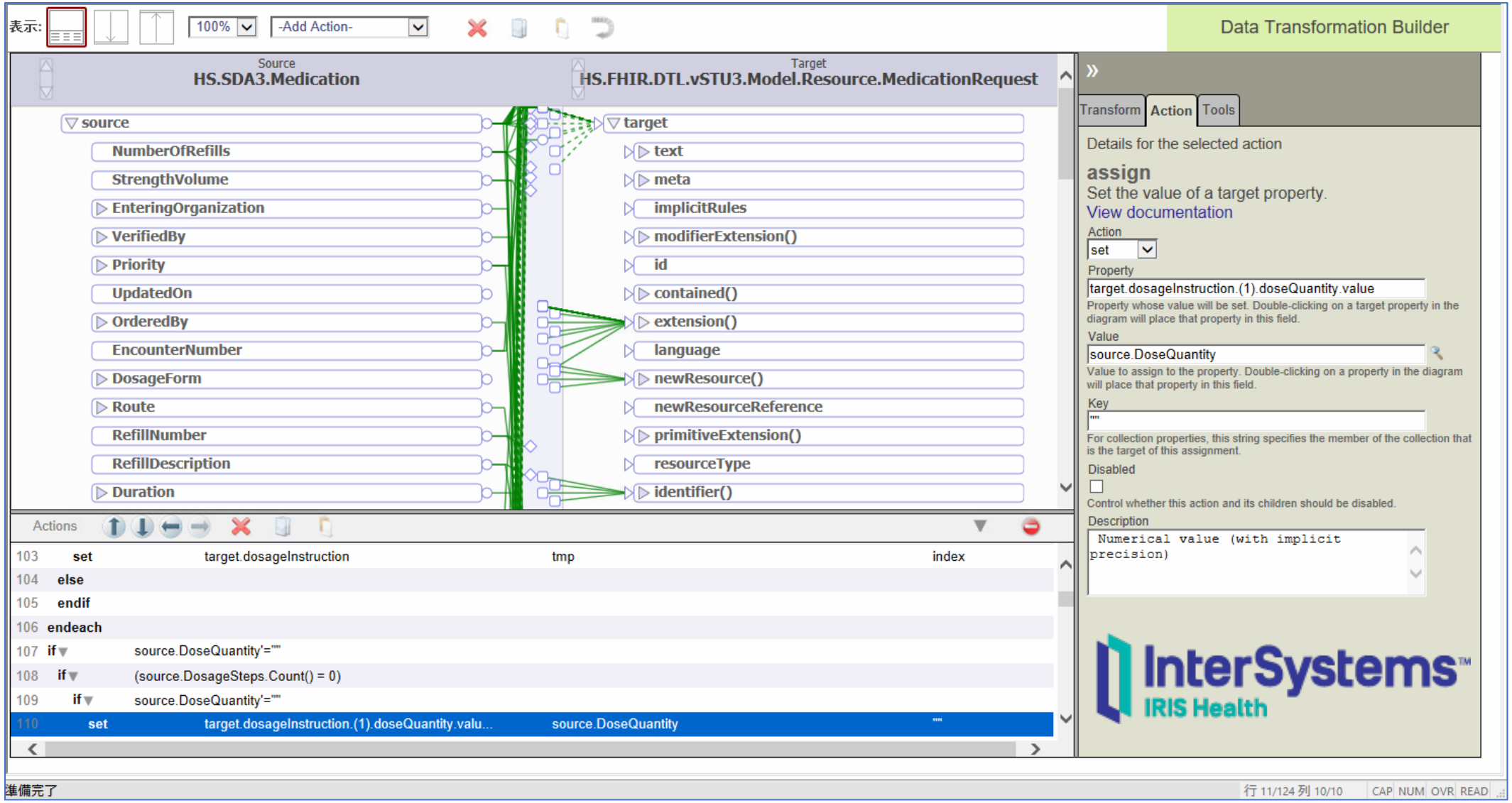
値の代入	i = 15;	i = 15	Set i = 15	i = 15
ライブラリ関数 特殊変数	include<cmath> f = cos(0.25);	f = Math.Cos(0.25)	$ZCOS(0.25) $Horolog	import math math.cos(0.25)
マクロ	#define DEBUG 1	#Const DEBUG=1	#define DEBUG 1 $$$DEBUG	キーワードなし（大文字表記が慣例）
クラス内オブジェクト	this->obj	Me.obj	..obj	self.obj
コマンド(ターミナルで用いる)	system(“C:test.txt");	System.Diagnostics.Process.Start(objProcIfm)	write “samplestring”	subprocess.call(“C:test.txt")

繰り返し	int i;
	for(i = 0; i < 10; i=i+1){
	}
		Dim i as Integer
		For i = 0 to 10 Step 1
		Next i
			For i=0:1:10 {
			}
				for i in range(10):
				#タブで関数内記述

