# 國 立 交 通 大 學

多媒體工程研究所

**碩 士 論 文**

以案例式規劃推論架構輔助 U 化學習應用系統設計

A Case-Based Planning Scheme to Support U-learning Application Design

研 究 生：王芙民

指導教授：曾憲雄　博士

**中 華 民 國　九 十 六　年 六 月**

# 以案例式規劃推論架構輔助U化學習應用系統設計

學生：王芙民　　　　　　　　　指導教授：曾憲雄 博士

國立交通大學資訊學院
多媒體工程研究所

## 摘　　要

近年來由於無線網際網路與嵌入式系統的發展，使得U化學習的研究愈來愈熱門，而在一個U化學習的環境中包含了許多跨領域的軟硬體技術，例如：學習教材系統、手提設備、嵌入式感測器等等，提高了U化學習應用程式的設計成本。然而，根據我們的觀察，多數U化應用程式都是由一些服務與其輸出入訊息，以及流程控制知識所組成的，若能重複利用前人的設計構想對於設計一個新的應用程式是相當有幫助的。因此，我們利用案例式規劃技術來協助設計新的應用程式，此外，我們使用細顆粒的案例再用技術來修正案例使其更容易符合設計者的需求。為了能重複利用現存的應用程式設計構想，我們提出一個階層式案例式規劃技術，將現存的應用程式視為案例並將之儲存，來輔助新的U化應用程式設計；在此階層式案例式規劃技術中，我們利用計劃來描述一個應用程式的流程控制知識，並且提出一個三層式的案例架構來輔助細顆粒的案例再用；其中，一個案例是由一組任務所組成，而每個任務又由一組服務所組成，此外，我們分別為案例與任務定義了一組特徵來描述之，並且參考現有的U化應用程式建立了一個訊息本體論。我們提出了一套案例擷取與案例修正的方式來使產生的計劃能夠符合設計者的需求。最後，我們做了一個試驗性質的植物園導覽實驗，而結果顯示產生的計劃是可接受的。

**關鍵字：　案例式推論、規劃、U化學習**

# A Case-Based Planning Scheme to Support U-learning Application Design

Student: Fu-Ming Wang Advisor: Shain-Shyong Tseng

Institute of Multimedia Engineering
National Chiao Tung University

## Abstract

In recent year, due to the development of wireless network and embedded system, researches about U-learning become more and more popular. However, the design of U-learning application is costly and time consuming since cross domain technologies are required such as software learning content system, handheld device, embedded sensor, etc. With our surveys, many U-learning application designs can be decomposed as a set of services and messages with similar Control Flow Knowledge. Thus, it is helpful if we can reuse previous design ideas when designing a new application. Therefore, the idea of Case Based Planning (CBP) approach is proposed to support the new design. Moreover, different from the traditional CBP, the idea of reusing fine-grained case is also proposed to adapt the requirement of new case design. Consequently, in order to reuse the design ideas of existing applications, we propose a Hierarchical-Case-based Planning (HCBP) scheme to store applications design as a case and support new U-learning application design. In the HCBP scheme, a plan is used to represent the Control Flow Knowledge of an application. To enable fine-grained case reuse, a three-layer case hierarchy is defined, where a case is composed of a set of tasks and each task is composed of a set of services. In the HCBP scheme, we define a set of features to describe case and task in U-learning, and a message ontology is constructed for the reference of existing U-learning applications. In the HCBP scheme, the case retrieval, and case adaptation operations are proposed to fulfill the requirement of the resulting plan. Finally, a trial experiment about designing *a new botanical garden guiding application* using HCBP is done, and the resulting plan is acceptable.

**Keyword: CBR, Planning, U-learning**

# 誌謝

這篇論文的完成，首先要感謝我的指導教授，曾憲雄老師。在研究所兩年的歲月裡，無論是在學術研究或是為人處世方面，皆讓我受益匪淺，尤其是我學到了對一個知識領域的研究方法、邏輯思考及表達能力的訓練，這將使我終生受用不盡。同時也感謝我的口試委員，楊鎮華教授、孫春在教授和黃國禎教授，他們給予了我相當多的寶貴意見，讓本論文更有意義與價值。

再來要感謝的是蘇俊銘學長、翁瑞峰學長和林喚宇學長，在這段期間內，常常麻煩他們在百忙之中騰出時間與我討論並給我建議、想法，協助我修改論文。此外，我也從他們身上學習了不少生活態度及為人處事的方法，在此深表感激。同時也感謝實驗室的同窗夥伴們，昂叡、信男、雨杰、東權、曉涵、嘉妮，在這兩年的時光裡，不管是學業上或是生活上，他們都陪伴著我渡過這段碩士生涯，很高興能夠交到可以這樣同甘共苦、互相扶持鼓勵的朋友。還有其他在身邊鼓勵我的朋友們，雖然無法在此一一提及，但我心裡真的非常感激有你們在我身邊。

最後要感謝的是我的家人，默默地支持與鼓勵，並不時地關心我，是我在心力交瘁時還能保持鬥志的原動力。日後，我會更加努力地繼續前進，不辜負他們的期望。

僅將本篇論文獻給每一位支持與幫助我的人。

# Tables of Content

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1. Introduction

In the past decade, E-learning becomes popular due to the widespread network and computers, and enables learning activities progressing at anytime and anyplace. Recently, with the development of wireless network, sensor technology and embedded system, it becomes possible to cooperate the computing and information powers in an environment. From the Weisor's vision [18], he described that a variety of computing and communication capability computers will be ubiquitous around our environment and the associated technology is called the ubiquitous computing. In recent years, the ubiquitous computing technology is further applied in seamless learning and is called U-Learning.

The computing and communicating capabilities (such as sensors, robots or RFID tags) can be embedded or attached to ordinary things and let them behave smartly to provide intelligent services. This feature of context-aware reasoning mechanism for the user centric service is called the ubiquitous intelligence [17]. In the U-Learning domain, the context-aware property can provide more intelligent services than E-learning. Therefore, we model the U-learning service design as a system composed of reasoning systems used for context interpretation, content retrieval systems, and devices used for context collection, content displaying, etc. In other words, a U-learning service design is a heterogeneous environment composed of software and hardware components to achieve context-aware.

Based on our observation, each component has the same model, which means all of them composed of input message, operation and output message, therefore, we take these components as service, then a U-learning application is the Control Flow Knowledge that describes the sequence of services. In an application, a service is triggered by an input message that happened in the environment, such as learner approach, after the first service operation is done, it sends its output message to another service to trigger it according to the Control Flow Knowledge, making a flow of service. Notice that services can be triggered parallelized other than just a linear sequence. Consequently, we can say that to design a U-learning application is actually design a control flow of services. However, as mentioned above, a U-learning application is composed of devices and systems, making it difficult and costly to design a new application because all services including software systems and hardware devices have their environmental and functional constraints, and to construct Control Flow of U-learning application involves many kinds of domain expertise such as education, software system and hardware device.

Many researches have been proposed to design some specific U-learning applications, such as Japanese polite teaching [21], knowledge awareness map [11], requirement satisfied learning [2], which will be introduced in next chapter. Since most of them have some similar features, such as getting learner's identity at the first of the application or trying to provide the most suitable content to learner, how to reuse the design ideas to of these features to reduce the cost of designing a new application is a challenging and interesting issue. Nevertheless, each U-learning application is an independent system, the design ideas only can be reused after designer survey all the articles of relative researches, and when system becomes larger and larger and paper becomes more and more, it is hard for a designer to afford the huge amount of information. Moreover, it is also hard to integrate multiple designs together and to fulfill each service's requirement and constraints manually, therefore, we want to propose a systematic mechanism to help retrieve and reuse existing Control Flow Knowledge.

However, in order to reach this goal, there are three issues we have to overcome. First, the Control Flow Knowledge is usually implicit and embedded in the system, how to represent it is an issue. Second, how to find similar U-learning application for reuse? Third, in order to integrate parts of different U-learning application, how to find and integrate these applications becomes a problem. Hence, we propose Hierarchical Case-based Planning (HCBP) scheme to solve the problems. In our approach, plan is used in order to draw out Control Flow Knowledge that embedded in U-learning application system originally; Case-based reasoning solves similar application retrieval, and hierarchical planning solves application integration and ensures the flow correctness. In the proposed scheme, designer is ask to give a task flow of an application first, and then HCBP system is used to retrieve and integrate applications, at last, an appropriate services set and Control Flow Knowledge is produced.

In the next chapter, some related researches are introduced and discussed. The HCBP scheme is introduced in Chapter3 and U-learning application is applied to HCBP in Chapter 4. Evaluation and discussion are addressed in Chapter 5, and at last, a conclusion about HCBP is given in Chapter 6.

# Chapter 2. Preliminaries

In this chapter, some common U-learning Applications are introduced first. Since ontology plays an important role in our research, previous researches using ontology in U-learning are next reviewed. Finally, we will introduce several previous researches about case-based planning and discuss the capability of existing case-based planning systems.

## 2.1. U-learning Application

At present, there are many U-learning applications have been proposed. Among them, several applications that are representative are introduced. More U-applications are shown in [3][7][14][16][22].

The U-learning system for Japanese Polite Expression proposed in [21] can provide learner proper polite expression should be used according to the occasion. In which PDA is used as the content display device and each learner are asked to fill in his or her own profile first. And learner's location is confirmed by GPS or RFID according to the environment is indoor or outdoor, or by schedule; therefore, the occasion is judged by location and time. When a learner talks to a conversational partner, the system gets the information for the person via the infrared data communication of the PDA, and then suggests propriety polite expression for the learner.

In [11], a personalized knowledge awareness map system is proposed to provide a knowledge awareness map that shows the distance and relation degree of learning materials and peer learners related to learner's input from a PDA. In the application, a leaner is asked to input a query topic that he or she is interested in, the system finds out the related materials and peer learners who are familiar with the learner's input topic, and then the locations of the most related materials and peer learners chosen are retrieved in location repository and distances between the learner and the retrieved materials and peer learners are calculated. Finally, a related material and a peer learner knowledge awareness map are generated respectively using the related degree and distance, and then displayed to learner.

In the requirement satisfied learning environment system [2], a teacher or a parent has to set some requirement to the learner first, e.g., the learner is not permitted to play video games without completing the homework. In the learning environment, RFID tags are deployed on the objects in order to detect the learner’s behavior. Afterwards, if something in conflict with the requirements was done, an alert is showed to the learner, parent or teacher.

As U-learning applications introduced above, each application has its own Control Flow Knowledge. To reuse the Control Flow knowledge in existing systems will be helpful when designing a new large and complex application.

## 2.2. Ontology in U-learning

Ontology is a knowledge representation model that specifies the concepts and relations of knowledge and has been used in various research domains, such as knowledge engineering, natural language processing, knowledge management, etc. to facilitate knowledge sharing and reuse. In [15], a context description model using ontology is proposed for U-learning. The author conceived that context aware is an interactive model between learners and service and two types of context ontology, learner ontology and service ontology for describing learners and services are proposed. The learner ontology contains learner profiles such as personnel profile, accessibility and preferences, calendar profile, social profile, and location profile, and the service ontology contains service profile such as input, output, pre-condition, and effect of service execution. In our research, we construct a message ontology to maintain the relations between messages used in U-learning and support case retrieval and adaptation. The message and message ontology will be illustrated in next chapter detailedly.

## 2.3. Case-based Planning

Case-Based Planning (CBP) is an approach that reuses existing plans to solve new problems that similar to previous ones. Traditional CBP is *single-shot*, that is, a single coarse-grained case is reused to solve the present problem. In general, a coarse-grained case-based planning system generate new plans by retrieving and adapting old ones in case base [4][6][13], and the retrieved plan should be the one that need least adaptation to fit the current situation. The disadvantage of traditional CBP is that it is inadequate to solve complex problems, and a more complicated adaptation method has to be designed to adapt the retrieved plan to the new problem.

On the other hand, there are some researches about fine-grained CBP have been proposed to enhance the capability of CBP and make multiple-case reuse possible. In[1], a fine-grained case-based reasoning system used to generate simulation plans is presented based on the concept that problem solving experiences can be partitioned and used as independent cases. In the system, when the retrieved cases are not similar enough, a secondary retrieval is executed. That is, the system tries to retrieve partial plans that satisfy the new problem instead of retrieving a complete one. In research [15], the author proposed an approach for plant-control design. Cases are used at multiple levels of abstraction to represent complex problem solutions as hierarchies, where abstract case solutions are used to act as problem decomposition knowledge, and concrete case solutions are executable programs. This kind of case hierarchy promotes the reuse of solution parts from different complete solutions to reduce the adaptation overhead. However, both of them are used in pure domains that are not as complicated as U-learning in which more information and constraints have to be concerned; therefore, their planning approaches are not suitable for U-learning.

In [10], a novel framework to support workflow modeling and design by adapting workflow cases from a repository of process models is proposed. In which the authors proposed a conceptual model of workflow cases, a similarity flooding algorithm for case retrieval and a domain-independent AI planning approach to workflow case composition. Comparing to other general case-based planning systems, the workflow modeling approach is special because it concerns the workflow when retrieving similar cases instead of just concerning some features that utilized for case description. However, when retrieving a similar section of workflow, only a totally matched section will be considered; therefore, this approach is not proper for U-learning application design because a U-learning application plan needs more flexibility.

# Chapter 3. Hierarchical-Case-based Planning

As mentioned in Chapter 1, there are three issues should be solved before the Control Flow Knowledge in existing U-learning application could be reused. First, how to represent the Control Flow Knowledge which is usually embedded in the system code? Second, how to find a similar U-learning application for reuse? Third, how to find and integrate some tasks of applications to construct a new application according to designer's requirement? Above all, in order to reuse Control Flow Knowledge of U-learning application, case-based reasoning is the approach that suitable for use, in which each application is considered as a case. Moreover, in our approach, plan is used to represent the Control Flow Knowledge of U-learning application, and a case adaptation algorithm is proposed to integrate and adapt the retrieved cases.

