Computer Science • 13 (4) 2012 http://dx.doi.org/10.7494/csci.2012.13.4.71

Andrzej Zalewski
Marcin Szlenk
Szymon Kijas

AN EVOLUTION PROCESS FOR SERVICE-ORIENTED SYSTEMS

Abstract *Evolution of service-oriented systems is quite a new research area, which becomes more and more important as engineering challenges move from enabling service-orientation onto the maintenance and evolution of already developed service-oriented systems. However, the development of suitable evolution processes and methodologies is still an open research problem. The evolution process presented in this paper has been designed to address the evolution of service-oriented systems, which are technically built out of a set of service compositions. The presented process comprises phases and tasks compliant with ISO 20000. The underlying model of service-oriented system consisting of business processes and corresponding service composition models has also been presented. A traceability model and tools supporting change impact analysis have also been provisioned for. Preliminary industrial validation indicates that the evolution process should be easy to adapt by the industry.*

Keywords service-oriented architecture, service engineering process, service composition, software evolution

1. Introduction

The founding principles of service-oriented architectures (SOA) as extensive reuse of existing, loosely-coupled services and easy to modify composition of business processes made out of those services, are supposed to facilitate system evolution breaking the border line between development and maintenance. However, existing SOA development methodologies (for a survey see section 3) are aimed at transforming legacy systems into service-oriented ones and offer a rather limited support for the evolution of SOA systems. They address the issue of the evolution of services identified at earlier stages, leaving alone the evolution of service compositions, which seem to be the core concept offered by service-oriented architectures.

There exist currently no evolution process crafted so as to address the evolution of SOA systems comprised of a set of service-compositions. This is the gap this paper is supposed to address.

The rest of the paper has been organised as follows: section 2 defines precisely the notion of service-oriented systems and the evolution of such systems used in this paper, section 3 presents related work, section 4 describes the model of service-oriented systems underlying the evolution process as well as the traceability approach, section 5 presents the evolution process (workflow, artefacts, impact analysis techniques), section 6 reports on industrial interviews validating our approach, section 7 discusses the contribution against other research in the field, and section 8 summarises the paper and presents research outlook.

2. Basic notions

The basic assumption underlying service-oriented architectures, and probably the one, that made SOA so popular, is the concept of developing a new functionality (i.e. new services) by the composition of loosely coupled (thus, easy to modify), independent and even stateless services. This was supposed to boost system modifiability, which is often the prevailing concern for business stakeholders nowadays. Hence, service composition should become the main way, in which new functionality is developed.

In the above context, we understand service-oriented system as a set of business processes composed out of services (independently of their internal or external origin). The evolution of SOA system is about modifying set of business processes, e.g. adding new processes, removing or changing existing ones.

3. Related work

The service-oriented world envisaged in section 2 is rather a world to come than the world we actually live-in. Pure service-oriented systems, i.e. built exclusively out of loosely-coupled, independent services, do not exist in reality. In practice, service-oriented systems are currently built on top of already existing non-service-oriented ones for internal integration [10]. Therefore, a lot of research effort was devoted to

address the issue of migrating legacy systems to SOA. Suitable methods can be found in e.g. [1], [2], [17], [8], [18].

The emergence of a market for third-party services and the deployment of more systems crossing organisational boundaries, possibly making their services publicly available, will change the above condition and make the evolution of service compositions a primary focus. Here, identification and development of services, so as to wrap existing functionality and enable interaction between systems, was the key. The evolution was understood as making changes to the services, i.e. their interfaces, functionality, etc. (compare e.g. [1]).

The research record on the maintenance and evolution of service-oriented systems is rather sparse. It concerns mainly selected evolution issues, like change traceability [15], change propagation [5], [13], versioning [7], impact analysis [4], and model driven-approaches to service composition, e.g., [12]. The research challenges in this field have been investigated in papers [10], [9], [6]. We found no work on evolution processes for service-oriented systems as the development of such processes and methodologies was indicated as one of the challenges. Nevertheless, maintenance has been included in a post-deployment phase in [3] as well as it has been provisioned for in the methodology presented in [11]. Evolution of services has been accounted for in the fractal process of SOMA methodology with the concept of successive iterations. In [12], authors propose to use change management mechanisms to control the evolution of service compositions. Therefore, the development of evolution processes and methodologies tailored for service-oriented systems understood as a set of service compositions is still an open research problem.

