December 2004

Process change identification using workflow specification matching

M. Yeoh
Loughborough University

P. Chung
Loughborough University

C. Anumba
Loughborough University

A. El-Hamalawi
Loughborough University

I. Motawa
Loughborough University

Process change identification using workflow specification matching

Yeoh, M.L.
Computer Science Department
Loughborough University
Leicestershire
LE11 3TU UK
m.yeoh@lboro.ac.uk

Chung, P.W.H.
Computer ScienceDepartment
Loughborough University
Leicestershire
LE11 3TU UK
p.w.h.chung@lboro.ac.uk

Anumba, C.J.
Civil and Building Engineering Department
Loughborough University
Leicestershire
LE11 3TU UK
c.j.anumba@lboro.ac.uk

El-Hamalawi, A.
Civil and Building Engineering Department
Loughborough University
Leicestershire
LE11 3TU UK
a.el-hamalawi@lboro.ac.uk

Motawa, I.A.
Civil and Building Engineering Department
Loughborough University
Leicestershire
LE11 3TU UK
i.a.motawa@lboro.ac.uk

ABSTRACT

Adaptive workflow systems and dynamic-change handling methodologies have been a focus of current workflow development. The goal of such systems is to develop a framework that allows changes to be carried out dynamically. However, a systematic change management approach that ensures changes are properly evaluated, implemented and reviewed has yet to be developed. A dynamic workflow system can be achieved by implementing change management strategies in current workflow systems that cater for change. This paper presents a matching algorithm to identify changes made to a workflow process over time by comparing different versions of the process, and to present the changes in a way that is meaningful to the users. The matching result is useful for change notification and for reviewing how a process has changed over time, for change management purposes. The algorithm is based on tree matching at the overall level and graph matching at the lower level. The matching algorithm is applied to an example construction process and results are presented in this paper.

Keywords

Workflow Management, Workflow Matching, Change Identification, Change Management.

INTRODUCTION

A workflow management system is defined by the Workflow Management Coalition (Hollingsworth, 1995) as "a system that completely defines, manages and executes workflow through the execution of software whose order of execution is driven by a computer representation of the workflow logic". Workflow is concerned with the use of information technologies to

support business processes, by management of the sequence of work activities and the invocation of appropriate human and IT resources associated with the various activity steps.

Conventional workflow management systems are lacking in supporting unstructured, fragmented, and dynamic processes, such as construction processes. Hence, adaptive workflow systems and dynamic change methodology have been a focus of current workflow development (Chung et al., 2003) (van der Aalst and Jablonski, 2000) (Ellis et al., 1995). The goal of such systems is to develop a framework that allows changes to be carried out dynamically. A systematic change management approach, which will ensure change is properly evaluated, implemented and reviewed, is required. A dynamic workflow system that caters for change can be achieved by incorporating change management strategies into current workflow systems (Yeoh et al., 2003) .

The motivation for process change identification is to provide support for the control of changes in a dynamic workflow system. Two important objective of identifying changes in a workflow specification are versioning and querying the past, and learning about changes:

- Versioning and querying the past: The processes of a dynamic business environment are frequently changing. Business analysts may want to version their business processes in order to record history of their process evolutions. The sequences of workflow definition versions can be used to query about the past, e.g. to query changes or to ask for the list of processes introduced over a period of time. In order to query the past, process matching and versioning are required.
- Learning about changes: By matching different versions of a workflow specification a description of the possible process changes can be constructed. Dynamic workflow changes are usually carried out in order to adapt to unforeseen circumstances or to meet specific user needs. These changes are usually unpredicted at the design stages and may occur in certain patterns and under certain circumstances. A workflow specification matching allows these changes to be described and recorded. The description and recording allows the patterns of process change to be clearly identified and these process change patterns can be generalized for future use.

In this paper, we describe a formal approach for identifying workflow changes by matching two versions of a workflow specification. The goal of workflow matching (WM) is to compare two versions of a workflow specification in order to detect the differences in terms of a series of workflow process changes. Workflow specification (WS) provides detailed information about processes of a business, its administration information and also its execution. In other words, a WS is the specification of a domain based on its activities, execution and resources. As a business evolves its workflow specification is modified which results in many different versions. By comparing different versions of a WS, business process analysts are able to evaluate the evolution of their business process and also to analyze the pattern of process changes. In case of change review, especially for unnecessary changes, the change manager is able to audit what had happened and feedback the result for future use.

