An Automatic Approach for Extracting Process Knowledge from the Web

Hua Xiao
School of Computing
Queen's University
Kingston, Ontario, Canada
huaxiao@cs.queensu.ca

Bipin Upadhyaya, Foutse Khomh, Ying Zou
Dept. of Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada
{ 9bu, foutse.khomh, ying.zou}@queensu.ca

Joanna Ng, Alex Lau
IBM Canada Laboratory
Markham, Ontario, Canada
{jwng, alexlau}@ca.ibm.com

***Abstract*— Process knowledge, such as tasks involved in a process and the control flow and data flow among tasks, is critical for designing business processes. Such process knowledge enables service composition which integrates different services to implement business processes. In the current state of practice, business processes are primarily designed by experienced business analysts who have extensive process knowledge. It is challenging for novice business analysts and non-professional end-users to identify a complete set of services to orchestrate a well-defined business process due to the lack of process knowledge. In this paper, we propose an approach to extract process knowledge from existing commercial applications on the Web. Our approach uses a Web search engine to find websites containing process knowledge on the Internet. By analyzing the content and the structure of relevant websites, we extract the process knowledge from various websites and merge the process knowledge to generate an integrated ontology with rich process knowledge. We conduct a case study to compare our approach with a tool that extracts ontologies from textual sources. The result of the case study shows that our approach can extract process knowledge from online applications with higher precision and recall comparing to the ontology learning tool.**

Keywords-Process knowledge acquisition, ontology, business process

I. INTRODUCTION

A business process is a set of logically related tasks that are linked by control flows and data flows to achieve given objectives. For example, "planning a trip for travelers" is a typical business process for travel agencies. The process contains tasks, such as "booking flight tickets", "searching for a map of the destination", and "reserving a hotel". A task in a business process is the lowest level of details to describe an operation. A sub-process describes a collection of tasks that can be reused in different contexts. Process knowledge, such as tasks involved in a process and the control flow and data flow among tasks, is critical for designing business processes. Such process knowledge enables service composition which integrates different services to implement business processes. Different tools are available for both professional and non-professional users to support service composition. Professional service composition tools, such as IBM WebSphere Integration Developer (WID) [18] and Oracle BPEL process Manager [20], provide a platform for business analysts and developers to build business processes by composing services. Lightweight service composition tools, such as IBM Mashup Center [17] and iGoogle [13], are provided to users at various skill levels to assemble online applications into a new application (*e.g.*, a business process). However, in the current state of practices, business processes are primarily designed by experienced business analysts who have extensive domain knowledge. It is challenging for novice business analysts and non-professional end-users to identify a complete set of services to orchestrate a well-defined business process due to the lack of process knowledge.

Research efforts have been devoted to sharing knowledge in public. Several knowledge bases are designed to allow machines to retrieve and process the knowledge stored in the knowledge bases. Ontologies use a formal way to represent knowledge as a set of concepts and relationships among the concepts. For example, the ontology of "travel" lists relevant concepts, such as "booking flight tickets", "hotel reservation", and "weather forecast". Ontologies are widely used for knowledge representation and sharing. DBpedia [5] and Freebase [9] are examples of knowledge bases that extract the knowledge from Wikipedia [29] and store it using Ontologies. Information extraction tools, such as Text2Onto [6], can extract ontologies from textual resources, such as text files or Web pages. However, existing knowledge bases and tools focuses on explaining high level concepts into more concrete concepts. Such knowledge description is lack of a stepwise description on how to complete a collection of tasks to achieve an objective. Moreover, Websites, such as on-line stores, and travel agencies provide specialized services to users. Such websites capture domain specific process knowledge and assist users in completing tasks following the embedded business processes. For example, expedia.com provides interactive user interfaces to allow users to complete various tasks in a trip planning process including tasks such as "buying flight tickets", "booking hotels" and "purchasing travel insurance".

To leverage the domain knowledge embedded in specialized websites, we propose an approach to extract the process knowledge from such websites. Our approach attempts to make the process knowledge available for non-expert users to use in service composition. More specifically, we analyze the navigation information in a website to identify the tasks needed for completing an embedded process. To provide comprehensive process knowledge for achieving an objective, our approach merges the process knowledge extracted from multiple websites that serve for the same objective (*e.g.*, travel planning). Generally, the extracted process knowledge in our paper is the knowledge of business processes instead of executable or abstract

978-0-7695-4463-2/11 $26.00 © 2011 IEEE
DOI 10.1109/ICWS.2011.85

business processes. Business process specification languages, such as BPEL [27] and BPMN [3], are designed to describe executable or abstract business processes and are not suitable for representing the extracted process knowledge. In our work, we use ontologies to represent the extracted process knowledge.

The remainder of this paper is organized as follows. Section 2 presents a meta-model to describe websites and a background on ontologies. Section 3 gives an overview of our approach that extracts process knowledge from multiple websites. Section 4 discusses the case studies. Section 5 gives an overview of related work. Finally, Section 6 concludes the paper and presents the future work.

