Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2010 Proceedings

Americas Conference on Information Systems (AMCIS)

8-2010

Information Systems Evolution: A Process Model for Integrating New Services

Jolita Ralyté
University of Geneva, jolita.ralyte@unige.ch

Nicolas Arni-Bloch
University of Geneva, nicolas.arni-bloch@unige.ch

Michel Léonard
University of Geneva, michel.leonard@unige.ch

Information Systems Evolution: A Process Model for Integrating New Services

Jolita Ralyté
University of Geneva, CUI, Switzerland
jolita.ralyte@unige.ch

Nicolas Arni-Bloch
University of Geneva, CUI, Switzerland
nicolas.arni-bloch@unige.ch

Michel Léonard
University of Geneva, CUI, Switzerland
michel.leonard@unige.ch

ABSTRACT

Integration of different components that compose enterprise Information System (IS) represents a big challenge in the IS development. However, it is indispensable in order to avoid fragmentation of the IS and redundancy between different IS applications. In this work we consider service-driven IS engineering as a prospective approach to deal with IS fragmentation and interoperability of different IS components. We introduce the notion of Information System Service (ISS) and propose a process model supporting legacy IS evolution by integration of new services. We claim that such an approach has to take into account a large number of integration situations and therefore has to be built by applying situational method engineering principals and defined as a collection of reusable method chunks.

Keywords

Information System evolution, information service, service integration, information overlap.

INTRODUCTION

Enterprise Information Systems (IS) are generally composed of applications and data that represent enterprise legacy. Due to constant changes and evolution that the enterprise undergoes, like organization restructuring and establishment of new business partners, business process reengineering and innovation, as well as evolution of information technologies, the enterprise legacy became heterogeneous and specialized by trade, department, service etc. Besides, new applications are constantly added to the legacy ones. This situation leads to fragmented IS and therefore to the redundancy between different IS parts which introduces the need for a permanent validation of data, processes and rules consistency. The common part of different IS components represents the interoperability area of the enterprise IS. If no integration is done between these IS components, the interoperability challenge is left to the human using this IS, i.e. the human have to validate "by hand" the consistency between different IS components. Such a human intervention generates extra cost and leads to a poor data quality.

The notion of service and service-driven engineering emerges as a prospective approach to deal with IS fragmentation and interoperability of different IS components not only in the case of inter-enterprise collaborations but also as intra-enterprise solution. The main advantages that IS engineering can take from service-orientation are as follows:

- Modularity – an IS is composed of a collection of interrelated and autonomous components – services – that allows to avoid chaotic IS fragmentation. The modularity is the basis of incremental and evolutionary development and therefore a prerequisite for legacy IS evolution with new components.
- Reusability – IS development is based on the reuse of services, generally representing best practices, as opposed to the more expensive "from scratch" development. The need for reuse is based on the fact that the number of enterprise application domains that should be considered by its IS constantly increases. Legacy IS becomes insufficient and requires to be extended in order to cover these new domains.
- Evolution – a service can be easily replaced by a new one as well as the IS can be extended with new services.

Current service-oriented approaches like Service Oriented Architecture (SOA) (Erl, 2007; Krafzig, 2004; MacKenzie et al., 2006) propose to rebuild enterprise IS architecture in terms of autonomous services to be composed in different ways. In this perspective, services have to be elaborated from scratch in order to avoid any overlap between them. However, the IS lifecycle is a continuous incremental and evolutionary process, i.e. applications and services are regularly added to the IS. It is not possible at each iteration to rethink the entire IS in order to guarantee the autonomy and correctness of the existing and new services. For that, we need methods supporting IS evolution in a modular and flexible way as proposed in (Papazoglou, 2008) but also by preserving legacy IS consistency and regulation policies.

