Reference Architecture Representation by an Ontology for Healthcare Information Systems Software Product Line

Francisca Losavio[1], Oscar Ordaz[1, 2]
francislosavio@gmail.com, oscarordaz55@gmail.com

[1] Laboratorio MoST, Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela
[2] Escuela de matemática, Universidad Central de Venezuela, Caracas, Venezuela

Abstract: Software Product Lines (SPL) are based on assets reuse for software development and claim to reduce costs and time to market, while increasing the quality of derived products, guaranteeing the SPL evolution. SPL is now a well known approach in academic and industrial practices. The specification of the SPL domain knowledge or SPL area, is crucial to design a reusable and evolutionary Reference Architecture (RA), being RA one of the major SPL artifacts; the concrete software products members of the SPL family in the specific domain, will be derived from this RA. On the other hand, ontologies, representing hierarchically organized knowledge, have been used to specify and verify consistency of the SPL domain knowledge. The goal of this work is to present an ontology, called HIS-RA Ontology, to specify the knowledge imbedded into RA for the Healthcare Information Systems Healthcare Information Systems (HIS) domain, called HIS-RA, which has been defined in previous works, and was specified as a non-directed connected graph (P, R), where P are the components or nodes and R the edges or connectors relating components. HIS-RA has been built by a bottom-up process using the knowledge on existing market products. The HIS-RA Ontology can be used to derive consistency rules for the construction of valid architectural configurations or feasible solutions (FS), for concrete products, constructed by instantiating HIS-RA. Quality requirements that the HIS-RA functionalities or core of common components must fulfill, are considered in order to guarantee the global quality of concrete products. The design of a semiautomatic FS derivation process using the HIS-RA Ontology presented here and its support tool, is an on-going work.

Keywords: Software Product Lines; Reference Architecture; Ontology; Quality Requirements; Healthcare Information Systems; HIS-RA; HIS-RA Ontology

1. INTRODUCTION

A *Software Product Line (SPL)* is a family of software intensive systems, called *products*, sharing a common and organized set of features that satisfy specific requirements of a market sector or domain. These features are developed from a *Reference Architecture (RA),* template or generic framework containing common set of assets that are reused in different products of the SPL family [1] [2]. The SPL development or *SPL Engineering (SPLE)* [3], considers two main lifecycles, *Domain Engineering (DE)* where RA is constructed, and *Application Engineering (AE)*, where concrete product configurations are derived from RA. The elicitation of domain knowledge is crucial in the SPL context, to define an SPL family with an adequate degree of generality. RA is the underlying structure common to all members of the family, holding functional and non functional commonality and variants, called the RA *variability model* [3]. A concrete product of the family can be derived by instantiating the *variation points* [3] of the RA variability model, to select different architectural variants. Moreover, functional components must fulfill a certain degree of quality in order to perform correctly their tasks. For example, the functionality in charge of the management of *Electronic Health Records (EHR)*, a core common component in *Healthcare Information Systems (HIS)*, needs to fulfill the EHR interoperability quality requirement, to be exported and shared in distant medical institutions or local departments within the same hospital. However, there may be several variant solutions to satisfy interoperability, since it can be solved by different technical mechanism on the market, e.g. engines or tools to translate into HL7 (Health Level 7) standard format [4], and choices must be made to select a convenient solution for a concrete product, w.r.t. security, portability, availability, etc., and/or the cost of the tools. This is a complex problem, since non functional properties are the main responsible of the SPL variability, and they must be considered early in SPLE, since they can originate a combinatorial explosion during the RA instantiation and product derivation in the AE lifecycle [5]. In consequence, the elicitation of domain knowledge is a huge task in SPLE, and most of it is embedded into RA, such as the core common functionality, domain quality properties, architectural style(s) and business rules [6] [11] [13] [26] [30]. On the other hand, ontologies, representing hierarchically organized knowledge [7], have been used to specify and verify consistency of SPL domain knowledge [9] [10] [21]]22].

The main goal of this work is to present an ontology [7], called *HIS-RA Ontology*, which has been defined to specify the knowledge imbedded into RA for the HIS domain (HIS-RA). In previous works, HIS-RA has been constructed by a bottom-up process using the knowledge on existing market products

[6] [11]]13]; HIS-RA was initially defined as a non-directed connected graph *(P, R),* where *P* are the components or nodes and *R* the edges or connectors relating components [6]; the graph representation facilitates the automatic generation of an initial RA configuration, that must be completed in subsequent steps to conform the final RA. The HIS-RA Ontology, which is another representation of HIS-RA, is used as a tool to deduce consistency rules to derive valid architectural configurations or *feasible solutions (FS)* for concrete products by instantiating HIS-RA in the AE lifecycle. This paper concerns only the DE lifecycle. Quality requirements that HIS-RA functionalities or core of common components must fulfill, are considered in HIS-RA and in HIS-RA Ontology to guarantee the global quality of the derived concrete products. Quality requirements are specified by the ISO/IEC 25010 standard Quality Model [12], to facilitate common understanding of the quality terminology.

Besides this introduction and the conclusion, this paper is structured as follows: the second section describes the context of our work, namely, the SPL, the HIS domain, the adaptation of the standard quality model to the HIS domain, and the HIS-RA; the third section presents the HIS-RA Ontology as a representation of HIS-RA, with query examples to show the advantages of using an ontological approach in the SPL context. Finally, the fourth section is dedicated to discuss works related with the subject of the present research.

2. Context

An ontology is an explicit specification of a conceptualization [7]. Ontologies are widely used to capture domain knowledge in an organized and structured way; they are used as SPLE tools to check consistency of the concrete product configurations derived from RA [8] [9]; however, few works consider the direct representation of RA by an ontology [10]. The feature model approach, focused on describing features directly perceived by the user, which are in general functional requirements (FR), is mostly used to build the RA variability model [19]; however non functional requirements (NFR), which are the major responsible of the SPL variability [5], are not deeply considered in this approach. Many extensions or adaptations of a feature model to treat NFR are found in the literature to overcome this problem [8]. In this work, an ontology has been defined to represent the RA imbedded domain knowledge. Our RA, the HIS-RA contains the knowledge on the HIS domain, see Figure 1 and Table I [11]. Just to clarify Figure 1, we briefly present the HIS-RA common components, namely, *a1, a2, a3, b1, b2, b3* and *c1* and the variability model constituted by the instanciable placeholders, the variation points [3] denoted by the *<<vp>>* stereotype, that must be present in all SPL RA to derive concrete products from it, namely. *<<a5>>* containing UI functional variants *a1. Web pages* and *a4-GUI*, *<<b8>>* containing variants to satisfy correctness-precision, *<<b9>>* holding variants to satisfy security (confidentiality and authenticity), *<<b10>>* with variants to satisfy interoperability, *<<b11>>* containing the functional variant UI server-side *a5* for client-side *a4*; <<c10>>, <<c11>>, <<c12>>, <<c13>>, <<c14>>, and <<c15>> holding variants satisfying quality properties in Data Layer; finally *<<d3>> Network* offers variants *d1. Internet* and *d2 Satellite*; all these variation points constitute the SPL HIS-RA variability model; all these HIS-RA elements will be revisited in Section 2.3. We recall that the original components and connectors found in the different HIS market products studied to build automatically the initial RA or *Candidate Architecture (CA)*, by performing the union of the respective graphs representing each product architecture, are shown in Table I [13]. The absence of the component or connector in the product is indicated by "-"; *aRb* means that component *a* is connected to component *b*; the respective symmetric relations are not shown.