II. META-MODEL FOR WEBSITES AND STRUCTURE OF ONTOLOGIES

A. A Meta-model for Websites

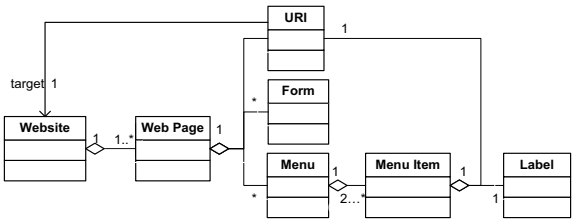

Figure 1. A meta-model for websites

Due to the diversity in the design and implementation of various websites, the appearance of the same information can be presented differently in various websites. For example, the navigation information (*i.e.*, the menu) of a website can be described as a table, a list, or parallel paragraphs with different fonts and sizes. To deal with such diversity in the website design and implementation, we summarize the common structures of websites using the meta-model shown in Figure 1.

Generally, a website contains a collection of related Web pages. Each Web page has a Uniform Resource Locator (URL) to indicate the address of the Web page on the Internet. Forms in a Web page are used to collect inputs from users. A menu is intended to guide users to navigate through different Web pages in the website. More specifically, a menu contains a group of menu items that link to different Web pages where a user can conduct tasks, such as select a product. Each menu item contains a label and a URL. The label shows the name of a menu item, and the URL is a link to a Web page. Menus can be implemented in different ways, such as HTML table [23], HTML list [23] or a set of sequential HTML hyperlinks. Figure 2 illustrates an example of website with a menu. The menu is represented using HTML list tags, *i.e.*, <ul> … … … . Essentially, the menu items indicate a set of tasks that a user needs to perform in order to complete one or more processes. The process knowledge can be captured in the menus.

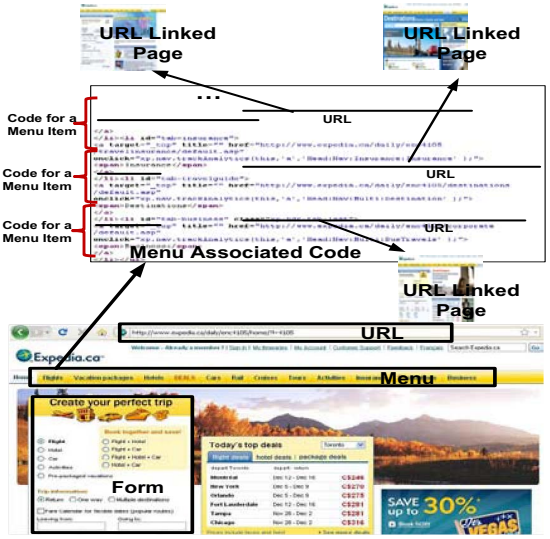

Figure 2. An example of a Web page

B. Structure of Ontologies

Ontologies can be formally described using ontology specification languages, such as Web Ontology Language (OWL) [24] and Resource Description Framework (RDF) [2]. Regardless of ontology specification languages, ontologies contain the following major components.

- Class: is an abstract description of a group of resources with similar characteristics.
- Individual: is an instance of a class.
- Property: describes an attribute of a class or an individual
- Relation: defines ways in which classes and individuals can be related to one another. Typical relations include subclass, partOf, complement, intersection and equivalent. Subclass extends a class to convey more concrete knowledge. PartOf relation indicates that a class is a part of another class. Class A is a child of class B, if class A is a subclass of B or class A is a partOf B. A complement of a given class refers to another class which contains all the members that do not belong to the given class. Intersection contains the members shared among multiple classes. Equivalent defines that two classes contain exactly the same set of individuals.

III. AN APPROACH TO EXTRACT PROCESS KNOWLEDGE FROM THE WEB

Figure 3 provides an overview of the steps that extract process knowledge from Websites. As shown in Figure 3, The objective of a business process is represented as a goal that can be described using a phrase or a set of keywords, such as “travel” and “apply credit cards”. We submit the goal to an existing Web search engine, such as Google, to find relevant websites. However, not all the websites returned from a search engine encode rich process knowledge. In our approach, we analyze the navigation information of websites and then use the navigation information to select the websites with desired process knowledge. In a selected website, we analyze the identified menu to recover sub-processes from the menu. We also

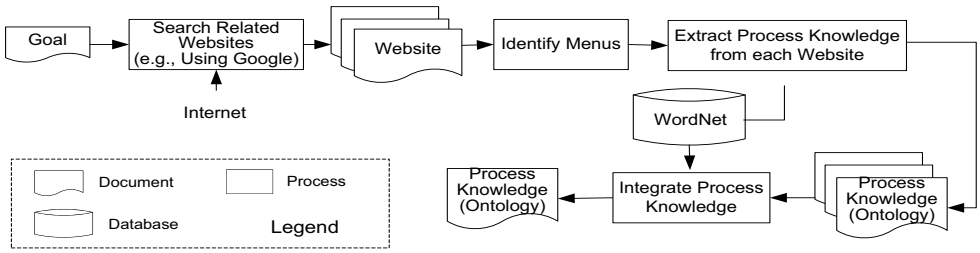

Figure 3. An overview of our approach

capture the tasks and properties of the sub-processes. Finally, we integrate the process knowledge extracted from different websites to form a more comprehensive ontology that elaborates process knowledge of a given goal.

