Ontological approach to derive product configurations from a Software Product Line Reference Architecture

Francisca Losavio,[i] Oscar Ramón Ordaz[ii] y Stephane Jean[iii]

RA Ontology for product configuration

Abstract

Software Product Lines (SPL) based on reuse, claim to improve evolution, time to market and decrease software development costs. Concrete software products or systems, members of the SPL family, are derived by instantiating a generic *Reference Architecture (RA)*, holding common and variant components. The construction of RA is a complex and costly task, as well as its usage for product derivation, due to the huge number of variants, essentially caused by non functional requirements variability. In consequence, the selection of an RA instance or *Feasible Solution (FS)*, meeting RA constraints and customer requirements, is not straightforward. In this work RA is built by a bottom-up process from existing products; RA and its instances are represented by a non-directed connected graph. The *HIS-RA Ontology* also represents RA and captures Healthcare *Integrated Information Systems (HIS)* domain knowledge. Moreover, FS must be *connected* (the induced graph by FS in RA has no isolated components), *consistent* (it verifies consistency rules among FS components), and *working* (it meets domain functional (FR) and non functional (NFR) requirements). The main goal of this paper is to define a semiautomatic process *(FFSP)*, to derive consistency rules using the *HIS-RA Ontology* built-in reasoning capabilities, to construct consistent, connected and working FS. Software quality is considered by FFSP in the traceability between FR and NFR, and it is specified by ISO/IEC 25010, to guarantee RA evolution and the overall concrete product configuration quality. FFSP is validated on a HIS domain a case study.

Keywords: software product line, reference architecture, ontology, quality product derivation, Healthcare Integrated Information System

---

[i] Doctora en Informática de la Universidad de Paris XI, docente e investigadora en la Escuela de Computación de la Facultad de Ciencias (Universidad Central de Venezuela), Coordinadora del Laboratorio de Modelos, Software y Tecnología (MoST).

[ii] Doctor en informática de la Universidad de Paris XI, Profesor titular, docente e investigador en Escuela de Matemática en la Facultad de Ciencias (Universidad Central de Venezuela), Miembro del Laboratorio de Modelos, Software y Tecnología (MoST).

[iii] Profesor de la Universidad de Poitiers y miembro del Laboratoire d'Informatique et d'Automatique pour les Systèmes.

Resumen

Las *Líneas de Productos de Software (LPS)* basadas en reutilización, mejoran la evolución, el tiempo de comercialización y los costos del desarrollo. Productos o sistemas de software, miembros de la familia SPL, son derivados instanciando una *Arquitectura de Referencia (AR)* genérica, que contiene componentes comunes y variantes. La construcción de AR es tarea compleja y costosa, así como su uso en la derivación de productos, debido al gran número de variantes, causado por los requisitos no funcionales. Como consecuencia, la selección de una instancia de AR (solución factible o *"Feasible Solution (FS)"*, que cumpla con restricciones y requisitos del cliente, no es directa. En este trabajo AR es construida por un proceso ascendente *("bottom-up")*, a partir de productos existentes; un grafo conexo no dirigido representa RA y sus instancias. La ontología *HIS-RA Ontology* también representa AR, capturando el conocimiento del dominio de *Sistemas de Información Integrados de Salud (SIS)*. Además, FS deben ser conexas, consistentes y convenientes ("*working*") respecto a las tareas que desempeñan para cumplir con requisitos funcionales (RF) y no funcionales (RNF). El objetivo de este trabajo es definir un proceso semiautomático *(FFSP)*, para derivar reglas de consistencia usando herramientas de razonamientos de *HIS-RA Ontology*, para construir FS *consistentes*, *conexas* y *convenientes*. En FFSP la calidad del software, especificada por ISO/IEC 25010, es considerada en la trazabilidad entre RF y RNF, garantizando la evolución de AR y la calidad global de la configuración del producto. FFSP es validado sobre un caso de estudio en el dominio SIS.

Palabras clave: línea de productos de software, arquitectura de referencia, calidad en la derivación del producto, Sistema de Información Integrado de Salud

I. Introducción

In the context of industrial software production, a Software Product Line (SPL) is a family of software intensive systems or 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)* or generic frame containing a common set of assets that are reused in different products of the SPL family [1]. In the literature [2] there are some differences between Product Line Architecture (PLA) and RA according to the design process (proactive top-down or extractive/reactive bottom-up) followed; in this context, however, the more widely used term RA will be considered. RA is represented by a connected graph (*P*, *R*), where *P* are nodes or *components* and *R* are edges or *connectors*. *Feasible Solutions (FS)* are "convenient" architectural configurations conformed by components, connectors and behaviors [3]; FS are derived from RA instances and must be connected, consistent and working. FS is said to be *connected* when the induced graph by FS in RA has no isolated components, *consistent* when it is compliant with consistency rules relating FS components, and *working* when it is compliant with main domain functional (FR) and non functional requirements (NFR).

The main goal of this paper is to define a process, called *FFSP: FindFS Process*, to derive consistency rules among FS components and construct connected, consistent, and working FS, using an ontological representation of RA, which captures the domain knowledge. FFSP is a semiautomatic process; the intervention of the *Application Engineer (AE)*, domain expert or design engineer plays a major role interacting with FFSP. Our approach is not based on the classic FODA (Feature-Oriented Domain Analysis) [6], as usual SPL engineering top-down approaches do [7]. In SPL, feature-based product configuration is the process of selecting the desired features for a given software product from a repository of features called a feature model [27]. This process is usually carried out collaboratively by people with distinct skills and interests (stakeholders). Collaboration benefits stakeholders by allowing them to directly intervene in the configuration process. However, collaboration also raises an important side effect, i.e., the need of stakeholders to cope with decision conflicts. Conflicts arise when decisions that are locally consistent cannot be applied globally because they violate one or more constraints in the feature model. Product configurations are directly derived from the feature model into coded modules, without considering at all the architectural structure organizing the features, causing further problems with the related architectural documentation, which in SPL is also considered an asset; the analysis of the FODA variability model is done applying constraints programming techniques [26] [27] [28], or ontology-based approaches [21] [29] [30] [38], which are used to verify consistency of variability models and/or represent feature models. However,

notice that a feature model is not an architectural model; according to FODA [6], only directly users-perceived, usually FR, are considered. In our ontological approach, RA components and connectors [3], which are architectural elements and not "features", are considered and our variability model is defined on these elements, including NFR required by FR, which is a major limitation of FODA; hence in our approach a clear traceability between FR and NFR is enhanced, to guarantee an evolutionary RA, which is one of the major SPL requirement. To cope with this problem, many adaptations of features models can be found in the literature to consider also NFR [7] [8] [9] [10] [11] [25]. In our approach, FS are derived directly by instantiating RA, which has been constructed by a bottom-up (extractive) semiautomatic process [12] [13], by refactoring architectures of existing products on the market. The bottom-up approach is claimed to be faster, less expansive, and more practical for industrial usage than classic SPL top-down approaches; however their limitation is that the number of existing products can be small [4]. In our case, domain knowledge has been captured from the available documentation on three widely used existing products and it is imbedded into RA, which has been built considering explicitly quality properties related to FR, capturing also the overall domain quality [12] [13]. However, if our initial baseline or candidate architecture obtained automatically from the union of the architectural configurations of the existing products considered, is found limited due to the bottom-up construction, it could be extended with new components, by studying future products that can be built for the SPL. The "modus operandi" to find *connected*, *consistent* and *working* FS by instantiating and operating *variation points* [4], denoted by <<*vp*>>, is supported by FFSP. FS are sets containing RA components conforming to an architectural configuration of a concrete product of the SPL family. A <<*vp*>> groups a set of *variants* sharing similar tasks, and they can provide different architectural solutions to satisfy quality properties and/or FR; they denote variability of FR and NFR. On the other hand, ontologies are tools used to specify domain knowledge [14] and are found in different SPL approaches, mostly to verify the consistency of variability models in feature models, as we have already pointed out, to help reasoning for product derivation [7].

An ontology is used in our approach to check the validity of deriving connected, consistent and working product configurations directly from the RA configuration, without involving feature models. Our RA, called HIS-RA, has been constructed for the *Healthcare Integrated Information Systems (HIS)* domain [13] [19]. The knowledge on HIS quality properties is imbedded into HIS-RA by construction. The *HIS-RA Ontology* has been defined in this work to represent HIS-RA and provide support to FFSP.

Besides this Introduction and the conclusion, the paper is structured as follows: section 2 presents the work context, describing HIS domain and HIS-RA; section 3 defines HIS-RA Ontology; section 4 presents FFSP. In section 5, FFSP is applied and

validated on a case study considering given customer requirements for a concrete product in HIS domain; finally section 6 discusses works related to this research topics.

II. Context

FFSP concerns the product specification stage of the SPL Derivation phase, in the Application Engineering lifecycle [4], [5]. Three basic stages [16] are considered: Product Specification, Product Development and On-site Product Configuration, see Figure 1. In this paper we are concerned only with the first step of Product Specification, the *Find Feasible Solutions* step. Our RA has been constructed in the Domain Engineering lifecycle [4] by a bottom-up process for the HIS domain, as it was mentioned above, considering three existing open-source products on the market. The representation of RA by a graph was used to derive automatically a first SPL candidate architecture after a similarity analysis of the architectural configurations of the existing products [12] [13]. Knowledge about NFR related to FR is specified by a standard quality model [15], to facilitate common understanding of software product quality terminology. FFSP is quality-centric, focused on the satisfaction of the quality properties required by the components instantiated from RA to conform concrete product configurations; note that quality properties are the main responsible of the SPL variability [17]. The goal is to ensure that this configuration will fulfill the product required quality, besides the usual functionality.

