Ruby による ADL parser の実装

立川察理 [1,2]

[1] 東京大学医学部附属病院企画情報運営部

[2] 成増厚生病院

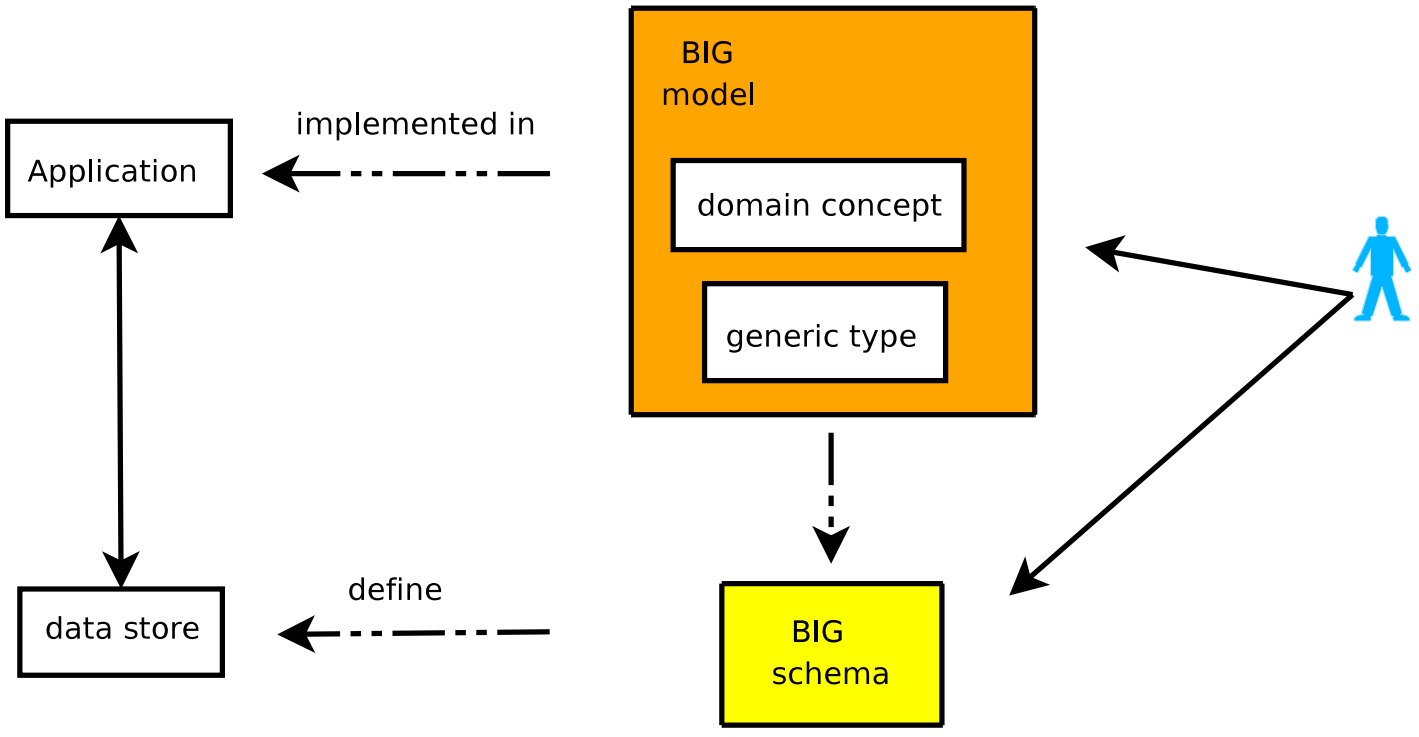
2009 年 10 月 31 日

openEHR の特徴

- openEHR は、オープンなライセンス Open Lisence である。
  - Free SoftwareFoundation GNU General Public Licence (GPL)
  - Lesser GNU Public Licence(LGPL)
  - Mozilla Public licence (MPL)
- openEHR は 2 段階モデル・アーキテクチャ two model approach を採用している。
  - 情報参照モデル Information Reference Model
  - アーキタイプ・モデル Archetype Model

