

**Master Thesis**

**Supporting Multilingual Forum  
by Language Service Composition**

Supervisor    Professor Toru Ishida

Department of Social Informatics  
Graduate School of Informatics  
Kyoto University

**Shunsuke TANIDUKA**

February 8, 2007

## **Supporting Multilingual Forum by Language Service Composition**

Shunsuke TANIDUKA

### **Abstract**

Many international forums are held by various communities such as NPOs nowadays. However, language barrier remains as an obstacle in these forums since participants' mother languages are different from one another. Many communication support tools have been developed and tested to support international forums by incorporating these tools with machine translations. However, it is difficult to translate their conversations using general machine translations since participants often use community-specific terms in these multilingual forums.

In the midst of such phenomena, the Language Grid is being developed. The Language Grid enables people to use language services easily and provides a framework that allows people to add new language resources and language services to the Grid. People can create new language services which are specialized to the community by connecting various language resources and language services using Web service workflow.

This research aims to support the creation of Web service workflow of language services specialized to the community. At present, automatic Web service composition is costly, so it is not suited for general use. Given this factor, we acquire user requested Web service workflow by tuning basic workflow (which is the building block of composite Web services).

To tackle this, we need to address the following issues.

- The structure of the workflow affects the service quality. So even if the content of the composite services are the same as a whole, their service quality can differ from one another since their structures differ from one another. We can improve the service quality by modifying the structure of the workflow. We call this *tuning* in this thesis. Until now professionals have tuned workflows using technical know-how. In this way, only specified professionals can use effective tuning for specified workflow. So we have to formalize tuning method as rules and make it

generally available.

- Many resources, i.e. basic workflows and tuning rules, are needed to realize the tuning of the basic workflow that we propose in this thesis. So the architecture to easily create and add resources is needed. We also need to satisfy the constraint of the community to the maximum in the process of workflow creation. Therefore we need to design architecture that satisfies the above two requirements.

To solve these two issues, we provide the following solutions.

### **(1) Tuning by operator**

Firstly, we formalize workflow tuning method as rules. Secondly, we define tuning operators as the unit of operation to describe the tuning rules. Defining of the operators places the user and the system on the common ground to operate the workflow and to allow mutual conversion of their operations. When tuning the workflow, there are plural points to modify. Modification depends on the former one, so the order of modifying affect the latter modification. We solved this issue by introducing the idea of *least commitment planning* in AI planning to the workflow.

### **(2) Interactive workflow generating system**

We propose an interactive workflow generating system. This system realizes the creation and addition of the resources to the system during workflow creation. The addition of the basic workflow to the system enables the abstraction and registration of the created workflow to the system. We also realize the acquisition of tuning rules from a series of user's manual operations on the workflow. By this, we can use rules which are not only described by professionals but also acquired from user's manual operations.

Next, to satisfy the constraint of community to the maximum, we allow users to intervene anytime in the process of workflow creation. User can not only provide necessary information to create workflow but also create workflow by maintaining mutually complementary relationship with system. This enables the creation of high-quality workflow. In the tuning process, user's intervention is equal to the operation on workflow. Therefore, the system can acquire the above tuning rules from user's series of operations.

## 言語サービスの連携による多言語フォーラムの支援

谷塚 俊輔

### 内容梗概

現在、NPO など様々なコミュニティが国際的なフォーラムを開き、議論を行っているが、これらのフォーラムにおいて参加者の言語の違いによるコミュニケーションの問題は依然として残っている。様々なコミュニケーション支援ツールが開発されてきており、機械翻訳を組み合わせることによってフォーラムを支援する試みが行われている。しかし、多言語フォーラムでは、コミュニティ特有の言葉が頻繁に利用されるために一般的な機械翻訳では翻訳が困難である。

言語サービスを容易に利用でき、またユーザ自らが新たな言語資源や言語サービスを追加できる仕組みを提供する言語グリッドプロジェクトが進められている。様々な言語資源・言語サービスを、Web サービスワークフローを用いて連結することでコミュニティに特化した言語サービスの作成が可能となる。

本研究ではコミュニティに特化した言語サービスの Web サービスワークフロー生成の支援を行う。複合サービスの自動連携はコストが高く、まだまだ一般的な利用には適していない。そこで、本研究ではワークフローの生成方法としてベースとなるワークフローをチューニングすることでユーザの求めるワークフローを獲得するという手法をとる。

取り組むべき問題は以下の 2 点である。

- **Web** サービスワークフローを作成する際に、複合サービス全体としては同じ内容のサービスであっても、ワークフローの構造によってサービス品質に差異が生じる。この構造を変更することでサービス品質を向上させることをチューニングと呼ぶ。これまでチューニングはワークフローを作成する際に専門家がノウハウを利用し、探索的に行ってきた。しかし、これでは有効なチューニングが特定のワークフローで、特定の専門家にのみ利用されるという問題がある。チューニングの手法をルールとして形式化し、汎用的な利用を可能とする必要がある。
- 本研究で用いるベースとなるワークフローをチューニングする手法を実現するためには、ベースとなるワークフロー、チューニングの手法などの資源が豊富である必要があり、資源の追加が容易に行われるアーキテクチャが求められる。また、それと同様にワークフローの生成過程においては、コミュ

ニティの制約が常に最大限に満たされる必要があるため、これらの 2 点を満たすようなアーキテクチャが求められる。

上記の問題に対し、以下のようなアプローチをとった。

### (1) オペレータを用いたチューニング

ワークフローのチューニング手法をルールとして形式化を行った。次に、チューニングルールを記述する際の操作の単位としてチューニングオペレータの定義を行った。オペレータを定義することで、ユーザとシステムがワークフローに対して可能な操作を共通化し、それぞれの操作を容易に相互変換することが可能となった。オペレータを定義する際に、AI プランニングにおける **least commitment planning** の考えを導入した。ワークフローのチューニング時には、複数の修正箇所が存在する場合がある。修正操作は前の操作に依存するため、修正箇所の選択の順序は後の操作に大きく影響する。**Least commitment planning** の考えをワークフローに導入することで、この問題を解決した。

### (2) 対話型ワークフロー生成システムの考案

ワークフローの生成過程において資源の生成・追加が行われる対話型ワークフロー生成システムのアーキテクチャの考案を行った。ベースとなるワークフローの追加は、生成プロセスの結果作成されたワークフローを抽象化し、システムに登録するというサイクルを実現する。また、チューニングルールの追加に関しては、ルールとして専門家が記述したものを蓄積・利用するだけでなく、ユーザが手動で行ったワークフローに対する操作系列からルールを獲得することを実現した。

次に、コミュニティの制約を最大限に満たすためにワークフローの生成過程において常時ユーザの介在を認めた。ユーザはワークフロー生成に必要な情報をシステムに提供するだけでなく、システムと対話的に相互補完の形でワークフロー生成のプロセスを行い、探索的に高品質のワークフローを作り出すというプロセスが実現された。チューニングプロセスにおけるユーザの介在はワークフローに対する操作であり、その操作系列から先述のルール獲得が可能となる。

# Supporting Multilingual Forum by Language Service Composition

## Contents

<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Supporting Multilingual Forum</b>	<b>4</b>
2.1 Language grid.....	4
2.2 Fundamental Tools for Supporting Multilingual Forum.....	5
2.3 Supporting Multilingual Forum by Web Service Composition.....	5
<b>Chapter 3 Approach to Improve the Quality of Composite Service</b>	<b>8</b>
3.1 Improvement of Workflow by Tuning .....	8
3.1.1 Community Specialized Translation.....	8
3.1.2 Tuning Example.....	11
3.2 Abstract Workflow .....	16
3.3 Tuning Operator .....	17
3.4 Tuning Rule .....	22
3.4.1 Formalization of Tuning .....	22
3.4.2 Description Example .....	23
<b>Chapter 4 Architecture</b>	<b>26</b>
4.1 Use Case .....	26
4.2 Interactive Workflow Generating System .....	27
4.3 Processing Flow .....	32
<b>Chapter 5 Implementation</b>	<b>35</b>
5.1 System Function.....	35
5.2 WFGenerator .....	38
<b>Chapter 6 Discussion</b>	<b>41</b>
<b>Chapter 7 Conclusion</b>	<b>43</b>
<b>Acknowledgments</b>	<b>45</b>
<b>References</b>	<b>46</b>

# Chapter 1 Introduction

Many international forums are held by various communities such as NPOs nowadays. However, language barrier remains as an obstacle in these forums since participants' mother languages are different from one another. Many communication support tools have been developed and tested to support international forums by incorporating these tools with machine translations. However, it is difficult to translate their conversations using general machine translations since participants often use community-specific terms in these multilingual forums.

There are various language services such as machine translations and dictionaries on the internet today. However, each service is independent and the environment that end user can use these services easily is not established. To solve these issues, the Language Grid[1] is being developed. The Language Grid enables people to use language services easily and provides a framework that allows people to add new language resources and language services to the Grid.

The Language Grid has two different goals, the horizontal Language Grid and the vertical Language Grid. The horizontal Language Grid has a goal to combine standard language services. The vertical Language Grid has a goal to make language resources which are mainly used in the specified community Web services and use these services. We can solve the difference of the mother language and support intercultural collaboration by combining the horizontal Language Grid and the vertical Language Grid.

We can create new community specialized language services by connecting various language resources and language services which are provided by the Language Grid with the Web service workflow. There are many editors to describe workflow. When using these editors, user needs to know the order of tasks clearly. Therefore the provision of editors does not support the creation of the workflow enough. The workflow has not only to satisfy user's requirement for the service but also to achieve high quality. Especially, the quality of the service is very important for language services.

This research aims to support the creation of the high-quality Web service workflow that is specialized to the community. At present, automatic Web service composition is costly, so it is not suited for general use. Given this factor, we acquire user requested Web service workflow by tuning basic workflow (which is the building block of composite Web services).

The structure of the workflow affects the service quality. So even if the content of the composite services are the same as a whole, their service quality can differ from one another. We can improve the service quality by modifying the structure of the workflow. We call this *tuning* in this thesis.

In this research, we need to address the following issues.

- Until now professionals have tuned workflows using technical know-how. In this way, only specified professionals can use effective tuning for specified workflow. So we have to formalize tuning method as rules and make it generally available.
- Many resources, i.e. basic workflows and tuning rules, are needed to realize the tuning of the basic workflow that we propose in this thesis. So the architecture to easily create and add resources is needed. We also need to satisfy the constraint of the community to the maximum in the process of workflow creation. Therefore we need to design architecture that satisfies the above two requirements.

To solve these two issues, we provide the following solutions.

### **(1) Tuning by operator**

Firstly, we formalize workflow tuning method as rules. Secondly, we define tuning operators as the unit of operation to describe the tuning rules. Defining of the operators places the user and the system on the common ground to operate the workflow and to allow mutual conversion of their operations.

### **(2) Interactive workflow generating system**

We propose an interactive workflow generating system. This system realizes the creation and addition of the resources to the system during workflow creation. The addition of the basic workflow to the system enables the abstraction and registration of the created workflow to the system. We also realize the acquisition of tuning rules from a series of user's manual operations

on the workflow. We can use rules which are not only described by professionals but also acquired from user's manual operation by this.

Next, to satisfy the constraint of the community to the maximum, we allow users to intervene anytime in the process of workflow creation. User can not only provide necessary information to create workflow but also create workflow by maintaining mutually complementary relationship with the system. This enables the creation of the high-quality workflow. User's intervention is equal to the operation on the workflow in the process of the tuning. Therefore, the system can acquire the above tuning rules from a series of user's operations.

This thesis is organized as follows: Chapter 2 describes supporting multilingual forum. Chapter 3 describes workflow tuning which is the method to improve service quality of composite service. Chapter 4 describes interactive workflow generating system. Chapter 5 describes the implementation of the system which is described in Chapter 4. Chapter 6 describes discussion. Chapter 7 describes summary.

