データエクスチェンジ

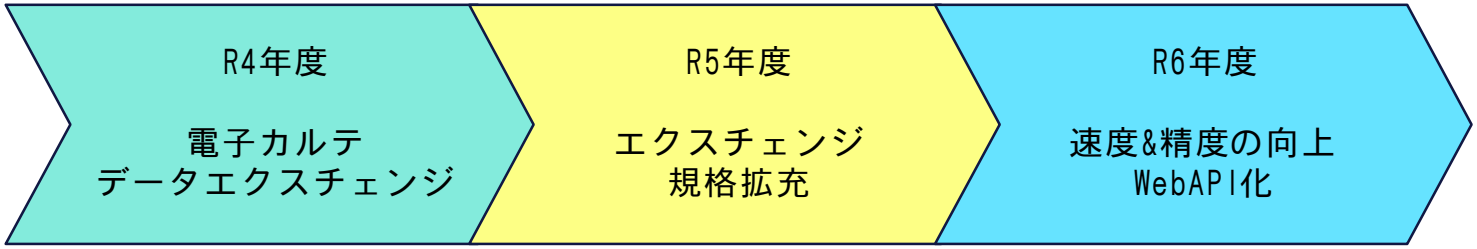
R4～6年度

2025.5.17

ラジエンスウエア株式会社
山田勝之

概要

令和４年度より先端的サービスの開発・構築に関する調査事業においてラジエンスウエアでは本事業のテーマ２電子カルテデータのエクスチェンジ部分を担当させて頂きました



対象医療情報規格

HL7-FHIR

MML

SSMIX(HL7 ver2.5)

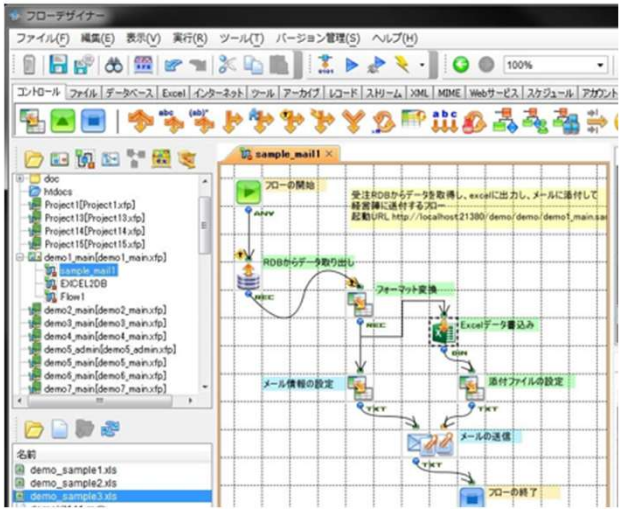
対象エクスチェンジデータ種別

患者基本・診断・処方・注射・検査結果の５種類

分類	HL7Ver2.5(SSMIX)	MML	FHIRリソース(JP Core プロファイル)
患者情報	ADT-00	mmlPi	Patient
診断履歴情報	PPR-01	mmlRd	Condition
検歴情報	OML-11	mmlLb	Observation_LabResult
処方箋	OMP-01	mmlPs	Medicationrequest.
注射	OMP-12	mmlInj	Medicationrequest-injection