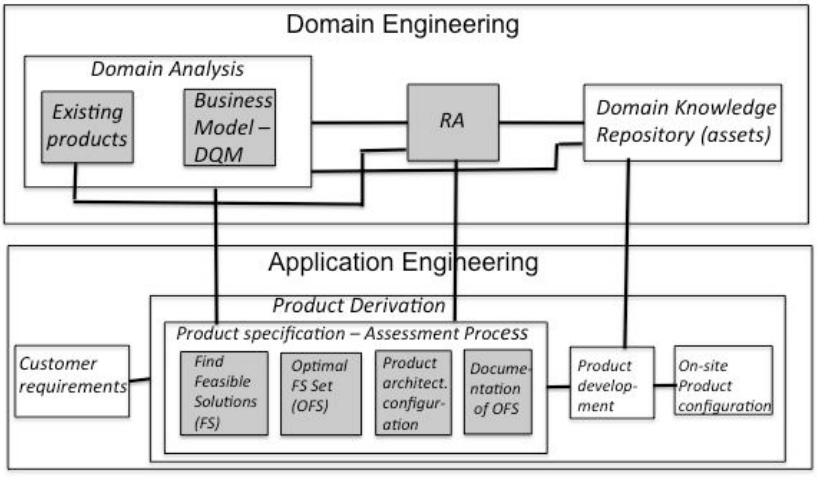

Fig. 1. SPLE Derivation Process – step Find Feasible Solutions corresponds to FFSP. Adapted from [16] and [39]

II.1 HIS Domain

For the scope of this study, the HIS domain is restricted to its basic functionalities, namely EHR (Electronic Health Records) management, patient attention for appointment scheduling and capture of demographic data, and medical reports including basic administrative services. Imaging and laboratory services, hospital rooms' management, nursing services, urgencies, etc. will not be considered here [18] [19]. HIS are complex information systems, generally located in different and distant institutions and with mandatory NFR requirements, such as interoperability, availability and security. They are supported by a hybrid event-based Client-server/Layers architectural style [3] [13], used in distributed Web-based systems. HIS must facilitate transparent sharing of different kinds of medical information, such as EHR, offering also telemedicine services that can be performed on-line 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 and other psychosocial factors makes difficult the sharing (interoperability) of EHR, and HIS general adoption is still difficult. Network providers are now offering HIS cloud solutions, which will not be discussed in this work, but could be considered to study HIS-RA evolution. According to [18] [19], the open-source systems OpenEMR, PatientOS with a 90% and 92% usage respectively, and Care2X, similar to OpenEMR, are used recently in health national projects. They have been refactored into the HIS-RA [12][13], see figure 3.

The HIS domain quality model (HIS-DQM) [12] [13] [15][19], shown in figure 2, constituted by seven *inherent* quality characteristics (that do not change even if software changes) and sub-characteristics, correspond to HIS global NFR. HIS-DQM was adapted in [12] [13] from [15] 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, the *quality attributes* are attained. In particular, priority has been added to this information, *(1≤ priority ≤ 3, where 1 is maximum)*, since it is relevant to consider instances of HIS-RA choices. HIS-DQM quality characteristics/sub-characteristics with their priority are: c*ompatibility (1) (interoperability), security (1) (authenticity, confidentiality, integrity), reliability (1) (availability-persistency), functional suitability (2) (completeness, correctness-precision), portability (2) (adaptability-scalability (2)), maintainability (3) (modifiability, modularity), and efficiency-performance (2) (time-behavior)*.

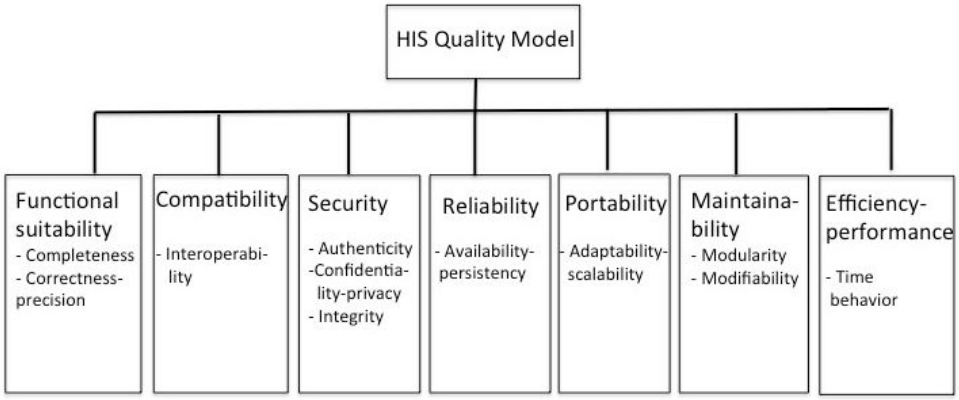

Fig. 2. Quality Model for the HIS domain (HIS-DQM) – adapted from ISO/IEC 25010 [12] [13].

Quality attributes could be included with internal metrics at this early stage of development, such as the existence (yes/no) of a mechanism to handle data availability-persistency as mirrors/replication mechanisms, or the time-behavior value of certain communication protocols involving algorithms to handle security or certain QoS[1] measures [20]. However, quality metrics are outside the scope of this work, which aims to establish a clear traceability among RA components requiring/providing quality properties.

II.2 HIS-RA

A UML[2] representation of HIS-RA is shown in figure 3, with variation points, common components and their connectors, representing the RA variability model to be instantiated for concrete product derivation in the SPL family. Functional variability is handled as usual, however non functional variability is treated extensively and it has been considered in RA by construction. Notice that HIS-RA is compliant with HIS-DQM; all quality properties are explicitly accomplished by some variant or component; *completeness* w.r.t. functionality, is implicitly assured because HIS-RA contains the core of common components, and *time-behavior* is implicitly contemplated in network protocols. In this work the RA instantiation to derive architectural configurations for a concrete product from established rules will be considered in FFSP. Notice that for a <<*vp*>> corresponding to a NFR, for example <<*b9*>> *Security Modules*, there are three variants which are solutions for the *security* quality sub-characteristics *confidentiality*, *authenticity*, *integrity* respectively (see figure 2) they are required by components *b1 Patient*, *b2 HER Man*. and *b3 Report System* (see figure 3); the connections between components

[1] Quality Of Service.
[2] Unified Modeling Language, www.omg.org/spec/UML/2.0

are of type provide/require, in the sense that, for example, *b1 Patent* requires security, that will be provided by *<<b9>> Security Modules* using its variants to solve *authenticity* and *confidentiality* and so on; finally a solution for *security* is provided to *b1*, *b2* and *b3*. This knowledge on RA and its domain is captured in the ontology that will be described in the next section.

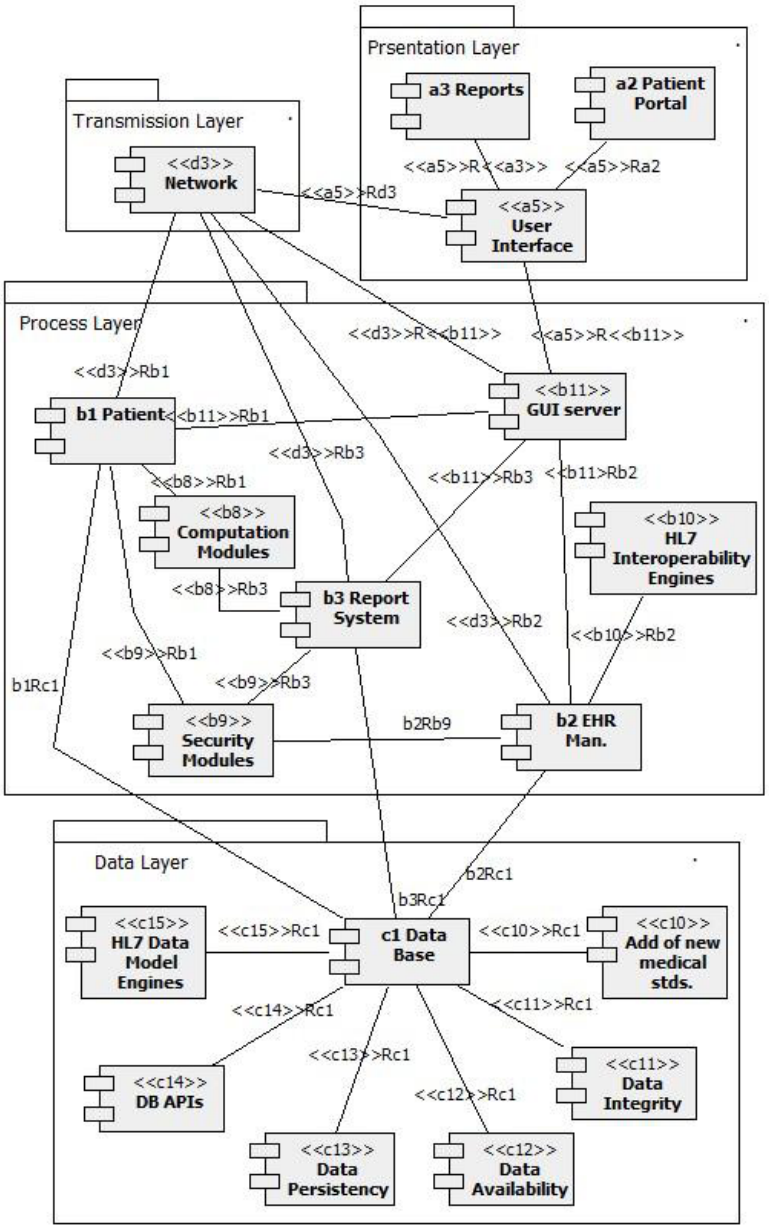

Fig. 3. HIS-RA with variability model [13]

III. HIS-RA Ontology

Ontologies represent knowledge organized in a hierarchical and structured way [14]. In the literature many works appear concerning variability management for product derivation from a feature model [8] [9] [10] [11] [21] [29] [30] [38]. However, few of them consider variability directly from RA [23] [24] [39], as in our present approach, nor as much the explicit handling of NFR. Our HIS-RA Ontology contains the domain knowledge imbedded in HIS-RA, including the variability model, and it is not a feature model representation but the representation of an architectural model. RA is generally constructed for a specific SPL domain; if it has to be constructed for another domain by our approach, the information captured in the ontology should be changed, maintaining globally the same hierarchical structure; but this happens also with feature models representations, that are domain specific. A partial view of the class hierarchy of HIS-RA Ontology is shown in figure 4; its main classes specify the following basic information on HIS-RA:

- *Common-Components;* - *Variation-Points* with their variants providing architectural solutions; for example, 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, see figure 5; - *Connectors*, with sub-classes *Connector* denoting the usual connector between two components *a*, *b*, denoted by *aRb*; - *Quality-Properties* are inherent quality properties [15]; they are related to *Components* by the *quality* object property to indicate that a component requires/provides this quality; - *SetB*, with sub-classes *InitialB* and *B* containing selections of HIS-RA components to conform convenient FS.