In this work we introduce the notion of *Information System Service* (ISS) (or *information service*) as a potential building block to support legacy IS evolution and we consider this evolution as an integration of new services into the legacy IS in order to avoid IS fragmentation and to preserve its data and rules consistency. Depending on service granularity and abstraction level, its integration into an existing IS can be more or less complex. Service granularity proposed by different authors varies from a simple utility that logs errors to a more abstract complete business process (Haesen et al., 2008; Quartel et al., 2007). In this work we focus our attention on services with a rather high level of granularity each of them representing a work unit with a precise semantic (Arni-Bloch and Ralyté, 2008).

While several approaches dealing with integration of IS data schemas and models were proposed in the literature (Batini et al., 1986; Park and Ram, 2004; Quix et al., 2007) a holistic approach dealing with IS extension with new components is still missing. Therefore, the aim of this paper is to propose a process model to guide new services integration into a legacy IS. The main challenge of this process is to deal with information overlap between the legacy IS and the new service which can be resolved in many different ways. To be able to capture the variability of situations and guidelines we build our approach for ISS integration in line with situational method engineering principals.

INTRODUCING THE NOTION OF SERVICE IN THE IS DOMAIN

The community of IS developers increasingly adopts service-oriented approaches to IS engineering. As discussed above, the main advantage of this approach is that it allows to combine three important systems development principles: the *modularity* by the means of the notion of service, the *reusability* of best practices and the *evolution* of IS through the new service integration.

Nowadays it is unthinkable to redevelop the whole enterprise IS "from scratch". Legacy IS has to evolve with new applications, components, and services from the marked or custom-made. New IS components have to be integrated with the legacy ones in order to avoid IS fragmentation, which can generate extra cost in the IS exploitation. This extra cost is mainly caused by information overlap between different parts of the IS. The enterprise has to manage this overlap either by means of software integration or by considering that different IS actors (users) will handle this information redundancy "by-hand". It is clear that each human involvement in the information overlap management is a potential cause of errors and can lead to a bad quality of enterprise data.

Traditionally, the integration between different enterprise applications is made by connecting them "point-to-point". This kind of integration generally results to a tangle of exchanges of non documented messages, sent in diverse ways and by using various adaptors and scripts. As a result, the IS becomes an imbroglio of applications difficult to maintain and to evolve.

In our work we consider a service-driven approach to deal with IS fragmentation and evolution. For that, we define a particular type of service – the IS Service (ISS) and consider IS evolution as integration of ISS into enterprise legacy IS. Following the traditional SOA, such integration would be limited to the exchange of messages between services. That means, only needs for services and capability to response would be considered but not the information overlap that different services could have. This allows to resolve "point-to-point" integration and represents a step towards integrated IS but does not allow to resolve the problem of information overlap management. The overlap supported by data redundancy maintained between services would continue to generate extra cost and bad quality of data and processes.

Before publishing a service in the registry of the enterprise, it is necessary to guaranty the integrity of the data as well as the alignment of the rules and processes on the IS policies. Besides, the ISS integration impact has to be evaluated taking into account technical, informational and business aspects. At the technical level it means to consider the cost of the interoperability between language, framework or hardware components. The evaluation at the business level consists in analyzing the capability of the enterprise to support new processes and to provide the necessary data. Finally, at the informational level links between data and the compatibility between processes and rules has to be considered. Therefore, we argue that the ISS integration cannot be limited to the exchange of messages between services but has to deal with information overlap and physical integration of services. To enable ISS integration, it is necessary to extend service specification with the knowledge about service data structure and semantics, behavior in terms of actions that can be executed

by the service, rules (data integrity and process rules) to be respected when realizing the service, and roles that are responsible for executing service actions. We define an ISS as a component of an information system representing a well defined business unit that offers capabilities to realise business activities and owns resources (data, rules, roles) to realize these capabilities. Formally, an ISS is defined as follows:

Definition 1 (Information System Service): An Information System Service is an autonomous coherent and interoperable component of an information system composed of four spaces: static, dynamic, rule and role: ς=<*sSs, sDs, sRs, sOs*> where*:*

- *sSs(ς): {Class}*, represents the set of classes of the service ς,
- *sDs(ς): {Action}*, represents the set of actions defined by the ser*vice* ς,
- *sRs(ς): {Rule}*, represents the set of rules that govern the service ς,
- *sOs(ς): {OrganizationalRole}*, represents the set of organizational roles that have rights and responsibilities on the service ς.