openEHR の特徴 - two model approach -

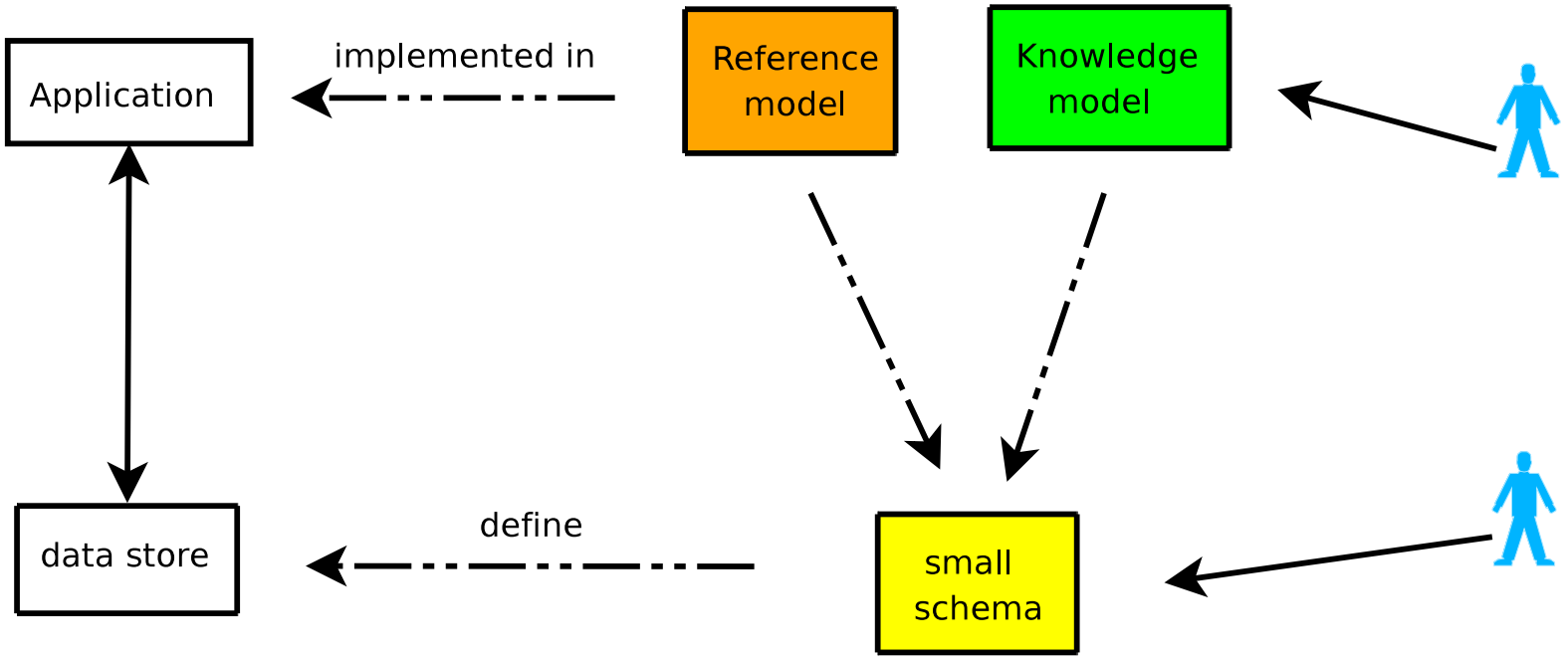
単一モデル・アーキテクチャ single model approach では、データ型やドメイン上の概念がひとつのモデル内に包含される。