A. Identifying a Menu from a Website

```
Function: identify_menu
Input: HTML code of a Web page
Output: a list of identified menu
1. Remove advertisements;
2. curr_node = root node of the input HTML;
3. queue = empty;
4. queue.push(curr_node);
5. while(queue is not empty){
6.   curr_node = queue.pop();
7.   if (curr_node has children){
8.       child_list = children node of curr_node;
9.    }
10.  else { continue;}
11.  if(child_list satisfies the features  of menu items){
12.      add children to the identified menu list;
13.   }
14.  else {
15.      add the nodes in child_list to queue;
16.   }
17.}//End the while loop
18. Output the identified menu list;
```

Figure 4. Algorithm to identify menu items

We propose an algorithm as shown in Figure 4 to describe the steps for identifying menus in a website. A Web page often contains advertisements irrelevant to the objective of the website. To filter out such noises, we apply the approach proposed by Gupta *et al*. [8] by manually registering the URLs of the well-known advertisement service providers. We parse the HTML document of a Web page to analyze the values of "src" and "href" attributes in a HTML node. We remove the HTML nodes, if the source or hyperlink (*i.e.*, "src" or "href") attributes of the HTML nodes refer to common advertisement servers. From line 4 to line 16 in Figure 4, our algorithm traverses the tree structure of a HTML document from the root node "<HTML>" using breadth-first search algorithm. The queue data structure is used as an assistant data structure to facilitate the tree traversal. Initially, the root of a HTML document (*i.e.*, <HTML> tag) is pushed into the queue. When the queue is not empty, a node is dequeued for further analysis. When we analyze a node, if we find that the node is not a menu, we push the children of the node into the queue.

We identify a menu from a HTML node if the child nodes of the HTML node are identified as menu items, *i.e.,* the child nodes of the HTML node satisfy the following features:

- Menu items are sibling HTML nodes with identical HTML structures. For example as shown in Figure 2, each menu item is an element of the HTML list tag, *i.e.*, ….
- A set of menu items are encapsulated by the same parent HTML tag. In the example shown in Figure 2, the menu items are encapsulated by the parent HTML tag ….
- A menu item contains a URL with a short descriptive text (*i.e.*, label) displayed on a Web page. As shown in Figure 2, the URL of a menu item is represented as a HTML href attribute which links to another Web page. The label of menu items in the example is surrounded by the HTML tag ….
- Identical menu items appear in the target Web pages which are linked by the URLs of menu items. In the example shown in Figure 2, the target Web page contains the same navigation menu items.

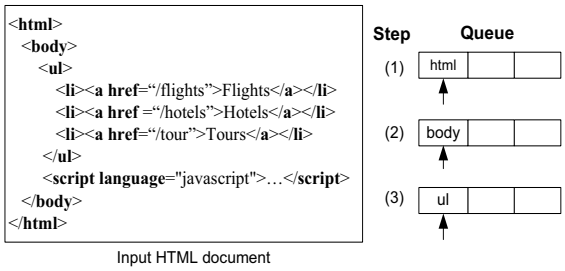

Figure 5. An example to identify navigation bars

Figure 5 is a simplified example which illustrates the major steps to identify menus. In step 1, the algorithm pushes the root node "<html>" into the queue then pop out the first node in the queue to check if it is a menu. Node "<html>" cannot be identified as a menu since the child of node "<html>" does not satisfy the features of menu items *i.e.*, the child node "<body>" does not have sibling nodes. Therefore, we push the child nodes of "<html>" into the queue for further analysis. In step 2, we pop out the next node, *i.e.*, node "<body>", from the queue to check whether it is a menu. Similarly, the node "<body>" is not a menu since its child node "<ul>" does not have sibling nodes. We add the child node "<ul>" to the queue. In step 3, we pop out the node "<ul>" from the queue. Assuming that this list node "<ul>" appears in all the three linked Web pages, *i.e.*, the Web pages with URLs of "/flights", "/hotels", "/tour" all contain the same node "<ul>". The children of the list node satisfy the features of menu items. Therefore, the algorithm identifies these children as menu items and recognizes the

node "<ul>" as a menu. At this point, the algorithm is terminated since the queue is empty.

B. Extracting Ontologies with Process Knowledge

In this section, we discuss our approach that analyzes the semantics between the goal and the menus of a website in order to select the websites with the desired process knowledge. Then we present our algorithm to extract process knowledge from each selected website.

