IBIMA Publishing
Communications of the IBIMA
http://www.ibimapublishing.com/journals/CIBIMA/cibima.html
Vol. 2016 (2016), Article ID 790219, 18 pages
DOI: 10.5171/2016.790219

Research Article

A Methodology for the Integration of Service Repositories

Carlo Batini and Marco Comerio

Department of Informatics, Systems and Communication (DISCo) University of Milano-Bicocca
Milano, Italy

Correspondence should be addressed to: Carlo Batini; batini@disco.unimib.it

Received date: 6 February 2015; Accepted date: 30 April 2015; Published date: 29 January 2016

Abstract

In the service development lifecycle it is worthwhile to distinguish between a conceptual phase, whose goal is the modeling of abstract services, and an implementation phase, that produces concrete services. Both abstract and concrete services are usually specified through functional and non-functional properties and other meta-information in registries called *repositories of services*. However, several repositories of services may coexist in a single provider organization, each potentially characterized by heterogeneities and conflicting representations, as actually happens in database design for conceptual schemas. In this paper, we present (i) a model for service description in service repositories and (ii) a methodology for the integration of service repositories guided by the discovery of correspondences between services in different repositories to be integrated. Several paradigmatic examples are provided to show the characteristics of our methodology.

Keywords: methodology, service repository, repository integration.

Introduction

The service development lifecycle usually follows stages in a pipeline that may include exploration, planning, design, production, and management, where a service catalogue or repository is a recognized enabler of standardization, more effective service discovery and composition source of knowledge, better management of stakeholder expectations, cost transparency, and pricing mechanisms (Fisher 2008). Notwithstanding the multidisciplinary efforts carried out in Service Science research (Chesbrough & Spohrer 2006), the design and planning of services in digital service ecosystems (Khadka et al., 2011) still sees a focus on the technological perspective as the prevailing one. Nevertheless, the situation is evolving, and it is worth noting that some of the most recent methodologies for the so called Service oriented software engineering life cycle compared in (Gu & Lago 2011), consider two levels of services, that, according to the approach followed, are called respectively *business*/*technological*, or *abstract*/*concrete*. So, it is now consolidating a view in the service life cycle that distinguishes among *abstract services*, providing the template for concrete realization in a particular setting (Oberle et al., 2013); whereas production and management phases consider *concrete services* as actually realized and implemented in a particular setting, being them manual, semi-automated, or else fully automated.

To challenge the above issues, in the context of the Italian SMART (Services and Meta-Services for smart eGovernment) project (2011-2014), a methodology for the service lifecycle has been defined. A central artifact in the methodology is the *repository of (abstract/concrete) services*, which plays in the service lifecycle the role of a software applications repository in software engineering, and can be used, among others, for (i) aggregation of elementary services into composite services; (ii) identification of correspondences between services and events of life; (iii) assessment and improvement of the efficiency of the service lifecycle, and (iv) optimization of service value.

In order to make a repository of services an integrated representation of all services produced within an organization, we propose to model it with two different semantic relationships (namely, *is-a* and *part-of* relationships), originally proposed in the areas of knowledge representation and conceptual modeling, highlighting similarities among services and part-whole relationships.

Methodologies for the conceptual modeling of several types of artifacts have long been investigated in computer science. To fix ideas, we focus on the area of conceptual database design, seen in comparison to abstract service design; the similarities among the two areas stem from the fact that conceptual schemas and (in our proposal) repositories of abstract services have similar structures, since is- a and part-of relationships are the pillars of models adopted for both of them. In modern organizations there can be operating hundreds (and even thousands) of databases developed in different times and by different teams, resulting in an extremely fragmented view of the overall information content managed in the organization.

Database schema integration, a long term research initially proposed, among others, in (Batini et al., 1986) has the goal of producing a reconciled representation of a set of database schemas; this is achieved discovering correspondences among concepts in the schemas, solving structural conflicts, and applying transformations on schemas. It is our point that in a few years, as the distinction between abstract and concrete phases will become more and more a distinctive characteristic in service engineering, a similar situation will emerge.
The proliferation of abstract service repositories, in the same organization or in cooperating organizations, will require methodologies for the integration of service repositories. Such methodologies have also several interesting applications for service provider companies acquisitions and fusions; the assets that are objects of the acquisition/fusion procedure can be modeled in terms of integrated repositories of services.

The integration of service repositories is the issue discussed in the paper. We propose a model for the representation of abstract services and for the representation of relationships among abstract services in a repository.

Further we analyze the possible correspondences and conflicts among service properties, and we propose an integration methodology guided by the discovery of correspondences between services in different repositories. Several paradigmatic examples are provided to show the characteristics of our methodology.

The paper is organized as follows. We first analyze (Section 2) related work in the literature on repositories and integration methodologies. In Section 3 we shortly describe the service lifecycle defined in the

SMART project, highlighting similarities between service design and database design, while in Section 4 we provide the model defined in SMART to represent services and service repositories. Section 5 describes the proposed service repository integration methodology. Finally, Section 6 draws conclusion and discusses future developments.

2. Literature review

To the best of our knowledge, the issue of integration of abstract service repositories is new in the literature, and contributions specifically related to this area are lacking or address it only indirectly. We will see in the following that, probably due to the wide extent of service science, the issue of service repository integration has been so far perceived, although not yet precisely focused.

Service repositories store structured descriptions of services, so they can be considered as a specific type of registry or database. Accordingly, we point out that this research may benefit by considering the literature and experiences on schema integration and ontology integration. Taking these issues into account, in the following we first discuss the role of service repositories in different domains of computer science. Then, we consider different approaches to service, database and knowledge integration.

2.1 The role of service repository

Registries, catalogues, and repositories in Service Oriented Architectures (SOA) are considered both as a technological and as a managerial resource; the two perspectives are sometimes mixed. Furthermore the terms registry, repository and catalogue are used often with similar meanings.

As to the technological perspective, in (Sun Microsystems Inc. 2005) it is observed that a SOA registry-repository is increasingly becoming a middleware solution for managing SOA adoption in organizations. Different information artifacts may describe a service or relate it to other services in a repository, among them: (i) WSDL files describing interfaces and bindings; (ii) XML files describing the documents exchanged by messages; (iii) organizational policies and business rules.