In a U-learning application, there are a lot of components involved in, crossing software and hardware domains. For example, sensors and RFIDs are needed to gather information in the environment; programs are needed for context interpretation and context-aware content retrieval; repositories are needed to store up learning materials and information like learner's portfolio; PDA, mobile phone, tablet PC, etc. are needed for content displaying. Therefore, inter-domain communication between these components to construct a U-learning environment is significant, and this kind of domain complexity increases the difficulty of planning for U-learning application design. Based on our observation, each component has the same operation pattern, that is, an input message, an operation and an output message. In order to simplify the planning process, each component is considered as a service here. Services are the primitive component in our approach; in other words, the Control Flow Knowledge of an application is actually a flow of services.

As mentioned in previous chapter, to construct a U-learning application is not an easy work due to the complex domains involved in. Besides, a U-learning application is adaptable to the situation of the environment; it may have different response in different situations. That is to say, U-learning application is more changeable than traditional E-learning application, and makes it even harder to generate a plan of Control Flow Knowledge. Therefore, when using CBR to reuse previous designed applications, a fine-grained approach is proposed. In a fine-grained approach, a new plan can be generated through combination or modification of several retrieved cases if needed, instead of reusing only the most similar coarse-grained case. The difference between fine-grained and coarse-grained case reuse is shown in Figure 3.1.

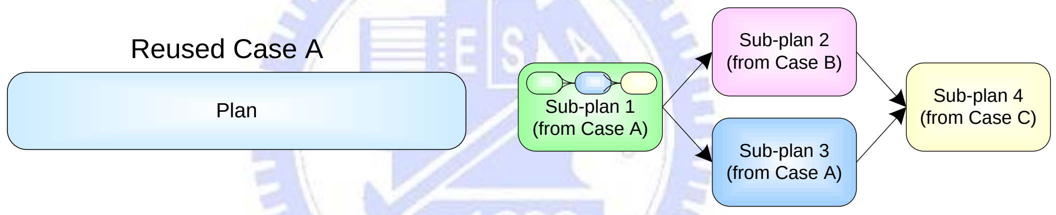


**Figure 3.1 Difference between coarse and fine grained case reuse**

It will take a lot of efforts for adaptation if only a coarse-grained case is reused in a complex domain such as U-learning. As a consequence, we propose a hierarchical case representation model let a case be reused in different layer of granularity, from coarse to fine, hierarchically. In our proposed case model, we defined the case granularity into three layers, where an application is considered as a case, a case is composed of tasks, and a task is composed of services. The detail of case representation will be illustrated in Section 3.2.

In order to reuse hierarchical-case, we proposed a Hierarchical-Case-based Planning (HCBP) scheme, in which designer is first asked to input a Desired Task Flow, the description of each Excepted Task, and the description about the whole case in his or her mind. Subsequently, a Possible Desired Case Feature Table is generated from the designer's input. Afterward, a most similar and reusable case is retrieved from the case base and then case adaptation is carried out according to the designer's input. In the proposed HCBP, the adaptation process includes case reuse and case revise process in traditional CBR. When adapting the most similar case, a fine-grained case retrieval is executed to retrieve parts of cases that suitable to be replaced or added into it. Finally, a resulting new case satisfy designer's input is generated, and then the new case is retained in the case base. The whole process is shown in Figure 3.2.

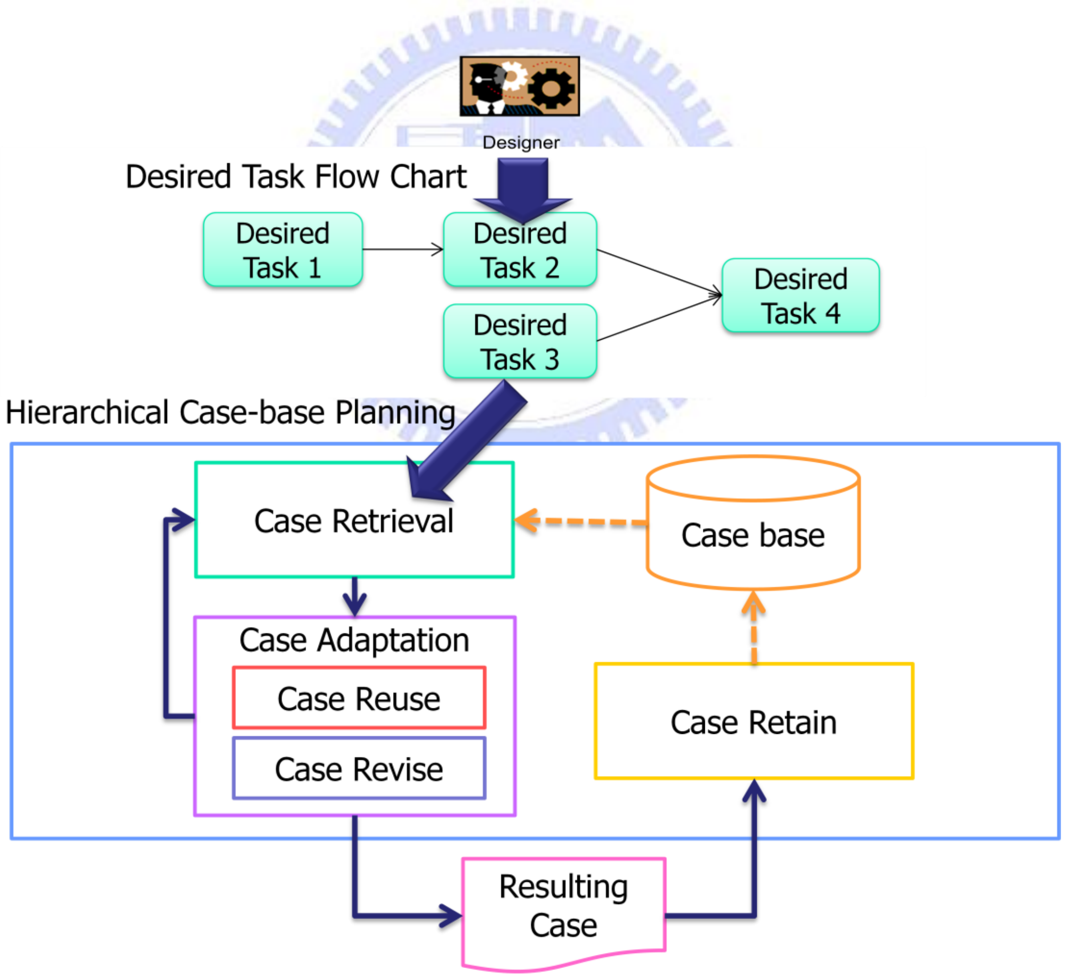


**Figure 3.2: Hierarchical-case-based planning**

## 3.1. Desired Task Flow

From user's point of view, a U-learning application can be seen as a flow of desired task. For example, when designing a museum guiding application, the designer may intend to get learner's ability first, and when learner approaches an exhibition, the approached exhibition is confirmed. Afterwards, the system will retrieve related materials or information about the exhibition that are suitable for the learner according to the learner's ability. At last, the retrieved material or information about the exhibition is displayed to learner by a mobile device. In the example, each step is considered as a desired task; the whole process of desired task flow of the example is shown in Figure 3.3.

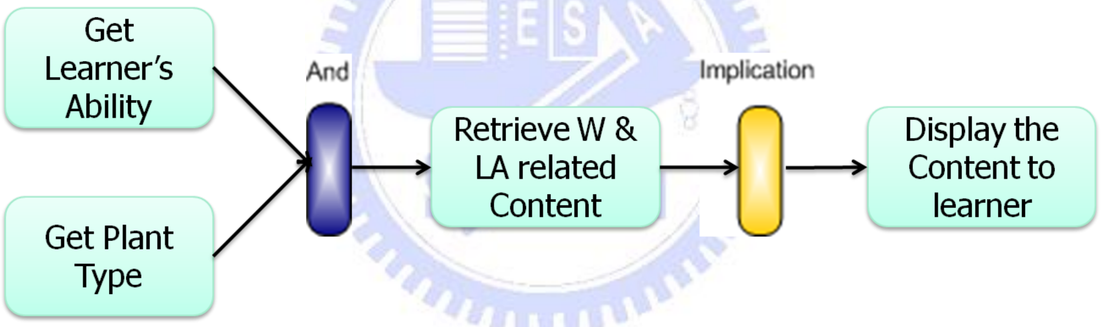

Get Learner's Ability
Get Plant Type
And
Retrieve W & LA related Content
Implication
Display the Content to learner

**Figure 3.3: Task flow of museum guiding application**

The definition of Desired Task Flow is shown as follows:

**Definition 3.1: Desired Task Flow**

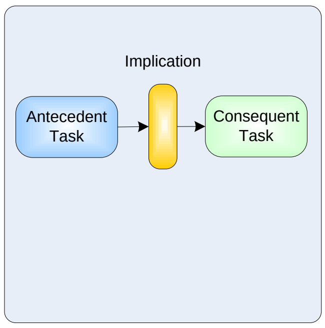
**DTF = (DT, DSR), where**

**1. DT = {$dt_1$, $dt_2$, …, $dt_n$}** is a finite set of desired tasks

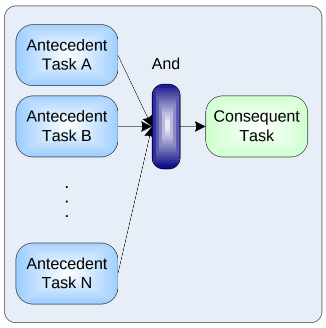
**2. DSR = {$dsr_1$, $dsr_2$, …, $dsr_n$}** is a finite set of sequence relations, will be illustrated later.

A U-learning application may be not always as simple as the one shown in Figure 3.3. In many circumstances, the relations between tasks are not only the “implication” but also some other relations. Here we defined several kinds of task relations for the designer’s input desired task flow. The first kind of relation is Implication, which is the most common relation appears in all applications, as shown in Figure 3.4(a). Other relations are extensions of the Implication relation, such as And, Or, and $C_n^m$. For the And relation, there are two situation, the first one is shown in Figure 3.4(b). In this situation, the subsequent task will be triggered only when all of its antecedent tasks are done, and in the second situation, after the antecedent task is executed, it will trigger all of its subsequent tasks as shown in Figure 3.4(c). For the Or relation, there are also two situations. In first situation, the subsequent task will be triggered when either one antecedent task has been done, as shown in Figure 3.4(d), and in the second situation, after the antecedent task is executed, it will trigger one of its subsequent tasks as shown in Figure 3.4(e). For the last relation $C_n^m$, the first situation is shown in Figure 3.4(f). In this situation, only when n antecedent task of the total m is done, the subsequent task is triggered. In the second situation, n subsequent tasks out of m will be triggered after the antecedent task is done, as shown in Figure 3.4(g).

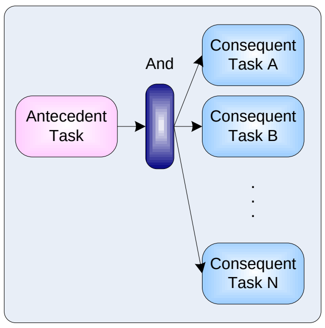
It is worth to mention that the relations defined here is the most frequent relations may appear in a U-learning application, they are not powerful enough to handle complex circumstances. In this thesis, only the relations defined above are discussed.



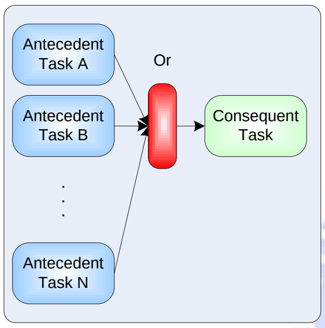
**(a) The Implication relation**



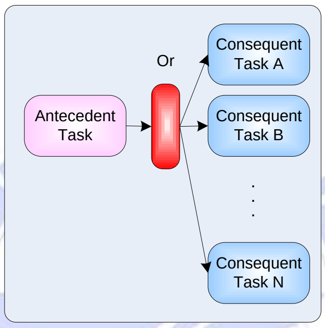
**(b) The And relation 1**



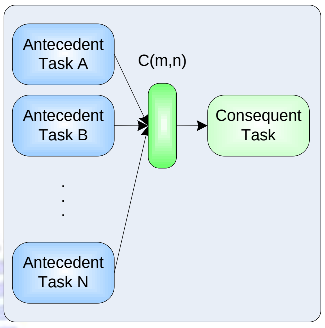
**(c) The And relation 2**



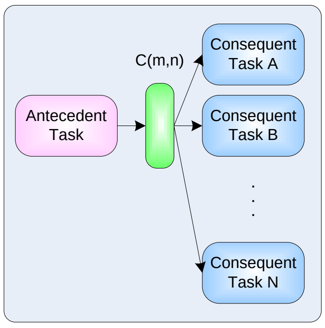
**(d) The Or relation 2**



**(e) The Or relation 2**



**(f) The $C_n^m$ relation 1**



**(g) The $C_n^m$ relation 1**

**Figure 3.4: Relations between desired tasks**

For each Desired Task in a Desired Case, a Desired Task Metadata is used to describe the task, and the Metadata is actually a set of features; for a Desired Case, a

Possible Desired Case Feature Table is used to describe the Case. After the Desired Task Flow is constructed, the designer is asked to fill in the feature tables for each Excepted Task and some feature values about the Desired Case. And then a complete Possible Desired Case Feature Table is generated by aggregating designer' input. Notice that the features for Desired Task are the same as Metadata of Task in case base, and the features for Desired Case are the same as Cases in case base. Details of these features and the aggregation process will be discussed in next section. The definition of Desired Task Metadata and Possible Desired Case Feature Table are shown below:

**Definition 3.2: Desired Task Metadata**

**DTMD =** (<F, V>$_1$, …, <F, V>$_n$)**,** F is the features that used for task description, and V is the values of features.

