openEHR in action!

Part 0

Description: Introduction of the use case, the tutorial's goals and the steps to be done.
Expected duration: 15 min
Activity type: tutor's introduction

Source data: a cancer registry .csv dataset.

Steps

1. Explore the dataset
2. Find out and take note of the cancer type(s) that are covered by your Registry and how they are codified

   Terminology: << >>
   Code(s): << >>
   Description(s): << >>

Part 1

Description: Extend the template and semantically describe some nodes with ontologies representing the correct concepts.
Expected duration: 30 min
Activity type: individual work

Assumptions

A1. Suppose you have already created a conceptual model from the source dataset. The model is formalised with a mindmap that can be seen in Figure 1 and downloaded here (.xmind file can be viewed with Xmind software).

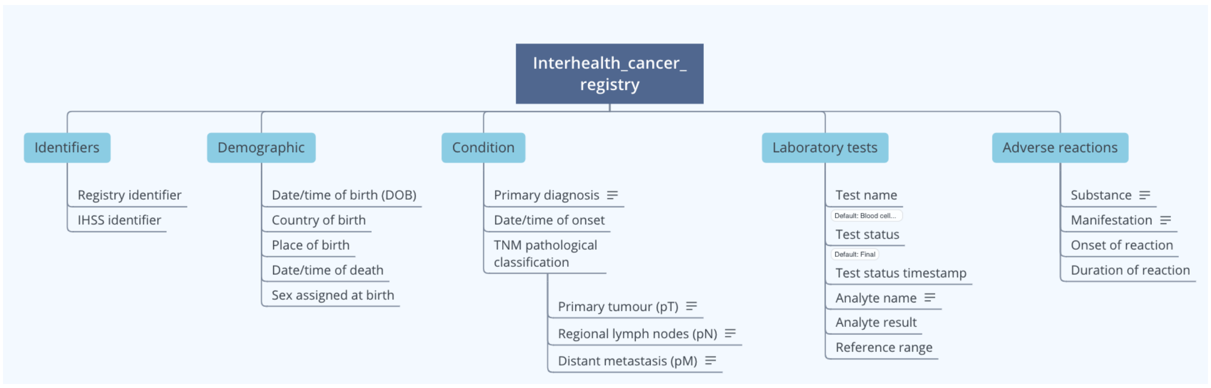

Figure 1

A2. Assume you have already started building the template and need to refine it in some parts. The archetype set and preliminary template can be downloaded here (the .zip file).

Steps

1. Access to the openEHR Archetype Designer (AD) using an account of your choice between GitHub, Microsoft or Google.
2. Create a new repository locally.
3. Import the template set.
4. Customise the template:
    a. Add the **date of death**:
        i. Search and download the archetype “Death Summary” from the Clinical Knowledge Manager (CKM).
        ii. Import the archetype in the repository of the AD.
        iii. Add the archetype “Death Summary” in the Demographic section.
        iv. Hide[1] all the elements that are not relevant.
        v. The date we have in the data is a point in time, not an interval: correct possible datatypes for this element.
    b. **Code the diagnosis** with ICD-10 terminology:
        i. Use Bioportal to search among the ICD-10 classes for the code that corresponds to the type(s) of cancer treated by your registry (e.g., ‘Malignant neoplasm of thyroid gland’ corresponds to C73).
        ii. Add the codes to the data node as `External codes > Local terms`
    c. Set the possible values of **Distant metastasis (pM)** node among the list here:[M0, M1, M1a, M1b, M1c, MX].
    d. Code the Analyte name using LOINC terminology following the dictionary:
        - LP7720-8 White blood cells
        - LP7536-8 Red blood cells
        - LP32067-8 Hemoglobin
        - LP70360-0 Platelets
        - LP418019-8 SARS coronavirus 2 Ag

Part 2

Description: Create openEHR compositions and store them in EHRBase
Expected duration: 15 min
Activity type: tutor’s demonstration

Assumptions

A1. Due to time limitations, this step is not performed during the tutorial but only described. Besides, students are provided with the tools to carry it out independently later on.

[1] **hint**: An element is hidden if it has Zero occurrence

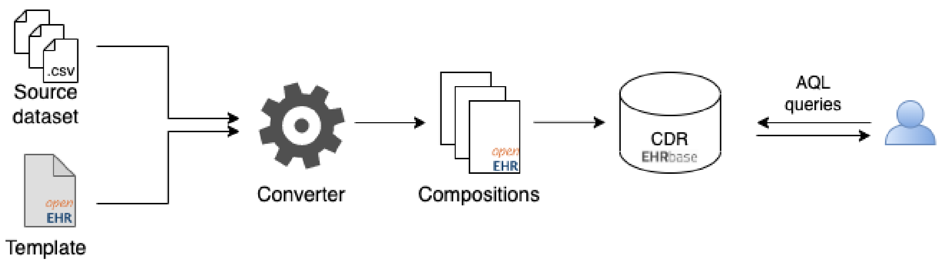

Figure 2

In particular, we have already performed this step using:
- A complete version of the template, downloadable here.
- The FLATEHR converter, an open-source software and a script to upload the compositions available on GitHub.
- An EHRBase instance as Clinical Data Repository (CDR)

Part 3

Description: Interact with EHRBase
Expected duration: 40 min
Activity type: individual work

Set up

Each Registry can access a different instance of the EHRBase containing the compositions obtained from the conversion of its source dataset. Connection parameters will be provided to each group.