A SITUATION-DRIVEN PROCESS MODEL FOR SERVICE INTEGRATION

The process model supporting legacy IS extension with new services has to deal with multiple challenges: to ensure the interoperability of the new service with the legacy IS, to guarantee the quality and consistency of the information, the conformity of actions, the respect of responsibilities and rules and preservation of the legacy IS. It is obvious that such a process can be quite cumbersome mainly because of the information overlap between IS services that can generate inconsistency of data, activities, rules or roles between the legacy and new services. As an example, let us consider University *Diploma Management Service (DMS)* as a legacy IS and University *Online Registration Service (ORS)* as a new service to be integrated in the IS. It is obvious that information overlap is inevitable between these two services – the same type of information (e.g. class *Person*, class *Student*, activity *UpdatePerson*, rule *RegistrationDate < LimitDate*, role *Student,* etc.) is defined in both of them. These shared elements (identical or similar) constitute the information overlap between the IS and the ISS and the integration process has to make this overlap consistent in order to enable reliable new service exploitation within the IS and non violation of the legacy IS consistency. Each of the four ISS spaces is susceptible to be in irregular overlap that needs to be resolved. As a consequence, the integration process mainly consists in resolving the information overlap inconsistency situations between the ISS and the legacy IS. First, the overlap has to be identified and characterized in the global overlap report. This report allows to evaluate the scale of the overlap inconsistency and to decide the way to deal with it. If this report does not indicate any important disparities between the elements of the ISS and IS, the conformity of the four spaces has to be validated next. Otherwise, it is necessary to update the ISS first in order to make the common elements compatible with the IS. If the discrepancy is considered as too important, the integration can be declared as impossible. Next step consists in organizing the overlap i.e. clarifying the relationships between the ISS and the IS by indicating the role of the service on common elements. That can also require to resolve some residual overlap inconsistencies. Finally, the integration result has to be consolidated in order to consider the new situations created by the service integration. To summarize, the process model is composed of five main steps:

- Identify and characterize information overlap,
- Validate information overlap conformity,
- Settle information overlap conformity,
- Organize information overlap,
- Consolidate service integration.

Due to the fact that there are many overlap situations to be considered and there are even more possibilities to deal with them, the process model cannot be limited to a simple prescribed set of activities. We adopt Situational Method Engineering (SME) approach which promotes modularization and formalization of method knowledge in the form of autonomous, interoperable and situation-driven method chunks (Mirbel and Ralyté, 2006) also named fragments or components (Harmsen, 1997; Brinkkemper et al. 1999; Karlsson and Wistrand 2006) that are stored in method repositories and can be assembled in different ways according to the situation at hand. Such a modular definition of methods allows to achieve a better flexibility in method application and to ensure that the method takes all engineering situations into account and provides the best fitting guidance for the project at hand.

In this work we apply the method chunk-driven SME approach (Mirbel and Ralyté, 2006) where the main building block, named a method chunk, integrates method product and process perspective in the same module. This approach is founded on the Map (Rolland et al. 1999) process modeling formalism, which allows to express process models in intentional terms

instead of fixed steps and activities. It provides a representation system based on a non-deterministic ordering of intentions and strategies. In fact, a process is represented as a labeled directed graph where intentions are nodes and strategies are edges between intentions. Since, many strategies can be used for achieving an intention, Map allows to represent complex, flexible and situation-driven process models including multiple techniques to achieve the intentions.

