Implementing Patient Recruitment on EURECA Semantic Integration Platform through a Groovy Query Engine

Brecht Claerhout, *Member, IEEE,* Kristof De Schepper, David Perez-Rey, Raul Alonso-Calvo, Jasper van Leeuwen and Anca Bucur

Abstract— **A substantial amount of clinical trials are subject to failure because they cannot recruit sufficient patients within the foreseen time and budget. Computer assisted evaluation of eligibility criteria is envisaged to improve the recruitment process by increasing coverage (i.e. making sure no eligible patients are 'missed') and speeding up the eligibility scanning process; and thus eventually reduce the overall failure rate of clinical trials. In this paper a new recruitment application is presented which assists in evaluating eligibility criteria based on available clinical patient data (e.g. Electronic Health Record data). The application leverages for the querying of clinical data on a generic semantic integration platform and a new Domain Specific Language (DSL) based on the Groovy programming language.**

I. INTRODUCTION

CLINICAL trials are crucial to medical research and drug development. In spite of careful design of clinical trial protocols (cf. trial protocol feasibility studies) a substantial amount of trials is still subject to failure because the aimed recruitment target cannot be reached within the given time and budget [1].

Computer aided recruitment applications [2] aim to improve the recruitment process by (partially) automating the evaluation of eligibility criteria. For this, they rely on electronically available patient data coming from Electronic Health Record (EHR) systems and clinical trial databases. The recruitment applications are designed to provide to recruiters an efficient interface to sift through large patient populations looking for eligible patients. Their goal is to increase coverage (i.e. making sure no eligible patients are 'missed') and speed up the eligibility scanning of patients; and as such reduce the overall recruitment failure rate of clinical trials.

In this paper we present a new patient recruitment application built within the context of the Enabling information re-Use by linking clinical REsearch and Care (EURECA) project (http://www.eurecaproject.eu/). EURECA is part of the 7th Framework Programme funded by the European Commission and aims to enable seamless, secure, scalable and consistent linkage of healthcare data residing in EHR systems with information in clinical research information systems (such as clinical trial systems) [3].

The presented patient recruitment application leverages on the EURECA semantic integration platform for retrieving patient information from heterogeneous data sources and on a novel query language and execution engine (based on the Groovy dynamic programming language) for formalising the complex queries associated with eligibility criteria.

This project/study is partially funded by the European Commission under the 7th Framework Programme (FP7-ICT-2009-6-270253).

Brecht Claerhout and Kristof De Schepper are with Custodix, Kortrijksesteenweg 214, 9830 Sint-Martens-Latem, Belgium (e-mail: brecht.claerhout@custodix.com & kristof.deschepper@custodix.com).

David Perez-Rey and Raul Alonso-Calvo are with Grupo de Informática Biomédica, Dept Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo S/N, 28660 Boadilla del Monte, Madrid, Spain. (e-mail: dperez@infomed.dia.fi.upm.es & ralonso@infomed.dia.fi.upm.es).

Anca Bucur and Jasper van Leeuwen are with Phillips Research, Healthcare Information Management, High Tech Campus 34, 5656 AE Eindhoven, The Netherlands (e-mail: anca.bucur@phillips.com & jasper.van.leeuwen@phillips.com).

II. BACKGROUND

A. Patient Recruitment Process

The objective of the patient recruitment process is to enrol a pre-defined number of patients (those needed for making the trial outcome statistically significant) into a clinical trial.

The process is started when a sponsor (be it a commercial or academic one) sends a finalised trial protocol to different pre-selected clinical sites (after completing all associated administrative processes). Based on the protocol's inclusion and exclusion criteria, investigators at those clinical sites can then start identifying suitable candidates for enrolment in the trial for their site.

The presented application is aimed to assist recruiters on-site with the patient selection; the actual enrolment and further follow-up in the clinical trial are outside of the scope of this work. The identification of candidate eligible patients is done by automatically matching the eligibility criteria of the trial with all patient data available on the site (i.e. EHR and other clinical data bases). Final evaluation of candidate eligibility is left to the recruiter (see further).