- 低レベルの要素と高レベルの概念が同一モデル中に混在する
- モデルが開発時のビジネスモデルしか反映してない
- 相互運用が困難である

openEHR の特徴 - two model approach -

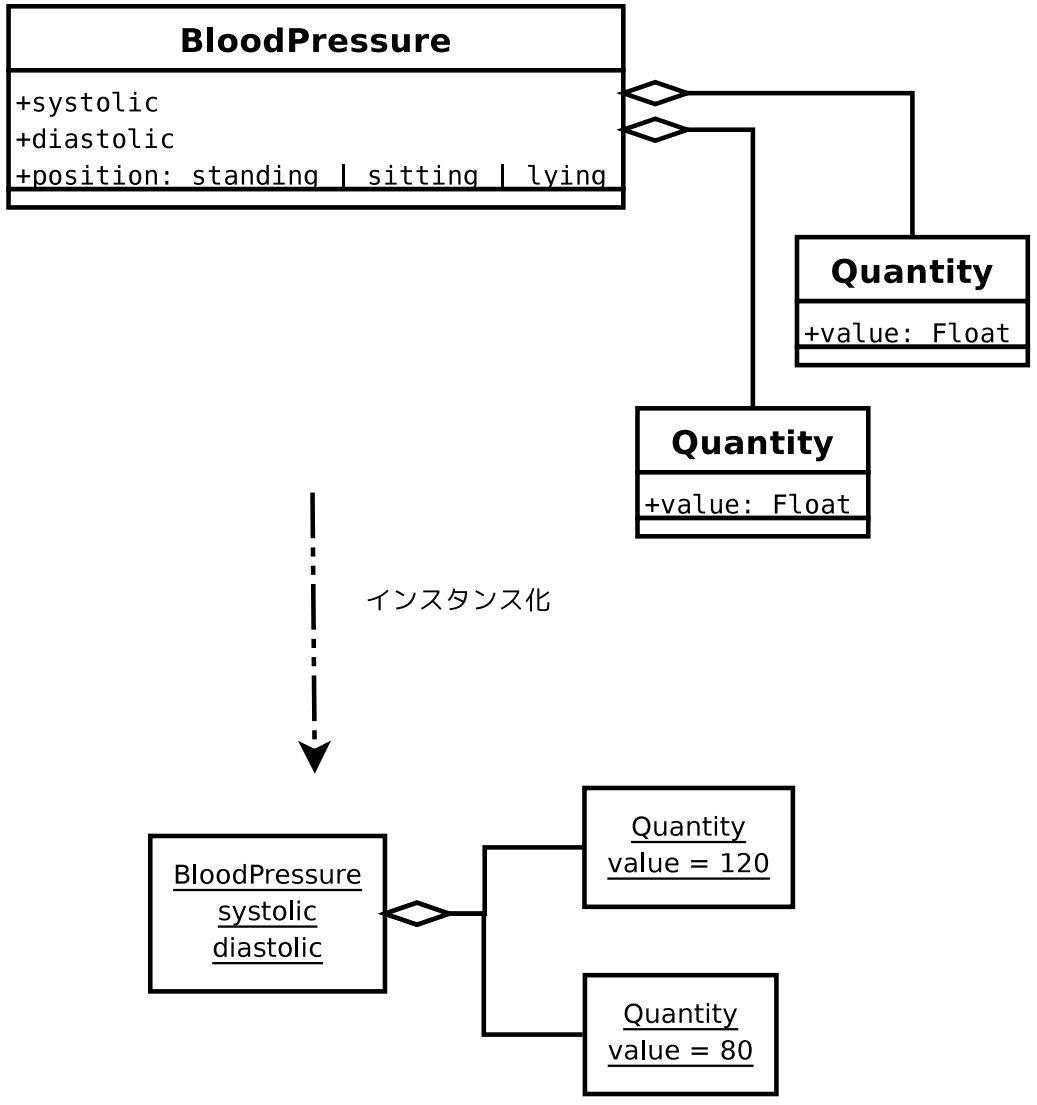
openEHR の 2 段階構成のアーキテクチャとは、情報 information と知識 knowledge を分割することである。