We define our approach for ISS integration as a collection of inter-related and reusable method chunks each of them addressing some specific activity in the ISS integration process and organized into a Map-based process model. The five method steps identified above constitute the intentions of our process model. Map formalism allows us to provide several different ways to deal with the realization of each intention. This knowledge is shown in the map as different strategies allowing to reach the intentions. For example, in order to settle overlap conformity we can proceed by semantic unification or by structural modification of the element in the four service spaces. The guidance of each strategy is captured in the form of one or several method chunks not formalized in this paper because of the lack of space.

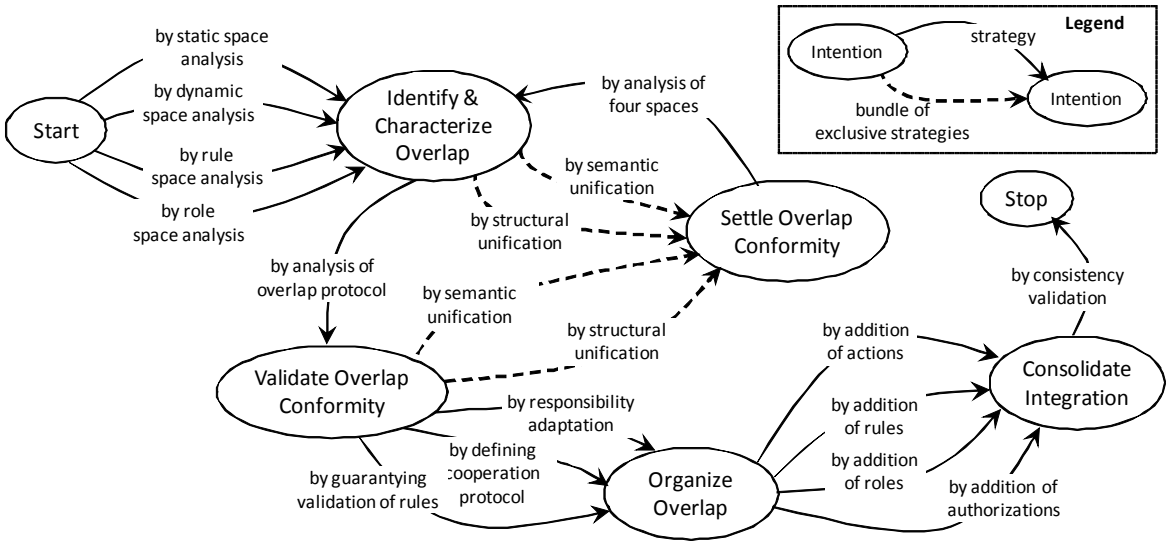

Fig. 1. Process model for information service integration into a legacy IS

Fig. 1 depicts our process model for ISS integration into an IS. Below we detail each of the five intentions and introduce the strategies allowing to reach them.

Overlap Identification and Characterization

The first step in the integration process consists in identifying the overlap situations between the IS and the ISS. During this step, the four information spaces of the service have to be analyzed and an overlap report is elaborated for each of them. The objective is to identify similar elements and to evaluate if they need some structural or semantic unification or not.

Static Space Overlap

Analysis of the static space overlap consists in identifying relationships between classes of the IS and the ISS. We say that a class is in overlap if it is used in the definition of the static space of more than one service. It means that the definition of the given information is used by several services to offer their capabilities.

Definition 2 (Class overlap): Given a class *cl*, we say *cl is in overlap if* $\exists \varsigma, \varsigma' \; \varsigma \neq \varsigma' : cl \in sSs(\varsigma) \wedge cl \in sSs(\varsigma')$ where $sSs(\varsigma)$ represents the static space of the service ς, i.e. its set of classes.

During this step common elements (classes and attributes) are identified as well as their structural and semantic disparities. In the end of this step, each static space element is identified as belonging to the overlap or not. In case of belonging to the overlap, the relationship of this element to the similar element in the IS has to be characterized. The characterization defines the semantic relationships like equivalent, specialization, generalization. In our example, the class *Candidate* of *ORS* is defined as a *specialization* of the concept *Person* in the legacy *DMS*.