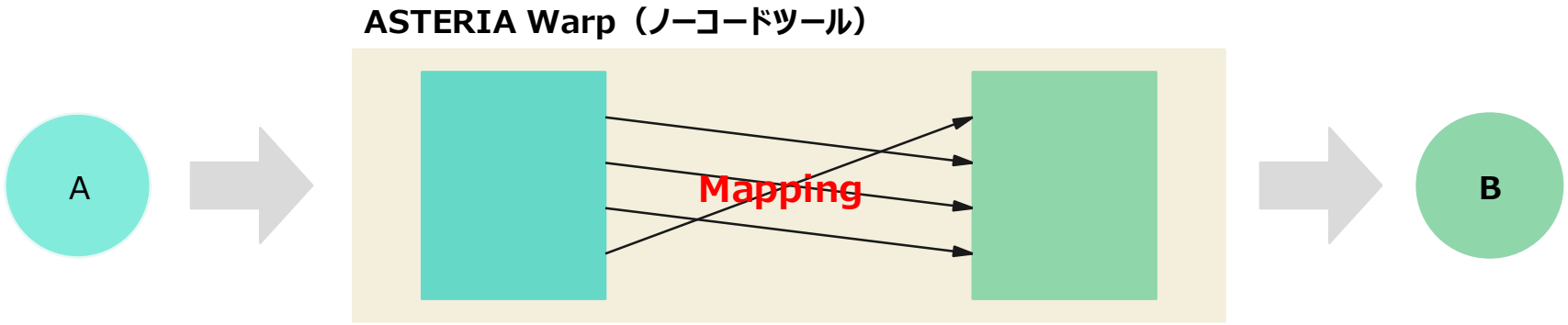
データエクスチェンジに用いたツール

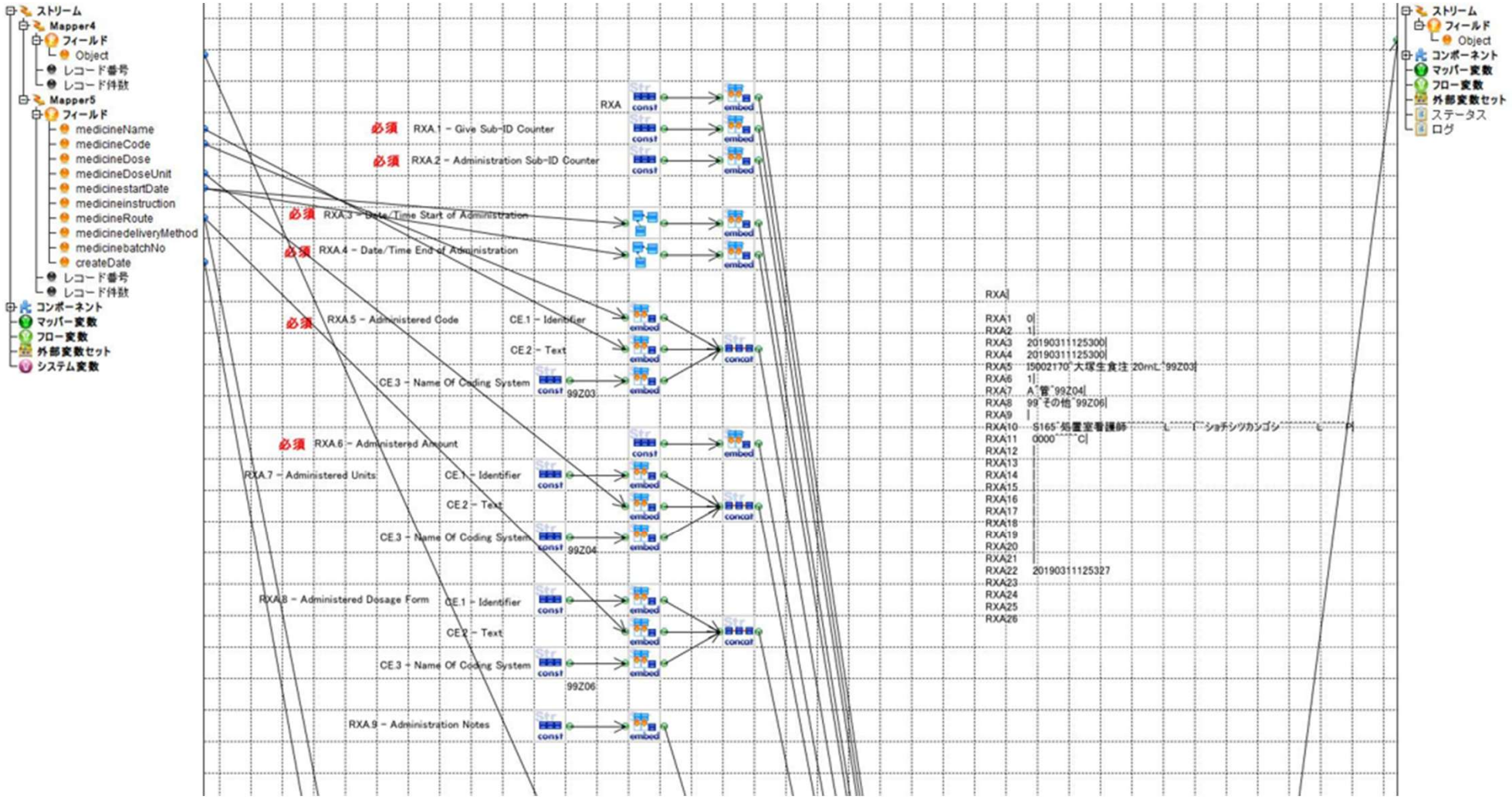
多様なデータ形式の取込やデータ項目の変化への対応を容易にする目的でノーコードツール（アステリア株式会社 ASTERIA Warp ）によるグラフィカルコーディングを用いた



