Customisable Transformation-Driven Evolution for Service Architectures

Aakash Ahmad
Lero - The Irish Software Engineering Research Centre
School of Computing, Dublin City University
Dublin, Ireland
ahmad.aakash@computing.dcu.ie

Claus Pahl
Lero - The Irish Software Engineering Research Centre
School of Computing, Dublin City University
Dublin, Ireland
cpahl@computing.dcu.ie

***Abstract*—Service-based architectures have now become commonplace, creating the need to address their systematic maintenance and evolution. We propose to enable transformation-driven evolution for service architectures in a semi-automated fashion. In contrast to the existing solutions like service wrapping, migration or run-time adaptation etc., the proposal supports primitive and customisable architectural transformations to support an incremental evolution for service architectures. An empirical approach is adopted to investigate the extent to which the architecture evolution tasks (i.e. modeling, transformation and refinement) can be automated and validated in context to the central hypothesis for architecture-centric software evolution. Based on the initial results, we plan to proceed toward modeling and automating the architectural evolution in a formal way.**

***Keywords*-Software Architectures; Architecture Reconstruction and Evolution; Evolution of SOA;**

I. Introduction

Service-Oriented Architecture (SOA) is considered to be a business centric, architectural approach for designing and developing distributed, enterprise systems. The established theory and practices on such service-based software primarily focuses the design or development efforts, often overlooking the requirements and complexities for SOA maintenance and evolution [1]. However, as service systems are developed and deployed the ultimate challenge lies in accommodating the changing requirements to prolong the productive life and economic value for existing software.

Based on the proposed life cycle of an SOA, the taxonomy of service-oriented software research in [1] prioritises the current and future research agendas, explicitly highlighting the needs for SOA evolution. In contrast, the current efforts like [2][1] concentrate on legacy wrapping or migration towards service software thus falling short of supporting an explicit evolution within SOAs. Alternatively, the work proposed in [3], [4] enables dynamic design and evolution for service orchestrations to facilitate the runtime adaptation for (web services in) service-based architectures.

Based on a review of related work, it is apparent that rigorous processes, frameworks and engineering methods etc are lacking in the existing efforts to support a coherent, stepwise evolution for service architectures. Nevertheless, this is helpful in identifying the problem and justifying the needs for the proposed solution, i.e. *how to enable customisable, transformation-driven evolution for service architectures in an automated fashion*. Therefore, we aim at improving the existing efforts like [4], [2] etc., by supporting an incremental evolution of SOA elements through transformation at different abstraction (meta, architecture, architecture style) levels using:

- An operational layering with a focus on operation and execution (what and how to change) consisting of basic transformation operators and patterns. Pattern notions allow to categorise the composed transformations by their impact on source and target architecture elements.
- A user-defined customisable layer focusing on design aspects (why to evolve) allows rule-based declarative specification of transformation goals to generate service-based architecture. SOA-specific styles are applied to refine the transformed architecture to support style-based evolution.

It is vital to preserve structural and semantic constraints of SOA elements at all layers to maintain an overall target architecture integrity. With a coherent architecture evolution framework, we can systematically address the (structural and behavioural) evolution issues for SOAs semi-automatically.

This paper is organised as follows. A review of the related work is summarised in Section II that helps in outlining the research questions in Section III. Research methodology and proposed solution are presented in section IV; followed by the results to date in Section V. The paper concludes with a summary of research validation and action plans.

II. Related Work

We aim for a practical applicability of the final solution. Therefore, it is of central importance to investigate the state of the art (wherever possible) both from an industrial and an academic perspective. We generally classified the review into four different areas comprising of i) service based software reuse (wrapping, migration, transformation etc.), ii) software design and architecture evolution, iii) dynamic (web) services adaptation and iv) formal architectural transformations. In the following, we identify the shortcomings

[1]Also tools as Visual Studio (http://www.microsoft.com/visualstudio/) and Eclipse Webtools (http://www.eclipse.org/webtools/).

along the possibilities to exploit the existing efforts to devise a systematic solution to SOA evolution issues.

In context to SOA development, the current industrial solutions are often driven by their commercial values (like ROI, time to market, competitive advantages etc.), rather than offering sustainable, generalised solution for architecture evolution challenges. For example, the recent generation of Eclipse and Visual Studio support legacy wrapping with a service layer supporting an ad-hoc (service-oriented) legacy reuse, also referred as service facade in [7]. In such solutions, the problem is strengthened with false assumptions that after modification the system will remain relatively stable not eventually replaced thus posing contradictions to the evolutionary nature of software.