4. Model of service-oriented system

Our understanding of the concept of service-oriented system is reflected by a system model, which underlies our evolution process and methodology. We assume that the key artefacts that document service-oriented systems are business process models expressed in BPMN. These can be categorised into two groups:

- business processes (BP) show the business processes of the organization in abstraction from how they are implemented in the IT systems as compositions of services;
- service compositions (SC) show the implementation of business processes as a composition of services with different scenarios of their invocations.

The terms business process model (BP model) and service composition model (SC model) will be used later, respectively. Business processes play the role of requirements specification, which are fulfilled by the appropriate service compositions.

Let us note that both kinds of models are BPMN diagrams that can show the same process or its fragments at different semantic levels. This is a typical example of a refinement relationship between models [14], where service composition model refines the business process model. This relationship can also concern models belonging to

the same group, i.e. two BP models or two SC models, where the refined model shows modified version of the same process or more details of its part. The example structure of this relationship is depicted in Fig. 1. Information about refinement dependencies is treated here as an integral element of model documentation.

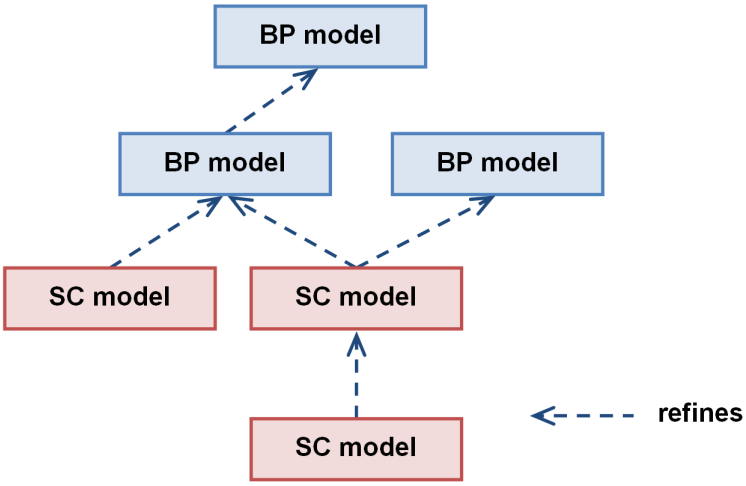

Figure 1. A refinement relationship between BPMN models.

The complete structure of SOA models are following a layered structure comprising of the following consecutive layers:

- business motivation goals, needs, etc;
- business process models (expressed in BPMN);
- services composition (expressed in BPMN);
- models of services (expressed with models promoted by such approaches as SOMA [1], SOMF [2], used to develop and evolve services alone);
- low level, detailed technical models (typically UML) and executable code.

Evolution is a business-driven process, in which a system is changed in response to continually emerging or changing business needs. An evolution step is a transformation between two consecutive versions of SOA models. This transformation is achieved by propagating changes through the layered model (i.e. transforming those models). It means that business needs are reflected by the changes to the business processes, these in turn enforce changes to services composition orchestration, and choreography. Changes to services composition may require changes to services models and so forth (changes to service definition and underlying software can be done using e.g. service-oriented development methods as SOMA [1]; however a detailed analysis of this issue is beyond the scope of this paper). The evolution is captured using the model presented in Fig. 2. The concept of architectural decisions [16] have been applied as a vehicle to represent transformations of SOA system models. Evolution consists of

"Evolution Step", which consist of a set of "Evolution decisions" reflecting the related business needs. Evolution decisions comprise two distinct sets of decisions representing changes to the models: "business process decisions" and "service composition decisions". The former ones represent changes made to business processes in reaction to business needs (business motivation), while the latter ones changes to service composition made to accommodate service composition to changes made to business processes. There are naturally many ways in which emerging or changed business needs can influence business processes and the same applies to the influence of changes of business processes on services composition. This is a kind of creative engineering task, which cannot be neither fully formalised nor automated.