The remainder of the paper is organized as follows. Sections 2 gives background information regarding the development of the method. Issues related to its implementation and the WM algorithm is discussed in Section 3. Section 4 describes the versioning and matching tool, called wfChange, developed to implement the WM algorithm. Section 5 describes an example that applies the matching technique to a construction project. Related work is summarized in Section 6. Section 7 concludes the paper.

WORKFLOW MATCHING OVERVIEW

WM is based on the XML Process Definition Language (XPDL) framework (WMC-Specification, 2002). XPDL uses XML as the mechanism for process definition interchange. It forms a common interchange standard that enables workflow products to continue to support arbitrary internal representations of process definitions. Matching based on the XPDL framework does not bind the matching algorithm to any particular representation, thus, allowing WM to be used with different workflow products and modeling tools.

```
<?xml version="1.0" encoding="UTF-8"?>
<xpdl:Package Id="1" Name="CFlow Package"
xmlns:xpdl="http://www.wfmc.org/standards/docs/xpdl">

<xpdl:PackageHeader>
    <xpdl:XPDLVersion>0.09</xpdl:XPDLVersion>
    <xpdl:Vendor>Lboro</xpdl:Vendor>
    <xpdl:Created>2004-02-02 14:10:57</xpdl:Created>
    <xpdl:Description>This is a testing workflow package
  </xpdl:Description>
</xpdl:PackageHeader>

<xpdl:WorkflowProcesses>
  <xpdl:WorkflowProcess Id="0" Name="SAINSBURY">
    <xpdl:Activities>
      <xpdl:Activity Id="a1"    Name="DEMONSTRATING THE NEED">
            <xpdl:Performer>ADMINISTRATOR</xpdl:Performer>
      </xpdl:Activity>
          <xpdl:Activity Id="a2" Name="CONCEPTUAL OF NEED">
           <xpdl:Performer>ADMINISTRATOR</xpdl:Performer>
      </xpdl:Activity>
    ……
  </xpdl:Activities>
  <xpdl:Transitions>
      <xpdl:Transition Id="t1" Name="" From="a1" To="a2">
       </xpdl:Transition>
      <xpdl:Transition Id="t2" Name="" From="a2" To="a3">
      </xpdl:Transition>
    ……
    </xpdl:WorkflowProcess>
  </xpdl:WorkflowProcesses>
</xpdl:Package>
```

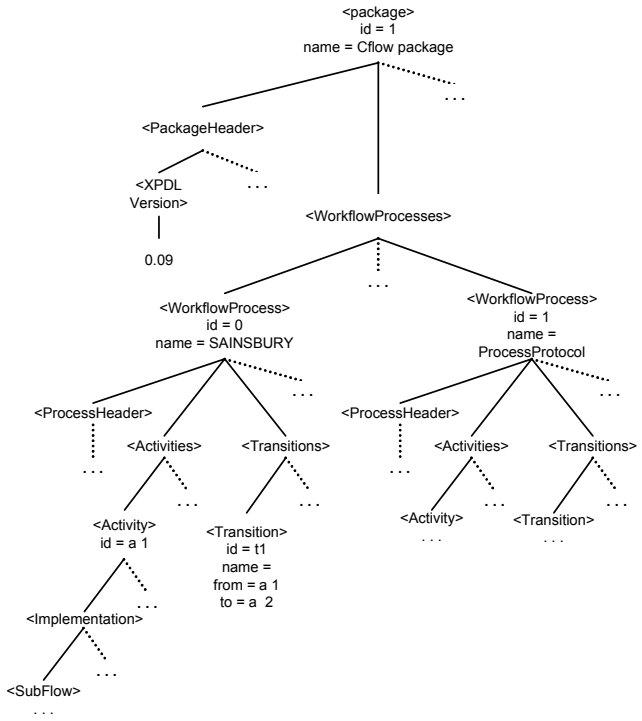

Figure 1: XML document and XML structured tree

Workflow process definition is hierarchically structured in XPDL, which can be represented as a structured XML tree as shown in Figure 1. The listing on the left shows an extract from an example workflow in xml format, which can also be presented as a tree structure as shown in the diagram on the right. Significant research into change detection methodologies for tree-structured XML documents has been done (Chawathe et al., 1996) (Wang, 2003). Change identification of tree structured information or xml documents focuses on identifying changes in a node or xml element, and also changes to its hierarchical relationship, such as movement of a node from one location to another. Tree matching methodologies are unsuitable for identifying workflow changes because they cannot handle the graph-structured information contained in XPDL, i.e. the workflow process definition or workflow graph (Figure2).