Considering SOA standards, the Universal Description, Discovery, and integration (UDDI) registry presents severe limitations, since it can only store links to service information artifacts, being in such a way a non-integrated solution for linking service components as well. Indeed, integrated solutions face the challenge of managing information on inter-service dependencies in terms of relationships such as Contains, Extends, Implements, Supersedes, and Uses. In our repository of abstract services, dependencies have a conceptual nature, being the *is-a* and *part-of* relationships adopted in knowledge representation and conceptual modeling.

As to managerial issues, in (Fisher 2008) it is claimed that a catalogue of IT services is a useful vehicle for achieving standardization, consistency, cost transparency and service level agreement. A catalogue is considered as an important factor for showing what IT services provide to the organization; the catalogue contributes by proactively displaying the offerings.

The important service elements captured by a service catalogue include:
 service name, description, identifier, key users, key support requirements;
 ownership, authority, accounting, change impact, service-level information;
 customer expectations regarding performance, pricing, deliverables;
 target audience, service pre-requisites, service restrictions; instructions for service invocation.

In the approach of (Kohlborn et al., 2009) the service repository is the knowledge used by the service portfolio to support the decision-making process in regard to (i) the introduction of new services; (ii) the improvement or change of existing services; (iii) business decisions geared towards the bundling of multiple services into one package, and (iv) the marketing and commercialization of services. A service repository plays a relevant role also in the approach to service portfolio management discussed in (Janssen & Feenstra 2006), that defines service portfolio as an instrument guiding decision making about the development, reuse, execution, maintenance and evaluation of services.

The issue of repository integration emerges when we move to papers that adopt a wider approach, considering also managerial, organizational and economic perspectives. In the Manifesto centered on semantically enabled SOAs presented in (Brodie et al., 2005) authors claim that "while SOA is widely acknowledged for its potential to revolutionize the world of computing, that success depends on resolving two fundamental challenges that SOA does not address, integration, and search or mediation". Furthermore, SOA "requires that the services interoperate or integrate with respect to data, protocol, and process", and "integration, an open problem of conventional computing, hence one of its greatest costs, led to SOA but is not resolved by SOA".

As to integration seen from a managerial point of view "the demand for integration is enormous. In conventional computing, integration accounts for over 50% of application development budgets". In the same paper, when citing (White & Abrams 2005), it is indirectly introduced the concept of integration of (service) repositories, when it is said that integration "assures semantic persistence and consistency between multiple data repositories, independent of any application, service or user request".

Integrated service modeling is claimed in (Wimmer & Tambouris 2002) and (Tambouris 2001) to be the most critical and effective goal in e-government projects to achieve the "one-stop-shop" approach in interactions between Public Administration and citizens. The role of a service repository in platforms for one- stop-shop of e-Government services is discussed in (Wimmer 2002).

2.2 Service, data and knowledge integration

Integration has been investigated in various fields of computer science, as a way to combine, reconcile and make interoperable different types of artifacts related to one or more organizations, their activities and owned ICT technologies and infrastructures. The integration of services, data schemas, and software components is a key issue in all information systems where several levels of cooperation have to be established between different organizations or players (Kamal et al., 2009; Khoumbati & Themistocleous 2006).

An area that has inspired the research issues dealt with in the paper is data management, where integration has long been investigated as to schema integration (Batini & Lenzerini 1984), and more recently as to data integration (Lenzerini 2002); with reference to schema integration, the research moved from a clean definition of correspondences, conflicts and heterogeneity in data bases (Spaccapietra & Parent 1994; Spaccapietra et al., 1992) to a focus on semantic integration through ontologies (McGuinness D. 1998), (Wache et al., 2001). Four types of conflicts are considered e.g. in (Spaccapietra & Parent 1994): a. semantic conflicts, b. descriptive conflicts, c. heterogeneity conflicts, and d. structural conflicts. Integration of knowledge is discussed in several papers, e.g. (Bell et al., 1995). The alignment and integration of ontologies is investigated in (Wang & Gasser 2002), where information integration is enabled by having a precisely defined common terminology.

3. Service design lifecycle in Smart and database design: a common framework

In the context of the SMART project, a methodology for the service lifecycle has been defined. Such methodology consists of five phases:

1. *Planning*, in which the strategic long-term activities in the production and delivery of services are identified.
2. *Reconstruction of the abstract and concrete services as-is and of the related public and private processes,* with an evaluation of the level of service quality and value in use for the users.
3. *Design of the abstract services to-be and of the related public processes*, in which the architectural choices are taken.
4. *Production of the concrete services to-be and of the related private processes*, in which services and processes are realized.
5. *Management*, in which services are delivered, and the resources for the maintenance or renovation of service levels are provided.

While developing the methodology, we have been implicitly inspired by database design methodologies that were conceived in 80s and 90s, see Section 2 for a comprehensive reference. We may observe that (see Figure 1):

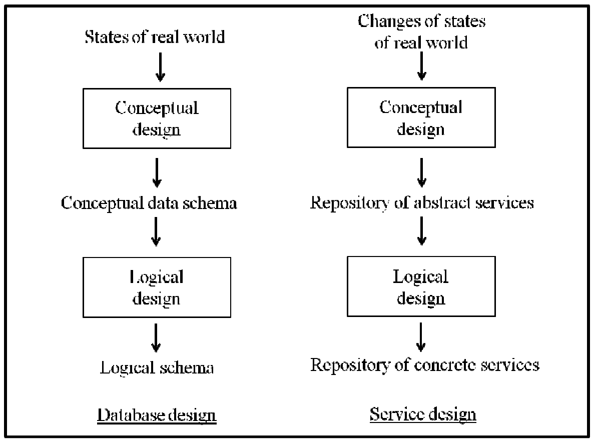


Figure 1: Database design and service design as related disciplines

1. A database is a representation of a set of states of the real world. Literature in databases distinguishes between an intension or *schema*, that represents a set of concepts and relationships between them, and an extension or *instance*, that represents evolving states of the real world in terms of values, whose corresponding classes are concepts and relationships in the schema.

2. The activity of database design clearly distinguishes between a conceptual step, whose goal is to conceive a schema represented in a conceptual model, e.g. the Entity Relationship model, and a logical step, whose goal is to translate the conceptual schema in a logical schema expressed in terms of a relational or XML model adopted in a database management system (ElMasri & Navathe 2011).