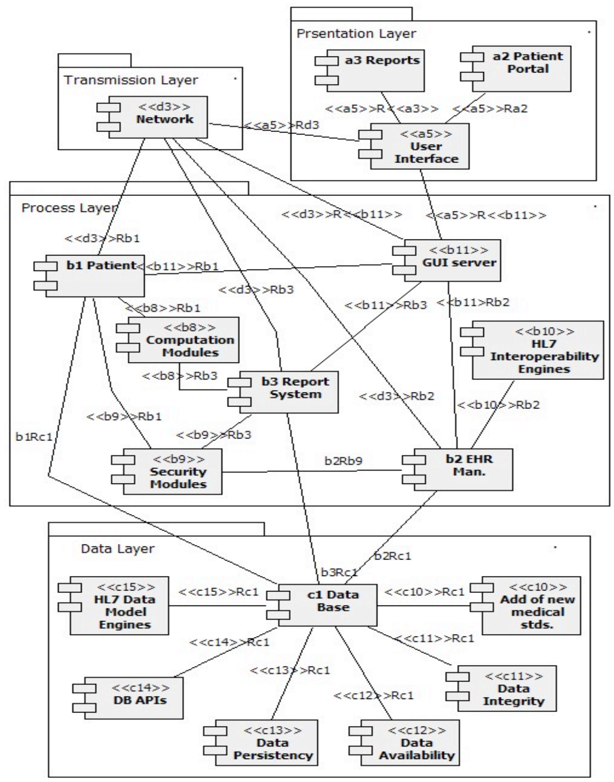

Figure 1: HIS-RA Represented in UML

Table I: Components/Connectors in each Product Architecture

Product OpenEMR	*Product PatientOS*	*Product Care2x*
a. Presentation Layer	a. Present. Layer	a. Present. Layer
Components:	Components:	Components:
a1. Web pages-Browser	-	a1
a2. Patient Portal	a2	a2
a3. Reports	a3	a3
-	a4. GUI	-
Connectors:	Connectors:	Connectors:
a1Ra2	-	a1Ra2
a1Ra3	-	a1Ra3
a1Rd1	-	a1Rd1
a1Rd2	-	a1Rd2
-	a4Ra2	-
-	a4Ra3	-
-	a4Rb5	-
b. Process Layer	b. Business logic	b. Process Layer
Components:	Components:	Components:
b1. Patient	b1. Patient business model	b1
b2. EHR Management	b2	b2

b3. Report System	b3	b3
-	b4. Mirth (HL7 engine)	-
-	b5. GUI – Server Side	-
Connectors:	Connectors:	Connectors:
b1Rc1	b1Rc1	b1Rc1
b1Rd1	-	b1Rd1
b1Rd2	-	b1Rd2
-	b1Rb5	-
b2Rc1	b2Rc1	b2Rc1
b2Rd1	-	b2Rd1
b2Rd2	-	b2Rd2
-	b2Rb4	-
-	b2Rb5	-
b3Rc1	b3Rc1	b3Rc1
b3Rd1	-	b3Rd1
b3Rd2	-	b3Rd2
-	b3Rb5	-
-	b5Rb1	-
-	b5Rb2	-
-	b5Rb3	-
-	b5Rd1	-
c. Data Layer	c. Data Layer	c. Data Layer
Components:	Components:	Components:
c1. Data Base	c1	c1
c2. HL7 CDA Engine	-	c3. HXP Engine
Connectors:	Connectors:	Connectors:
c1Rc2	-	-
-	-	c1Rc3
d. Transmission	d. Transmission	d. Transmission
Components:	Components:	Components:
d1. Internet	d1	d1
d2. Satellite	-	d2

Notice that in HIS-RA, quality properties are built-in, and their traceability to functional components is clearly established; this issue is a contribution of our present research to the complex problems of guaranteeing the satisfaction of the global domain quality by RA, and we have not found this solution clearly stated or treated in other works dedicated to general RA design, independently of an SPL context; the importance of quality issues in RA design is mentioned, but "how" to handle this crucial aspect is not clearly specified [2] [25] [26] [27].

2.1 The HIS Domain

HIS are intensive or complex information systems, generally located in different and distant institutions and with mandatory NFR requirements, such as interoperability, availability and security. They are generally supported by a hybrid distributed Client-server/Layers architectural style, [11] [15], see Figure 2 from Wikipedia. The communication/transmission tier crosscuts the other tiers, and it is supported by a Web Server, e.g. Apache, where SOA[1] appears as an event-based style for communication.

HIS must facilitate transparent sharing of different kinds of medical information, such as patient clinical records (EHR), offering also telemedicine services that can be performed online at remote locations, with wide information technology support. Moreover, in actual medical practice, SPL for HIS have not yet been completely defined, developed and adopted; the lack of agreement on standards makes difficult the interoperability of EHR [4], and HIS general adoption is still difficult. Recently, network providers are offering commercial HIS cloud solutions, which will not be treated in this work.

[1] Service-Oriented Architecture

According to [16] [17], the open source systems OpenEMR and PatientOS, with a 90% and 92% usage respectively, and Care2X (similar to OpenEMR), adopted recently in health national projects in underdeveloped countries, have been refactored into the HIS-RA considered in this work [11], see Figure 1 and Table 1, where the HIS domain for our HIS SPL is restricted to its basic functionalities, i.e., EHR management, patient attention for appointment scheduling and capture of demographic data, and emission of medical reports including basic administrative services.

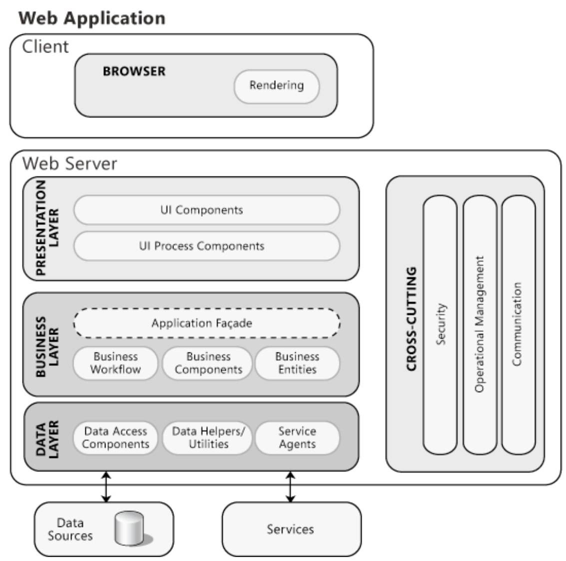

Figure 2: Hybrid Architecture Client-Server (Event-based)/Layers

2.2 HIS Domain Quality Model (HIS-DQM)

The HIS-DQM [11], shown graphically in Figure 3 adapted in [11] from [12], is constituted by seven *inherent* software quality characteristics (that do not change even if software changes) [12] and sub-characteristics, representing HIS global NFR; it was adapted in [11] from the ISO/IEC 25010 quality model [12], as input to the HIS-RA bottom-up construction process. HIS-DQM represents the global quality of the HIS SPL family; it is a hierarchical model, where high-level, usually non-measurable, quality characteristics are refined into sub-characteristics, until the measurable elements, called *quality attributes*, are attained. In particular, priority has been added to this information, (*1≤ priority ≤ 3,* where *1 high, 2 medium* and *3 low*), since priority is relevant to consider choices to select variant architectural solutions; the HIS-DQM quality characteristics/sub-characteristics with their priority, are: *compatibility (1) (interoperability), security (1) (authenticity, confidentiality, integrity), reliability (1) (availability-persistency), functional suitability (2) (correctness-precision, portability (2) (adaptability-scalability (2)), maintainability (3) (modifiability, modularity),* and *efficiency-performance (2) (time-behavior)*.

Quality attributes could be included with internal metrics at this early stage of development, such as for example, the existence (yes/no) of a mechanism to handle data availability-persistency as mirrors/replication mechanisms, or time-behavior values of certain communication protocols to handle security or certain QoS[2] measures [18]. However, metrics are not considered in this work. What is assured is that the quality property required by some functionality is accomplished or "implemented" by some mechanism or tool.

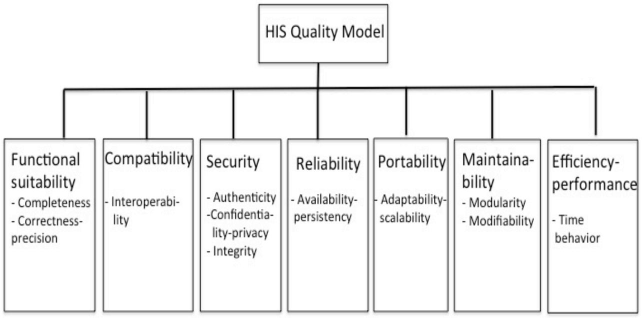

Figure 3: Hierarchical Structure of Quality Model for the HIS Domain (HIS-DQM)

2.3 HIS-RA

HIS-RA is represented by a non-directed connected graph *(P, R)*, where *P* are the nodes or components and *R* are the edges or connectors relating components [6] [11] [13]. HIS-RA contains a core of Common Components (CC), which are in general functional components representing main functionalities (FR), and the set of variation points [3] or generic components, denoted by the <<*vp*>> stereotype in Figure 1, used to instantiate RA; recall that each <<*vp*>> groups a set of variant components sharing similar tasks, representing alternative mechanisms or different architectural solution choices to satisfy quality properties derived from NFR, called also "implicit functionalities" [12], which in our approach are "imbedded" into RA. These quality properties are required by functionalities to completely accomplish their tasks. For example in Figure 1, CC *b2 EHR Man.* is connected to the variation point *<<b10>> HL7 Interoperability Engines,* meaning that this <<*vp*>> will provide a solution to achieve interoperability for EHR sharing*,* which is the achieved by variant *b4 Mirth Engine* (variants are not shown in Figure 1).

