Journal of Biomedical Informatics 62 (2016) 243–264

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

# Publication, discovery and interoperability of Clinical Decision Support Systems: A Linked Data approach

Luis Marco-Ruiz [a,b,*], Carlos Pedrinaci [c], J.A. Maldonado [d,e], Luca Panziera [c], Rong Chen [f], J. Gustav Bellika [a,b]

[a] Norwegian Centre for e-Health Research, University Hospital of North Norway, P.O. Box 35, N-9038 Tromsø, Norway
[b] Department of Clinical Medicine, Faculty of Health Sciences, UIT The Arctic University of Norway, 9037 Tromsø, Norway
[c] Knowledge Media Institute, The Open University Walton Hall, Milton Keynes MK7 6AA, UK
[d] Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain
[e] Veratech for Health SL, Doctor Lluch 299, 46011 Valencia, Spain
[f] Department of Learning, Informatics, Management and Ethics, Karolinska Institutet, Tomtebodavägen 18, SE 17177 Stockholm, Sweden

A R T I C L E  I N F O

*Article history:*
Received 7 March 2016
Revised 5 July 2016
Accepted 7 July 2016
Available online 8 July 2016

*Keywords:*
Clinical Decision Support
Linked Data
Semantic Web Service
Semantic interoperability
Service Oriented Architecture
Medical ontologies

A B S T R A C T

*Background:* The high costs involved in the development of Clinical Decision Support Systems (CDSS) make it necessary to share their functionality across different systems and organizations. Service Oriented Architectures (SOA) have been proposed to allow reusing CDSS by encapsulating them in a Web service. However, strong barriers in sharing CDS functionality are still present as a consequence of lack of expressiveness of services' interfaces. Linked Services are the evolution of the Semantic Web Services paradigm to process Linked Data. They aim to provide semantic descriptions over SOA implementations to overcome the limitations derived from the syntactic nature of Web services technologies.
*Objective:* To facilitate the publication, discovery and interoperability of CDS services by evolving them into Linked Services that expose their interfaces as Linked Data.
*Materials and methods:* We developed methods and models to enhance CDS SOA as Linked Services that define a rich semantic layer based on machine interpretable ontologies that powers their interoperability and reuse. These ontologies provided unambiguous descriptions of CDS services properties to expose them to the Web of Data.
*Results:* We developed models compliant with Linked Data principles to create a semantic representation of the components that compose CDS services. To evaluate our approach we implemented a set of CDS Linked Services using a Web service definition ontology. The definitions of Web services were linked to the models developed in order to attach unambiguous semantics to the service components. All models were bound to SNOMED-CT and public ontologies (e.g. Dublin Core) in order to count on a lingua franca to explore them. Discovery and analysis of CDS services based on machine interpretable models was performed reasoning over the ontologies built.
*Discussion:* Linked Services can be used effectively to expose CDS services to the Web of Data by building on current CDS standards. This allows building shared Linked Knowledge Bases to provide machine interpretable semantics to the CDS service description alleviating the challenges on interoperability and reuse. Linked Services allow for building 'digital libraries' of distributed CDS services that can be hosted and maintained in different organizations.

## 1. Introduction

### 1.1. Clinical Decision Support Systems interoperability and reuse

The term Clinical Decision Support Systems (CDSS) encompasses a wide range of recommendation systems that vary in purpose and complexity ranging from small logic modules that implement simple lists of order sets, to complex decision algorithms that compile the knowledge contained in nationally recommended guidelines [1]. Nowadays, it is generally acknowledged that CDSS contribute to improve health care, reduce costs and support access to the latest evidence [2–4]. However, their development costs are high as a consequence of the highly skilled professionals needed for knowledge engineering and development tasks [5–7]. For example, Field et al. estimated a cost of circa 49,000 USD only for the initial development of a set of CDS artifacts for medication alerts [8]. When it comes to more complex CDSS such as Computer Interpretable Guidelines (CIGs), the

* Corresponding author at: Norwegian Centre for e-Health Research, University Hospital of North Norway, P.O. Box 35, N-9038 Tromsø, Norway.
*E-mail address:* Luis.Marco.Ruiz@telemed.no (L. Marco-Ruiz).

http://dx.doi.org/10.1016/j.jbi.2016.07.011
1532-0464/© 2016 Elsevier Inc. All rights reserved.

development becomes even more complicated elapsing over longer periods and being more error-prone [9]. Furthermore, for large deployments, it is necessary to create dictionaries of terms and data templates which need significant resources to be maintained. Maviglia and Sordo [10] estimated the cost of including one concept definition in their CDS terms dictionary at 6 h, allocating around 300 h to cover the approximately 50 concepts that they process per month.

As a consequence of the high development and maintenance costs, several studies have pointed to the need of reusing CDS artifacts and dictionaries of terms across organizational boundaries to avoid replicating development efforts [11–14].

In order to reuse artifacts, early approaches such as the Arden syntax [15] or GLIF [16] focused on defining standards to specify reusable decision logic. However, the deployment of the artifacts in a new Electronic Health Record (EHR) still required re-implementation of the other components that compose the CDSS. This requires redefining data mappings to the EHR, mapping terminological concepts, and re-testing the CDSS behavior. In summary, building all the parts that are not logic from scratch so it complies with the data and execution restrictions of the new environment where the CDS artifact is deployed.