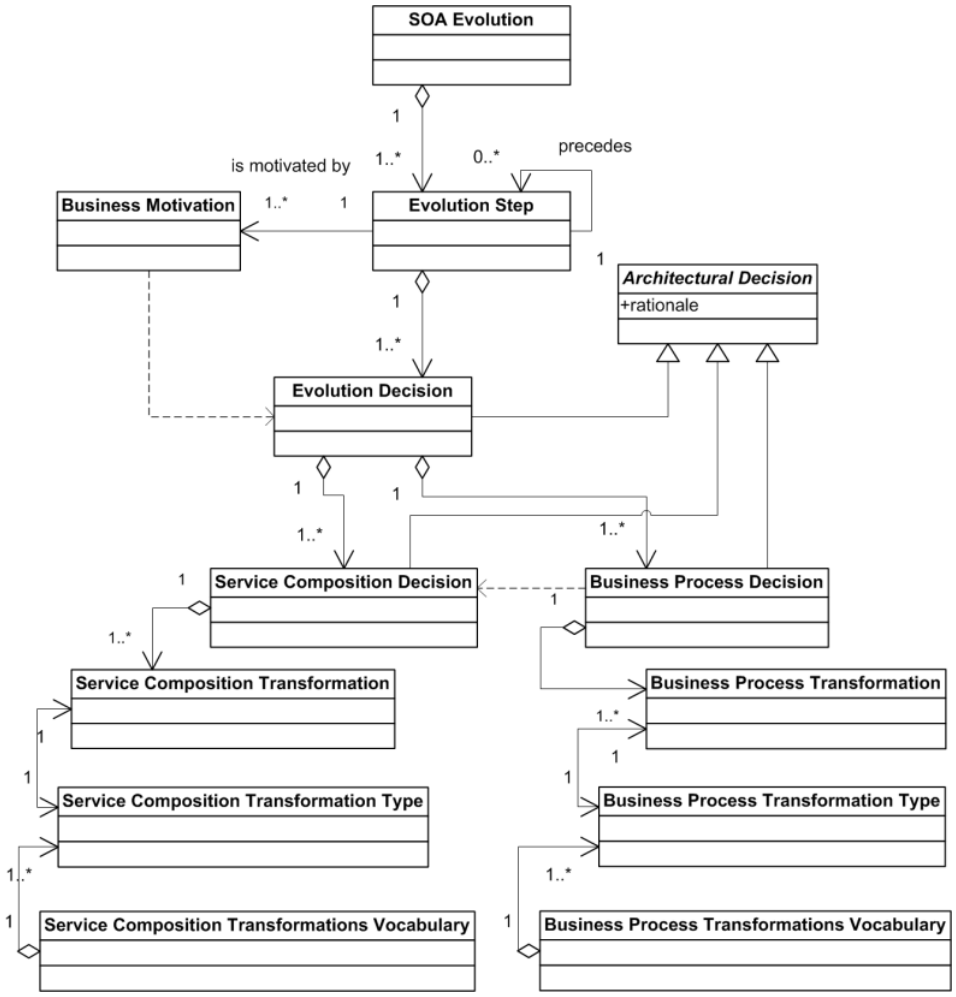

Figure 2. Structure of SOA evolution model.

We propose to facilitate this process by:

- the vocabulary of business process transformations; the latter can be used to perform and represent any change to business process models;
- the identification of possible changes to service processes composition resulting from respective changes to business process models;
- the use of architectural decisions to support a coarse-grained traceability from business needs down to service orchestration changes, while change details can be preserved as a superposition of a number of transformations predefined in transformation vocabulary (they can even be identified with the approach and tools presented in [5]);
- documenting, when needed, the logic underlying a certain change as a rationale of architectural decisions.

5. The evolution process for service-oriented systems

The evolution process presented here comprises of:

- The modification process (section 5.1) the order of phases and tasks, in which changes are assessed, approved, made and reviewed. As many changes are usually processed at the same time, many instances of the development process can be running at the same time;
- The artefacts (section 5.3), which are the products of phases and tasks;
- The methods for change impact analysis (section 5.4);
- The mechanisms supporting the detection of overlapping changes being processed at the same time (section 5.5).

Typically, the definition of a development process comprises the first two of the above components. However, as evolution is a permanent condition for service-oriented systems, the artefacts describing system design can be changed many times during system lifecycle. Changes introduced simultaneously may overlap, i.e. the same artefact may be modified by more than one instance of the modification process at the same time. These conditions have to be addressed appropriately.

5.1. The modification process

System evolution consists of a series of steps. Each of these steps is accomplished with the modification process, which new instance is initiated for every submitted Request for Change document (RFC). These general properties and phases of the modification process have been presented in Fig. 3.