**Definition 3.3: Possible Desired Case Feature Table**

**PDCFT** = (<F, V>$_1$, …, <F, V>$_n$)**,** F is the features that used for case description, and V is the values of features.

## 3.2. Case Representation

As mentioned above, we represent case in three layers --- Application, Task and Service. And in our HCBP scheme, a case is composed of Input Message, Output Message, a Possible Case Feature Table, a Task Flow and a set of Service Flow, where Input Message is the message that triggers a Task and Output Message is the message that generated from a Task, the definition of Message is shown as below.

**Definition 3.4 Message**

**M = {$m_1$, $m_2$, …, $m_n$}** is a finite set of messages

Possible Case Feature Table is a set of features that give extra information about a case and they are the same as features in PDCFT; Task Flow is the Control Flow Knowledge of an application represented in task level granularity, and Service Flow is the Control Flow Knowledge of an application represented in service level granularity. Each Service Flow is mapped to a task, as shown in Figure 3.5, where Task 1 mapping to a Service Flow that composed of Services 1, 2, and 3. Each Task has Input Message and Output Message, and Task Metadata for task description, where the feature defined for Task Metadata are the same as features in DTMD; each service has its own Input Message and Output Message but no metadata because service is the primitive component, and we consider that I/O message is enough to describe a service's function. In the proposed scheme, Case base maintains a repository to store services, tasks, messages and the application cases which link to the tasks and services, and when a generated new application has to be retained, it will be stored as a case and decomposed into a set of tasks with corresponding services. Service, Task and Case are defined as follows and case structure is shown in Figure 3.5.

**Definition 3.5 A Service is a three-tuple**

**S = (SIM, SOM, RD), where**

1. **SIM = {$im_1$, $im_2$, …, $im_n$},** where $im_i \in M$
2. **SOM = {$om_1$, $om_2$, …, $om_n$},** where $om_i \in M$
3. **RD** is the Resource Description of service

**Definition 3.6 A Task is a four-tuple**

**T = (TIM, TOM, SF, TMD), where**

1. **TIM = {$im_1$, $im_2$, …, $im_n$},** where $im_i \in M$
2. **TOM = {$om_1$, $om_2$, …, $om_n$},** where $om_i \in M$
3. **SF = (S, SSR), SSR** includes relations “Implication” and “And”
4. **TMD = (<F, V>$_1$, …, <F, V>$_n$)**, **F** is the features that used for case description, and **V** is the values of features.

**Definition 3.7 A Case is a four-tuple**

**C = (AIM, AOM, TF, PCFT), where**

1. **AIM = {$im_1$, $im_2$, …, $im_n$},** where $im_i \in M$
2. **AOM = {$om_1$, $om_2$, …, $om_n$},** where $om_i \in M$
3. **TF = (T, TSR), TSR** includes relations “Implication” and “And”
4. **PCFT = (<F, V>$_1$, …, <F, V>$_n$), F** is the features that used for case description, and **V** is the values of features.

It is worth to mention that, for PDCFT to PCFT, they are almost the same despite of PDCFT is in the Desired Case that designer inputs and PCFT in Cases that in case base, and so as DTMD to TMD.

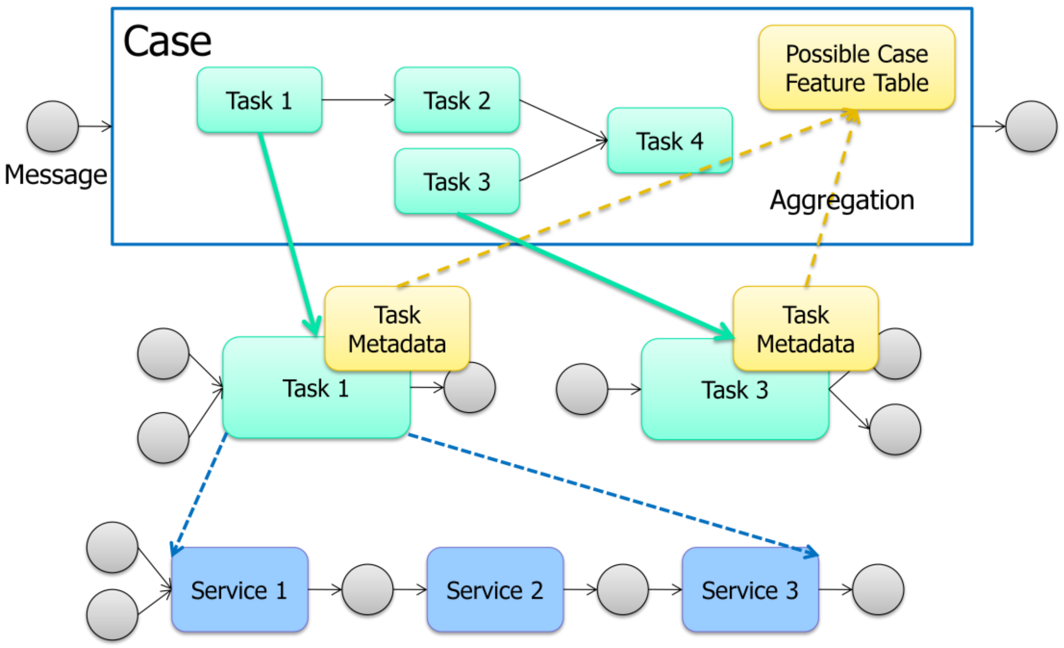


**Figure 3.5: A case schema in HCBP**

Here we defined the features for case and task description, respectively, as shown in Table 3.1 and Table 3.2. For task, there are three kinds of features in the feature table, where the first kind is for human reading, and their data type of feature value is string that can be input by designer. The second kind of feature is used to indicate input and output messages, where their data type is String that selected from the message ontology. The third kind of feature is used to record some other information about a task, their data types are String defined in category that selected from the message ontology. Case has an additional feature type that used to record information about a whole case, and the feature value data type is String defined in category.

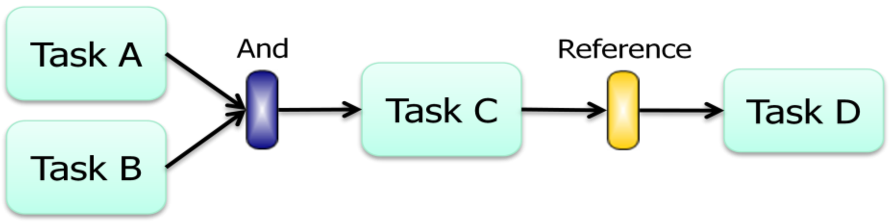
Table 3.1: Feature Types of task

| Feature Type | Data Type |
|---|---|
| Human Reading | String |
| I/O Message | String selected from Message Ontology |
| Task Feature | Category or String selected from Message Ontology |

Table 3.2: Feature Types of case

| Feature Type | Data Resource |
|---|---|
| Human Reading | Designer Input |
| Case Feature | Designer Input |
| I/O Message | Union of "Input Message" of first tasks and "Output Message" of last tasks in Desired Task Flow |
| Task Feature | Union of "Task Feature" of task in Desired Task Flow |

To generate a case feature table, the main process is to aggregate the feature values in task metadata, as shown in Table 3.2, column "Data Resource". For Input Message of a case, the feature value is union of Input Message value of leading tasks in the task flow; for Output Message of a case, the feature value is union of Output Message value of end tasks in the task flow; and for features belong to "Task Feature", value of each feature is generated by union of corresponding feature value in task metadata, as shown in Figure 3.6.



**Task A**

| Feature Type | F_Name | F_Value |
|---|---|---|
| Human Reading | $HRF_1$ | … |
| | $HRF_2$ | … |
| I/O Message | IM | Ma |
| | OM | Mb |
| Task Feature | TF1 | TFVa |
| | TF2 | TFVb |
| | TF3 | X |

**Task B**

| Feature Type | F_Name | F_Value |
|---|---|---|
| Human Reading | $HRF_1$ | … |
| | $HRF_2$ | … |
| I/O Message | IM | Mc, Md |
| | OM | Me |
| Task Feature | TF1 | TFVc, TFVd |
| | TF2 | X |
| | TF3 | TFVe |

**Task C**

| Feature Type | F_Name | F_Value |
|---|---|---|
| Human Reading | $HRF_1$ | … |
| | $HRF_2$ | … |
| I/O Message | IM | Mb, Me |
| | OM | Mf |
| Task Feature | TF1 | TFVf |
| | TF2 | TFVg, TFVh |
| | TF3 | TFVi |

**Task D**

| Feature Type | F_Name | F_Value |
|---|---|---|
| Human Reading | $HRF_1$ | … |
| | $HRF_2$ | … |
| I/O Message | IM | Mf |
| | OM | Mg |
| Task Feature | TF1 | X |
| | TF2 | X |
| | TF3 | TFVj |

**Case Feature Table**

| Feature Type | F_Name | F_Value |
|---|---|---|
| Human Reading | $HRF_1$ | … |
| | $HRF_2$ | … |
| Case Feature | $CF_1$ | CFVa, CFVb |
| | $CF_2$ | CFVc |
| I/O Message | IM | Ma, Mc, Md |
| | OM | Mg |
| Task Feature | TF1 | TFVa, TFVc, TFVd, TFVf |
| | TF2 | TFVb, TFVg, TFVh |
| | TF3 | TFVe, TFVi, TFVj |

Designer Input

**Figure 3.6: Case feature table generation**

## 3.3. Message Linking and Message Ontology

In the whole hierarchical planning process, how to make sure that the service flows retrieved from different cases could be linked together is a critical problem. Therefore, we define a message ontology to maintain the relations between messages. Assume that the similarity between two messages can be traced through the distance between their positions in the Message Ontology, similar messages will be placed under the same parent, forming a hierarchical relationship between messages. Moreover, the relations between messages can help us to check if two services or tasks could be linked together. For example, personal context is a kind of context; learner profile and learner portfolio are both personal context. If a service A has an input message type Personal Context, and a service B has an output message type Learner Profile, then we can know these two services could be linked together because Personal Context and Learner Profile are compatible by checking the relation in the ontology. The example is shown in Figure 3.7.

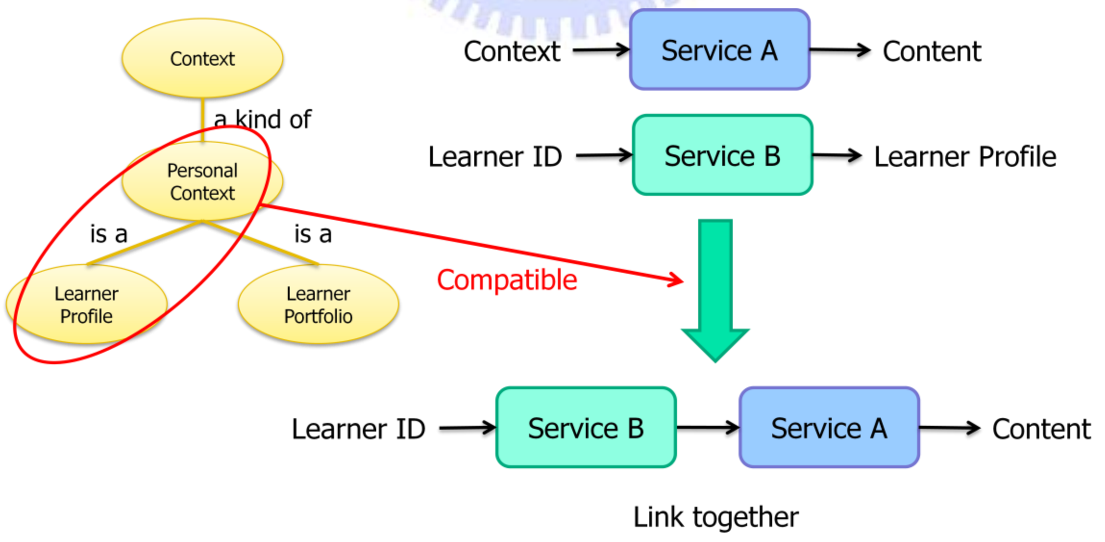


**Figure 3.7: Message linking and message ontology**

The message ontology is defined as below:

**Definition 3.8: Message Ontology**

**MO = (M, R), where**

1. **M =** {$m_1$, $m_2$, …, $m_n$} is a finite set of message in case
2. **R** = {$r_1$,$r_2$,…,$r_n$} is a finite set of relations

## 3.4. Case Retrieval

In case retrieval, similarity measurement is the main process and it determines if the retrieved cases are good enough to generate a new case satisfy designer's input. Based upon the proposed hierarchical case representation model, we propose a similarity measurement method, in which a case similarity is the combination of feature similarity and task flow similarity. As shown in Figure 3.8, after the designer provides the information about a new application, a *complete case retrieval* process is executed. It results in that the feature and task flow similarities between the desired case and cases in case base are calculated, and the most similar case is chosen. Afterward, if the retrieved case is not good enough, the system will try to adapt the case in *Case Reuse* by task or task flow Insertion, Deletion and Replacement, where the task or task flow retrieval is done by the *partial case retrieval*. However, in *case adaptation* process, if the retrieved task is not similar enough to the needed desired task, the retrieved task is revised by Message Insertion, Deletion and Service Insertion, Deletion, and Chaining according to the needed desired task in *Case Revise* process. After that, the revised task is used in the retrieved case and the case similarity is calculated again to see if it satisfies the designer's demand. This process will continue until the retrieved case adapted to the designer's demand and then returned to designer.