In service design we may identify a similar distinction; differences lie in the fact that services correspond to changes of states of real world, and the outputs of conceptual and logical design correspond to repositories of abstract and concrete services. In the next section we will describe the conceptual model adopted for abstract service repository creation.

4. A conceptual model for abstract service descriptions

In this section, we introduce the conceptual model adopted for abstract service description. First of all, we introduce the concept of service. Among the different definitions proposed in the literature, we adopt the definition of service in (Grönroos 1990) integrated with the perspective proposed in (Normann & Ramìrez 1993).

- ***Definition 1 (service)***: a *service* s consists in an activity or series of activities $\{a_1,...,a_n\}$ of more or less intangible nature, that take place in an exchange between a supplier and a customer, where the object of the transaction is an intangible good, so that both the supplier and the customer co-create and obtain value from the transaction. The exchange produces a *change of state* in the portion of real world of interest for the customer.

As in the design of databases also in the design of services it is worthwhile to distinguish two representation levels, corresponding to:

- ***Definition 2 (abstract service)***: an *abstract service* is a conceptual description of a service, expressed in a model independent form from their implementation. An example of abstract service is "Hotel Reservation", a conceptual description of the main characteristics that a service for reserving a room in a hotel must have. The abstract service cannot be invoked.

- ***Definition 3 (concrete service)***: a *concrete service* is provided by a specific provider and expressed in terms of a specific implementation. An example of concrete service is "Splendor Hotel Reservation", a description of the main characteristics of the service for reserving a room in the Splendor Hotel. The service can be invoked and the characteristics can be verified and monitored.

Abstract services, considered in the following, are characterized by the following *properties*: (i) a name; (ii) a set of functional properties; (iii) a set of non-functional properties; (iv) a data schema.

- ***Definition 4 (functional properties)***: *functional properties* $FP(s)=\{fp_1,...,fp_n\}$ of a service *s* define what the service does for the customer. Each fp_i in FP(s) enables a change of state of the real world, coherently with the goals expressed by the users in requirements collection. Considering e.g., a service for "Hotel Reservation", an associated functional property is "reserve a room"

In the following, functional properties will be defined through natural language descriptions, in such a way that they can be seen as an extension of the service names.

- ***Definition 5 (non-functional properties)***: *non-functional properties* $NFP(s)=\{nfp_1,...,nfp_m\}$ of a service *s* define how the service performs the change of state associated with the functional properties. Non-functional properties have been investigated by several authors, such as (O'Sullivan et al., 2005), (Becha & Amyot 2012), and (Zeng et al., 2004) and they concern technical qualities, business terms and legal aspects of the service. Following our example on the "Hotel Reservation", examples of NFPs are *price* and *payment options*. Let's note that, since we are considering an abstract service, the NFPs do not assume specific values.

- ***Definition 6 (data schema)*** The data schema describes at a conceptual level, adopting the Entity Relationship model, the portion of real world on which the service operates. An example of data schema for the "Hotel Reservation" service is reported in Figure 2.

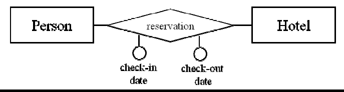
Service name	Hotel Reservation
Functional property	Reserve a room in a hotel for a given period
Non-functional properties	1. Price 2. Payment method
Data schema	 Person reservation Hotel check-in date check-out date

Figure 2: Example of abstract service description

Coming back to the parallel between database design and service design in Figure 1, in conceptual database design the schema is represented as a set of entities and relationships between them; it seems worthwhile in conceptual service design to represent abstract services, rather than as a set (of unrelated services), as a schema made of services and conceptual relationships among them. We focus in the following on two basic conceptual relationships adopted in conceptual modeling (Batini et al., 1991, Olivé 2007): the *part-of* and the *is-a* relationships.

 Definition 7 (part-of relationship): A *part-of relationship* holds between a service s_1 (the part) and a service s_2 (the whole) when s_1 is one of the components into which s_2 can be divided. The production of service $s_1$contributes in part to the production of service s_2.

Part-of relationships induce a distinction between two types of services: *elementary* and *composite* services. Basically, a service s_j is an elementary service if and only if it does not exist a service s_i with a part-of relationship with s_j. If the opposite holds, s_j is a composite service.

 Definition 8 (is-a relationship): An *is-a relationship* holds between a service s_1 (*child service*) and a service s_2 (*parent service*) when s_1 is a specialization of s_2. s_1 inherits all the properties of s_2 and s_1 has additional properties not owned by s_2.

We need also to adopt a graphical convention for representing a service *s* and the two sets of services that are in *part-of* and in *is-a* relationship with *s*. In Figure 3 we show the representation of the "change of address" service and two groups of services that are related with it through *is-a* and *part-of* hierarchies.

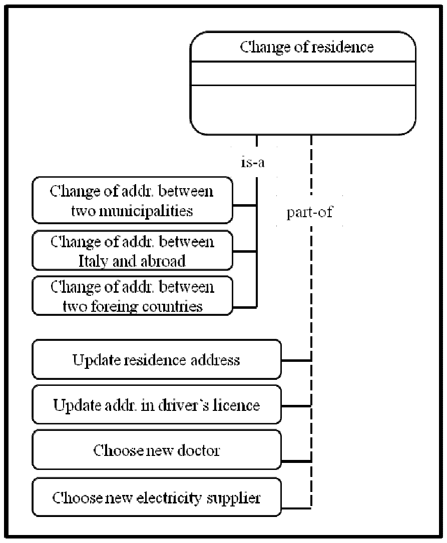

Change of residence
is-a
part-of
Change of addr. between two municipalities
Change of addr. between Italy and abroad
Change of addr. between two foreing countries
Update residence address
Update addr. in driver's licence
Choose new doctor
Choose new electricity supplier