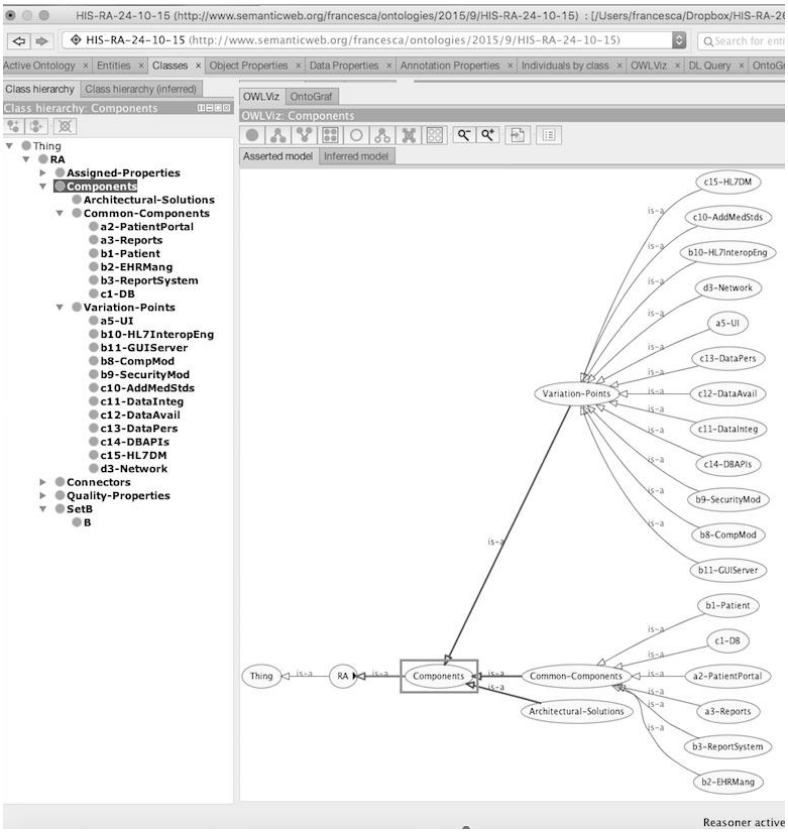

Fig. 4. HIS-RA Ontology - partial view of the class hierarchy; source: authors

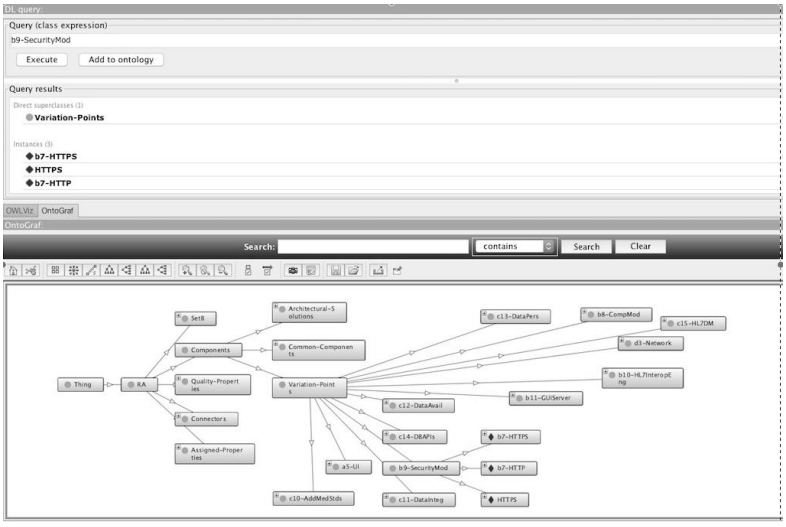

Fig. 5. HIS-RA Ontology - variants b7-HTTP, b7-HTTPS, HTTPS of <<vp>> b9- SecurityMod; source: authors

Constraints that usually appear in feature models, are also specified in 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. Class *Assigned-Properties* with sub-class *Cost*, specifies the cost assigned to variant components; *has_ArchSol* and its inverse *is_architectural_solution* are object properties to relate architectural solutions with *Common-Components* and *Variation-Points*; the object property *has_cost* (*low, medium, high, undetermined*) relates *Variation-Points* and *Architectural-Solutions* with cost value; *Common-Components* do not require explicit cost because they are mandatory; *has_priority* (*one*, *two*, *three*) is an object property assigning priority to each *Quality-Char*; other object properties 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 is a customer requirement, respectively; has_value specifies a priority value for each quality characteristics, *requires_ ...*, *provides_ ...*, and *Mandatory*, *Optional* with range "boolean" for each required/provided quality characteristics. The list of these properties and available architectural solutions is shown in figure 6. Queries for mandatory and optional components are shown in figure 7, see also comments in Table 1.

HIS-RA Ontology

http://www.semanticweb.org/francesca/ontologies/2015/9/HIS-RA-24-10-15
runs on Protegé 5,[3] for Mac OS X El Capitan.

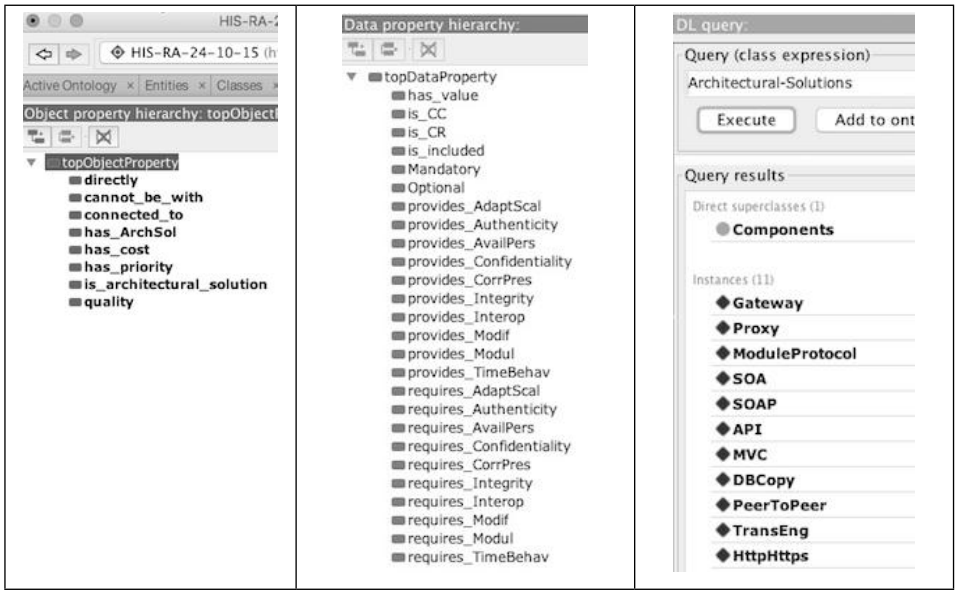

Fig. 6. Object and Data Properties with DL query for available architectural solutions; source: authors

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

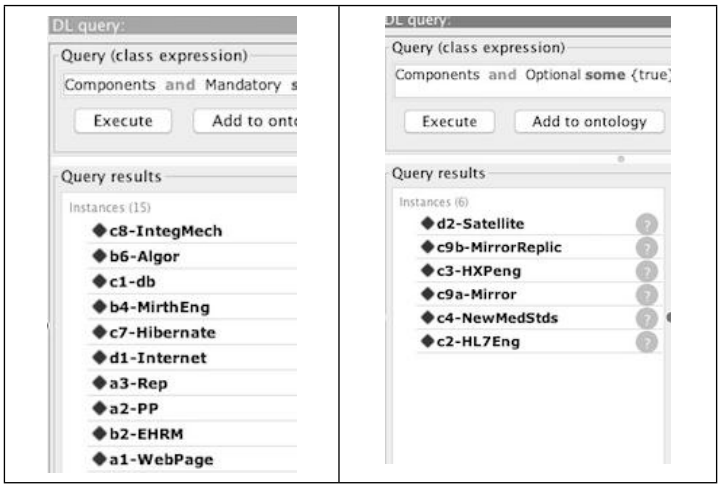

Fig. 7. Mandatory and Optional components of HIS-RA; source: authors

The Application Engineer (AE) constructs the *Constraints Table (CT)* from the HIS-RA Ontology; Table 1 shows a quick glance of the ontology. CT contains, namely, *Common-Components (CC)* and *Variation-Points* with their variants (instances of Variation-Points in the ontology) in columns 1 and 2 respectively. Column 3, for *Consistency Rules*, which are presented now in Table 1 to abridge this presentation, but they are actually to be derived in Section IV by FFSP, from relations among ontology elements.

In the consistency rules, let *P*, *Q* be Boolean expressions involving functional and variant components, respectively. Clause *XOR {a, b}* or a *XOR b* means that only *a* or *b* can be present at the same time in a configuration, and *XOR {a,{b, c}}* means that only *a*, or only *b* and *c* are present. Clause *a AND b* means that components a and b are both present in a configuration. The expression *P => Q* is interpreted as follows: *"fact P implies that fact Q must be true"*. For example *b1 AND b3 => b6* means that functionalities *b1-Patient* and *b3- ReportSystem* require *b6-Algo* to perform correctly with adequate precision. Column 4 in Table 1 shows the *Quality-Properties* related to each component, representing the HIS-DQM; CC, representing the RA functionality, *requires Quality-Properties*, and non functional variants *provide* these quality properties, with the exception of d1-Internet which is a CC and its Quality-Properties are satisfied directly by network protocols. In column 5, *Constraints* among components are presented; they are also used by AE for the derivation of the consistency rules.