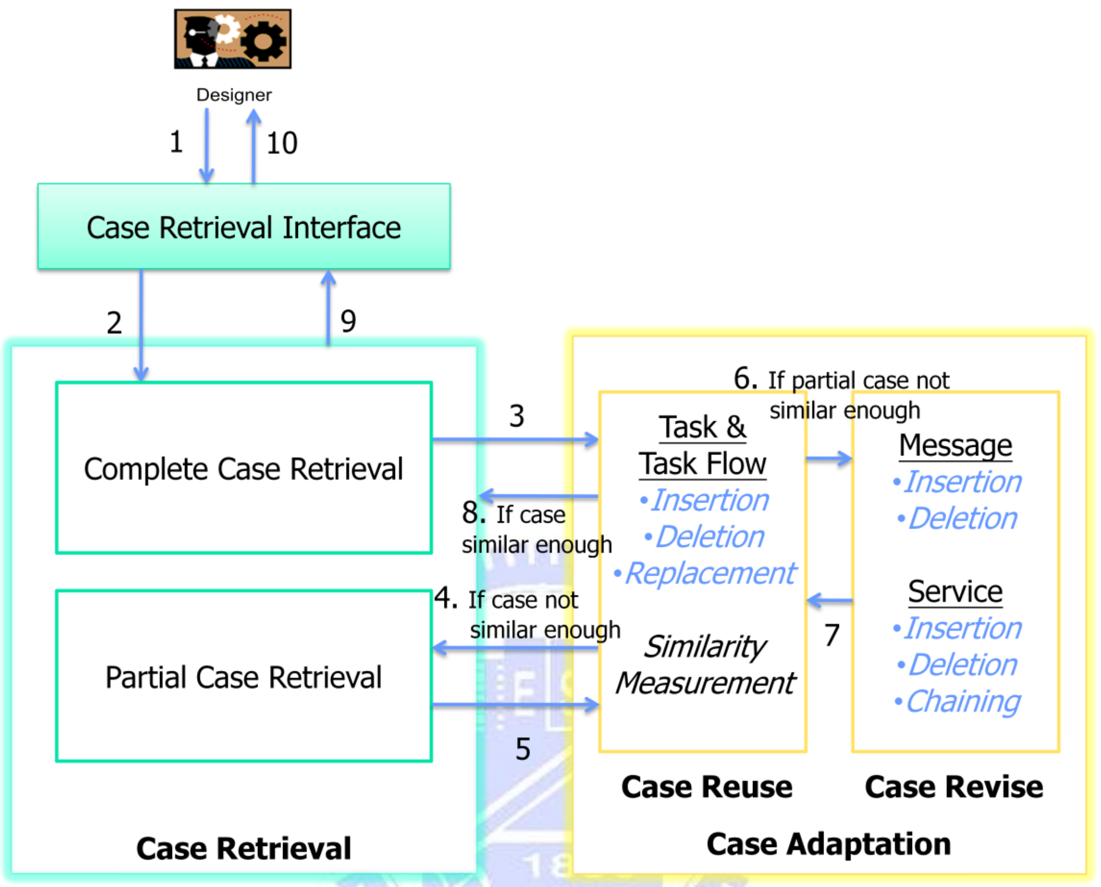


**Figure 3.8: Case Retrieval and Case Adaptation**

As mentioned in Section 3.2, despite of the features for human reading, there are only two kinds of feature value type --- String in category and String in Message Ontology. To calculate similarity of features that have feature value type "String in category", a general string matching approach is used, that is, if matched, similarity equals to 1; otherwise it equals to 0. For features that have more than one feature value, the similarity is the average of similarity of each value, as shown in Figure 3.9. To calculate similarity of features that have feature value type "String in Message Ontology", an Ontology-based similarity measurement approach is used, where the similarity between two strings is 1/(d+1), and d is the distance between them in the ontology. And for features that have more than one feature value, the similarity is the average of similarity of each value too, as shown in Figure 3.10. The definition of similarity calculation is shown as below:

**Definition 3.9 Category-based Feature Similarity**

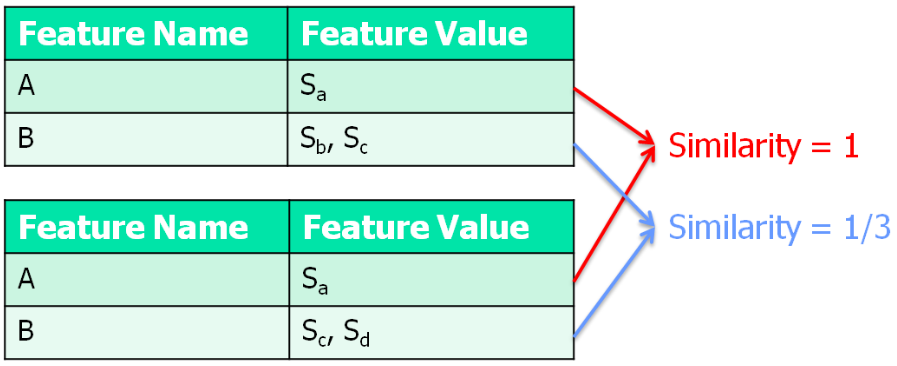
**CBFS** = $\frac{|\mathbf{FV_A \cap FV_{A'}}|}{|\mathbf{FV_A \cup FV_{A'}}|}$ (Jaccard similarity coefficient) , where

1. $FV_A$ is the Feature Value of Feature A in one Case and $FV_{A'}$ is the Feature Value of Feature A in the compared Case

**Definition 3.10 Ontology-based Feature Similarity**

**OBFS** = $\frac{\sum(\mathbf{1/(d_i+1)})}{\mathbf{FVPN}}$, where

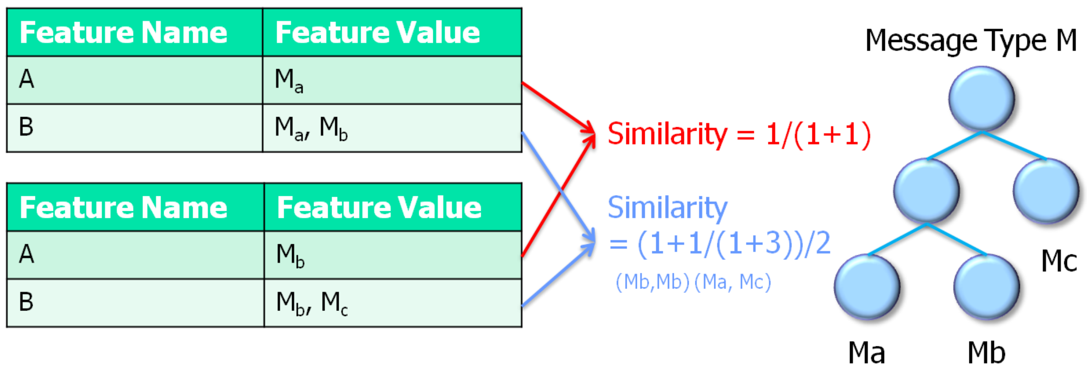
1. FVPN is the number of Feature Value Pair, which means for a Feature in two Cases A, B, if a feature value in A and a feature value in B are the same or their distance in ontology > threshold and less than any other combination, then they become a Feature Value Pair. If a value is not in any Feature Value Pair, it becomes a Pair with an "empty value".
2. $d_i$ is the distance of Feature Value Pair i



| Feature Name | Feature Value |
|---|---|
| A | $S_a$ |
| B | $S_b$, $S_c$ |

| Feature Name | Feature Value |
|---|---|
| A | $S_a$ |
| B | $S_c$, $S_d$ |

**Figure 3.9: Similarity of features value type "String in Category"**



| Feature Name | Feature Value |
|---|---|
| A | $M_a$ |
| B | $M_a$, $M_b$ |

| Feature Name | Feature Value |
|---|---|
| A | $M_b$ |
| B | $M_b$, $M_c$ |

**Figure 3.10: Similarity for features value type "String in Message Ontology"**

On the other hand, task flow similarity is composed of Coverage Similarity and Sequence Similarity[9], where the coverage means the number of similar tasks between the Desired Task Flow and the compared Task Flow in case base, and the sequence means the similarity of task order between the Desired Task Flow and the compared Task Flow. In order to calculate task flow similarity, the similar task pairs between the desired task flow and the compared flow must be found first, where the similar task pair means two tasks, one in desired task flow and the other in compared task flow, are similar enough and we take them as the same when calculating task flow similarity. According to the physical meaning of coverage, the coverage similarity is the number of similar task pair among the total number of task in desired task flow or in compared task flow (depends on different situations), defined as follows:

**Definition 3.11 Coverage Similarity**

1. **CvS** = $\dfrac{\sum \mathbf{TMatch}(\mathbf{T_x})}{\mathbf{TN}}$ is the coverage similarity between the query desired task flow and the compared case task flow, where
   - **TMatch()** is shown in **Algorithm 3.1**
   - **TN** is the total number of task in the compared task flow

**Definition 3.12 Coverage Similarity'**

1. $\mathbf{CvS' = \frac{\sum TMatch(T_x)}{TN'}}$ is the coverage similarity between the query desired task flow and the compared case task flow, where
   - **TN'** is the total number of task in the desired task flow

```
Algorithm 3.1: TMatch
Input: T
Output: Match
Definition of Symbols:
CT_i: The Compared Task i in the Compared Task Flow
Step 1. Compare T with the tasks in the case task flow, if Max(Similarity(T,CT_i)) >
        threshold, then set (T, CT_m) a Task Pair and set Match = 1, else Match = 0
Step 2. Return Match
```

And for sequence similarity, the main idea is to find similar possible sequence, where the possible sequence means the combination of every two tasks in task flow. For example, for task flow A→B→D, there are three combinational pairs A→B, B→D, and A→D. In order to calculate sequence similarity, the first thing to do is to find similar sequence pairs, which means two sequences A→B and A'→B', A→B is in desired task flow and A'→B' is in the compared task flow, where (A, A') and (B, B') are two similar task pairs. The similarity of a similar sequence pair is the average of similarity of the two similar task pair, as shown in Figure 3.11, and the overall sequence similarity for a case is the average of each similar sequence pair among all possible sequence. The definition is shown as follows:

**Definition 3.13 Sequence Similarity**

1. **SS** $= \frac{\sum \mathbf{SqMatch}(\mathbf{Sq_y})}{\mathbf{C_2^{TN}}}$ is the sequence similarity the query desired task flow and the compared case task flow, where

   - **SqMatch()** is shown in **Algorithm 3.2**
   - **Sq = (ST, DT)** is the possible sequence generate from the query desired task flow
     - **ST = {$t_1$, $t_2$, …, $t_n$},** where **$t_i$** $\in$ T is the source task in task sequence
     - **DT ={$t_1$, $t_2$, …, $t_n$},** where **$t_i$** $\in$ T is the destination task in task sequence
   - $\mathbf{C_2^{TN}}$ is the total number of possible sequence generate from the desired task flow

Algorithm 3.2: SqMatch

**Input:** Sq

**Output:** SqMatchS

**Step 1.** Find if there is a matched sequence in the compared case task flow, if found then go to step 2, else end.

**Step 2.** SqMatchS = (TSSimilarity(ST)+TSSimilarity(DT))/2

**Step 3.** Return SqMatchS

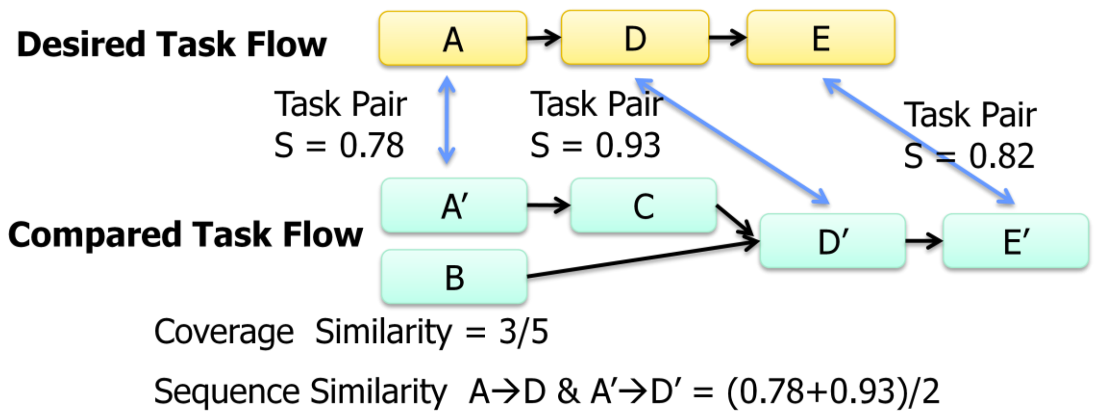


**Figure 3.11: Sequence Similarity**

In *Complete Case Retrieval*, a complete case similarity is calculated to retrieve an integral case that is most similar to the desired case, in which both Case Feature Table Similarity and Task Flow Similarity are used. The Case Feature Table Similarity indicated the similarity between the Possible Desired Case Feature Table and the Possible Case Feature Table, that is, the average similarity of each feature in the feature table. In *Partial Case Retrieval*, a Partial Case Similarity is calculated to retrieve a part of a case that is most similar to a part of Desired Task Flow, in which only Task Flow Similarity is used. The Complete Case Similarity and Partial Case Similarity are defined as below:

**Definition 3.14: Complete Case Similarity (CCS)**

1. **CCS = CFTS + TFS**, where
   - **CFTS** is the similarity of Case Feature Table
   - **TFS** is the similarity of Task Flow
2. **CFTS** = $\frac{\sum \mathbf{CFS_i} \times \mathbf{CFW_i}}{\mathbf{CFN}}$ where
   - $\mathbf{CFS_i}$ is the similarity of Case Feature i

- $\mathbf{CFW_i}$ is the weight of Case Feature i
- **CFN** is the total number of Case Feature