B. EURECA Architecture: Semantic Integration Layer

The semantic integration layer of EURECA aims to provide homogenous access to patient data across applications. To homogenize information from different systems such as Clinical Trial Management (CTMS), laboratory or EHR systems; a Common Information Model (CIM) was devised. As shown in Fig. 1, the CIM includes two main components: (i) the Common Data Model (CDM) and (ii) a core set of concepts. The CDM, based on Health Level 7 Reference Information Model (HL7 RIM) [4], is the

978-1-4799-3163-7/13/$31.00 ©2013 IEEE

structure and set of terms onto which ad-hoc data models from different institutions are mapped (through Extraction, Transformation, & Loading (ETL)). The core set of concepts includes the set of terms used in the CDM and the corresponding relationships among them. It is based on Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT) [5], extended with domain specific terminologies such as Logical Observation Identifiers, Names and Codes (LOINC) [6] for laboratory tests, Human Genome Organisation Gene Nomenclature Committee (HGNC) [7] for gene names and stored in a SESAME triple store server [8]. For any other terminology, terminology services such as BioPortal [9] are used.

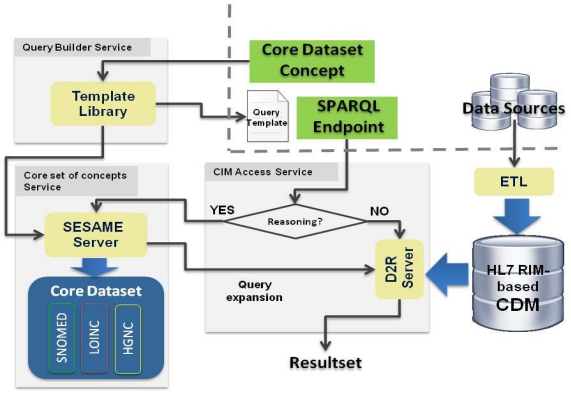

Fig. 1. Common Information Model

Additional components were required to facilitate data retrieval, providing a Simple Protocol And Resource description framework Query Language (SPARQL) endpoint and a query builder service (SPARQL templates). The SPARQL endpoint is provided by the CIM access service and a D2R server that map the HL7 RIM-based relational CDM to RDF. To add semantic reasoning according to the core set of concepts, a query expansion mechanism has been implemented. It is based on two main features: (i) SNOMED-CT organization, that follows a hierarchical structure using "is_a" relationships (e.g. searching for patients that receive *anthracycline* treatment requires expanding the initial query with all drugs within the *anthracycline* family, since treatment will be recorded as *epiribucin* or *idarubicin*, but never as *anthracycline*) and (ii) the concept normalization provided by the SNOMED Normal Form (e.g. queries for "breast cancer" patients should retrieve the same results that patients for "carcinoma", "finding site", "breast" patients).

Finally, the query builder component aims to encapsulate the structure of the data model and the SPARQL syntax. Given a concept from the core set, the query builder returns the corresponding SPARQL template with possible filters and operations. As a result, the semantic integration layer implemented provides the required tools to homogeneously retrieve semantically-aware clinical data.

III. Patient Recruitment Application

A. Computer Aided Recruitment

In this paper we present a computer aided (vs. automatic) recruitment solution, which assists the recruiter in the otherwise fully manual process of checking whether a patient matches a trial.

Apart from possible legal complexities involved, full automation of this process is not desirable. There are several reasons for this, for example not all data required for evaluation will always be present for a patient; criteria which require absence of a condition can only be manually confirmed (cf. was the condition not recorded or did the condition never occur?); some criteria might be too complex to formalise in the application; etc.

Finally, additional to the above humans can more easily handle unstructured data (e.g. scanned written text which might be accessible through the EHR) and typically have a bigger medical knowledge base (e.g. to infer missing information) than computer systems.