エータエクスチェンジの基本動作

マッピング仕様を元にノーコードツールを用いて入力データをターゲットとするデータ規格に変換



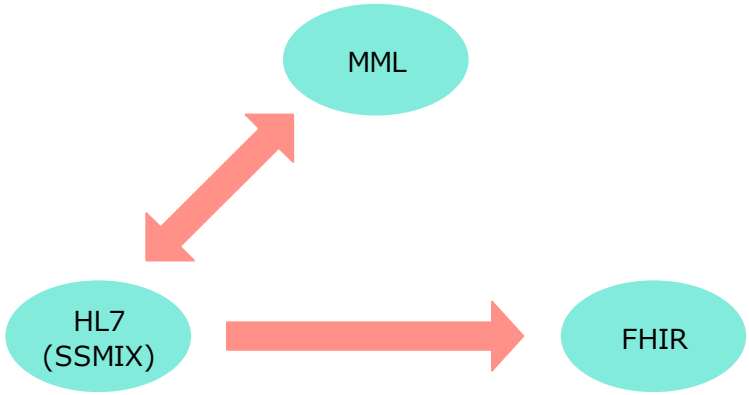

ストリーム
Mapper4
フィールド
Object
レコード番号
レコード件数
Mapper5
フィールド
medicineName
medicineCode
medicineDose
medicineDoseUnit
medicinestartDate
medicineinstruction
medicineRoute
medicinedeliveryMethod
medicinebatchNo
createDate
レコード番号
レコード件数
コンポーネント
マッパー変数
フロー変数
外部変数セット
システム変数
RXA
必須 RXA.1 - Give Sub-ID Counter
必須 RXA.2 - Administration Sub-ID Counter
必須 RXA.3 - Date/Time Start of Administration
必須 RXA.4 - Date/Time End of Administration
必須 RXA.5 - Administered Code
CE.1 - Identifier
CE.2 - Text
CE.3 - Name Of Coding System
99Z03
必須 RXA.6 - Administered Amount
RXA.7 - Administered Units
CE.1 - Identifier
CE.2 - Text
CE.3 - Name Of Coding System
99Z04
RXA.8 - Administered Dosage Form
CE.1 - Identifier
CE.2 - Text
CE.3 - Name Of Coding System
99Z06
RXA.9 - Administration Notes
RXA|
RXA1 0|
RXA2 1|
RXA3 20190311125300|
RXA4 20190311125300|
RXA5 15002170^大塚生食注 20mL^99Z03|
RXA6 1|
RXA7 A^管^99Z04|
RXA8 99^その他^99Z06|
RXA9 |
RXA10 S165^処置室看護師^L^^^ショチシツカンゴシ^L^^^^|
RXA11 0000^^^C|
RXA12 |
RXA13 |
RXA14 |
RXA15 |
RXA16 |
RXA17 |
RXA18 |
RXA19 |
RXA20 |
RXA21 |
RXA22 20190311125327
RXA23
RXA24
RXA25
RXA26
ストリーム
フィールド
Object
コンポーネント
マッパー変数
フロー変数
外部変数セット
ステータス
ログ