EHR management, patient appointment services and edition of medical reports with some administrative features, such as billing services, were the basic functionalities considered sufficient to illustrate our approach, as it was already mentioned; other important healthcare services such as on-line consultation, imaging and laboratory, hospital rooms management, nursing, urgencies, etc. will not be considered for this study, but of course HIS-RA can be extended to include more healthcare facilities.

As it was also pointed out, HIS-RA has been constructed in previous works [6] [11] [13], by a semiautomatic bottom-up process focused on the study of the three existing HIS open-source market products mentioned in Section 2.1 (see Table I). The HIS-RA representation in UML 2.0 [14] was shown in Figure 1; UML has been used as an Architecture Description Language (ADL) [15], because it is a widely used standard language, even if not an executable one; many RA are found in the literature using this specification [25] [26] [27]; in particular, we have used UML, because it seemed adequate to represent the graph structure defining all our architectural configurations, and facilitating the automatic generation of the initial CA from existing market products (see Table I) [13]. The ontological representation of HIS-RA, goal of the present work, is used to facilitate reasoning on the consistency of the relations among HIS-RA components, deriving logically correct consistency rules, to perform the selection of a "convenient" architectural configuration for a concrete product, fulfilling domain and customer requirements.

Recall that CC in HIS-RA are the following (see Table 1): *a2. Patient Portal,* and *a3. Reports*, which are user interface push-buttons to access main HIS functionalities, namely *b1. Patient* (appointment services and capture of demographic data), *b2. EHR Man.* (EHR management) and *b3. Report System* (medical reports and basic administrative services, such as billing); besides these main functional components, we have component *c1. Data Base*, for general data services. The variation points conforming the variability model are, namely *<<a5>> User Interface* and *<<d3>> Network* which are the only functional variation points; all the remaining ones are non functional variation points, whose variants are the implicit functionalities (also components) already mentioned, to satisfy quality properties required by each functionality, namely, *<<b8>> Computation Modules* to guarantee Correctness-precision, required by *b1* and *b3*, *<<b9>> Security Modules* to satisfy Security (*Authenticity, Confidentiality, Integrity*), required by *b1*, *b2* and *b3*, *<<b10>> HL7 Interoperability Engine* to fulfill EHR *Interoperability* in Process Layer, where all components (*b1, b2* and *b3*) uses Data Layer services; we have *<<c10>> Add. of new medical stds.*, for data *Adaptability-scalability* to new medical standards, such as catalogues and hand-outs to help to achieve diagnosis; *<<c11>> Data Integrity* to satisfy data consistency, *<<c12>> Data Availability* to guarantee backups, *<<c13>> Data Persistency* to have *Persistency* in data; notice that these are standard services provided by usual database system managers; *<<c14>> DB API*, considers database *Portability* to other platforms, and *<<c15>> HL7 Data Model Engines*, to translate EHR data into customizable formats, to achieve additional *Interoperability*. Finally, in Transmission Layer we have *<<d3>> Network* including variants *d1 Internet* and *d2 Satellite* (only <<*vp*>> are shown in HIS-RA, Figure 1). Notice that *d1* is a variant that is also a mandatory CC, because all products of the HIS SPL family will require Internet, since HIS are Web-based system, but not all of them will have the Satellite facility. All mentioned qualities properties are specified by the HIS-DQM that was described in Section 2.2, see Figure 3, and specified by the standard ISO/IEC 25010 [12]; for the specific definitions of the HIS-DQM quality properties, see [13]. More details on the HIS-RA configuration and its mapping to the HIS-RA Ontology will be provided in the next section.

[2] Quality Of Service

3. HIS-RA ONTOLOGY

3.1 General Description

In the literature many works appear concerning variability management for product derivation, from a feature model [8[[9] [19] [20] [21] [22], where characteristics directly perceived by the user are mostly considered, which is not the case of quality properties appearing later-on as implicit functionalities to satisfy the quality required by functionalities; recall the e.g. *b2-HER Man,* which requires interoperability (implicit functionality) that appears in HIS-RA as a solution mechanism (translation engine) to implement this quality requirement. However, few of these works consider variability directly from RA [10], as in our approach, nor as much the explicit handling of NFR and related quality properties, which in our case are imbedded into the RA knowledge. Notice that names of HIS-RA components and ontology elements may have a slightly different spelling, but they maintain the same semantics.

The HIS-RA Ontology presented here concerns HIS-RA representation and the capture of the HIS domain knowledge; however, notice that if RA has to be constructed for another domain by this approach, the information captured in the ontology should be changed, maintaining globally the same hierarchical structure; but this happens also with feature models [19], that are always domain specific.

3.2 The HIS-RA Ontology Development Lifecycle

The classical *V-Model* for prototype development [28] has inspired the present development process for the HIS-RA Ontology, where three models are considered: *User, Conceptualization* and *Implementation*, with a final *Evaluation* step.

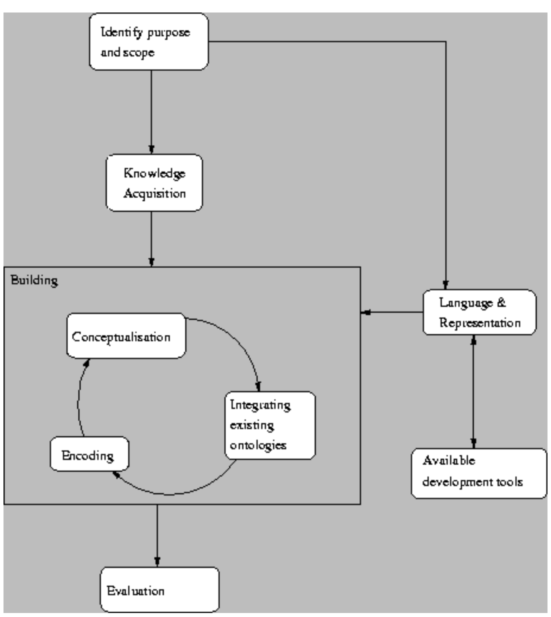

Figure 4: Ontology Lifecycle Inspired in the V-Model

According to the V-Model, the quality provided for the ontology by each model constructed on the V left-hand side following the lifecycle in Figure 4 [28], must be evaluated against the running in-use ontology, on the V right-hand side.

The HIS-RA Ontology development lifecycle considers the following steps:

User Model

- *Identify purpose and scope*: the goal of HIS-RA Ontology is to represent the HIS domain knowledge contained in HIS-RA.
- *Knowledge acquisition:* it is represented by the Business Model and Domain Analysis [17], performed to construct HIS-RA, that was built by a bottom-up process by studying existing market products used in the HIS Domain [11] [13].

Conceptualization Model

- *Conceptualization:* terminology on software quality is from the standard ISO/IEC 25010 [12]; standard HL7 was chosen for EHR interoperability [16]. The terminology on the variability model was taken from [3], and it is now adopted by the ISO/IEC 26550, new SPL reference model standard [30]. The terminology on architectural elements (components, connectors, etc.) is non-standard [15], but it is widely accepted by the software community.
- *Integrating existing ontologies*: An ontology on software quality standards, relating different standards with their general metrics was defined in [18] to improve Web services discovering; it could be integrated to HIS-RA Ontology, but it was outside the scope of the present work.

Implementation Model

- *Language and representation:* OWL (Ontology Web Language) of the W3C was chosen as one of the most used language in SPL development with ontologies [19] [21] [22].
- *Available development tools:* Protegé 5[3] for Mac OS X El Capitan, Protegé DL query, Owl Viz, OntoGraf.

Evaluation

- Each model has been validated on the prototype Protegé 5 version of HIS-RA, as specified by the V-model approach for global ontology quality: persistency, maintainability, efficiency (time and resources, availability), functional suitability (reasoning capacity). HIS-RA has been used to derive "convenient" concrete products architectural configurations in [24] [29], according to the ASSPRO process guidelines, where consistency rules were derived manually, without an ontology; the present work shows that they can be derived by querying the ontology; however, the complete evaluation of the ASSPRO derivation process using the HIS-RA Ontology is still an on-going work.

Figures 5, 6, 7, 8, and 9 illustrate examples of the expressive power of the HIS-RA Ontology, implemented in Protegé 5, running on Mac OS X El Capitan

[3] protege.stanford.edu/products.php

http://www.semanticweb.org/francesca/ontologies/2015/9/HIS-RA-24-10-15

A partial Protegé Owl Viz view of the class hierarchy is shown in Figure 5, *Common-Components* and *Variation-Points* providing architectural solutions. Figure 6 shows an OntoGraf hierarchical view of *Architectural-Solutions* and *Cost*, sub-classes of *Components* which are *Assigned-Properties* (they can change even if software does not change), respectively; *Cost* values *low, medium, high* and *undetermined* are also shown. Figure 7 presents a Protegé Description Logic Query or DL Query, showing instances *HTTPS, b7-HTTPS, b7-HTTP* as variants of the *b9-SecurityMod* variation point, displaying also an OntoGraf view of the class hierarchy; variation point *b9-SecurityMod* groups three solutions considering internet protocols, like HTTP/HTTPS in Transmission Layer, combined with modules to handle biometrics data, etc. in Process Layer.