B. Architecture

Fig. 2 gives an architectural overview of the patient recruitment application. It contains seven loosely coupled components, logically grouped in four architectural layers. The components of the semantic integration and data/access services layers are part of the EURECA semantic integration layer, discussed in II B.

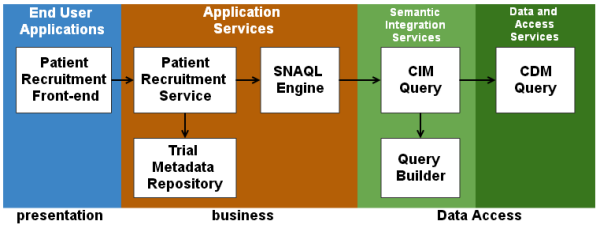

Fig. 2. Architectural layer overview

The *patient recruitment front-end* provides a visualisation of the recruitment functionality to the end-user of the application. The main layout and functionality of this font-end, is briefly discussed in III C.

The *patient recruitment service* is the main driver component of the recruitment application. It offers the back-end functionality needed to complete the different steps in the recruitment process. Through the recruitment service, Domain Specific Language (DSL) scripts (see further, SNAQL scripts) for querying the patient data are fed to the *SNAQL engine* which executes them. The engine's functionality is discussed in IV B.

The *trial metadata repository* is a central store containing information about actively recruiting clinical trials including for each trial the set of eligibility criteria (inclusion and exclusion criteria). Internally, the repository relies on the Biomedical Research Integrated Domain Group (BRIDG) domain analysis model (http://www.bridgmodel.org).