参考画面

令和4年度

データエクスチェンジパターン

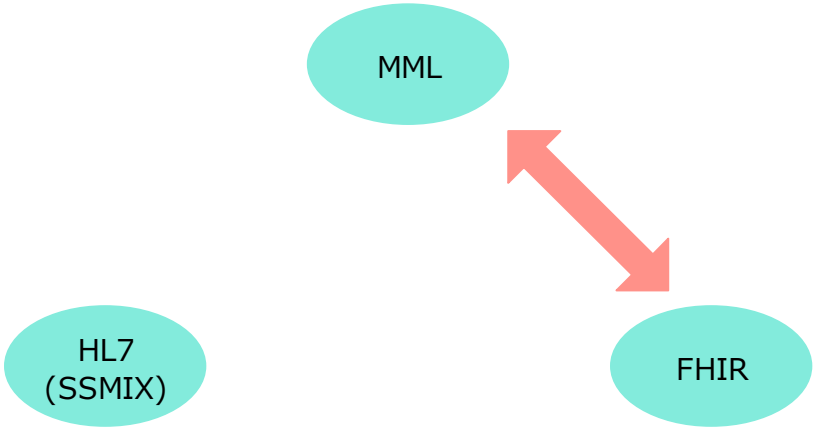
- HL7→HL7-FHIR
- HL7→MML
- MML→HL7



令和5年度

データエクスチェンジパターン拡充

- MML→HL7 FHIR
- HL7 FHIR→MML



R4,5年度エクスチェンジ

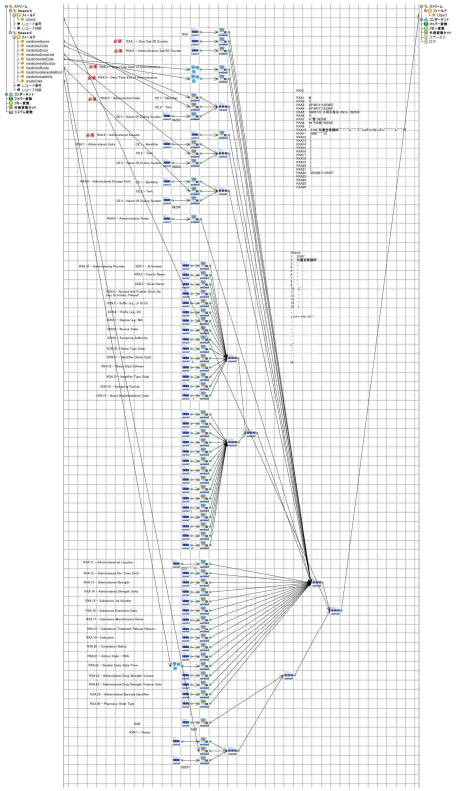
R4年度
ノーコードツールの基本機能による 単純なマッピングによるフォーマット変換
ノーコードツールで苦手な処理も工夫して実装 （ループや再起処理）

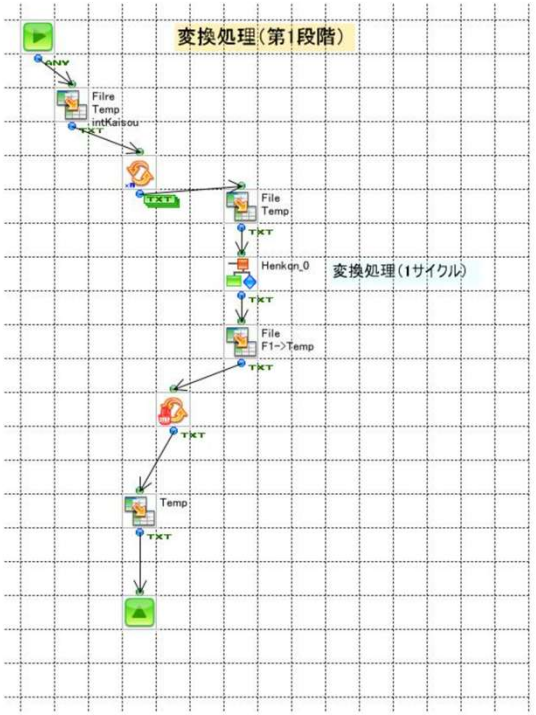
⇩

変換に時間がかかる フローの複雑化

R5年度
ノーコードツールでは不得意な部分をJavaインタープリタで処理及びJavaコンポーネント化
各データ規格を単純マッピングからテンプレート化しメンテナンス性の向上を図る