クラス定義	C++	Visual Basic .NET	ObjectScript	Python

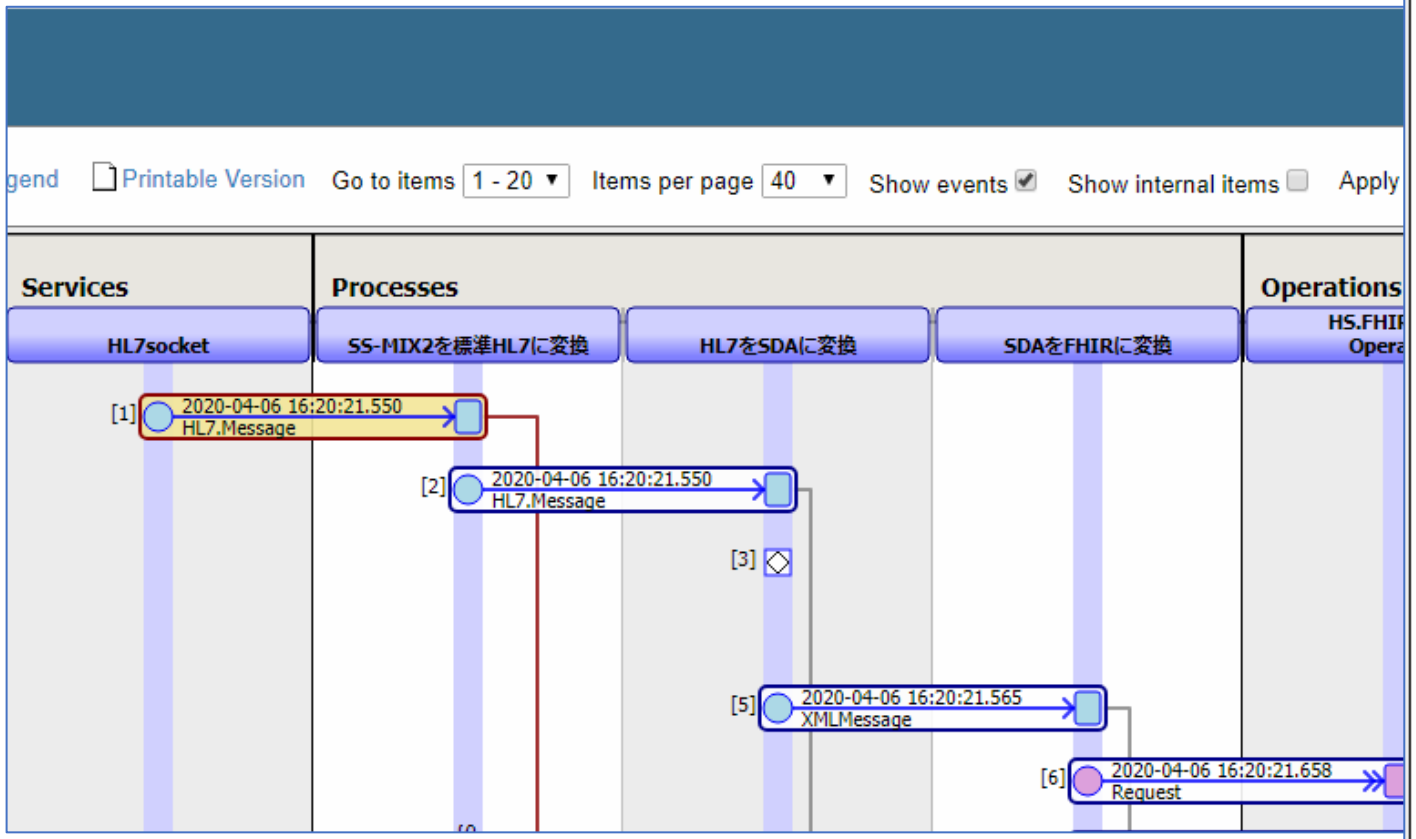
	#include <stdio>
	#include <string>
	class Cls1 : public ClsBase
	{
	public:
	int i;
	/*コメントtype1*/
	//コメントtype2
	void NewID()
	{
	//処理を記述
	}
	//戻り値のある関数
	std::string
	NewID2(std::string
	filename)
	{
	
	std::string s;
	//文字列操作
	s = std::string("a") +
	std::string("b");
	return s;
	}
	};
		Public Class Cls1
		
		Inherits ClsBase
		
		Public i as Integer
		
		‘コメントtype1
		Sub NewID()
		
		‘処理を記述
		End Sub
		‘戻り値のある関数
		Function NewID2(ByVal
		filename as String) as
		String
		
		Dim s as String
		‘文字列操作
		s = “FHIR” + “IRIS”
		
		Return s
		End Sub
		End Class
			Class User.Main.Cls1 Extends
			User.Main.ClsBase
			{
			///行頭に半角スペース
			␣Property i as %Numeric;
			
			/*コメントtype1*/
			///コメントtype2
			ClassMethod NewID()
			{
			//処理を記述
			}
			//戻り値のある関数
			ClassMethod NewID2(filename
			as %String) As %String
			{
			
			␣#dim s as %String
			//文字列操作
			␣Set s = “FHIR”_”IRIS”
			
			␣Quit s
			}
			}
				class cls1:
				
				#変数宣言不要
				
				#コメント
				
				def NewID():
				#処理を記述
				
				def NewID2(filename):
				#処理を記述
				
				#変数宣言不要
				s = “FHIR”+”IRIS”
				
				return s

データ変換画面(SDA⇒FHIR)