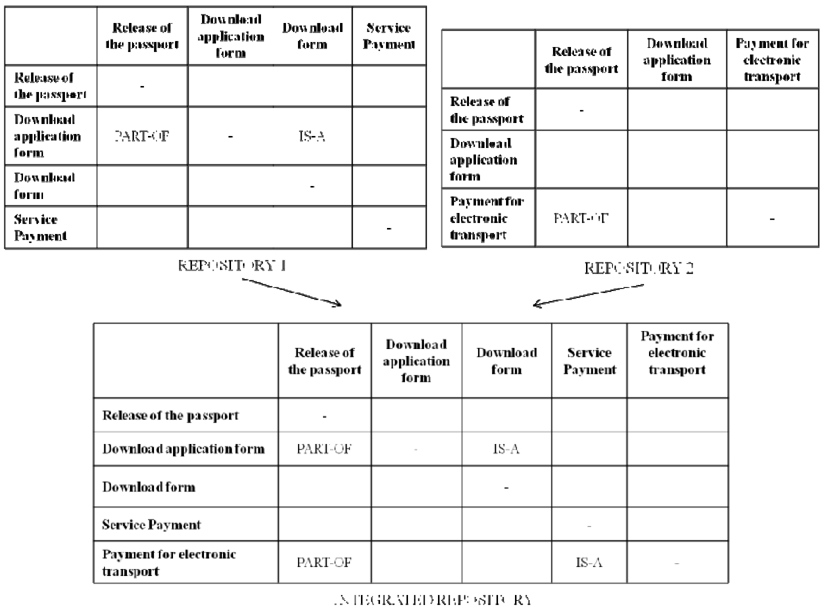
Figure 3: Examples of is-a and part-of relationships

We now have available all the machinery to introduce the concept of repository of (abstract) services.

- ***Definition 9 (repository of services)***: A repository of services is the set of services $s_1, s_2, \ldots, s_n$, together with the *part-of* and *is-a* relationships defined among them.

5. A methodology for service repositories integration

In this section, we describe a methodology for the integration of service repositories. Basically, the methodology supports the process whose pictorial representation is exemplified in Figure 4.



	Release of the passport	Download application form	Download form	Service Payment
Release of the passport	-			
Download application form	PART-OF	-	IS-A	
Download form			-	
Service Payment				-

REPOSITORY 1

	Release of the passport	Download application form	Payment for electronic transport
Release of the passport	-		
Download application form			
Payment for electronic transport	PART-OF		-

REPOSITORY 2

	Release of the passport	Download application form	Download form	Service Payment	Payment for electronic transport
Release of the passport	-				
Download application form	PART-OF	-	IS-A		
Download form			-		
Service Payment				-	
Payment for electronic transport	PART-OF			IS-A	-

INTEGRATED REPOSITORY

Figure 4: An example of service repository integration

The methodology for service repository integration has in input two service repositories R_1 and R_2 and produces in output an integrated service repository IR. It is organized in three phases: (i) pre-integration, (ii) correspondence elicitation and analysis, and (iii) integration. We now describe in detail the three phases.

5.1 Pre-integration

In this phase we carry out a normalization activity on the properties of services of the two repositories, in such a way to homogenize as far as possible service descriptions, and consequently make easier subsequent correspondence elicitation and analysis.

The normalization may involve complex functional properties that make implicit the presence of lower level services. We make an example to illustrate this point. The service "Localization of points of interest" has the functional property "Identify a museum OR identify a park OR identify a historic mansion OR identify a historic monument that can be of interest for the user" meaning that the localization service spans over the four places mentioned in the property. It is worthwhile to transform the service creating four new services referring each one to a specific type of point of interest and associating an is-a relationship between the service itself and the four new services. In this way, we favour the identification of correspondences with services in the second repository.

After the normalization of functional properties, we proceed with the normalization of non-functional properties. We assume to adopt a vocabulary of non-functional properties, where each property is defined with a name, possible values and dependencies with other properties. The normalization consists in checking the conformity between the description in the repository and the adopted vocabulary. Let us consider the following example: a service is characterized by the non-functional property *service payment*. This property does not conform to the adopted vocabulary, where three different properties (i.e., *absolute price*, *tax on price*, *payment method*) are proposed to express the *service payment* property. Therefore, the property must be substituted with the three mentioned properties.

5.2 Correspondence elicitation and analysis

In the correspondence elicitation and analysis phase, services of two repositories R_1 and R_2 are analyzed in order to identify the presence of correspondences among them. Correspondences among two services s_1 in R_1 and s_2 in R_2 relate the whole services s_1 and s_2 and their properties (names, functional properties, non-functional properties and data schemas), putting in evidence semantic relationships between them. In this paper, we focus on four different types of correspondences between services s_1 in R_1 and s_2 in R_2:

1. *Identity*, when s_1 and s_2 correspond to the same service. An example of identity is the case of two "change of address" services with identical properties.
2. *Is-a*, when s_2 is a specialization of s_1 e.g., the service "change of address between two foreign countries" is a specialization of the service "change of address". This type of correspondence leads to the definition of an is-a relationship in the integrated repository.
3. *Part-of*, when s_1 and s_2 are in a part-whole relationship, e.g., the service "payment for electronic passport" is a part of the composite service "release of passport". This type of correspondence results in a new part-of relationship in the integrated repository.
4. *Relatedness*, when s_1 and s_2 are recognized as different services sharing similar properties and, as a consequence, they are potential siblings in an is-a hierarchy. An example of relatedness is the case of services "authorization to open a cafeteria" and "authorization to open a restaurant".

5. *Unrelatedness,* when s1 and s2 are different services and no semantic relationship is defined between them.

The elicitation of correspondences between services *s1* in R1 and *s2* in R2 is guided by the analysis of their properties. In this paper, we propose to perform first the local analysis of properties and then the global analysis of the two services as a whole.

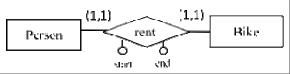
Figure 5 shows possible results of the local analysis of each type of property. Since in the reference model of Figure 2 names and functional properties consist in descriptions in natural language, their local analysis determines if the descriptions are identical, similar or unrelated. The local analysis of non-functional properties, defined as a set of terms taken from a reference vocabulary, determines if the sets are equal, disjoint or partially overlapping. The local analysis of data schemas is such that: (i) names of entities, relationships and attributes are compared in order to identify semantic relationships between them; (ii) types of concepts are compared to identify similarities and dissimilarities.

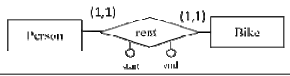
The local analysis of schemas is supported by the usage of WordNet (http://wordnet.princeton.edu/), a large lexical database of English terms, where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (*synsets*), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations (i.e., hyponym, hypernym).

