Requirements-Driven Software Service Evolution

Feng-Lin Li^{1,2}

¹ Dept of Information Engineering and Computer Science, University of Trento, Trento, Italy

² School of Software, Tsinghua University, Beijing, China

Abstract. Evolution is of key importance for software services as they need to satisfy ever-changing customer requirements under open and dynamic operating environment. The majority of current research on service evolution is focused on maintaining the compatibility between evolved service and existing clients, while few works look into its root cause. This paper proposes to study service evolution from a requirements' perspective. The key research problems - change propagation and traceability - in requirements-driven service evolution are discussed and the concept of "*feature*" is used to bridge the gap between requirements model and services model. A feature-oriented approach is presented, which includes model transformation & refinement (from goal model to feature model), and specification derivation (from feature model to service design space), to effectively address these challenges.

1 Introduction

For many years, scientists have been wondering how the turtle shell comes into being. Until recently, the unearthed fossils of *Odontochelys* (toothed turtle with half-shell, the oldest known tortoise) in China may help to answer this evolutionary enigma. The discovery provides interesting evidence stating that turtle shell formed from the bottom - plastron first - and grew bony extensions of ribs and bone formation above backbones that eventually joined to form the carapace that the modern turtles have.

In this example, certain existing features (*rib* and *backbone*) evolve and find a new use. As time unfolds, new features (*back shell*) develop. In a similar spirit, in software /service engineering, we can view services as a set of functional and non-functional features, which can evolve, to adapt to ever-changing requirements and environment.

We treat feature as evolution unit because of its two-fold interpretation: feature as atomic requirement, and feature as atomic functionality/quality of software service[1]. Moreover, in the days of *on-demand* use and *pay-as-you-go* consumption, it is sensible to think of service as a bundle of features that users choose from and subscribe to.

With the promotion of Service Orientation, software services are receiving increasing attention because they offer a loosely coupled architecture for enterprise application integration. Meanwhile, services reside in an open and constantly changing internet environment, and need to cope with ever-changing user requirements. Accordingly, services need to change continuously, which would give rise to evolution.

This thesis aims to study the mechanisms of requirements-driven service evolution. Our proposal is inspired by biological evolution and uses a feature-oriented approach to propagate requirements change to service evolution.

2 Problem Statement

According to Jackson and Zave's definition [2], the requirements problem is stated as: Given a set of requirements R , and a set of domain assumptions D , find a specification S such that $D, S \models R$. In our proposal, we assume that a specification is a set of services, and each service has a set of features.

We use an *Airline Ticket Booking* service to illustrate the research problems under discussion. In Fig 1, the functional and non-functional requirements are represented as hard-goals (ovals) and soft-goal (clouds); domain assumptions and specifications are represented as resources (rectangles) and tasks (hexagons) respectively. A case of the requirements problem is: to satisfy the goal "*Ticket be Paid Online*"(R), the service need to support task "*Processing Credit Card Payment*"(S) under the domain assumption "*Customer owns a Credit Card with Available Credits*"(D).

Fig. 1. The Partial Goal Model the Airline Ticket Booking Service

In this example, several changes are possible. For instance, since credit card information could be stolen and misused, customers may require the payment to be authorized. To fulfill this new requirement R_1 , a new functionality "*Validate Customer Identity*" may be added into the existing specification S . Such requirement change leads to a propagation problem: **given $D, S \models R$ holds, if requirement R changes into R' , how to evolve specification S to S' so that $D, S' \models R'$ still holds?**¹

Similarly, the change of domain assumption would also result in specification evolution. E.g., if a customer who has only debit card also wants to buy ticket online (the assumption "*Customer owns a Credit Card with Available Credits*" is violated), then a new feature "*Processing Debit Card Payment*" would be in need. In this case, the problem can be stated as: **given $D, S \models R$ holds, if domain assumption D changes into D' , how to find a new specification S' so that $D', S' \models R$ still holds?**

Meanwhile, service providers themselves will sometimes update existing services because of economic cost or technical progress. The unilateral evolution would raise the question of whether the changed specification could still satisfy the original requirements. This traceability problem can be noted as: **given $D, S \models R$ holds, if the specification S changes to S' , would the entailment $D, S' \models R$ still be true?**

With the problem settings as our basis, we aim to solve the following principal research problems in requirements-driven service evolution: How to project the changes

¹ Keep in mind that changes could occur in both functional and non-functional requirements.

in user requirements to service design space (propagation)? How to evaluate whether existing (evolved) services still satisfy changed (original) requirements (traceability)?

3 The Proposed Solution

This work proposes a systematic approach to address the above presented problems. Basically, we specify requirements using goal model, and then identify features from the fully refined requirement model. At the same time, we associate each operational goal with a service, and compose services when their goals are closely related. Thus a service will have a set of cohesive features and is able to satisfy relevant goal(s). The change of requirements will be first manifested in goal model, and then be propagated to services through features.