HL7変換のサポート

InterSystems IRIS Health



Header | Body | Contents

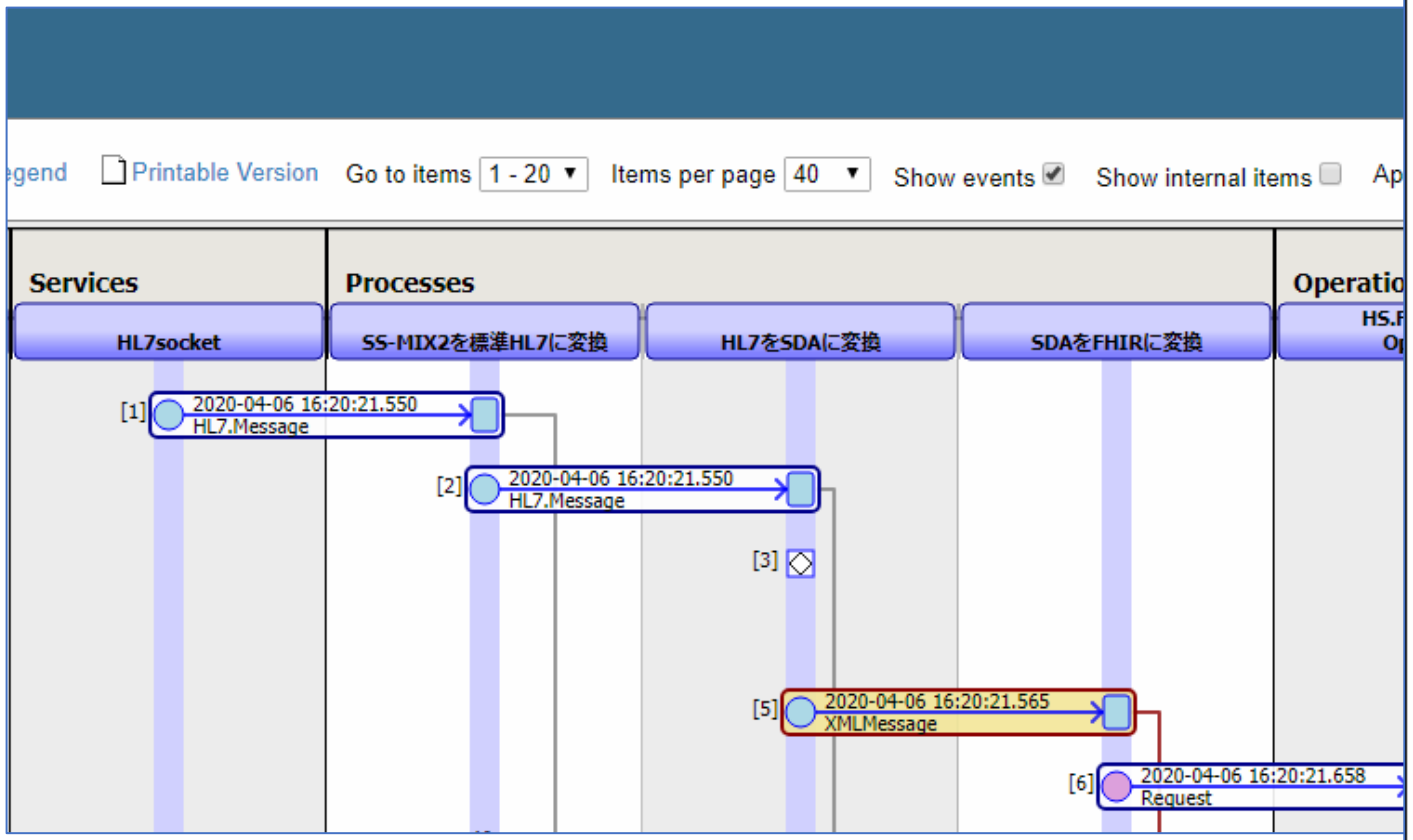
View Full Contents　　View Raw Contents

HL7 **OMP_O09** Message - Id = **1039**, DocType = '**YUYAMA:OMP_O09**', Message Type '**薬剤/処置オーダ**', **19** Segments

1	MSH	^~\& | PC-ORDERING/AD | HIS | GW | GW | 20200322163813 |
2	ZGW	80006006 | 20200322180000 | OMP-01 ^ 処方オーダ ^ L | 003200
3	PID	0001 | | 80006006 | | テスト　ユヤマＩＦ ^ ^ ^ ^ ^
4	ZIN	01 ^ 協会けんぽ ^ L
5	ORC	NW | 003200051030710 | | 1 | | | | | 2020032216381
6	RXE	| | | | | | | | | | | | | | | 006550 |
7	TQ1	1 | 2 ^ T | 3N123 & 分3　朝・昼・夕食後 & L | | | 2 | 202
8	RXO	186400 ^ 結合型エストロゲン ^ MDCHOT ^ 186300 ^ プレドニン錠
9	RXR	PO | | | | | |
10	ORC	NW | 003200051030710 | | 2 | | | | | 2020032216381
11	RXE	| | | | | | | | | | | | | | | 006550 |
12	TQ1	1 | 2 ^ T | 2A13 & 分2　朝・夕食直後 & L | | | 2 | 202003
13	RXO	186400 ^ 結合型エストロゲン ^ MDCHOT ^ 186300 ^ プレドニン錠
14	RXR	PO | | | | | |
15	ORC	NW | 003200051030710 | | 3 | | | | | 2020032216381
16	RXE	| | | | | | | | | | | | | | | 006550 |
17	TQ1	1 | 2 ^ T | 1V1 & 分1　朝食前 & L | | | 2 | 2020032218 |
18	RXO	186400 ^ 結合型エストロゲン ^ MDCHOT ^ 186300 ^ プレドニン錠
19	RXR	PO | | | | | |