## Chapter 2 Supporting Multilingual Forum

Firstly, this chapter describes the fundamental technology that is used for supporting multilingual forum by language service composition. Secondly, the fundamental tools for supporting multilingual forum by language service composition and the approach for supporting multilingual forum by Web service composition are presented.

### 2.1 Language grid

The Language Grid[1] is being developed. The Language Grid is a framework to increase the accessibility and the usability of language services and enable users to create new language services easily by combining appropriate language services.

The Language Grid has two different functions as shown in Figure 1, the horizontal Language Grid and the vertical Language Grid. The horizontal Language Grid connects existing language services in order to cover standard national languages. Those services are often created by linguistic professionals. Typical examples are dictionaries and translation services. To support intercultural activities in community, the vertical Language Grid creates new language services.

It is the most important to connect language services which are developed by

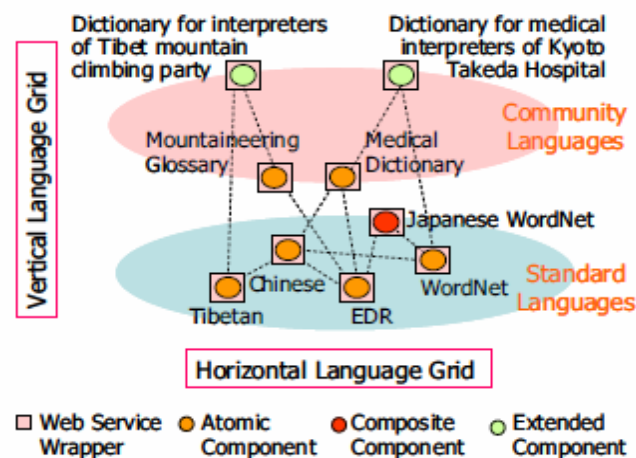


Figure 1: Language Grid[1]

various providers and developers in the Grid. Language services are implemented as Web services. Web services communicate one another with SOAP (Simple Object Access Protocol) message according to the interface that is defined by WSDL (Web Service Description Language). In addition, there are workflow description languages such as BPEL4WS[7] (Business Process Execution Language for Web Services) in order to connect Web services. The Language Grid connects language services with these technologies.

## **2.2 Fundamental Tools for Supporting Multilingual Forum**

Until now various tools for supporting multilingual communication have been developed. AnnoChat<sup>1)</sup> is a multilingual chat tool. This translates messages automatically by using machine translations. This tool enables users to communicate one another with their mother languages. Spark2<sup>2)</sup> is an idea generation support system and corresponds with network. This also enables users whose native languages are different from one another to communicate with their mother languages by incorporating machine translations.

These tools are effective for supporting multilingual forum. However, the forum is the field that people who belong to the community communicate with one another. So community-specific terms are often used. For this reason, general machine translation can not translate sufficiently. In addition, the literature [4] shows that the asymmetric nature of machine translations interrupts the communication in the machine translation mediated communication. So we also need to solve this issue.

From these requirements, the provision of the community specialized translation services can support multilingual forum. In this research, we focus on this approach.

## **2.3 Supporting Multilingual Forum by Web Service Composition**

To support multilingual forum, we adopt the approach to connect language

---

<sup>1)</sup> <http://yoshino.sys.wakayama-u.ac.jp/spark/?lng=ja&page=AnnoChat>

<sup>2)</sup> <http://yoshino.sys.wakayama-u.ac.jp/spark/?lng=ja&page=SPARK2>

resources and language services which are wrapped as Web services and create community specialized translation services. The Language Grid provides language services and Google and Amazon have begun to release Web service API. Then Web services are spreading widely. Given this factor, many researches about Web service composition have been studied in order to create composite Web services by incorporating multiple Web services. Web service composition is the creation of the sequence of multiple Web services, that is to say, the workflow.

Web service composition is classified to the vertical composition and the horizontal composition [2]. The vertical Web service composition is the search of the workflow in order to satisfy user's requirement assuming that each task is assigned unique concrete service. About this, many approaches that are based on the AI planning such as SHOP2 [3], which is a typical HTN-planner, have been proposed. SHOP2 transforms the Web service description that is described in OWL-S to the domain knowledge of SHOP2 and plan to the goal with this knowledge. In addition, the literature [6] has tried to semi-automatic composition. This supports the vertical Web service composition with the Web service description that is described in OWL-S.

The horizontal Web service composition is the selection of the concrete Web services. This selects concrete Web services to each task and searches the most appropriate combination of concrete Web services for the abstract workflow that is created preliminarily in order to satisfy the constraint. About this, the literature [2] regarded Web services that were assigned to each task and as variables and formalized Web service composition as the constraint satisfaction problem.

In the Web service composition of the translation service, the vertical composition corresponds to the constraint of the pair of the source language and the target language, and the horizontal composition corresponds to the assignment from multiple language resources and language services to the task on workflow.

We target users who have no professional knowledge about workflow in this research. The vertical Web service composition needs professional knowledge.

So the vertical Web service composition is not appropriate for users who are targeted in this research. In addition, there are many editors to describe workflow. However, they can not support the creation of the high-quality workflow enough. This is because users have to know the order of tasks clearly when they use them.

The vertical Web service composition is similar to the automatic planning in that they search the processing procedure of tasks. In planning research, the approach that discovers the trouble spot in existing plan and repairs it under the constraint and creates better plan has been studied [11].

We focus on this and introduce the repair method in planning to Web service composition in this research. This enables us to repair basic workflow and acquire better one. We repair the workflow in order to improve the quality of the composite service in the Web service workflow. We call this repair *tuning*. We use a typical abstract workflow and modify it arbitrarily in the workflow tuning. So we do not need to treat the constraint between tasks of the workflow strictly. In addition, it is easy for user to create a workflow since they can refer the basic workflow.

# Chapter 3 Approach to Improve the Quality of Composite Service

There are two versions of the Web service. One is the atomic Web service and the other is the composite Web service. The quality of the atomic Web service depends on the instance. The quality of the composite Web service depends on two versions furthermore. The first is the quality of the structure of the abstract workflow. The second is the quality of the instances that are assigned to each task to organize the abstract workflow.

The quality of the composite Web service depends on the quality of the abstract workflow. So we can improve the quality of the service by changing the composition of the abstract workflow. We call this *tuning* and focus on this.

In this chapter, firstly the improvement of the quality of workflow by tuning is presented with an example. Secondly, tuning operators which are the unit of operation of tuning and tuning rule is presented.

## 3.1 Improvement of Workflow by Tuning

### 3.1.1 Community Specialized Translation

Recently there are many machine translations<sup>1)</sup> which are available on the internet. These machine translations are aimed for translating general sentences. On the other hand, there is a community oriented machine translation service<sup>2)</sup>. This is aimed for translating community-specific sentences by registering the community-specific terms that is difficult to translate by general machine translations.

One approach to translate community-specific sentences appropriately is to extract community-specific terms in sentences and translate them by parallel dictionary and the rest of sentences by general machine translation.

---

<sup>1)</sup> <http://honyaku.yahoo.co.jp/>

<sup>2)</sup> <http://www.yakushite.net/cgi-bin/WebObjects/YakushiteNet.woa/wa/main>

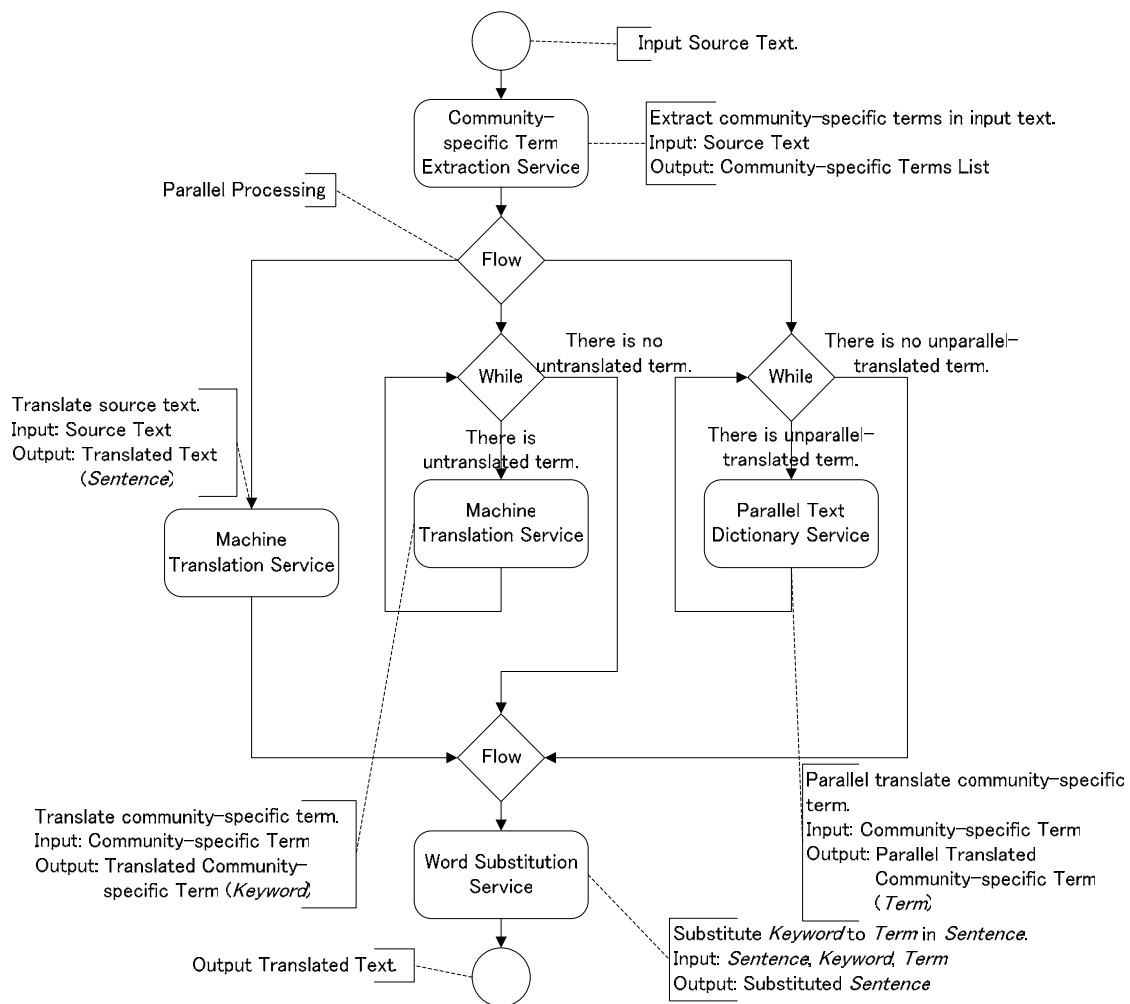


Figure2: Substitute Translation Workflow

According to this approach, we call the translation to replace terms in the translated sentence to the right word of parallel dictionary *substitute translation*. The workflow to execute substitute translation, which is described in BPMN, is shown in Figure 2.

Services that compose the workflow of Figure 2 are following 4 services.

- Community-specific Term Extraction Service

Service content: Extraction of the community-specific terms in the source text.

Input: Source text

Output: Community-specific terms list in the source text

- Machine Translation Service

Service content: Translation of the source text.

Input: Source text

Output: Translated text

- Parallel Text Dictionary Service

Service content: Translation of the community-specific term.

Input: Community-specific term

Output: Translated community-specific term

- Word Substitution Service

Service content: Substitution from words (Keywords) to other words (Terms) in the text.

Input: Text, Keywords, Terms

Output: Substituted text

The processing procedure of Figure 2 is following.

1. Extract the community-specific terms in the source text.
2. Parallel process following 3 processes.
  - (a) Translate the source text.
  - (b) Translate the community-specific terms.
  - (c) Translate the community-specific terms.
3. Substitute output of 2- (b) to output of 2-(c) in output text of 2-(a).

This processing procedure enables us to translate the text that includes the community-specific terms appropriately. The literature [4] saw the asymmetric nature of the communication in the machine translation mediated communication as a problem. We can assure the symmetric nature of the communication only about community-specific terms by using this substitute translation method.