3. **TFS = CvS+SS**

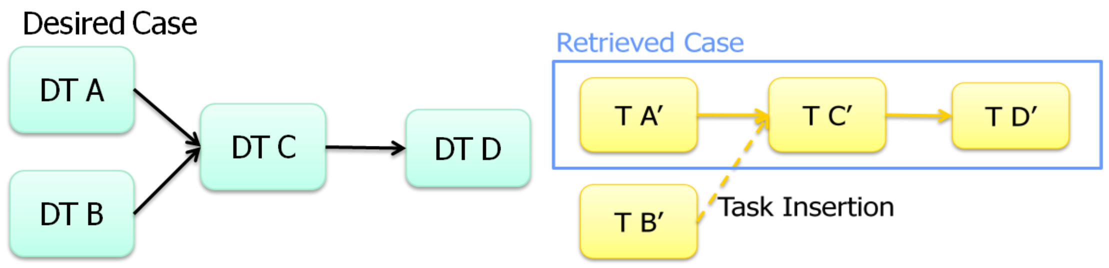
**Definition 3.15 Partial Case Similarity (PCS)**

---

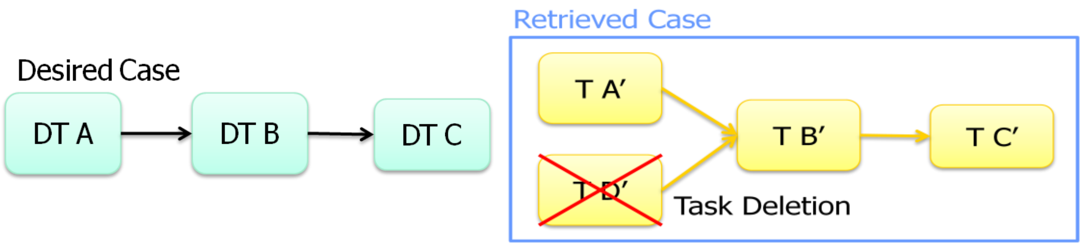
**PCS = CvS′+SS**

## 3.5. Case Adaptation

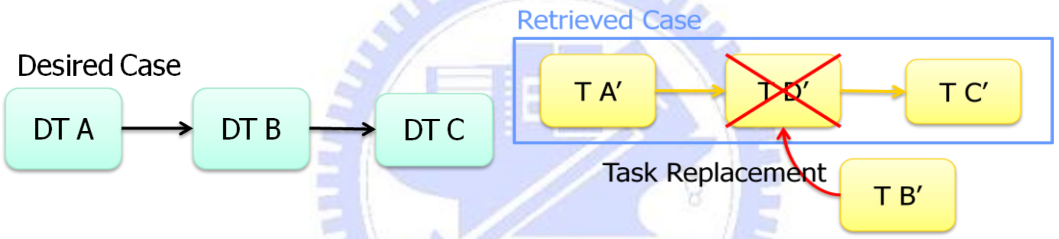
In *Case Adaptation*, there are two main processes --- *Case Reuse* and *Case Revise*, as shown in Figure 3.8. Case Reuse includes some planning operations that adapt the plan in task level, such as Task (Task Flow) Insertion, Deletion, and Replacement, where Task Insertion is an operation used to insert tasks according to the demands in the Desired Case when the retrieved case lack some tasks, as shown in Figure 3.12 (a); Task Deletion is an operation used to delete tasks according to the demands in the Desired Case when the retrieved case has some tasks unnecessary, as shown in Figure 3.12 (b); Task Replacement is an operation used to replace tasks by more suitable ones according to the demands in the Desired Case, as shown in Figure 3.12 (c).



(a) Task Insertion



(b) Task Deletion

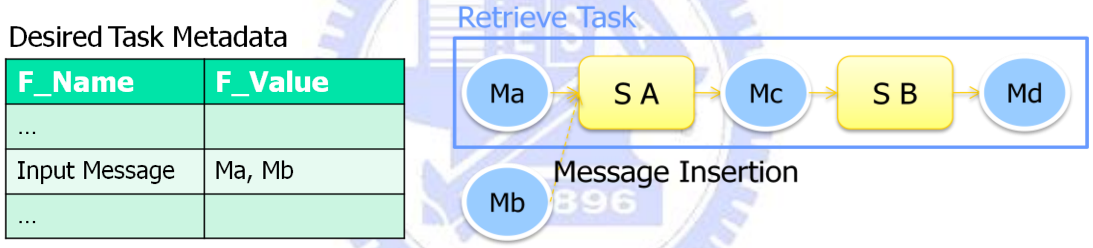


(c) Task Replacement

**Figure 3.12: Task (Task Flow) Planning Operation**

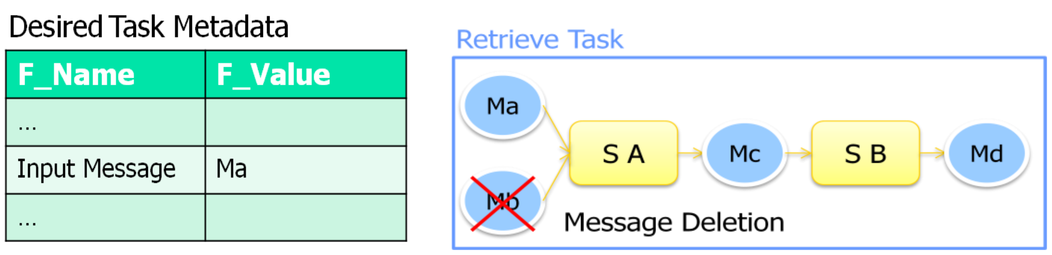
Case Revise includes planning operations that adapt the plan in Service level, such as Message Insertion, Message Deletion, Service Insertion, Service Deletion, and Service Chaining, where Message Insertion is used to add Input Message or Output Message to a Service as shown in Figure 3.13(a), and Message Deletion is used to delete Input or Output Messages that are unnecessary, as shown in Figure 3.13(b), notice that some constraints must be defined in advance to prevent from generating strange services. Service Insertion and Deletion are the operations that are used to add demanded services and to delete services that are not necessary according to the description in the Desired Task Metadata, as shown in Figure 3.14 (a)(b). Service

Chaining is the last means if no similar task in case base could be reused, this operation will try to find a service flow between Input and Output Message by support of Message Ontology, as shown in Figure 3.14 (c). For example, to find a service flow between Input Message A and Output Message B, the system will select a service S that has Output Message B first, and then try to find another service has the Output Message Type that is compatible to the Input Message Type of service S, as shown in Example 3.3. This process will continue until a service flow is found to connect Input Message A and Output Message B, or the process is failed if there is no service flow could be found and it means that new task or new service has to be added into repository.



| F_Name | F_Value |
|---|---|
| … | |
| Input Message | Ma, Mb |
| … | |

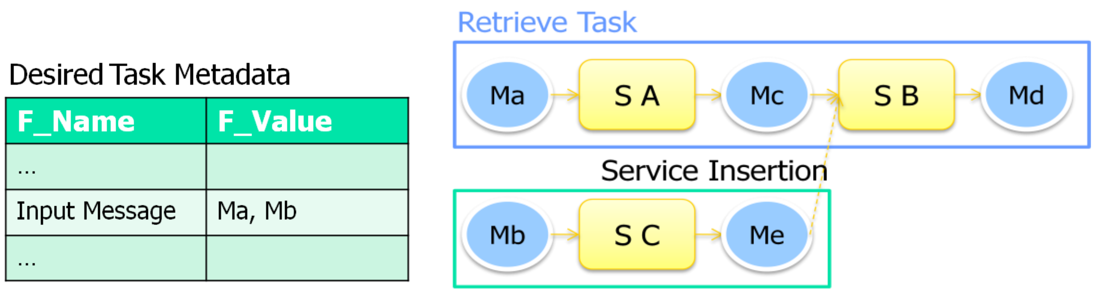
**(a) Message Insertion**



| F_Name | F_Value |
|---|---|
| … | |
| Input Message | Ma |
| … | |

**(b) Message Deletion**

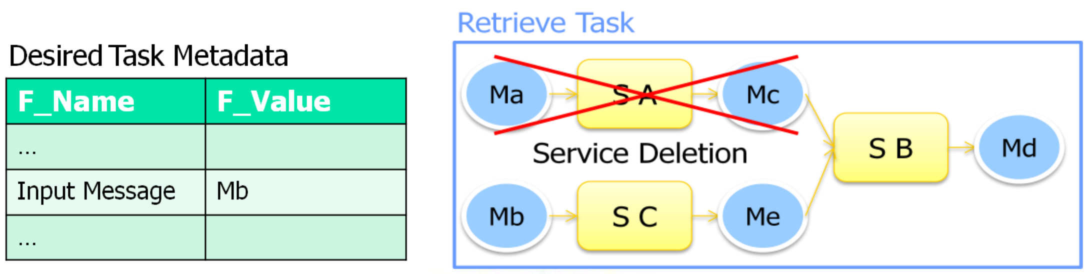
**Figure 3.13: Message Planning Operation**



Desired Task Metadata

| F_Name | F_Value |
|---|---|
| … | |
| Input Message | Ma, Mb |
| … | |

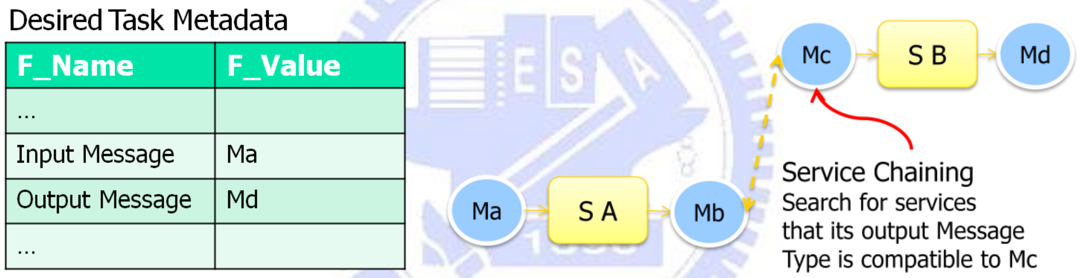
(a) Service Insertion



Desired Task Metadata

| F_Name | F_Value |
|---|---|
| … | |
| Input Message | Mb |
| … | |

(b) Service Deletion


Mc
S B
Md
Service Chaining
Search for services that its output Message Type is compatible to Mc
Ma
S A
Mb

Desired Task Metadata

| F_Name | F_Value |
|---|---|
| … | |
| Input Message | Ma |
| Output Message | Md |
| … | |

(c) Service Chaining

**Figure 3.14: Service Planning Operation**

In the whole Case Adaptation process, similarity between the retrieved case and designer's input is confirmed first. If the retrieved case is similar enough, then no adaptation has to be done, else Case Reuse process will compare the Task Flow of the retrieved case and the Desired Task Flow from the designer. Afterward, the most dissimilar part will be found, where the dissimilar part could be a single Task or a Task Flow, and then planning operations in task level is executed to adapt the retrieved case to designer's input. If Task Insertion or Task Replacement is needed, the Partial Case Retrieval process mentioned in last section is executed to find Task or

Task Flow that are similar to the query Desired Tasks or Desired Task Flow for Case Adaptation. However, if the retrieved Task or Task Flow itself is not similar enough to the query Desired Task or Desired Task Flow, it will be helpless to insert or replace the retrieved Task or Task Flow in the retrieved Case. Therefore, a Task itself must be adapted first before inserted or replaced in the retrieved Case, that is, the Case Reuse process is executed to adapt the retrieved Task to the query Desired Task. In which the retrieved Task is revised by the service level planning operations mentioned above according to the Desired Task Metadata. Afterward, the adapted Task could be used for Task Insertion and Task Replacement. On the other hand, for the retrieved Task Flow that itself is not similar enough, the Desired Task Flow is decomposed into single Desired Tasks and corresponding Tasks are retrieved respectively. The one that has the highest similarity among the all retrieved Tasks is chosen, and then it will be inserted or replaced in the retrieved Case if its similarity is high enough, or it will be adapted by Case Revise first, and then used for Task Insertion and Task Replacement. The adaptation process will be repeated until the retrieved Case reaches an acceptable status. The algorithm of Case Reuse and Case Revise is shown as follows:

**Algorithm 3.3: Case Reuse**

```
Input: Retrieved Case, Designer's Input(PDCFT, DTF, DTMDs of Desired Tasks in DTF)
Output: Adapted Case
Definition of Symbols:
PoDTF: part of DTF
RTN: Task number of the retrieved case
DTN: Task number of the desired case
Step 1: Test if similarity of the Retrieved Case > a threshold.
        Yes → Return the Retrieved Case
Step 2: Find PoDTF that is most dissimilar to Retrieved Case
Step 3: Call PartialCaseRetrieval(PoDTF, DTMDs of PoDTF)
Step 4: Test if Similarity of the Retrieved Partial Case similar > a threshold.
        Yes → Do Operations according to Designer's Input.
            ● If some Desired Tasks have no similar Task Pair, Task Insertion is executed.
            ● Else If the ((RTN >DTN) & CS is low) or ((RTN > DTN) & SS is low), Task Deletion is executed
            ● Else If TS of an Similar Task Pair < threshold, Task Replacement is executed
            ● go to Step 1
        No → If Partial Case is a Desired Task Flow, go to Step 5.
             If Partial Case is a Desired Task, go to Step 6
Step 5: Retrieve a similar task for each Desired Task in the flow, select the most similar
        one, then go to Step 4.
Step 6: Call Revise(Retrieved Task, corresponding DTMD), then go to Step 4 →Yes.
```

Algorithm 3.4: Case Revise

**Input:** Retrieved Task, corresponding DTMD

**Output:** Adapted Task

**Step 1:** Do Operations to the Retrieved Task according to DTMD

- If the retrieved Task in a Similar Task Pair lack I/O Message which is in the Desired Task, *Message Insertion* is executed
- If the retrieved Task in a Similar Task Pair has extra I/O Message which is not in the Desired Task, *Message Deletion* is executed
- If the retrieved Task in a Similar Task Pair lack some Task Feature Value which is in the Desired Task, *Service Insertion* is executed
- If the retrieved Task in a Similar Task Pair has extra Task Feature Value which is not in the Desired Task, *Service Insertion* is executed
- If Task Similarity < a threshold, Service Chaining is executed

**Step 2:** Test if the Adapted Task similar enough (Similarity > threshold)

**Yes** ➔ Return

**No** ➔ Use Service Chaining to generate a plan, in which its Input & Output Messages are conform to description in DTMD.