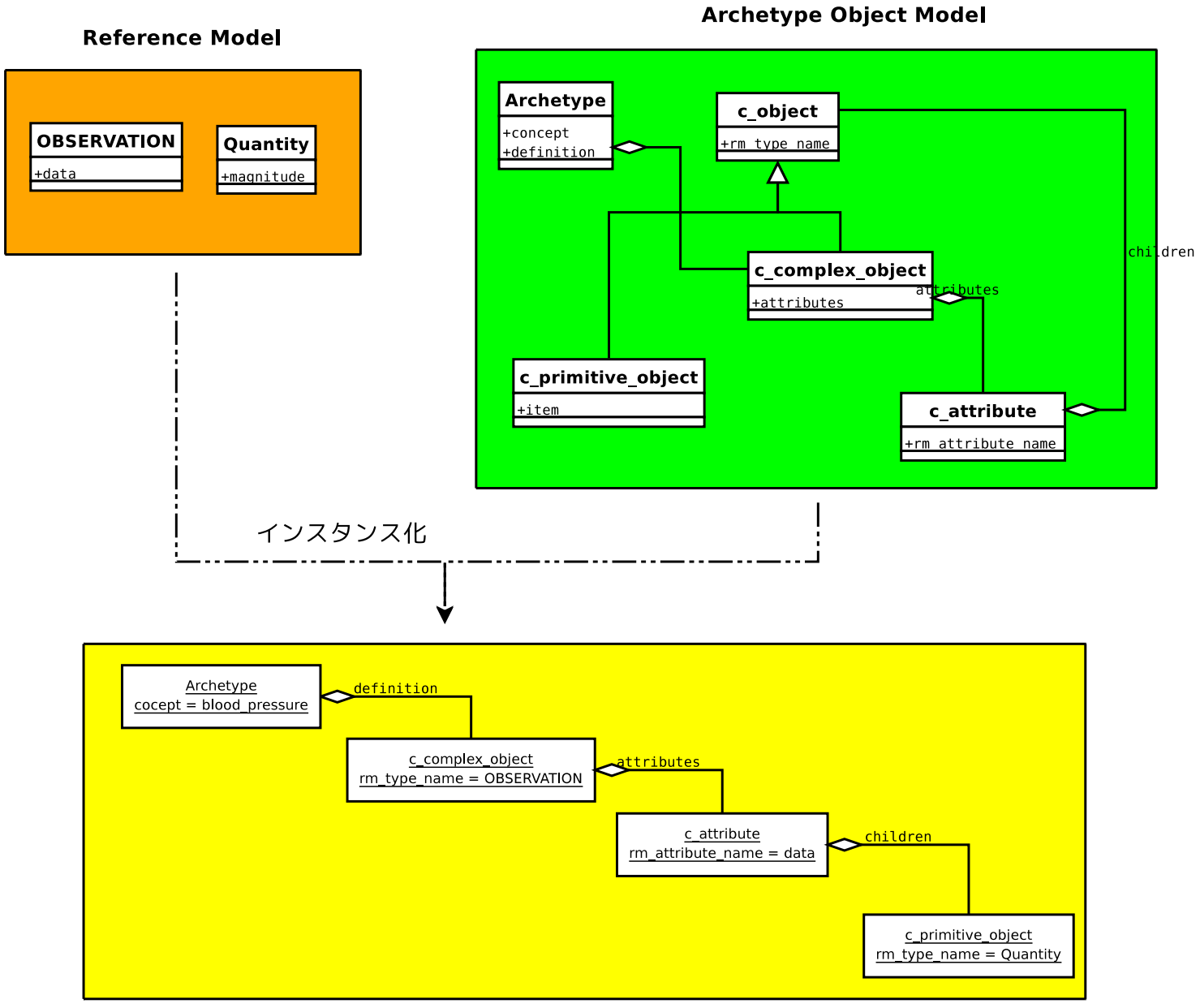
- 低レベルの要素を組み合わせて高レベルの概念を構築する
- 概念を動的に生成できるのでビジネスモデルの変遷に追随できる
- 相互運用に利用しやすい