3.1 Conceptual Model

The relation between goal, feature and service is shown in the conceptual model of Fig. 2. As it shows, a goal could be either hard-goal or soft-goal, both of which can be decomposed into sub-goals; a goal is usually achieved by more than one service, and a service is able to support more than one goal; generally, a goal would imply the corresponding features of a service: a functional feature is able to support more than one hard-goal and contribute to several soft-goals; also, non-functional feature would reflect the soft-goals of customers.

Fig. 2. The Conceptual Model of Goal, Feature and Service

On the other hand, service has two types: *atomic* and *composite*; structurally, a service is a set of operations with input, output, pre-condition and effect (IOPE); meanwhile, a service could be characterized as a set of functional and non-functional features, which could be further refined. Through refinement to functional features, we are able to derive service operations and capture its behavior. Also, non-functional features will be described properly to support evolving services.

3.2 Methodology

In accordance with the conceptual model, we propose a feature-oriented approach to deal with the change propagation from goals to services, and its reverse process. The approach includes two phrases: *model transformation & refinement* and *specification derivation*. The former focuses on identifying and refining features; the later would derive service operations from refined feature model and build service behavior model based on its operations.

— Model Transformation and Refinement

In our approach, we first need to have a refined goal model available, in which each operationalized goal will be assigned to an atomic or composite service. It is possible for more than one operational goal to be associated with the same service. Also, services could be composed according to the structure of goal graph. E.g., if a goal g is AND-decomposed into operational goal g_1 and g_2 , which are associated with service s_1 and s_2 , then s_1 and s_2 could be composed if g_1 and g_2 are closely related.

Next, we need to identify feature from the refined goal model, which includes tasks, domain assumptions and quality constraints (the concretization of soft-goal). Regarding the refinement of an operational goal g_o , there are some general guidelines for identifying candidate features: a) each OR-decomposed task of g_o ; b) each/a combination of AND-decomposed tasks of g_o ; c) each (or several cohesive) quality constraint(s). As for the domain assumptions, since they are usually attached to tasks, whether it should be considered as feature candidates needs further exploration.

After the coarse-grained features are identified, we need to further refine them to derive specification. The refinement includes [3]: *decomposition* (refine a feature into its constituent features), *specialization* (make it more detailed), and *characterization* (identify attributes of feature). E.g., a “Check-out” feature can be decomposed into “Pricing” and “Taxation”; then “Taxation” could be specialized to “Fixed-Rate Taxation” and “Rule-Based Taxation”; finally, at least two attributes “Amount” and “Tax-rate” could be characterized from the “Fixed-Rate Taxation” feature.

— Specification Derivation

Through model transformation, we get to know what features a service will have; on the other hand, based on the fully refined feature model, together with a set of proper enacting rules, we are able to derive the operations that a service needs.

Briefly, the enacting rules are as follows: in the refinement process of a feature f , a) if f is decomposed into several constituent features $f_{1...m}$, then f will be mapped to an operation and be viewed as the virtual root of the sub-diagram that starts from it; b) if f is specialized into sub-features $f_{1...m}$, then each f_i ($1 \leq i \leq m$) will be a concrete value of one specific parameter of service operation. The parameter stands for the feature specialization and the operation is the lowest virtual root along the backward path from f ; c) if f is characterized as a set of attributes $a_{1...n}$, each attribute a_i ($1 \leq i \leq n$) will be mapped to a parameter of the specific operation.

It is expected progress when moving from feature model to service design space. However, it is insufficient without specifying its behavior because service is composite by nature for better fulfilling extensive functionalities.

In this proposal, we model service as hierarchical state machine and specify transitions using an *event-condition-action* (ECA) language on the basis of service operations. The syntax of the ECA language could be briefly noted as:

$$t_i : \text{event} [\text{guard}] / \text{action} [\text{effect}]$$

in which t_i is the label of transition, *event* is external world-change phenomena (e.g. sending or receiving a message), *guard* is the condition upon which corresponding *action* (usually refers to service operations) will be performed, and finally the *effect* (e.g. service state change) will be brought about.

As an example, *?bookTicket[HasTicket]/reserveTicket() [tickets-1]* describes when a “bookTicket” message is received, if the guard (“HasTicket”) holds, the service will make a reservation and preserve a seat (action: *reserveTicket*). Consequently, the number of available tickets will be decreased (effect).

On the other hand, we model non-functional features as quality constraints. For each non-functional feature, we at first identify its soft-goal and key performance indicator (KPI), and then model the KPI’s contribution to the soft-goal as its value varies. Similar to soft-goal contribution, the support a KPI provides could be sufficient/partial positive or negative (++/+/-/-).