HL7からSDA(Standard Data Architecture)への変換

InterSystems IRIS Health



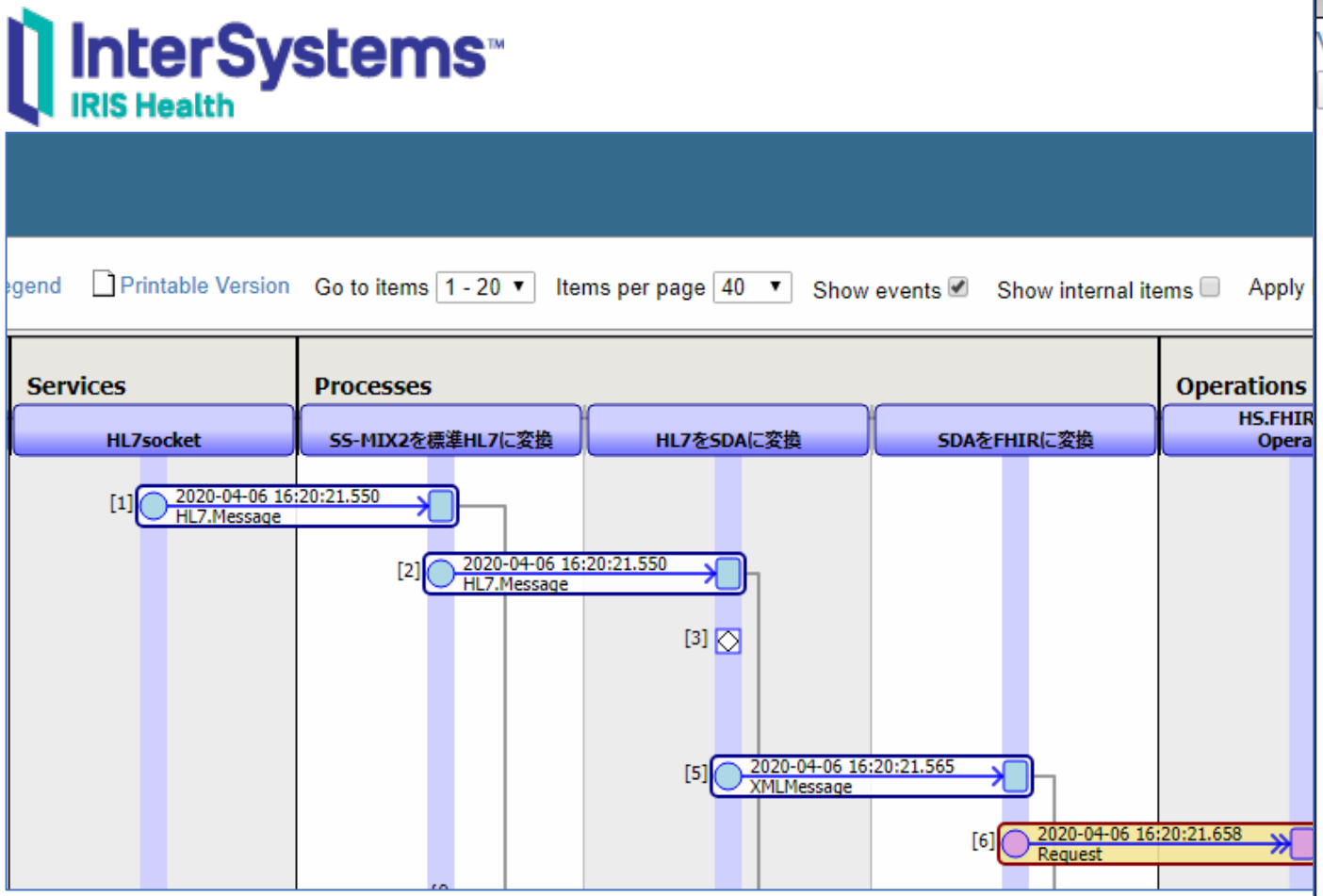
Header | Body | Contents

View Full Contents | View Raw Contents

Expand All

```
<?xml version="1.0" ?>
<!-- type: SSMIX2toFHIR.Message.XMLMessage id: 410 -->
<XMLMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:s="http://www.w3.org/2001/XMLSchema">
    <Name>SDAStream</Name>
    <ContentStream><![CDATA[<?xml version="1.0"
    encoding="UTF-16"?>
    <Container
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:isc="http://extension-
    functions.intersystems.com">
    <EventDescription>OMP_O09</EventDescription>
    <SendingFacility>HIS</SendingFacility>
    <Patient>
    <Aliases>
    <Name>
    <Extension>
    <RepresentationCode>IDE</RepresentationCode>
    </Extension>
    <FamilyName>テスト　ユヤマＩＦ</FamilyName>
    <Type>Legal</Type>
    </Name>
    <Name>
    <Extension>
    <RepresentationCode>SYL</RepresentationCode>
    </Extension>
    <FamilyName>テスト ユヤマIF</FamilyName>
    <Type>Legal</Type>
    </Name>
    </Aliases>
    <PatientNumbers>
    <PatientNumber>
```

SDAからFHIR(Fast Healthcare InteroperableResources)への変換



Header | Body | Contents

View Full Contents | View Raw Contents

Expand All

```
<?xml version="1.0" ?>
<!-- type: HS.Message.FHIR.Request  id: 920 -->
<Request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:s="http://www.w3.org/2001/XMLSchema">
    <TimestampUTC>2020-04-06T07:20:21Z</TimestampUTC>
    <ContentType>application/json+fhir</ContentType>
    <Payload>{
      "resourceType": "Bundle",
      "entry": [
        {
          "fullUrl": "Patient/49515690-f5fc-431e-8343-
d03c856065e6",
          "request": {
            "method": "PUT",
            "url": "Patient/49515690-f5fc-431e-8343-
d03c856065e6"
          },
          "resource": {
            "resourceType": "Patient",
            "birthDate": "1998-01-01",
            "gender": "male",
            "id": "49515690-f5fc-431e-8343-d03c856065e6",
            "identifier": [
              {
                "assigner": {
                  "reference": "Organization/ef583fb7-
4864-4b66-b13d-98c625c2ebb3"
                },
                "value": "80006006"
              }
            ]
```

HL7 v2.x（SS-MIX2）からFHIRへの変換結果

サンプル

```
MSH|^~¥&|PC-ORDERING/AD|HIS|GW|GW|20200322163811||OMP^O09|5000095|P|2.5||||||ISO IR87||ISO 2022-1994|
ZGW|80006006|20200322180000|OMP-01^処方オーダ^L|003200051030709|INS|04^精神科神経科^L|K60^北病棟６階^L|663^663号室^L|
PID|0001||80006006||テスト　ユヤマＩＦ^^^^^^^^L^I~テスト ユヤマIF^^^^^^^^L^P||19980101|M||||||||||||||||||||||||||
ZIN|01^協会けんぽ^L|
ORC|NW|003200051030709||1|||||20200322163811|YAKU^日本電気^^^^^^^^L^I~エヌ イーシー^^^^^^^^L^P||99904^Ｄr.精神科^^^^^^^^L^I~ンdr.セイシンカ^^^^^^^^L^P|||||04^精神科神経科^L||||||||||||X|
RXE|||||||||||006549|||||||||||||||||||||||||||||
TQ1|1|14^ｇ|3N123&分3　朝・昼・夕食後&L|||14|2020032218|||YCMN19^医師の指示どおり服用||||
RXO|103200^アデノシン三リン酸二ナトリウム^MDCHOT^103200^アデホスコーワ顆粒10%(100mg/g)^L|3||G^ｇ^L|01^散・錠・カプセル^MR9P^^^L|SOD^B^L^^一包化指定~DVD^0-0-0-0-0-0-0^L^^不均等指示~HCM^^L^^テスト処方箋コメント|||||42|G^ｇ^L||||Y||||TOC^30^L^^臨時処方|||||||
RXR|PO|||||
ORC|NW|003200051030709||1|||||20200322163811|YAKU^日本電気^^^^^^^^L^I~エヌ イーシー^^^^^^^^L^P||99904^Ｄr.精神科^^^^^^^^L^I~ンdr.セイシンカ^^^^^^^^L^P|||||04^精神科神経科^L||||||||||||X|
RXE|||||||||||006549|||||||||||||||||||||||||||||
TQ1|1|14^Ｔ|3N123&分3　朝・昼・夕食後&L|||14|2020032218|||YCMN19^医師の指示どおり服用||||
RXO|201050^オキシメテバノール^MDCHOT^201000^メチコバール錠５００μｇ^L|3||T^Ｔ^L|01^散・錠・カプセル^MR9P^^^L|SOD^B^L^^一包化指定~DVD^0-0-0-0-0-0-0^L^^不均等指示~HCM^^L^^テスト処方箋コメント|||||42|T^Ｔ^L||||Y||||TOC^30^L^^臨時処方|||||||
RXR|PO|||||
ORC|NW|003200051030709||2|||||20200322163811|YAKU^日本電気^^^^^^^^L^I~エヌ イーシー^^^^^^^^L^P||99904^Ｄr.精神科^^^^^^^^L^I~ンdr.セイシンカ^^^^^^^^L^P|||||04^精神科神経科^L||||||||||||X|
RXE|||||||||||006549|||||||||||||||||||||||||||||
TQ1|1|14^包|3N123&分3　朝・昼・夕食後&L|||14|2020032218|||||||
RXO|^^MDCHOT^152350^ツムラ柴苓湯エキス顆粒:3.0g包^L|3||PO^包^L|01^散・錠・カプセル^MR9P^^^L|SOD^B^L^^一包化指定~DVD^0-0-0-0-0-0-0^L^^不均等指示~HCM^^L^^テスト処方箋コメント|||||42|PO^包^L||||Y||||TOC^30^L^^臨時処方|||||||
RXR|PO|||||
[EOF]
```

