ASREF: An Adaptive Service Requirements Elicitation Framework Based on Goal-Oriented Modelling

Wei Qiao Lin Liu Jian Xiang

Knowware Group, School of Software, Tsinghua University, Beijing, China 100084
{qiaow05, xiangj05}@mails.tsinghua.edu.cn, linliu@tsinghua.edu.cn

Abstract*:* This paper formulates the service-oriented requirements analysis process as a feedback control system, in which the classical "once for all" philosophy is replaced with a continuous negotiation and adaptation process. Based on existing requirements model and new service request, the proposed service requirements elicitation framework ASREF aims to achieve an optimal service supply and demand relationship. Current control variable is the similarity of the service requirements model and the service capability model. Major contributions include: a dynamic questionnaires system for service requirements elicitation, the construction process of goal-oriented service models based on user's answers to the elicitation questions and the dual feedback control mechanism to service requestor and provider. A virtual travel agency example is used to illustrate the proposed ideas.

Keywords: Services, Requirement Engineering, Dynamic Questionnaire, Feedback

1 Introduction

Service orientation is becoming a leading form of web-based software applications, a common feature of which is the need to understand and characterize what the customer wants and to compose services to meet those requirements effectively. While many techniques of conventional requirements elicitation could potentially be applied to services, the open dynamic nature of service orientation introduces new challenges. Comparing with the conventional requirements engineering, in which the elicitation process usually takes a "face-to-face" mode, the case of services has to take a "back-to-back" mode. In the former case, requirements engineers obtain original requirements information from the targeted customers directly using methods such as interviews and task analyses. In opposite, the later means that services requestors and services providers have to conduct a double blind search during the requirements elicitation process due to the fact that they have seldom chance to interact face to face in advance of the development of services or the formation of business agreements.

Developing a new methodology to assure the quality of requirements elicitation in the

"back-to-back" mode is a critical task of requirements engineers in the services era. The success of services as a business and computational paradigm depends on how well the two sides understand requirements and constraints of each other. Ill-defined and misrepresented requirements of services may lead to unbalanced service-level agreement, or no agreement can be formed in the worst case. Thus, Requirements engineering for services plays a definitive role in service engineering. In order to achieve efficient service design, publication, discovery, binding and evolution, requirements facilities to handle these service requirements issues automatically and systematically is needed. It is even more preferable if the service manipulation framework can be adaptive to the evolving requirements of users, and incrementally obtain new service capabilities according to the changing needs.

Thus, the essential questions regarding the elicitation process should include: How a prospective service provider can map its core competence into a maximum set of users requirements? Where and how can we acquire and accumulate valuable service requirements knowledge? Is it from existing published service profiles, from Service-Level Agreements, or from logs of user queries? Unfortunately, there is no systematic approach in requirements engineering or service engineering that can address these issues yet. We need a mechanism to guide the service providers through the process of transforming legacy systems into easily reusable and adaptable services according to user's real needs, which should lead into a win-win situation in the service-oriented world.

Based on our previous work on a service modelling ontology [7] and service requirements elicitation mechanism, *SREM* [21], this paper further elaborates and enhances the ideas by formulating the service-oriented requirements analysis process as a feedback control framework *ASREF*. Thus, the classical "once for all" elicitation philosophy is replaced with a continuous negotiation and adaptation process. Based on the existing goal-oriented service requirements model and new service requests, the proposed service requirements elicitation framework *ASREF* dynamically issues elicitation questions to help form an optimal service supply and demand relationship. Current control variable is the similarity of the service requirements model and the service capability model.

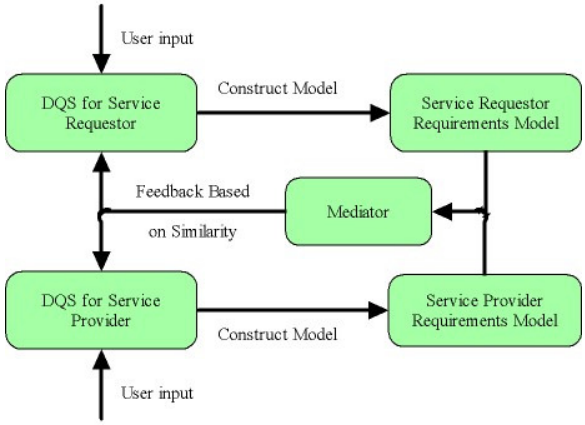


Fig. 1. ASREF Architecture

As shown in **Fig. 1**, *ASREF* is an iterative process. First, rough requirements statements are extracted by a Dynamic Questionnaire System (*DQS*) residing on either service requestors or service providers' site. Then the requirements models, which are based on the Service Requirements