1) Selecting Websites that capture process knowledge

Websites usually use a menu to guide a user through each step of a process. Quite often, a website without menus provides only simple services without detailed process knowledge. Such a simple website is filtered out. Moreover, a website may contain more than one menu. Some menus are used to represent general information instead of the desired process knowledge to meet the goal. For example, a menu which contains menu items "Home", "Contact", "About", and "Login" are used by many websites. This menu does not reveal any process knowledge relevant to the goal. In our paper, we select the Websites that have at least one menu in a website relevant to the goal. To identify the relevance between a menu and a goal described by keywords. We propose a metric, *average Semantic similarity degree*, to measure the semantic similarity between a goal and a menu. We apply the similarity measure proposed by Wu and Palmer [30] to calculate the similarity degree of words which are the basic elements of a menu and a user's goal. *Average semantic similarity degree* is the average value of the *semantic similarity degrees* between the label of each menu item and the goal, as defined in Eq. (1):

$$Average\_Sim = \frac{\sum_{i=1}^{n} sim(goal, label_i)}{n} \quad (1)$$

Where n is the total number of menu items in the menu, and $label_i$ represents the label of the i-th menu item.

To ensure that one of the menus in the website is relevant to the goal, we sort the identified menus based on the *average semantic similarity degree* from high to low. If the highest average semantic similarity degree is greater than a threshold, such a website is relevant to the goal. Otherwise, the menus identified from this website are not related to the goal. Therefore, such a website is filtered out.

2) Extracting Process Knowledge from a Website

We propose an algorithm to extract process knowledge from the selected website. As listed in Figure 6, the input of the algorithm is a goal description, a website (*i.e.*, a collection of Web pages in a website) which contains process knowledge, and a set of menus identified from the website. The extracted process knowledge is represented as an ontology. The goal description is created as a root class for the ontology as shown in line 2 in Figure 6. Then the menu with the highest average semantic similarity degree is converted into classes which are added to the ontology as child classes of the root class (as shown in lines 3 and 4). The remainder menus with lower *average similarity degree* may describe the details of a class in the ontology. For the example of a "travel planning" website, the menu with the highest average similarity degree may contain a menu item "flight". Another menu in the website may contain menu items such as "business class", "economy class" and "airport lounge". The latter menu provides the detailed information for the menu item "flight" in the former menu. By comparing the remainder menu items with the existing classes in the ontology, we can find the relations between the remainder menu items and the classes in the ontology. In our algorithm, if the algorithm identifies a "subclass" or "PartOf" relations between a menu item from the remainder menus and a class in the ontology, such a menu item is created as a new class and added into the ontology.

```
Function: extract_process_knowledge
Input: G —goal description
       W —a website with process knowledge
       menu_list — menus identified from website W and
            sorted based on average semantic similarity
            degree from high to low
Output: On —an ontology with process knowledge
1.{ On = Create an empty ontology;
2.  root_class = Create a root class in On using G;
3.  curr_menu = the first menu in menu_list;
4.  Covert curr_menu to subclasses of root_class;
5. for each menu item in the remainder menu_list {
6.    new_class = convert the menu item into a class;
7.    Identify relations between new_class and the
       existing classes in On;
8.   if(exist a relation between new_class and the
        classes in On){
9.          add new_class to On based on the relation;
10.    }
11. }//End for loop
12. Extract properties for classes in On;
13. Output ontology On;
14.}
```

Figure 6. Algorithm to extract the ontology from a website

WordNet is a lexical database which groups words into sets of synonyms and connects words to each other via semantic relations. We use WordNet [22] to identify class relations. The following word relations defined in WordNet are used to identify class relations.

- **Hypernym** represents a "kind of" relation. For example, car is a hypernym of vehicle. hypernym relation is converted as a subclass relation in the ontology.
- **Hyponym** means that a word is a super name of another. For example, vehicle is a hyponym of car. Hyponym is the inverse of hypernym. In the ontology, super class indicates hyponym relation.
- **Holonym** describes whole-part (*i.e.*, partOf) relation. For example, building is a holonym of window.
- **Meronym** is the inverse of holonym and represents part-whole relation. For example, window is a meronym of building. Meronym relation is converted to PartOf relation in the ontology.

WordNet can identify highly semantic related relations, such as flight is a kind of transportation. However, due to the lack of domain knowledge, the relations between two words cannot be recognized when they are related in business processes without strong semantic relations. For example, in the process of "travel", "hotel" can has a partOf relation with

"travel". However, WordNet cannot recognize such relations. In addition, a business process may use phrases (*i.e.*, more than one word) to describe tasks or the input and output of tasks. For example, "first class", "business class" and "economy class" could be the input parameters of task "searching for flight tickets". WordNet is designed as a general lexical database and does not have the capability of recognizing phrases.

As aforementioned, each label in a menu item is associated with a URL which indicates a path for the Web page to be retrieved from the server. A path shows the hierarchical structure for organizing the linked web pages in different menu items. For example, a "flight" menu item is linked to *www.flightcentre.ca/flights* and a "business class" menu item is connected to *www.flightcentre.ca/flights/business-class*. The information related to "business class" is stored under the directory of "flights". The organization of directory structure suggests partOf relations between these two menus items (*i.e.*, "flight" and "business class"). We can infer that "business class" is a subclass of "flights" since the URL of "business class" is in the sub-directory of the URL of "Flights".