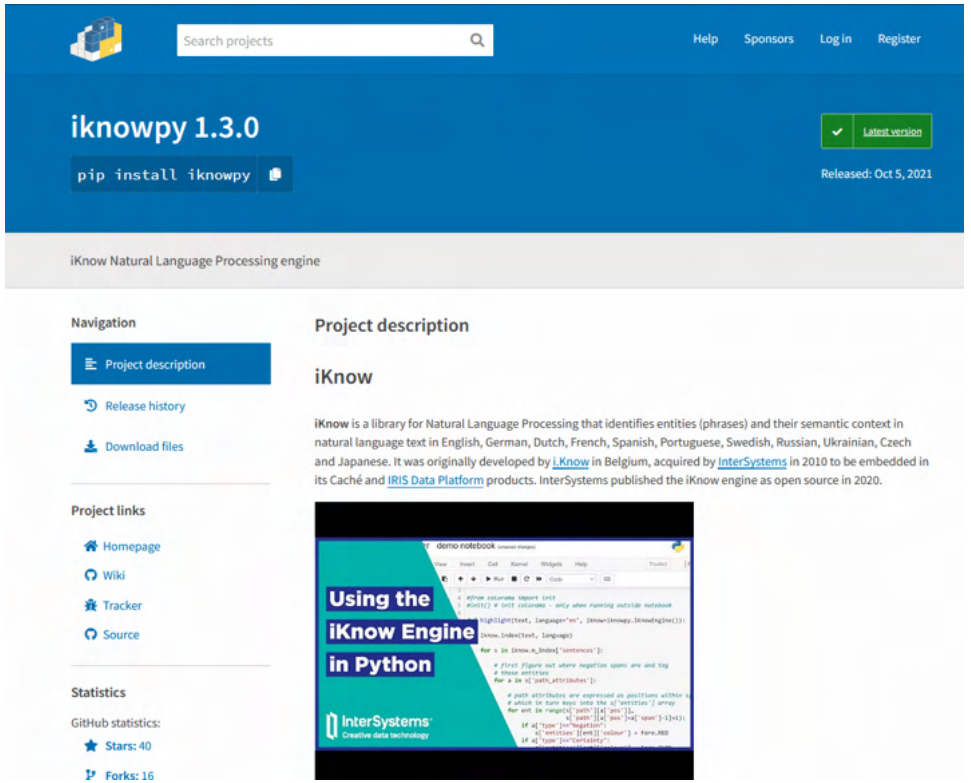

```
[
  "resourceType": "Bundle",
  "entry": [
    {
      "fullUrl": "Patient/6b47d3de-c9fb-43ea-b787-079edfd3345d",
      "request": {
        "method": "PUT",
        "url": "Patient/6b47d3de-c9fb-43ea-b787-079edfd3345d"
      },
      "resource": {
        "resourceType": "Patient",
        "birthDate": "1998-01-01",
        "gender": "male",
        "id": "6b47d3de-c9fb-43ea-b787-079edfd3345d",
        "identifier": [
          {
            "assigner": {
              "reference": "Organization/eb03b69b-8d19-4e3b-8f2a-9690f9fba1a4"
            },
            "value": "80006006"
          }
        ],
        "managingOrganization": {
          "reference": "Organization/eb03b69b-8d19-4e3b-8f2a-9690f9fba1a4"
        },
        "name": [
          {
            "extension": [
              {
                "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
                "valueCode": "IDE"
              }
            ],
            "family": "テスト　ユヤマＩＦ",
            "text": "テスト　ユヤマＩＦ　",
            "use": "official"
          },
          {
            "extension": [
              {
                "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
                "valueCode": "SYL"
              }
```

iknowpy (InterSystems NLP)とは

- **自然言語処理のうち、**
- **①分節化、**
- **②意味素（特にエンティティ）のラベル付け、**
- **③ドミナンス（文章中の意味素の主題を反映した相対値）計算**
- **を行うもの**
- **IRISの製品版と同等の機能**