Table 1. Constraints Table (CT); source: HIS-RA Ontology

Components	Variants	Consistency Rules	Quality Properties	Constraints
a5-UI	***a1-WebPage (Browser)***	*XOR {a1, a4};*	- AvailabilityPers, Authenticity, Confidentiality, Integrity, TimeBehavior, AdaptabilityScal, Interoperability (a1)	**a1**, **a4** cannot be present together in an Architectural Configuration (AC); one of them **is mandatory FR**; modif. and modul. are solved by MVC in a4 and b5; time-behavior, authenticity, confidentiality, integrity, are solved by internet protocols (HTTP, HTTPS); availability depends on network connectivity for a1 and a4. However all other quality properties will be considered in Process and Data Layers. Usability does not depend on the architecture or style, it has not been considered in this context and it is not part of the HIS-DQM
	a4-GUI (stand alone)		- AvailabilityPers, Modifiability, Modularity, Authenticity, Confidentiality, Integrity, TimeBehavior (a4)	
a2-Patient Portal, a3-Reports		*a2 AND a3;*	- TimeBehavior , Modularity, Modifiability, Usability (a2) - TimeBehavior, Modularity Modifiability, Usability (a3)	**a2, a3 are CC mandatory FR**; they are UI push buttons, they are part of a1 and a4, and the same quality properties for a1 and a4 hold
Components	**Variants**	**Consistency Rules**	**Quality Properties**	**Constraints**
b1-Patient		*- b1 AND b3 => b6;* *- b1 AND b2 AND b3 AND c1 => XOR {{c8, b7-HTTP}, {c8, HTTPS}, {c8, b7-HTTPS}};* *- b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}};*	- CorrectPrecision (b1) - Authenticity, Confidentiality, Integrity (b1) - AvailabilityPers (b1) - AdaptabilityScal (b1)	**b1**, **b2**, **b3** are CC **mandatory FR** - satisfied by b6, which is a **mandatory NFR** - only one of the three alternatives holds in AC; c8 **is mandatory NFR** for database - only one of the two alternatives holds in AC; c7 is **mandatory** for database; c9a, c9b optional, only one can be present in AC
b2-HERMang		*- b1 AND b3 AND c1 =>XOR {{c4, c5}, {c4, c6}};* *- b2 => XOR {b4, {b4, c2}, {b4, c3}};*	- Interoperability (b2) - AvailabilityPers (b2)	- only one of the two alternatives holds in AC; **c5 or c6 are mandatory NFR** for database; but only one of them can be present in AC; - only one of the three alternatives holds in AC; c2 or c3 are optional, but **b4 is mandatory HIS NFR**
b3-Report System		*- b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}};* *- b1 AND b2 AND b3 AND c1 => XOR {{c8, b7-HTTP}, {c8, HTTPS}, {c8, b7-HTTPS}};* *- b1 AND b3 AND c1 =>XOR {{c4, c5}, {c4, c6}};* *- b1 AND b2 AND b3 AND c1=> XOR {{c8, b7-HTTP}, {c8, HTTPS}, {c8, b7-HTTPS}};* *- b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}};* *- b1 AND b3 => b6;*	- Authenticity, Confidentiality, Integrity (b2) - AdaptabilityScal (b3) - Authenticity, Confidentiality, Integrity (b3) - AvailabilityPers (b3) - CorrectPrecision (b3)	- only one of the two alternatives holds in AC; c7 **is mandatory NFR**; - only one of the three alternatives holds in AC; c8 is **mandatory NFR** for database; - only one of the two alternatives holds in AC; c4 is CR; c5 or c6 is **mandatory NFR** for database - only one of the two alternatives holds in AC; c8 is **mandatory** for database - only one of the three alternatives holds in AC; c7 is **mandatory NFR** for database - satisfied by b6; **b6 is mandatory NFR**

Continúa...

b10-HL7 InteropEng	***b4-MirthEng***	*- b2 => XOR {b4, {b4, c2}, {b4, c3}};*	- Interoperability (b4)	- only one of the three alternatives holds in AC; but components in one of the alternatives are mandatory HIS NFR
b11-GUIServer	***b5-GUI ServerSide***	*b5 <=> a4;*	- Modifiability (b5), Modularity (b5), TimeBehavior (b5)	- satisfied by MVC
b8-CompMod	***b6-Algor***	*b1 AND b3 => b6;*	- CorrectPrecision (b6)	- satisfied by b6, **mandatory NFR**
b9-Security Mod	***- b7-HTTP - HTTPS - b7-HTTPS***	*b1 AND b2 AND b3 AND c1 => XOR {{c8, b7-HTTP}, {c8, HTTPS}, {c8, b7-HTTPS}};*	- Authenticity, Confidentiality, - Integrity *(b7-HTTP, HTTPS, b7-HTTPS)*	- only one of the three alternatives holds in AC and it is **mandatory**; b9 solves security (auth., confid., integ.); combined with protocols HTTP/HTTPS, in Transmission Layer; **c8** is **mandatory NFR** for database
Components	**Variants**	**Consistency Rules**	**Quality Properties**	**Constraints**
c1-DB		*- b1 AND b3 AND c1 =>XOR {{c4, c5}, {c4, c6}}; - b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}}; - b1 AND b2 AND b3 AND c1 => XOR {{c8, b7-HTTP}, {c8, HTTPS}, {c8, b7-HTTPS}};*	- AdaptabilityScal (c4, c5, c6) - AvailabilityPersistency (c7, c9a, c9b) - Integrity (c8)	**c1 is mandatory FR; c5** or **c6 are mandatory NFR** - only one of the two alternatives holds in AC, c4 is CR, c5 or c6 are **mandatory NFR** - satisfied by c7, c9a or c9b; c7 **mandatory** for database - satisfied by c8, **mandatory** for database
c15-HL7DM	***c2-HL7Eng c3-HXPeng***	*=> b2 => XOR {b4, {b4, c2}, {b4, c3}};*	- Interoperability (c2, c3)	- only one of the three alternatives holds in AC; c2, c3 optional
c10-AddMed Stds	***c4-NewMed Stds ****	*- b1 AND b3 AND c1 => XOR {{c4, c5}, {c4, c6}}.*	- AdaptabilityScal (c4)	- only one of the two alternatives holds in AC; c4 optional, c5 or c6 **mandatory NFR** for database
c14-DBAPIs	***c5-JDBC c6-ODBC***	*- b1 AND b3 AND c1 =>XOR {{c4, c5}, {c4, c6}};*	- AdaptabilityScal (c5, c6)	- only one of the two alternatives holds in AC; ; c4 optional, c5 or c6 **mandatory NFR** for database APIs provide portability to Oracle, MySQL, PostgreSQL;
c13-DataPers	***c7- Hibernate***	*- b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}};*	- AvailabilityPers (c7)	- only one of the two alternatives holds in AC; c7 is **mandatory** for database
c11-DataInteg	***c8-IntegMech ****	*- b1 AND b2 AND b3 AND c1=> XOR {{c8, b7-HTTP}, {c8, HTTPS}, {c8, b7-HTTPS}};*	- Integrity (c8)	c8 is **mandatory** for database; it does not guarantee authenticity and confidentiality, it must be used with b9 security solutions; only one of the three alternatives holds in AC, and it is **mandatory**;
c12-DataAvail	***c9a-Mirror c9b. MirrorReplic.***	*- b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}};*	- AvailabilityPers (c9a, c9b)	- only one of the two alternatives holds in AC; c9a, c9b optional; c7 **mandatory** for database
d3-Network	***d1-Internet*** ***d2-Satellite***	*XOR {d1, {d1, d2}};*	- TimeBehavior, AvailabilityPers, AdaptabilityScal, Authenticity, Confidentiality, Integrity (d1) - TimeBehavior, AvailabilityPers, AdaptabilityScal, Authenticity, Confidentiality, Integrity (d2)	**d1 is mandatory FR**, d2 is optional FR; AdaptabilityScal depends on web services; availability depends on network connectivity; HTTP/HTTPS protocols solve limited security; HTTP alone is not possible due to HIS security requirements; these protocols are not considered here separately, they are used in Process Layer to increase security level

IV. Process to find Feasible Solutions (FFSP)

Two basic steps are involved in this semiautomatic process:

A. Derive consistency rules;

B. Construct FS

HIS-RA Ontology, is the main input to FFSP; interaction with AE is involved. All the consistency rules described in Table 1 are implemented by queries on the HIS-RA Ontology. Queries are written in Protegé DL query; however SPARQL,[4] the standard query language of the semantic Web, can be used as well. Each step is described in separate sub-sections in what follows, using an informal algorithmic notation.