Ontology (*SRMO*), are constructed using a "Natural Language Processing (NLP) based approach. Naturally, there are gaps between the two models constructed separately from the two sides due to their different intentions and starting points. From service providers' view, they have to meet users' needs by adjusting their own model and capability. On the other hand, the requestors' model should be refined to remove the inconsistency, misrepresentation, incompleteness and especially over or under constraints. As a result, a dual adjustment is conducted to bridge the gap between the two models. It is necessary to have a quantitative measurement to evaluate how well the two sides match with each other. When the similarity value reaches a preset threshold, the iteration could be terminated. Otherwise the feedback is provided to the *DQS*, which generates new questions based on it.

DQS includes a set of questions which can be organized into a dynamic questionnaire. The questionnaire based technique is adopted because the face-to-face communication between the requirement engineers and the users is impossible. From answers to this questionnaire, a graphical requirements model describing key elements and structure of each service requirements can be built. This model is based on the Service Requirements Ontology (*SRMO*), which inherits some basic concepts from the agent-oriented requirements modeling framework *i**. Besides requirements modeling, *ASREF* includes a mechanism for improving the quality of models iteratively based on a set of heuristic rules.

The structure of the paper is arranged as the following: Section 2 introduces the goal-oriented ontology for service requirements elicitation *SRMO*, the dynamic questionnaire system *DQS* and how requirements models are built from *DQS* using NLP techniques. Section 3 describes in detail the elicitation mechanism, including the satisfaction measurement of two models, and the feedback system. Section 4 introduces experiments and an example application of *SREM*. Section 5 discusses related work. Section 6 concludes the paper.

2 Services Requirements Model Construction

2.1 Service Requirements Modeling Ontology (SRMO)

Ontology-based approach is widely used to solve problems that require common knowledge and understanding. Concepts, relationships and their categorizations in an ontology can be used as a source of shared knowledge in a specific application domain. Using ontology based approaches, many existing semantic processing techniques can be employed in requirements analysis without getting into rigorous NLP techniques. *SRMO* defines the ontology for service requirements modeling based on concepts of goals and actors, which inherits its key modeling concepts from *i**, and extend it with the concepts of provider actor, requestor actor and services. Providers have capable services, while requestors have required services. Every actor in the service network is motivated to get their required service served [7].

Figure 2 shows the concepts in the proposed service requirements ontology. It models services at a higher level of abstraction than current service ontology such as, *OWL-S*, which focuses more on service design details not accessible to general end-users. The level of abstraction adopted by the proposed mechanism aims to achieve more precise profiling of service requirements. At the same time, *SRMO* are designed to be easily mapped to *OWL-S* description when necessary. Individual service requests can be easily collected and crystallized as requirements knowledge.

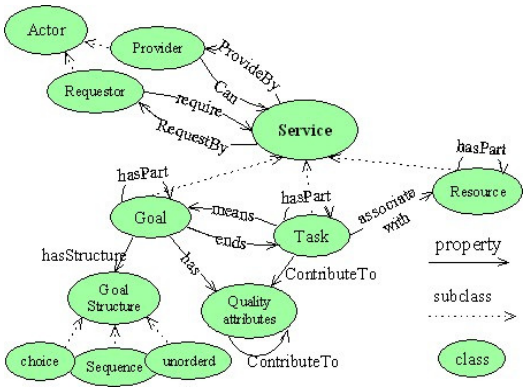


Fig. 2. Service Requirements Modeling Ontology

2.2 Dynamic Questionnaire System

A series of questions and answer schemas are presented to formulate the rough sketch of requirements model. The questions can be organized to 5W2H style [22], that is Who, What, Why, When, Where, How and How well. According their functions, the 5W2H questions can be categorized into 5 series as in table 1:

Table 1. Questions Catalog

Series	Type	Purpose	Sample Questions
S1	Who	Identify actors	Who is involved in the service?
S2	What	Identify goals	What is your goal?
		Identify tasks	What task do you want to perform?
S3	Why	Refine goals upstream	Why is this goal needed?
	How	Refine goals downstream	How can this goal be broken down? How can this goal be achieved?
S4	Why	Refine tasks upstream	Why is this task needed?
	How	Refine tasks downstream	How can this task be decomposed?
S5	How well, When, Where	Identify and refine Quality Attributes	How well do you need this quality attribute to be? Which task influence this quality? Is this quality influence another quality? When and Where do you want the task to be performed?