Dynamic Space Overlap

Identification of the dynamic space overlap consists in identifying actions in ISS and IS having some functional overlap e.g. producing the same effect on a class in overlap such as updating an attribute or creating a new object of this class. For example, some services can only access the objects of the class in overlap but cannot change their state; other services can only create new objects of the class but cannot modify or delete them. In order to refine the notion of overlap, we define the *effect overlap*. An effect is in overlap if it can be generated by more than one service. It means that several services share the same responsibility on the information represented by the class. For example, the effect <*create an object of the class FollowedDiploma*> can be generated by both services *DMS* and *ORS* and therefore it is in overlap.

Definition 3 (Effect overlap): Given an effect *ef*, we say *ef is in overlap if* $\exists \varsigma, \varsigma' \ \varsigma \neq \varsigma' : ef \in sDs_{effects}(\varsigma) \wedge ef \in sDs_{effects}(\varsigma')$. $sDs_{effects}(\varsigma)$ represents the set of effects that the service ς can generate.

The objective of this step is to produce a report of functional overlap between the IS and ISS and if necessary to consider some modifications in the ISS dynamic space as for example removing or adapting some actions in order to be compliant with the IS regulation. For example, the question to be considered is "Should we keep the *registerStudentToDiploma* action in the *ORS* or not because it already exists in the legacy *DMS*.

Rule Space Overlap

Identification of the rule space overlap consists in analyzing rules (conditions and integrity constraints) defined in the IS and ISS rule spaces and identifying similarities and potential inconsistencies, e.g. considering if the integrity constraints defined in the ISS rule space could be violated by the actions of the legacy IS and vice versa. The set of classes and attributes that participate in the validation of the rule *r* define its validation context *(context(r): {class})*. A rule is in overlap if some classes of its context are in overlap.

Definition 3 (Rule overlap): Given a rule *r*, we say *r is in overlap if* $\exists cl: cl \in context(r) \wedge cl$ *is in overlap.*

In the case of integrity constraints, it is essential to consider the effects in overlap that could violate it. For example, the rule *UniversityRegistrationRequest.dateOfRequest<Diploma.registrationLimit* defined in the new service *ORS* has in its context the class *Diploma* which also belongs to the static space of the service *DMS*. Therefore, this rule is in overlap.

Organisational Role Space Overlap

Finally, identification of the organisational role space overlap deals with the analysis and characterization of roles defined in the IS and ISS and their responsibilities on the corresponding service actions. A role is in overlap if it belongs to several services.

Définition 1 (Role overlap): Given a role *ro*, we say *ro* is in overlap if $\exists \varsigma, \varsigma' \ \varsigma \neq \varsigma' : ro \in eOs(\varsigma) \wedge ro \in eOs(\varsigma')$ where $eOs(\varsigma)$ represents the organizational role space of the service ς, i.e. its set of roles.

In case of completely independent services (without static space overlap), the overlap of roles does not cause any particular problem. But, in case of static or dynamic space overlap, it is important to guarantee that the role in overlap has the same authorisations on both (legacy and new) services.

Overlap Conformity Validation

The aim of this step is to ensure that all overlapping elements (classes, actions, rules and roles), identified and characterised in the overlap report, are conform to each other. The conformity of two elements means that they can be substituted one by another. Two classes are considered as conform if they have the same name, their sets of attributes are identical as well as the sets of methods and they have the same super-classes. Two actions are conform if they have the same name, identical sets of parameters and effects, and identical pre- and post-conditions. Two rules are conform if they have the same name and their content is identical. Finally, two roles are conform if they have the same name and there is no inconsistency in their responsibilities.

The conformity of each couple of overlapping elements has to be validated by the means of the function of equivalence (=). If a couple of overlapping elements is not conform an adaptation of one of them (generally the element of the new service) is realised in the next step.