The modification process consists of four phases compliant with the requirements for change management process defined in ISO 20000:2005 standard:

1. Change assessment submitted change is assess in terms of its impact (on quality attributes, SLA, other processes, etc.), urgency, cost, benefits and risks.

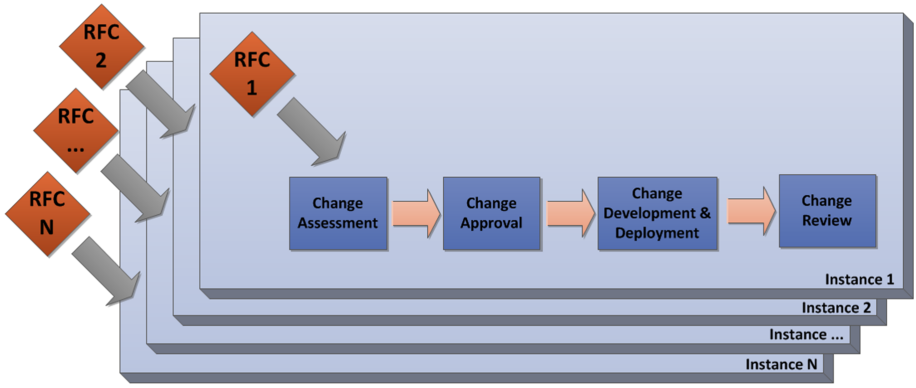

Figure 3. Overall Structure of the Evolution Process.

2. Change approval on the basis of information gathered during the Change assessment phase, considering business priorities and other business factor decision makers decide whether to proceed with change. The phase includes also scheduling the approved changes and resolution of change overlaps.
3. Change development and deployment this is a configurable part of the Modification process; different processes can be applied for Change development agile: Feature Driven Development, Scrum, XP or non-agile: waterfall, RUP. The choice will depend on the established development practices and experience of the development team.
4. Change review this is also an optional phase, which presence is required by ISO 20000, however, it should be defined if the organisation undertakes the change review activities. The detailed workflow of the Modification process has been presented in Fig. 4. It shows tasks and associated artefacts. The most complicated seems to be the "Change assessment" phase. It starts from the "Preliminary assessment", in which changes described in Request for Change (RFC) are assessed on the basis of expert knowledge of business and system analysts in terms of impact on functionality, quality (including Service Level Agreements), effort needed to complete the changes and risks connected with the change process. The results of such an assessment are examined ("Preliminary Assessment Approval"), whether they contain an amount of information sufficient for the approval at the stage of "Change Approval and Scheduling". If more detailed information on change impact is needed "Change Prototyping" is performed. The change prototype comprises models of changed business processes and service compositions, developed to the extent appropriate for the "Detailed Assessment". It does not have to be complete with regard to the designed change, fully verified nor tested. The rest of the process structure seems to be self-explanatory.

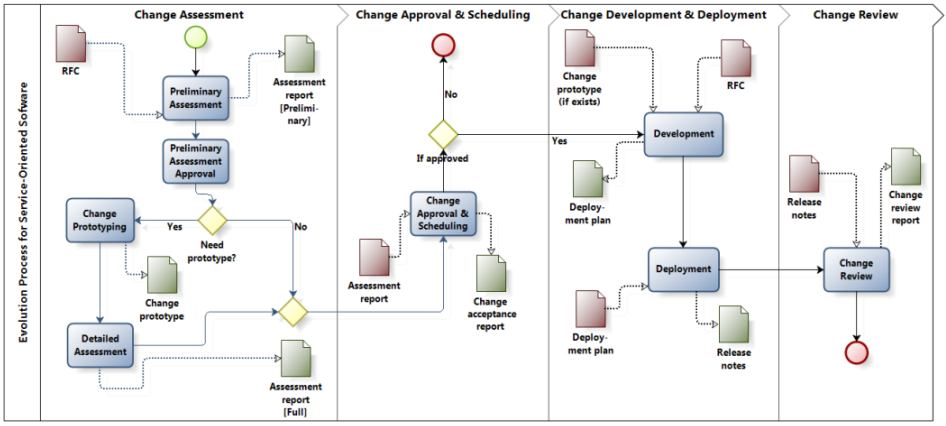

Figure 4. Detailed workflow of the evolution process for service-oriented systems.