Series1: This series includes all questions regarding actors. E.g., "Who is involved in the service?" is used to identify actors. "Who is the best one to satisfy your needs?" is used to pick up the user preferred service provider. Answers to these questions can be modeled as *Actors*: 

Series2: This series includes all questions to identify model elements. Answer to "What is your primary goal?" could be a *goal* (

), which state a condition to be satisfied with the help of the

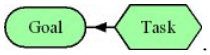
service; answer to "What action do you want to perform?" could be a *task* (

), which specify a procedure of actions to be performed during the service execution; answer to "What resource do you need?" could be a *resource* (

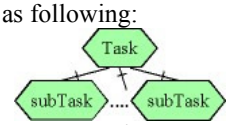
), which is made available by the service.

Series3: Questions in this series are used to refine *goals*. The WHY questions, e.g. "why is this goal needed?" can be used to explore the goal tree upstream, finding out the fundamental intentions of the requestor. On the other hand, the HOW questions can be used to explore goal tree downstream. The question "How can the goal be broken down" decomposes the goal into sub-goals, which is modeled as the following:

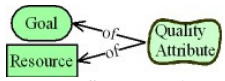
Answers to "How can the state of the *Goal* are achieved?" identify the task achieving a goal. It adds a means-ends link between *Goal* and *task*, *task* is a means for achieving *Goal*, while *Goal* being the end of performing the *task*. We may have various *tasks* for one *Goal*; each *task* stands for one alternative way of achieving this *Goal*. The answers can be modeled as:



Series4: Similar to series3, series4 also includes WHY and HOW questions, but they are used to refine tasks. WHY questions, such as "Why is this task needed?" are used to find high-level tasks or goals, while the HOW questions, for example "How can this task be decomposed?", decompose task into sub-tasks, which can be modeled as following:



Series5: HOW WELL, WHEN, WHERE questions are all fallen into series5. They can elicit requirements on *Quality Attributes,* which are the decision-making rules for evaluating its quality. Answers to HOW WELL questions can be modeled as:



This series also includes other questions to refine *Quality*. For example, how does a specific *task* influence quality of service? How does one quality attribute influence another quality attribute? Answers to these questions can be modeled as follows:

In addition, WHEN/ WHERE questions also contribute to enrich the *Quality Attributes* we extract.

These questions help us collect requirement fragments from requestors and providers. Figure 2 shows an example integrated view for a requirement model constructed from all question answers.

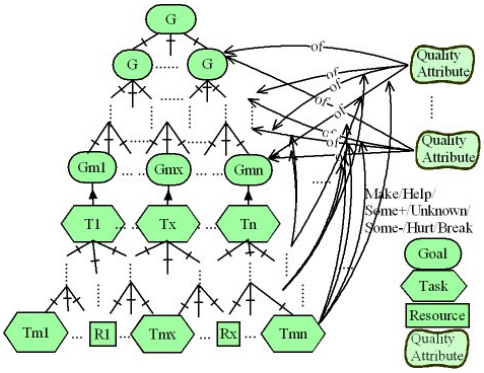


Figure.3. A Hierarchical Requirement Model in SRMO

3 Iteration of Service Requirements Elicitation

This section introduces the iteration process to make the requirements we gathered evolved. As a mediator, *ASREF* supports following functions*:* first, it extracts requirements fragments iteratively from either requestors. Second, it accumulates these requirements to a knowledge repository of graphical goal-oriented models for service providers about the needs and preference of service requestors with regard to the their capable services. Third, it serves as a matchmaker for the two sides. Using knowledge provided by the repository, *ASREF* makes it possible for requestors to find out the services matching their needs as well as for providers to connect to their potential customers. Finally, *ASREF* improves the requirements quality iteratively using feedbacks of matching results. For example, it can elicit hidden requirements which requestors are not aware of, or suggests service providers to adapt their function or quality to fit into popular user needs.

3.1 Satisfaction Measurement

Being a match maker for the two sides, *ASREF* needs a satisfaction measurement. Because the requirements models are based on ontologies, the matching process is an ontology similarity calculation process in nature [25].