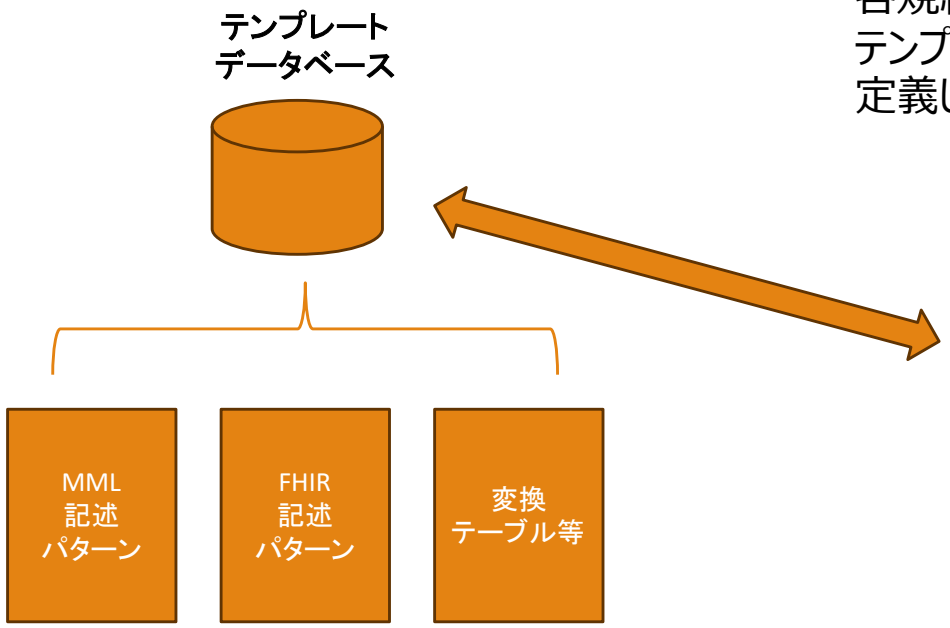
コンポーネント化





コンポーネント化によるフローの簡素化

テンプレート化



各規約のフォーマットの骨組み、変換の為の対応リストを
テンプレート部品化し、あらかじめローカルデータベースに
定義しておく

ノーコードツール
フロー中からデータベースにアクセスしテンプレートを取得する。 テンプレートの記述に沿ってエクスチェンジ処理を行う

テンプレートによる実際の変換処理イメージ
（例：患者基本情報のMML→FHIR変換）

FROMTO	MODULE	KEYWORD	

MMLFHIR	Patient	0_FIRST	{
			"resourceType": "Patient",
			"id": "%%＜/mml:Mml/mml:MmlBody/mml:MmlModuleItem/
			"meta": {
			"lastUpdated": "%%＜/mml:Mml/mml:MmlBody/mml:MmlI
			},
			"language": "ja",
			##＜GetTable,MMLFHIR,Patient,identifier＞##,
			##＜GetTable,MMLFHIR,Patient,name＞##,
			##＜GetTable,MMLFHIR,Patient,telecom＞##,
			##＜GetTable,MMLFHIR,Patient,gender＞##,
			##＜GetTable,MMLFHIR,Patient,birthDate＞##,
			##＜GetTable,MMLFHIR,Patient,address＞##
			}

Patient用MML-FHIR変換用のテンプレートを読み込み変換処理を開始

```
{
  "resourceType": "Patient",
  "id": "%%＜/mml:Mml/mml:MmlBody/mml:MmlModuleItem/mml:docInfo/mml:docId/mml:uid＞%%",
  "meta": {
    "lastUpdated": "%%＜/mml:Mml/mml:MmlBody/mml:MmlModuleItem/mml:docInfo/mml:confirmDate＞%%"
  },
  "language": "ja",
##＜GetTable,MMLFHIR,Patient,identifier＞##,
##＜GetTable,MMLFHIR,Patient,name＞##,
##＜GetTable,MMLFHIR,Patient,telecom＞##,
##＜GetTable,MMLFHIR,Patient,gender＞##,
##＜GetTable,MMLFHIR,Patient,birthDate＞##,
##＜GetTable,MMLFHIR,Patient,address＞##
}
```

入力ファイルの情報から、データベースより取得したコンテンツ情報をもとにしたテンプレートの構文を解析し変換する

行頭から順に処理をする

```
{
  "resourceType": "Patient",
  "id": "%%＜/mml:Mml/mml:MmlBody/mml:MmlModuleItem/mml:docInfo/mml:docId/mml:uid＞%%",
  "meta": {
    "lastUpdated": "%%＜/mml:Mml/mml:MmlBody/mml:MmlModuleItem/mml:docInfo/mml:confirmDate＞%%"
  },
  "language": "ja",
##＜GetTable,MMLFHIR,Patient,identifier＞##,
##＜GetTable,MMLFHIR,Patient,name＞##,
##＜GetTable,MMLFHIR,Patient,telecom＞##,
##＜GetTable,MMLFHIR,Patient,gender＞##,
##＜GetTable,MMLFHIR,Patient,birthDate＞##,
##＜GetTable,MMLFHIR,Patient,address＞##
}
```

変換元MMLより取得
埋め込み

テンプレートより取得
埋め込み

identifierのテンプレート

```
"identifier": [
  {
    "system": "urn:oid:1.2.392.100495.20.3.51.1##＜医療機関ID＞##",
    "value": "%%＜
/mml:Mml/mml:MmlBody/mml:MmlModuleItem/mml:content/mmlPi:PatientModule/mmlPi:un
iqueInfo/mmlPi:masterId/mmlCm:Id＞%%"
  }
]
```

R4,5年度課題

- 医薬品コード、検査コード、病名コード等、各コードシステムへ対する必要がある。

  R5年度の授業にてテーブル参照による動的な対象IDやコードの取得の仕組みはコンポーネント化し構築しており、実際の変換においてはこの仕組みを活用できる。
  ただし、具体的なコード値等については、継続的にテーブルの内容をメンテナンスしながら、精度の高いテーブルを維持していく必要があるため、このようなテーブルを共通的にメンテナンス・管理する体制づくりを並行して推進する必要がある。