# Chapter 4. U-learning Application HCBP System

In this Chapter, we apply the proposed HCBP scheme to U-learning.

## 4.1. Case Hierarchy for U-learning Application

As mentioned before, a case is represented as three layers of hierarchical-case representation model, where the first layer is the application layer of representing a case in the case base; the second layer is the task layer and the third layer is the services layer.

Based on our observation, most U-learning applications nowadays have the same process model; that is, at the beginning of an application, the system will collect context information that is needed in the environment first, afterwards the collected contexts are used to help retrieve the most suitable content for the learner, and at last, the retrieved content is displayed to the learner in varies ways. In order to simplify the Desired Task Flow construction process for application designer, we divided tasks into three kinds --- Context Collection & Interpretation, Context Retain &Content Retrieval and Output, as shown in Figure 4.1. Context Collection & Interpretation is a group of tasks that are used to gather contexts from the environment and interpret the gathered context according to different use; Context Retain & Context Retrieval include tasks that retain the gathered context in repository and tasks that retrieve context-aware content; Output is a group of tasks that are used for adaptable content display to learners according to the situation of an environment. These three kinds of tasks are used for a designer to design a desired task flow of a new application.

As shown in Figure 4.2, a museum guiding example is given to illustrate the hierarchical-case representation model for U-learning. In the museum guiding application, in order to provide a learner with suitable information about an exhibition according to the learner's ability, there are two tasks in type Context Collection & Interpretation, "Get Learner Ability" and "Get Exhibition Identity", where in type "Context Retain & Content Retrieval", a task "Retrieve Learner-Ability-related content" is needed to retrieve content about the exhibition that suitable for the learner. At last, a task "Display Learner-Ability-related Content" is in type Output to show the content to learner by web pages. Furthermore, each task is mapping to a flow of services. Take "Get Learner Ability" for example, it is composed of three services, at first the learner's identity is detected by a "Detect Learner ID" service, and then the learner ID is used to retrieve Learner Portfolio in repository by a service "Get Learner Portfolio", and finally, Learner Ability is determined by a service "Ability Judgment".

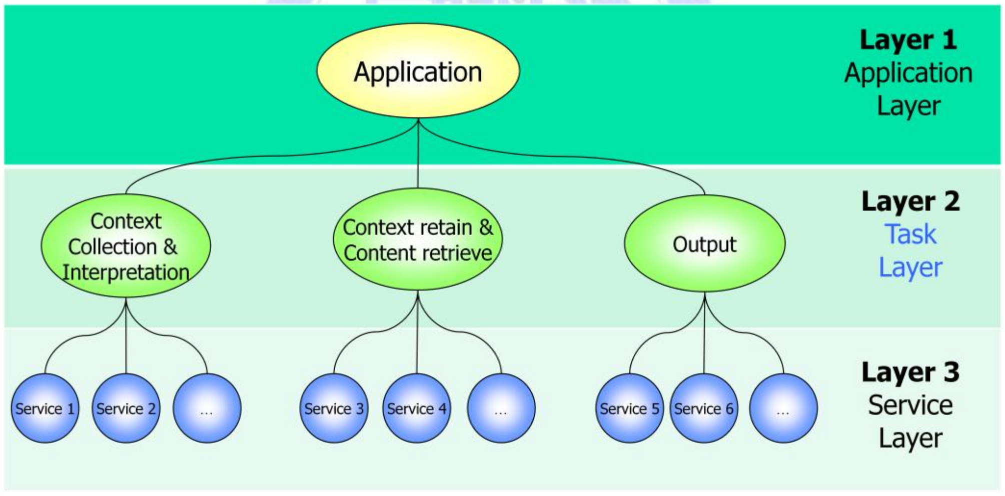


**Figure 4.1: Case Hierarchy**

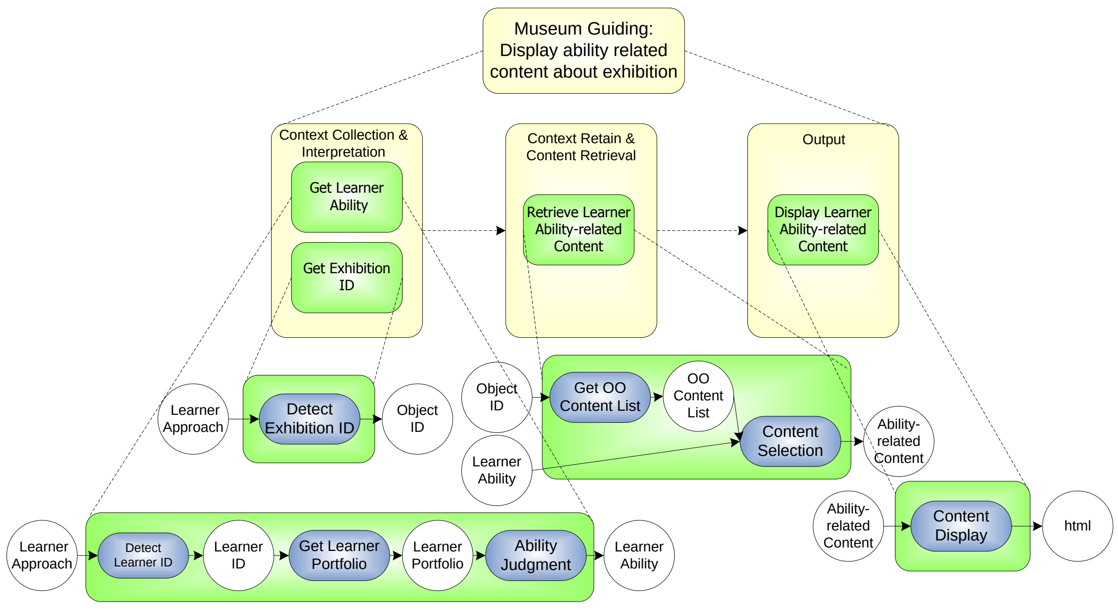


**Figure 4.2: Case representation for museum guiding**

## 4.2. Message Ontology

In order to utilize the message ontology to support case retrieval and planning for U-learning application, we designed a base skeleton of the message ontology aiming at U-learning.

We divided messages into four kinds, external message, content message, context message and computing messages, respectively, where external message denotes messages accepted from real environment; for example, weather of an environment, learner approach, physical location, etc. Content message denotes the contents that are suitable to be displayed of to learner, for example, map, context-aware content, alert, etc. Context message denotes context that provides information about learner and environment, such as learner profile, location, time, etc. Computing message denotes the communication messages used between some components in the system, such as a dump query to a learner profile repository. In Figure 4.3, a message ontology constructed from a weather information probing application and a museum guiding application is demonstrated, in which the nodes that written in bold text are the base skeleton we designed for U-learning, and the other nodes are inserted into the ontology during application analysis. Notice that when designing a new application, the designer can add new message nodes into the message ontology if there is no proper existing message to use.

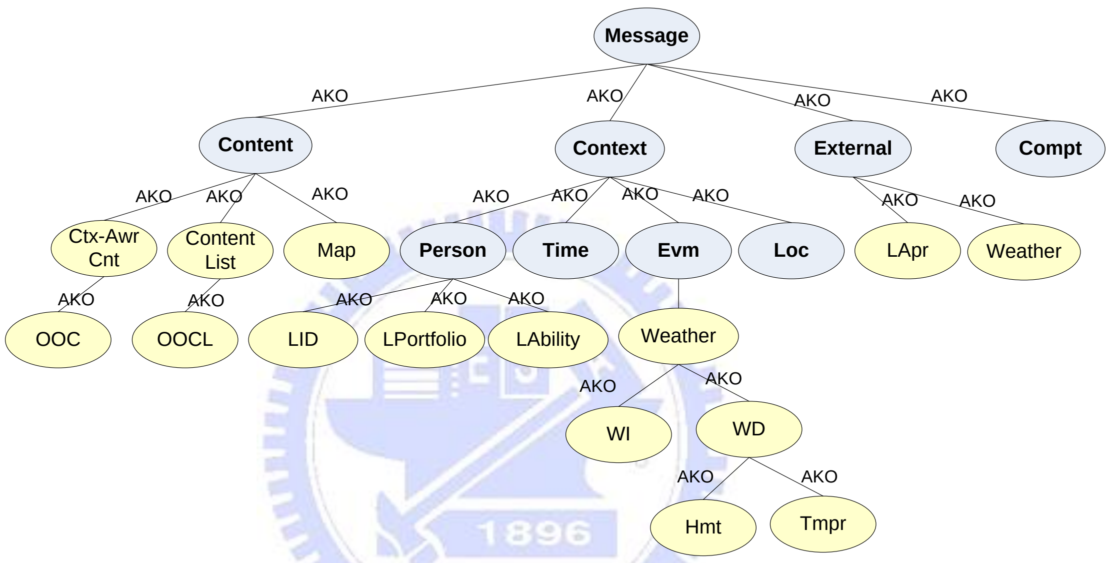


**Figure 4.3: Message Ontology constructed from two applications**

## 4.3. Features for U-learning

As mentioned in previous chapter, each Task contains a TMD for task description and each case contains a PCFT for case description. Both TMD and PCFT are a set of Feature-Value pairs, and some feature values in PCFT are generated from TMD as illustrated in Section 3.2. Here we define a set of Feature for TMD and PCFT based on demands of U-learning, respectively. Notice that DTMD and PDCFT are almost the same with TMD and PCFT despite that they are input by designer for query. Table 4.1 shows the Features that we defined for Task description, where

“Name” and “Goal Specification” are the Features belong to type Human Reading defined in Section 3.2 and their values are strings inputted by designer arbitrarily; “Input Message Type” and “Output Message Type” belong to I/O Message and their values are selected from the Message Ontology; “Collect”, “Retain”, ”ContentRelated”, “Target” and “ContextCollectionType” are Features belong to type Task Feature, where “Collect” denotes what contexts should be collected by the output of a task; “Retain” denotes what kind of information should be retained in a repository such as learner’s location; “ContentRelated” denotes what kind of contexts are used to retrieved content; “Target” denotes whom should the content displayed to, such as a learner or a learner’s supervisor; “ContextCollectionType” denotes the needed context are collected actively by the system or input by learner directly. Table 4.2 is the Features defined for Case description, where there are four additional Features defined. “Interaction” denotes the interaction type in the application, such as interaction between people or between a person and an object; “Environment” denotes surrounding information such as indoor or outdoor; “Network” denotes the available network in the environment of the application, such as Bluetooth, 802.11, WiMax, etc. It is worth to mention that these Features may be insufficient for new complex applications, and new Features could be added into the Feature Table if there is a demand.

Table 4.1: Features for Task Description

| Feature Name | Data Type | Possible Value |
|---|---|---|
| Name | String | X |
| Goal Specification | String | X |
| Input Message Type | String | Select from message ontology |
| Output Message Type | String | Select from message ontology |
| Collect | String | Select from message ontology |
| Retain | String | Select from message ontology |
| ContentRelated | String | Select from message ontology |
| Target | String(Category) | Learner, supervisor |
| ContextCollectionType | String(Category) | Active, passive |

Table 4.2: Features for Case Description

| Feature Name | Data Type | Possible Value |
|---|---|---|
| Name | String | X |
| Goal Specification | String | X |
| Input Message Type | String | Select from message ontology |
| Output Message Type | String | Select from message ontology |
| Collect | String | Select from message ontology |
| Retain | String | Select from message ontology |
| ContentRelated | String | Select from message ontology |
| Target | String(Category) | Learner, supervisor |
| ContextCollectionType | String(Category) | Active, passive |
| Interaction | String(Category) | Person-person, person-location, person-object |
| Environment | String(Category) | Indoor, Outdoor |
| Network | String(Category) | Bluetooth, 802.11, WiMax, ... |

# Chapter 5. Application and Discussion

In this Chapter, a new U-learning application --- Botanical Garden is designed by means of the proposed HCBP scheme.

## 5.1 Botanical Garden

Assume that there is a botanical garden, in which there are about a hundred kinds of plants. For each kind of plants there is a RFID Tag used to identify the plants, and each learner in the garden use a PDA that has RFID Reader function. When a learner approaches some plant, the RFID Reader will trigger the RFID Tag to get the ID of the plant for further use such as searching for learning content in repository about the plant.

To design a U-learning application, a designer will imagine a scenario in his or her mind first. Suppose that if there is a scenario of botanical garden in a designers mind, such as "when a learner approaches a plant in the botanical garden, get the learner's ability and detect weather type first, and retrieve content about the plant according to the learner's ability and the weather type, then display the retrieved content to the learner". To construct the task flow of the botanical garden guiding application, the designer is asked to construct the Desired Tasks belonging to each task type first, as shown in Figure 5.1. Afterward, the designer is asked to construct the relations between the desired tasks, as shown in Figure 5.2.