In order to get the overall similarity, it is necessary to calculate each element's *MatchScore* separately, including goals, tasks and resources. The *MatchScore* has major two considerations: functional similarity and non-functional similarity. The non-functional similarity *simSG* is captured by following formula,

$$simSG(s,t) = \frac{1}{m}\sum_{i=1}^{m}\frac{Min[Q_i(s),Q_i(t)]}{Max[Q_i(s),Q_i(t)]} \quad . \tag{1}$$

Where s and t are elements being compared; $Q_i(s)$ means value of the i*th* quality attribute of s; m is the maximal number of attributes the two elements have. *MatchScore* is calculated as follow:

$$MatchScore(s,t) = k^n * sim(s,t) * simSG(s,t) \quad . \tag{2}$$

sim(s,t)=1 if existing *t* in service provider which is the same with *s* in service requestor, otherwise

set the value to 0. k^n is a penalty factor which reflect the structure distance. k can be set to a fix number such as 0.9, and n is the level distance of s and t. Combination of $k^n * sim(s,t)$ captures the functional similarity of the two elements. Finally, the overall satisfaction degree can be calculated as follow:

$$satisfaction(S,T) = \frac{\sum_{i=1}^{\#Nodes(S)} MatchScore(S_i, T_i)}{\#Nodes(S)} \quad . \tag{3}$$

Where S and T are models of requestor and provider; *#Nodes(S)* returns the number of nodes in S. The calculate process will be explained in more detail with an example in section 4.

3.2 Reaction rules based on feedbacks

If the satisfaction degree cannot reach the threshold, the iteration will continue. Feedbacks are provided to the DQS, which generate new elicitation questions. In general, there are eight rules including the feedbacks and respective actions taken by DQS. The formal presentation of these rules is given by SWRL language in the OWL file [28].

Rule1: Provider model needs refinement.

Feedback: Certain goals or tasks of service provider are so general that the requestor's model is difficult to match them. Suppose there are model fragments as follows:

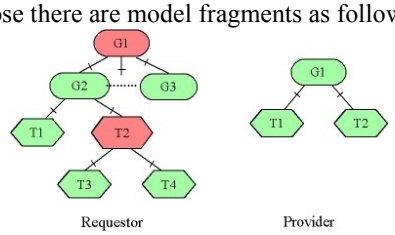


Fig. 4. A provider model needs refinement

It is obvious that the G1 and T2 of provider have not been decomposed enough, so we provide this information to DQS of providers.

Action: With this feedback, DQS should raise more HOW questions for service provider to break down the coarse goals and tasks so that the requestor's model is matched better.

Rule2: Requestor model needs refinement.

Feedback: Contrary to previous class, it is the requestor instead of provider who includes high-level goals or tasks. As shown in the following figure Fig. 5, G1 and T2 of requestor have not been decomposed completely.

Action: DQS should raise more HOW questions for service requestor to break down the particular goals and tasks.

Rule3: Laddering Provider's model.

Feedback: In certain situation, service providers didn't realize that their functional capability can achieve a higher level goal. As shown in following figure, the provider considers that it can only satisfy the goal G2 in the beginning. But after the requestor shows a fact that satisfaction of G2 can lead to the satisfaction of G1, the provider knows he can raise the level of goal G2 to G1. Such information can be provided to DQS of service providers.

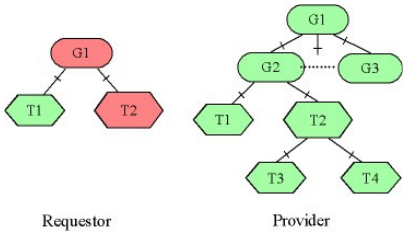


Fig.5. A requestor model needs refinement

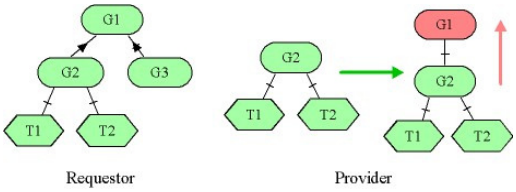


Fig. 6. Low Level of Provider's Goal

Action: More WHY questions should be asked to provider so that exploring the provider's goal tree upstream. By doing this, a service provider can map its core competence into a maximum set of users requirements that can be satisfied by it.

Rule4: Seeking Alternatives.

Feedback: When some functional requirements, modeled as task sub-trees, are not satisfied by provider, it is reasonable to find an alternative option. If a goal of provider is linked by some tasks through means-end relationship, these tasks are replaceable. We can offer this information to improve the satisfaction degree.