openEHR の特徴 - two model approach -

単一モデル・アーキテクチャでは巨大な静的モデルからインスタンスが生成される。

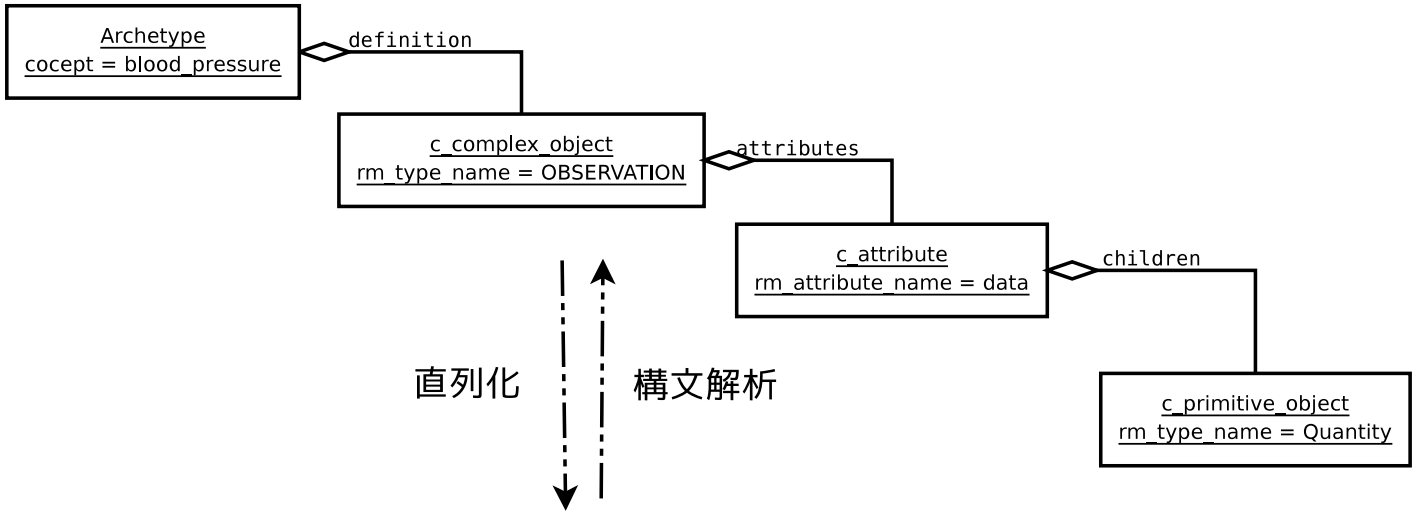


openEHR の特徴 - two model approach -



openEHR の特徴 - two model approach -

Archetype Definition Language, ADL とは、アーキタイプの直列化表現のひとつ。すなわち、アーキタイプを言語として表現した形式である。




```
Archetype
concept = <blood_pressure>
definition
  OBSERVATION[at0004] maches {
    data maches {
      Quantity <
          magnitude = <0.0..500>
      >
    }
  }
```

ADL 概説

言語は以下の 2 つのレベルに分割される。

- 構文論
  言語が想定する語の並び (文法) を規定する。
- 意味論
  計算機によって処理される内容を規定する。

Ruby での例

```
"代入変数" = 2 + 3        # 文法的に不正

var = 1 + undefined_var # 文法は正しいが、意味的に不正
```

構文論 - セクション -

ADL の構文は大きく 4 つのセクションから構成される。

- header セクション
  - archetype
  - specialize
  - concept
  - language
- description セクション
  作成者等のメタ情報を記述する。
- definition セクション
  アーキタイプの制約を記述する。
- ontology セクション
  アーキタイプ内で利用される語彙を定義する。

構文論 - 2 つの文法 -

ADL はセクションによって 2 つの文法を使いわける。

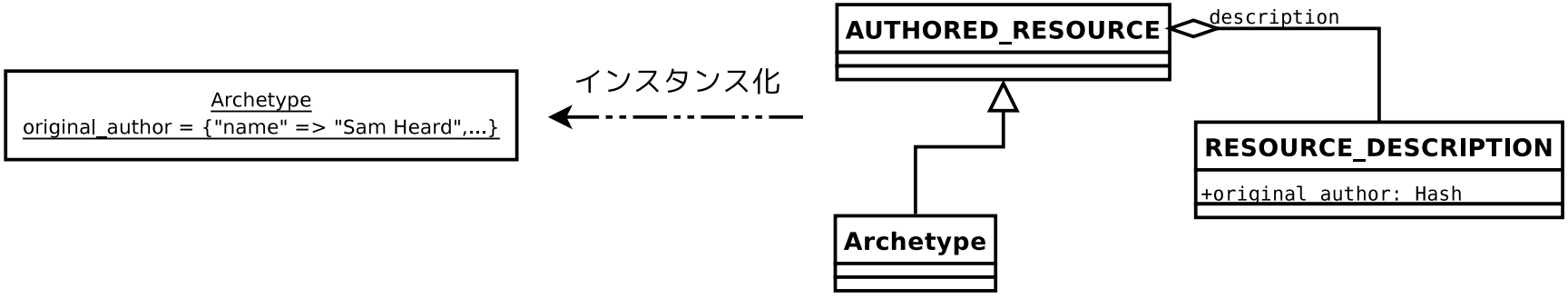
- dADL
  definition 以外の全てのセクションの記述に用いる。情報モデルに基づいたインスタンス・データを形式的に表現する。
- cADL
  definition セクションの記述に用いる。オブジェクト指向情報モデルの方法でデータに対する制約を記述し、アーキタイプを形式的に表現する。