EHRBase[2] can be used via the latest version of the openEHR REST API to which the backend conforms and data can be queried through model-based queries expressed using the Archetype Query Language (AQL).

Try interacting with your CDR in two ways:
- using an API platform, Postman in our case;
- using a frontend application, openEHR tool in our case.

Steps

Depending on time and your preferences, **you can choose** from which of the two methods to start.

1. Use Postman to interact with EHRBase for retrieving a composition
    a. To set up the software, import the collection containing some REST calls already written and the environment with the global variables (the .json files found here).

[2] EHRBase documentation: https://ehrbase.readthedocs.io/en/latest/index.html

b. In the Environment, modify the values of the following global variables to the proper ones:
   `“REST_API_ADDR”`
   `“REST_API_PORT”`
   `“TEMPLATE_ID”`

c. Try yourself some `GET` requests.

d. Try to retrieve a composition, perform the following steps using the available requests:
   i. Get the list of available templates.
   ii. Set properly the value of the `“TEMPLATE_ID”` variable in the environment.
   iii. Retrieve[3] all the composition for the specific template id.
   iv. Choose one of the composition_id and set it in the global variables as `“LAST_COMPOSITION_ID”.`
   v. Use the `“GET COMPOSITION by UUID”` request to retrieve the composition.

2. <u>Use openEHR tool to interact with EHRBase for retrieving the composition of a specific patient in your dataset.</u>
   a. Via browser, access the web application at the given endpoint of the openEHR tool instance of your group.
   b. Navigate the menu to the `“GET template”` functionality and retrieve the list of all the available templates to get the template_id.
   c. Navigate the menu to the `“GET EHR”` functionality and get the EHR of a patient of your choice from your dataset by filling in the fields `SubjectID` (the ihss_identifier, e.g. ‘IHSS_PAT_1234’) and `SubjectNamespace` (‘*National*’). Find and take note of the `ehr_id/value (e.g., 1b2fafe5-df03-47db-b320-72dad5d59c788).`
   d. Navigate the menu to the `“RUN AQL query”` functionality, then adapt the following query to retrieve all the compositions for the given `ehr_id/value.` Take note of the returned `composition_id` (e.g., cf7dba02-303b-4383-acf5-5629907c349d::local.ehrbase.org::1).

```
select
   c/uid/value as uid
from EHR e
contains COMPOSITION c
where e/ehr_id/value = '1b2fafe5-df03-47db-b320-72dad5d59c78'
```

   e. Retrieve the composition using the `“Get Composition”` functionality

Part 4 (if time allows)

Decription: perform some simple queries written in Archetype Query Language (AQL) to retrieve the data

[3] **hint:** Check in the `“query”` folder of the collection to find the proper request

Expected duration: 20 min
Activity type: individual work

Perform some simple AQL queries from the file of examples[4]. You can run the queries using one of the two ways as you prefer:

1) using the openEHR tool by simply run the query from the interface
2) using Postman:
    a) open the tab *Query Params* of the "`query_ad_hoc`" request
    b) Copy the AQL query in the value of the "*q*" key
    c) Select the text you just pasted, right-click and *EncodeUriComponent*
    d) Send the request to see the results

Quick links

Terminologies, ontologies and search portals

- Bioportal: https://bioportal.bioontology.org/
- Ontology Lookup Service (OLS): https://www.ebi.ac.uk/ols/index
- LOINC: https://loinc.org/
- ICD-10: https://bioportal.bioontology.org/ontologies/ICD10
- SNOMED: https://www.snomed.org/

Tools

- Xmind software: https://xmind.app/
- openEHR Archetype Designer (AD): https://tools.openehr.org/designer/#/
- Clinical Knowledge Manager: https://ckm.openehr.org/ckm/
- FLATEHR: https://github.com/crs4/flatehr#flatehr
- EHRBase: https://github.com/ehrbase/ehrbase
- Postman: https://www.postman.com/
- openEHR tool: https://github.com/sasurfer/openEHR-tool

Documentations

- EHRBbase dcumentation: https://ehrbase.readthedocs.io/en/latest/index.html
- openEHR REST API: https://specifications.openehr.org/releases/ITS-REST/latest
- Archetype Query Language (AQL): https://specifications.openehr.org/releases/QUERY/latest/AQL.html

[4] If you open the text file using Visual Studio Code, you can benefit from the syntax highlighting of a plug-in for AQL queries. Documentation here: https://marketplace.visualstudio.com/items?itemName=NedapHealthcare.openehr-adl-lsp

Materials

- Source data folder (csv and json): https://github.com/crs4/interhealth2022/tree/main/openEHR_in_action/starting_datasets
- Mindmap https://github.com/crs4/interhealth2022/tree/main/openEHR_in_action/1.model_mindmap
- Preliminary template: https://github.com/crs4/interhealth2022/tree/main/openEHR_in_action/2.preliminary_template
- Final template: https://github.com/crs4/interhealth2022/tree/main/openEHR_in_action/3.final_template
- Postman configuration files: https://github.com/crs4/interhealth2022/tree/main/openEHR_in_action/4.postman_conf
- Query examples: https://github.com/crs4/interhealth2022/blob/main/openEHR_in_action/5.AQL_query_examples