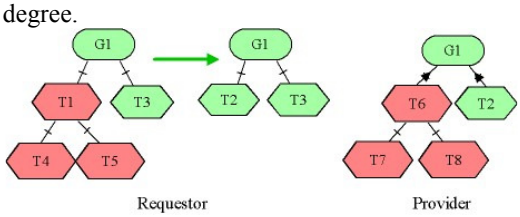


Fig. 7. Alternative Option

Action: DQS asks the requestor whether or not he want to replace the task with another equivalent one. This action increases opportunities of matching the two sides.

Rule5: Adding New Services.

Feedback: If there is no alterative option to choose, and the actions introduced in class 1-4 does not work, we should inform DQS the result.

Action: Suggest the provider to add new services to meet user's requirements, if it is needed. We can measure the worthiness of implementing that service by utilizing the knowledge accumulated, which indicates how often that function is requested.

Rule6: Outsourcing Services.

Feedback: Same as above.

Action: Within a service network, it is possible to use abundant outside services. DQS can advise

the provider to extend its competence by using outsourced services while keeping transparent to end users. This would extremely improve the matching ratio.

Rule7: Quality Conflict Resolution

Feedback: *Quality Attributes* have extremely important impact on matching results. It is common that one task has different types of contribution to the same quality attribute in diverse perspective. As illustrated in following figure, task T1 *helps* the quality attribute of G1 in requestor's view as well as *hurts* it in provider's view. This conflict of quality attributes will be reported to DQS.

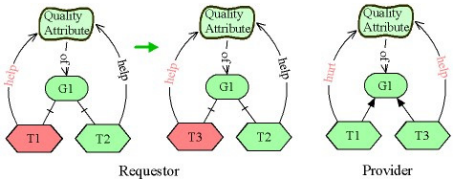


Fig.8. Quality Conflict

Action: Similar with rule 4, DQS suggests that the requestor replace the task with an alternative that helps the quality attribute, say, T3 in this example.

Rule8: Quality negotiation and tradeoff

Feedback: If the quality attributes requested are too difficult to fulfill by provided services, we should take actions to make a trade-off.

Action: DQS should either suggest the requestors to reduce the unnecessary quality requirements, or advise the providers to increase the quality degree they offered.

4 Example Case Study

The proposed approach is applied to a widely referenced example in the domain of e-Tourism, which was originally built as a use case of the *WSMO* project [26]. There is an online travel agency, WorldStar, providing various e-Tourism services to customers, such as booking of flights, hotels, rental cars, etc. Mr. Joe, a customer living in Beijing, wants to enjoy his 10 days vacation in Paris. In order to save time, he decides to employ an online travel agency to deal with travel issues for him. But Joe has not taken an international travel before, so he doesn't know how to express his requirements to travel agencies and which agent meets his needs the best. In this situation, our approach, *ASREF*, is applied to elicit Joe's requirements and build model based on it.

In order to elicit Joe's initial requirements, a set of basic questions are generated by the *DQS*. Due to limitation of space, we skip the step of raising questions, but starts building an initial requirements model from the answers to these questions as in Fig. 9.

Where the meanings of labels are:

G1: One-stop travel services are provided.
G2: Ticket to Paris is booked.
G3: Paris hotel is booked.
G4: Car is rented in Paris.
T1: Book flight ticket from Beijing to Paris.
T2: Book train ticket from Beijing to Paris.
T3: Search possible train connections.
T4: Select one train connection.

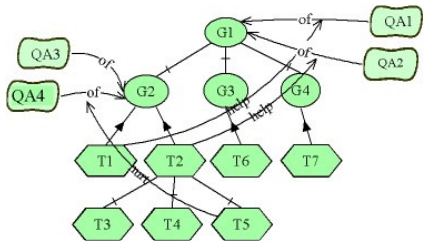


Fig. 9. Initial Model of Joe

T5: Get the ticket.
T6: Book hotel room online.
T7: Rent a Benz car.
QA1: Convenience; **QA2:** Cost;
QA3: Security; **QA4:** Response time

On the other hand, models of travel agencies have been constructed using the same approach when they registered in. Take WorldStar as an example, the Constructed model is shown as follow:

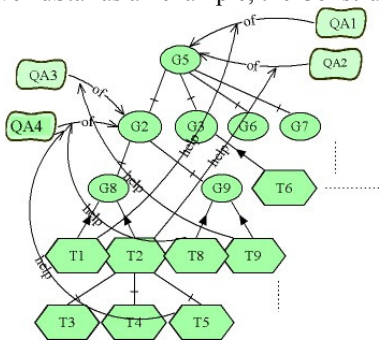


Fig. 10. Initial Model of WorldStar

Beside labels defined above, these labels are added:

G5: Benefits earned by providing travel services.
G6: Travel guide is provided.
G7: Meal plan is provided.
G8: Ticket is booked.
G9: Ticket is delivered.
T8: Deliver electronic ticket.
T9: Deliver physical ticket by express.
T10: Deliver physical ticket by priority express.