A workflow is a Directed Acyclic Graph (DAG) and referred to here as a workflow graph. In this representation, vertices represent a process or a task, and branches between vertices represent relations between processes or tasks. A workflow graph consists of a single entry node and a single exit node, which represent the start and the end of a workflow process, respectively. A node in a workflow graph can be further decomposed into lower levels, where each decomposition can be in the form of another workflow graph or single nodes. Each <WorkflowProcess> sub-tree in XPDL indicates a workflow graph, which can be represented as <WorkflowProcess> = (<Activities>, <Transitions>). The decomposition of a process to its subtasks is indicated by <Activity>'s child element <sub-flow>. These are shown in Figure 1.

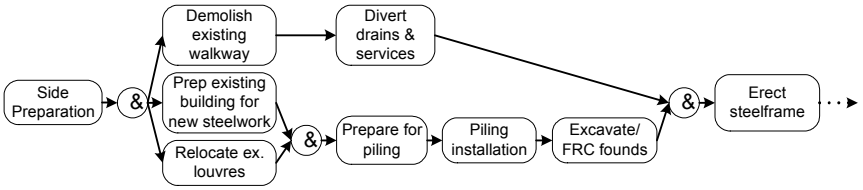

Figure 2: A sample building construction workflow

Standard tree matching tools have problems identifying workflow changes. Specifically, the semantics of workflow graph changes cannot be handled. Workflow changes referred to here are changes to workflow process definition. (van der Aalst

and Jablonski, 2000) identified four basic types of workflow change operations: extend (add), reduce (delete), replace and re-link (move). Change operations will be discussed in the next section.

A graph-based matching methodology is required to identify workflow changes. Many different applications of Graph matching have been addressed in the literature. For example, pattern matching, image or object recognition for computer vision (Kubicka et al., 1990) (Myers et al., 2000) and in other scientific disciplines (Koch and Lengauer, 1997) (Lipton et al., 1989). A method to identify change or transformation of a graph is "graph edit distance". Edit distance is a fundamental concept for measuring the similarity of symbolic data structures (Wagner and Fischer, 1974) (Chawathe et al., 1996) (Myers et al., 2000). It is based on a set of edit operations. Most of these algorithms are direct descendants of the dynamic programming techniques introduced by (Levenshtein, 1966), for finding the edit distance between strings. Graph edit-distance build on the finding of matching between two graphs and computation of the minimum cost edit sequences. The edit operations considered include insertion and deletion of vertices and edges, and vertices and edges substitutions (Myers et al., 2000). Graph edit distance is not required here because the information provided by graph edit distance is available already within the XPDL structure.

In summary it has been shown that the main goal of WM is to analyze the transformation of a workflow process by identifying various change operations. XPDL standard framework is used to allow the matching algorithm to be free from binding to any particular representation. Although XPDL is tree-structured, tree-matching techniques alone are not suitable for workflow change identification. A combination of tree matching and graph matching is required. Differences between two graphs are described using a series a workflow change operations. Changes made should not be simply presented using the three basic edit distance operations, i.e. added, deleted and edit. More comprehensive change operations such as move, copy, expand, replace, added dependency, deleted dependency and block operations should be used instead.

WM ISSUES AND ALGORITHM

The WM problem has several key aspects:

- Two layers of matching. The top layer deals with a tree matching of XPDL data. The lower layer implements graph matching for each workflow process. Tree matching is used to match the overall <Package> definition, such as package's applications, participants, data-structure, workflow processes, etc. By matching the <WorkflowProcess> nodes, matched workflow graphs are identified. A graph matching technique is then applied to identify the differences between the matched workflow graphs. This allows changes to workflow processes to be identified, as well as changes to package information.
- Unique identifiers and conceptual terms. The matching is based on the assumption that each workflow process and task has both a unique identifier and is associated with an unambiguous conceptual term defined in the domain ontology. An identifier allows each node to be distinctly identified; thus, greatly reducing the complexity of the matching. Meanwhile, the conceptual terms permit the matching result be more comprehensible for workflow analysis.
- Change Operations. Four types of process change introduced by (van der Aalst and Jablonski, 2000) are extend, reduce, replace and re-link. Extend is defined as the introduction of new entities, such as adding a task or a role, whereas reduce is defined as the deletion of parts from process definition. Replace is a mixture of reducing and extending a specific part of a workflow graph. Re-link is used to represent a rearrangement of a task, here no tasks are added or deleted but merely reordered. These process change operations do not fully describe changes in a workflow. For instance, the extend, reduce, replace and re-link operations present only limited operations regarding the changes that have occurred. Thus more appropriate change operations are introduced here.