Table 1 shows, which roles are engaged in tasks comprising the modification process. They are compliant with the roles defined in the Rational Unified Process.

Table 1
Mapping of tasks and roles.

Tas k	Role(s) engaged
Preliminary Assessment	Business Architect, System Analyst
Preliminary Assessment Approval	Business Architect, System Analyst
Change Prototyping	Business Architect, System Analyst, Business Designer
Detailed Assessment	Business Architect, System Analyst
Change Approval & Scheduling	Change Control Manager, Stakeholders (business)
Development	Development team, comprising for example: Business Architect, System Analyst, Business Designer, Software Architect, Designer, Implementer, Integrator, Test Manager, Test Analyst, Tester
Deployment	Deployment Manager, Integrator
Change Review	Business Architect, System Analyst

5.2. Configuring the modification process

The "Development" task of "Change Development and Deployment" phase as well as "Change review" are designed as extension points, where most suitable methods can be applied. Hence, the modification process can be customized so as to reflect experience, knowledge and skills of the development team.

Table 2 summarises the available configuration options. Both agile and non-agile methods can be used to develop changes, and its choice should depend on the development practices established within the organisation, experience and knowledge of the development team.

Table 2
Configuration of service oriented-system evolution process.

Change Assessment	Change Approval	Change Development & Deployment	Change Review
Always exists	Always exists	Always exists (Scrum, RUP, Waterfall, XP, FDD, etc.)	Optionally — Review procedures

Table 3 indicates, how basic concepts of the development methods are expressed in terms of the system model underlying the evolution process. Here, the scope of the development work (planned, completed, etc.) is expressed as a set of business process and/or service-compositions, which are subject to changes.

Table 3
Mapping the agile and non-agile concepts onto "Change Development & Deployment" phase.

	List of business processes subject to changes	List of business processes selected to change in a iteration	List of business processes modified during the iteration and integrated with other changes implemented before
Scrum	Product Backlog	Sprint Backlog	Working increment of the software
Extreme Programming	User Stories	Release Plan	Small Release
Feature Driven Development	Features List and Development Plan	Features selected to develop in a iteration	Completed client-valued function
RUP	Set of Use Cases	Set of Use Cases	Set of Use Cases

5.3. The artefacts

The artefacts of the evolution process have been presented and described in Table 4.

Table 4
Artefact list.

Artefact	**Description**
RFC	The change is described in business or technical terms. The document contains also explanations of the change and indications concerning its importance/priority.
Assessment report [Preliminary] or [Full]	The document includes: • change of scope: – list of business process as subject to change, – list of service compositions subject to change, • impact analysis description of change impact on: – quality (including SLAs), e.g. reliability, performance, business continuity, etc.,; – list of business processes affected by the changes (e.g. requiring revision); – overlapping changes; • cost estimates, • identified risks, • attachments (other documents used for or created during the assessment process), in case of [Full] version of the document change prototypes are included here.
Change prototype	Set of business process and service composition models containing: • modified versions of existing business processes and service compositions; • models of new processes introduced • list of removed business processes and service compositions. The above models are drafts of the assessed changes. They have not been fully developed, verified, tested.
Change acceptance report	The document contains: • notes explaining the need and rationale for the approved change, • effort / cost estimated, • allocation of the cost within budget (the source of change financing); • time schedule for change development and deployment; • attachments including: RFC, Assessment reports and Change prototype.
Deployment plan	Deployment plan contains of installation/deployment instruction of new release.
Release notes	Report on the deployment containing list of bugs, which have been corrected or not in the developed version.
Change review report	Defined individually by the organisation.

5.4. Support for change impact analysis

Business process or service compositions (both are BPMN models) can be added or modified and their models can undergo changes or be removed to carry out the change described in RFC. The two most general states in which a BPMN model can be:

- Up to Date (UTD) the model is up-to-date, i.e. it reflects at present all the existing requirements that are related to the process and the process aspects shown in this model and it is consistent with all the models it refines.
- To Be Updated (TBU) the model should be reviewed so as to decide whether any changes or even its removal is required to respond to the new requirements or changes in the other models the given one refines.