Names and functional properties
 Identical
 Similar
 Unrelated
Non functional properties (sets of)
 Same set
 Inclusion
 Overlap
 Disjunction
Data schemas
1. Names of concepts (Entity, Relationship, Attribute)
 Identity
 Synonymy
 Hyponym-Hypernym
 Unrelated
2. Types of concepts
 Same concept, same types
 Same concept, different types (e.g., Attribute *vs* Entity)
 Different cardinalities between two Relationships representing the same concept

Figure 5: Correspondences between properties of services

The results of local analysis steps must be jointly evaluated in the global analysis to identify which type of correspondence exists between the two services. Let consider the example in Figure 6. Services are identical in terms of names, functional properties and data schemas but their non-functional properties are different (the set of the second service is included in the set of the first one). In particular, the difference is on properties related to service payment. The global analysis of the service properties leads to the conclusion that services are different. The correspondence of the two services is of "relatedness", since they are two specializations (i.e., "rent a free bike" and "rent a charged bike") of the same service "rent a bike".

Service name	Rent a bike
Functional property	Rent a bike for a given period
Non-functional properties	1. Absolute Price, 2. Payment Method, 3. Physical location, 4. Service Accessibility, 5. Weekly Service Availability
Data schema	

Service name	Rent a bike
Functional property	Rent a bike for a given period
Non-functional properties	1. Physical location, 2. Service Accessibility, 3. Weekly Service Availability
Data schema	

Property	Local Analysis	Global Analysis
Names + Functional properties	*Identical*	**RELATEDNESS** The two services are different. The first service should be renamed "Rent a free bike ". The second one should be renamed "Rent a charged bike".
Non-functional properties	*Inclusion*	
Data schemas	*Identical*	

Figure 6: An example of local and global analysis of service properties

The correspondence elicitation and analysis phase produces a matrix M where services of R_1 are in the columns, services of R_2 are in the rows and the intersections are filled with the identified correspondences.

5.3 Integration

This phase conducts to the creation of the integrated repository IR. It is performed with a two-steps process. First, a coarse-grained integration is performed considering only the is-a and part-of hierarchies of R_1 and R_2.. Then, a fine-grained integration is performed to add service properties. In the following we describe the two steps in terms of goals, inputs and activities.

<u>1. Coarse-grained Integration</u>
Goal: integrate R_1 and R_2 producing the integrated repository IR.
Input:
 is-a and part-of hierarchies of R_1 and R_2 (see, as an example, Figure 3);
 correspondence matrix M produced in the previous phase

Activities:

1. For each "*identity* correspondence" in M between s_i and s_j
 Merge s_i and s_j;
 Delete row of s_j from M.

2. For each "*is-a* correspondence" in M between s_i and s_j
 Add a new is-a relationship in IR between s_i and s_j;
 (if applicable) push services up in the new is-a hierarchy.
3. For each "*part-of* correspondence" in M between s_i and s_j
 Add a new part-of relationship in IR between s_i and s_j.
4. For each "*relatedness* correspondence" in M between s_i and s_j
 (if necessary) rename s_i and/or s_j;
 Add a new service s_x in IR;
 Add a new is-a relationship in IR between s_i and s_x and between s_j and s_x.

<u>2. Fine-grained Integration</u>
Goal: add service properties in the integrated repository IR.

Input:
 properties of services in R_1 and R_2;
 correspondence matrix M produced in the previous phase;
 integrated repository IR.

Activity:

For each identified correspondence in M between s_i and s_j and for each service property of the reference model,
 identify the rules to be adopted to merge the properties in IR considering the results of the local analysis. Figure 7 shows

examples of rules for integrating properties of services for the identity correspondence;

 apply the rules.

correspondence	**property**	**local analysis**	**rule for merging s_i and s_j in IR**
IDENTITY	*name + functional properties*	identical	keep the same name and FPs
		similar	choose the most expressive name
		unrelated	define a new name and FPs
	non functional properties	the same set	keep the same
		inclusion	keep the largest set
		overlap	merge the two sets
		disjunction	merge the two sets
	data schema (names of concepts)	identity	keep the same
		synonymy	choose the most expressive name
		hyponym-hypernym	keep the hyponym
	data schema (types of concepts)	same concept, same	keep the current types of concepts
		same concept, different type	choose the most expressive type
		different cardinalities	choose the most restrictive cardinality

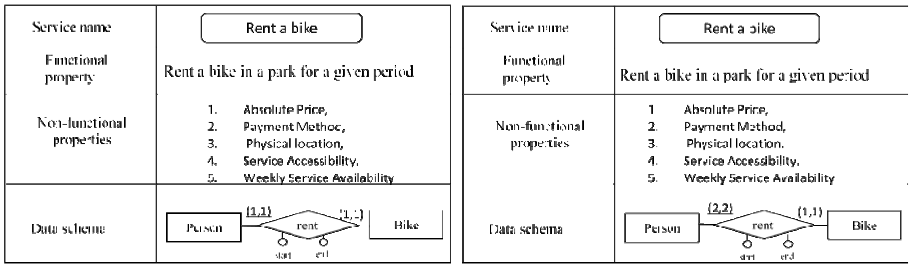
Figure 7: rules for integrating properties of services with an identity correspondence

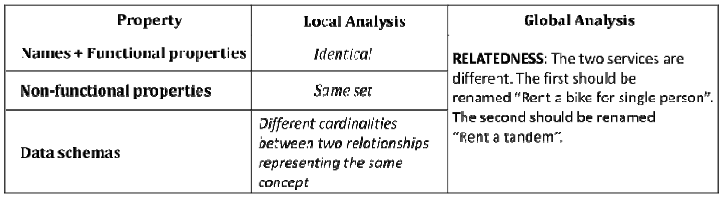
5.4 Examples of repository integration

An exhaustive discussion of all procedures to solve all cases of correspondences in the integration process is outside the scope of the paper. In the following, we show several paradigmatic examples.

Example 1 – An heterogeneity in the data schemas drives the integration process.