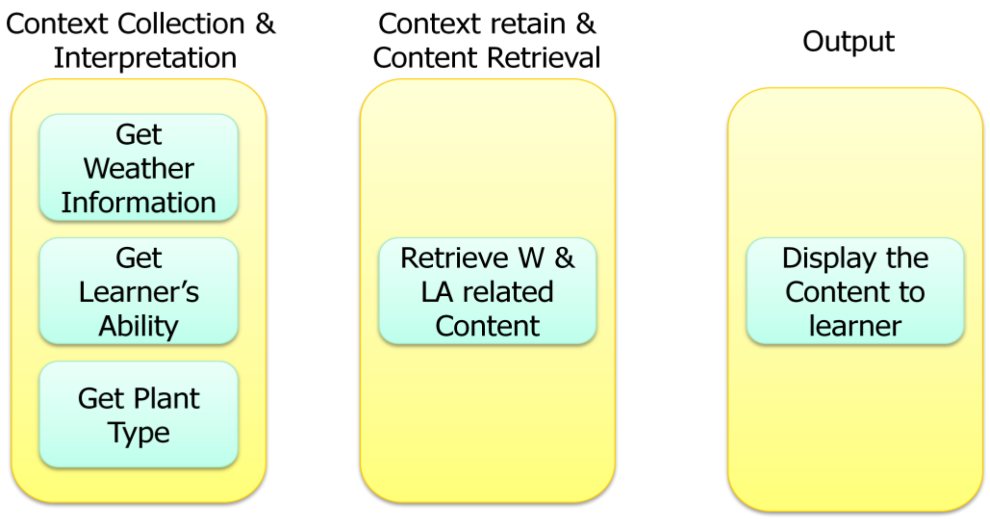

Context Collection & Interpretation
Get Weather Information
Get Learner's Ability
Get Plant Type
Context retain & Content Retrieval
Retrieve W & LA related Content
Output
Display the Content to learner

**Figure 5.1: Desired Tasks**

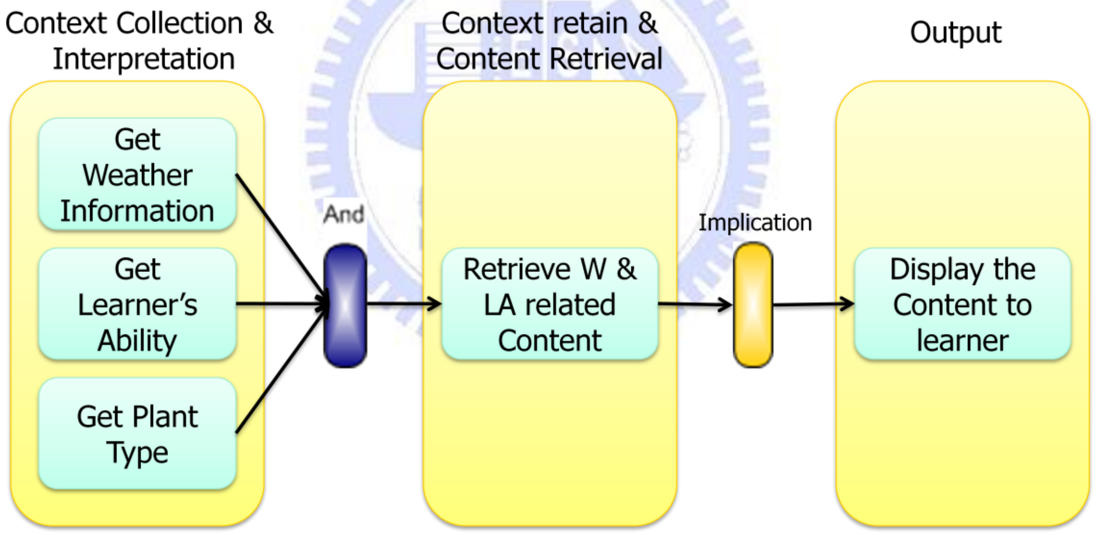

Context Collection & Interpretation
Get Weather Information
Get Learner's Ability
Get Plant Type
And
Context retain & Content Retrieval
Retrieve W & LA related Content
Implication
Output
Display the Content to learner

**Figure 5.2: Desired Task Flow**

Next, the Desired Task Metadata has to be filled for each desired task and a several feature values that describe the whole application, as shown in Figure 5.3, and then the Possible Desired Case Feature Table is generated by aggregating the tables in Figure 5.3 as illustrated in Section 3.2, and the result table is shown in Figure 5.4.

| Feature Name | Feature Value |
|---|---|
| Name | Get Weather Information |
| Goal Specification | Get the weather type |
| Input Message Type | Weather |
| Output Message Type | Weather Information |
| Collect | Humidity, Temperature |
| Retain | X |
| ContentRelated | X |
| Target | X |
| ContextCollectionType | Active |

| Feature Name | Feature Value |
|---|---|
| Name | Get Learner's Ability |
| Goal Specification | Judge the learner's ability |
| Input Message Type | Learner Approach |
| Output Message Type | Learner Ability |
| Collect | Learner Profile |
| Retain | X |
| ContentRelated | X |
| Target | X |
| ContextCollectionType | Active |

| Feature Name | Feature Value |
|---|---|
| Name | Get Plant Type |
| Goal Specification | Detect the plant's type |
| Input Message Type | Learner Approach |
| Output Message Type | Object ID |
| Collect | X |
| Retain | X |
| ContentRelated | X |
| Target | X |
| ContextCollectionType | Active |

| Feature Name | Feature Value |
|---|---|
| Name | Retrieve W & LA related Content |
| Goal Specification | Retrieve proper content about .. |
| Input Message Type | Weather, Learner Ability, Object ID |
| Output Message Type | Context-aware Content |
| Collect | X |
| Retain | X |
| ContentRelated | Weather, Learner Ability |
| Target | X |
| ContextCollectionType | X |

| Feature Name | Feature Value |
|---|---|
| Name | Display the Content to learner |
| Goal Specification | Display the content … |
| Input Message Type | Context-aware Content |
| Output Message Type | Context-aware Content |
| Collect | X |
| Retain | X |
| ContentRelated | X |
| Target | Learner |
| ContextCollectionType | X |

| Feature Name | Feature Value |
|---|---|
| Name | Botanical Garden Guiding |
| Goal Specification | When a learner approach a plant, display content about the plant according the learner ability and weather |
| Interaction | Person-Object |
| Environment | Outdoor |
| Network | Wimax |

**Figure 5.3: Desired Task Metadata**

| Feature Name | Feature Value |
|---|---|
| Name | Botanical Garden Guiding |
| Goal Specification | When a learner approach a plant, display content about the plant according the learner ability and weather |
| Input Message Type | Weather, Learner Approach |
| Output Message Type | Context-aware Content |
| Collect | Humidity, Temperature, Learner Profile |
| Retain | X |
| ContentRelated | Weather, Learner Ability |
| Target | Learner |
| ContextCollectionType | Active |
| Interaction | Person-Object |
| Environment | Outdoor |
| Network | Wimax |

**Figure 5.4: Possible Desired Case Feature Table**

After the PDCFT is generated, a most similar case in the case base could be retrieved by the PDCFT and the DTF, as shown in Figure 5.5. To simplify the calculation, assume that each Feature Weight here is the same, the retrieved case is shown in Figure 5.6.

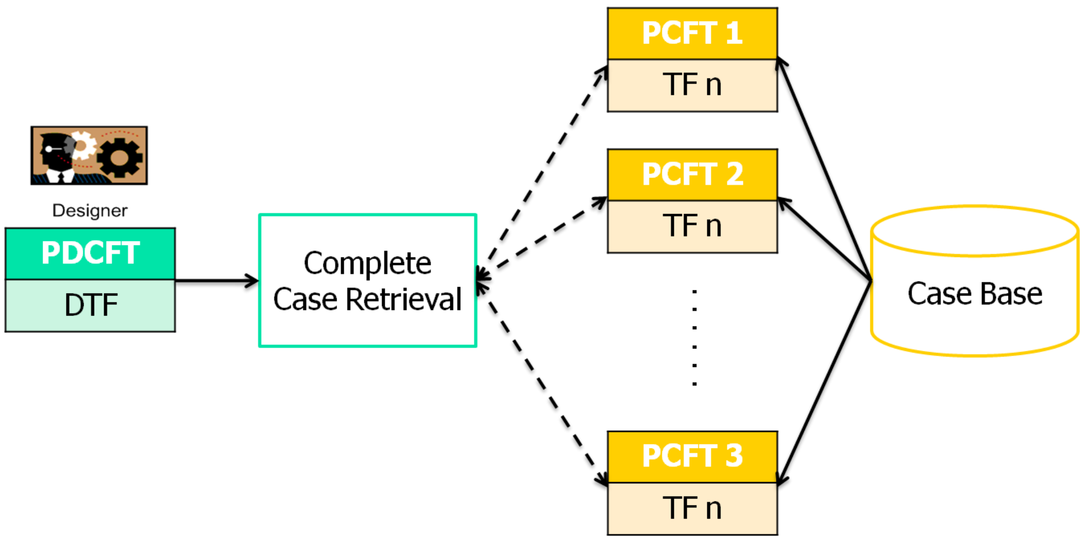
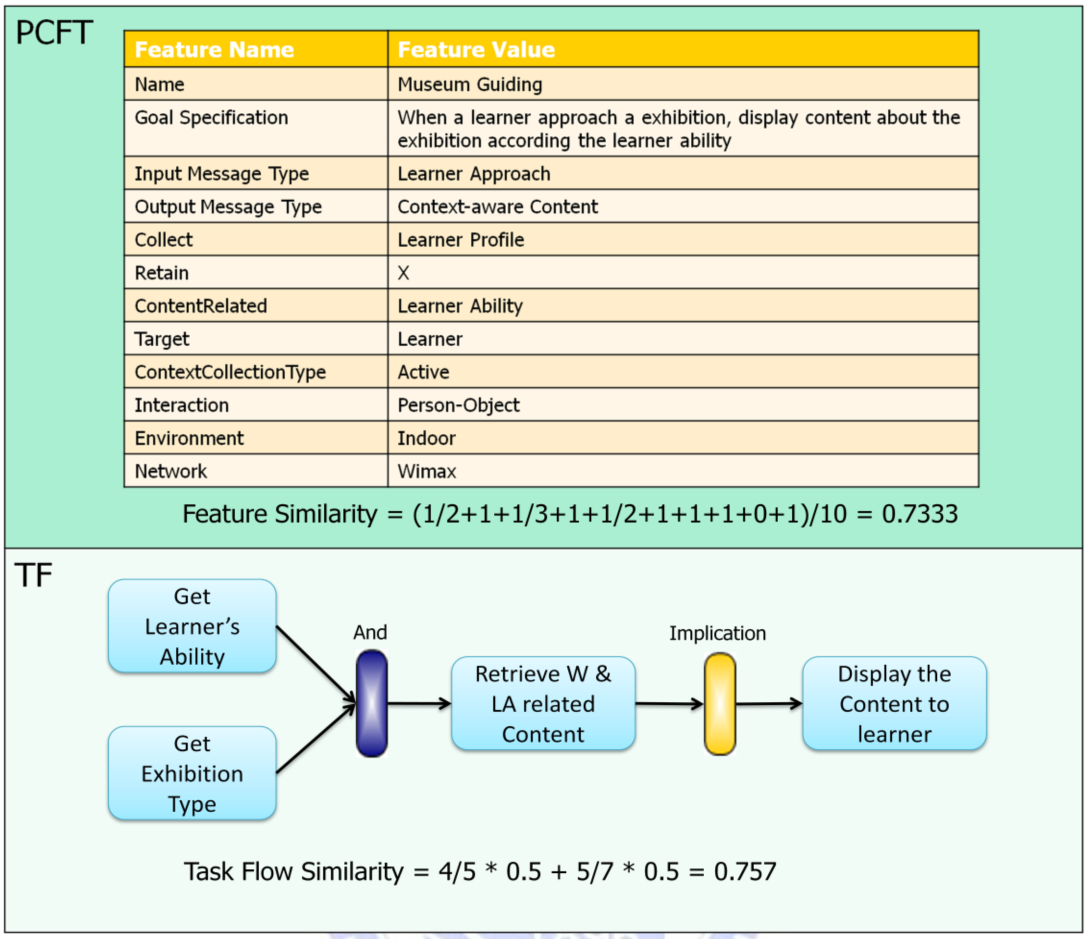


**Figure 5.5: Complete Case Retrieval**


PCFT

| Feature Name | Feature Value |
|---|---|
| Name | Museum Guiding |
| Goal Specification | When a learner approach a exhibition, display content about the exhibition according the learner ability |
| Input Message Type | Learner Approach |
| Output Message Type | Context-aware Content |
| Collect | Learner Profile |
| Retain | X |
| ContentRelated | Learner Ability |
| Target | Learner |
| ContextCollectionType | Active |
| Interaction | Person-Object |
| Environment | Indoor |
| Network | Wimax |

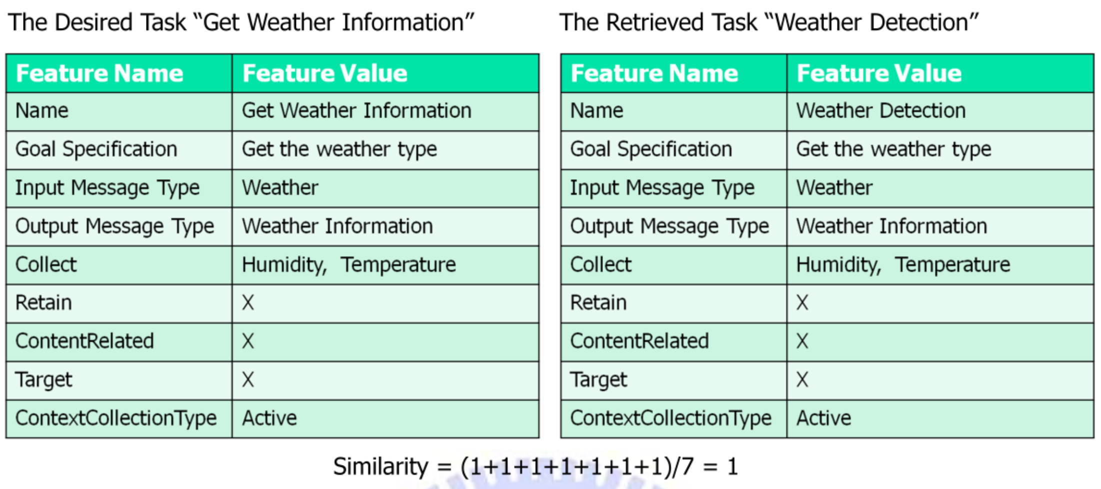
Feature Similarity = (1/2+1+1/3+1+1/2+1+1+1+0+1)/10 = 0.7333

TF
Get Learner's Ability
Get Exhibition Type
And
Retrieve W & LA related Content
Implication
Display the Content to learner

Task Flow Similarity = 4/5 * 0.5 + 5/7 * 0.5 = 0.757

**Figure 5.6: The Retrieved Case**

By comparing the Desired Task Flow and the Task Flow of the retrieved case, we can discover that one more task similar to “Get Weather Information” has to be inserted into the retrieved case. Therefore, the system will try to retrieve a partial case by the DTMD of the Desired Task “Get Weather Information”. The most similar one retrieved is shown in Figure 5.7, and no adaptation to the retrieved Task has to be carried out. After that, there is still one problem, that is, the Input Message of the Desired Task “Retrieve W & LA related Content” and Task “Retrieve LA related Content” in the retrieved case is different as shown in Figure 5.8; this will make tasks unable to link together. Therefore, an operation “Message Insertion” is executed and the result is shown in Figure 5.9. Afterward, operation “Task Insertion” could be carried and the finally, resulting Case is shown in Figure 5.10.