3) Extracting properties and tasks from associated Web pages

In the extracted ontology from a website, a class in the ontology is mapped to a menu item in the website, and therefore a class is associated to a linked Web page by a URL. We further analyze the linked Web pages to recover the properties and children for each class.

TABLE 1 MAP FORM ELEMENTS TO CLASS PROPERTIES

Form element		Class properties and relations
Name	**example**	
Label with input area	First name:	A property of the class
Radio Buttons	Male / Female	Class properties with "OR" relation
Checkboxes	I have a bike / I have a car	Properties of the class
Select list (drop-down list)	Destination Toronto (Toronto, Chicago, New York, Washington)	Class properties with "OR" relation
Title of the form	*Search Flight Ticket*	If the text on the title or submit button is relevant to the class based on *Semantic similarity degree*, we convert the text of title or submission button as a subclass of the class, and put all the extracted properties of the form as the properties of the subclass.
Submission Button	Search Hotel	

HTML forms are often designed to take a user's input in order to provide a service to the user. In our approach, we extract the HTML forms from the Web page linked to a class defined in the extracted ontology, and check if the title and content of the HTML form is relevant to the class. More specifically, the label of input fields (*e.g.*, text fields, password fields and radio buttons) are converted to properties of the class. Table 1 lists the mapping between the elements of forms and the properties of classes defined in an ontology.

C. *Integrating Process Knowledge Extracted from Different Websites*

```
Function: integrate_process_knowledge
Input: G — goal description
       onto_list — Ontologies extracted from different
                   related websites
Output: int_onto — an integrated ontology
1.{  Create an empty ontology int_onto;
2.   create the root class for int_onto using G;
3.   queue = empty;
4.   queue.push(root class);
5.   while(queue is not empty){
6.     curr_class = queue.pop();
7.     match_cls_list = find classes that match with
       curr_class from the input ontologies;
8.     for each cls in match_cls_list {
9.       Integrate properties from cls to curr_class;
10.      Add children of cls to curr_class;
11.      Push children into the queue;
12.    }
13.  } //End the while loop
14.  output int_onto;
15. }
```

Figure 7. Algorithm to integrate process knowledge

Each relevant website contains partial information of the process knowledge. To obtain more complete process knowledge, we integrate the process knowledge (*i.e.*, ontologies) extracted from multiple websites. Figure 7 presents our algorithm that integrates process knowledge. As a starting point, we use the goal description to create the root class of the integrated ontology. We gradually add the knowledge (*i.e.*, classes, properties and relations) from an extracted ontology into the integrated ontology. We use the variable *curr_class* to store the current class that we are analyzing in the algorithm. As shown from lines 7 to 9 in Figure 7, starting from the root of the new ontology, we use the current analyzing class (*i.e.*, represented by *curr_class*) to search for the matching classes defined in the input ontologies. Two classes are matching if the names of these two classes are the same or synonyms according to WordNet. The properties of the matching classes may vary in different ontologies. We merge the properties of the matching classes to the integrated ontology; so that the *curr_class* in the integrated ontology can include all the properties. As shown in line 10 of Figure 7, we add the child classes of the matching classes from different input ontologies to the integrated ontology. We recursively merge the child classes of the matching class into the integrated ontology following the conditions listed in Table 2. If a child class of the matching class does not exist in the integrated ontology, we insert the child class into the integrated ontology as a child class of *curr_class* as shown in the second row in Table 2. If another class in the integrated ontology has a subclass or partOf relation with the child class of the matching class, we insert the child class into the integrated ontology by adjusting the relations among these three classes (*i.e.*, *curr_class*, *a_class* and child class as shown in Table 2). More specifically, as shown in the third row in Table 2, if there exists a class that is the child of *curr_class* and is the parent of the child class, then we add this class as the child of *curr_class* and the parent of the child class. The fourth row in Table 2 shows that, if there exists a class which is the child of both *curr_class* and the

child class, then we add this class as a child of the child class.

TABLE 2. OPERATIONS TO ADD A CHILD CLASS

Condition	Operation
The child class* does not exist in the integrated ontology	curr_class → child class
Exist a_class**: curr_class, a_class, a_class, child class	curr_class → a class → child class
Exist a_class**: curr_class, child class, a_class, a_class	curr_class → child class → a class

* child class represents the child class of the *curr_class*;
** a_class is in the integrated ontology

Figure 8 uses an example to illustrate the main idea of the algorithm that merges two ontologies in a stepwise fashion. In step 1, we create a root class A using the goal description. Then we find matching classes from the input ontologies (1) and (2) where we identify two matching classes for A. Consequently, we add the properties from the matching classes to A in the integrated ontology. In steps 2 to 3, we find the children (*i.e.*, B and C) of A from ontology 1 and add them into the integrated ontology. In step 4, class C is a child class of A in ontology 2. However, class C exists in the integrated ontology. Therefore, instead of adding another class C to the integrated ontology, we merge the properties of C from ontology 2 with the properties of C in the integrated ontology. In step 5, H is a child of A. Assuming that WordNet database indicates that H is the parent of B, we add H as a child of A and move B to be the child of H instead of the child of A.