In the context of architectural evolution, the Software Architecture Evolution Model (SAEV) [8] provides a guideline to enable the evolution of (component and connector based) architectures at different abstraction (meta, architecture, application) levels. In contrast, the work in [2] supports the automation of architectural migration towards an SOA using graph transformation rules over a model of the annotated source code in a formal way. However, these solutions are limited because these are assumed to be deployed for internal integration/operation, where there is still some control over deployed services when compared to heterogeneous, distributed services architectures.

One of the recent initiatives include UML4SOA [3] that defines a high-level domain specific language for modeling and transforming web service orchestrations to support the dynamic service composition. Also, in contrast to dynamic service adaptation [4], we limit our approach to a declarative (user-defined) composition of atomic services to include the composed services in service architectures as in [9].

The discussion above is helpful in outlining the limitations along with the challenges and potential in supporting the architecture-centric evolution for SOAs. It is worth mentioning here the recent series of workshops focusing on the research agendas for Maintenance and Evolution of SOA (MESOA) [1]. These primarily aim at identifying and addressing the fundamental problems for SOA evolution by pinpointing the research needs as: . . . *community wide efforts are required to develop processes, frameworks, transition patterns etc; to support systematic maintenance and explicit evolution for potentially distributed service architectures* . . . .

Central Hypothesis: In an attempt to realise this potential, we formulate the central research hypothesis, i.e. *the application of architectural transformations while preserving its integral constraints could support a dependable architecture-centric software evolution.*

A layered evolution comprising of i) primitive transformations that can develop the foundation for ii) user-defined rules supporting customisable architecture evolution at different abstractions. It is vital to preserve the structural and semantic properties of SOA elements at all levels to ensure dependability (correctness) of the target architecture.

III. Research Questions

In context to the identified limitations and the central hypothesis for architecture evolution, the primary question to focus the central research ideas is:

How to enable customisable, architecture-centric evolution for service software in a (semi-) automated way?

It triggers the following sub-questions:

RQ1 - How to enable service-based software evolution in an semi-automated way?

(Allow automated transformations at different abstraction (meta, architecture) levels [8] with appropriate user intervention to guide the evolution tasks.)

RQ2 - How to enable customisable (architecture-centric) software evolution?

(Allow layered evolution by composing primitive (meta-level) transformations into user defined (architecture-level) transformations and refinement.)

RQ3 - What degree of formalism is required to support the modeling, transformations and (correctness) verifications for the target architecture?

(Allow graph-theoretical modeling, transformation and verification of target architecture as highlighted in [2])

Objectives: The first two sub-questions (RQ1, RQ2) focus on the 'how' (to evolve) perspectives in context to the Lehman's Laws for Software Evolution that corresponds to the application of engineering methodologies to enable software evolution. The later sub-question (RQ3) emphasises the need for a formal approach to execute and verify the evolution tasks ensuring target architecture validity. Based on these questions we outline the project objectives as:

1) Automation: The aim is to allow the automation of primitive and customisable architecture transformations with appropriate user (architect's) intervention.

2) Dependability: A formal approach can achieve the necessary dependability that allow software architects to trust automated transformations for target architecture integrity.

IV. Methodology and Proposed Solution

We propose to develop an applicable solution to SOA evolution issues, following an empirical approach with Canonical Action Research (CAR) methodology [10]. Within CAR we followed and slightly adjusted the Cyclical Process Model (CPM) consisted of: *i) Diagnosing*, *ii) Planning and Intervention* followed by *iii) Evaluation and Reflection*.

Architecture Evolution Cycle: To set a realistic scope for the proposal, we focus on developing three activities collectively referred as the evolution cycle for service architectures, presented in Figure 1 and are explained below.

The evolution cycle is partially inspired by the SOA-Migration Horseshoe [11] that represents a generic process model for migrating legacy software towards an SOA. Our proposal in Figure 1, however is restricted to architectural

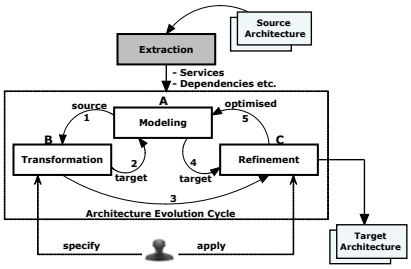

Figure 1. Evolution Cycle for Service Architectures

modelling (compared to the enterprise modelling in SOA-Migration Horseshoe). This restriction is necessary to set and achieve realistic goals i.e to model and transform architectural entities rather than the entire enterprise model. In addition to the steps proposed in the migration horseshoe we propose to refine the transformed design to mitigate the counter-productive transformation effects for a robust target architecture. Furthermore, we rely on existing literature like [12] (and the reverse engineering steps in) [2] for source architecture extraction instead of accommodating the extraction task implicitly in the evolution cycle. Also, we propose to support the user input (customisable transformations and style based refinement) into the evolution process that is lacking in the horseshoe model.