WM Change Operations

WM provides a measure of the difference or similarity between two versions of a workflow graph. In this section the workflow change operations proposed to represent these workflow transformations are described. In order to fully define the workflow transformation, a series of process change operations are introduced together with the basic operations given by (van der Aalst and Jablonski, 2000). Instead of using extend and reduce to represent addition and deletion of both tasks and task properties, an update operation is introduced. Addition and deletion operations are used here to deal with the introduction and removal of tasks and processes. The Update operation is used to denote changes to the properties of a task or a process, such as reallocation of a task to a different participant or the use of a new application to accomplish the task. The operation re-link is referred to here as Move. The replace operation remains unchanged.

The transformation of a workflow graph from version v1 to version v2 can be represented by a series of addition, deletion, move, replace, update and expand operations (Figure 3). Details of these operations and the dependency and block change operations are described here.

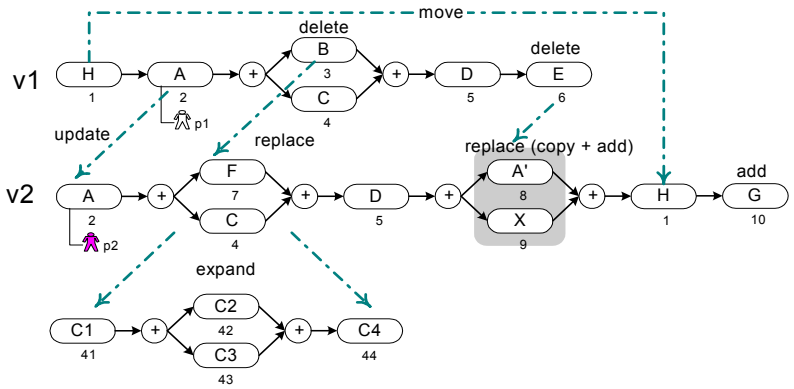

Figure 3: An example of a workflow transformation (v1 → v2)

- Task Addition – An Addition operation indicates that a new task has been introduced. For example, in Figure 3, task X, G, C1, C2, C3 and C4 are added. Insertion and addition of new processes and tasks are necessary in many situations. A task addition operation can be in the form of an insertion between two existing tasks or an addition at the rear of the workflow graph. Insertion of a new task/process includes deletion and adding of links, indicating dependency, linking task immediately preceding and succeeding the inserted task. Adding of a new task at the end of the graph consists of adding a new task and its links.
- Task Deletion – An Delete operation indicates a task has been removed. During the realization of a process plan, a change occurred such that a particular task may no longer be required. Similar to the addition operation, the deletion of the links connected to the deleted task is considered as part of the delete operation rather than separation operations. Deletion of a compound process, which consists of several decompositions, is represented as a single deletion rather than a list of deletions. In figure 3, task B and E are deleted.
- Task Update – An Update Operation indicates changes to the properties of a task. These changes include change of administrative information, change to operational details or changes to participant's details. For example, in Figure 3, task "A" the participant property is updated from p1 to p2 in version V2.
- Task Copy – A Copy Operation indicates creation of a duplicated task. It involves adding a new copy of an existing task in the original workflow graph into the new graph. Copy operation may indicate routine work that needs to be done frequently, such as a review process, or a rework due to an unsatisfied process execution or change of user requirements. Instead of noting the change simply as an addition operation, copy is used to indicate that a similar process exists in V1.
- Task Move – A Move operation indicates a task has been deleted and added at different locations in a process. A task may be moved to a later stage in workflow process execution due to change of its importance or caused by delay. Here, a single Move operation is used to represent the deletion of a task from its location in the original graph and its insertion elsewhere in the new workflow graph. Thus, the Move operation consists of deletion of a task and its associated links from an existing location and adding of the deleted task and new links at a new location.
- Process Expand/Contract – An Expand operation shows the decomposition of a task into smaller subtasks. As a business evolves, an increasingly complex and important task is usually further decomposed into several layers. An Expand operation involves addition of a single task or a series of tasks and links. For example, in Figure 3, task C is expanded into four subtasks (C1, C2, C3, C4). Contract is the reverse of the Expand operation in that its subtask(s) have been deleted.
- Task Replace – The Replace Operation is to identify the substitution of a task by another in a particular process path. Instead of showing the change as a deletion of an existing task and addition of a new task at the same location, a replace operation is used. For example, figure 4, task B is substituted by task F and it is presented as a Replace operation instead of two basic operations of deleting B and adding F. Similarly, it would be more meaningful to represent change to task E as a replace operation than a series of delete, copy and add.