A. Derive consistency rules

Input: HIS-RA Ontology

begin

1. ***Construction of the consistency rules***

Rules are of the form P => Q; examples are given to illustrate the procedure;

for each quality property, AE queries the ontology to look for components requiring/providing the property, in what follows; two alternative cases are considered:

case 1: only one component requires the quality property

query example:

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

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

where

P = b2-EHRMang and Q=XOR{b4,{b4,c2},{b4,c3}} or

Q = XOR{b4,c2,c3,{b4,c2},{b4,c3}}

Are boolean expressions reflecting that both <<*vp*>> or just one, namely, *b10-InteroperabilityEng={b4-MirthEng}* and / or *c15-HL7DM={c2-HL7Eng,c3-HXPeng} can be instantiated to provide Interoperability*,

case 2: several components require the quality property;

query example:

Components and requires_AvailPers some{true} => b1-Patient, b2-EHRMang, b3-ReportSystem, c1-DB

Components and provides_AvailPers some {true} => c7-Hibernate, c9a-Mirror, c9b MirrorReplic

[4] protegewiki.stanford.edu/wiki/SWRLTab

for each component requiring *AvailPers* in the above query, AE queries again the ontology, to determine exactly which variant providing *AvailPers* is connected to the component, as follows:

query example:
Variation-Points and connected_to some
{b1-Pat} and provides_AvailPers some
{true} => c7-Hibernate, c9a-Mirror, c9b-MirrorReplic
showing that component *b1-Patient* is connected to three variants providing *AvailPers*, namely *c7-Hibernate*, *c9a-Mirror*, and *c9b-MirrorReplic*; the query is repeated for b*2-EHRMang*, *b3-ReportSystem*, and *c1-DB*;
AE conforms P_i => Q_i, where P_1 = *b1-Patient*, P_2 = *b2-EHRMang*, P_3 = *b3-ReportSystem*, P_4 = *c1-DB* and Q_i is conformed as in case 1. AE groups those components in P_i having the same Q_i with an "AND" operator, taking into account that *<<vp>> c13-DataPers = c7-Hibernate, and c12-DataAvail = {c9a-Mirror, c9b MirrorReplic}*; two rules are obtained: b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}} and b1 AND b2 AND b3 AND c1 => XOR {c7, {c7, c9a}, {c7, c9b}};

end for;
end for;
Note: transitivity of the *connected_to* object property is used in this query;
end;
Output: *Consistency Rules in CT, column 3, table 1.*

B. Construct FS

Recall that FS are sets of RA components, and they should be connected, consistent and working; these properties will be verified by construction. Consistency is verified because FS are constructed applying consistency rules; the verification of the working property is two folded:
- verify functional compliance with FS components (CC and functional CR, see Section V.2), and – verify non functional compliance with quality properties required by FS components. Connectivity is verified by construction.

Input: *Consistency Rules from CT, HIS-RA Ontology, Customer Requirements (CR)*, see Section V.2;
begin
1. Construct Initial B set:
Conformed by CC and those functional components explicitly stated in CR; the following query is used to confirm that Initial B is connected, since it shows that

any Initial B component is directly connected to some other initial B component:

begin
for each component of Initial B, *saying a1-WebPage* ***do:***
InitialB and directly some {a1-WebPage}
=> a2-PatentPortal, a3-Reports, d1-Internet
shows that component a1 is directly connected to components a2, a3 and d1, all present in Initial B set.
end for:
end;

2. Construct B set:

Using the r.h.s. of the Consistency Rules, variants are being added to Initial B set to satisfy quality properties required by its component, as follows:

begin
for each Consistency Rule P => Q
if its r.h.s. Q holds only one variant
then
it is added to the initial B set
else
it must be checked
if some of those variants are CR
then
variants are added to the initial B set
end if;
end if;
end for;
Initial B set is updated as the new set B;
end;

B set is the "core" of components that will be present in all the FS derived from it; the presence of CC assures the functional aspect of the working property required by FS. Moreover, Initial B contains functionalities that require quality properties provided by some variant, implying that this variant is connected to Initial B, and since Initial B is connected then B is also connected.

3. Analyze B

begin
- AE analyzes all B components checking for the HIS-DQM quality properties

that can be missing, assuring the non functional aspect of the working property;
- since each consistency rule corresponds to a HIS-DQM quality property, see CT, table 1, let T1,T2 … Ts be sets expressing the multiple variants' alternatives of the r.h.s. of the consistency rule for missing quality properties;

end;

4. Generate alternatives

begin

The process continues with T_1,T_2 … Ts; the alternatives to derive FS from set B are achieved using the Δ operator, as follows:

$$\{FS(1), FS(2) \ldots FS(|T_1|\ldots|T_s|)\} = B \Delta T_1 \Delta T_2 \Delta \ldots \Delta T_s$$

where Δ is defined as follows:

Let X, Y sets, then $X \Delta Y = \{X \cup y : y \in Y\}$. *If* $X = \{FS(1), \ldots, FS(q)\}$ *then* $FS(1), \ldots, FS(q) \} \Delta Y = \{FS(i) \cup y : y \in Y, 1 \leq i \leq q\}$;

Δ is similar to a *"concat"* operator. For example, let us take the Interoperability consistency rule: *b2 => XOR {b4, {b4, c2}, {b4, c3}}*; the r.h.s has three alternatives; then the expression $B \Delta \{b4, \{b4, c2\}, \{b4, c3\}\}$ indicates that three new architectural configurations are derived from B, not necessarily FS, that are

$B \cup b4$, $B \cup \{b4,c2\}$, $B \cup \{b4,c3\}$.

Note: Δ "translates" the XOR expression used in the rules.

end;

end;

Output: *consistent*, *connected*, and *working FS*

V. Application of FFSP to a HIS case study

V.1 Applying FFSP step by step to derive FS for HIS concrete products according to customer requirements

The complete FFSP process will be now applied and validated on a case study considering explicit *customer requirements (CR)* for a concrete HIS product, described in Section V.2. More examples on rules' derivation and FS construction are included following the FFSP steps; components names in queries are abridged to shorten the presentation.

A. Derive Consistency Rules

1. Construction of the consistency rules

- Case 1: only one component requires a quality property; recall the *Interoperability* example in Section IV:

Components and requires_Interop some {true}
=> b2
Components and provides_Interop some {true}
=> b4, c2, c3

Interoperability is required by b2-HERMang and it is provided either by b4-MirthEng or by two alternative solutions *c2-HL7Eng*, *c3-HXPeng*; then two rules are derived: *b2 => XOR {b4, {b4, c2}, {b4, c3}}* and *b2 => XOR {b4, c2, c3, {b4, c2}, {b4, c3}}*. AE decides for the first rule, because *b4* assures the property which is mandatory for HIS, and can be combined with optional *c2* or *c3*, at database level, see figure 7.

- Case 2: more than one component require a quality property, for example *Integrity*; from queries:

Components and requires_Integrity some {true}
=> b1, b2, b3, c1
Components and provides_Integrity some {true}
=> c8, b7-HTTP, b7-HTTPS, HTTPS

four CC require *Integrity*, namely *b1-Patient*, *b2-HERMang*, *b3-ReportSystem*, and *c1-DB*; another query can be performed for each CC to determine exactly which one of the variants providing the property is connected to each CC;
from queries:

Variation-Points and connected_to some {b1-Pat} and provides_Integrity some {true}
=> c8, b7-HTTP, b7-HTTPS, HTTPS
Variation-Points and connected_to some {b2-EHRM} and provides_Integrity some {true}
=> c8, b7-HTTP, b7-HTTPS, HTTPS
Variation-Points and connected_to some {b3-RS} and provides_Integrity some {true}
=> c8, b7-HTTP, b7-HTTPS, HTTPS
Variation-Points and connected_to some {c1-db} and provides_Integrity some {true}
=> c8, b7-HTTP, b7-HTTPS, HTTPS

indicating that each CC can require the four solutions resulting from the query; hence we have rule *b1 AND b2 AND b3 AND c1 => XOR {{c8, b7-HTTP}, {c8, b7-HTTPS}, {c8, HTTPS}}*; moreover, *c8-IntegMech* is mandatory for *c1-db*, see figure 7; hence rule *b1 AND b2 AND b3 AND c1 => XOR {c8, {c8, b7-HTTP}, {c8, b7-HTTPS}, {c8, HTTPS}}* is also considered.

B. Construct FS.

1. Construct the Initial B set

Conformed by CC and functionalities explicitly required in CR, *a1-WebPage*, then we have:

Initial B = {a1, a2, a3, b1. b2, b3, c1, d1}

The following queries confirm that Initial B is connected:

InitialB and directly some {a1-WebPage} => a2, a3, d1

InitialB and directly some {b1-Pat} => c1, d1 ,..., etc.

However, Initial B is not an FS because consistency and working properties are still missing:

Connectivity: it is verified for Initial B and all sets derived from it.

Working: it is verified only for the functional part w.r.t. CC

Consistency: it will be verified in the next step

2. Construct the B set

There are multiple variants' alternatives required by Initial B functionalities to satisfy *Interoperability*, *Security* (*Authenticity*, *Confidentiality*, *Integrity*) and *AvailbilityPers*, and these properties are not CR (see Section V.2), they will be treated in step 3; AdaptabilityScal has also multiple alternatives *c4-NewMedStds* and *c5-JDBC*, which are explicit CR, hence they are included in B; *CorrectPrecision* required by *b1-Patient* and *b3-ReportSystem*, is satisfied by *b6-Algor* that is included for being the unique alternative; then we have:

B = Initial B = {a1, a2, a3, b1. b2, b3, b6, c1, c4, c5, d1}

Set B is connected and consistent, but it is not an FS because the working property is not complete:

Consistency: it is verified with rules in CT, table 1;

Working: it is functionally suitable but it is not compliant with the non functional part of the property, because quality properties of the HIS-DQM are still missing.

3. Analyze B

Missing quality properties are: *Interoperability*, *Authenticity*, *Confidentiality*, and *Integrity*

4. Generate alternatives

The Δ operator is used to satisfy *Interoperability* :

B Δ {b4, {b4, c2}, {b4, c3}} = {{a1, a2, a3, b1. b2, b3, b6, c1, c4, c5, d1} Δ {b4, {b4, c2}, {b4, c3}}} = {{a1, a2, a3, b1. b2, b3, b6, c1, c4, c5, d1, b4}, {a1, a2, a3, b1. b2, b3, b6, c1, c4, c5, d1, b4, c2}, {a1, a2, a3, b1. b2, b3, b6, c1, c4, c5, d1, b4, c3}} = { FS(1), FS(2), FS(3)};

These 3 new FS (numbered 1 to 3, and outlined in light grey in the first column

of Table 2) derived from B are not yet working FS, because *Authenticity* and *Confidentiality* are missing; components involved in each FS are outlined in dark grey in the rows ofTable 2, and the "core" of FS components can be appreciated; the same notation FS(i) is maintained for legibility.

Using again the *Δ* operator to have *Authenticity* and *Confidentiality* that are solved in Process Layer, and *Integrity*, which is implicitly solved in Transmission Layer with HTTP/HTTPS protocols, rule *b1 AND b2 AND b3 AND c1 => XOR {{c8, b7-HTTP}, {c8, b7-HTTPS}, {c8, HTTPS}}* is applied because *c8* is mandatory for *c1-db*, to obtain:

FS(4), FS(5), ..., FS(12) = {FS(1), FS(2), FS(3)} Δ {{c8, b7-HTTP}, {c8, b7-HTTPS}, {c8, HTTPS}}.