In the above view, the state of the model M will change to TBU state whenever:

- the requirements that are related to the process or the process aspects shown in M changes;
- any changes are made to the models that are refined by M;
- the set of models that are refined by M changes (new refinement dependencies for M are added or some of the existing ones are removed, e.g. due to the removal of the model being refined).

Let us note that changes to one of the models enforce all of its refined models to be examined, but not necessarily to be changed. The above models state transitions are depicted in Fig. 5.

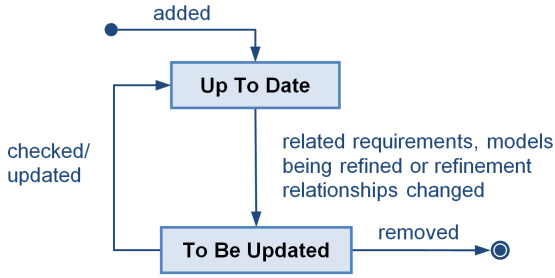

Figure 5. State transitions for a model.

Requests for Change represent new or changed requirements. In the Change assessment phase RFC is associated with existing BPMN models (i.e. business processes or service compositions, that are supposed to be changed) or new models added so as to implement the change. Therefore, the development of a new model will be in some sense initiated by submitted RFC or RFCs. The example structure of such relationships is depicted in Fig. 6. Information about links between submitted RFCs and the existing BPMN models, together with the state transitions model for BPMN models (presented before) give a simple tool that can be used both for the change assessment and change implementation. The examples of its usage are shortly outlined below.

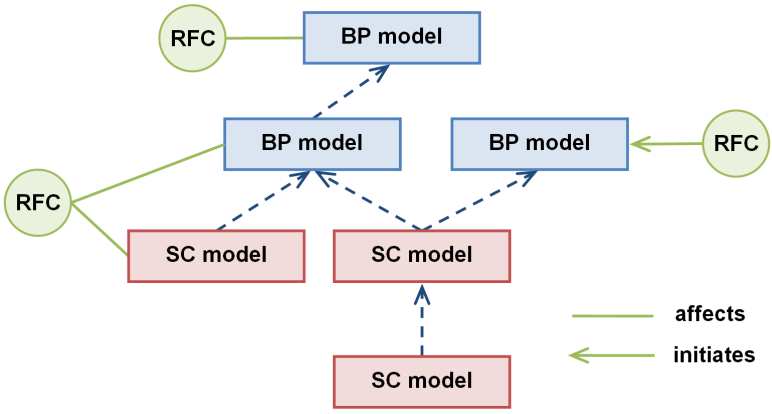

Figure 6. Relationships between BPMN models and Requests for change (RFC).

A usage scenario during change assessment

Submitted RFCs are first considered by business architects and associated with the existing business process models which makes their states change to TBU state. Business architects also decide whether and which additional business process models should be constructed in response to RFCs.

Submitted RFCs may not only be of a business nature but also be just technical, relating to services and their compositions. This is why they should be then considered by software architects in an analogical way, but this time in the context of services composition models. As a result of these activities the information about new required models and models that require checking and possible changes is obtained.

A usage scenario during change implementation

First, business architects construct new business process models that are required in response to submitted RFCs, adding refinement relationships if needed. Next, they check and potentially modify business process models that are in TBU state, starting with the models on top of the refinement hierarchy. Because a model modification changes the states of refined models to TBU state, this process is usually iterative and ends when all business process models reach UTD state. Modifications of a model also include its removal. Analogical activities are performed in such a case by the architects.

5.5. Support of change conflict detection and resolution

As many changes can be processed at the same time, some of them can turn out to depend on each other. Two changes depend on each other if there are business processes and/or service compositions modified by both of them. Identification and resolution of overlaps takes place in the “Change approval and scheduling” phase.

There are two general approaches to address the issue of overlapping changes:

- to schedule overlapping changes so that one of them is performed when the other one is completed,
- to process the dependent changes together as if they were a single change.

Let us consider an example illustrating how the "refines" relations between the models can be employed to detect overlapping changes. Dean's office system is designed to manage students and teachers data (e.g.: personal data repository, providing personal data to other systems, exchanging data with other departments). The relations between business process and service composition models have been shown in Fig. 7. The change described by RFC1 concerns "Dean's office BP model", while RFC2 concerns "Students data processing BP model". It is obvious that the trees of artefacts refining "Dean's office BP model" and "Students data processing BP model" contain common nodes, i.e. there are models that have to be revised in the course of implementing both changes. The changes described in RFC1 and RFC2 overlap (require modifications to the same artefacts) and such a problem has to be resolved.