（ただしデータベース格納などの機能はない）

Pythonライブラリとして使用する



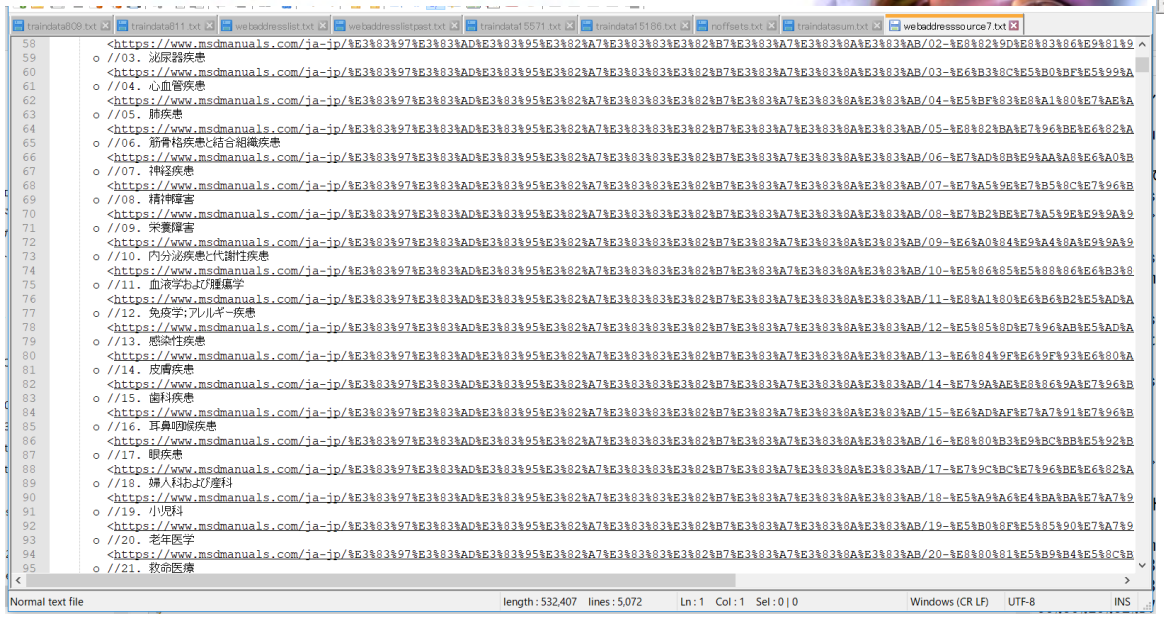
インストール： pip install iknowpy

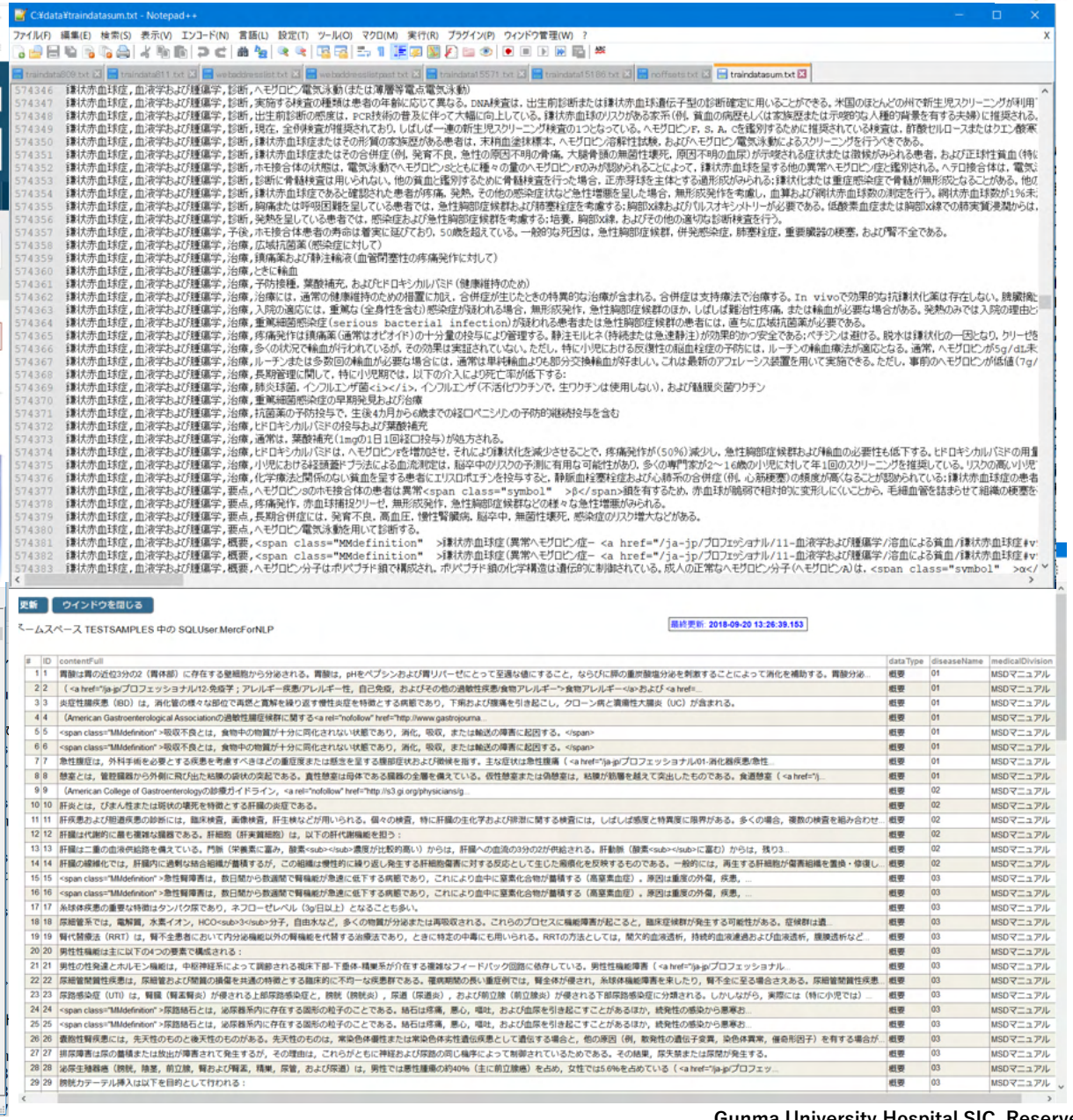
最新版は1.3.0

mecabは「丘の上の雲」を「丘」と「上」と「雲」に分解するとして、iknowpyは「丘の上の雲」を一つの意味素とする→前者の切り分けには含まれない「司馬遼太郎の有名な著書」という情報で切り出せる

→エンティティをベクトルとして扱った際、固有の値が出やすいと考えられる

PythonによるWebスクレイピング





メルクマニュアルを網羅的にWebクロールしNLPでプロセスした結果

糖尿病 Explore!

1300759

主要なエンティティ	frequency	dominance
治療		4.4088
症状		4.2838
genericdrug		3.6477
1日1回		3.6304
コルチコステロイド		3.4864
原因		3.3857
通常		3.3776
必要		3.2952

類似エンティティ		
糖尿病	7402	6870
糖尿病患者	1086	1040
糖尿病性腎症	503	359
糖尿病性ケトアシドーシス	358	322
糖尿病網膜症	354	311
糖尿病性神経障害	180	180
糖尿病のリスク	168	168
糖尿病合併症	129	129
糖尿病の既往	91	91
糖尿病患者の足のスクリーニング	86	86

Proximity Profile (近接性)	
例	118149
dm	77214
高血圧	59639
b	42136
a	40693
患者	39136
炭水化物代謝異常症/糖尿病	29357
a href	27956
脂質異常症	23206
代謝性疾患/糖尿病	21918