- エクスチェンジ処理速度
  処方や検査など情報の数が複数になるデータについては数量に応じて処理時間も伸びる

- 変換元先のデータ粒度の違いによる不足データの補完をどこまで許容するかについて共通認識を形成しておく必要がある。

  PHRとの連携や、必要なリソースや文書単位で指定して取得できるようにする。など、項目不足時の暫定対応として、入力される固定値等についての共通的な認識を作ると共に、並行してPHR等から不足分の情報を取得できるような仕組みが必要である

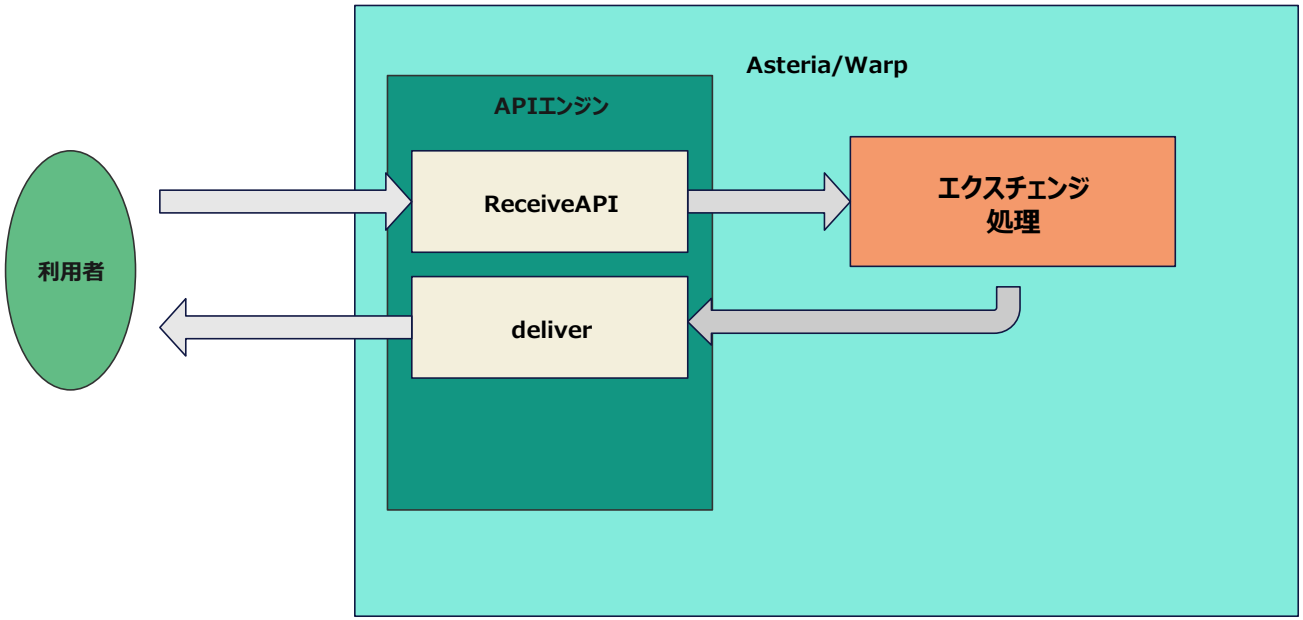
令和6年度

データ変換サービスの実装に向けた調査検討

- WebAPI化への対応
  APIフローへの接続と出力
- 変換精度の向上
  複数サンプルにより変換検証
- MML→HL7FHIR、HL7→HL7FHIR
  エクスチェンジフローをR4年度の単純Mappingからテンプレート方式へ変更
- 速度改善
  更なるコンポーネント化と処理フローの見直し