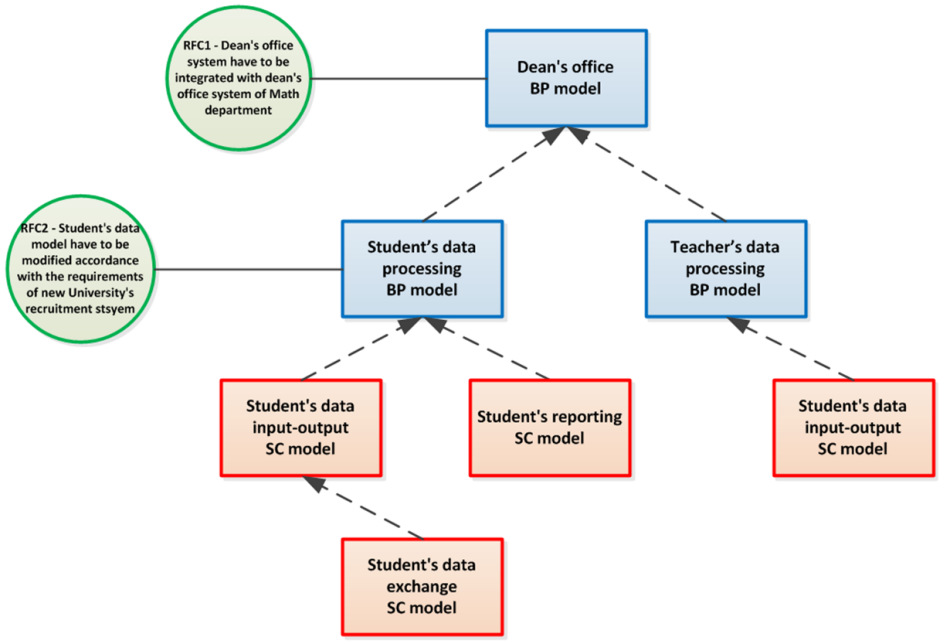

Figure 7. An example illustrating the analysing of change overlaps.

6. Process validation

The short industrial interviews have been carried out so as to validate the proposed evolution process. After a short presentation of the process, the interviewed were asked the following questions:

- Do you perceive the presented service-oriented system evolution methodology easy to include within the procedures of your organisation?
- Do you find traceability solution sufficient for your needs?

We interviewed 11 companies. Eight companies found the process very easy to include into their procedures, two found it rather easy, and one found it rather difficult. Seven companies found the traceability model fully sufficient, 3 rather sufficient and only one insufficient. During face to face discussions the interviewed emphasised that compliance with ISO 20000 is an important factor enabling real life industrial application of the evolution process.

7. Discussion

Our research comes back to the very roots of service-oriented architecture and focuses on service compositions. As the number of available third-party services grows, they should become the main means of developing new or changed functionality. Such service-oriented systems are supposed to be highly modifiable, which will facilitate frequent business-driven changes. Hence, the evolution of service compositions will dominate the life-cycle of such systems.

The evolution process presented in section 5 was designed so as to make it easy to introduce it in the conditions of corporate IT and their management procedures. On the contrary to many service-oriented methodologies (like [1], [2]), we are not trying to impose an entirely new approach. Instead, we provide an evolution process compliant with the popular ISO 20000:2005 standard. Because of that the target organisation does not have to revise its system maintenance procedures as there must be a really important reason to change established, proven development and maintenance practices.

Flexibility of the process, which allows an organisation to use any of the most popular development process for developing service compositions is another factor making industrial adoption easier, as an organisation can use its own development team without enforcing any unknown methodology.

The proposed traceablity model provides for a coarse-grained tracking of model dependencies and changes. Such a simple model should be quite easy to implement with one of the existing version management system or its extension. If more detailed information about changes made to the models is needed an approach and tools of [15] can be used.

8. Conclusion

The evolution of service-oriented systems is becoming more and more important as service-oriented systems mature and the number of third-party services grows. This rises a number of research challenges, among them is the development of methodologies and processes.