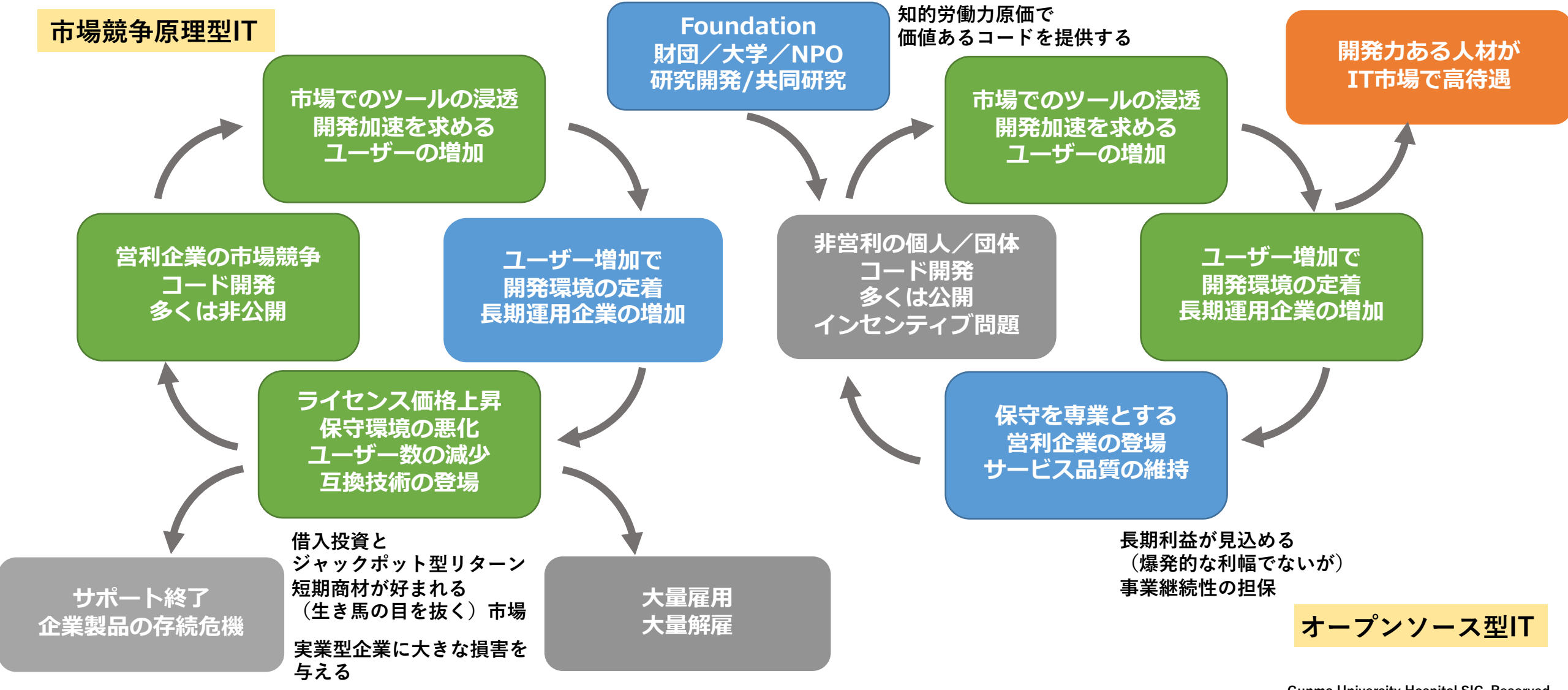
ソース　エンティティ・ベクター

1300096	:SQL:1300099:1300099	糖尿病 ,特に関連する慢性腎臓病がある患者の場合 ...
1299772	:SQL:1299773:1299773	副腎機能不全 , 糖尿病 ,蠕虫感染症 ...
1299759	:SQL:1299763:1299763	糖尿病 ...
1299552	:SQL:1299554:1299554	糖尿病 ...