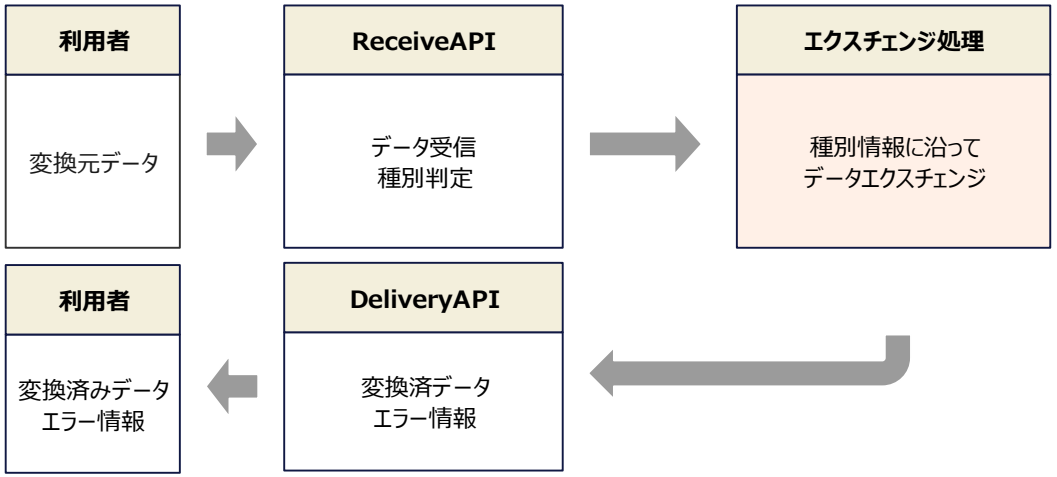
データ変換 WebAPI

利用者がWebから容易に扱えるRESTful APIとして提供



WebAPIとの接続

上位のWebAPIフローよりデータ種別と実データがエクスチェンジフローへ渡され、その情報に沿って
変換処理が行われる。



処理速度

プログラム実行時間の計測

患者、処方、注射、診断、検査のMMLからFHIRへの変換のプログラム実行時間

	ファイルサイズ (KB)	変換時間(秒)				
		1回目	2回目	3回目	平均	平均-(a)
患者	6.21	3.452	3.358	3.405	3.41	**1.11**
	13.3	3.374	3.343	3.390	3.37	**1.08**
処方	5.88	3.608	3.608	3.593	3.60	**1.31**
	19.7	5.296	5.343	5.514	5.38	**3.09**
注射	5.88	3.342	3.421	3.421	3.39	**1.10**
	15.8	7.045	7.405	7.436	7.30	**5.00**
診断	5.7	3.905	3.921	4.015	3.95	**1.66**
	39.6	7.545	7.498	7.670	7.57	**5.28**
検査	6.2	3.875	3.655	3.843	3.79	**1.50**
	51.1	85.651	87.682	88.136	87.16	**84.87**
	327	693.934	695.748	681.033	690.24	**687.95**
処理ファイルがない場合(a)		2.187	2.328	2.359	2.29	

主な課題

- **HL7FHIRに必須項目であって変換元データより取得ができないものへの対応**
- **連携するサービス・基盤で管理・付番するべき各種識別情報（患者ID、処方番号 等）**
- **各種コードシステムでローカルコードが付与される変換元データへの対応**
- **診断情報における診断状況（Condition.verificationStatus）の分類**
- **空データが設定されている場合の対応**
- **項目数が多いデータの変換時のエクスチェンジ処理速度**
- **元データに不正な形式でデータが設定されている場合の対応**

■HL7FHIRに必須項目であって変換元データより取得ができないものへの対応

FHIRリソースの一部項目は「必須（required）または強く推奨（must support）」とされており、以下のようなケースで変換元データ（SS-MIXやMMLなど）から取得できない項目が存在します。

リソース例	必須項目（FHIR）	変換元に存在しない理由	対応案
Patient	identifier.system	MML/SS-MIXにsystem情報が明示されない	定数として補完・OIDから推定
Practitioner	name	医師コードはあるが氏名が存在しないことがある	定数テーブルまたはID変換関数で補完
Observation	status, category	検査結果の項目値のみで、状態区分がない	FHIR仕様に基づき固定値設定
MedicationRequest	intent, status	MMLでは意図や状態の明示的定義がない	定数またはテンプレートで補完

■連携するサービス・基盤で管理・付番するべき各種識別情報（患者ID、処方番号 等）

連携処理においては、元データ（MMLやSS-MIX）に識別情報が含まれていない、または形式が不統一な場合がある。そのため、FHIR変換後の一貫性ある識別情報管理のために、以下のようなIDは連携基盤側で一元管理・付番すべきとされる。

管理・付番が必要な識別情報の例		
識別情報項目	用途	付番・管理すべき理由
患者ID	Patient.identifier など	MML内では施設ごとのローカルIDのため、一意性が担保されない
医療機関ID	Organization.identifier 等	施設名などでは不十分。OIDなど標準識別子が必要
処方番号	MedicationRequest.identifier	MMLでは明示的に存在しないケースもあり、連番で生成が必要
文書ID	各リソースの id、identifier	変換後の一意な管理のため、連携サービス側で発行するUUID等が必要
検査ID	Observation.identifier	MMLでは検査コードのみで識別できず、管理用の一意キーが必要
変換ジョブID	変換処理全体のトラッキング	ログ管理・トレーサビリティのため、基盤側での付番が望ましい