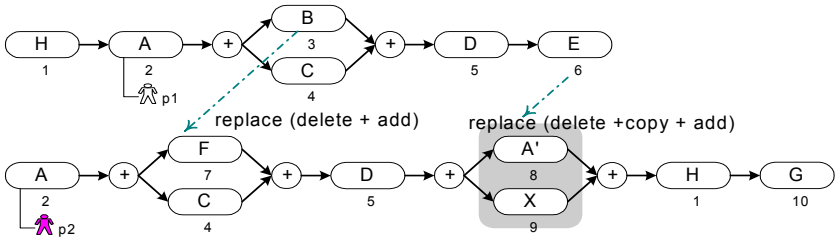

Figure 4: Replace Operations

- Dependency addition and deletion - Two types of dependency change-operations considered here are addition and deletion. These are denoted by adding or deleting of links between tasks in a workflow graph. These changes to the links are not the direct effect or part of any task change operation described earlier. Rather, they are the direct result of adding a new link from one task to another task where previously there were no links. No other structural changes are involved. Identifying changes to task dependency are an important part in assessing workflow evolution, especially on critical tasks. Tasks or processes at a higher level are usually more critical. As a consequence of change in dependency association, subtasks may require additional changes.
- Block operations - Block operations consist of Block Add, Block Delete, Block Copy and Block Move. These operations indicate a collection of tasks have been changed as a group. Instead of presenting a change as a series of additions, deletions, copies and moves, which does not tell much about the whole process, block operations allow identification of changes to a related set of tasks at a higher conceptual level.

Matching algorithm

WM algorithm is based on tree matching of the overall WS and also graph matching between corresponding workflow processes resulting from the tree matching. With the presence of conceptual term (label) representing each task and also a unique identifier, along with its standardized structure based on XPDL schema, the development of the WM algorithm is relatively straightforward. The first step of the algorithm is to parse both of the XPDL-xml files into in a DOM (W3C, 2004) tree-structure. Then the pair of xml trees are matched to identify differences between them. The tree matching process will generate a list of corresponding workflow processes, which will be further manipulated to find their differences. The WM algorithm can be summarized in the following steps:

```
WM (xml1, xml2)
Input: Two xml files
Output: A list of change operations for each matched workflow process
```

```
Parse xml1 and xml2 into DOM1 and DOM2

Compute Tree Matching between DOM1 and DOM2,
and generate a list of corresponding WorkflowProcesses(G1, G2)

For each matched workflow graph in (G1, G2)
        diff(g,g')
Endfor
```

Figure 5: WM Algorithm

The following algorithms deal with matching of workflow graphs, diff(g, g'). A workflow process element in XPDL or workflow graph is a directed acyclic graph. A workflow graph is represented as W = (V, E, L), where V and E are finite sets and every element of E is a two-element subset of V. L is the set of labels for each vertex. For example, in Figure 6, is a digraph with vertex set V = {a, b, c, d}, edge set E = {ab, ac, bd, cd} and label set L = {C1, C2, C3, C4}. If e $\in$ E then e = {ab} for some a, b $\in$ V.

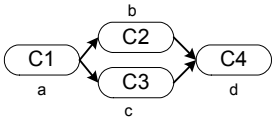

Figure 6: Workflow graph as a DAG

The first phase of graph matching is to perform a bipartite matching of vertices of both graphs, based on their unique identifier and also their labels (Figure 7). A list of matched, added and deleted vertices is generated from the matching process. Similarly, the set of edges of both versions are compared to generate a list of added edges, deleted edges and matched edges.

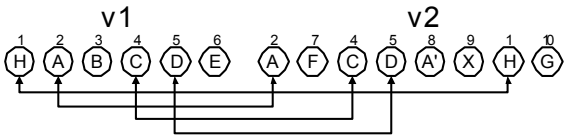

Figure 7: example bipartite comparison

In the second phase the unmatched vertices are further processed to identify the following operations: addition, deletion and copy.