With respect to *Security (authenticity, confidentiality, Integrity)* we have to point out that it is a HIS priority requirement; it has been accomplished in RA by combining security offered by components *d1, d2*, in transmission layer by protocols *(HTTP, HTTPS)* for messaging, with special components in Process Layer, *b7* special mechanisms, to increase the security level to satisfy *authenticity* (password, biometrics, etc.) and *confidentiality* (access policy); *integrity* is left to be solved in Data Layer by *c8* special mechanisms. We recall that HIS-RA is not a service-oriented architecture but a hybrid distributed client-server (event-based)/layered architecture.

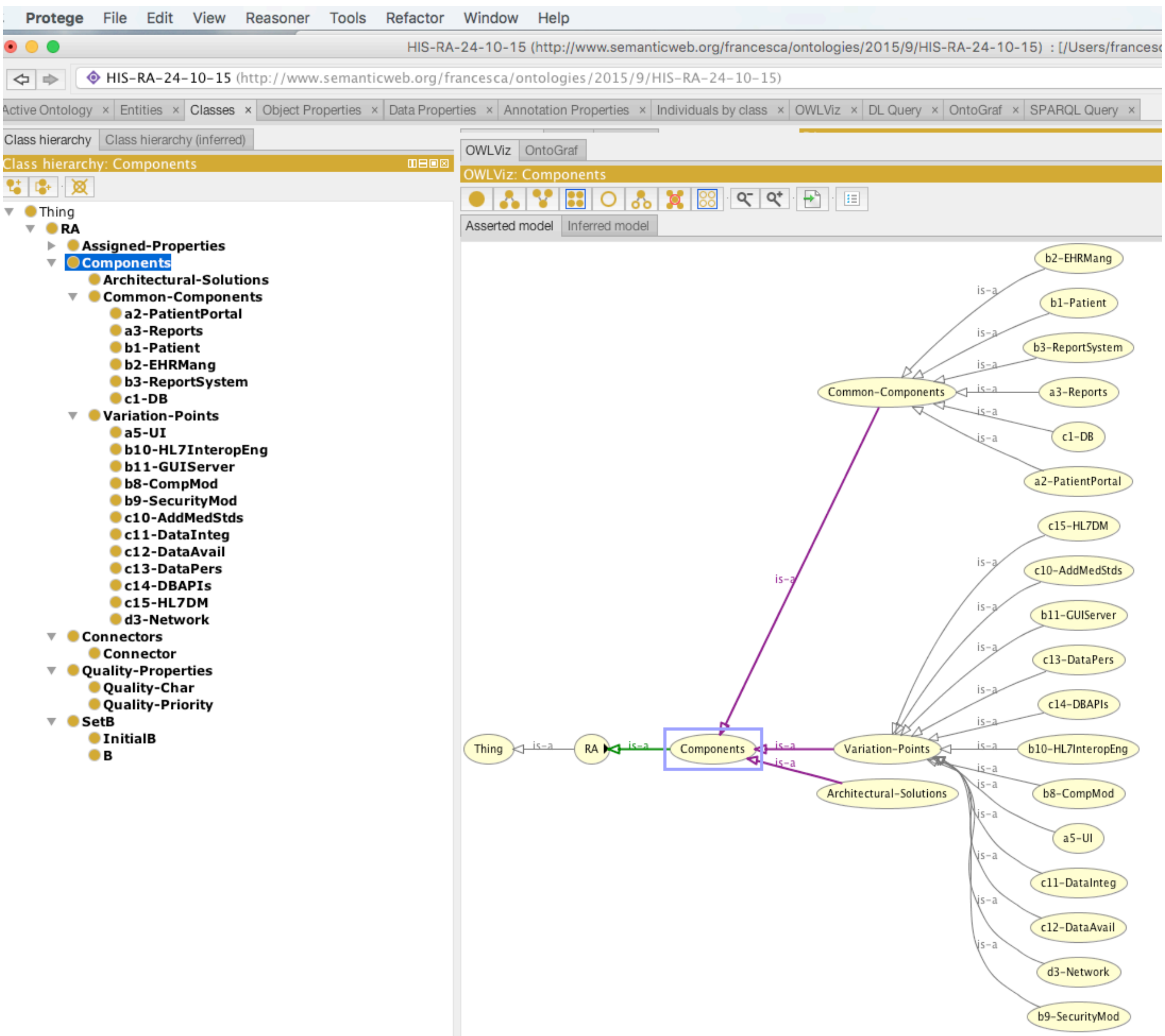

Figure 5: Owl Viz View of the Hierarchy of Common-Components and Variation-Points Classes

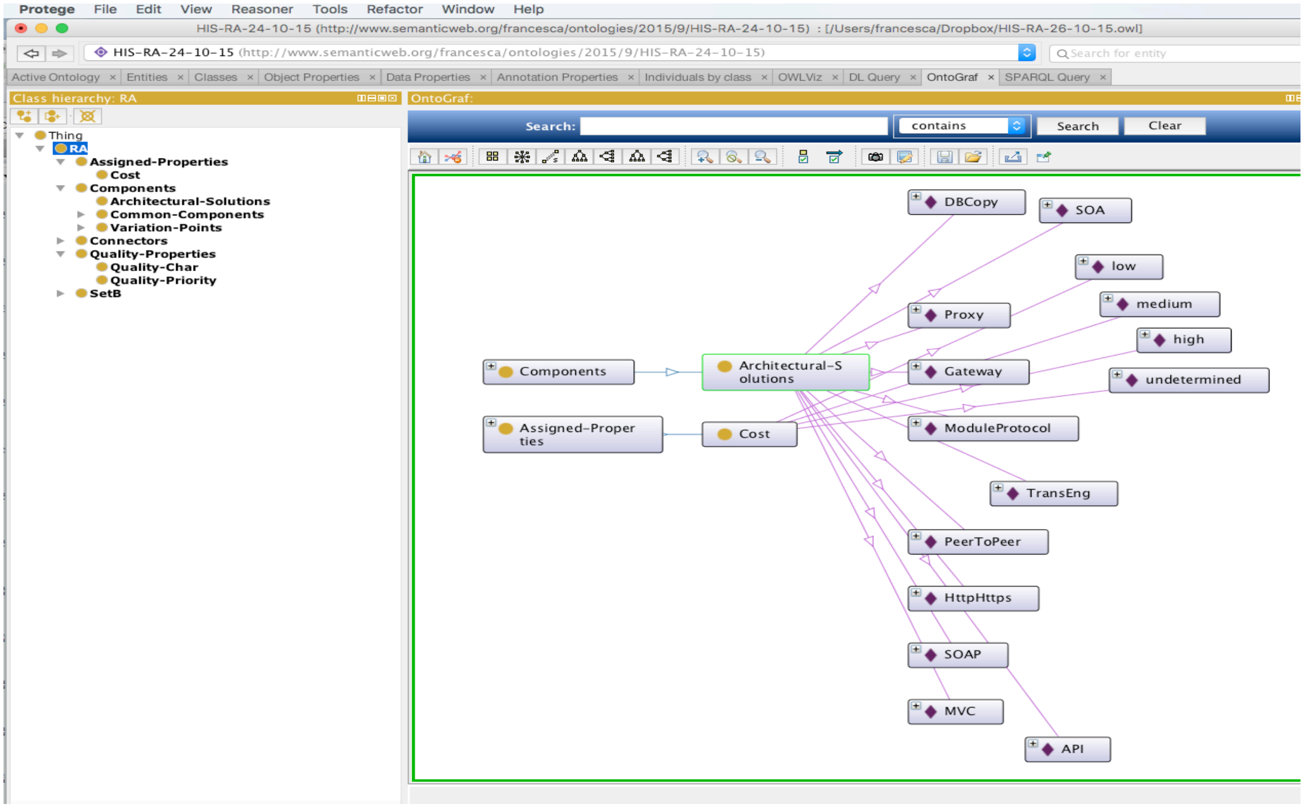

Figure 6: OntoGraf View of Architectural-Solutions and Cost Sub-classes Hierarchy, with Cost Values