Overlap Conformity Settlement

Because of the fact that the ISS was developed independently, a number of disparities can exist between its specification and the one of the IS. These disparities can be of semantic nature (the same name but different meaning or vice versa) or of structural nature (different set of attributes in similar classes). Before to be able to integrate the new ISS into the IS these discrepancies have to be resolved first. That means, it can be necessary to modify the concerned classes, actions, rules and roles. As shown in Fig.1 this knowledge is captured in two strategies named *by semantic unification* and *by structural unification*. In fact, each of them is a bundle of exclusive strategies – one per information space to be updated.

Semantic Unification

The semantic unification mainly consists in unifying service terminology, i.e. resolving the problem of the synonyms and homonyms that could exist between a couple of elements from the IS and the ISS. The situation of synonymy exists when two elements have different names but represent the same phenomenon. In the contrary, the homonymy means that two elements have the same name but different meaning. Both situations can be considered in two ways:

1) By adopting the terminology of the IS: the names of concerned elements in the ISS are replaced by those of the IS in case of synonymy or by defining new names of the ISS elements in case of the homonymy.
2) By maintaining both terminologies. In case of synonymy alias is added to the ISS element (several names for the same element). In case of homonymy, a postfix or prefix is added on the corresponding names.

Structural Unification

The structural unification deals with the transformation of different ISS elements in the four information spaces. As mentioned above, there is a different strategy for each space providing several tactics to deal with the situation at hand.

The *structural static space unification* can propose to transform an attribute into a class, to decompose a class in two, to add an attribute, etc. Fig. 2 illustrates two examples of disparity in the structure of similar classes. In both cases, the difference is in the number of attributes. In the case of *Address*, several attributes (*adRoad*, *adCity*, *adCountry*) in one class are defined as one attribute (*adPost*) in the other class. In the case of *Diploma*, the attribute *lectureRegistrationLimit* is missing in one of the definitions.

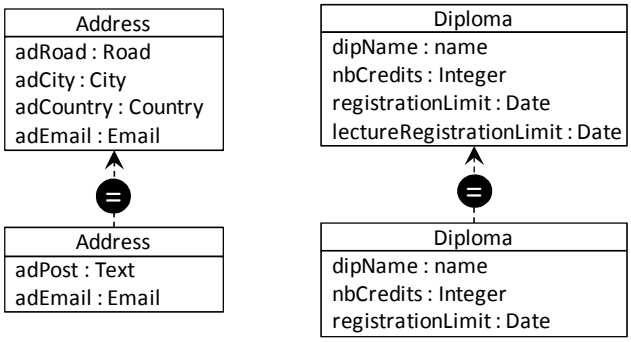

Fig. 2. Example of structural disparity

The *structural dynamic space unification* aims to unify the new service actions with the ones of the legacy IS. In fact, it is necessary to guaranty that equal actions produce the same effects. If not, an effect can be removed from the action or in the contrary added to the action or two actions integrated into one.

The *structural role space unification* means the adaptation of organisational roles and their responsibilities in executing service actions. It is important to make sure that the new service respects the organisational regulations defined in the legacy IS. It can consist in adding, removing or merging some roles; transferring a responsibility from one role to another.

Finally, the *structural rule space unification* aims to guarantee that the new service will respect the rules defined in the IS. There are several possibilities to adapt the service rule space. A rule can be removed if it is too difficult to maintain it once the service is integrated in the IS, or it requires some modification of the IS, or it could hinder the execution of some IS actions. Alternatively, the content of the rule can be modified in order to make this rule compatible with the IS. Finally, a condition type rule can be transformed into an integrity constraint and vice versa.

Overlap Organisation

The objective of this step is to clarify the relationships between the new service and the legacy one. It consists in adapting service responsibilities on common elements, defining a cooperation protocol in order to clarify the visibility of effects on common elements, and guarantying the consistency of service while dealing with integrity constraints. In our process model (see Fig.1) the corresponding guidelines are defined as three strategies: *by responsibility adaptation*, *by defining cooperation protocol* and *by guarantying validation of rules*.