Deletion operation: The new version is parsed to determine any tasks that are in the old version but are absent from the new version. Any vertices not found in the new version are deemed to have been deleted.

Add and Copy: An added vertex will have a new identifier however it may have an identical label and its properties may be similar to an existing vertex. To differentiate whether a newly added vertex is a new task or a copy of an existing task, the vertex is searched for in the old version. If it exists then it is a Copy, otherwise it is an Add. The search is partly implemented during the search for deletions where a list of potentially added or copied vertices is prepared.

In the third phase the matched vertices are further processed to identify the following operations: update, expand/contract, move, replace and dependency change. For each corresponding process or matched workflow graph, computation of the differences between the pair of matched workflow graphs is then carried out. For update and expand/contract the match is based on vertex changes. For move, replace and dependency change the matching is based on changes to vertex connections.

Update: A vertex can be composed of a number of elements, such as participants, priority state or subflow. If any of the participants have changed then the vertex is marked as updated. If an element is a subflow then the process expand and contract will be carried out.

Process Expand and Contract: This process determines whether a subflow has changed. If this is a new subflow (ie it does not exist in the original) then it is marked as an Expand. If it existed in the old version but the contents of the subflow are different then it is marked as Subflow-change. If it existed in the original but is not in the new graph then it is a Contract.

Move: The logic for a move is based on the idea that if a node (a) has moved, then at least one node from the old graph that was in front of (succeeding) the moved node (a), will now be behind (proceeding) the moved node (a) in the new graph. To determine this a breadth first search is performed on the old graph in a forward direction from the position of the moved node in the old graph. Then, for each of the nodes found (i), a breadth first search is performed on the new graph for all preceding nodes (j) in the new graph to ascertain if there is a match i=j. If there is a match then (a) has moved.

In the example given in figure 3 H is identified as a move. The nodes (a,b,c,d,e) are retrieved from the first breadth fist search of the old graph. The nodes (b,e) are omitted from further consideration because they are flagged as deleted already. A breadth first search of all preceding nodes from the node (h) position in the new graph occurs to determine if any the nodes (a,c,d) is found. If the result is true then node (h) is marked as a move.

Replace: Some combinations of additions and deletions can be more meaningful by identifying them as replace. Replace occurs when most of the elements in the two graphs (or sub-parts thereof) match, but one or a small set of vertices between 2 matched vertices has been deleted in the old graph and one or a small set of vertices has been added in the same place in the new graph. This is illustrated in figure 4 where task B in the old graph is replaced by task F in the new graph.

Dependency Addition and Deletion: The algorithm parses the links of all matched vertices where there are no changes to the vertices but links have been added or removed. These are then marked as dependency-add or dependency-delete.

In the fourth phase the resulting list of change operations is further processed to identify the Block operations.

A block is a set of vertices that are linked together where all have had the same operation performed upon them. Block operations are exactly the same as operations performed upon single vertices except, their identification as a set, enables a more meaningful description of the changes to each individual vertex to be used, which is the Block-*operation*.

IMPLEMENTATION

The algorithm presented in this paper has been implemented in a workflow versioning and matching tool called wfChange. wfChange consists of two main components, a mapping and versioning subsystem and the WM. The architecture of wfChange is shown in Figure 8. The mapping and versioning subsystem forms the interface between a workflow system and wfChange. It imports the workflow definition between a connected workflow system database and the wfChange system. A mapping table specified by the users is created to allow the import of workflow definition from the conventional workflow database to xpdl framework in the wfChange. When a new version of a workflow specification is created, it is indexed and installed in the repository. WM plays a central role in the system. The WM computes a matching result of a version of a workflow specification with a previous version from the repository and outputs the result in various ways. The matching result is appended to the existing sequence of results for the workflow specification. Changes are presented as a series of meaningful change operations as described earlier. Changes to process dependency and basic operations are presented in a dependency matrix.