In the fact, however, it happened that we assigned concrete Web services to tasks of this workflow but community-specific terms in the text did not substituted appropriately. The cause is that in the case of word *A* in sentence, machine translated result of *A* is word *B*, in the case of word *A* in word,

Table 1: Parallel Dictionary

Japanese	English
モノ	MONO

Table 2: Machine Translation

Japanese	English
モノ	thing
これは私のモノです.	This is my article.

machine translated result of  $A$  is word  $C$ . For example, we assume that we want to translate community term “モノ” in Japanese to “MONO” in English. Word pair of parallel translation dictionary is described in Table 1. The machine translation translates “モノ” in Japanese to “thing” in word and translates “モノ” in Japanese to “thing” in sentence such as “これは私のモノです.” in Japanese. As a result, we want to substitute “thing” in sentence to “MONO”, but “thing” can not be substituted because there is no “thing” and only “article”.

Moreover, the workflow that is shown in Figure 2 tries to substitute *Keyword* to *Term*. However, the machine translation returns a word as a sentence that includes period. So the workflow often can not find out the place to be substituted and can not execute substitution well.

To solve these problems and improve the quality of the service, we tune the workflow.

We describe an example of tuning and the effect that tuning affects the quality of service in the next section.

### 3.1.2 Tuning Example

The preceding section described an example of the Web service workflow that realizes the community specialized translation service and problems of this workflow. Examples of tuning to solve these problems are following.

1. Multi-hop substitution using intermediate language

Firstly, this tuning substitutes community-specific terms in the source text to intermediate languages and translates the result of the substitution using a machine translation. The machine translations can not translate intermediate languages, so the substitution service can find out the places of the community-specific terms in the translated text surely. Secondly, this tuning substitutes the places of intermediate languages in the translated text to the right words that are acquired from

the parallel text dictionary. We can correspond to fuzzy results of machine translations according to this processing procedure..

2. Lower-casing the result of machine translations

Machine translations return sentences, so the head of the sentences is upper case. This can interrupt the substitution of words. This tuning solves this problem by lower-casing the result of machine translations.

3. Sentence casing the result of the community specialized translation

The community specialized translation service that is created by the workflow translates as a sentence, so this service needs to upper-case the head of the translation result.

Figure 3 shows the workflow that the above 3 tuning rules are applied to the workflow of Figure 2.

Next, we compare the qualities of the before tuning workflow and the after tuning workflow. We evaluated the qualities of them with 12 test data which are created for the test of the community specialized translation service for facilitator's communication of NPO Pangaea<sup>1</sup>.

There are many methods to evaluate the quality of the translation such as BLEU[10]. However, BLEU needs the right translation result. We target the creation of the translation services that are customized for the each community in this research. So BLEU is not appropriate for the tuning that we propose because it is costly and difficult to create test data sets and right translation results for the each community. For this reason, we need automatic evaluation method that does not need right translation results.

Consequently we use following 2 evaluation criteria in this research.

- The rate of the community-specific terms translation

The rate of the community-specific terms translation is represented as following formula (1). In this formula,  $C$  means the rate of the community-specific terms translation,  $NRC$  means the number of the right translated community-specific terms and  $NC$  means the number of the community-specific terms in the text.

---

<sup>1</sup>) <http://www.pangaeaan.org/>

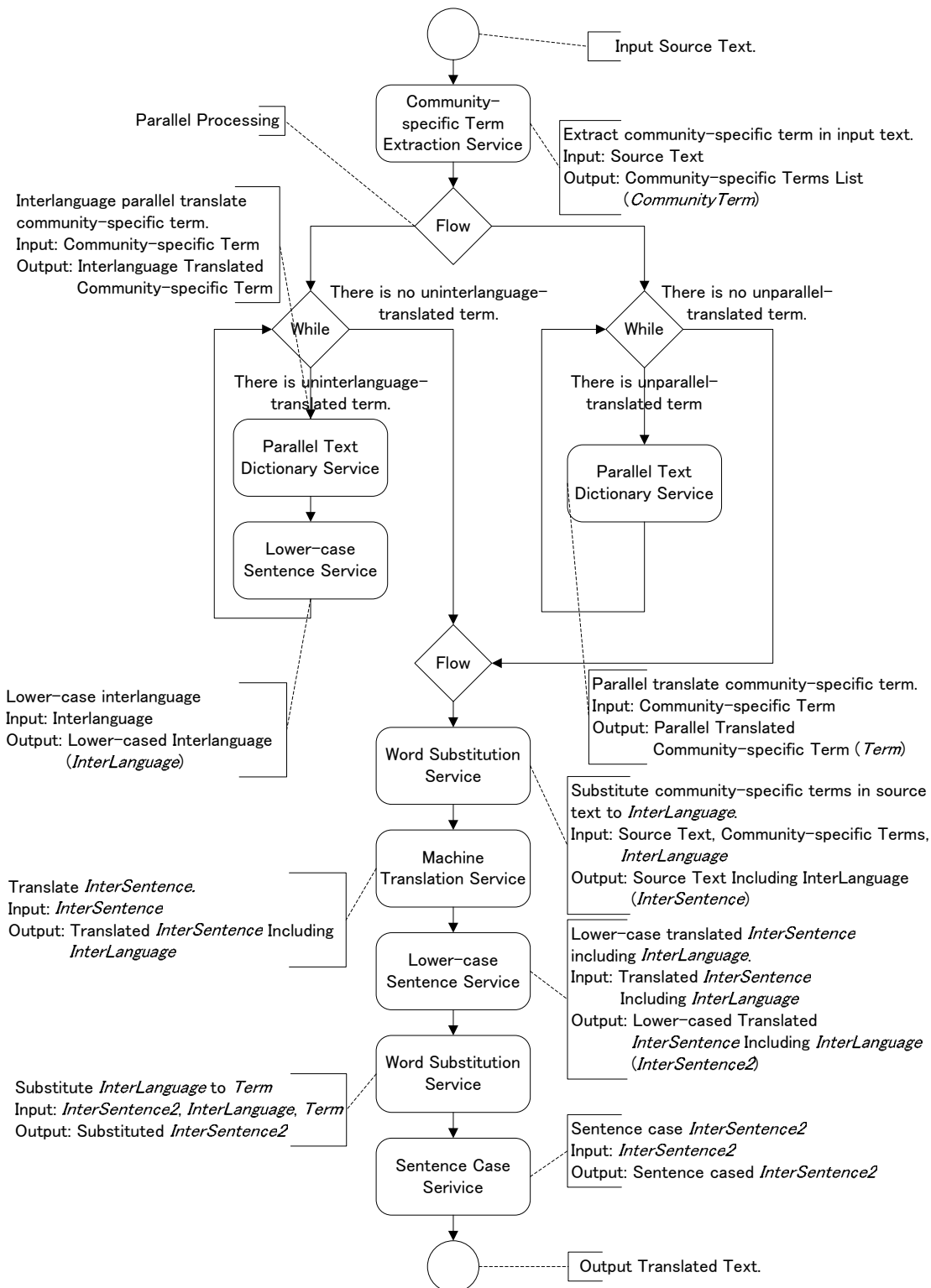


Figure3: Substitute Translation Workflow after Tuning

Table3: Evaluation of the Quality of Translation

		WF1	WF2
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data1	Similarity (%)	10.86242	13.08999
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data2	Similarity (%)	0.00000	0.00000
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data3	Similarity (%)	0.00000	0.00000
Test	Rate of the community-specific terms translation (%)	33.33333	100.00000
Data4	Similarity (%)	0.00000	0.00000
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data5	Similarity (%)	39.60656	28.40593
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data6	Similarity (%)	0.00000	0.00000
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data7	Similarity (%)	0.00000	0.00000
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data8	Similarity (%)	0.00000	28.00810
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data9	Similarity (%)	13.50059	13.50059
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data1	Similarity (%)	0.00000	0.00000
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data1	Similarity (%)	42.21823	42.21823
Test	Rate of the community-specific terms translation (%)	0.00000	100.00000
Data1	Similarity (%)	49.39585	58.31881
	Average rate of the community-specific terms translation (%)	2.77778	100.00000
	Average similarity (%)	12.96530	15.29514

$$C = \frac{NRC}{NC} \quad (1)$$

We think that the community-specific terms are more important than general words in the communication of the community. For this reason, we introduce this evaluation criterion.

- The similarity between input test data and back translation result

When creating translation service, we create the back translation service at once. This evaluation criterion is to compare the similarity between the input source text of the translation service and the output text of the back translation service. For example, when creating the Japanese-English translation service, we create the English-Japanese translation service at once. We introduce the method of the literature [5] to compare the similarity. The Literature [5]

Table4: Test Data1

			WF1	WF2
Test Data1	アナログ作業のときの参加者の配置はどうしたらいいですか？	Back translation result	どのように、アナログ仕事である時にすべきであるか 参加者の配置されるか？	どのように、アナログ仕事である時にすべきであるか 参加者の配置されるか？
		Similarity(%)	10.86242	13.08999

Table5: Test Data5

			WF1	WF2
Test Data5	パンゲアネットに入り、地球、国、村とたどることで、参加者の家に行けませす。	Back translation result	パンゲアネットに入り、地球、国、および村によって引き返すことによって参加者の家に行くことができる。	パンゲアネットに入り、地球、国、および村によって引き返すことによって参加者における家に行くことができる。
		Similarity(%)	39.60656	28.40593

compares the similarity between the back translation result and the input text, the result of BLEU and evaluated value of personal subjectivity and shows the effectiveness.

We evaluated 2 translation services which were created by WF1 (the workflow that is before tuning, described in Figure2) and WF2 (the workflow that is after tuning, described in Figure3) using above 12 test data by the above 2 evaluation criteria. This evaluation is showed in Table 3. Some test data that are showed that WF1 was better than WF2 about the similarity. But the evaluation as a whole was that WF2 was better than WF1 by 3 percent on average. Almost all the community-specific terms were not translated in WF1 and only 3 percent of terms were translated about the rate of the community-specific terms translation. WF2 translated all the community-specific terms of 12 test data appropriately.

We discuss the result of Test Data1 that the difference of the similarity between WF1 and WF2 is close to the difference of the average similarity and Test Data5 that WF1 is better than WF2 in the similarity in terms of the concrete example sentence of the test data and the result of the back translation further. Table4 shows an example sentence, the result of back translation and the similarity of Test Data1. Table5 shows them of Test Data5. In addition,

underlined parts of example sentences show community-specific terms.

### **Test Data1**

Comparing results of back translation of WF1 and WF2, the 3% difference of the similarity was caused by the difference of the number of well back translated community-specific terms.

### **Test Data5**

The community-specific terms of this example sentence were not translated as the community-specific terms, but back translation result was the same as one of WF2 in WF1. In addition, machine translations could not execute preposition analysis appropriately in WF2 because WF2 uses the intermediate language. For these reasons, the similarity of WF2 is less than one of WF1.

From the above results, we found that these tuning could translate community-specific terms surely but the quality of translation decreased in some example sentence since the intermediate language lacks the part of the speech information and machine translations can not translate sentences appropriately. The effect that the part of the speech information affects the quality of the machine translation was less on average. So the average similarity of WF2 is higher than one of WF1. Other tuning can solve these problems that this tuning can not solve.

From these, we found that tuning can increase the quality of the community specialized translation service.

## **3.2 Abstract Workflow**

The preceding section described the workflow tuning with examples. This section describes the abstract workflow that is an object of tuning. The abstract workflow is the workflow that the content of each task and the sequence of the execution are decided and concrete Web services are not assigned to any tasks. The abstract workflow is executed when concrete Web services are assigned to its all tasks.

There are many workflow description languages. We use BPEL4WS[7] in this research. BPEL4WS is the workflow description language based XML. We can create complicated process flow by incorporating the invocation of the Web

service and the manipulation of data with BPEL4WS. The element that organizes the workflow is called activity in BPEL4WS and various activities are defined. We select activities which are used for the general creation of the workflow and classify them as below in this research.