These new 9 FS (numbered 4 to 12, and outlined in light grey in the first column of Table 2, are still not working FS, because *AvailabilityPers* is still missing.
According to rule *b1 AND b2 AND b3 AND c1 => XOR {{c7, c9a}, {c7, c9b}}*, the next alternatives are considered to have *AvailabilityPers*:

FS(13), ... FS(30) = {FS(4), FS(5) ,..., FS(12)} □ *{{c7, c9a}, {c7, c9b}}*.

These last 18 FS (numbered 13 to 30, and outlined in grey in the first column of Table 2, are now connected, consistent and working; each one of them represent a feasible concrete product architectural configuration.
In total 30 architectural configurations have been found considering domain FR, NFR and CR. FFSP is being automatized by a support tool now under construction.

CR that have been used in FFSP to configure a concrete HIS product, are outlined in the next section.

V.2 Customer Requirements for a concrete product

CR must be "feasible", in the sense that they should be satisfied with the present SPL RA. CR that are not present in RA will not be considered in this work, however they could be included modifying RA.

Table 2. FS obtained applying FFSP

FS	Components																				
	Set B – Core of FS Components												FS Variants								
	a1	a2	a3	b1	b2	b3	b6	c1	c4	c5	d1	b4	c2	c3	c8	b7-HTTP	b7-HTTPS	HTTPS	c7	c9a	c9b
FS(1)	■	■	■	■	■	■	■	■	■	■	■	■									
FS(2)	■	■	■	■	■	■	■	■	■	■	■	■	■								
FS(3)	■	■	■	■	■	■	■	■	■	■	■	■		■							
FS(4)	■	■	■	■	■	■	■	■	■	■	■	■			■	■					
FS(5)	■	■	■	■	■	■	■	■	■	■	■	■			■		■				
FS(6)	■	■	■	■	■	■	■	■	■	■	■	■			■			■			
FS(7)	■	■	■	■	■	■	■	■	■	■	■	■	■		■	■					
FS(8)	■	■	■	■	■	■	■	■	■	■	■	■	■		■		■				
FS(9)	■	■	■	■	■	■	■	■	■	■	■	■	■		■			■			
FS(10)	■	■	■	■	■	■	■	■	■	■	■	■		■	■	■					
FS(11)	■	■	■	■	■	■	■	■	■	■	■	■		■	■		■				
FS(12)	■	■	■	■	■	■	■	■	■	■	■	■		■	■			■			
FS(13)	■	■	■	■	■	■	■	■	■	■	■	■			■	■			■	■	
FS(14)	■	■	■	■	■	■	■	■	■	■	■	■			■	■			■		■
FS(15)	■	■	■	■	■	■	■	■	■	■	■	■			■		■		■	■	
FS(16)	■	■	■	■	■	■	■	■	■	■	■	■			■		■		■		■
FS(17)	■	■	■	■	■	■	■	■	■	■	■	■			■			■	■	■	
FS(18)	■	■	■	■	■	■	■	■	■	■	■	■			■			■	■		■
FS(19)	■	■	■	■	■	■	■	■	■	■	■	■	■		■	■			■	■	
FS(20)	■	■	■	■	■	■	■	■	■	■	■	■	■		■	■			■		■
FS(21)	■	■	■	■	■	■	■	■	■	■	■	■	■		■		■		■	■	
FS(22)	■	■	■	■	■	■	■	■	■	■	■	■	■		■		■		■		■
FS(23)	■	■	■	■	■	■	■	■	■	■	■	■	■		■			■	■	■	
FS(24)	■	■	■	■	■	■	■	■	■	■	■	■	■		■			■	■		■
FS(25)	■	■	■	■	■	■	■	■	■	■	■	■		■	■	■			■	■	
FS(26)	■	■	■	■	■	■	■	■	■	■	■	■		■	■	■			■		■
FS(27)	■	■	■	■	■	■	■	■	■	■	■	■		■	■		■		■	■	
FS(28)	■	■	■	■	■	■	■	■	■	■	■	■		■	■		■		■		■
FS(29)	■	■	■	■	■	■	■	■	■	■	■	■		■	■			■	■	■	
FS(30)	■	■	■	■	■	■	■	■	■	■	■	■		■	■			■	■		■

In consequence, CR FR must be compliant with CC and each CR must be "binded" to the RA variant components [5] to satisfy required quality properties; inherent quality properties (they can change if software changes) are not necessarily

explicitly specified by customer in CR; they must be identified from CR and binded to each FR and NFR by the AE; assigned properties (they can change even if software does not change), such as *cost* (*low*, *medium*, *high*, *undetermined*) are also specified by the AE, for each variant.

Assigned software properties, are not considered quality properties [15], however they are involved in decision-making for product derivation. For this case study, the CR considered to derive a concrete HIS product are: *low cost HIS*, with basic functionalities, namely a *Web-based system* with a Web-page interface (running on a commercial browser), the facility to *handle patient data for scheduling appointments and capture demographic data, EHR management services* to handle and share EHR on-line, and the facility of *medical reports* including some *additional administrative features*. With respect to NFR, the facility for *medical data evolution*, such as the addition of new medical standards, is explicitly required, as well as *portability to a Java platform* for an Oracle database.

AE must study the feasibility of these explicit CR w.r.t. the present HIS-RA, and "translate" them in terms of HIS-RA ontology elements, i.e., *a1-WebPage, b1-Patient*, *b2*, *EHRMang*, *b3-Reports*, *c1-DB* (Oracle) with variants *c5-JDBC* API for portability to Java platform, and *c4-AddNewMedStds* for *AdaptabilityScal* to new medical standards. Quality properties required by FR, such as *Interoperability*, and mandatory domain NFR, such as *Security*, are obtained from the HIS-RA Ontology. The low-cost requirement is not used in this example, however it can be considered to evaluate the 18 FS obtained; an heuristic weight-based approach can be used to associate a weight to each FS, as objective to determine an optimal FS set, and study options with lowest cost FS [39].

VI. Related works

In general, the management of NFR variability in the SPL context to derive "convenient" product configurations meeting SPL and CR FR is not an easy task, as can be appreciated from the works discussed in this section. Ontologies have been widely used in different stages of SPL development approaches to capture domain knowledge and provide support to check consistence of the SPL models with their built-in reasoning engines; of particular interest for this work are the following topics:

- consistency of feature and variability modeling of NFR using constraints programming techniques and/or ontological approaches. A simple example of a feature model is shown in Figure 8, often found in the literature, where security is a NFR that is required by other FR, however in this context it is treated as another functionality; notice that if the credit card option, which

can be mandatory, is selected, a high security level must be assured, hence in a product configuration, the rule "CreditCard implies High" should be verified. It can be appreciated that a feature model has nothing to do with an architectural configuration in the sense of [3], as it was shown in Figure 3;
- domain quality modeling;
- derivation of architectural configurations for concrete products from the SPL RA.

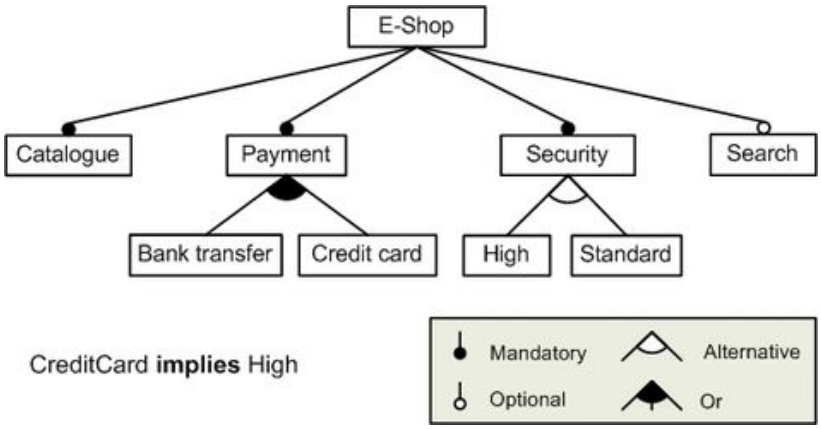

Fig. 8. Feature model representing an electronic shop [6].

VI.1 Consistency of feature and variability modeling of NFR using constraints programming techniques and/or ontological approaches

On these topics a lot of research work has been accomplished through a decade; a total of 24 papers have been reviewed, following a literature review inspired from [37].

a. Search: "software product line" + "configuration" + "constraint programming"

In this search, 9 interesting works were found, and 3 were selected, mainly dedicated to the configuration problem in feature models involving consistency checking, without considering the ontological approach:

Paper [26] deals with the problem of validating the consistency of feature models. It describes how to reason on feature models using constraint programming, providing three contributions to debugging feature model configurations: (1) a technique for transforming a flawed feature model configuration into a constraint satisfaction problem (CSP) and show how a constraint solver can derive the minimal set of feature selection changes to fix an invalid configuration, (2) it

is shown how this diagnosis CSP can automatically resolve conflicts between configuration participant decisions, and (3) experiment results that evaluate the technique are presented. They describe how to reason on feature models using constraint programming.

In [27] Collaborative Product Configuration (CPC) is focused. SPL configuration is in general carried out by a single person that is in charge of representing the interests of all stakeholders and managing decision conflicts on their own, conducing to errors and a time-consuming process. First, they treat CPC by describing and validating collaborative configuration scenarios. They discuss how collaborative configuration can be described in terms of workflows that safely guides stakeholders during the configuration process. Second, a preliminary set of reasoning algorithms tailored to the feature modeling domain to provide automated support for product configuration, is proposed. In addition, they compare empirically the performance of the proposed algorithms to that of a general-purpose solution.