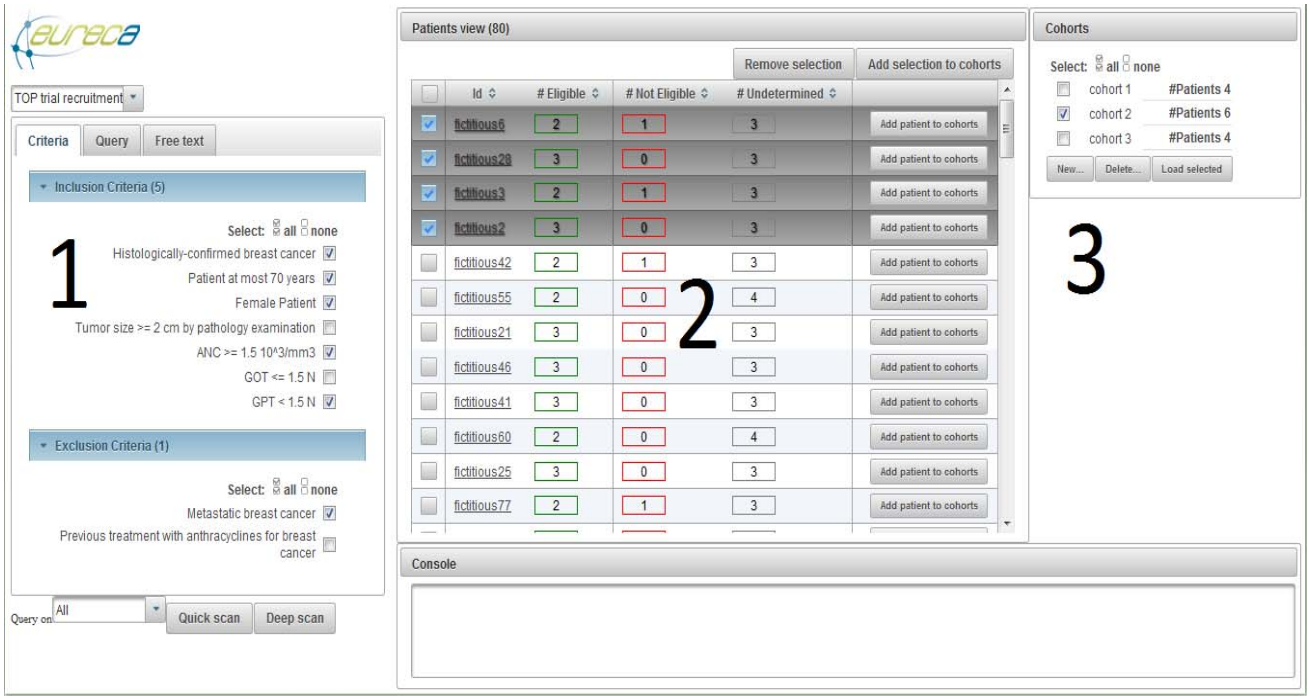

Fig. 3. Overview user interface of the patient recruitment application

C. *Application Front-end*

The trial recruitment application provides a web based GUI front-end (see Fig. 3) based on the "shopping cart" metaphor, which is commonly used in internet web stores. The general idea behind this metaphor is that a recruiter should be able to quickly sift through patients of interest ("shopping objects") and put candidate trial subjects in cohort groupings ("shopping basket"). These groupings allow recruiters to follow an iterative process of checking eligibility, i.e. checking the most interesting patients first, and allow them to gradually (manually) look up missing data for final eligibility evaluation.

The interface is structured in three main functional components: the filter block (1), patient set overview (2) and the cohort block (3). The filter block provides three different types of filtering mechanisms for browsing through the patient dataset available in the local EHR system: a criteria, query and free text filter.

The criteria filter is used for scanning the patient databases based on the trial criteria. Clinical trial criteria are stored as predefined formalised SNAQL scripts (created by expert users) in the central trial meta-data repository. The downloaded scripts are executed using the SNAQL engine, determining whether a patient would be eligible or ineligible according to a criterion or whether no automated decision can be made based on the available structured data. Because the scripts are predefined, the recruiter is not required to have knowledge about the SNAQL language itself.

In the query filter, a recruiter can define custom SNAQL scripts for querying structured patient data. This is useful for example in the case that an investigator considers the trial formalisation of a criterion not to be the most appropriate for his dataset (e.g. when information is to be inferred), or when he considers relaxed selection thresholds to be acceptable (e.g. minor differences in lab values). This filter requires a recruiter with a basic technical background in order to define these SNAQL scripts.

Finally the free text filter enables a user to filter patient sets based on a free text search. This is a useful tool for the further manual evaluation of eligibility (lookup in unstructured patient information).

The results of the filter operations are shown as a list of patients (2). Detailed information about a patient can be consulted by clicking on a patient entry (the user is forwarded to the original data source, e.g. an external EHR interface). Patients can be put in cohorts (3) ("baskets" in the shopping cart metaphor) which can be re-used as source for filtering (1). The cohort mechanism thus can be used as a temporary storage point for sorting and further examining sets of candidate eligible patients. This allows for an iterative mechanism of applying filters and screening patients for eligibility.

The query and free text filter act as true query filters, i.e. the result sets only contains matching patients. The criteria filter on the other hand behaves differently, it annotates the patient list with the number of criteria that respectively evaluate as eligible, ineligible or unknown, for each patient. The user interface allows patients to be ranked according to these numbers.

In a typical usage scenario, a recruiter first executes a criteria filter and groups the patients in different cohorts according to ranking. Starting with the most interesting group, he can then further analyse the groups with the filter mechanisms available. Eventually, he will save patients considered eligible in a separate cohort.

IV. GROOVY-BASED ENGINE

A. SNAQL Groovy-based DSL

A novel medical query language was designed to avoid that users need to have deep technical knowledge (e.g. Structured Query Language (SQL), SPARQL) and awareness of the underlying semantic data models for querying data during the recruitment process. The objective was to allow the definition of queries in a language close to natural language, while keeping the implementation effort to a minimum. Contrary to approaches [10] concentrating primarily on developing a fully specified formal language for defining clinical criteria, balancing implementation effort with end-user usability was our primary concern.