構文論 - dADL -

情報参照モデルに基づき、そのインスタンスを生成する。

dADL

```
description
    original_author = <
        ["name"] = <"Sam Heard">
        ["organisation"] = <"Ocean Informatics">
        ["date"] = <"17/04/2006">
        ["email"] = <"sam.heard@oceaninformatics.biz">
    >
```



構文論 - **ontology** セクション -

ontology セクションにて語彙を規定する。

dADL - ontology セクション -

```
terminologies_available = <"SNOMED-CT", ...>
term_definitions = <
    ["en"] = <
        items = <
            ["at0000"] = <
                description = <"...">
                text = <"Blood pressure">>
            ["at0004"] = <
                description = <"...">
                text = <"systolic">>
            ["at0005"] = <
                description = <"">
                text = <"diastolic">>>>>
term_binding = <
    ["SNOMED-CT"] = <
        items = <
            ["at0004"] = <[SNOMED-CT(2003)::163030003]>
            ["at0005"] = <[SNOMED-CT(2003)::163031004]>>>>
```

構文論 - cADL -

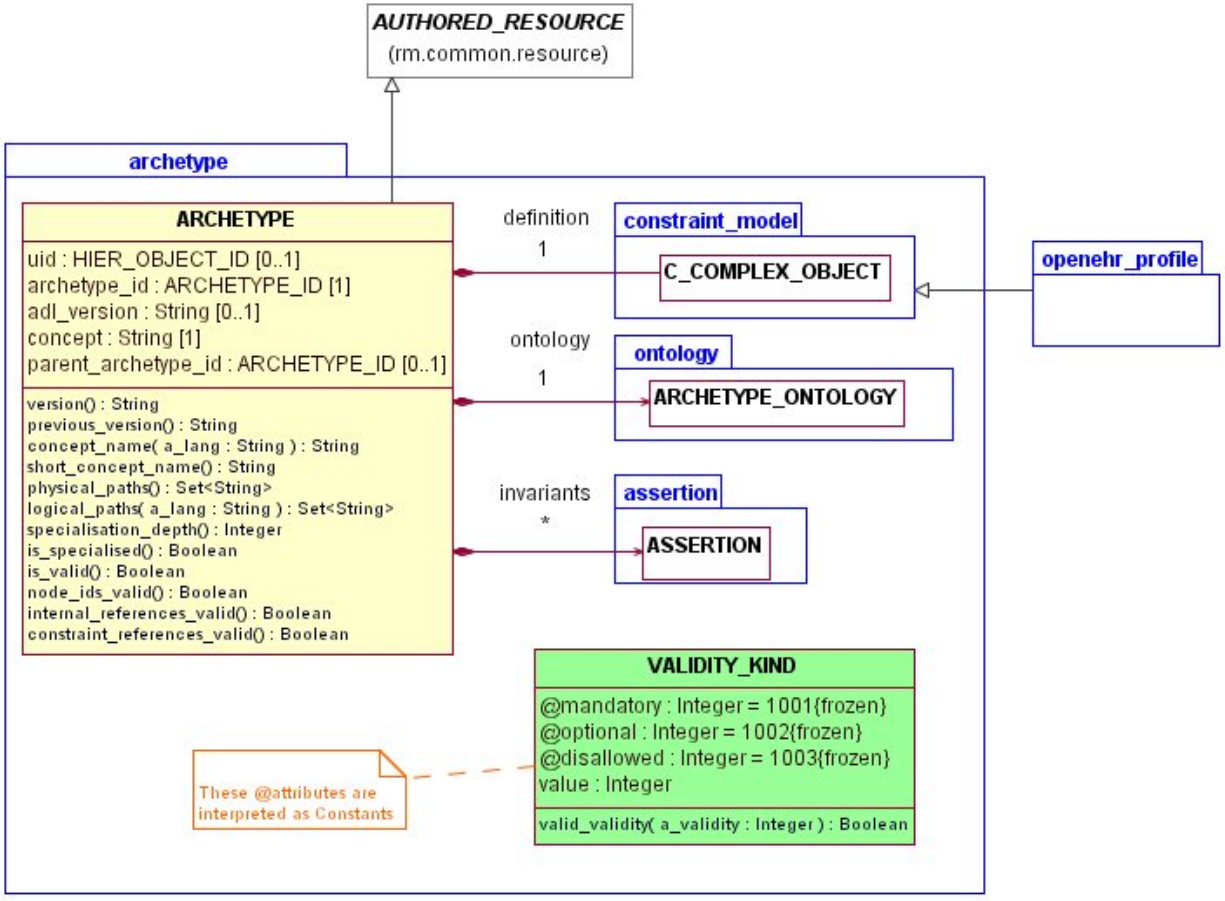
構文中ではモデルの属性名と値が交互に出現し、制約を表現する。

cADL - definition セクション -

```
OBSERVATION[at0000] matches {   -- Body weight
  data matches {
    HISTORY[at0002] matches {       -- history
      events cardinality matches {1..*; unordered} matches {
        EVENT[at0003] occurrences matches {0..*} matches {
          data matches {
            ITEM_SINGLE[at0001] matches {   -- Simple
              item matches {
                ELEMENT[at0004] matches {       -- Weight
                  value matches {
                    C_DV_QUANTITY <
                      property = <[openehr::124]>
                      list = <
                        ["1"] = <
                          units = <"kg">
                          magnitude = <|0.0..1000.0|>>>>
                }}}}}}}}}}
```