### **Activity**

- invoke: the activity that invokes Web service

### **Assignment**

- assign: the activity that executes data passing between variables

### **Structure**

- sequence: the control activity that executes child activities sequentially
- switch: the control activity that executes conditional branching for child activities (case or otherwise activity)
- while: the control activity that executes child activities iteratively
- flow: the control activity that executes child activities parallel

In addition to the above activities, we call the element that organizes the abstract workflow task and call the relationship between tasks edge. We formalized the abstract workflow (AWF) as below with these activities.

$$\text{AWF} = (\text{TA}, \text{ED})$$

- TA = (AC, AS, ST) : a set of tasks
  - AC: a set of Activity
  - AS: a set of Assignment
  - ST: a set of Structure
- ED: a set of edges between elements of TA

This formalization enables us to treat tuning as operation such as add, delete and modify for the elements of each set of AWF.

Next section describes operators which are the unit of tuning operations.

## **3.3 Tuning Operator**

The tuning for the workflow consists of combinations of various operations. If there are operators for the unit of tuning operations, we can describe tunings as rules with a set of operators. We think that there are following two tuning patterns as shown in Figure 4.

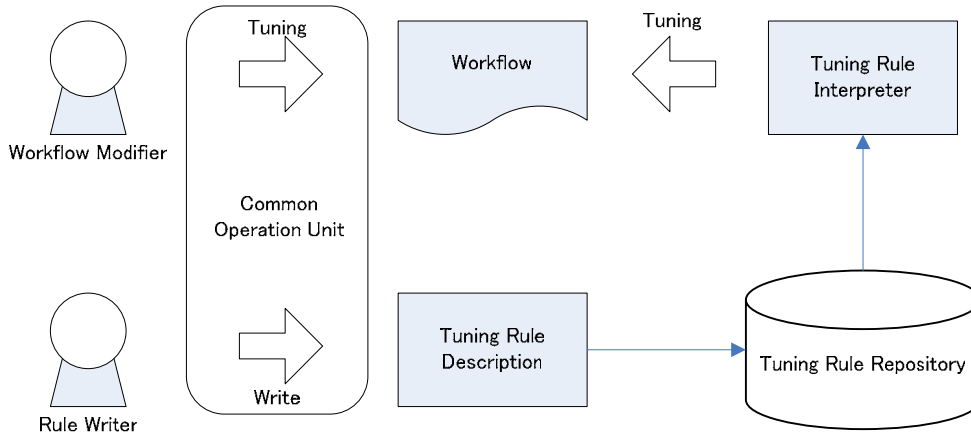


Figure 4: Tuning for Workflow

- Tuning that is described preliminarily.
- Tuning that user execute manually for modifying workflow.

The definition of operators that are the unit of common operations for the above 2 tuning patterns enables us to convert them mutually. This can enables us to acquire rules from user's manual operations.

We discuss the requirement for the unit of the operation in terms of the tuning description. Firstly, we need to allow user only to operate relatively for tasks of workflow. If we allow user to operate any tasks to any places in the workflow, it is difficult to assure the validity of the tuned workflow. Secondly, we need to reduce user's load for the procedure of operations. Tuning operations are executed sequentially and the former operation can complicate the latter operation. In this case, user has to rewrite tuning.

The literature [8] proposed operations for the workflow. But these operations did not satisfy the above requirements.

So we introduce the method of least commitment planning[9] that is proposed in planning research. Least commitment planning executes the procedure that creates nondeterministic parts temporarily and commits the parts later in the process. This procedure enables to avoid rollback process. This is also similar for the operation of the workflow. For example, the selection of one place from many places to add task can complicates the latter processing. The solution of this problem means that the second requirement for operator is satisfied. In addition, the definition of operations that create

nondeterministic parts such as least commitment planning satisfies the first requirement.

Based on the above ideas, we defined tuning operators. There are 4 control structures of the workflow that is target in this research, which are sequence, selection, loop and parallel. The definition of the operators in this research does not allow the addition of the control structures newly, which are selection and loop. We do not need to control the processing of sequence and parallel. However, we need to control the processing of selection and loop by using variable data. The false description of these variable data can cause the wrong processing such as endless loop. In this case, the result of the workflow processing can not return appropriately, so the system can not understand whether tuning is wrong or controlling processing is wrong. For this reason, we targeted only sequence and parallel in this definition and use only control structures that are controlled surely in the workflow targeted tuning about selection and loop.

We use Activity and Task in following description of operator definition. They are defined as below in this research.

- Activity: Tasks to invoke Web service in workflow
- Task: All tasks in workflow

In addition, the operational object of operators can be multiple tasks. In case that the operator targets more than 3 tasks, the operator can correspond by repeating operation that targets 2 tasks. So we defined operators that target less than 2 tasks.

We describe tuning operators as below.

### **Add(Task)**

This operator adds new Task that is not included in the workflow. We can describe the relationship between existing task in the workflow and adding task with other tuning operators by adding a task.

### **Delete(Task)**

This operator deletes Task that is included in the workflow. We can not describe the relationship between existing task in the workflow and deleting task with other tuning operators by deleting a task.

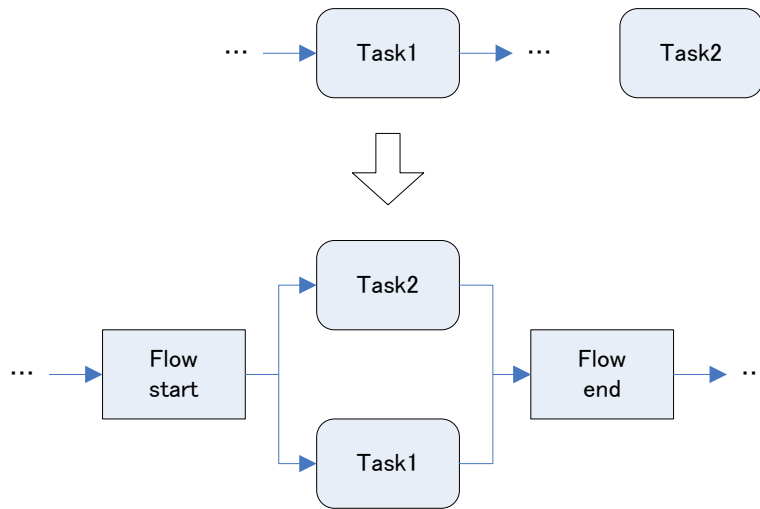


Figure5: Parallelize(Task1, Task2)

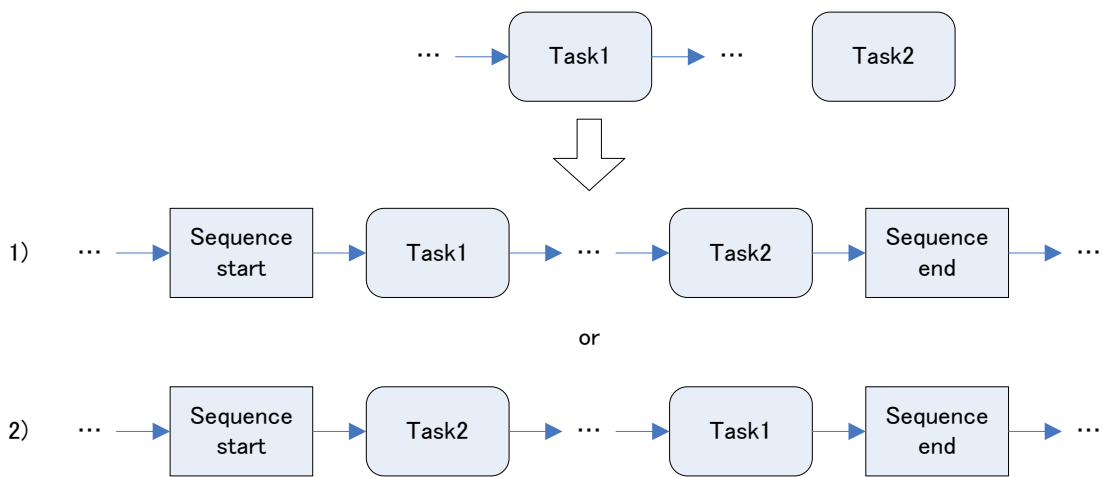


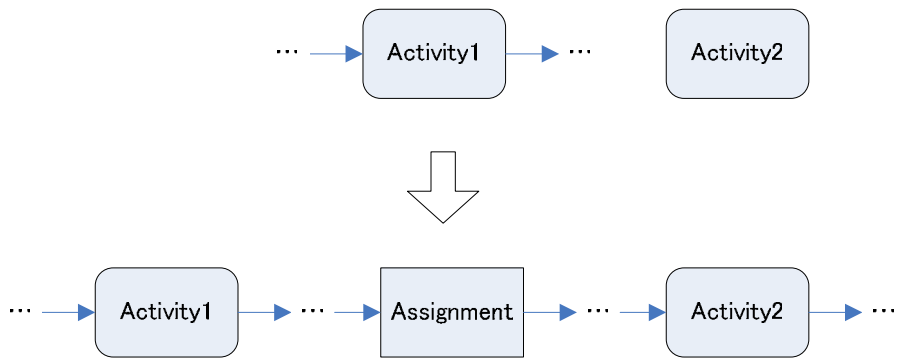
Figure6: Sequentialize(Task1, Task2)

**Parallelize(Task1, Task2)**

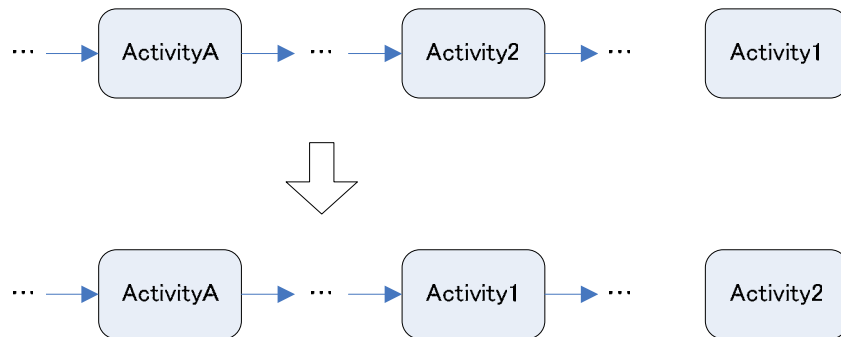
This operator parallelizes Task1 and Task2 that are included in the workflow. In the case that Task1 is already parallel with other Task, this operator parallelizes Task2 with them in a similar way. In other case, this operator adds flow task to the workflow and describes the parallel processing newly. We show an example of Parallelize(Task1, Task2) in Figure 5.

**Sequentialize(Task1, Task2)**

This operator sequentializes Task1 and Task2 that are included in the workflow. In the case that Task1 is already sequential with other Task, this operator



☒ 7. Set(variable of Activity1, variable of Activity2)



☒ 8. ChangeActivity(Activity1, Activity2)

sequentializes Task2 with them in a similar way. In other case, this operator adds sequence task to the workflow and describes the sequential processing newly.

The Sequentialize operator can not describe the positional relationship. Figure 6 shows an example of Sequentialize(Task1, Task2). Sequentialize(Task1, Task2) can generate 2 results, which are 1) Task1 is prior to Task2 and 2) Task1 is posterior to Task2 , as shown in Figure 6. This operator can not describe the number of tasks between Task1 and Task2. Sequentialize can describe only the sequential processing relationship between tasks.

### **Set(variable of Activity1, variable of Activity2)**

This operator describes that the data of the output variable of Activity1 is used for the data of the input variable of Activity2. We show an example of Set(variable of Activity1, variable of Activity2) in Figure 7. Set operator can describe the sequential relationship between Activities. This operator also can not describe the number of tasks between Activity1 and Activity2 and describes only the place of adding Assignment that is described data passing information.

### **ChangeActivity(Activity1, Activity2)**

This operator changes Activity2 that is included in the workflow to Activity1. We show an example of ChangeActivity(Activity1, Activity2) in Figure 8.