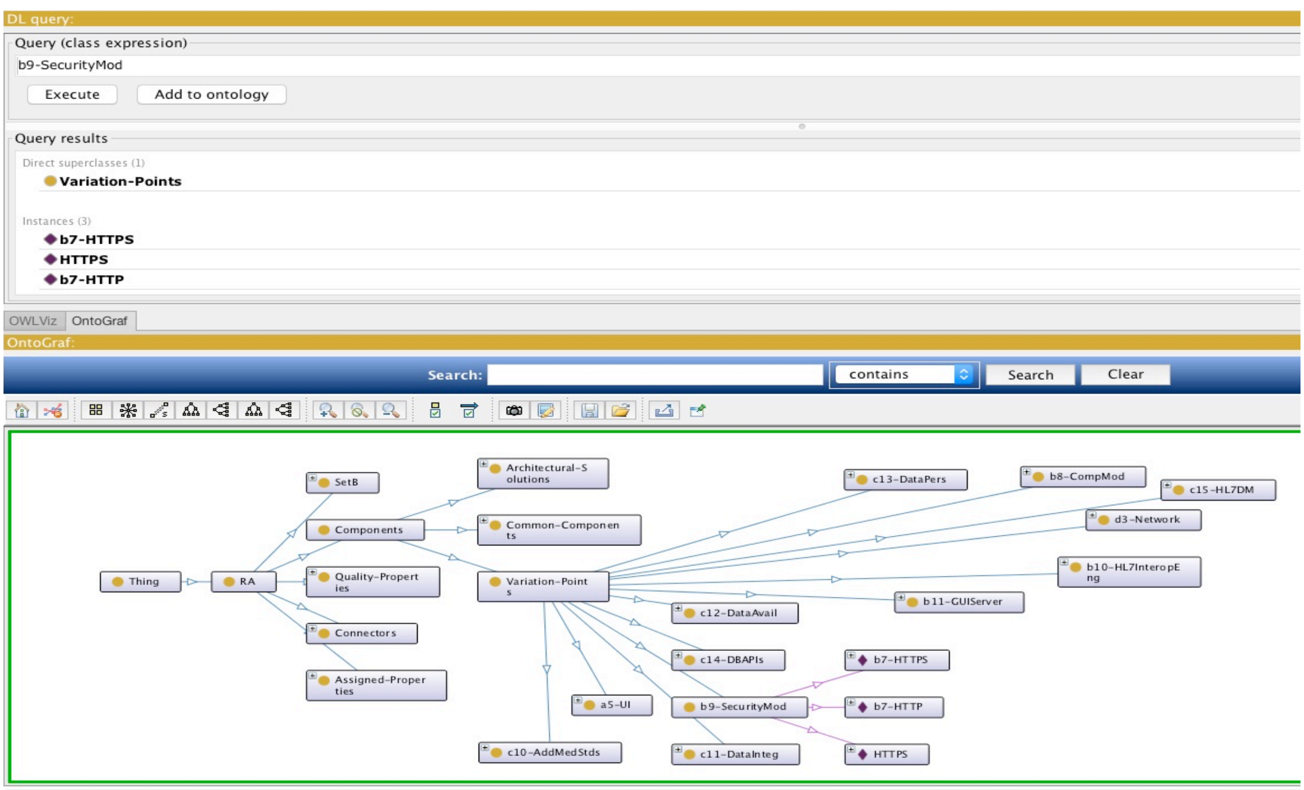

Figure 7: DL Query to Retrieve Instances *b7-HTTP, b7-HTTPS, HTTPS* as Variants of <<vp>> *b9-SecurityMod*

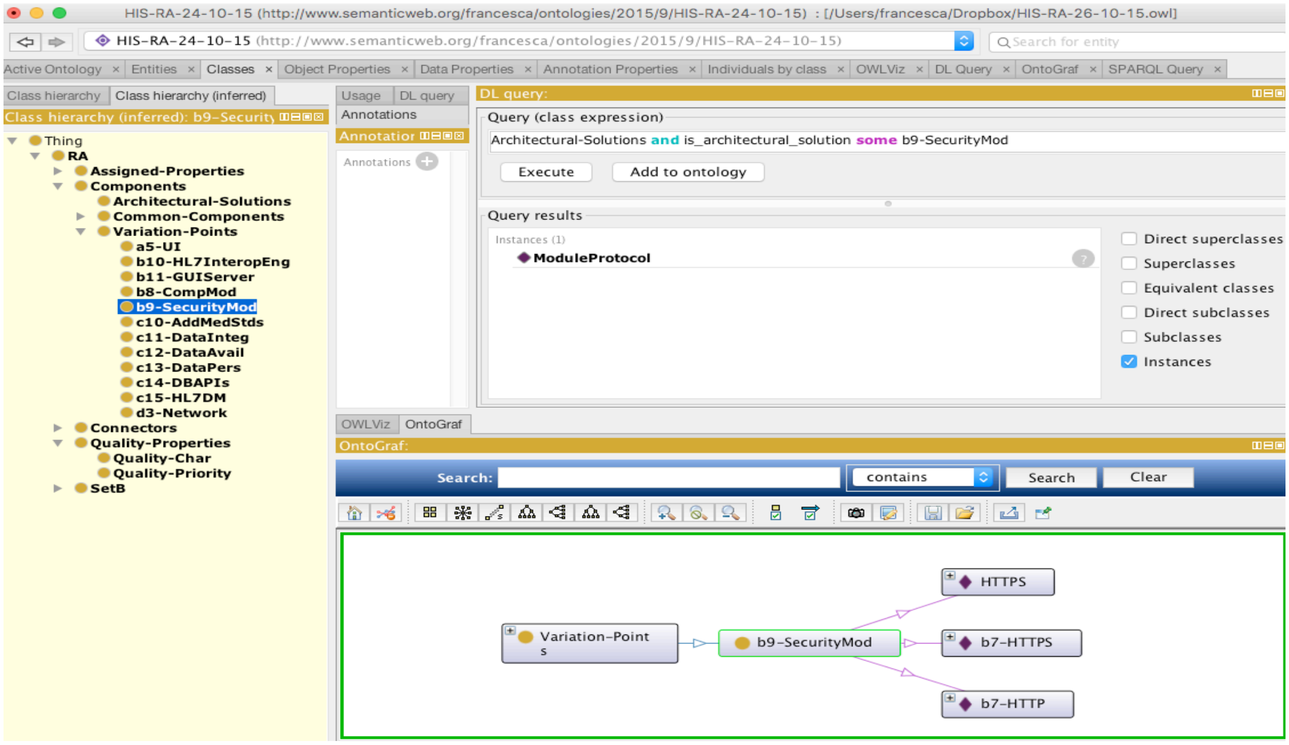

Figure 8: DL Query to Retrieve *ModuleProtocol* Architectural Solution for <<vp>> *b9-SecurityMod*; Variants for Security are also Shown