The *adaptation of service responsibility* means determining the effects that the service is authorised to realise to be compliant with the legacy IS. Because the service was developed independently of the context in which it will be integrated, it generally defines effects for each class of its static space in order to be completely autonomous in the manipulation (creation, update, delete, etc.) of the objects of its classes. When a service is integrated in an existing IS, some of the effects can become inconsistent with the legacy IS regulations. Therefore, the responsibility of the service can be modified by specifying which of the overlapping effects can still be provoked by the service and which ones are now forbidden. When an effect is removed from service dynamic space the set of actions triggering this effect have also to be removed. In the contrary, if a new effect is added to the service dynamic space it is necessary to validate that at least one action can trigger it.

In our example, the *ORS* has the class *Diploma* and defines all effects necessary to manipulate the objects of this class (*create Diploma*, *update Diploma*, etc.). Because the *DMS* already produces all these effects it is possible to remove from *ORS* the responsibility of these effects.

When a class is in overlap between several services, the question of visibility of the effects that a service can trigger on the objects of this class has to be considered. Even though two classes were declared as equivalent in the previous step it is still possible that when a service creates an object of this class this object is not visible for other services. The definition *of the overlap protocol* consists in defining the cooperation between services and resolving this kind of situations.

Finally, the *validation of rules*, and especially the validation of integrity constraints, has to be guaranteed to ensure that the new service respects the regulation policy of the IS. Indeed, the rule defined on overlapping classes can be known by one service (e.g the legacy IS) but unknown by the new service and therefore it can be violated by this service. In this step, the objective is to guarantee that all the rules defined in the IS cannot be violated by the ISS and vice versa. Knowing that integrity constraints have a strong impact on service actions and consistency of service data, their consolidation can be done in several ways: a rule can be added or removed from the service rule space, it can also be transformed into a simple condition or its content can be modified. Each of this strategy is captured in a separate method chunk (see (Arni-Bloch et al., 2009) for more details).

Consolidation of the Integration

The last step in our process model aims to ensure that the obtained integration is valid. In fact, the integration of a new service in the IS can generate new situations that did not exist before, neither in the new service nor in the legacy IS. Therefore, these new situations can require extending the service with rules or actions. Addition of new actions entails to consider the roles in charge of realising them. Therefore, this step is reached by revising the four service spaces and adding if necessary corresponding elements in each of them.

CONCLUSION

While the use of service-oriented approaches is growing in the IS development, considering this approach as an exchange of messages in order to integrate IS services is not sufficient. Before publishing a service to be reused in some composition, the validation of the data consistency, rules soundness, process compatibility and organizational roles compliance have to be guaranteed. This is a complex task that needs models representing different information spaces of service definition and methods to help engineers for taking the right choices during the integration process.

In this paper we propose the notion of Information System Service (ISS) as composed of four information spaces: static, dynamic, rules and organizational roles. The formalization presented in this paper is based on the notion of information service as defined in MISS (Arni-Bloch and Ralyté, 2008). However, the proposal can be generalized to other services metamodels such as: WSDL (WC3, 2007), WSPER (Dubray, 2007) or Service specification reference model (Andrikopoulos et al., 2008). These metamodels already define the notions of action (named operation or method) and information model (object-oriented or some kind of type system). To apply our approach on these metamodels, we need to extend them with the notions of rule and role and to annotate the operation or method with the notion of effect, which is a pivot concept to identify actions that could invalidate IS rules.

The main contribution of this paper is the process model for ISS integration into legacy IS which aims to guide the IS engineer in such a complex task. We claim that such an approach is necessary based on situational method engineering principals, i.e. it has to be modular, reusable and flexible. Therefore, we define this approach as a collection of inter-related method chunks dealing with a huge number of ISS integration situations. This kind of flexibility and adaptability to a specific situation leads us to a new kind of approach for IS integration. Besides, the Map formalism used for representing this process model enables its evolution by adding new strategies and intentions into this map and therefore new method chunks.