In [28] constraint programming, and in particular Boolean constraint programming, has been used to support analysis of variability models such as FODA [6]. Constraint programming is used here also to specify product lines. The focus on variability, variation points or dependencies is switched to the concept of constraints that apply to variables. It is claimed that this approach is richer than the one based on dependencies. For instance, many constraints cannot be specified only with feature models dependencies (optionality, mandatory, etc., see Figure 8). The approach was implemented by a prototype tool, and its scalability explored with industry case studies. These experiments show that constraint programming encompasses existing SPL models such as FODA or the Orthogonal Variability Model (OVM) [4] and opens way to new possibilities such as reasoning simultaneously with different models during domain or application engineering.

However, a common point of these 3 works is that they do no handle explicitly NFR in the variability model which is included into or orthogonal to the feature model. However they are interesting and relevant works from a theoretical point of view, but not for NFR variability management.

b. Search: "software product line" + “configuration” + “ontology” + “web services“

In this search Web services were included to analyze also aspects of dynamic configuration; a total of 12 works were found interesting, and from these, 8 were selected for a more detailed discussion for their relevance to the topic:

The early work in [38] claimed that the lack of a formal semantics of feature models has hindered the development of this area. This paper presents a Semantic Web environment for modeling and verifying feature models using ontologies.

They use OWL DL (a decidable dialect of OWL) to precisely capture the relationships among features in feature diagrams and configurations. They use the ontology reasoning engines, like the OWL RACER, to check automatically for the inconsistencies of feature configurations. As part of the environment, they develop a CASE tool to facilitate the visual development, interchange and reasoning of feature diagrams represented as ontologies.

The work of Czarnecky et al. [8] explores the relation between feature models [6] and ontologies, such as OWL[5] (Ontology Web Language), an ontology representation language developed by W3C; basic feature models are thought of as a hierarchy plus a propositional formula; the notational spectrum is analyzed considering also UML to establish a boundary between ontologies and feature modeling; this synergy is considered promising for the use of reasoning-based support tools.

In paper [29] authors state that current mobile middleware is designed according to a 'one-size-fits-all' paradigm, which lacks the flexibility for customization and adaptation to different situations, and does not support user-centered application scenarios well. They describe an ongoing intelligent mobile middleware research project called PLIMM that focuses on user-centered application scenarios. PLIMM design is based on SPL ideas which make it possible for specialized customization and optimization for different purposes and hardware/software platforms. To enable intelligence, the middleware needs access to a range of context models. These contexts are modeled with OWL, focusing on user-centered concepts. The basic building block of PLIMM, where OWL context ontology logic reasoning is used. Their approach also addresses the handling of ontology evolutions resulting from the adaptation of ontology to changes and the consistent propagation of these changes to all related artifacts, using frame-based SPL configuration techniques.

Product configuration is discussed in [30] as a crucial mean to implement the mass customization paradigm claimed by SPL, assembling a set of customizable components to satisfy both customers' needs and technical constraints. With the aim of enabling efficient and effective development of product configurations by reusing configuration knowledge, an ontology-based approach to model product configuration knowledge is presented. The ontology-based product configuration models are hierarchically organized. At the lower level, a configuration meta-model is defined. Based on this meta-model, domain-specific configuration knowledge can be derived by reusing or inheriting the classes or relations in the meta-model. Configuration models are formalized using OWL. As a result, configuration models have well-defined semantics due to the logic semantics of OWL, making it possible to automatically detect inconsistencies of configuration knowledge bases.

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

Furthermore, configuration constraints are represented in SWRL, a rule language based on OWL. Finally, actual configuration processes are carried out using JESS, a rule engine for the Java platform, by mapping OWL-based configuration facts and SWRL-based configuration constraints into JESS facts and JESS rules, respectively.

The difficulty in designing SPL RA for service-based systems of ubiquitous computing, with highly dynamic evolutionary environment is discussed in [9]; 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 NFR, including also QoS values for product configuration, is proposed to increase flexibility and adaptability; a different ontology is used to specify the "device" (concrete product configuration features) matching NFR. 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 requested 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 [10] this subject is treated again, claiming 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 NFR 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 [11] 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 [21] as a domain ontology to model the SPL variability, including a model for NFR, the Quality Attribute (QA) profile. Each ontology in QA represent a quality property or attribute, for example security, including metrics. It has been provided with formal semantics by implementing a translation into a general-purpose knowledge representation language with formal semantics and inference support. The modeling concepts include components and features with compositional structure and attributes, the interfaces of components and connections between them, and constraints. The semantics of Kumbang is rigorously described using natural language and a UML profile. A prototype tool for solving variability has been implemented. The tool however is not interoperable and cannot transform automatically different models.

c. Conclusion on the topic: consistency of feature and variability modeling of NFR, constraints programming techniques and/or ontological approaches

- Feature models do not handle properly NFR and even less in modern dynamic Web services environments; an extensive survey on NFR variability modeling techniques is presented in [25].
- Ontological approaches help formalization and reasoning, and are used to verify and specify feature models; the OWL ontology language is generally used.
- None of the reviewed works consider modeling the RA with an ontology, nor a bottom-up approach.
- 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, which must be verified for consistency using different techniques based on classic problem solvers tools, or using ad-hoc or ontology built-in reasoning engines, to obtain runtime coded modules without considering RA, which is claimed to be the major SPL asset in most SPL development approaches; connections between modules have to be established according to their interfaces; this step becomes very complex without considering 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 architectural configurations (see Figures 3 and 8).

VI.2 Domain Quality Modeling using standards and ontological approach

Quality modeling refers to model the quality of a software product [15]. In the SPL context, this quality must be captured for the domain, where the SPL products portfolio is specified [5]; 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; in the context of our work, they 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 FR or NFR and its traceability is crucial to determine which RA component is requiring/providing the quality property, in order to check the global completeness and correctness of the derived product configuration. Another problem is the terminology used in the quality properties definitions, which are domain specific.

a. Search: "quality standards" + "ontology" + "web services"

Semantic Web Services are one of the most promising research directions to improve the integration of applications within and across enterprise boundaries, and now SPL are being designed also for these kind of interoperable, loosely coupled and distributed systems. The following works concern the capture of knowledge on Web Services (WS) domain and the handling of different quality standards, to unify quality terminology, using an ontological approach.

In this search 13 works were found interesting, and 5 were selected from these for a detailed discussion:

In [31] some of the main issues related to the semantic modeling of Web Services were outlined; it provides an overview of the Web Service Modeling Ontology (WSMO) - an ontology for Semantic Web Services. The design principles of this ontology are highlighted and a short description of the top-level elements is given. WSMO is presented as an ontology for semantically describing Semantic Web Services. Taking the Web Service Modeling Framework (WSMF) [35] as a starting point, WSMO refines and extends this framework, and develops a formal ontology and language. WSMF consists of four different main elements for describing semantic Web Services: (1) ontologies which provide the concepts and relationships used by other elements, (2) goals that define the users' objectives, i.e. the (potential) problems that should be solved by Web Services, (3) Web Services descriptions that define various aspects of a Web Service, and (4) mediators which bypass interoperability problems. With respect to quality modeling and standards, W3C standards were considered, but not other standards for QoS as in [20]; moreover, SPL are not mentioned.

An ontology for Semantic Web Services is proposed in [32] to enrich Web Services description. Distinguish from the existing ontologies [31], the proposed ontology is based on both functionalities and performances, and it is organized as a layered construction. The discovery related with the proposed ontology is also discussed. Based on the Service Oriented Architecture, the proposed ontology is helpful for requesters to find their suitable services according to their own preference (see [22]). Besides, as an example, an ontology for the learning resource is provided. However, quality modeling seems to be limited to performance.

In [33] Web Services are considered a new way of building software applications based on services that are available through the Internet. However, Web Services still face many problems that are limiting their adoption. One of the causes of this problem is the lack of metadata about the quality attributes of Web Services, which make Service Requesters reluctant to integrate Web Service with their applications. This paper proposes a novel ontology that describes a model of the requester-oriented Web Services' quality attributes. The ontology is based on previous quality models which have been refined and modified specifically to address the quality issues as they relate

to the requester of Web Services. Also an analysis will describe how some of the quality attributes in the previous model can be evaluated using different types of test cases.

Paper [36] outlines the importance of standard quality modeling, stating that growing Web Service usage, design and composition methods require precise and reliable information about Web Services quality. Such quality description has to be compatible with universal software quality models, so designers will be able to gather and decompose quality requirements addressing them to different Information Technology solution components, including used Web Services. This article proposes Web Services Quality Model based on the ISO/IEC Software product Quality Requirements and Evaluation (SQuaRE) model [15]. Prior to model definitions the authors present extended search on quality related issues in literature regarding Web Services and software in general. Authors refer to general need for measuring and publication of Web Service Quality measures considering limited trust and temporal character of measures. This article does not include technical solutions, but focuses on quality model showing its relevance to business needs. However, ontological approaches are not mentioned.

In [20] an ontology is proposed 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 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 also metrics. This work has been applied to WS discovery in [22]; in general, standards on software quality and the relationships established between them are not considered, yet these standards could be used as a shared understanding between service providers and customers, easing the WS discovery process. 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 to simplify expressions of WS discovery queries. Relationships between different standards are used to return WS even if they are described with quality properties defined by a different standard. Finally, NFR can be expressed as user preferences and are used to rank WS fulfilling FR during the discovery process.

b. Conclusion on the topic Domain Quality Modeling, standards and ontological approach

The use of standards is one of the major "best practices" claimed in Software Engineering to facilitate understanding among software development teams;

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

however software community frequently states that the use of standards reduce flexibility, thus avoiding widespread usage. With respect to quality standards, the terminology is different depending on the domain and Web services standards provide different definition according to the abstraction level they are describing, such as ISO/IEC 25010 for quality model, ISO/IEC 13236 for QoS general metrics, and W3C standards [20] [22]. Ontological approaches are found useful tools to unify standard terminology. However, our general opinion is that software community should pay more attention to this issue; software interoperability, for example is difficult to be achieved without the appropriate use of standards.