The Desired Task "Get Weather Information"

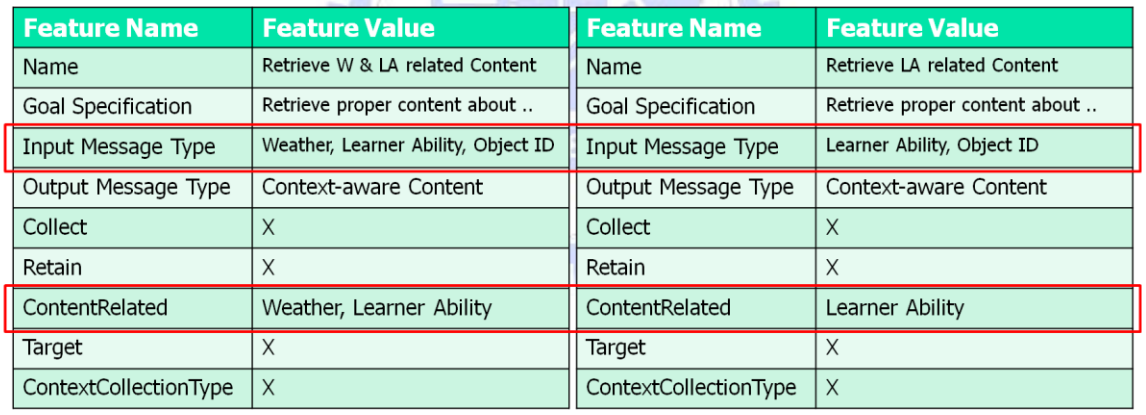
| Feature Name | Feature Value |
|---|---|
| Name | Get Weather Information |
| Goal Specification | Get the weather type |
| Input Message Type | Weather |
| Output Message Type | Weather Information |
| Collect | Humidity, Temperature |
| Retain | X |
| ContentRelated | X |
| Target | X |
| ContextCollectionType | Active |

The Retrieved Task "Weather Detection"

| Feature Name | Feature Value |
|---|---|
| Name | Weather Detection |
| Goal Specification | Get the weather type |
| Input Message Type | Weather |
| Output Message Type | Weather Information |
| Collect | Humidity, Temperature |
| Retain | X |
| ContentRelated | X |
| Target | X |
| ContextCollectionType | Active |

Similarity = (1+1+1+1+1+1+1)/7 = 1

**Figure 5.7: The Retrieved Task**



| Feature Name | Feature Value |
|---|---|
| Name | Retrieve W & LA related Content |
| Goal Specification | Retrieve proper content about .. |
| Input Message Type | Weather, Learner Ability, Object ID |
| Output Message Type | Context-aware Content |
| Collect | X |
| Retain | X |
| ContentRelated | Weather, Learner Ability |
| Target | X |
| ContextCollectionType | X |

| Feature Name | Feature Value |
|---|---|
| Name | Retrieve LA related Content |
| Goal Specification | Retrieve proper content about .. |
| Input Message Type | Learner Ability, Object ID |
| Output Message Type | Context-aware Content |
| Collect | X |
| Retain | X |
| ContentRelated | Learner Ability |
| Target | X |
| ContextCollectionType | X |

**Figure 5.8: Difference Between Desired Task “Retrieve W & LA related Content” and Task “Retrieve LA related Content”**

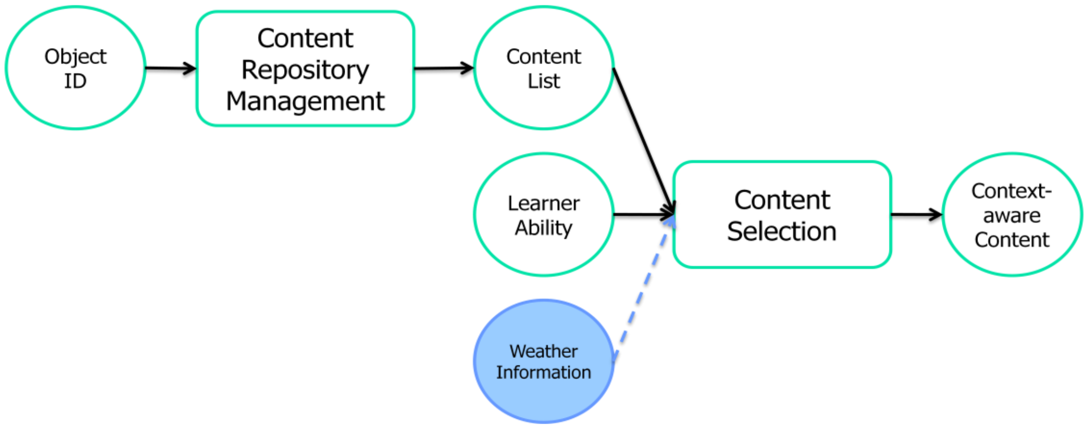


**Figure 5.9: Message Insertion**

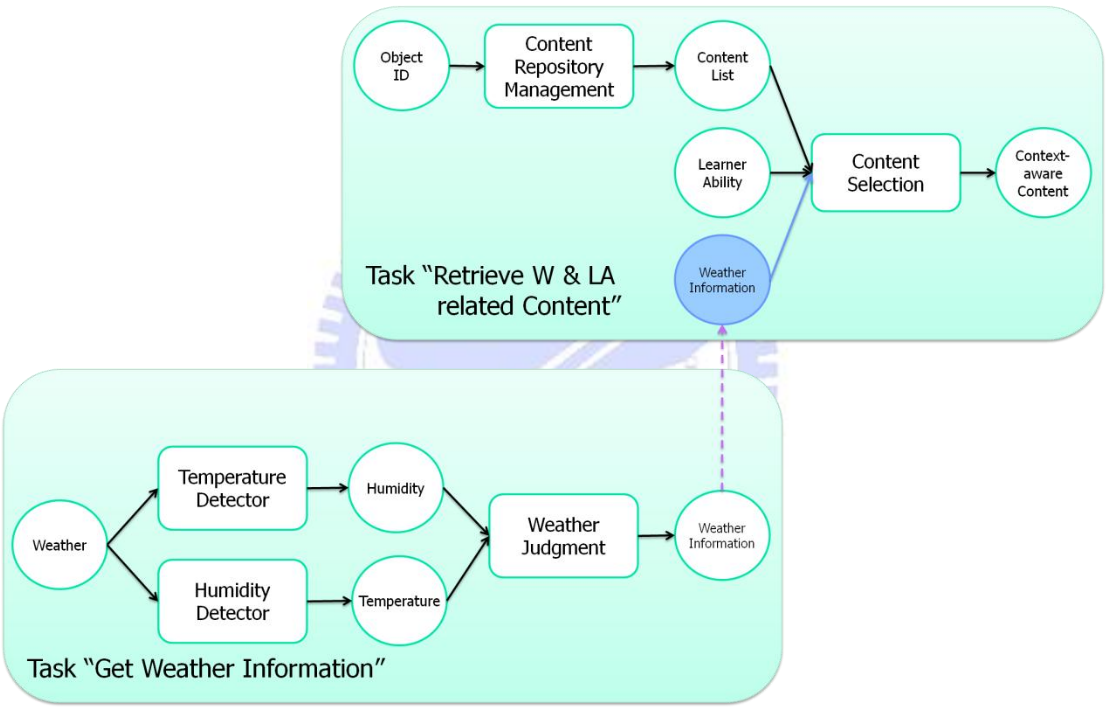


**Figure 5.10: Task Insertion**

In this section, we give a new Botanical Garden Guiding application, and we apply the proposed HCBP scheme to retrieve a plan for the Botanical Garden Guiding application. The resulting plan is a combination of two U-applications, a museum guiding application and a weather type detection application, and the result shows that it is rational and applicable.

# Chapter 6. Conclusion

In this thesis, we propose a Hierarchical-Case-based Planning scheme, in which we design a hierarchical-case representation model, a hierarchical-case retrieval and case adaptation approach to support fine-grained case reuse. In our approach, ontology is used to support case-based planning processes, where an ontology-based case similarity measurement is utilized to retrieve similar case in case base, and an ontology-based service or tasks chaining approach is proposed to sustain the planning process. Afterwards, we apply the HCBP scheme to support U-learning application design, and a message ontology is constructed for U-learning. With the HCBP, an application designer can reuse the design ideas in existing systems to construct a new application, by means of constructing a desired task flow in his or her mind and filling in some tables to retrieve similar cases in the case base.

In the near future, we will try to enhance the ability to handle more complex relations between tasks and then designers can create complicated new applications. On the other hand, we will also try to adopt heuristic adaptation rule in Case Adaptation process, such as in order to get learner's location, GPS is better when in an outdoor scenario and RFID is more suitable in an indoor scenario, to enhance the reliability of the resulting plan.

# Reference

[1] P. Alexander, J. Holtman, G. Minden, "Case-Based Planning for Simulation," presented at Expert Planning Systems, Brighton, 1991.

[2] Z. Cheng, S. Sun, M. Kansen, et al., "A Proposal on a Learner's Context-aware Personalized Education Support Method based on Principles of Behavior Science," presented at Internation Conference on Advanced Information Networking and Application (AINA), 2006.

[3] J. Cho, "An Exhibition Reminiscent System for Ubiquitous Environment," presented at IEEE International Conference on Computer and Information Technology (CIT), 2006.

[4] K. Hammond, "Case-Based Planning: Viewing Planning as a Memory Task," presented at Academic Press, 1989.

[5] Gwo-Jen Hwang, "Criteria and Strategies of Ubiquitous Learning", IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2006

[6] J. Kolodner, Simpson, R., and Sycara, K., "A process model of case-based reasoning in problem solving," presented at proceedings of the Ninth IJCAI, 1985.

[7] Goro Kunito, Naoharu Yamada, Tatsuo Takakashi, "Architecture for Providing Services in the Ubiquitous Computing Environment," presented at IEEE Internal Conference on Distributed Computing Systems Workshops (ICDCSW), 2006.

[8] Luyi Li, Yanlin Zheng, Hiroaki Ogata, et al., "A Framework of Ubiquitous Learning Environment" Computer and Information Technology, 2004

[9] Yi-Huang Lin, Shian-Shyong Tseng, "A Recommendation Scheme of Personalized Learning Activities Based on Learning Design Standard", 2005

[10] T. Madhusudan, B. Marshall, "A case-based reasoning framework for workflow model management," *Data & Knowledge Engineering* vol. 50, pp. 87 - 115 2004.

[11] H. O., Y. Y, Moushitr M. El-Bishouty, "Personalized Knowledge Awareness Map in Computer Supported Ubiquitous Learning," presented at International Confernece on Advanced Learning Technologies (ICALT), 2006.

[12] Reinhard Oppermann, Marcus Specht, "Adaptive mobile museum guide for information and learning on demand," Human Computer Interaction (HCI), 1999.

[13] C. Riesbeck, and Schank, R., *Inside Case-Based Reasoning*: Erlbaum, Hillsdale, NJ, 1989.

[14] K. Sakamura, "Ubiquitous Computing Technologies for Ubiquitous Learning," presented at IEEE International Workshop on Wireless and Moblie Techinologies in Education (WMTE), 2005.

[15] B. Smyth, Mark T. Keane, and Padraig Conningham, "Hierarchical Case-Based Reasoning Integrating Case-Based and Decompositional Problem-Solving Techniques for Plant-Control Software Design," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 13, pp. 793 - 812 2001.

[16] Y. Vogiazou, B. Raijmakers, M. Eisenstadt, "A research process for designing ubiquitous social experiences," Nordic conference on Human-computer interaction, 2006.

[17] J. F Weng, S. S. Tseng, and N.K. Si, "Constructing the Ubiquitous Intelligence Model based on Frame and High-Level Petri Nets for Elder Healthcare", 2006

[18] Xi Yali, Yang Weiqiang, Yamauchi Noriyoshi, et al., "Real-time Data Acquisition and Processing in a Miniature Wireless Monitoring System for Strawberry during Transportation", TENCON, 2006

[19] J. H. Yang, "Context Aware Ubiquitous Learning Environments for Peer-to-Peer Collaborative Learning ", Journal of Educational Technology and Society, vol. 9, no. 1, pp. 188-201, Jan, 2006.    (SSCI)

[20] Stephen J.H. Yang, Angus Huang, Rick Chen, et al, “Context Model and Context Acquisition for Ubiquitous Content Access in Ubiquitous Learning Environments” IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2006

[21] C. Yin, Y. Yano, "Ubiquitous-Learning System for the Japanese Polite Expressions" presented at International Workshop on Wireless and Mobile Technologies in Education (WMTE), 2005.

[22] G. Zhang, M. Lin, "A Framework of Social Interaction Support for Ubiquitous Learning," presented at International Conference on Advanced Information Networking and Application (AINA), 2005.