To alleviate this problem, CDS researchers turned their sight to the SOA paradigm aiming for the reutilization of the whole CDS system [12,17–19]. In a nutshell, a SOA implementation encapsulates the CDS system inside a Web service which is shared among several clients. This approach switches CDS reuse from a paradigm of sharing decision logic to a paradigm of sharing CDS functionality. Thus avoiding the need to re-implement the system when a new client requires its functionality.

The encapsulation of CDS artifacts as Web services allows delegating expensive tasks related to the implementation, maintenance and governance of the CDS system. However, this delegation comes at a price. When a client relies on a CDS service maintained by a third party, the client does not have precise information about the features of the system beyond a syntactic definition in an Interface Definition Language[1] (IDL). As a consequence, barriers to enable client-service semantic interoperability (SIOp) have been detected related to difficulties understanding the semantics of the CDS service interfaces. Dixon et al. [12] and Wright et al. [20] detected major challenges to enable client-service SIOp related to difficulties in understanding the semantics of the CDS service interfaces when sharing CDS services among 4 organizations. When it comes to large health networks, such as those in European public health systems, SIOp becomes much more complex, and yet reusing such artifacts becomes even more appealing. The systems in a health network usually employ different standards and terminologies. In fact, not even the representation of Web service messages as Clinical Information Models (CIMs) [21] annotated with standard terminologies has resolved this issue [12]. When the clinical models are annotated with a standard terminology, the terminology codes add a certain degree of semantics indirectly, but the structure is still a syntactic description with a standard code as identifier. This structure contains implicit knowledge in the labels and descriptions expressed as natural language, but lacks a proper ontological foundation [22]. The attributes and labels of the model specify information structures, but they are not defined as concepts interrelated by meaningful machine-understandable relationships. The relationships among concepts need to be not only human readable, but also machine computable for CDS systems to function effectively and safely across EHRs [23]. For example, it is not possible to unambiguously infer that a particular label refers to a semantic relationship between two concepts inside a CIM, or that an attribute is semantically equivalent to another one in another CIM. This represents a major issue in CDS functionality reuse since accurate understanding of the concepts and the relationships referenced in the CDS service interface is a necessary condition to understand how to invoke it. For example, let us consider that a CDS for drug dosing is available and its valid input is an anticoagulant drug. It is not possible to automatically infer that the system may be invoked with an instance of Xarelto® because it is the trade name of the active substance Rivaroxaban; which, in turn, is a subtype of anticoagulant. These limitations are not only related to clinical knowledge specification, but also to the properties needed to express metadata for the governance of the system.

Overcoming these challenges requires adequate support for capturing and sharing clear unambiguous definitions of every CDSS, covering among other aspects the information structures consumed and produced by the service, the version of the system, the institution hosting it etc. Such definitions cannot be provided by Web services alone due to the syntactic nature of their underlying technologies (e.g. SOAP, WSDL or UDDI) [24].

Several areas of software engineering and artificial intelligence have already studied these challenges. The research on software components reuse has provided powerful mechanisms to unambiguously specify the system interfaces and also allow to automate tasks traditionally performed by humans.

### *1.2. Software components reuse*

One of the most prominent research efforts regarding software components reuse has been performed by the Semantic Web community. As a result, they defined the Semantic Web Services (SWS) paradigm as Web services that are extended with semantic annotations to define the system properties in a machine interpretable fashion [25,26]. Thus encapsulating the component in a Web service that describes the system interfaces using an Interface Definition Language (IDL), at a syntactic level (e.g. WSDL), and semantic annotations to reference ontologies, at a semantic level. Examples of ontologies to attach semantic descriptions to Web services are WSMO [27] and OWL-S [28].

The reuse of software components through SWS lies in the implementation of mechanisms that allow the **publication** of the component; the **discovery** of the component by third parties; and, once discovered, the analysis of the component interfaces by the clients to understand the meaning of the information exchanged; i.e. **interoperate at a semantic level** [25]. These mechanisms should allow consumers to automate discovery and analysis of the system using machine-interpretable descriptions. In the SWS domain, to express the various types of system properties and interfaces four different types of semantics have been defined [26]:

(1) Functional semantics – describe which task the system performs (e.g. the system provides support for the treatment of Atrial Fibrillation).
(2) Data semantics – describe the information model consumed by the service operations (e.g. the system processes as input a *stroke prevention review* and provides as output a *stroke risk alert*).
(3) Execution semantics – describe exceptional behaviors such as the correctness of the service execution, conditions to execute the system and runtime errors. These type of semantics appear at runtime and are not usually covered by CDS standards.
(4) Non-functional semantics – describe properties of the system deployed not included in the previous categories. Examples of these properties are the issuer of the service, the version, the date of publication etc. (e.g. the system was issued by Cambio Healthcare Systems).

[1] In this paper the term Interface Definition Language makes reference to the languages used to specify Web services interfaces. Examples are the Web Service Definition Language (WSDL) or the Web Application Description Language (WADL) used to describe the Web service operations, messages, data types etc.