The process proposed in this paper provides a versatile, easily adaptable scheme, using which service-oriented system comprised of a set of service compositions can evolve. The short industrial survey indicates that practitioners have generally found it compliant with the practices that they have established for system change management. Therefore, they found it quite easy to introduce in the specific conditions of their organisations.

Further research will include:

- Extensive industrial case studies,
- Developing software tool support to accompany evolution methodology,
- Methodology for making and documenting changes made to the system in the course of the evolution process.

Acknowledgements

The present research has been supported by a grant from the Ministry of Science and Higher Education of the Republic of Poland under grant no. 5321/B/T02/2010/39.

References

[1] Arsanjani A., Ghosh S., Allam A., Abdollah T., Ganapathy S., Holley K.: Soma: A method for developing service-oriented solutions. *IBM Systems Journal*, 47(3), 2008.

[2] Bell M.: *Service-Oriented Modeling: Service Analysis and Design and Architecture*. Wiley Publishing, 2008.

[3] High R., Kinder S., Graham S.: *IBM's SOA Foundation: An Architectural Introduction and Overview*. November, 2005.

[4] Hirzalla M., Zisman A., Cleland-Huang J.: *Using Traceability to Support SOA Impact Analysis*. IEEE World Congress on Services (SERVICES), Washington, DC, USA, July 2011.

[5] Hoa K., Ghose A.: Supporting change propagation in the maintenance and evolution of service-oriented architectures. *Software Engineering Conference (APSEC)*, 30(3), 2010.

[6] Kontogiannis K., Lewis, G.A. Smith D.: *The Landscape of Service-Oriented Systems: A Research Perspective for Maintenance and Reengineering*. SEI, 2007.

[7] Laskey K.: Considerations for SOA versioning. *Enterprise Distributed Object Computing Conference Workshops*, 16, 2009.

[8] Lewis G., Morris E., Smith D., Simanta S.: *SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment.* Software Engineering Institute, Carnegie Mellon University, 2008.

[9] Lewis G., Smith D.: *Service-Oriented Architecture and its Implications for Software Maintenance and Evolution.* FoSM 2008, IEEE, s, October 2008.

[10] Lewis G., Smith D., Kontogiannis K.: *A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems, TECHNICAL NOTE.* March 2010.

[11] Mittal K.: *Build Your SOA Part 1: Maturity and Methodology.* IBM, May 2005.

[12] Orriéns B., Yang J., Papazoglou M.: *Model driven service composition.* ICSOC 2003, Springer-Verlag, 2003.

[13] Ravichandar R., Narendra N., Ponnalagu K., Gangopadhyay D.: Morpheus: Semantics-based incremental change propagation in SOA-based solutions. *IEEE International Conference on Services Computing*, 7–11, July 2008.

[14] Rumbaugh J., Jacobson I., Booch G.: *The Unified Modeling Language Reference Manual.* 2004.

[15] Sindhgatta R., Sengupta B.: An extensible framework for tracing model evolution in SOA solution design. In *OOPSLA Companion(2009)*, pp. 647–658.

[16] Tyree J., Akerman A.: Architecture decisions: Demystifying architecture. 22(2): 19–27, 2005.

[17] Winter A., Ziemann J.: Model-Based Migration to Service-Oriented Architectures. *Proc. of the International Workshop on SOA Maintenance Evolution (SOAM 2007), 11th European Conference on Software Maintenance and Reengineering* (CSMR 2007), Amsterdam, March 20–23, 2007.

[18] Ziemann J., Leyking K., Kahl T., Werth D.: *SOA Development Based on Enterprise Models and Existing IT Systems.* Exploiting the Knowledge Economy: Issues, Applications and Case Studies, Edited by Paul Cunningham, IOS Press, 2006.

Affiliations

Andrzej Zalewski
University of Technology, Institute of Automatic Control and Computational Engineering, Warsaw, Poland, a.zalewski@elka.pw.edu.pl

Marcin Szlenk
University of Technology, Institute of Automatic Control and Computational Engineering, Warsaw, Poland, m.szlenk@elka.pw.edu.pl

Szymon Kijas
University of Technology, Institute of Automatic Control and Computational Engineering, Warsaw, Poland, s.kijas@elka.pw.edu.pl

Received: 20.03.2012
Revised: 12.07.2012
Accepted: 3.09.2012