TABLE I
SNAQL EXAMPLE

Patients having received mastectomy and post-surgical irradiation for the primary diagnosis, who developed a distant relapse during the 5 first years after this surgery.

```
def MASTECTOMY='172043006'
def MALIGNEOPLAS='128462008'
def POSTOPRADIO='168525009'

def condition =
diagnosis of:MALIGNEOPLAS occurred
after:{lastOccurrence of:{procedure
of:MASTECTOMY}}, within: 5.years

result = var{condition} and {procedure
of:POSTOPRADIO}
```

The presented query DSL (called "Snaggletooth" Query Language (SNAQL)) leverages on the DSL-facilitating features of the Groovy scripting language for offering advanced functionality (e.g. temporal query constructs) at minimal effort. These include operator overloading, closures, metaprogramming (Meta Object Protocol (MOP)) support and named parameter support through argument maps.

The power of the native DSL capabilities of Groovy comes at a trade-off: DSL syntax still requires inclusion of compiler hints (':', '.', brackets). Despite this, the SNAQL maintains a uniform syntax that is human readable (see table 1). One solution for hiding these hints to the end-user is to introduce a pre-processor step that decorates a more "natural language" form of SNAQL with the needed compiler hints (subject of future work).

B. Groovy Execution Engine

The SNAQL execution engine is not an interpreter, but rather the definition of a context in which the SNAQL code makes sense. The context defines the classes and methods that represent the "medical concepts" and "functional" keywords of the DSL. Within this context, the SNAQL script is plain groovy code which instantiates objects (patient lists) and performs methods on them (the functions).

Fig. 4 shows a high level overview of the engine, illustrating the two fundamental classes constituting the execution context.

The *ConceptQuery* class represents medical base concepts of SNAQL in their SPARQL query form. Whenever a *ConceptQuery* is instantiated, the engine retrieves the SPARQL statement (cf. query builder service) that allows lookup of patients matching the DSL statement. Methods defined on *ConceptQuery* define DSL operators which can be implemented as modifiers on the SPARQL string, effectively implementing a "lazy SPARQL query execution" mechanism.

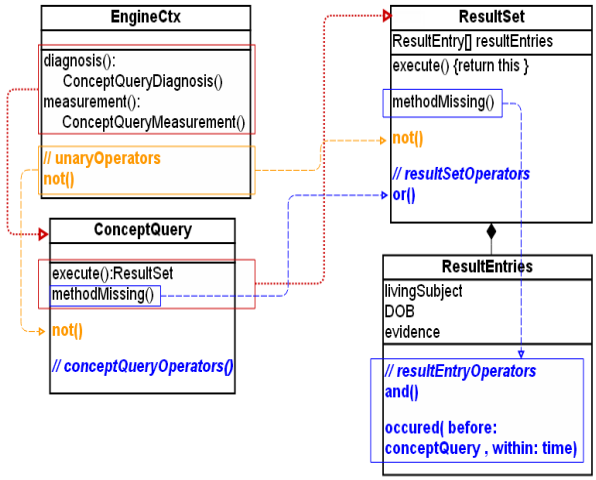

Fig. 4. Query Engine Class Diagram Overview

The *ResultSet* class is the natural representation of a patient cohort in the engine. Methods defined on *ResultSet* define functions which can only be implemented on a list of patients (because they are too complex for SPARQL query modification, e.g. most time related functions).

The "lazy SPARQL query execution" mechanism relies on the generic *methodmissing()* MOP method (triggered when a method is not found on a Groovy object), a *ConceptQuery* will be executed (perform the SPARQL query) and is transformed to a *ResultSet*. Eventually all intermediate (and final) query results are available as *ResultSet* objects. With every intermediate result being from the same class, the DSL can rely on the object oriented "fluent interface paradigm" for the language constructs. Finally, dynamic typing and closures provide nesting functionality in the DSL without the need for any specific code in the engine.