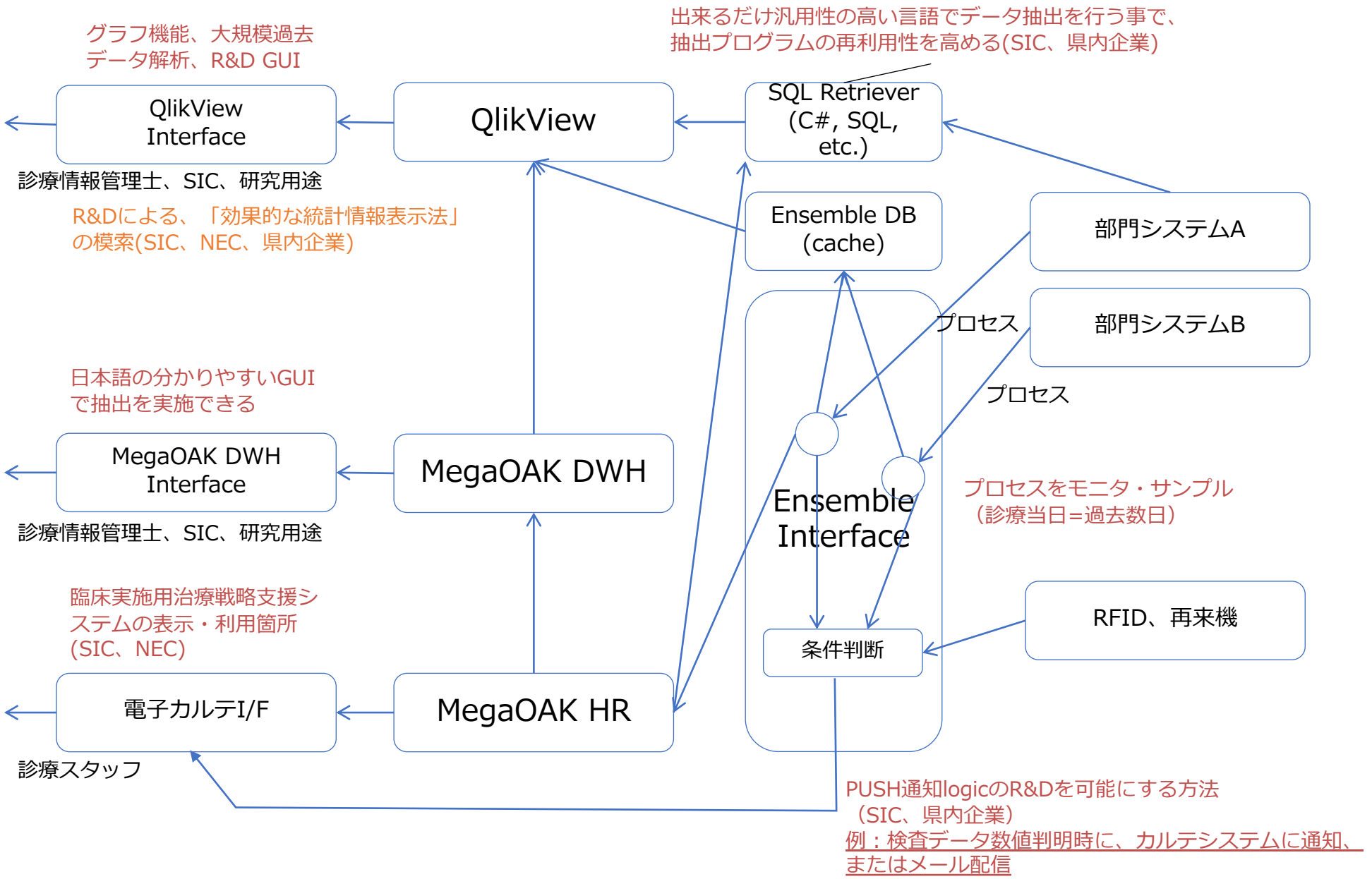
コードデモ

ITにおける有償市場の必要性と、オープンソースサイクルの必要性

- **過剰にインフレーションしやすい市場的性質**
- **医療ITでは信頼や継続性という「ITビジネス」の性質に衝突する特徴がある**



データベース観点からみたシステム接続概要



医療情報プロジェクトの方向性「本質を捉えて省力化し、危機に備える」

今日現在の危機とは何か
・労働者人口（人間の総労働可能時間）の減少によるサービス業における質の低下と安全性の低下
・分業、記録管理の増大による本質的業務時間の減少
・業務の専門化、分業化による、コミュニケーション（用語、手段、異職種の立場の相互理解）の困難化と円滑な遂行の困難化

医療情報分野における将来的な理想とは何か
・競争ではなく、協調によって維持される医療環境への貢献
・サスティナブルなワークサイクルでの医療の量および質、安全性の担保向上
・本質的でない労働中ストレスの低い職場環境の醸成
・労働者の自然なスキル向上を伴う、教育環境の整った職場環境

電子カルテの省力化：
HRを基本とし、大学病院向けは仮想化環境での提供を続ける
アルゴリズムとViewの分離→現場の困っている点を解決する点を探して実施
「Push（必要時お知らせ）の活用」→らくらく開発において埋め込んでおく
「以前のオーダーについて再送する」操作をスマートフォンで行う
医事-カルテ-物流マスタの連動、再編成→一般化アルゴリズムの開発（群大）

「次期大学病院発スタンダード」開発の核となる方針（今後5-10年の中で着手、段階的移行）

電子カルテ・ソフトウェア

「ノンカスタマイズ」移行（医療現場の独自性が統合されなければならない。ここにどう挑むか）
「ソフトのレイヤー移行」（大病院、中病院、小病院、クリニック）×（急性期・亜急性期・回復期）
「コンシェルジュ」移行（使う人に便利を提供する、必要時Push、頻用メモリ、マスタ再構成後AI）
「大更新を延期できる」運用への移行？（長期、何年に1回の更新だと、人材スキルが維持できる？）
「仮想化とハイパーコンバージド」「クラウド」「オンプレミス」の適応領域の明確化
ソフトウェア（AI,アルゴリズム）に十分な開発投資が回るための仕掛けづくり
海外向けにはクラウド＋スマホでのソリューション提供（維持管理を）
紙との共存領域の明確化（カメラ撮影後の手書き認識など）

医療ネットワーク

「レスポンス」化によるインフラ簡素化移行（仮想化、アンテナ削減）
「光ファイバー」＋「中継器」化移行（スイッチ、APを減らし、維持管理を容易に）
「スマホ長寿命化」（5-10年使える端末を目指す）
「送受信電波強化」（アンテナ少なくても使える、そういう用途に特化したスマホがあればいいのに）
「Wi-Fiビーム応用」（屋外通信技術を応用する、DENGYOなど）
「チャンネル整理移行」（できればシングルチャネル化）

機器、端末

シンクラ端末「買取」での長寿命化移行（ハードウェアを長く大事に使う方針）
耐久性、耐災害性、汎用性の高い端末の選定
テクノストレスを低下させる装置選択テスト（電子ペーパー、反射型ディスプレイ、音声、ジェスチャー、VR）

ハードウェア構成の種類と特徴

	オンプレミス	クラウド	仮想化＋ ハイパーコンバージド
現場での速度	◎	△	○
ストレージ許容 （10TBを超える）	◎	○	△
BCP能力	○	△	◎
ソフトウェアの 長期利用	△	○	◎
ネットワーク障害の 影響	◎	△	◎
多地点同時作業	△	◎	△
障害切り分け	◎	○	△
サイト被災害時の データ/システム保全	△	◎	○
保守要員の地域分散	○	◎	△
利用用途	レスポンスが 必要な業務（外来） 医療機器接続	データ送受信/保存/ プロセスの少ない用途 スマホの代替処理 地域連携	インターフェース サーバ、管理サーバ等 停止が許容されにくい 領域

ネットワーク構成の種類と特徴

	光ファイバー	有線メタル	Wi-Fi(2.4GHz)	Wi-Fi(5GHz)
通信距離	◎	○	△	▲
通信速度	◎	○	▲	△
取り回し	△	◎	○	○
価格 （コンバータ含）	▲	◎	△	△
ノイズ干渉	◎	○	▲	△
接続しやすさ	▲	△	○	○
用途	サーバ間接続 ファットクライアント	プリンタ シンクラデスクトップ	シンクラノート スマホ通信	スマホ通信 （速度が必要な場合）

端末構成の種類と特徴

	有線ファット デスクトップ	有線シンクラ デスクトップ	無線シンクラ ノートPC	スマートフォン
プロセス速度	◎	○	△	▲
ハードウェアとの接続性	◎	○	△	▲
ネットワーク障害時の 独立動作	◎	▲	▲	▲
故障時 メンテナンス性	▲	○	◎	○
管理性	▲	○	○	○
情報の分散性	▲	◎	◎	○
価格	▲	◎	○	△
画質	◎	○	△	▲
レスポンス性	◎	○	△	△
可搬性	▲	△	○	◎
スペース/電源	▲	△	○	◎
用途	外来端末 機器接続 検査など独立動作	病棟端末	病棟端末 ラウンド端末	PDA代替 オンデマンド 新しい情報手段

ご清聴ありがとうございました

Python+iknowpy…
IRISが変えるデータ分析

群馬大学医学部附属病院システム統合センター
鳥飼　幸太