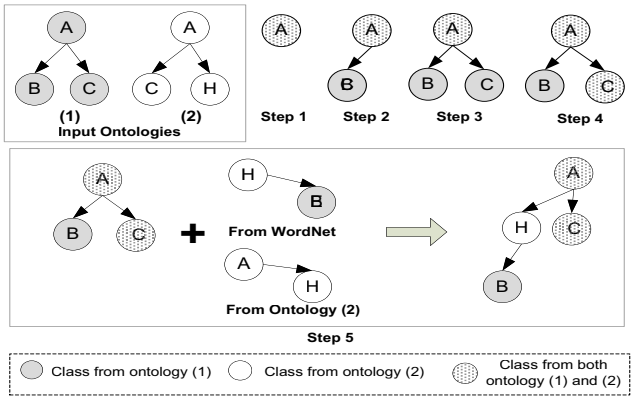

Figure 8. An example of integrating ontologies

IV. CASE STUDY

The objectives of our case study are to: (1) Evaluate the quality of the process knowledge extracted from each website; (2) Evaluate the quality of the integrated process knowledge.

A. Setup

We gather 10 goals (*e.g.*, plan a trip, buy a car, and find a job) from two websites: eHow [7] and WikiHow [28] which use articles and videos to provide online how-to instructions for fulfilling various goals. The collected goals are from different domains. To form a business process, each of the gathered goals needs more than one online service to achieve. Table 3 lists the goals used in our case study. The goals are distributed across 10 domains. For each goal, we use Google [11] as the Web search engine to collect the websites related to goals by submitting the goal description (*i.e.*, keywords) to Google. For each query used to search for relevant websites, we collect the first 8 results and apply our approach to each result. 8 results are the maximum number of results that we can get using Google AJAX Search API [10]. In total, we analyze 80 related websites for 10 goals and generate 80 ontologies. For each goal, we produce an integrated ontology to combine the process knowledge extracted from the 8 websites returned by Google.

TABLE 3. LIST OF GOALS

No.	Goal Description	domain
1	Apply university	Education
2	travel	Travel
3	choose a gift	Shopping
4	Buy a cell phone	Shopping and Electronics
5	Buy a car	Shopping and Automobile
6	Buy a house	Real estate
7	Find a job	Career
8	Tax report	Tax
9	Canada health insurance	Insurance
10	Apply credit card	Finance

B. Evaluating the Quality of Process Knowledge Extracted from a Single Website

To evaluate the quality of ontologies (*i.e.*, process knowledge) extracted from each website, it would be ideal if we could compare our approach with other tools which are specially designed for extracting process knowledge from commercial websites. According to the best of our knowledge, there are no public available tools similar to ours which can extract process knowledge from public Web pages without the prior knowledge of the server side source code. In this case study, we compare the ontologies extracted using our approach with the ontologies extracted using Text2Onto [5]. Text2Onto can extract ontologies from unstructured or semi-structured textual resources. However, the extracted ontologies from Text2Onto represent all the information conveyed in the textual resources instead of process knowledge. Text2Onto is the available tool that is the most similar to ours.

We apply our approach and Text2Onto tool to extract an ontology from the same website. We evaluate the effectiveness of both approaches by measuring the precision and recall. As shown in Eq (2), the precision is the ratio of the total number of items correctly extracted by an approach to the total number of items extracted by the approach. Recall is the ratio of the total number of items correctly extracted by an approach to the total number of items existed in websites.

$$Precision = \frac{|\{relevant\ items\} \cap \{retrieved\ items\}|}{|\{retrieved\ item\}|} \quad (2)$$

$$Recall = \frac{|\{relevant\ items\} \cap \{retrieved\ items\}|}{|\{relevant\ items\}|} \quad (3)$$

To assess the process knowledge extracted from each website, a subject, who is one of the authors, spent 3 days manually examining the 80 websites to extract process knowledge. The subject is a graduate student with 5 years experience working on business process related projects. His knowledge and experience on business processes ensure that he can properly identify the process knowledge from these websites. To avoid any interference, the results from text2Onto and our prototype were not disclosed to the subject before he finished the analysis. We calculate the precision and recall by comparing the result from text2Onto and our approach with the result from the subject.

Table 4 lists the recall and precision of our approach and the Text2Onto tool. The results show that our approach can effectively extract most of the processes information from each website. It can achieve a recall of 0.80 and a precision of 0.82 comparing with Text2Onto which has a recall of 0.35 and a precision of 0.06. There are two major reasons that cause Text2Onto to have the low recall and precision. Firstly, Text2Onto simply extracts concepts from textual sources and does not have any mechanism to filter the concepts irrelevant to the goal. For most of Websites, Text2Onto identifies a large number of irrelevant concepts, such as copyright information and advertisements. Secondly, Text2Onto splits many phrases into separated words which make important phrases lose their meaning. For example, "Used Car Price Quotes" is a useful activity when a user wants to buy a used car. But Text2Onto may remove the adjectives and only keep the nouns "car" and "Quotes".