We can describe the tuning with these operators. Tuning can be packaged with respect to each coherent unit as described in Section 3.2. We call this package *tuning rule*. Next section describes tuning rule.

## **3.4 Tuning Rule**

The tuning for the abstract workflow has coherent semantic unit. We call it *tuning rule* in this research. This section describes the formalization of tuning rule with the operators that are defined in the preceding section and an example.

### **3.4.1 Formalization of Tuning**

The tuning has been executed for particular workflow searchingly and has not accumulated these tuning know-how. If we describe tuning know-how as rules and accumulate them, we can use them generally.

To satisfy this requirement, we formalized the tuning rule(TR) as below.

TR = (TR-ID, PC, O)

- TR-ID: the specific ID of tuning rule
- PC = (AC, AS, ST): a set of Tasks that mean the precondition
  - AC: a set of Activity. Activity is presented as below.

Activity = (Activity-ID, operation, input, output)

- Activity-ID: the specific ID of Activity
- operation: the method of Activity

- input: the input variable of operation
- output: the output variable of operation
- AS: a set of Assignment. Assignment is presented as below.
  - Assignment = (Assignment-ID, from, to)
    - Assignment-ID: the specific ID of Assignment
    - from: the source variable of Assignment
    - to: the destination variable of Assignment
- ST: a set of Structure. Structure is presented as below.
  - Structure = (Structure-ID, structure-type, AC)
    - Structure-ID: the specific ID of Structure
    - structure-type: the type of Structure (sequence, while, flow)
    - AC: a set of Activity included in Structure
- O: an order set of operator.

This formalization enabled us to describe and accumulate the tuning know-how. In the next section, we describe an example of the tuning rule.

### 3.4.2 Description Example

This section describes an example of the tuning rule “Multi-hop substitution using intermediate language” that is described in Section 3.1.2 according to the format of the tuning rule that is defined in the preceding section. The description example is shown in Table 6. We organized the procedure that we tuned workflow using “Multi-hop substitution using intermediate language” with respect to each operator that is defined in the preceding section and described it as tuning rules.

“:precondition” and below means PC of the tuning rule. In this example, PC is composed of following things.

- Activity: Translation1, Translation2, ParallelDictionary1, TermReplacement1, TermExtraction1
- Assignment: Assignment1, Assignment2, Assignment3
- Structure: Flow1

If this tuning rule is executable, above these match tasks of the workflow. So operators in the tuning rule can refer each task using above IDs such as Translation1.

**Table6: Description Example of Tuning Rule**

<pre> :precondition Activity((Translation1), (translation), (translateRequest), (translateResponse)) Activity((Translation2), (translation), (translateRequest), (translateResponse)) Activity((ParallelDictionary1), (paralleldictionary), (searchRequest), (searchResponse)) Activity((TermReplacement1), (termreplacement), (replaceArrayRequest), (replaceArrayResponse)) Activity((TermExtraction1), (termextraction), (searchRequest), (searchResponse)) Structure((Flow1), (flow), (Translation1)) Assignment((Assignment1), (Translation1/translateResponse/translateReturn/result)             , (TermReplacement1/replaceArrayRequest/source)) Assignment((Assignment2), (Translation2/translateResponse/translateReturn/result)             , (TermReplacement1/replaceArrayRequest/keyword[i])) Assignment((Assignment3), (ParallelDictionary1/result[0]/target)             , (TermReplacement1/replaceArrayRequest/term[i]))  :operators Sequentialize((Translation1), (Flow(Translation1))) Sequentialize((Translation1), (TermReplacement1)) Set((Translation1/translateResponse/result/target)     , (TermReplacement1/replaceArrayRequest1/source)) Add((TermReplacement2)) Sequentialize((TermReplacement2), (TermReplacement1)) Set((TermReplacement2/replaceArrayReturn), (Translation1/translateRequest/source)) Add((MediateCodeParallelDictionary1)) ChangeActivity((MediateCodeParallelDictionary1), (Translation2)) Set((Start/source), (TermReplacement2/replaceArrayRequest2/source)) Set((Start/soLang), (Translation1/translateRequest/soLang)) Set((Start/targetLang), (Translation1/translateRequest/targetLang)) Set((TermExtraction1/result), (TermReplacement2/keyword)) Set((MediateCodeParallelDictionary1/result/target), (TermReplacement2/term)) Set((MediateCodeParallelDictionary1/result/target), (TermReplacement1/keyword)) </pre>
---

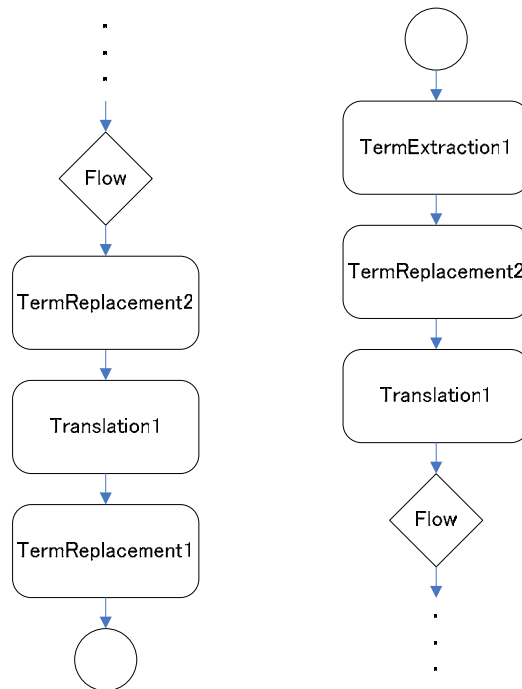


Figure9: Nondeterministic Tuning Example

“:operators” and below means O of the tuning rule.

Tuning operators that are defined in the preceding section can generate nondeterministic parts. There are 2 patterns of the workflow when the following operator is executed as shown in Figure 9 in this example.

```
Set((TermExtraction1/result), (TermReplacement2/keyword))
```

When the following operator is executed, the workflow in Figure9-left is selected by the constraint of the following operator.

```
Set((MediateCodeParallelDictionary1/result/target), (TermReplacement2/term))
```

Multiple workflows can be acquired finally in some tuning rules and tuning object workflows. However, we do not mention which to select in this case.

As above, the formalization of the tuning rule with tuning operators enabled us to describe existing tuning methods as rules.

## Chapter 4 Architecture

Chapter 3 described the workflow tuning and tuning operators that are the unit of the tuning operation and the tuning rule. This chapter describes the interactive workflow generating system that realizes the creation and addition of basic workflows and tuning rules in the process of the creation of the workflow and can satisfy the constraint of the community to the maximum.

### 4.1 Use Case

There are various workflows that users require with respect to each community. Users do not understand what workflows they create and how they create workflows.

We take in the approach to combine the automatic process of workflow creation and user's intention and create workflows searchingly in this research. The automatic process of the creation executes the tuning and instantiate the abstract workflow according to user's constraint and preference. User can interpose his intention into the automatic process of workflow creation anytime. So user can modify the workflow anytime. The automatic process of workflow creation and user's interventions are executed mutually iteratively and the

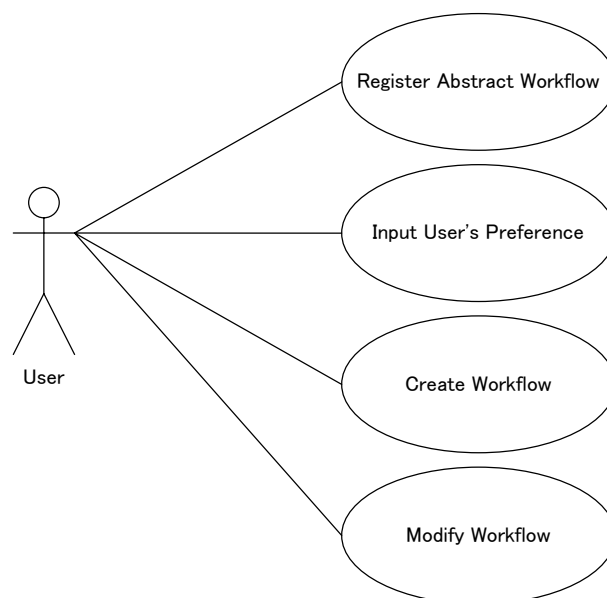


Figure10: Use Case Diagram

workflow is created. We call the system that realizes this process *interactive workflow generating system*. Use case diagram of the interactive workflow generating system is described in Figure 10. In this system, user firstly registers user's basic abstract workflow. Secondly user inputs user's preference. Then user starts to create the workflow and user modifies the workflow anytime in this process of the creation. The registration of the basic abstract workflow and user's modification are optional.

We describe the system architecture that satisfies the use case in the next section.

## 4.2 Interactive Workflow Generating System

We show the system architecture of the interactive workflow generating system in Figure 11. The system architecture consists of User Side, Repository Side and

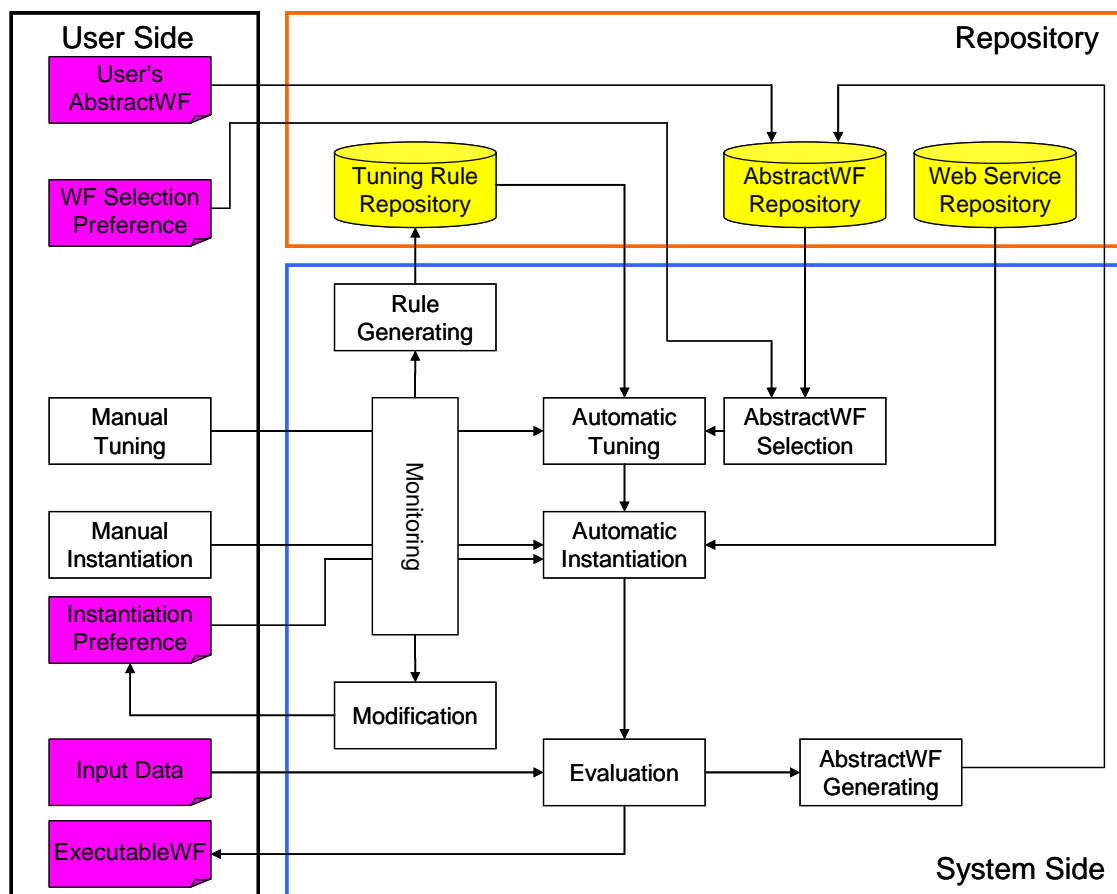


Figure11: System Architecture

System Side.