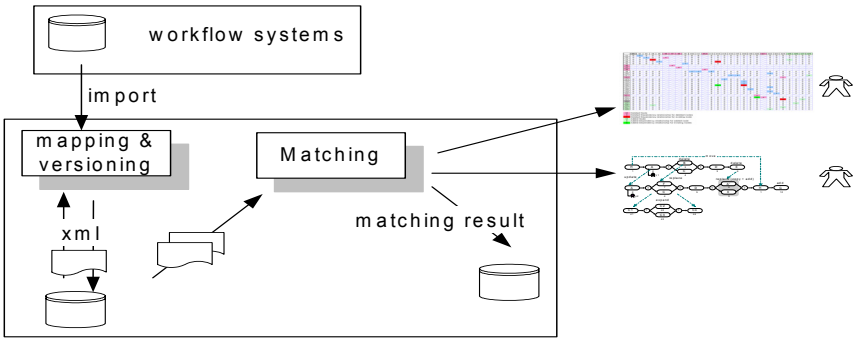

Figure 8: wfChange system architecture

EXAMPLE

In this section a study of the WM algorithm is presented. The example shows that the WM achieves its goals in identifying meaningful WM change operations described earlier.

This is a workflow (v1) for building a 5 bed room detected house including a swimming pool (P1)		This is a updated version of v1 where drainage problem on site required construction of a smaller property (P2), v1	
a1	start	a1	start
a2	obtain plans for (P1)	a2	obtain plans for (P1)
a3	excavate	a3	excavate
a4	lay foundation (standard)	a4	lay foundation (special)
a5	construct rough brick framework	a5	construct rough brick framework
a6	excavate swimming pool	a9	add framework for roof
a7	lay waterproofing for pool	a10	rough exterior plumbing
a8	construct pool site	a11	rough electrical work in house
a9	add framework for roof	a13	install roofing material
a10	rough exterior plumbing	a14	install windows and doors
a11	rough electrical work in house	a15	tile roof
a12	install plumping equipment for pool	a16	construct outer brick casing
a13	install roofing material	a17	rough interior plumbing

a14	install windows and doors	a18	exterior painting
a15	tile roof	a19	exterior fixtures
a16	construct outer brick casing	a20	interior painting
a17	rough interior plumbing	a22	lay flooring
a18	exterior painting	a23	install interior fixtures
a19	exterior fixtures	a24	stop
A20	interior painting	a25	obtain plans for (P2)
A21	install fitted kitchen	a26	secure drainage area
A22	lay flooring	a27	inspection of drainage repair
A23	install interior fixtures	a28	install computer network cabling
A24	stop		

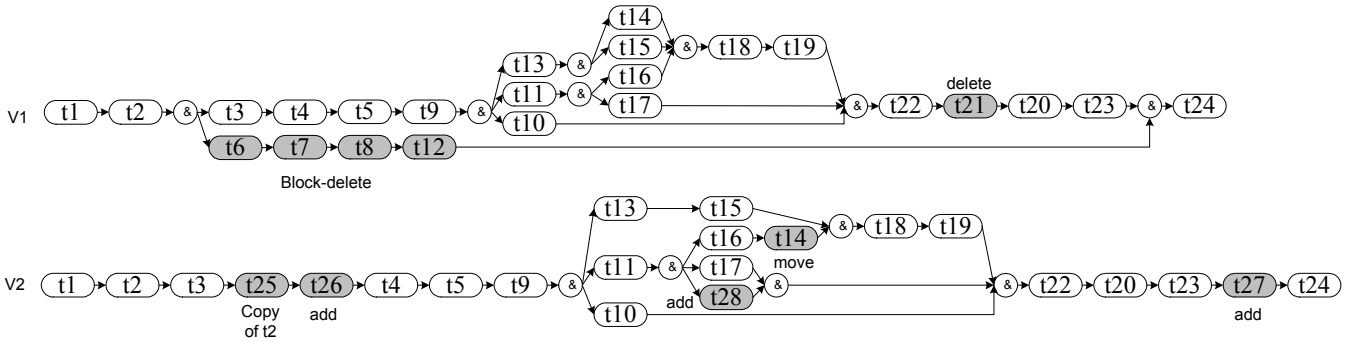

Figure 9: An example of a simple building and construction workflow process model

In the example wfChange was implemented on two versions of a simple construction process (Figure 9). Some of the results of running the program are illustrated in figure 10. This is a dependency matrix.