■各種コードシステムでローカルコードが付与される変換元データへの対応

MMLやSS-MIXなどの変換元データでは、診療行為・検査・診断名・薬剤などの項目に対して、標準コード（例：ICD-10、JLAC10、HOTコードなど）ではなく、ローカルコードが付与されているケースが存在する。
これにより、FHIRに変換する際には以下のような課題が生じる：
FHIRは標準コード体系（SNOMED, LOINC, RxNorm等）を前提、ローカルコードでは他システムとの連携性が確保できない（code.coding.system に適切なURIやOIDが必要）

ローカルコードが付与されている変換元データに対しては、
WARPテンプレート上で TableConvert 関数を利用して標準コードへ変換し、
補完が必要な場合は定数設定や事前定義ファイルで対応する。

■診断情報における診断状況（Condition.verificationStatus）の分類

FHIR仕様における verificationStatus の意味	
Condition.verificationStatus は、診断情報がどのような確実性・判断段階にあるかを示す属性で、FHIRでは以下のように分類されます：	
verificationStatus 値	意味
provisional	暫定（仮診断）
differential	鑑別診断
confirmed	確定診断
refuted	否定された
entered-in-error	入力ミス
unconfirmed	未確認（FHIR R4で非推奨）

MMLでの診断（<pastIllness> や <problem> など）は「確定/暫定」などの状態情報は必ずしも明示されない。
「confirmed」や「provisional」などへのマッピングが必要

MMLやSS-MIXの診断情報には診断状況が明示されないケースがあるため、
FHIR変換時には変換テーブル・定数補完・テンプレート関数を用いて、
Condition.verificationStatus（確定/暫定など）を分類・付与する必要がある。

■空データが設定されている場合の対応

MML や SS-MIX などの変換元データでは、以下のような「空データ」（未入力・空文字・null）が存在します
FHIRで必須項目であるにもかかわらず、元データに値が存在しない
FHIR変換時に、空データをそのまま出力するとバリデーションエラーの原因になる
空値の意味を明確にしないと、他システムとの連携で意味的欠損が発生

空文字（""）のケースは "unknown" に自動補完
これにより、FHIRで必須の列挙型項目に「空」のまま出力されないよう対処

■元データに不正な形式でデータが設定されている場合の対応

MMLやSS-MIXなどの変換元データでは、以下のような不正形式のデータが含まれているケースがあります
日付項目に「20231」「999999」など不正な形式の文字列
数値項目に全角文字や記号混入（例：「１２３．０」や「5mg(1錠)」）
コード体系に存在しない値（ローカル定義で標準に準拠しない）
デリミタ誤り、桁数不足、ゼロ埋め不足などの形式逸脱
これらは、FHIR変換時の構文エラーや、意味的な誤解、連携先での受け取り拒否などにつながるため、事前の検知と補正が必要

元データに不正な形式が含まれている場合、テンプレートやJavaコンポーネントの各関数により、
可能な限り補正・置換・既定値補完などの多層的な対応が取られる構成となっています。
これにより、FHIR変換後の品質を維持しつつ、バリデーションエラーや業務影響を回避仕組みを用意する

■項目数が多いデータの変換時のエクスチェンジ処理速度

MMLやSS-MIXからHL7 FHIR形式への変換において、以下のような状況では処理パフォーマンスの課題が発生する

- 1患者あたりの診療データ件数が非常に多い（例：診断・処方・検査など）
- リソース単位（Condition、Observation、MedicationRequest 等）で多数のFHIRリソースが生成
- テンプレートやコード変換が大量に実行され、逐次処理によって遅延

項目数が多い患者データの変換時には、**テンプレート処理・外部参照・Java関数呼び出し**の頻度が処理速度のボトルネックになります。
R6年度にはJava関数導入・定数補完・処理の単純化により、**エクスチェンジ処理の高速化・安定化を図り速度向上を図っている**

まとめ

いずれのエクスチェンジについても
「構造の違い」「情報の欠損」「標準コード未対応」「処理負荷」
の課題が存在します。

令和４～令和6年度にかけてのエクスチェンジへの取り組みの中で変換テンプレート、関数化、定数補完、ID発番、コード変換表テーブルなどの仕組みにより、これらの課題に対応できるフレームワークを用意してまいりました。

データのエスチェンジの運用、実装に向けては実際のコード変換テーブルの内容やID取得の方法などを整理、エクスチェンジの際のルールの取り決めが必要になると考えております。

ご清聴ありがとうございました。