A. Modeling

Instead of solely focusing on the global (architectural) view of the system as in [2], we propose to transform the architectural elements at different abstraction levels namely at the meta, architecture and architecture-style level supporting layered evolution, often overlooked in the existing solutions.

B. Transformation

The aim is to support a transformation driven evolution for service architectures by means of i) primitive (pattern based) and ii) customisable (user defined rule based) transformations to instantiate service-oriented architecture. The combination of modeling and transformation is exploited in [13] to solve the architecture evolution problem by utilising the Model Driven Engineering (MDE) techniques. Instead, we focus on developing an evolution framework that enables customisable, automated transformations and refinement with an appropriate user intervention in the process. However, we believe that the proposed solution has the required flexibility (thanks to its underlying metamodel(s)) to accommodate it as an MDE based solution, if required.

C. Refinement

While executing pattern based transformation we observed some counter productive architectural transformations (we refer to them as transformation anti-patterns) that need to be identified and minimised for and optimised target architecture. Therefore, style-based refinement not only minimises the inherited flaws into the transformed architecture but also supports a robust and extensible architecture that better meets target specifications with enforced stylistic constraints.

V. Results and Contribution

At the time of writing, we concentrate on initial sub-questions (RQ1, RQ2) that focus on enabling the transformation of source service architecture elements at different abstraction levels toward the target architecture. In this context, we have developed a coherent service architecture evolution framework presented in Figure 2 that realises the evolution tasks illustrated in Figure 1. Preliminary results for this work are validated and accepted as a peer-reviewed conference publication [14], that are summarised below:

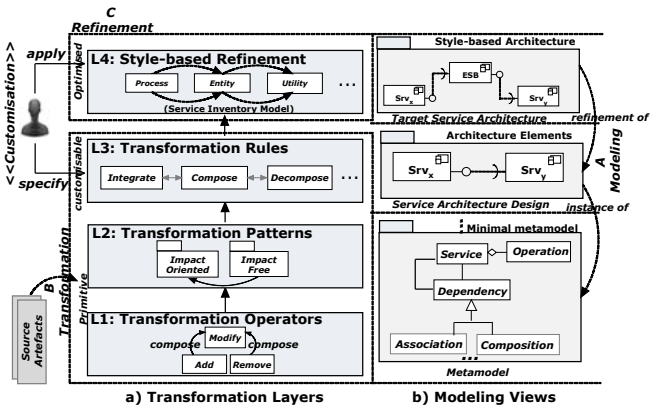

Figure 2. Layered Service Architecture Evolution Framework

Modeling: A metamodel (based on UML 2.0) has been developed to model the fundamental SOA elements (atomic and composite services) with enforced structural constraints for the target architecture integrity. It serves as a foundation for automated transformation at upper layers.

Transformation: The transformation layers (L1, L2, L3) are developed and preliminary validated to enable primitive and customisable SOA transformations. Transformation patterns are vital in classifying and reusing the re-currant architectural changes during primitive transformations.

Refinement: It mainly corresponds to future work that aims at applying the service architecture styles/patterns to refine the transformed architecture. The benefit for such refinement is to optimise the transformed architecture following certain configurations with enforced stylistic constraints. Currently, we focus on service inventory model detailed in [7]. However, the work proposed in [15] present an architecture style ontology as a modeling language that serves as a guideline to formally apply more conventional (pipe and filter, hub and spoke etc) styles into target SOA.

VI. Validation and Action Plan

We follow a step-wise approach to evaluate the individual research tasks and their sub-tasks that determine the validity of the overall work. We extend and utilise (wherever

possible) the technologies and literature like (CAR [10], SAAM [16], UML 2.0 etc.) to evaluate the central hypothesis for architecture-centric software evolution as:

- *Scenario-Based Validation:* Currently, we focus on electronic payment system evolution cases studies to validate the applicability of the proposed architecture evolution framework. Based on the metamodel transformation we have been successful in validating the primitive transformations (operators when composed into transformation patterns i.e L1 and L2) using ATL[2] model to model transformations. An overall validation (including L3 to generate the transformed architecture) must be performed using SAAM [16] to evaluate hypothesis for architectural transformations.

- *Prototype Evaluation:* Once we achieve the intermediate results (i.e user-defined transformation rules at L3) we plan to develop a prototypical transformation engine to validate the customisable, automated transformations. The tool and the generated data shall form the basis for the practitioners (software engineers and architects) for a survey and usability based analysis to evaluate the effectiveness and applicability of the proposed solution in an industrial context.