### **User Side**

In User Side, user inputs the basic abstract workflow, user's preference for the abstract workflow selection, user's preference for the workflow instantiation and data to evaluate the workflow. User monitors the automatic processing of System Side and can interpose user's intention into the processing anytime (Manual Tuning, Manual Instantiation). After processing, user can acquire the executable workflow that reflects user's intention.

We describe the detail of each part as below.

- **User's AbstractWF**

This is the abstract workflow that is created by user. This is a base of the translation service workflow that is created in this system. In the case that user inputs User's AbstractWF, this is registered to AbstractWF Repository in Repository Side and the preference to select the abstract workflow (WFSelection Preference) is modified and User's AbstractWF is selected preferentially. In the case that user does not input User's AbstractWF, the system selects the appropriate abstract workflow according to the preference.

- **WFSelection Preference**

This is user's preference to select the abstract workflow which is registered in AbstractWF Repository in Repository Side. For example, the content of the preference consists of the following things.

- Translation source language
- Translation target language
- Translation domain (e.g. medical care, education, etc)

According to these preferences, AbstractWF Selection in System Side selects the appropriate abstract workflow.

- **Manual Tuning**

This is user's manual tuning operation for the abstract workflow. The operation consists of add, delete, modify and move tasks. These operations are recorded with respect of each tuning operator that is defined in the preceding section and Rule Generating in System Side describes the tuning rule from the series of user's operations.

- **Manual Instantiation**

This is user's manual assignment of the concrete Web services to each task of the abstract workflow. The automatic assignment of the system is executed according to Instantiation Preference. When user assigns the concrete Web services manually, Acquisition in System Side reflects user's operation to Instantiation Preference.

- **Instantiation Preference**

This is user's preference about the assignment of the concrete Web services to each task of the abstract workflow.

- **Input Data**

This is test data to evaluate the quality of the workflow. Sentences that include community-specific terms are used for Input Data. Evaluation in System Side evaluates the workflow with this.

- **ExecutableWF**

This is the executable workflow to satisfy user's preference for the translation service. If user executes this actually, user needs to deploy this workflow as composite Web service to Web service server.

## **System Side**

In System Side, the system firstly selects the abstract workflow, executes tuning and instantiation with data from User Side and Repository Side. Secondly, the system evaluates the executable workflow that is created by the system. If the evaluation is higher than a standard, the system returns the executable workflow to user and abstracts the workflow and registers it to the repository. If the evaluation is lower than a standard, the system proposes the modification of the preference to user. In addition, the system monitors the user's operations and creates tuning rules and reflects it to the preference according to the result.

We describe the detail of each part as below.

- **Rule Generating**

This makes user's manual tuning that Monitoring monitors tuning rules and registers them to the repository. User's manual tuning is recorded by the unit of operators that were defined in section 3.3 and described as tuning rules that

was formalized in section 3.4. We can register the generated rules to Tuning Rule Repository in Repository Side directly. However, we think that it is also important for professional to validate the validity and the effectiveness.

- **AbstractWF Selection**

This selects the abstract workflow that is suited for user's preference from the repository. User's preference is WFSelection Preference that user inputs in User Side. According to this preference, this selects the abstract workflow from AbstractWF Repository.

- **Automatic Tuning**

This selects a tuning rule from executable rules that are registered in the repository and executes it for the abstract workflow that is selected in AbstractWF Selection.

- **Automatic Instantiation**

This assigns the usable concrete Web services to each task of the tuned abstract workflow according to user's preference. User's preference is Instantiation Preference that user inputs in User Side. According to this preference, this selects Web services from Web Service Repository and assigns them to tasks.

- **Monitoring**

This monitors user's manual operations. Target operations are modifications for the workflow (Manual Tuning) and the assignment change of the concrete Web services (Manual Instantiation). According to the result of the monitoring, this executes Rule Generating and Acquisition.

- **Acquisition**

This acquires the information of user's manual assignment change of the concrete Web services that Monitoring monitors and modifies Instantiation Preference.

- **Modification**

This modifies user's preference according to the evaluation of generated workflow. Evaluation in System Side evaluates the executable workflow that is created by the system according to Input Data. If the evaluation is lower than a standard, this proposes the modification of WFSelection Preference and Instantiation Preference. If user allows the modification, the system executes

the process of workflow generation again according to modified the preference.

- **Evaluation**

This evaluates the service quality of the generated workflow. If the evaluation is higher than a standard, the system returns ExecutableWF to user and abstracts the workflow again and registers it to AbstractWF Repository. If the evaluation is lower than a standard, the system executes Modification.

- **AbstractWF Generating**

This abstracts the generated workflow and registers the abstract workflow to AbstractWF Repository. This enables us to reuse the generated workflow as a basic workflow in the subsequent process of the workflow generation.

### **Repository Side**

In Repository Side, there are repositories that accumulate abstract workflows, tuning rules and concrete Web services. Each repository executes the input and the output with User Side and System Side as shown in Figure 11.

We describe the detail of each part as below.

- **Tuning Rule Repository**

This accumulates tuning rules that are used in the automatic tuning phase of the process of workflow generation. Tuning rules are accumulated in the form that was formalized in section 3.4.

- **AbstractWF Repository**

This accumulates basic abstract workflows that are the tuning targets. Basic abstract workflows are described and accumulated in the form that was formalized in section 3.2.

- **Web Service Repository**

This accumulates the information of the usable concrete Web service. The system assigns the concrete Web services to tasks of the workflow according to the accumulated information.

This Section described the architecture of the interactive workflow generating system. In the next section, the processing flow of the system architecture that was described in this section is presented.

### 4.3 Processing Flow

This section describes the processing flows of the system architecture that was presented in the preceding section and the Tuning phase and the Instantiation phase.

#### System Processing Flow

We describe the system processing flow in Figure 12. Firstly, user inputs user's information such as User's AbstractWF, WFSelection Preference, Instantiation Preference and Input Data. In the case that there is User's AbstractWF in user's information, the system registers it to AbstractWF Repository. Then the system executes AbstractWF Selection according to user's information. The basic abstract workflow is selected and the system executes the tuning for it in Tuning process. The processing in Tuning process is described later. When Tuning process finishes, the system assigns the concrete Web services to each task of the tuned abstract workflow in Instantiation process. The processing in

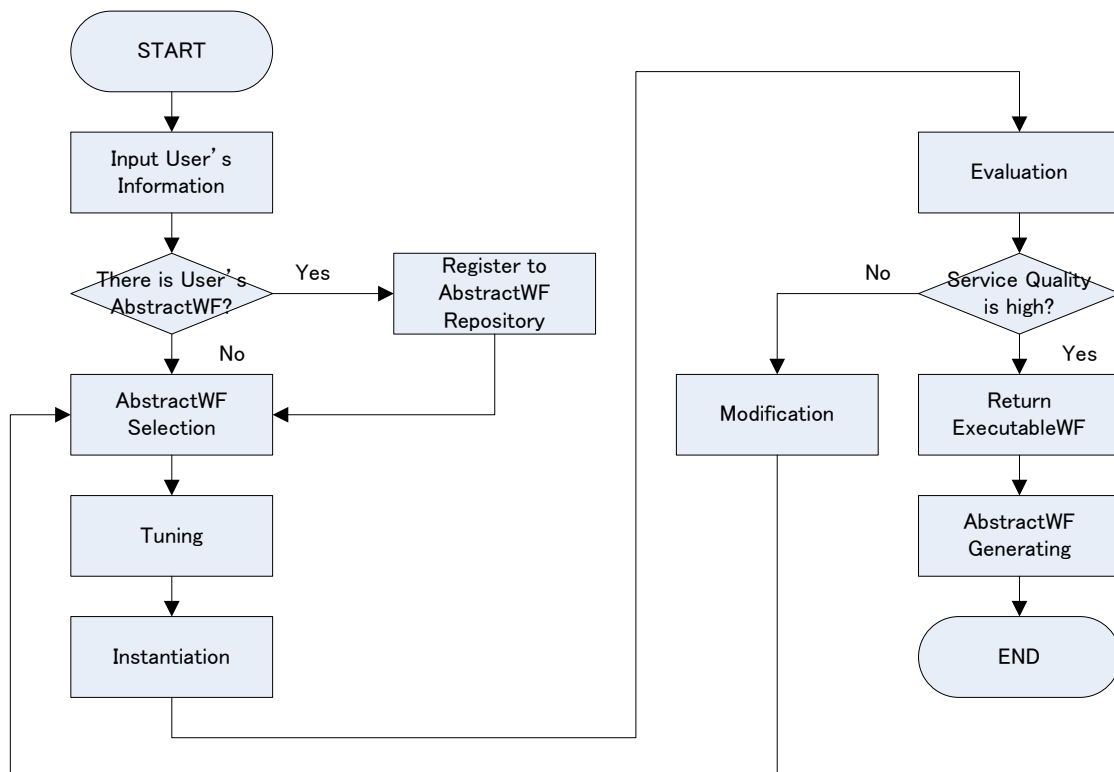


Figure12: System Processing Flow

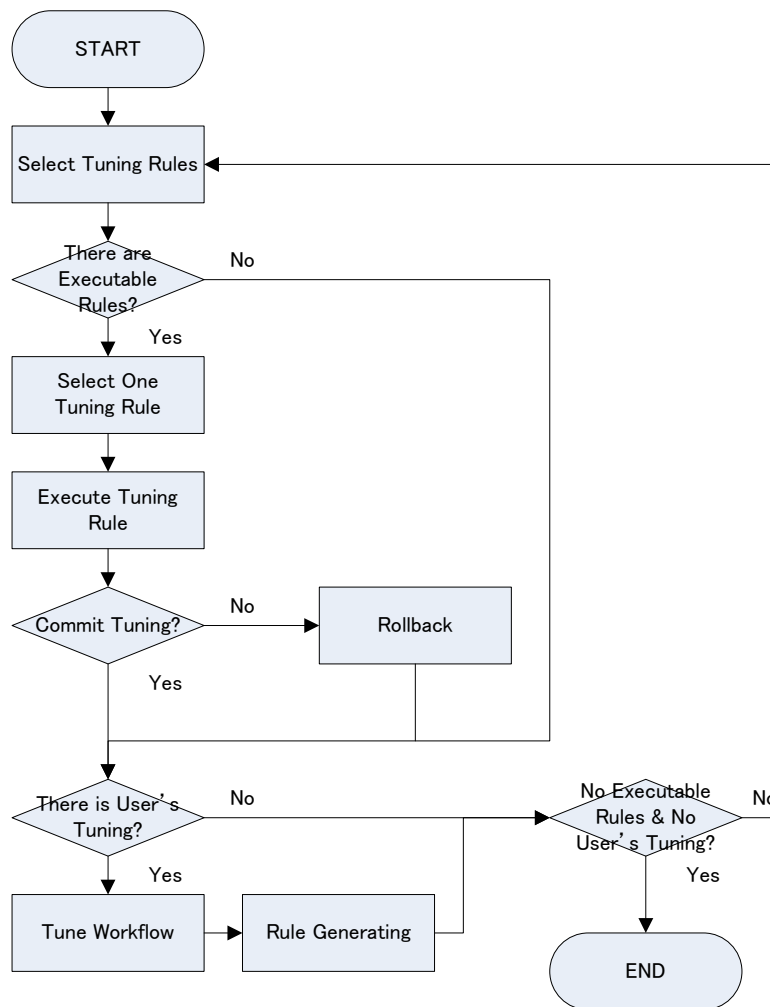


Figure13: Tuning Processing Flow

Instantiation process is described later. Then the system evaluates the service quality of the generated workflow according to user's information. If the evaluation is lower than a standard, the system modifies user's preference and executes the processing from AbstractWF Selection again. If the evaluation is higher than a standard, the system returns the executable workflow to user and executes AbstractWF Generating.