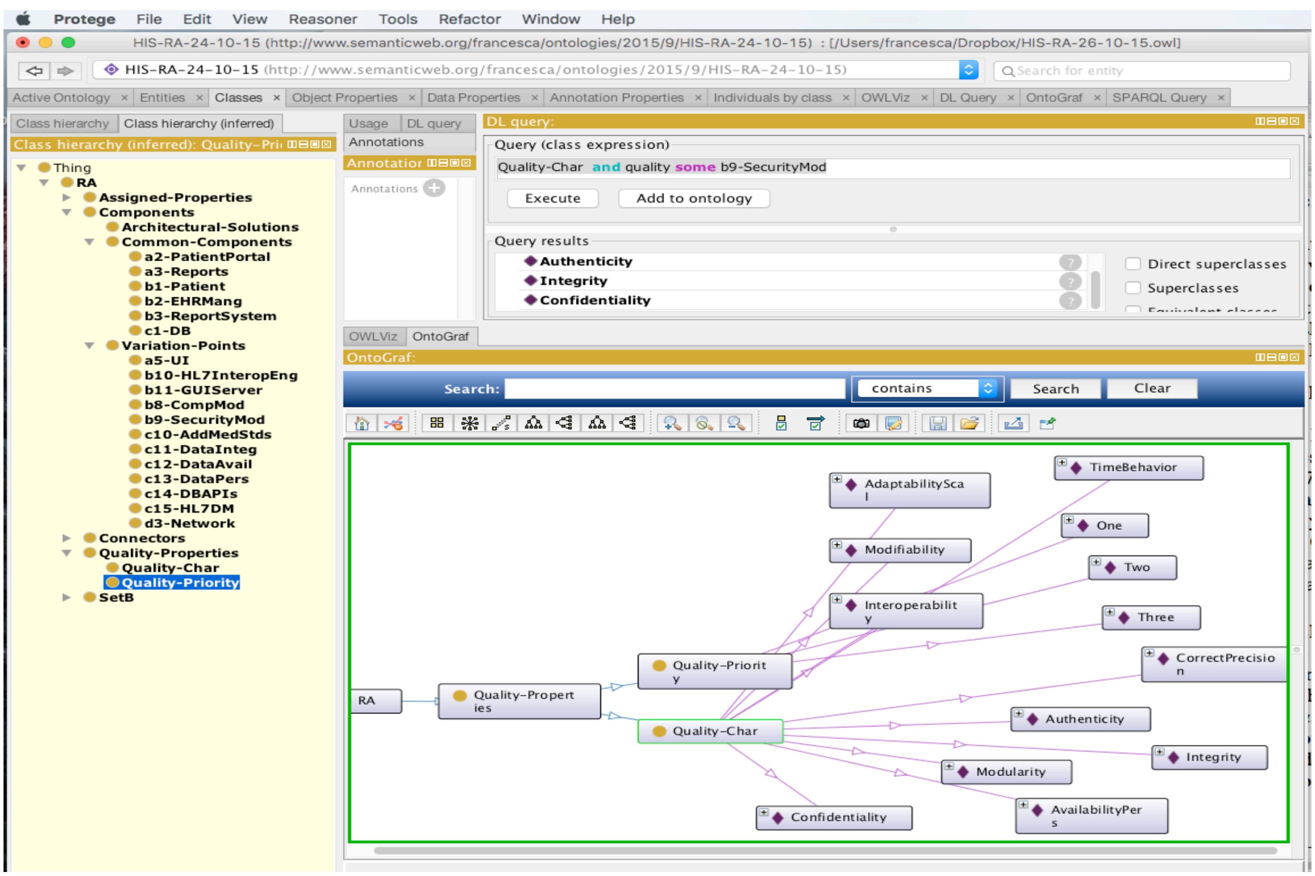

Figure 9: Hierarchical View of Class and Sub-class *Quality-Property* and *Quality-Char*; *Quality-Priority* Values are also Jndicated, with an Example of DL Query to Find Quality Properties of <<vp>> *b9. SecurityMod*

3.3 DL Query Examples
More examples of DL queries are shown in what follows:

1) Figure 8 presents the DL query

```
Architectural-Solutions and
is_architectural_solution some b9-
SecurityMod
⇨  ModuleProtocol
```

showing that instance *ModuleProtocol* (a module, if the solution is treated in Process layer or a protocol, if it is handled in Transmission Layer) is an architectural solution of *b9-SecurityMod* variation point.
Notice that *Quality-Properties* are inherent software properties, i.e., they do not change even if software chßanges) [15], and are related to *Components* by the *quality* object property to indicate that a component requires/provides this quality; Figure 9 shows an OntoGraf hierarchical view of the *Quality-Property* class and *Quality-Char* sub-class; *Quality-Priority* values, *one* (maximal priority)*, two*, and *three* are also indicated.

2) The DL query

```
Quality-Char and has_priority  exactly
1  {One}
⇨  Interoperability, Authenticity,
   Integrity. AvailabilityPers,
   Confidentiality
```

to retrieve quality properties with highest priority.

3) The DL query

```
Quality-Char and quality some b9-
SecurityMod
⇨  Integrity, Authenticity,
   Confidentiality
```

to retrieve quality properties required by the *b9. SecurityMod* variation point. Similar queries can be made for other variation points or common components to find their quality properties, for example:

4)
```
Quality-Char  and  quality  some  b2-
EHRMang
⇨  Interoperability, Authenticity,
   AvailabPers, Confidentiality,
   Integrity
```

to retrieve qualities for common component *b2-EHRMang*.
Notice that *Connectors*, with sub-class *Connector,* denote the usual connector between two architectural components *a, b,* denoted by *aRb* (see Table I). Finally, class *SetB* and sub-classes *InitialB* and *B* contain sets of selections of HIS-RA components to conform valid architectural configurations, for new SPL products' derivation from RA. *SetB* conforms a core of components that will be present in all valid architectural configurations derived instantiating RA; this process, called ASSPRO, has been defined in [29].

3.4 Representation of Constraints
Constraints appearing in feature model-based notations [19], are also specified by the HIS-RA Ontology, as data or object properties, such as *Mandatory* and *Optional* for components and *cannot_be_with* when two components cannot be present at the same time in the same architectural configuration. Other object property are *connected_to* and *directly* to specify the indirect connection by transitivity and the direct connection between two components, respectively. Data properties *is_CC* and *is_CR* indicate if a component is common or it is a customer requirement (CR), respectively; CR are used to configure the RA instances for the derivation of new SPL products according to the customer demand; *has_value* specifies a priority value for each quality characteristics, *requires_* …, *provides_* …, and *Mandatory, Optional* with range *Boolean* for each required/provided quality characteristic. Figure 10 shows the list of these properties and the available architectural solutions which were displayed graphically in Figure 6.

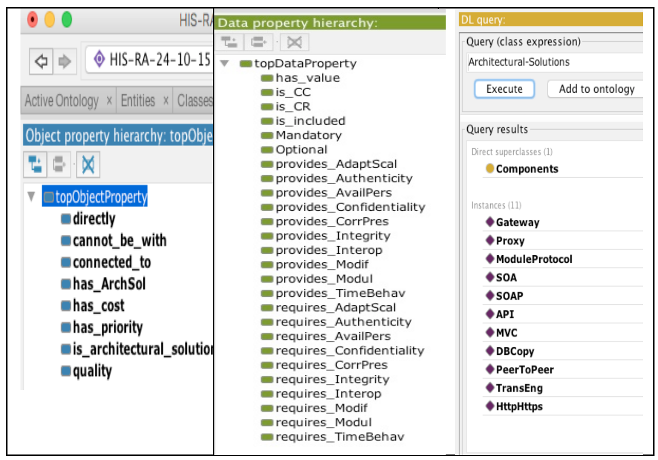

Figure. 10: Object and Data Properties Indicating Relations with Available Architectural Solutions

Figure 11 shows DL queries to retrieve mandatory and optional components.

3.5 Derivation of Consistency Rules
Relations requires/provides and constraints among components can be verified using the HIS-RA Ontology. We will start considering first the priority quality properties (see Section 2.2 HIS-DQM), namely *Interoperability, Security (Authenticiy, Confidentiality, Integrity), AvailabilityPers,* and then *AdaptabilityScal* and *CorrectPrecision*. We proceed as follows:

For each quality property, the ontology is queried by the Application Engineer (AE) to look for components requiring/providing the property. Let's take for example Interoperability which has priority 1 (see the second query example in Section 3.1):

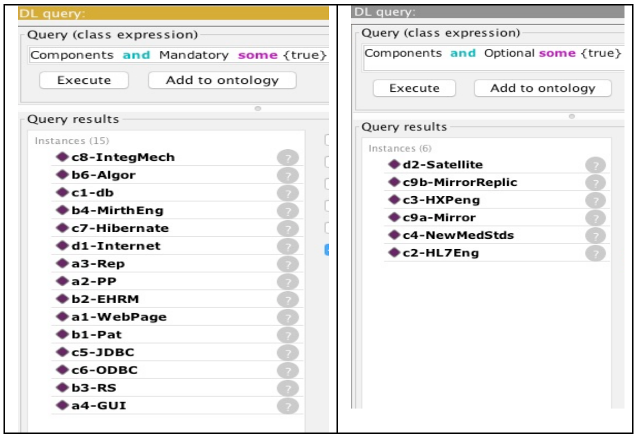

Figure 11: Mandatory and Optional Components of HIS-RA

Interoperability

From queries:

```
Components and requires_Interop some {true}
=> b2-EHRMang

Components and provides_Interop some {true}
=> b4-MirthEng, c2-HL7Eng, c3-HXPeng
```

we have that *Interoperability* is required by *b2-EHRMang* and it is provided either by variants *b4-MirthEng* or by two alternative solutions *c2-HL7Eng, c3-HXPeng*, see Figure 11; in this case, from the query, there are two possible rules to be followed, since according to the constraints, *b4* is a variant satisfying a mandatory NFR and *c2, c3* are variants satisfying optional NFR:

b2 => XOR {b4, {b4, c2}, {b4, c3}} or
b2 => XOR {b4, c2, c3, {b4, c2}, {b4, c3}}.

The expression $P \Rightarrow Q$ for the rule is interpreted as follows: *"fact P implies that fact Q must be true"*. For example *b1 AND b3* $\Rightarrow$ *b6* means that functionalities *b1-Patient* and *b3-ReportSystem* require *b6-Algo* to perform correctly their tasks with adequate precision; clause *XOR {a, b}* means that only *a* or *b* can be present at the same time in an architectural configuration; *XOR{a,{b, c}}* means that only *a* is present or only *b* and *c* are present in a configuration.

In the example shown above, AE decides for the first rule, because *b4-MirthEng* is a mandatory NFR for HIS to achieve interoperability, and it can be combined with *c2-HL7Eng, c3-HXPeng* which are optional.

For the remaining priority quality properties, similar queries are performed.

4. Related Works

Ontologies have been widely used in different stages of SPL development approaches; of particular interest for this work are the following topics related with our present research: - general RA design; - feature and variability modeling of NFRs; - domain quality modeling; - derivation of architectural configurations for concrete products from the SPL RA.