In Figure 8 we have the case of two services that at a first glance could be considered identical. Looking in more detail at the two data schemas, we see that the two relationships in the data schemas, while representing the same concept have different cardinalities. This difference leads us to the conclusion that the two services are different although having the same name. They produce a different change of state in the portion of real world of interest for the customer: the first one produces the rent of a bike for a single person, the second one the rent of a tandem for two persons. So, in the global analysis we identify a correspondence of type “relatedness”, since the two services are potential siblings in an is-a hierarchy.





Property	Local Analysis	Global Analysis
Names + Functional properties	*Identical*	**RELATEDNESS**: The two services are different. The first should be renamed "Rent a bike for single person". The second should be renamed "Rent a tandem".
Non-functional properties	*Same set*	
Data schemas	*Different cardinalities between two relationships representing the same concept*	

Figure 8: An example of correspondence elicitation driven by an heterogeneity in the data schema

Figure 9 shows the result of the integration process of the two services in Figure 8. According to the proposed coarse-grained integration, the identified relatedness correspondence leads to: (i) rename the two "rent a bike" services in "rent a bike for single person" and "rent a tandem", (ii) create a new service "rent a bike", (iii) create two is-a relationships between the new service and the two renamed services.

The rule used for fine-grained integration consists in pushing-up common service properties in the is-a hierarchy. This rule focuses on the inheritance property of is-a relationships, i.e., properties of the *parent service* are inherited by the *child service*. Since functional and non-functional properties of "rent a bike for single person" and "rent a tandem" are identical, they are attributed to their parent (rent a bike) in the is-a hierarchy.

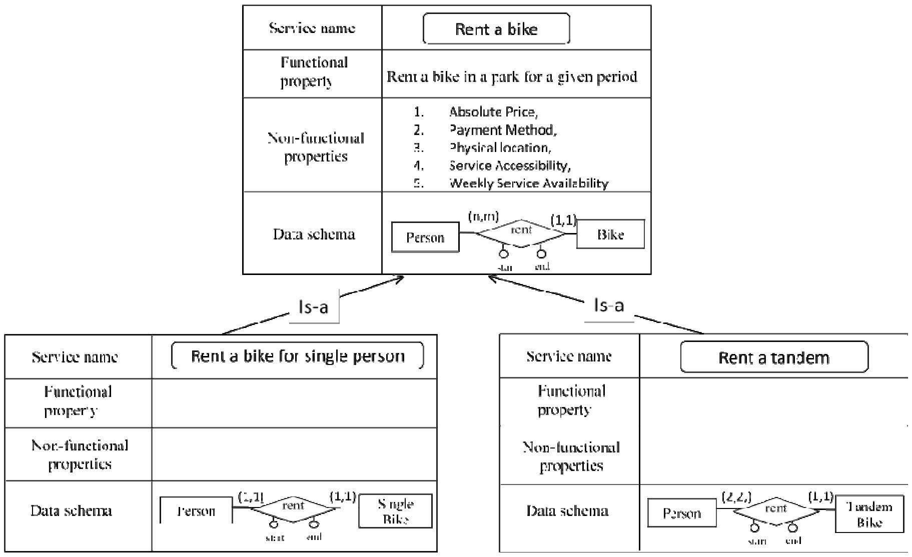
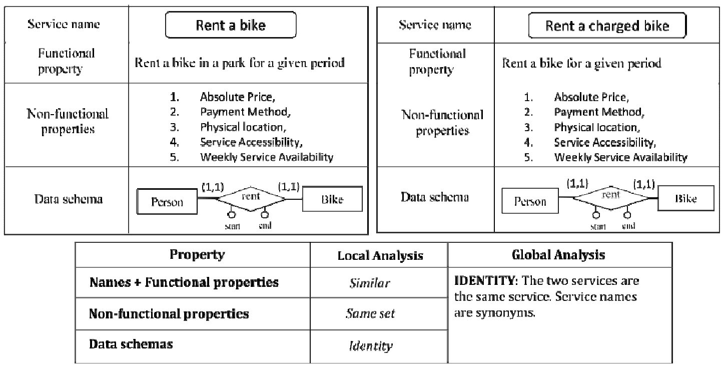


Figure 9: The result of the integration process of the two services in Figure 8.

Example 2 - A discovered synonymy guides the integration process.

In Figure 10 we have the case of two services that at a first glance appear similar but not identical. The local analysis of non-functional properties and data schemas reveals that the two services produce the same change of state in the portion of real world of interest for the customer. Basically, there is a correspondence of type “identity” between them. In the coarse-grained integration, the two services are merged. Then, for the fine-grained integration rules in Figure 7 (a. “keep the most expressive name and FP”, b. “keep the same (set)”, c. “keep the same (names)” and d. “keep the current types of concepts”) are used.



Service name	Rent a bike
Functional property	Rent a bike in a park for a given period
Non-functional properties	1. Absolute Price, 2. Payment Method, 3. Physical location, 4. Service Accessibility, 5. Weekly Service Availability
Data schema	Person (1,1) rent (1,1) Bike; start, end

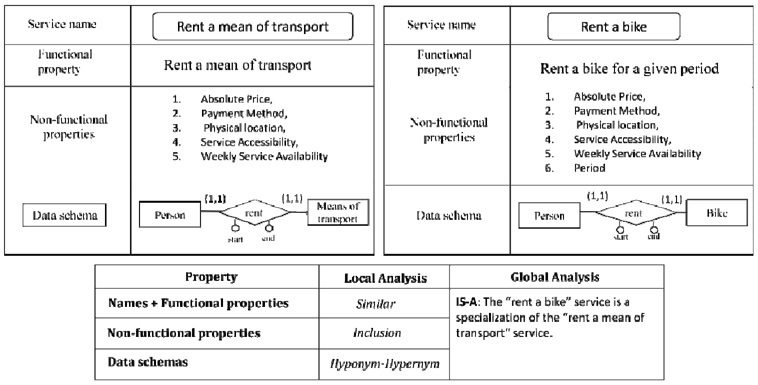
Service name	Rent a charged bike
Functional property	Rent a bike for a given period
Non-functional properties	1. Absolute Price, 2. Payment Method, 3. Physical location, 4. Service Accessibility, 5. Weekly Service Availability
Data schema	Person (1,1) rent (1,1) Bike; start, end

Property	Local Analysis	Global Analysis
Names + Functional properties	*Similar*	**IDENTITY:** The two services are the same service. Service names are synonyms.
Non-functional properties	*Same set*	
Data schemas	*Identity*	

Figure 10: An example of correspondence elicitation driven by the analysis of the full-set of properties.