V. EVALUATION

A first prototype implementation of the presented patient recruitment tool was deployed within the EURECA project. The prototype was tested on two datasets, provided by the EURECA clinical partners, containing anonymised patient data coming from real clinical trials. The results of these tests provided important feedback for improving the initial prototype. A thorough evaluation will be performed in a later (more mature) iteration of the tool.

VI. Conclusion

A computer aided tool for recruitment of patients in clinical trials was presented. The solution contains a graphical user interface based on the shopping basket metaphor which assists a recruiter in a user friendly way during the recruitment process. Automated evaluation of trial criteria is supported through a novel query engine based on the Groovy programming language. Patient data is made available to this engine through a semantic integration platform developed as part of the EURECA project. Together with the engine, a DSL (SNAQL) has been introduced which allows for user-friendly querying of clinical data. This DSL is used both for the formalization of trial criteria and for allowing recruiters to easily filter patient data.

Future work includes further elaboration of the SNAQL engine function library and the construction of an advanced GUI builder for the query DSL which hides technical peculiarities of the SNAQL formulation from the end-user. Next to this, the recruitment application will be thoroughly evaluated and finally piloted on different clinical sites as part of the EURECA validation process.

References

[1] T. Reynolds, "Clinical trials: can technology solve the problem of low recruitment?," in *British Medical Journal*, 2011, vol. 342, pp. 1338.

[2] M. Cuggia, P. Besana and D. Glasspool, "Comparing semi-automatic systems for recruitment of patients to clinical trials," in *I. J. Medical Informatics*, 2011, vol. 80, no. 6, pp. 371-388.

[3] R. Vdovjak, B. Claerhout and A. Bucur, "Bridging the gap between clinical research and care - approaches to semantic interoperability, security & privacy," *HEALTHINF*, 2012, pp. 281-286.

[4] G. Beeler, J. Case, J. Curry, A. Hueber, L. Mckenzie, G. Schadow and A.-M. Shakir. (2004, May 10). *HL7 reference information model* (V 02-04) [Online]. Available: http://www.hl7.org/library/data-model/RIM/C30204/rim.htm

[5] K. Donnelly, "SNOMED CT: the advanced terminology and coding system for ehealth," in *Studies in Health Technology and Informatics - Medical and Care Compunetics*, 2006, vol. 121, no. 3, pp. 279–90.

[6] C.J. McDonald, S.M. Huff, J.G. Suico, G. Hill, D. Leavelle, R. Aller, A. Forrey, K. Mercer, G. DeMoor, J. Hook, W. Williams, J. Case and P. Maloney, "LOINC, a universal standard for identifying laboratory observations: a 5-year update," in *Clinical Chemistry*, 2003, vol. 49, no. 4, pp. 624-633.

[7] R.L. Seal, S.M. Gordon, M.J. Lush, M.W. Wright and E.A. Bruford, "Genenames.org: the HGNC resources in 2011," in *Nucleic Acids Research*, 2011, no. 39, pp. 514-519.

[8] J. Broekstra, A. Kampman and F. Harmelen, "Sesame: a generic architecture for storing and querying RDF and RDF schema," *The Semantic Web*, 2002, no. 2342, pp. 54-68.

[9] N.F. Noy, N.H. Shah, P.L. Whetzel, B. Dai, M. Dorf, N.B. Griffith, C. Jonquet, D.L. Rubin, B. Smith, M.A. Storey, C.G. Chute and M.A. Musen, "Bioportal: ontologies and integrated data resources at the click of a mouse," in *Nucleic Acids Research*, 2009, pp. 170-173.

[10] C. Weng, S. W. Tu, I. Sim and R. L. Richesson, "Formal representation of eligibility criteria: a literature review," in *Journal of Biomedical Informatics*, 2010, vol. 43, no. 3, pp. 451-467.