4.1 General RA Design

Three works were found very interesting and worth to be considered on this topic, even if they are not concerned with the design of SPL RA, and they will not be compared here.

The early work in [25] focuses a bottom-up approach to RA design, using a combination of guidelines from IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems and RUP[4], applying the proposition to the Learning Management System (LMS) domain as a case study. so what is really treated is a particular LMS RA. They handle domain quality properties, but they do not specify them systematically as a quality model, nor their traceability; they disappear in the model and in the corresponding RA instance; actually they could have handled it in the use case model, following RUP, as included use cases, but they did not mention this issue; they instantiate their LMS RA to a concrete LMS system. With respect to quality attributes they state: "the evaluation in most of the cases was qualitative and not quantitative as there are no metrics yet for these techniques", and this is correct; however, the presence or not of components ensuring the specific quality can be observed and measured with a Boolean "yes" or "not"; in our approach we know the domain quality properties in the HIS-DQM, and we associate to each functionality the required quality property and their possible provider; we assure that each quality property required by a functionality is fulfilled by some component providing a kind of mechanism.

Paper [26] presents a general reference model to specify RA at a high abstraction level; instantiating this model, a particular RA can be derived; main features of their reference model are actually used in our HIS-RA, thus being compliant with their model.

Finally, paper [27] defines a RA by integrating the security quality property for a cloud computing solution, stating that many cloud security issues are also true for any kind of distributed system using Web applications, as in our HIS case, where security is handled "ad hoc", as it is done in most RA design approaches [2], when quality requirements are explicitly considered; but this is not always the case in SPL, where code is generated from feature models specifications [19], which do not necessarily treat quality issues. We do not pretend to design a Security RA, because interoperability and availability should also be considered for the HIS domain, since they are HIS priority quality requirements, so we should have, besides a Security RA, an Interoperable RA (and that could be solved using for example a Service-Oriented RA, which is one of our on-going work), and w.r.t. availability, data layer will assure this aspect with standard data base replication or mirror mechanisms. In our approach, security is imbedded into HIS-

[4] Rational Unified Process

RA, as all the other HIS priority quality properties are. However, if we consider a cloud solution for Service-Oriented HIS RA, which is outside the scope of this paper, having Security, Interoperability and Availability imbedded into HIS-RA could be a good choice, following our approach.

4.2 Feature and Variability Modeling of NFR

The early work of Czarnecki, Hwan, Kim and Trygve [20] explore the relation between feature models [19] and ontologies, such as the W3C OWL[5]; basic feature models are considered a hierarchy plus a propositional formula; the notational spectrum is analyzed considering also UML [14] to establish a boundary between ontologies and feature modeling; this synergy is considered promising for the use of reasoning-based support tools.

The difficulty in designing SPL RA for service-based systems of ubiquitous computing, with highly dynamic evolutionary environments is discussed in [8]; it is claimed that feature models are incapable of capturing NFR and their fast changes at runtime; annotation of the feature model with an ontology to treat NFRs including also QoS values for product configuration, is proposed, to increase flexibility and adaptability of these systems; a different ontology is used for the "device" (features of the concrete product configuration) matching NFRs. Every configuration instance generated from the feature model also instantiates the ontology attributes with values specifying the set of capabilities that a NFR should satisfy. During the process of validating the configuration, these attributes are checked against the capabilities of the requesting device. Once the feature model ontology is fully annotated with the device ontology, they proceed to runtime analysis and reasoning over both ontologies to ensure the validity of configured products for the target device.

In [21] this subject is also treated, claiming again that feature models are not suitable to support adaptive engineering of service-oriented systems, due to their highly dynamic environment; they state that ontology languages can be easily used to express feature models, enriching them with NFRs treatment, adding inference and reasoning over constraints for product derivation of the SPL family, thus creating more adaptive service composition.

Also in the context of dynamic SPL environments [22] states that features models have limitations and must have a more formal representation in order to be dynamically reconfigured at runtime. The OntoSPL ontology is proposed to model ontology-based feature models, and a set of SPARQL queries is presented in different scenarios, that can be executed to automatically reconfigure SPL products specified in OntoSPL. On the specific topic of variability modeling, Kumbang is proposed in [9] as a domain ontology to model the SPL variability, including NFRs; it has been provided with formal semantics by implementing a translation into a general-purpose knowledge representation language with formal semantics and inference support. A prototype tool for solving variability has been implemented.

Conclusion on this topic: - feature models do not handle properly NFRs and even less in present highly changing dynamic environments; - ontological approaches help formalization and reasoning and are used to specify feature models, essentially to deal with the NFRs problem; - none of the reviewed works consider modeling the SPL RA with an ontology, nor a bottom-up approach as we do; - the concrete product configuration (based on features, which must be converted into components or modules to obtain code) is directly derived from the feature model, often represented by an ontology, to obtain runtime coded modules without considering the RA structure of components and connectors; connections between modules have to be established according to their interfaces; this step becomes very complex without RA as an intermediate abstraction level, and this issue complicates the whole derivation stage, since the structure of feature models, represented by feature trees, are not representations of architectural configurations.

4.3 Domain Quality Modeling

Quality modeling refers to model the quality of a software product [12], i.e., the product is described by a set of properties or characteristics, sub-characteristics ,…, attributes and metrics. In the SPL context, this quality must be captured first for the domain, where the SPL products' portfolio is specified and then to satisfy FRs; it is a key issue for the SPL RA design since qualities properties drive the RA design process, being the main responsible for the SPL variability model, and in the context of our work, drive the process of identifying architectural configurations based on the HIS-RA Ontology. In the SPL product configuration context, a quality property, solved or satisfied by a component which is a concrete architectural solution, is always related to FRs or NFRs and its traceability is crucial to determine which RA component is requiring/providing the quality property, in order to check the global correctness of the derived product configuration. Another problem is the terminology used in the quality properties definitions, which varies with the domain.

The following works treat the above problems and concern the Web Services (WS) domain and the handling of different quality standards by an ontology, to unify quality terminology and stakeholders' understanding.

An ontology is proposed in [18] to specify domain knowledge on software product quality; on one hand, to integrate different standards on software product quality at different abstraction levels, to unify terminology and

[5] http://www.w3.org/TR/owl-features/

characterize reusable domain knowledge; on the other hand, to facilitate WS identification based on their quality properties and the retrieval of the corresponding metrics. This characterization can be integrated in a more global approach for SPL product configuration, considering the HIS-RA Ontology defined in this work, to specify metrics that are not included at present.
This work has been applied to WS discovery in [23]; in general, standards on software quality and the relationships established between them, are not given much importance in the literature; some authors consider that standards lacks flexibility, yet these standards can be used as a shared understanding between services providers and customers, easing the WS discovery process and moreover, the wide use of standards is considered a best practice of Software Engineering, and even more in the industrial software production context claimed by SPL. An extension of OWL-S[6] is defined to describe QoS according to these quality standards. Then, an approach based on this extension of OWL-S to improve the discovery process is developed, with an extension of SPARQL that simplifies the expression of WS discovery queries. Relationships between different standards are used to return WS even if they are described with quality properties defined by a standard different from the one used to express queries. Finally, NFRs can be expressed as user preferences and are used to rank WS fulfilling FR during the discovery process.

4.4 Product Configuration from RA

Results on the generation of product architectures directly from RA are presented in [10], following ontology-based feature modeling and MDD (Model Driven Development). The work claims that the ontological approach to model features has more expressive power, it is shorter and provides less complex descriptions. The ontology is used to capture features, constraints and semantic relations between features and architectural elements, such as components and variation points. Ontology reasoning engines developed “ad hoc” are used to determine architectural elements of a feature selection (introduced by queries) and drive the generation of transformation rules that perform the concrete product derivation from the SPL RA. The ontology, written in Protegé, translates the feature tree, and the reasoning engine captures the concrete product requirements introduced as queries, which are validated for consistency against the ontology, and translated into rules; these rules are used to generate the RA instance corresponding to the queries for the concrete product, using an architecture description language (ADL); the RA instance is written in another ADL (subset of the first one) to be translated into the concrete product architecture, using the transformation rules. The support tool is OntoAD.

This work share common points with our approach, since product configurations are derived directly from RA, using an ontology. However, main differences are the following:

- their ontology specifies the feature tree [19], including certain architectural features, but not the RA, which is specified in a separate ADL; moreover, the concrete product configuration is obtained by translating the RA instance into another architectural description language using the transformation rules, due to the MDD approach;
- NFRs are not mentioned, in our opinion this is the main weakness of this interesting approach.