ASREF then compares Joe's initial model with all models of current registered travel agencies, and returns a ranking of agency with best matching ratio. To illustrate how to get the satisfaction degree, we calculate it between models of Joe and WorldStar. Suppose all *Quality Attributes* can be expressed by a number, for example, the security level Joe desired can be set to 0.8 as well as the value of WorldStar provided can be set to 1. In addition, the response time Joe desired is 15 minutes, but the WorldStar can only offer it in 30 minutes. Using formula 1 we can get: simSG(G2,G2')=1/2*(0.8/1 + 15/30)=0.65. The functional similarity of G2 can be get by $k^n sim$, where sim(G2,G2')=1 because matched goal G2' exists in WorldStar's model. The penalty factor *k*

is set to 0.9 and *n* is 0 because G2 and G2' is in the same level. *MatchScore* values of all the nodes in Joe's model are calculated as the following table.

Table 2. Matching Ratio Calculate Process

Nodes	*Match Score*	*simSG*	$k^n sim$
G1	0	N/A	0
G2	0.65	0.65	1
G3	1	1	1
G4	0	N/A	0
T1	0.9	1	0.9
T2	0.9	1	0.9
T3	1	1	1
T4	1	1	1
T5	1	1	1
T6	1	1	1
T7	0	N/A	0

According to formula 3, the overall satisfaction degree is (0.65+1+0.9+0.9+1+1+1+1)/11=0.677. If we set the threshold to 0.92, the iteration continues.

We can find the node with low *MatchScore* in Table 2 to locate the problem. Then the heuristic rules mentioned in section 3.2 could be applied to adjust the two models step by step.

Iteration 1: We noticed that *MatchScores* of nodes T1 and T2 in Joe's model are all 0.9 because the value of *n* is 1 in $k^n sim$. It can be easily understood: T1 and T2 are direct children of G2 in Joe's model, but they are grandchildren of G2 in WorldStar's model. So there is 1 level distance between them. The level distance is caused by G8, which isn't in Joe's model, which means G2 has not been decomposed completely. In this situation, we apply Rule2 to refine the service requestor's goal. After received this feedback, *DQS* raises more *HOW*-type questions to break down G2, such as: "In order to get the ticket, what other goals should be achieved?", "Shouldn't the ticket be delivered?" etc. By answering such questions, Joe realizes that he ignored the delivery step. Then our *ASREF* help append G8 and G9 in his model. Due to limitation of space, we skip the process of calculating satisfaction degree at the end of this iteration, and give the value, 0.72, directly. Although the satisfaction degree is improved, it still doesn't reach the threshold. So the next iteration begins.

Iteration2: Because the new goal G9 is added, it's necessary to apply Rule2 again to elicit its means. Suppose question "In which way do you prefer to receive your ticket?" is asked, and the answer is "By priority express". Thus T10 is attached as a means of G9. After such an adjustment, the satisfaction degree is still lower than expected, so the iteration continues.

Iteration3: In the process of re-calculating satisfaction degree, we find that *MatchScore* of T10 just added is zero because there is no corresponding task provided by WorldStar. Joe wants to get the ticket by priority express, but WorldStar only provides delivery services either by normal express or by sending electronic tickets in an email. In order to conduct agreement, Rule4 is applied to recommend Joe to pick up another choice. *DQS* asks "For your requested delivery method is not available, why not try to use an electronic ticket?" If Joe answers "yes", then the T10 in his requirements model is replaced by T8. Again, the process continues.

Iteration4: Improvement of models takes place on both the customer side and the service provider side. In this step, we find *MatchScore* of G4 is zero. It is also because there is no corresponding node in WorldStar's model. Joe wants to rent a car when he arrives at Paris, but WorldStar has not realized

this customer's requirement. Thus both Rule5 and Rule6 can be applied. That means *DQS* can either suggest WorldStar to develop a new car renting service, or outsource this service to a third-party partner. Once such actions are executed, G4 can be attached as a child node of G5 in WorldStar's model. Again, the process continues.