VI.3 Product Configuration from RA

Search: "software product line" + "configuration" + "reference architecture"

The search provided many results about SPL development methodologies, which include the RA design and configuration steps, but they were general frameworks that did not precise specific technics or models to direct configuration of products from RA; only 3 works were found really relevant to be discussed for our topic:

Paper [34] is an early work, previous to FODA [6], pointing out that there are few guidelines or methodologies available to develop and deploy SPL beyond existing domain engineering approaches, which is still happening today. They developed the PuLSE (Product Line Software Engineering) methodology to enable the conception and deployment of SPL within a large variety of enterprise contexts. It is centered on the incremental construction of the RA, where, besides FR, also quality issues are considered. The RA instantiation for specific products is performed in the PuLSE Usage phase, validating a single product against CR, by elaborating a product specification and a configuration model; implementation-specific decisions are collected that will have to be resolved during reference instantiation; these decisions and their possible resolutions are captured in the configuration model that extends a decision model. Driven by the configuration model and the product specification, the architecture for the instance is defined. The architecture is then validated against the product specification. The validated architecture is part of the product and is entered into the product configuration history. However, reasoning engines, problem solving or constrains technics used to validate the concrete instance, were not specified. PuLSE is the result of a bottom-up effort: the methodology captures the lessons learned from technology transfer activities with industrial customers.

Results on the generation of product architectural configurations directly from RA are presented in [23], following ontology-based feature modeling and MDD (Model Driven Development). It is claimed that the ontological approach to model features has more expressive power, it is shorter, more

formal 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 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 a reasoning engine is constructed to capture 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 and not RA, which is specified by an ADL; moreover, the concrete product configuration is obtained by translating the RA instance into another ADL using transformation rules, due to the MDD approach; - NFR are not mentioned, being this the main weakness of this approach. In our approach, RA is directly specified by the HIS-RA Ontology, containing explicit information on functional and non functional variation points, components which are solutions to NFR, constraints, mandatory/optional, etc. present also in feature models. The rules used to derive concrete architectural configurations are verified by the ontology, and clear traceability between components requiring/providing quality to satisfy FR and NFR is established using these rules. In consequence, all FR and NFR are satisfied in the concrete architectural configurations (FS) that are derived from RA, thus guaranteeing the overall quality of the concrete products of the SPL family. However, we were not able to compare the number of FS obtained by FFSP with results from other works, since similar approaches were not found; nevertheless, FFSP can be validated with different customer requirements using the present HIS-RA.

Finally, our recent work appears in the search performed [39], and it is the basis of the present work, as it was outlined in the Introduction section. It defines the product configuration and derivation process, called ASSPRO, directly from the RA, which has been built by a bottom-up process studying existing market products; the use of ontological approaches to derive and validate the consistency rules for products' configuration was planned as future work and main results of this ongoing research are shown in the present paper.

VII. Conclusion

A process to verify connected, consistent and working architectural configurations or FS has been presented and validated on a case study in the Healthcare Integrated Information System domain. HIS-RA Ontology has been developed to represent the HIS domain knowledge imbedded into the SPL HIS-RA; this ontology has been used to verify connectivity, consistency and working properties of the FS found. A more sophisticated reasoner, such as the SWRL plug-in for Protegé,[7] SWRLTab, could be used to produce more elegant queries, however the built-in DL Query engine satisfied the derivation of our consistency rules. The HIS application example may appear naïve, because HIS-RA has few components, however it shows the complete RA instantiation process, and it can be didactically useful to present this kind of development that in general is only partially shown the literature. The semiautomatic process to find convenient FS will be included as the first step of the Assessment Process (ASSPRO) [39], considering steps *Finding an optimal set of FS, Configuration of the Concrete Product Architecture,* and *Documentation of Optimal FS*, still under development. ASSPRO will offer a set of optimal FS to customer, computed considering a weight-based heuristic, and an assessment documentation to help deciding about the most convenient product to be derived from RA, compliant with customer requirements. A first draft of this process can be found in [24], and the last version in [39]. The approach presented for FFSP guarantees complete satisfaction of quality requirements in FS architectural configurations for the concrete product. This issue is not deeply considered in usual proactive or top-down SPL approaches, which are mostly feature model-based, and where in general NFR are given limited attention.

Aknowledgment

This research has been partially funded by the Consejo de Desarrollo Científico y Humanístico (CDCH), Universidad Central de Venezuela, DARGRAF (Diseño Arquitectónico Reactivo de software modelado por GRAFos) PG 03-8730-2013-2 project.

[7] protegewiki.stanford.edu/wiki/

References

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

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

[3] Shaw M., Garlan D. (1996). Software Architecture. Perspectives of an emerging discipline, Prentice-Hall.

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

[5] ISO/IEC NP 26550 (2013) Software and Systems Engineering – Ref. Model for Software and Systems PL. ISO/IEC JTC1/SC7 WG4.

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

[7] Matinlassi M. (2004). Comparison of software product line architecture design Methods: COPA, FAST, FORM, KobrA and QADA, ICSE'04.

[8] Czarnecki K., Hwan C., Kim P.,Trygve K. (2006) Feature Models are Views on Ontologies, SPIC 2006.

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

[10] Mohabbati B., Nima Kaviani N., Dragan Gašević D. (2009) Semantic Variability Modeling for Multi-staged Service Composition, Proceedings of the 13th Software Product Lines Conf., Vol 2., 2009.

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

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

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

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

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

[16] Elsner C. (2012). Automating Staged Product Derivation for Heterogeneous Multi–Product-Lines, Doctoral Thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany.

[17] Siegmund N., Rosenmuller M., Kuhlemann M., Kastner C., Apel S., Saake G. (2012). SPL Conqueror: Towords Optimization of Non-functional Properties in SPL, Soft. Qual. Jour., Vol. 20, No. 3-4, Sept, pp. 487-517(31).

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

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

[20] Losavio F., Matteo A., Levy N. (2009) Web Services Domain Knowledge with an Ontology on Software Quality Standards 3rd Int. Conf. on Internet Technologies and Applications (ITA'09), UK, pp.74-85, CAIR (Center for Applied Internet Research), Glyndwr University, 8-11 September.

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

[22] Jean S., Losavio F., Matteo A., Leyv N. (2010) An extension of Owl-S with Quality Standards, 4th Inter. Conf. on Research Challenges in Information Science (RCIS 2010), pp 483-494, IEEE (Print Version ISBN #978-1-4244-4840-1), Niza, France, May 10-21.

[23] Hector A. Duran-Limon, F. Castillo-Barrera E., Lopez-Herrejon R. (2011) 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.

[24] Losavio F., Ordaz O. (2015) Quality-Based Heuristic for Optimal Product Derivation in Software Product Lines, 6th Inter. Conf. On Internet Technology & Applications (ITA'15), pp. 113-129, Glyndwr, North Wales, U.K. 8-11 September.

[25] Esteller V., Losavio F., Matteo A., Ordaz O. Modelos de Variabilidad con Requisitos no Funcionales en un Contexto de Producción Industrial de Software, Revista Venezolana de Computación (ReVeCom), ISSN: 2244-7040, Vol. 1, No. 2, pp. 12-22, Diciembre 2014, Selección de los Mejores Artículos de CoNCISa 2014

[26] http://www.svc.net.ve/revecom

[27] White J., Schmidt D.C., Benavides D., ; Trinidad P. (2008) Automated Diagnosis of Product-Line Configuration Errors in Feature Models, Software Product Line Conference, 2008. SPLC '08. 12th International, pp. 225 – 234, 8-12 September.

[28] Mendonca M., Cowan D. (2010) Decision-making coordination and efficient reasoning techniques for feature-based configuration, Science of Computer Programming 75, pp. 311–332, Elsevier

[29] Salinesi C., Mazo R., Djebbi O., Dia D. (2011) Constraints: The core of product line engineering, Research Challenges in Information Science (RCIS), Fifth International Conference on, 19-21 May 2011, pages 1-10.

[30] Zhang W., Kunz T., Hansen K.M. (2007) Engineering Complex Computer Systems '07. 12th IEEE International Conference on, pp. 148 – 160

[31] Yang D., Dong M., Miao R. (2008) Development of a product configuration system with an ontology-based approach, Computer Aided Design, Volume 40, Issue 8, August, pp. 863–878, Elsevier

[32] WSMO Working Group, co-chair: Christoph Bussler, John Domingue, and Dieter Fensel, Web Service Modeling Ontology (WSMO) - An Ontology for

Semantic Web Services, Position paper at the W3C Workshop on Frameworks for Semantics in Web Services, June 9-10, 2005, Innsbruck, Austria

[33] Qiu Q., and Xiong Q. (2007) An Ontology for Semantic Web Services

[34] R. Perrott et al. (Eds.): HPCC, LNCS 4782, pp. 776–784, 2007. © Springer-Verlag Berlin Heidelberg

[35] Hanna S., and Alawneh A. (2010) An Approach of Web Service Quality Attributes Specification, IBIMA Publishing Communications of the IBIMA, Vol. 2010, Article ID 552843, 13 pages, http://www.ibimapublishing.com/journals/CIBIMA/cibima.html

[36] Bayer J., Flege O., Knauber P., Laqua R., Muthig D., Schmid K.,Widen T., DeBaud JM (1999) PuLSE: A Methodology to Develop Software Product Lines, Best Paper Award at the Symposium on Software Reusability‘99 (SSR’99), Los Angeles, May

[37] Fensel, D. Bussler, C. (2002) The Web Service Modeling Framework WSMF, Electronic Commerce Research and Applications, 1(2).

[38] Abramowicz W., Hofman R., Suryn W., Zyskowski D. (2008) SQuaRE based Web Services Quality Model, Proceedings of the International MultiConference of Engineers and Computer, Vol I IMECS 2008, 19-21 March, Hong Kong.

[39] Kitchenham, B y Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews In Software Engineering. Keele University and University of Durham, Technical report EBSE-2007-01.

[40] Wang H., Li YF, Sun J.,, Zhang H., Pan J. (2005) A Semantic Web Approach to Feature Modeling and Verification, Workshop on Semantic Web

[41] Losavio F., Ordaz O, Márquez H. (2015) Assessment for quality product derivation from a software product line reference architecture, Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS) 5(2), pp. 48-59.