Jan 11, 2018

First order - Time and date of next meeting - Monday afternoon or Tuesday. I will send another poll specifically for that day

Goal:

Reduce duplication of effort and resources.

Determine what our final interface looks like - single service? Microservices? Integrated browser suite or mutliple web sites?

Make things easy for the user and for us.

Method steps:

1. What are current assets
    a. What users are we currently serving with each service
    b. What "types" of assets do we have
2. Where do current assets fall short of what we need
    a. Where do we have assets that aren't utilized or not needed
    b. What is priority of need
    c. What demands are we expecting from the future and new users
3. What is feasible for the future - determine constraints (including cost)
    a. In short term
    b. In long term
4. What metrics will we use to evaluate services and usage.
    a. What tools are we missing the would help us gather these metrics

Lyubov - wants to see a structure which describes the services. Has information on the users and their needs and how they are currently being met, and what they might need in the future. Within each service category we also want technical details - what physical assets are part of it (servers, databases) and what people are working to provide the services. From the user perspective - what services do we not provide.

Gilberto - go bottom up. At the end do we want a single service. What are the improvements that need to be done to LexEVS to make it that service. It already has a lot of our offered services (maps, VS, REST, etc). What do we need to have a single infrastructure?

Action items:

Tracy - send out poll to determine meeting time for Monday/Tuesday

Tracy / Rob - identify services we use before next meeting

Tracy / Rob - identify services editors use

Jason - list NG products (Term Browser, Metathesaurus Browser, Term Suggestion Application, Report Writer, EVSRESTAPI, SPARQL Report Writer)

Tools and services inventory:

Reporting/Editing Tools:

1. NCI Report Writer (LexEVS RMI, mySQL DB)
    a. 28 templates supporting
        i. CDRH
        ii. CTS
        iii. CareLex
        iv. FDA
        v. GAIA
        vi. GENC
        vii. GUDID
        viii. ICS
        ix. NCI
        x. NICHD
        xi. SPL
        xii. eStability
    b. standalone Java program to generate ancillary Neoplasm Core formats
2. Protege
    a. NCI edit tab
    b. Pellet reasoner
    c. Lucene query
    d. Reporting

            i. Export fields based on a Lucene Query
            ii. Export fields based on a list of codes
            iii. Export a report based on a root concept
3. FormatProtegeReport (POJO)
    a. Extract terms, defs, properties of interest from a Protege report
4. GenerateCDISC Java program (OWL API wrapper)
    a. ADaM
    b. CDASH
    c. Glossary
    d. SDTM
    e. SEND
5. nci-diff-CDISC Java program (POJO)
    a. Changes between CDISC updates
6. ODM_Converter (Scala)
    a. Format CDISC odm.xml, html, PDF
7. SPARQL Report Writer

Communications/Publications:

1. NCI Meta browser https://ncim.nci.nih.gov
2. NCI terms browser -
    a. https://nciterms.nci.nih.gov
    b. https://ncit.nci.nih.gov
3. CDISC Tracker for new term requests - https://tracker.nci.nih.gov/projects/CDISC/summary
4. Term Suggestion - https://nciterms.nci.nih.gov
    a. CDISC Term Suggestion https://ncitermform.nci.nih.gov/ncitermform/?version=cdisc
5. NCI_Thesaurus email account
6. Cancer.gov
    a. https://www.cancer.gov/research/resources/terminology
    b. JIRA https://tracker.nci.nih.gov/browse/
7. Github issue tracker for Protege https://github.com/NCIEVS
8. EVS download page - https://cbiit.cancer.gov/evs-download
9. Wiki
    a. EVS-LexEVS - LexEVS
    b. EVS public - EVS Wiki
    c. EVS private - https://wiki.nci.nih.gov/display/EVSproj
    d. Protege - Protege and NCI Protege
10. ServiceNow https://service.cancer.gov
    a. Ticketing system for Systems support requests

Operations:

1. ProtegeKbQA Java program (OWL API wrapper)
    a. 24 tests supporting business rules and quality control of data in Thesaurus baselines
    b. Metrics about Thesaurus data
2. OWLDiff (OWL API wrapper)
    a. Diff report of two baselines
3. GenerateOWLAPIInferred (OWL API)
    a. Generate the inferred version of Thesaurus
4. OWLSummary Java program (OWL API wrapper)
    a. Details report
    b. Summary report
    c. Used for statistics and generation of the MEME config file
5. OWLScrubber Java program (OWL API)
    a. Remove properties not for publication
6. Lexical Variant Generation (LVG) (Java suite by NLM)
    a. Replace reserved list of characters with ASCII equivalents
7. LexEVSCompare Java program (LexEVS RMI)
    a. Matching performed by the editors
8. TopBraid (COTS)
    a. Conversion of CDISC reports to OWL/RDF
9. NDFRT_File_Processing (POJO)
    a. Process and post monthly NDFRT files
    b. Process and post monthly SPL files
10. Wusage - for extracting web usage of EVS related web sites and FTP
    a. https://sitestats.nci.nih.gov
11. OWL Conversion programs
    a. HGNCtoOWL
    b. CTCAEtoOWL
    c. NDFRTtoOWL
12. Protege_Baseline_Export (Java)
    a. Export a Protege baseline from the command line (scriptable)
13. ExtractBranches (OWL API)
    a. Generate branch files from Thesaurus for publication
14. FormatHistory (POJO)
    a. Generate history files for publication
15. ValueSetProduction (POJO)
    a. Automate the value set file administration during each baseline

16. Stardog (COTS)
17. LexEVS runtime (Java)
    a. Administration of LexEVS

APIs

1. LexEVS java api
    a. 6.4 (Java 7) - https://lexevsapi64.nci.nih.gov/
    b. 6.5 (Java 8) - https://lexevsapi65.nci.nih.gov/
2. CTS2 - https://lexevscts2.nci.nih.gov
3. EVSRESTAPI https://evsrestapi.nci.nih.gov
4. SPARQL
    a. https://sparql-evs.nci.nih.gov/sparql
    b. https://sparql-evs.nci.nih.gov/ctrp