Iteration5: Another node with zero valued *MatchScore* is G1, which can't find any matched goal in WorldStar's model. Joe desires a one-stop travel services, including ticket service, hotel booking service, car renting service, etc. At the beginning, WorldStar didn't offer the car renting service, so it can't be called as "one stop travel services". Thus, G1 is not contained in WorldStar's model. After G4 was added in iteration 4, WorldStar has the capability to provide so called "one stop travel services", so G1 is achievable. The raising of goal level to G1 is based on the Rule4.

Iteration6: According to Table2, G2 got a low *MatchScore* because of the mismatch of its *Quality Attributes*, especially the QA4, response time. Joe requests the ticket should be booked in 15 minutes, but WorldStar only promises to response in 30 minutes. Here Rule8 should be applied to have a negotiation. Depending on different cases, *DQS* either recommends Joe to increase the response time or suggests WorldStar to reduce it. Suppose Joe re-sets this value to 20 and WorldStar changes it to 25, then the result would be quite impressive. The overall satisfaction degree reaches 0.973, which has reached the threshold, so the iteration can be terminated.

Above example demonstrated an example execution scenario for the proposed service requirements elicitation process, including how to generate questionnaire, how to construct requirement models and how to improve these models based on rules step by step.

5 Related Work

Requirements elicitation has focused primarily on developing, implementing, and evaluating various techniques, methods, and tools. Many of these were adopted from other research areas such as the social sciences [2, 4] and knowledge engineering [9, 11]. These approaches reduced the complexity of the elicitation process and improved the quality of the requirements. In fact, there are more than one hundred approaches used for requirements elicitation. [5] has examined a few number of the traditional techniques such as interviewing, observation, and task analysis at a relatively high level. In a more recent survey on the theory and practice of requirements elicitation [4], more approaches were examined including those based on goals [3], scenarios [10], viewpoints [11], and domain knowledge [12].

Our approach can be considered as a natural migration of the goal-oriented requirements elicitation approach into services-oriented computing. In the service-oriented requirements engineering, the elicitation process has to be conducted within the service life-cycle, with assistance of web-based tools. Due to specific requirements of the services context, a questionnaire technique [18] is employed. Satisfaction measurement actually is the semantic similarity of ontologies [20]. Finally, we presented a set of adjust rules to improve the quality of models. Service requirement negotiation [14] servers the same purpose.

There is an early day homonymous requirement methodology *SREM*[1] for large embedded computer systems, descriptions of real world objects, data requirements and message processing. There are also other slightly related works such as [2] which targets at similar problems on collecting users' preferences in order to guide service composition. It suggests user to present a service request with an external e-Service schema of a finite state machine. We consider such work as targeting as similar problem using different conceptual mindsets and treatments.

6 Conclusion

In summary, this paper propose a service-oriented requirements analysis framework ASREF, which turns the classical "once for all" philosophy into a continuous negotiation and adaptation process. Based on existing requirements model and new service request, ASREF aims to achieve an optimal service supply and demand relationship based on dual feedback loop. Current control variable is the similarity of the service requirements model and the service capability model. Major technical contributions include: a dynamic questionnaires system for service requirements elicitation, the construction process of goal-oriented service models based on user's answers to the elicitation questions and the dual feedback control mechanism to service requestor and provider. A virtual travel agency example is used to illustrate the proposed ideas.

In the future, the ASREF service requirements elicitation framework will be integrated with an actual service execution platform SAFARY[27], so that the proposed techniques can be evaluated and improved. The goal-oriented service modelling ontology SRMO will be used in combination with existing service ontologies such OWL and WSMO, so we need to look deeper into extension and integration mechanism of service ontologies.

Acknowledgement

Partial financial support from NSF China (no.60503030), National High Tech. 863 Program (no.2006AA01Z155), National Basic Research and Development 973 Program (no.2002CB312004), and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList) is gratefully acknowledged.