Example 3 – Joint analysis of non functional properties and data schemas leads to the discovery of an is-a correspondence.

The local analysis of names and functional properties reveals that the services in Figure 11 are similar. The local analysis of non-functional properties and data schemas provides additional information for the identification of correspondences between the two services: (i) the set of non-functional properties of “rent a mean of transport” is included into the set of “rent a bike”, and (ii) the term “mean of transport” is an hypernym of “bike”. The results of the local analyses lead to the identification of correspondence of type “is-a” between the two services.



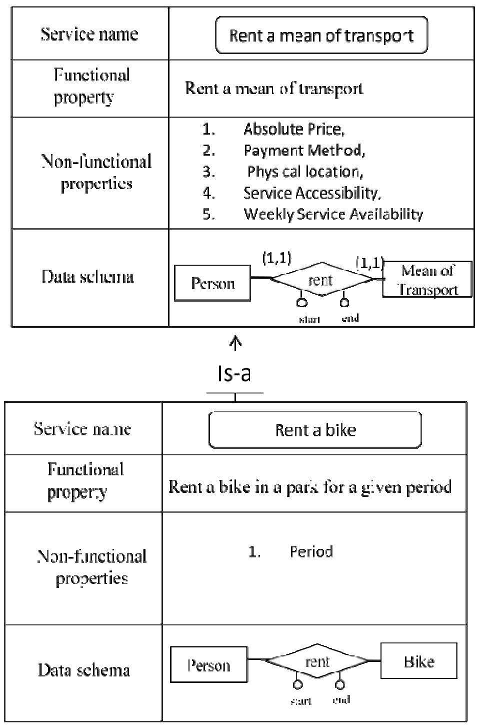
Service name	Rent a mean of transport
Functional property	Rent a mean of transport
Non-functional properties	1. Absolute Price, 2. Payment Method, 3. Physical location, 4. Service Accessibility, 5. Weekly Service Availability
Data schema	Person (1,1) rent (1,1) Means of transport; start, end

Service name	Rent a bike
Functional property	Rent a bike for a given period
Non-functional properties	1. Absolute Price, 2. Payment Method, 3. Physical location, 4. Service Accessibility, 5. Weekly Service Availability 6. Period
Data schema	Person (1,1) rent (1,1) Bike; start, end

Property	Local Analysis	Global Analysis
Names + Functional properties	*Similar*	**IS-A**: The “rent a bike” service is a specialization of the “rent a mean of transport” service.
Non-functional properties	*Inclusion*	
Data schemas	*Hyponym-Hypernym*	

Figure 11: An example of correspondence elicitation driven by the analysis of non-functional properties and data schemas.

In Figure 12 we show the result of the integration process of the two services in Figure 11. Thecoarse-grained integration leads to the creation of an is-a relationship between the two services. The rule used for fine-grained integration consists in pushing-up common service properties in the is-a hierarchy. Since the two services share four non-functional properties, such properties are attributed to the parent service ("rent a mean of transport") in the is-a hierarchy.



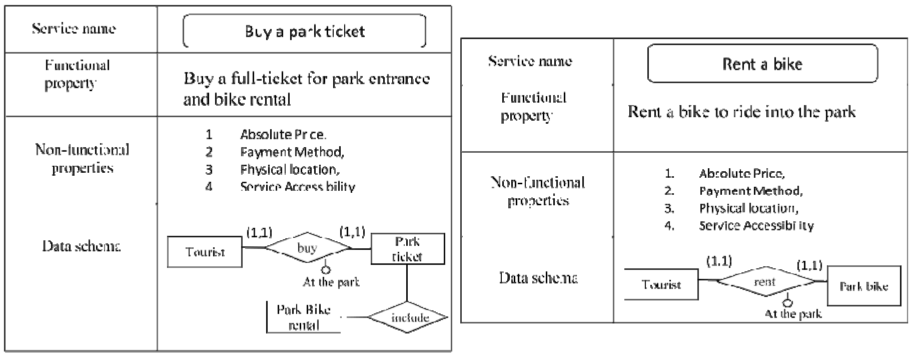
Service name	Rent a mean of transport
Functional property	Rent a mean of transport
Non-functional properties	1. Absolute Price, 2. Payment Method, 3. Physical location, 4. Service Accessibility, 5. Weekly Service Availability
Data schema	Person (1,1) rent (1,1) Mean of Transport; start end

Service name	Rent a bike
Functional property	Rent a bike in a park for a given period
Non-functional properties	1. Period
Data schema	Person rent Bike; start end

Figure 12: The result of the integration process of the two services in Figure 11.

Example 4 – Joint analysis of functional properties and data schemas leads to the identification of a part-of correspondence between two services.

In Figure 13 we have the case of two services that at a first glance appear unrelated. The local analysis of functional properties reveals that the "buy a park ticket" service offers two functionalities (i) buy a full-ticket for park entrance, and (ii) rent a bike. The latter coincides with the functionality offered by the "rent a bike" service. This local analysis suggests investigating if a correspondence of type "part-of" exists between the two services. This assumption is confirmed by the local analysis of data schemas since (i) some identities between names are present and (ii) the schema associated to the service "rent a bike" is a view of the schema associated to the service "buy a park ticket". In the integrated repository a new part-of relationship between services is defined.



Property	Local Analysis	Global Analysis
Names + Functional properties	*Similar*	**PART-OF**: The "rent a bike" service is a part-of the "buy a park ticket" service.
Non-functional properties	*Same set*	
Data schemas	*Identities between names*	

Figure 13: An example of correspondence elicitation driven by the analysis of functional properties and data schemas.

Conclusions

A service repository is actually an information base, containing reconciled and structured information on the whole set of services provided by a given organization. The proliferation of service repositories in the same organization or in cooperating organizations requires focusing the attention on a methodology for the integration of service repositories. In this paper, we presented a model for service description in service repositories and a methodology for the integration of services repositories guided by the discovery of correspondences between services in repositories to be integrated. Several paradigmatic examples have shown the feasibility of the methodology.
Future work deals with (i) the design and implementation of a tool to support the adoption of the proposed methodology for service repository integration and, (ii) the experimentation on real case studies.