TABLE 4. RECALL AND PRECISION FOR ONTOLOGIES EXTRACTED FROM EACH WEBSITE

	Average Recall	Average Precision
Text2Onto	0.35	0.06
Our approach	0.80	0.82

We examined the false positives of our approach and found that one of the major problems is due to the JavaScript code. Our current approach does not parse and make use of any information from JavaScript code. It makes our approach miss some process knowledge in the Web page since JavaScript can be used to display any text on Web pages. We believe that if our approach takes JavaScript into consideration, it would increase the recall and precision.

C. Evaluating the Integrated Process Knowledge

After we extract the process knowledge from 8 websites, our approach combines the process knowledge from the websites relevant to the same goal to generate an integrated ontology. We import the same set of websites to Text2Onto and run Text2Onto to extract an ontology. To provide the standard ontologies to evaluate the recall and precision of our generated ontologies, the subject examined all the returned websites and manually extracted the process knowledge from those websites. The manually identified process knowledge is described as an ontology and treated as the standard for comparisons. Table 4 shows the results of our approach and the Text2Onto tool. The results demonstrate that our approach can achieve a much higher average recall and precision.

The results of this case study show that our approach can achieve a higher average recall and precision comparing with Text2Onto, for the purpose of extracting process knowledge. The major reason is that Text2Onto is designed as a general tool to extract ontologies from textual resources instead of focusing on extracting process knowledge.

TABLE 5. RECALL AND PRECISION FOR INTEGRATED ONTOLOGIES

	Average Recall	Average Precision
Text2Onto	0.52	0.05
Our approach	0.87	0.78

D. A Practical Use of Extracted Process Knowledge

A prototype of the proposed approach was implemented and integrated into our smart service composition system [26]. The prototype is developed in Java and uses OWL API [21] as the ontology parser. Java API for WordNet Searching [15] is adopted to access the WordNet database. The detailed information on the smart service composition system is described in our earlier publications [25][26].

In the smart service composition system, we use the goal description provided by users to search for related websites and extract process knowledge. Then the system uses the extracted process knowledge to generate a process and compose services to help users fulfill their goal. A user can edit the generated process by removing or adding tasks. We record the modifications as the user's preferences. When the user wants to accomplish the same goal, our prototype provides the previous refined process.

V. RELATED WORK

Our work is related to two research areas: process mining and recovering, and information extraction.

Process mining and recovering are techniques to extract business process information from event logs recorded by information systems or source code. Agrawal *et al*. [1] present an approach to constructs process models from the log of past. Their approach can generate a process model with the control flow of the business process from the logs of unstructured executions of a process. Francescomarino *et al*. [8] trace the Web system executions and analyze the Graph User Interface of the application during its execution to infer business processes. Zou *et al*. [31] presents an approach to automatically recover business process definitions from business applications. However, Zou's approach needs to analyze the source code of business applications running on the server. The source code of business applications is confidential data and generally not available. Our approach extracts process knowledge from publicly available Web pages without requiring the source code of business applications or event logs.

Information extraction systems transform unstructured documents or semi-structured documents into structured data. Cimiano and Völker [5] developed a framework to learn ontologies from text documents. An ontology learning tool named Text2Onto is developed by Cimiano and Völker. Our paper uses Text2Onto as a baseline to evaluate our approach. Chang *et al*. [4] summarize and compare the existing Web information extraction systems which extract

information from semi-structured documents (e.g., HTML documents). However, those systems are designed to extract information in different aspects instead of focusing on process knowledge. Therefore, the extracted information has too many noises to be used for generating processes and composing services. Similar to our work, Liu and Agah [16] provide an approach to search for process knowledge from the Web. Their approach retrieves the processes explicitly published on the Web, such as www.eHow.com and www.wikiHow.com. The outcome of Liu and Agah's work are semi-structured text descriptions which are difficult to be processed by machines. In addition, Liu and Agah's work does not integrate the process knowledge from different sources to get more complete process knowledge. Our approach can extract and integrate the process knowledge implicitly described by existing online applications. The extracted data are represented as ontologies which can be easily processed by machines. Hoxha *et al*. [14] provide an approach to recovers the process following the submission buttons step by step using automatically form filling techniques. However, automatically form filling is difficult to achieve nowadays. Our approach analyzes the static Web pages and does not require automatically form filling.

VI. CONCLUSIONS AND FUTURE WORK

Process knowledge is essential for service composition. In this paper, we present an approach to extract process knowledge from the Web. By analyzing the content and structure of relevant websites, our approach can extract and merge process knowledge from various websites to generate an integrated ontology with rich process knowledge. We show through a case study that our approach effectively extracts process knowledge from websites.

In future work, we plan to integrate the process knowledge extracted using our approach with online publicly available process databases, such as The MIT Process Handbook Project [19]. We also plan to recruit more subjects to better improve our evaluations.

ACKNOWLEDGMENTS

This work is financially supported by NSERC and the IBM Canada CAS Research.