Reference:

1. Alford M.W., Software Requirements Engineering Methodology (SREM) at the Age of Two, COMPSAC78. Proceedings: 332-339, 1978.
2. Berardi D., Calvanese D. etc, Automatic composition of e-services that export their behavior, Proc. Of 1st International Conference on Service Oriented Computing, 2003, pp. 43-58.
3. Dardeene A., van Lamsweerde A., Fickas S., Goal-Directed Requirements Acquisition. Science of Computer Programming 20(1-2): 3-50, 1993
4. Zowghi D., Coulin C., Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In Engineering and Managing Software Requirements, Springer: US, 2005.
5. Goguen J.A., Linde C., Techniques for Requirements Elicitation. International Symposium on Requirements Engineering, 152-164, January 4-6, San Diego, CA, 1993.
6. Hudlicka E., Requirements Elicitation with Indirect Knowledge Elicitation Techniques: Comparison of Three Methods. International Conference on Requirements Engineering, 4-11, April 15-18, Colorado Springs, CO. 1996.
7. Liu L., Chi C., Jin Z., Yu E., Strategic Capability Modelling of Services. The second Workshop of Service-Oriented Computing Consequences and Experience of Requirements, SOCCER 2006, Paris, France.
8. Maiden N.A.M., Rugg G., Knowledge Acquisition Techniques for Requirements Engineering.

Requirements Elicitation for Systems Specification, July 12-14, Keele, UK, 1994.
9. Munindar P. Singh, Michael N. Huhns. Service-Oriented Computing: Semantics, Processes, Agents. Wiley, London, UK, 2005
10. Potts C., Takahashi K. etc, Inquiry-Based Requirements Analysis. IEEE Software: 21-32, 1994
11. Sommerville I., Sawyer P., Viller S., Viewpoints for Requirements Elicitation: A Practical Approach. International Conference on Requirements Engineering, 74-81, April 6-10, Colorado Springs, CO, 1998.
12. Sutcliffe A, Maiden N., The Domain Theory for Requirements Engineering. IEEE Transactions on Software Engineering 24(3): 174-196, 1998.
13. Breitman, K. K. and J. C. S. do Prado Leite., Ontology as a requirements engineering product. Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International: 309-319.
14. Briggs, R. O. and P. Gruenbacher., EasyWinWin: managing complexity in requirements negotiation with GSS. Proceedings of the 35th Annual Hawaii International Conference on System Sciences: 10, 2002.
15. In, H. P., D. Olson, et al., Multi-criteria preference analysis for systematic requirements negotiation. COMPSAC 2002: 887-892, 2002
16. Kaiya, H. and M. Saeki., Using Domain Ontology as Domain Knowledge for Requirements Elicitation, IEEE Computer Society Washington, DC, USA: 186-195, 2006.
17. Kof, L., Natural Language Processing: Mature Enough for Requirements Documents Analysis?, 10th Intl. Conf. on Applications of Natural Language to Information Systems: 15-17, 2005.
18. Moore, J. M. and F. M. Shipman Iii., A comparison of questionnaire-based and GUI-based requirementsgathering, The Fifteenth IEEE International Conference on Automated Software Engineering: 35-43, 2000.
19. van der Raadt, B., J. Gordijn, et al., Exploring Web Services from a Business Value Perspective, Requirements Engineering: 114-134, 2005.
20. Zhang, R., I. B. Arpinar, et al., Automatic Composition of Semantic Web Services. International Conference on Web Services, 2003.
21. Jian X, Lin L, Wei, Q, JingWei, Y., SREM: A Service Requirements Elicitation Mechanism based on Ontology. COMPSAC 2007, to appear.
22. Julio Cesar Sampaio do Prado Leite, Yijun Yu, Lin Liu, Eric S. K. Yu, John Mylopoulos: Quality-Based Software Reuse. CAiSE 2005: 535-550.
23. FernÃ¡ndez-LÃ³pez, etc: "Methontology: from ontological art towards ontological engineering". "Proc. Symposium on Ontological Engineering of AAAI". 1997.
24. Leite, J.C.S.P., Franco, A.P.M.v. A Strategy for Conceptual Model Acquisition. Proceedings of the First International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 243-246, (1993).
25. B. Aleman-Meza, C. Halaschek-Wiener, S. S. Sahoo, A. Sheth, I. Budak Arpinar. Template Based Semantic Similarity for Security. Applications Technical Report, LSDIS Lab, Computer Science Department, University of Gerogia, January 2005.
26. WSMO's use case (VTA): http://www.wsmo.org/2004/d3/d3.3/v0.1/.
27. J. Xiang, W. Qiao, Z.. Xiong, T. Jiang, L. Liu. SAFARY: A semantic web service implementation platform. In: Proceedings of APSEC-SOPOSE'06. Bangalore, India, December 9, 2006
28. ASREF OWL file: http://learn.tsinghua.edu.cn:8080/2005212713/ASREF.ow