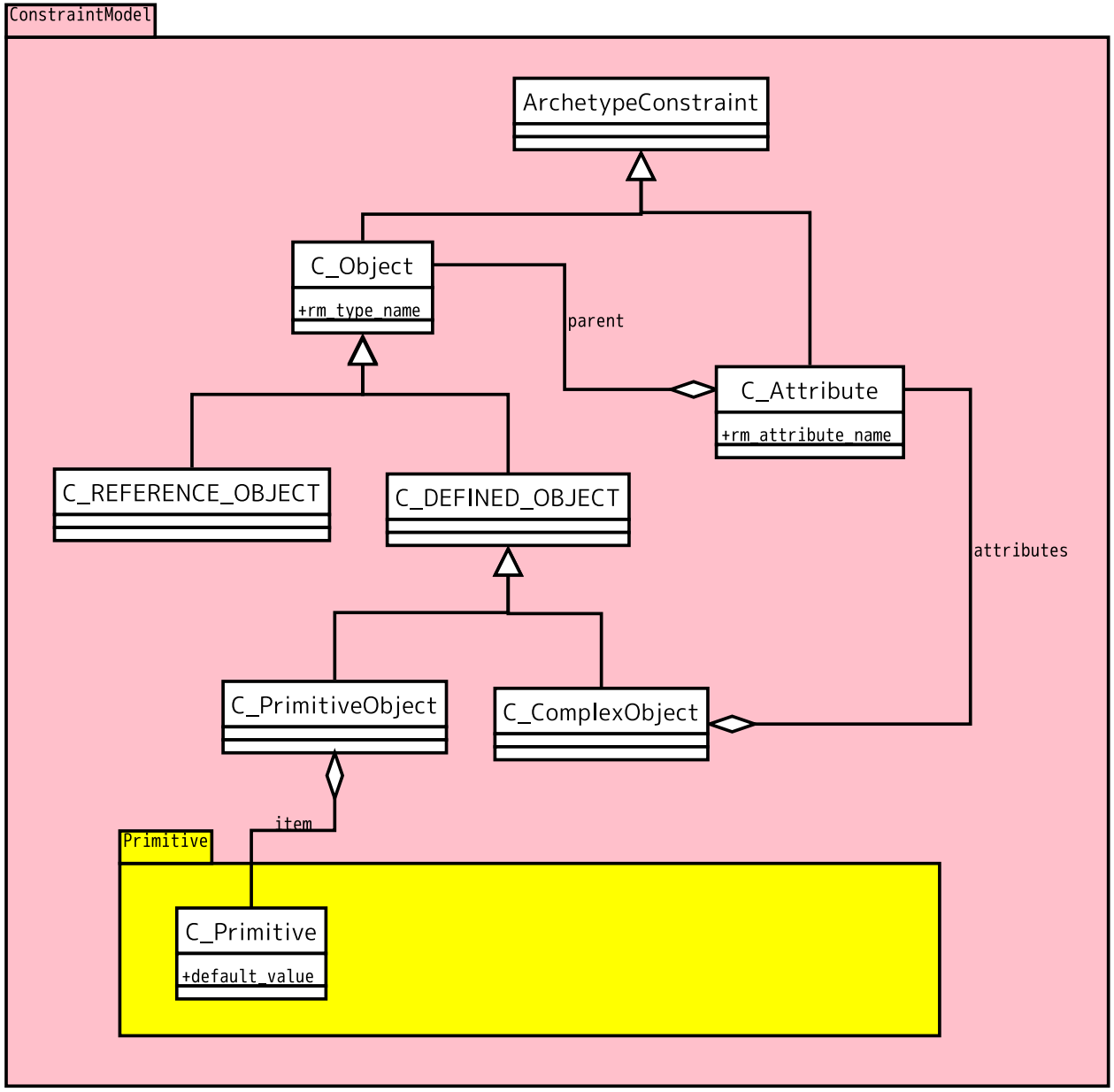
意味論 - Archetype Object Model -

Archetype Object Model(AOM) は、アーキタイプの意味論をオブジェクト指向モデリングの手法で規定する。



意味論 - Constraint Model -

Constraint Model が階層制約を規定する。



意味論 - 検証 -

ADL の意味関数は、AOM のインタンスを生成し、それに基づいて各種の制約を検証する。

- AOM インスタンス生成の過程で AOM の構造をチェックする
  適切な識別子があるか、必須セクションがあるか、ontology セクションで定義された語彙との整合性など。
- AOM インスタンスに基づいてアーキタイプをチェックする
  - アーキタイプ参照の検証 (internal_ref,archetype_slot)
  - 特化 (specialization)
- AOM インスタンスに基づいてデータをチェックする
  - オブジェクト階層のチェック
  - データ制約のチェック

Ruby 版 ADL パーサの現状

- ADL 1.4 を対象とする
- 構文論
  現時点で
  http://www.openehr.org/svn/knowledge/archetypes/dev/adl/openehr/
  以下、総計 315 個の ADL をパース可能。
- 意味論
  - 参照モデルは openEHR 参照モデルのみ
  - 制約の検証は現時点ではほとんど未実装である
    各 AOM クラスに意味関数を実装し、AOM インスタンス上で再帰下降的に実行するという設計を考えている。