REFERENCES

[1] R. Agrawal, D. Gunopulos, and F. Leymann, "Mining Process Models from Workflow Logs", Proc. of Sixth International Conference on Extending Database Technology, 1998, pp. 469-483.

[2] D. Beckett, B. McBride, RDF/XML Syntax Specification (Revised), W3C Recommendation, 2004

[3] Business Process Model and Notation (BPMN), FTF beta for version 2.0, avaialbe at: http://www.omg.org/cgi-bin/doc?dtc/09-08-14, last accessed on Jan. 25, 2011

[4] C. H. Chang, M. Kayed, M. R. Girgis, K. Shaalan, "A Survey of Web Information Extraction Systems," IEEE Transactions on Knowledge and Data Engineering, Vol. 18(10), Oct.2006, pp. 1411-1428.

[5] DBpedia, http://dbpedia.org/, last accessed on Jan. 11, 2011

[6] P. Cimiano, J. Völker, "Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery," Proc. of the 10th International Conference on Applications of Natural Language to Information Systems, Alicante, Spain, Jun. 2005, pp. 227-238.

[7] EHow, http://www.ehow.com/, last accessed on Feb. 13, 2011

[8] C. D. Francescomarino, A. Marchetto and P. Tonella, "Reverse Engineering Of Business Process Exposed As Web Applications", European Conference On Software Maintenance And Reengineering, Kaiserslautern, Mar. 24-27, 2009, pp. 139-148.

[9] Freebase, http://www.freebase.com/, last accessed on Feb. 1, 2011

[10] Google AJAX Search API: Class Reference, http://code.google.com/apis/ajaxsearch/documentation/reference.html, last accessed on Feb. 11, 2011

[11] Google, http://www.google.com, last accessed on Feb. 9, 2011

[12] S. Gupta, G. Kaiser, D. Neistadt, P. Grimm, "DOM-based Content Extraction of HTML Documents," Proc. of WWW 2003, Budapest, Hungary, May 20-24, 2003

[13] iGoogle, http://www.google.com/ig, last accessed on Feb 13, 2011

[14] J. Hoxha, and S. Agarwal, "Semi-automatic Acquisition of Semantic Descriptions of Processes in the Web," Proc. of 2010 International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, Canada, Aug. 31- Sept. 3, 2010

[15] Java API for WordNet Searching (JAWS), http://lyle.smu.edu/~tspell/jaws/index.html, last accessed on Feb. 9, 2011

[16] Y. Liu, and A. Agah, "Crawling and Extracting Process Data from the Web," Proc. of ADMA, 2009, pages: 545-552

[17] IBM Mashup Center, https://greenhouse.lotus.com/wpsgh/wcm/connect/lotus+greenhouse/lotus+greenhouse+next+site/home/products/ibm+mashup+center, last accessed on Feb. 2, 2011.

[18] IBM WebSphere Integration Developer, http://www-01.ibm.com/software/integration/wid/, last accessed on Feb. 13, 2011

[19] MIT Process Handbook Project, http://ccs.mit.edu/ph/, last accessed on Jan, 25, 2011.

[20] Oracle BPEL process Manager, http://www.oracle.com/technetwork/middleware/bpel/overview/index.html, last time accessed on Feb. 13, 2011

[21] OWL API, http://owlapi.sourceforge.net, last accessed on Jan. 7, 2011

[22] Princeton University, "About WordNet," 2010, Available at: http://wordnet.princeton.edu, last accessed on Feb. 9, 2011

[23] D. Raggett, A. L. Hors, I. Jacobs (editors), "HTML 4.01 Specification," W3C Recommendation, Dec. 24, 1999, available at http://www.w3.org/TR/html4/, last accessed on Feb. 9, 2011

[24] M. K. Smith, C. Welty, D. L McGuinness, "OWL Web Ontology Language Guide," W3C Recommendation, 2004

[25] H. Xiao, Y. Zou, R. Tang J. Ng, L. Nigul, "An Automatic Approach for Ontology-Driven Service Composition," Proc. IEEE International Conference on Service-Oriented Computing and Applications 2009, Taipei, Taiwan, 14-15 Dec. 2009, pp. 1-8.

[26] H. Xiao, Y. Zou, R. Tang, J. Ng, L. Nigul, "A Framework for Automatically Supporting End-Users in Service Composition," In book The Smart Internet, LNCS, Vol. 6400, Springer-Verlag, 2010, pp. 115-136

[27] Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, last accessed on Jan. 18, 2011

[28] WikiHow, http://www.wikihow.com/, last accessed on Feb. 14, 2011

[29] Wikipedia, http://www.wikipedia.org/, last accessed on Feb. 13, 2011

[30] Z. Wu and M. Palmer, "Verb Semantics and Lexical Selection," In 32nd Annual Meeting of the Association for Computational Linguistics, 1994, pp. 133-138.

[31] Y. Zou, J. Guo, K. C. Foo, M. Hung, "Recovering Business process from Business Applications," Journal of Software Maintenace and Evolution: Research and Practice, Vol. 21(5), Sept. 2009, pp. 315-348