Acknowledgment

The work presented in this paper has been partially supported by the Italian project PON01_00861 SMART (Services and Meta-services for smART eGovernment) and by the project (CUP E41I13000220009) SPAC3 (Smart services of the new Public Administration for the Citizen-Centricity in the Cloud) co-financed by the Lombardy region

References

1. Batini, C., Ceri, S. & Navathe, S.B., 1991. *Conceptual database design: an Entity-relationship approach*, Benjamin-Cummings Publishing Co., Inc. Redwood City, CA, USA.

2. Batini, C. & Lenzerini, M., 1984. A Methodology for Data Schema Integration in the Entity Relationship Model. *Software Engineering, IEEE Transactions on*, SE-10, pp.650–664 ST – A Methodology for Data Schema Integ.

3. Batini, C., Lenzerini, M. & Navathe, S.B., 1986. Comparison of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4), pp.323–364.

4. Becha, H. & Amyot, D., 2012. Non-Functional Properties in Service Oriented Architecture – A Consumer's Perspective. *Journal of Software*, 7(3). Available at: http://ojs.academypublisher.com/index.php/jsw/article/view/jsw0703575587.

5. Bell, P., Davis, E.A. & Linn, M.C., 1995. The knowledge integration environment: theory and design. In *The first international conference on Computer support for collaborative learning*. CSCL '95. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., pp. 14–21. Available at: http://dx.doi.org/10.3115/222020.222043.

6. Brodie, M. et al., 2005. *Semantically Enabled Service-Oriented Architectures: A Manifesto and a Paradigm Shift in Computer Science*, Insbruck.

7. Chesbrough, H. & Spohrer, J., 2006. A research manifesto for services science. *Communications of the ACM*, 49(7), pp.35–40.

8. ElMasri, R. & Navathe, S., 2011. *Fundamentals of Database Systems* 6th ed., Addison Wesley.

9. Fisher, C., 2008. Opportunity-driven IT service management. *Journal of Digital Asset Management*, 4(6), pp.377–394.

10. Grönroos, C., 1990. *Service Management and Marketing. Managing the Moments of Truth in Service Competition*, Lexington Books.

11. Gu, Q. & Lago, P., 2011. Guiding the selection of service oriented software engineering methodologies. In *SOCA (2011) Vol 5*. pp. 203–223.

12. Janssen, M. & Feenstra, R., 2006. From Application to Service Portfolio Management: Concepts and Practice. In *ECEG*. Marburg, Germany, pp.225–234.

13. Kamal, M.M., Themistocleous, M. & Khosrow-Pour, M., 2009. Investigating Enterprise Application Integration Adoption in the Local Government Authorities. In C. G. Reddick, ed. *Handbook of Research on Strategies for Local E-Government Adoption and Implementation: Comparative Studies*. Hershey, PA; London, UK: Information Science Reference, pp. 661–686.

14. Khadka, R. et al., 2011. An Evaluation Of Service Frameworks For The Management Of Service Ecosystems. In *PACIS 2011 Proceedings*. Available at: http://aisel.aisnet.org/pacis2011/93.

15. Khoumbati, K. & Themistocleous, M., 2006. Integrating the IT Infrastructures in Healthcare Organisations: A Proposition of Influential Factors. *Electronic Journal of e-Government*, 4(1), pp.27–36.

16. Kohlborn, T., Korthaus, A. & Rosemann, M., 2009. Business and Software Service Lifecycle Management. In *13th International Enterprise Distributed Object Computing Conference (EDOC)*. pp. 87–96.

17. Lenzerini, M., 2002. Data integration: a theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '02. New York, NY, USA: ACM, pp. 233–246. Available at: http://doi.acm.org/10.1145/543613.543644.

18. McGuinness D., L., 1998. Ontological Issues for Knowledge-Enhanced Search. In *Frontiers in Artificial Intelligence and Applications,*. IOS Press, Washington, DC.

19. Normann, R. & Ramìrez, R., 1993. From Value Chain to Value Constellation: Designing Interactive Strategy. *Harvard Business Review*.

20. O'Sullivan, J., D., E. & ter Hofstede, A.H.M., 2005. *Formal description of non functional service properties-Technical Report*, Brisbane.

21. Oberle, D. et al., 2013. A unified description language for human to automated services. *Information Systems*, 38(1), pp.155–181.

22. Olivé, A., 2007. *Conceptual Modeling of Information Systems*, Berlin- Heidelberg: Springer Verlag.

23. Spaccapietra, S. & Parent, C., 1994. View integration: a step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 6(2), pp.258–274.

24. Spaccapietra, S., Parent, C. & Dupont, Y., 1992. Model Independent Assertions for Integration of Heterogeneous Schemas. *Vldb Journal*, 1(1), pp.81–126. Available at: http://www.springerlink.com/index/10.1007/BF01228708.

25. Sun Microsystems Inc., 2005. Effective SOA Deployment using an SOA registry repository: a practical guide.

26. Tambouris, E., 2001. An integrated platform for realising online one-stop government: the eGOV project. In *12th International Workshop on Database and Expert Systems Applications, 2001*. pp. 359–363.

27. Wache, H. et al., 2001. Ontology-based integration of information --- a survey of existing approaches. In H. Stuckenschmidt, ed. *IJCAI--01 Workshop: Ontologies and Information Sharing*. pp. 108–117.

28. Wang, J. & Gasser, L., 2002. Mutual online concept learning for multiple agents. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. ACM Press, pp. 362–369.

29. White, A. & Abrams, C., 2005. *Service-Oriented Business Applications Require EIM Strategy*,

30. Wimmer, M., 2002. Integrated Service Modelling for Online One-stop government. *Electronic Markets*, 12(3), pp.149–156.

31. Wimmer, M.A. & Tambouris, E., 2002. Online One-Stop Government. In *the IFIP 17th World Computer Congress - TC8 Stream on Information Systems: The e-Business Challenge*. pp. 1–14.

32. Zeng, L. et al., 2004. QoS-Aware Middleware for Web Services Composition. *IEEE Trans. Softw. Eng.*, 30(5), pp.311–327. Available at: http://dx.doi.org/10.1109/TSE.2004.11.