Currently, we focus our effort on identifying and evaluating different situations that can occur in the ISS integration process, defining method chunks satisfying these situations and defining situational indicators to support method chunks selection according to the situation at hand. A tool support is under development to store and publish the method chunk knowledge. Finally, evaluation of our approach in the real scale enterprise projects is our current and future preoccupation.

REFERENCES

1. Andrikopoulos, V., Benbernou, S. and Papazoglou, M. P. (2008) Managing the evolution of service specifications. *Proc. of the 20th Int. Conf. on Advance Information Systems Engineering – CAiSE'08*, LNCS 5074, Springer Berlin, 359–374.
2. Arni-Bloch N., Ralyté J. and Léonard, M. (2009) Service-Driven Information Systems Evolution: Handling Integrity Constraints Consistency. *The practice of Enterprise Modeling. Proceedings of the 2nd IFIP WG 8.1 Working Conference, PoEM 2009. A. Persson and J. Stirna (Eds.)*, LNBIP 39, Springer Berlin, 191–206.
3. Arni-Bloch, N. and Ralyté, J. (2008) MISS: A Metamodel of Information System Service. *Proc. of the 17th Int. Conf. on Information System Development (ISD'08)*, Paphos, Cyprus, Springer US, 177-186.
4. Batini, C., Lenzerini, M. and Navathe, S.B.(1986) A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4), 323–364.
5. Brinkkemper, S., Saeki, M., Harmsen, F. (1999) Meta-Modelling Based Assembly Techniques for Situational Method Engineering. *Information Systems*, 24 (3), 209-228.
6. Dubray, J.-J (2007). Wsper an abstract soa framework. Technical report. www.wsper.org.
7. Erl, T. (2007) SOA Principles of Service Design. *Prentice Hall PTR*.
8. Haesen, R., *et al.* (2008) On the definition of service granularity and its architectural impact. *Proc. of the 20th Int. Conf. on Advance Information Systems Engineering – CAiSE'08*, LNCS 5074, Springer Berlin, 375–389.
9. Harmsen, A.F. (1997) Situational Method Engineering. Moret Ernst & Young, Amsterdam, The Netherlands.
10. Karlsson, F. and Wistrand, K. (2006) Combining method engineering with activity theory: theoretical grounding of the method component concept. *European Journal of Information Systems*, 15, 82–90.
11. Krafzig,D., Banke, K. and Slama, D. (2004) Enterprise SOA: Service-Oriented Architecture Best Practices. *Prentice Hall PTR.*
12. MacKenzie, M., *et al.* (2006). Reference model for service oriented architecture 1.0. *Technical report, Oasis*.
13. Mirbel, I. and Ralyté, J. (2006) Situational Method Engineering: Combining Assembly-Based and Roadmap-Driven Approaches. *Requirements Engineering*, 11(1), 58–78.
14. Papazoglou, M. P. (2008). The challenges of service evolution. *Proc. of the 20th Int. Conf. on Advance Information Systems Engineering – CAiSE'08*, LNCS 5074, Springer Berlin, 1-15.
15. Park, J. and Ram, S. (2004) Information systems interoperability: What lies beneath? *ACM Trans. of Information Systems*, 22(4), 595–632.
16. Quartel, D.A.C., et al. (2007) Cosmo: A conceptual framework for service modelling and refinement. *Information Systems Frontiers*, 9(2-3), 225–244.
17. Quix, C., Kensche, D. and Li, X. (2007) Generic schema merging. In *Proc. of the 19th Int. Conf. on Advance Information Systems Engineering – CAiSE 2007*, LNCS 4495, Springer Berlin, 127–141.
18. Rolland, C., Prakash, N. and Benjamen, A. (1999) A Multi-Model View of Process Modelling, *Requirements Engineering,* **4**(4), 169–187.
19. WC3 Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language (2007) http://www.w3.org/TR/wsdl20/.