A high-level validation plan of the research tasks in context to their individual evaluation in presented in Figure 3.

Evolution Task	Sub-task	Technique	Status	Evaluation
Modeling	*Metamodel*	UML 2.0	Complete	Structural Constraints
	Architecture	Pattern-based Transformation	In progress	SAAM [16]
Transformation	*Primitive*	Operators and Patterns	Complete	M2M (XML)
	Customisable	Declarative Rules [9]	Future Work	SAAM
Refinement	*Style Application*	Service Inventory Patterns [7]	Future Work	Architectural Scenarios, SAAM

Figure 3. The Validation Plan w.r.t Evolution Tasks and their Evaluation

Action Plans: in an overall context are summarised as:

- An investigation of ontology-based architecture representation approaches, based on [6], [5] for architecture description and [15] for pattern modelling, as the basis of architectural evolution.

- A formal graph-based transformation, based on the ontology-based architecture representation, of source service architectures towards the target architecture that comprises the primary future work (RQ3), guided by [2].

- A significant challenge lies with semi-automated declarative composition (of service orchestration and choreography) included in target SOA, as presented in [9].

- An interesting identification is the emergence of transformation anti-patterns resulting from counter-productive pattern-based transformations. We strive for the identification of these anti-patterns and their resolution (refinement) that is vital to achieve target SOA with desired specifications.

ACKNOWLEDGEMENT

This work is supported, in part by Science Foundation Ireland through grant 03/CE2/I303_1 to Lero - The Irish Software Engineering Research Centre (www.lero.ie).

[2]Atlas Transformation Language (ATL). http://www.eclipse.org/atl/

REFERENCES

[1] G. Lewis, D. Smith, N. Chapin, and K. Kontogiannis, "MESOA 2009: 3rd International Workshop on Maintenance and Evolution of Service-Oriented Systems," *IEEE International Conference on Software Maintenance*, 2009.

[2] R. Heckel, R. Correia, C. Matos, M. El-Ramly, G.Koutsoukos, and L. Andrade, "Architectural Transformations: From Legacy to Three-Tier and Services," in *Software Evolution*. Springer Verlag, 2008, pp. 139–170.

[3] P. Mayer, A. Schroeder, and N. Koch, "MDD4SOA: Model-Driven Service Orchestration," in *12th IEEE Intl Conference on Enterprise Distributed Object Computing*, 2008.

[4] H. Verjus and F. Pourraz, "A Formal Framework For Building, Checking And Evolving Service Oriented Architectures," in *European Conference on Web Services*. IEEE, 2007.

[5] C. Pahl, "Layered Ontological Modelling for Web Service-oriented Model-Driven Architecture", *European Conference on Model-Driven Architecture - Foundations and Applications ECMDA'2005*, LNCS 3748, Pages 88-102, 2005.

[6] C. Pahl, "Semantic Model-Driven Architecting of Service-based Software Systems," *Information and Software Technology*, vol. 49, no. 8, pp. 838–850, 2007.

[7] T. Erl, *SOA Design Patterns*. Prentice Hall, 2009.

[8] N. Sadou, D. Tamzalit, and M. Oussalah, "How to Manage Uniformly Software Architecture at Different Abstraction Levels," in *24th Intl Conf on Conceptual Modeling*, 2005.

[9] S. Ponnekanti and A. F. , "Sword: A Developer Toolkit for Web Service Composition," in *11th International World Wide Web Conference*, 2002.

[10] R. Davison, M. Martinsons, and N. Kock, "Principles of Canonical Action Research," *Information Systems Journal*, vol. 14, no. 1, pp. 55–86, 2004.

[11] A. Winter and J. Ziemann, "Model-based Migration to Service-oriented Architectures," in *International Workshop on SOA Maintenance and Evolution*, 2007.

[12] C. Matos, "Service Extraction from Legacy Systems," in *4th International Conference on Graph Transformations*, 2008.

[13] B. Graaf, "Model-Driven Evolution of Software Architectures," in *Ph.D. Thesis, Delft University of Technology*, 2007.

[14] A. Ahmad and C. Pahl, "Pattern-based Customisable Transformations for Style-based Service Architecture Evolution," in *Intl Conf on Next Generation Web Services Practises*, 2010.

[15] C. Pahl, S. Giesecke, and W. Hasselbring, "Ontology-based Modelling of Architectural Styles," *Information and Software Technology*, vol. 51, no. 12, pp. 1739–1749, 2009.

[16] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-Based Analysis of Software Architecture," in *IEEE Software*, 1996.