サンプル

```
require "lib/adl_parser.rb"
parser = ::OpenEhr::ADL::Parser.new
adl =  File.read("test/adl/
                  openEHR-EHR-OBSERVATION.blood_pressure.v1.adl")
archetype = parser.parse(adl)
```

ADL の意義

すでに OWL で実現可能なものを、さらに別の構文や意味論が必要か？

- XML よりも記述が簡潔で可読性が高い。
  参照モデルの型の省略、終了タグの不在。
- 多言語や標準用語集への対応。
  ontology セクションでの語彙の分離による。

少なくとも Ruby での開発に関して述べれば、OWL を処理する実用的なライブラリーがない現状では、それを実装するよりも ADL を処理するライブラリを開発するほうが実現可能性が高い。

応用

主に Rails のプラグインとしての利用を想定している。

- EHR 間相互接続
  EHR データとそれに対応するアーキタイプの送受信によって EHR 間での意味論的統合を可能とする。
- 意思決定支援システムと EHR との接続
  意思決定支援システムと複数の施設間とを接続させる手段とする。

今後の展望

- Medinfo 2010 に合わせて、プロジェクト最初のメジャーリリース (Capetown release) を出す。
- ここで、完全な ADL パーサー、すなわち AOM オブジェクト生成、アーキタイプ制約の検証、ならびに ADL 直列化などを実現したい。

ありがとうございました