### **Tuning Processing Flow**

We describe the Tuning processing flow in Figure 13. Firstly, the system selects the executable tuning rules for the selected workflow. If there are executable rules, the system selects one tuning rule and executes it. Then the system

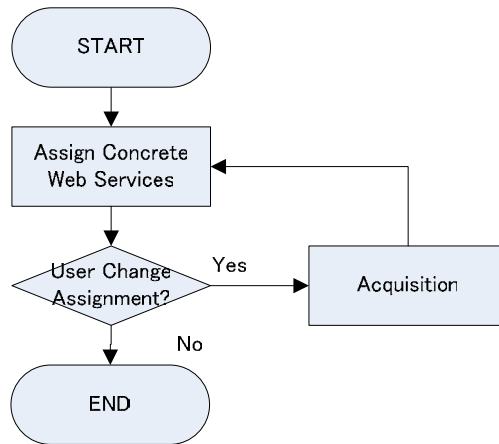


Figure14: Instantiation Processing Flow

executes commitment or rollback for the result of tuning. Secondly, if user executes manual tuning for workflow, system acquires the series of user's operation and describes it as rule. Then if there are no executable rules and no user's manual tuning, system stops Tuning. System executes this process iteratively.

### **Instantiation Processing Flow**

We describe the Instantiation processing flow in Figure 14. The system assigns the usable concrete Web services to each task according to user's preference. If user changes some assignments, the system modifies user's preference in Acquisition. To satisfy user's modification operation and user's preference, the system assigns the concrete Web services to each task again. The system executes this process iteratively.



To evaluate the service quality of the tuned workflow and display the workflow, the system needs to make the generated workflow executable. To execute the composite Web service workflow, we need to deploy it to the Web service server. However, it is very difficult to deploy the composite Web service automatically in the implementation since there are many issues for automatic deployment of composite Web service. In addition, the evaluation of the service quality is often executed in the process of workflow generation. So it is very inefficient to deploy the service to the server and invoke it in this system.

For these problems, this application adopts the method to execute each task of the workflow according to the processing procedure. We call this implementation LocalExecuteWFProcess. We deploy the composite Web

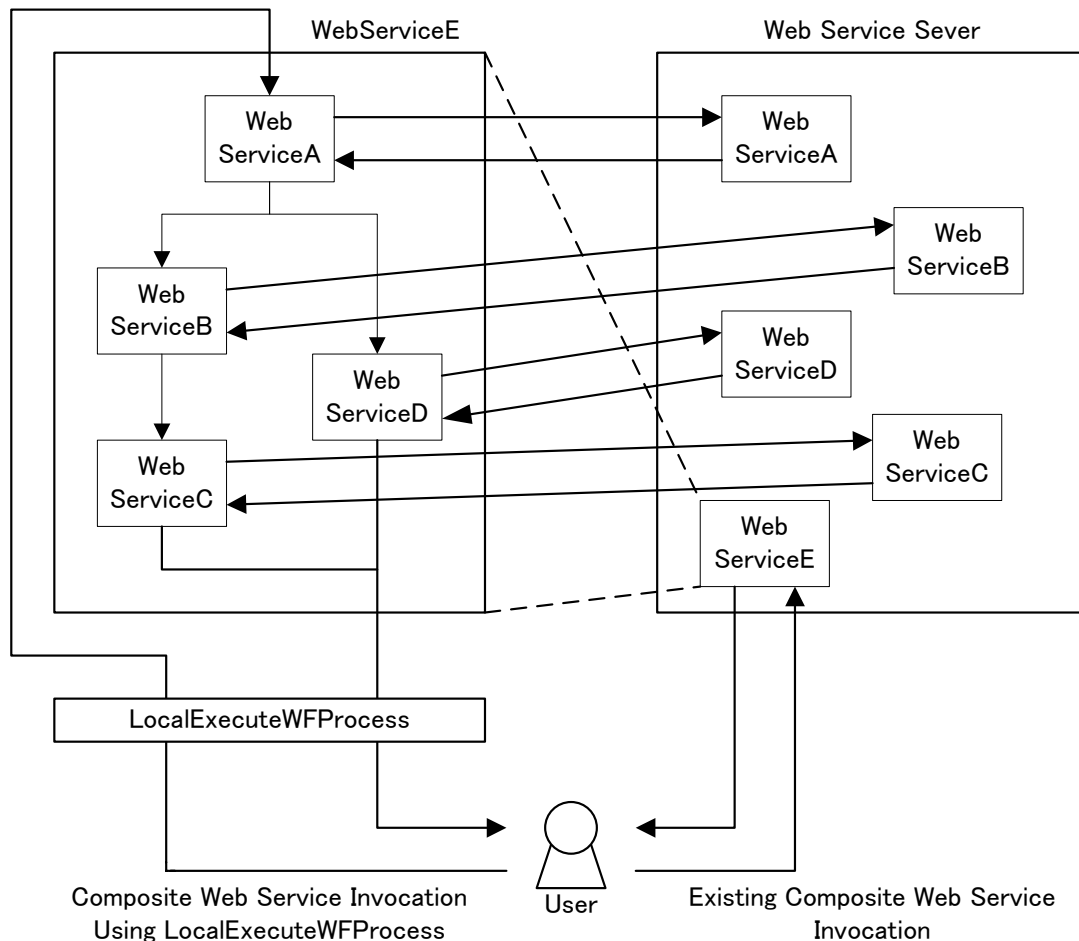


Figure16: LocalExecuteWFProcess

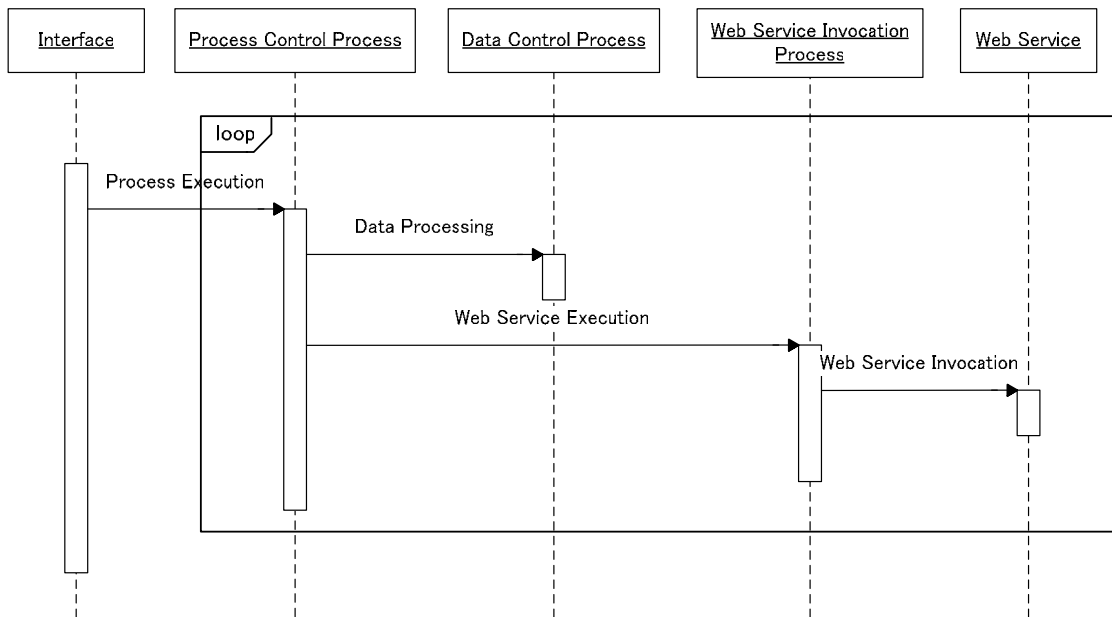


Figure17: Sequence Diagram of LocalExecuteWFProcess

service and invoke it in the existing composite Web service invocation and LocalExecuteWFProcess invokes each Web service that composes the composite Web service according to workflow locally as shown in Figure 16.

The sequence diagram of LocalExecuteWFProcess is shown in Figure 17. The generated workflow is executed in the process control process. Firstly, the system processes variable data in data control process. In the case that the processing task is Activity, data control process processes the value of the input variable that is necessary for the Web service invocation. In the case that the processing task is Assignment, the data control process processes the data passing between variables. In the case that the task is Structure, the data control process processes the value of the control variable. Then in the case that the task is Activity, the system passes the task to the Web service invocation process. The Web service invocation process invokes the Web service that is deployed on the server with SOAP protocol. The system executes this series of the processes iteratively.

LocalExecuteWFProcess executes the following processes for tasks that were classified in section 3.2.

- **Activity:** The system invokes the Web service with SOAP protocol and receives the invocation result.
- **Assignment:** The system executes data passing between variables.
- **Structure:** The system controls the processing flow according to the type of the structure.

LocalExecuteWFProcess enables us to evaluate the Web service workflow efficiently. In the case that we deploy the generated workflow as the composite Web service on the server, we just have to add necessary information to deploy it on the server, so it is very efficient for us.

The next section describes WFGenerator that is application to satisfy the above system function.

## 5.2 WFGenerator

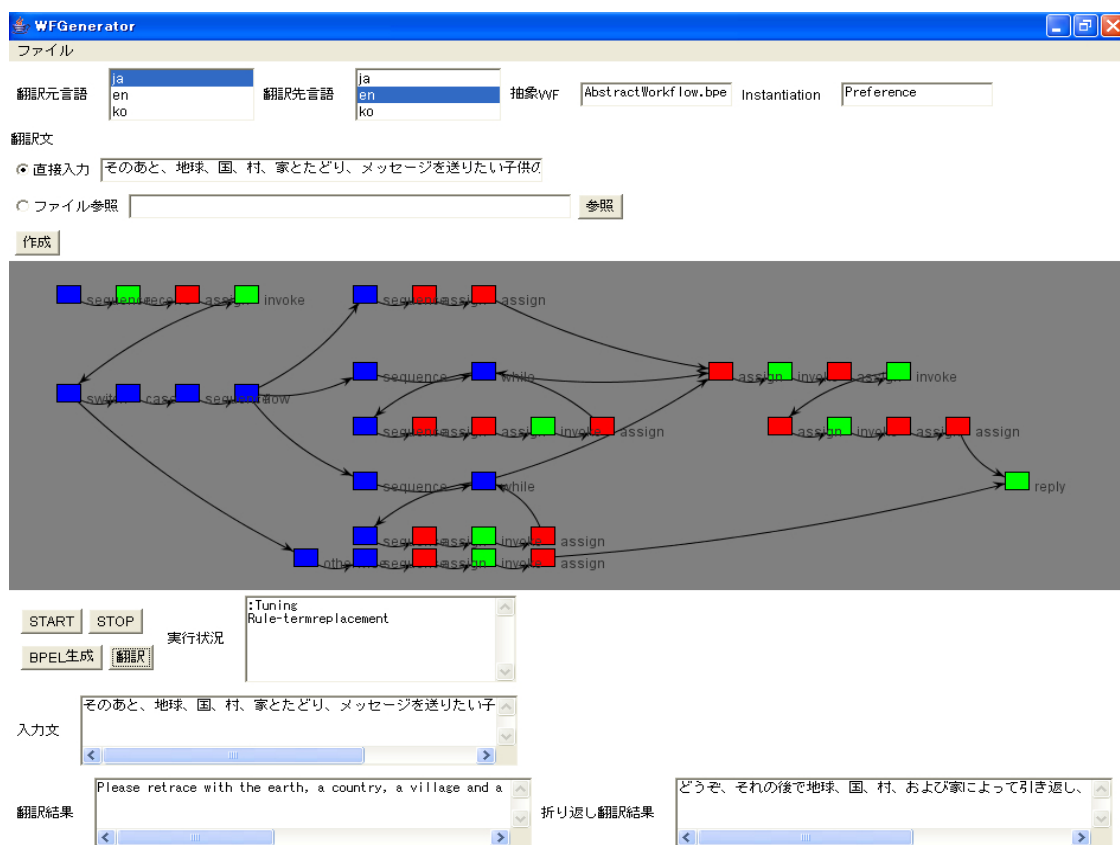


Figure18: WFGenerator

We implemented WFGenerator which satisfies the system function which is described in the preceding section. User interface of WFGenerator is shown in Figure 18.