In our case, RA is directly specified by the HIS-RA Ontology, containing also information on functional and non functional variation points (components that are solutions to quality properties, not much used in “classic” feature models), constraints, mandatory/optional, etc. (used also in feature models). The rules used to derive concrete architectural configurations are verified with the ontology, and clear traceability between components requiring/providing quality to satisfy FRs and NFRs requirements is established using these rules. In consequence, all FRs and NFRs are satisfied in the concrete architectural configurations that are derived from RA, guaranteeing the overall quality of concrete products of the SPL family [24].

5. CONCLUSION

A SPL reference architecture modeled by an ontology, the HIS-RA Ontology, has been presented in this work to illustrate its expressive power and potential in deriving consistency rules to be used in concrete product derivation from HIS-RA. The SPL domain concerns Healthcare Information Systems, which are intensive systems with enough complexity and that still do not have generally adopted SPL; our HIS are restricted to EHR management systems, with basic HIS functionalities, for patent attention and medical reports. The HIS-RA Ontology is being used to retrieve the domain knowledge imbedded into RA; a main contribution is the ontological treatment of the quality properties related to FR, establishing their clear traceability to guarantee their complete accomplishment, and the variability model, imbedded in the ontology. The use of the HIS-RA Ontology inference capacity to derive consistency rules, based on the classic feature models constraints among components has been discussed. However we did not evaluate the ontology in this paper, because the results are being used in an on-going work to complete the ASSPRO semiautomatic process [29] to find valid feasible solutions or architectural configurations for SPL concrete products’ derivation, which were presented without the ontology; the ASSPRO support tool, involving the interaction with the AE for querying the ontology, is also an on-going work. More sophisticated queries could have been formulated with the SPARQL Protegé reasoner or other available open-source reasoners; however, the Protegé DL query tool has been sufficient for our purpose of retrieving RA information and derive consistency rules involving components’ constraints. The ontological RA model can be mapped into another

[6] www.w3.org/Submission/OWL-S

domain, however, the AE or domain expert plays a major role in the RA ontology definition and cannot be avoided.

ACKNOWLEDGMENT

We wish to thank the referees for their pertinent comments and suggestions. This research is partially funded by the Consejo de Desarrollo Científico y Humanístico (CDCH) de la Universidad Central de Venezuea, DARGRAF PG 03-8730-2013-2 project.

REFERENCES

[1] P. Clements and L. Northrop, *SPL: practices and patterns*, 3rd ed. Readings, MA, Addison Wesley, 2001.

[2] E.Y. Nakagawa, P.O. Antonio and M. Becker, *RA and PLA: a subtle but critical difference*, ECSA 2011, LNCS 6903, pp. 207-211, Springer-Verlag, Berlin, Heidelberg, 2011.

[3] K. Pohl, G. Böckle and F. van der Linden, *SPL engineering - foundations, principles, and techniques*, Springer IXXVI, pp. 1-467, 2005

[4] I. De la Torre, Y. Castaño, F. Diaz Pernaz, J. Diaz J., M. Antón, M. Martínez, D. Gonzalez, D. Bota and F. López, *Categorización de los estándares de las HCE,* Universidad de Valladolid, 2010, http://www.revistaesalud.com/index.php/revistaesalud/article/view/395/778

[5] N. Siegmund, M. Rosenmuller, M. Kuhlemann, C. Kastner, S. Apel, and G. Saake, *SPL Conqueror: Towords Optimization of Non-functional Properties in Software Product Line*, Software Quality Journal, Volume 20, Numbers 3-4, September, pp. 487-517(31), 2012.

[6] F. Losavio, O. Ordaz, N. Levy and A. Baiotto, *Graph Modeling of a Refactoring Process for Product Line Architecture Design*, JLDP, Lille, 47-58, 7-11 November 2012.

[7] T. Gruber, *Toward Principles for the Design of Ontologies Used for KnowledgeSharing*, Available as Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University, 1993.

[8] N. Kaviani, B. Mohabbati, D. Gasevic, and M. Finke, *Semantic Annotations of Feature Models for Dynamic Product Configuration in Ubiquitous Environments,* 4th International Workshop on Semantic Web Enabled Software Engineering, 7th International Semantic Web Conference, 2008.

[9] T. Asikainen, T. Mannisto and T. Soininen, *Kumbang: A domain ontology for modelling variability in software product families*, Advanced Engineering Informatics 21, pp 23–40. Elsevier, 2007.

[10] H.A. Duran-Limon, F. Castillo-Barrera E., Lopez-Herrejon and R., *Towards an Ontology-Based Approach for Deriving Product Architectures*, 15th Intern. Software Product Line Conference SPLC '11, Volume 2 Article No. 29, ACM August 21-26, Munich, Germany New York, NY, USA ©2011, 2011.

[11] F. Losavio, O. Ordaz and V. Esteller, *Quality-Based Bottom-up Design of Reference Architecture applied to HIS*, RCIS 2015, pp. 76-81, IEEE, May, Athens, Greece, 2015.

[12] ISO/IEC 25010, *Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*, ISO/IEC JTC1/SC7/WG6, 2011.

[13] F. Losavio, O. Ordaz and V. Esteller, *Refactoring-Based Design of Reference Architecture*, RACCIS, Vol 5, No 1, pp. 32-48, Enero-junio 2015, http://www.fundacioniai.org/raccis

[14] Object Management Group (OMG), *Unified Modelling Language Superstructure, version 2.0* (formal/05-07-04), August 2005, www.omg.org/spec/UML/2.0

[15] M. Shaw and D. Garlan, *Software Architecture. Perspectives of an emerging discipline*, Prentice-Hall, 1996.

[16] S. Samilovich, *OpenEMR – Historia Clínica Electrónica de codigo abierto y distribuición gratuita, apta para su uso en el sistema de salud* JAIIO CAIS, Argentina, 2010. http://www.39jaiio.org.ar/sites/default/files/Programa_CAIS_39AIIO_v8.pdf

[17] F. Losavio, O. Ordaz and I.Santos, *Proceso de análisis del dominio ágil de sistemas integrados de salud en un contexto venezolano*, ENL@CE, Vol. 12, No. 1, pp.101-134 , Enero-Abril 2015, ISSN: 1690-7515, 2015, http://www.produccioncientifica.luz.edu.ve/index.php/enlace/index

[18] F. Losavio, A. Matteo and N. Levy, *Web Services Domain Knowledge with an Ontology on Software Quality Standards*, ITA'09, UK, pp.74-85, 8-11 Sept. 2009.

[19] K. Lee, K. Kang and J. Lee, *Concepts and Guidelines of Feature Modeling for Product Line Software Engineering*, Proceedings of the 7th. International Conference on Software Reuse: Methods, Techniques, and Tools, ISBN: 3-540-43483-6, pp. 62-77, 2002

[20] K. Czarnecki, C. Hwan, P. Kim and K. Trygve, *Feature Models are Views on Ontologies*, SPIC 2006.

[21] B. Mohabbati, N. Kaviani and D. Gašević, *Semantic Variability Modeling for Multi-staged Service Composition*, Proceedings of the 13th Software Product Lines Conference, Vol 2., 2009.

[22] T. Tenorio, D. Dermeval and I. Bittencourt, *On the Use of Ontology for Dynamic Reconfiguring Software Product Line Products*, Conference Paper January 2014, ResearchGate, http://www.researchgate.net/publication/275771587

[23] S. Jean, F. Losavio, A. Matteo and N. Leyv, *An extension of Owl-S with Quality Standards*, RCIS 2010, 483-494, IEEE, Niza, France, May 10-21, 2010.

[24] F. Losavio and O. Ordaz, *Quality-Based Heuristic for Optimal Product Derivation in Software Product Lines*, ITA'15, pp. 113-129, Glyndwr, North Wales, U.K. 8-11 September 2015.

[25] P. Avgeriou, *Describing, Instantiating and Evaluating a Reference Architecture: A Case Study*, Enterprise Architect Journal, Fawcette Technical Publications, Jun. 2003.

[26] E.Y. Nakagawa, F. Oquendo and M. Beecker, *RAModel: A Reference Model for Reference Architectures*, European Conference on Software Architecture, 2012.

[27] E.B. Fernandez, R. Monge R. and K. Hashizume, *Building a security reference architecture for cloud systems*, WICSA 2014

[28] D.M. Jones, T.J.M. Bench-Capon and P.R.S. Visser, *Methodologies for Ontology Development*. In J. Cuena, editor, ITi and KNOWS Conference of the 15th IFIP WCC, pp. 62-75, London, UK, 1998.

[29] F. Losavio, O. Ordaz and H. Márquez, *Assessment for quality product derivation from a software product line reference architecture***,** RACCIS 5(2), pp. 48-59, 2015.

[30] ISO/IEC NP 26550: *Software and Systems Engineering – Reference Model for Software and Systems Product Lines*. ISO/IEC JTC1/SC7, 2013.