NonFunctional Feature: *Comfort Level*

SoftGoal : *Travel Comfort*

KPI: *Leg Room Space*

Contribution : *Leg Room Space* $\geq X$: ++ ; ...

Take the non-functional feature “*Comfort Level*” in airline service as an example: its soft-goal and KPI are “*Travel Comfort*” and “*Leg Room Space*” respectively. We model the contribution of “*Leg Room Space*” to “*Travel Comfort*” as “++” if it is $\geq X$. To better satisfy the soft-goal, one way is to offer a “*Priority Boarding*” feature, which enable customers who board first to place carry-on luggage in overhead bins and thus gain more “*Travel Comfort*” by enhancing the KPI - “*Leg Room Space*”. Afterwards, following the aforementioned method, the “*Priority Boarding*” feature will derive new service operation(s) and evolve service behavior model.

3.3 Evaluation

To evaluate the approach, we will develop a tool that extends classical feature model with refinement relationship, transforms refined feature to service operation automatically, enacts service behavior model on the basis of its operations systematically, and demos how clients should evolve synchronously when service changes. In brief, we will validate our approach in a semi-automatic experiment setting.

4 Related Work

Much interesting work has been done on mapping goal model to feature model. For example, Yu et al. [4] discuss the generation of feature model from goal model automatically through one-to-one mapping between goal and feature. Similarly, K. Uno et al. [5] present a systematic approach to derive feature model from an integrated goal

model (which is obtained by merging multiple goal graphs) through identifying common and variable features. Their work is valuable for establishing traceability between requirements and features; however, it is insufficient in supporting of evolutionary changes at either goal- or feature- model level.

Also, feature has recently been applied to model diversified functionalities of services and enabling service customization. T. Nguyen et al. [6] propose a template-based approach to build the mapping between feature and service operations. Their approach customizes service interfaces according to selected features, but could not support service evolution due to the lacking of feature semantics.

In software service evolution, the well-known challenge is to keep the compatibility between evolved service and existing client applications at either structural [7] or behavior level [8][9]. Through our extensive survey [10], we found that starting from the definition of service evolution given by M. Papazoglou [9], many efforts have committed themselves to solve the problem by using kinds of tactics, such as versioning, design pattern/adaptor, model and theory [7], but few of them paid enough attention to the change propagation from requirements to services, one of the fundamental problem of evolution.

5 Conclusion

In this paper, a feature-oriented approach is proposed to address the problems in requirements-driven service evolution. The methodology is designed for propagating changes from requirements to services as well as tracing influence back from services to requirements. It is able to identify how service structure should evolve and how service behavior should adapt when requirements change. Being different from the majority of work that merely concentrate on the incompatibility (i.e. the influence of evolution), we are able to answer the more fundamental problems in service evolution: what needs to evolve and how should it evolve.

References

1. Reid Turner, C., Fuggetta, A., Lavazza, L., Wolf, A.L.: A conceptual basis for feature engineering. *Journal of Systems and Software* 49(1), 3–15 (1999)
2. Jackson, M., Zave, P.: Deriving specifications from requirements: an example. In: 17th International Conference on Software Engineering, ICSE 1995, pp. 15–15 (1995)
3. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. DTIC Document (1990)
4. Yu, Y., do Prado Leite, J.C., Lapouchnian, A., Mylopoulos, J.: Configuring features with stakeholder goals. In: *Proceedings of the 2008 ACM Symposium on Applied Computing*, pp. 645–649 (2008)
5. Uno, K., Hayashi, S., Saeki, M.: Constructing feature models using goal-oriented analysis. In: 9th International Conference on Quality Software, QSIC 2009, pp. 412–417 (2009)
6. Nguyen, T., Colman, A.: A Feature-Oriented Approach for Web Service Customization. In: 2010 IEEE International Conference on Web Services (ICWS), pp. 393–400 (2010)
7. Andrikopoulos, V.: A Theory and Model for the Evolution of Software Services. Open Access publications from Tilburg University (2010)

8. Ryu, S.H., Casati, F., Skogsrud, H., Benatallah, B., Saint-Paul, R.: Supporting the dynamic evolution of web service protocols in service-oriented architectures. *ACM Transactions on the Web (TWEB)* 2(2), 13 (2008)
9. Papazoglou, M.P.: The Challenges of Service Evolution. In: Bellahsène, Z., Léonard, M. (eds.) *CAiSE 2008*. LNCS, vol. 5074, pp. 1–15. Springer, Heidelberg (2008)
10. Li, F.L., Liu, L., Mylopoulos, J.: Software Service Evolution: A Requirements Perspective. In: *Proceedings of the 2012 IEEE International Conference on Computer Software and Applications Conference (COMPSAC)*, Izmir, Turkey, July 16–20, pp. 353–358 (2012)