User interface consists of following 3 sections.

1. User Input Section

User inputs user's preference, the basic abstract workflow and test data to evaluate the quality of the service.

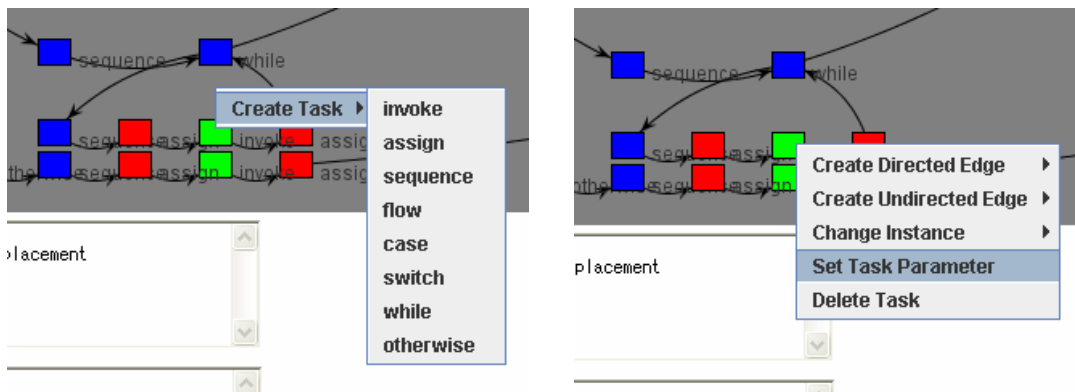


Figure19: User Operation for Graph

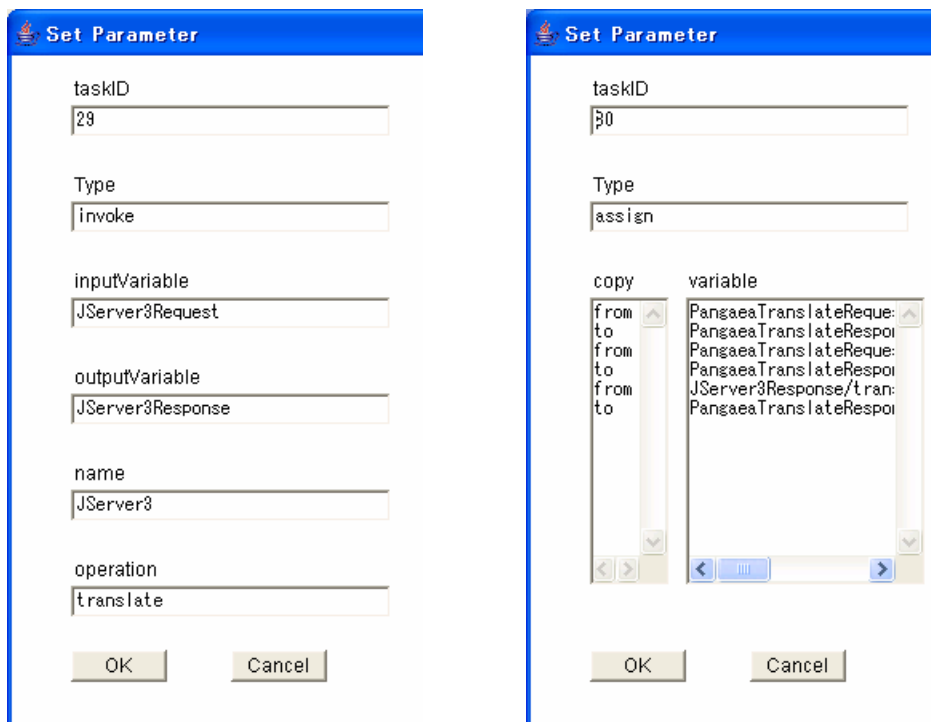


Figure20: Parameter Setting of Task

## 2. Graph Display Section

The application displays the graph of the abstract workflow. The graph of the tuned workflow is displayed each time that tuning rules are executed. User's manual tuning operation is operated to the graph directly using mouse.

Examples of the direct operation are shown in Figure 19. User can execute the operation such as add and delete task, add and delete edge between tasks, and set parameter of task. Examples of the parameter set of tasks are shown in Figure 20. For Activity, user can set the input and the output variable and the operation of the Web service as shown in Figure 20-left. For Assignment, user can set data passing between variables as shown in Figure 20-right.

## 3. Translation Result Display Section

Test data to evaluate the translation service, the result of the translation and the result of the back translation are displayed.

In this chapter, we described the implementation application that focused on the interactive tuning part of the interactive workflow generating system. This application enabled user to execute the process of workflow generation that is composed of the workflow tuning interactively with the system.

## **Chapter 6 Discussion**

In this research, we proposed the workflow tuning and the interactive workflow generating system to support the creation of the Web service workflow of the community specialized high-quality language service. We discuss the workflow tuning and the interactive workflow generating system as below.

### **Tuning by operator**

We defined tuning operators that are the unit of the operation for the tuning to improve the quality of the workflow and formalized the tuning rule. We can describe the existing tuning know-how as the tuning rule by using tuning operators. Tuning operators enabled us to describe the workflow tuning that has been tried to describe searchingly and generally. Herewith we can use the tuning methods that are used in various domains mutually.

It is possible to describe tuning rules directly, but it is much better to create rules intuitively. So when user tunes the workflow directly, the acquisition of user's operations as rules enables us to be easy to describe tuning rules. This was achieved by our definition of tuning operators as the unit of operating workflow and our formalization of the tuning rule.

Tuning operators which were defined in this thesis do not allow us to add tasks to execute selection or loop newly because it is possible that they are not sure to execute the control processing. But in fact, these tasks become a pillar of the workflow. So the correspondence to this issue is our future work.

In addition, we also need to discuss the method to validate tuning rules.

### **Interactive workflow generating system**

We proposed the interactive workflow generating system that is appropriate for the approach to create the community specialized workflow by tuning the basic workflow in this thesis. We think that the workflow tuning is very efficient for supporting the creation of the community specialized workflow. Resources such as abstract workflows that are objects of tuning and the tuning rule are very important in this approach. This system realized the creation and the addition of these resources in the process of workflow creation.

This system also realized the process that user and the system create the

workflow interactively, so this system can reflect user's intention to the workflow any time. This enabled the system to satisfy various demands of the community to the maximum.

To implement the whole system, we need to discuss each processing in the architecture in greater detail. For example, AbstractWF Selection requires the strategy to select one abstract workflow from many candidates. Auto Tuning also requires the strategy to select one tuning rule from many executable tuning rules. We need to discuss the above issues.

## Chapter 7 Conclusion

To solve the communication problem which is caused by the difference of participants' mother languages in international forums which are held by various communities such as NPOs, we focused on supporting the creation of Web service workflow. The workflow is the high-quality language service which is specialized to the community.

We adopted the approach to tune basic workflow and acquire workflow which user requires. In this approach, we needed to solve the following issues.

- The structure of workflow affects the service quality. So even if the content of the composite services are the same as a whole, their service quality can differ from one another. We can improve the service quality by modifying the structure of the workflow. We call this *tuning* in this thesis. Until now professionals have tuned workflows using technical know-how. In this way, only specified professionals can use effective tuning for specified workflow. So we have to formalize tuning method as rules and make it generally available.
- Many resources, i.e. basic workflows and tuning rules are needed to realize the tuning of the basic workflow. So the architecture to easily create and add resources is needed. We also need to satisfy the constraint of the community to the maximum in the process of workflow creation. Therefore we need to design architecture that satisfies the above two requirements.

To solve these two issues, we provided the following solutions.

### **Tuning by operator**

We formalized workflow tuning as rules. Then we defined tuning operators as the unit of operation to describe the tuning rules. This definition placed the user and the system on the common ground to operate the workflow and to allow mutual conversion of their operations.

### **Interactive workflow generating system**

We proposed the architecture of interactive workflow generating system. This architecture realized the creation and addition of the resources, i.e. basic

workflows and tuning rules, to the system during workflow creation. About tuning rule, this system not only accumulates and uses tuning rules that professionals described but also acquires tuning rules from a series of user's manual operations.

Next, to satisfy the constraint of the community to the maximum, we allowed users to intervene anytime in the process of workflow creation. User can not only provide necessary information to create workflow but also create workflow by maintaining mutually complementary relationship with the system. This realized the searching creation of the high-quality workflow. In tuning process, user's intervention is the operation on workflow. This system can acquire tuning rules from a series of user's operations.

In addition, we discussed the above solutions as below.

### **Tuning by operator**

This enabled us to describe existing tuning know-how as tuning rules and use tuning method generally. In addition, this made it easy to convert user's manual tunings and tuning rules mutually and acquire tuning rules from user's manual tunings.

### **Interactive workflow generating system**

This system creates and adds resources which are necessary for tuning in the process of the workflow creation. So this enabled us to lead user's use to accumulation of resources. Then this system also enabled correspondence to various request of the community since the user and the system create workflow interactively.

Our research contribution is to prepare a framework to treat workflow tuning which is efficient for supporting workflow creation for each community.

## **Acknowledgments**

The author would like to express sincere gratitude to the supervisor, Professor Toru Ishida at Kyoto University, for valuable advice and discussion.

The author would like to express his thanks to the advisers, Associate Professor Takashi Yoshino at Wakayama University, Lecturer Tomohiro Kuroda at Kyoto University, and Assistant Professor Satoshi Oyama, for valuable advice and discussion.

The author would like to express his thanks to Associate Professor Shigeo Matsubara at Kyoto University, and Mr. Yohei Murakami at National Institute of Information and Communication Technology, for valuable advice and discussion.

Finally, the author would like to thank all members of Professor Ishida's laboratory for their various supports.

## References

- [1] Ishida, T., Language Grid: An Infrastructure for Intercultural Collaboration. In *IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06)*, pp.96-100, (2006).
- [2] Ben Hassine, A., Matsubara, S. and Ishida, T., A Constraint-based Approach to Horizontal Web Service Composition, In *Proc. Of the 5<sup>th</sup> International Semantic Web Conference (ISWC-06)*, (2006).
- [3] Sirin, E. ,Parsia, B. ,Wu, D. ,Hendler, J. and Nau, D., HTN Planning for Web Service Composition Using SHOP2, In *Journal of Web Semantic*, Vol. 1, No.4, pp.377-396, (2004).
- [4] Yamashita, N. and Ishida, T., Effects of Machine Translation on Collaborative Work, In *Proc. Of the International Conference on Computer Supported Cooperative Work (CSCW-06)*, (2006).
- [5] Uchimoto, K. , Hayashida, N. , Ishida, T. and Isahara, H., “Automatic Rating of Machine Translatability”, In *Proc. Of 10th Machine Translation Summit(MT Summit X)*, pp. 235-242, (2005)
- [6] Sirin, E., Hendler, J., Parsia, B., Semi-automatic composition of web services using semantic descriptions, In *Web Services: Modeling, Architecture and Infrastructure Workshop in Conjunction with ICEIS*, (2003).
- [7] Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S. and Weerawarana, S., Business Process Execution Language for Web Services.  
<http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>
- [8] Muller, R., Greiner, U., Rahm, E., AgentWork: A Workflow System Supporting Rule-Based Workflow Adaptation, *Journal of Data and Knowledge Engineering*, (2004).
- [9] Weld, D., An introduction to least-commitment planning. *AI Magazine* 15(4):27-61. (1994).
- [10]Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J., BLEU: a method for automatic evaluation of machine translation. Technical Report

RC22176(W0109-022), IBM Research Report. (2001).

- [11] Rabideau, G., Knight, R., Chien, S., Fukunaga, A. and Govindjee, A., Iterative repair planning for spacecraft operations in the ASPEN system. *In International Symposium on Artificial Intelligence Robotics and Automation in Space (i-SAIRAS)*. (1999).