The dependency matrix shows the changes to the dependency relationships, and deletion and addition of tasks between the two versions. Presenting information in the form of a dependency matrix is another way of adding meaning to the system as a whole.

		t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18	t19	t20	t21	t22	t23	t24	t25	t26	t27	t28
start	t1	0	1	0	0	0				0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
obtain plans	t2	0	0	1	0	0	X			0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
excavate	t3	0	0	0	X	0				0	0	0		0	0	0	0	0	0	0	0		0	0	0	A	0	0	0
lay foundation	t4	0	0	0	0	1				0	0	0		0	X	0	0	0	0	0	0		0	0	0	0	0	0	0
construct rough brick framework	t5	0	0	0	0	0				1	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
execuvate summing pool	t6							X																					
lay waterproofing for pool	t7								X																				
construct pool site	t8												X																
add framework for roof	t9	0	0	0	0	0				0	1	1		1	0	0	0	0	0	0	0		0	0	0	0	0	0	0
rough exterior plumbing	t10	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		1	0	0	0	0	0	0
rough electrical work in house	t11	0	0	0	0	0				0	0	0		0	0	0	1	1	0	0	0		0	0	0	0	0	0	A
install plumbing equipment for pool	t12																								X				
install roofing meterial	t13	0	0	0	0	0				0	0	0		0	0	1	0	0	0	0	0		0	0	0	0	0	0	0
install windows and doors	t14	0	0	0	0	0				0	0	0		0	0	0	0	0	1	0	0		0	0	0	0	0	0	0
tile roof	t15	0	0	0	0	0				0	0	0		0	0	0	0	0	1	0	0		0	0	0	0	0	0	0
construct outer brick casing	t16	0	0	0	0	0				0	0	0		0	A	0	0	0	X	0	0		0	0	0	0	0	0	0
rough interior plumbing	t17	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		1	0	0	0	0	0	0
exterior painting	t18	0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	0		0	0	0	0	0	0	0
exterior fixtures	t19	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		1	0	0	0	0	0	0
interior painting	t20	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		0	1	0	0	0	0	0
install fitted kichen	t21																				X								
lay flooring	t22	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	A	X	0	0	0	0	0	0	0
install interior fixtures	t23	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		0	0	X	0	0	A	0
stop	t24	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
obtain plans	t25	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	A	0	0
secure drainage area	t26	0	0	0	A	0				0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
inspection of drainage repair	t27	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		0	0	A	0	0	0	0
install computer network cabling	t28	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0		A	0	0	0	0	0	0

tZ Deleted Node
X Deleted Dependency relationship for deleted nodes
X Deleted Dependency relationship for existing node
tZ Added Node
A Added Dependency relationship for added node
A Added Dependency relationship for existing nodes

tZ start/end
0/1 Dependency State. 1: dependency exists
0: no dependency

Figure 10: Dependency Matrix

RELATED WORK

Zhuge (2002) introduced a workflow process matching approach for process retrieval for re-use. (Zhuge, 2002) The approach was based on the inexact graph isomorphism matching technique. Zhuge uses structural ontology to identify the distance of a node on a workflow graph. However, Zhuge's focus was on identification of similarity between graphs for retrieval, the identification of changes made was not considered.

CONCLUSION

Workflow Matching (WM) is motivated by the problem of detecting changes for process change management purposes. In this paper, a methodology for WS changes identification that allows users to detect and view the distance and similarity of two versions of a WS has been presented. This paper describes the application of WM and a prototyped system (wfChange) developed to test the WM algorithm. An example taken from a construction process is used to test the WM algorithm. The result of the matching is shown.

REFERENCES

1. Chawathe, S. S., Rajaramana, A., Garcia-Molina, H. and Widom, J. (1996) Change Detection in Hierarchically Structured Information, Proc. of the 1996 ACM SIGMOD International Conference on Management of Data. Montreal, Canada
2. Chung, P. W. H., Cheung, L., Stader, J., Jarvis, P., Moore, J. and Macintosh, A. (2003) Knowledge-based process management--an approach to handling adaptive workflow, *Knowledge-Based Systems,* **16,** 149-160.
3. Ellis, C., Keddara, K. and Rozenberg, G. (1995) Dynamic change within workflow systems, *Wirtschaftsinformatik,* **37,** 613-615.
4. Hollingsworth, D. (1995) Workflow Management Coalition.
5. Koch, I. and Lengauer, T. (1997) Detection of distant structural similarities in a set of proteins using a fast graph-based method, Proceedings of the 5th International Conference on Intelligent Systems in Molecular Biology. Halkidiki, Greece
6. Kubicka, E., Kubicki, G. and Vakalis, I. (1990) Using graph distance in object recognition, Proceedings of the 1990 ACM annual conference on Cooperation. Washington, D.C., United States
7. Levenshtein, V. A. (1966) Binary codes capable of correcting deletions, insertions, and reversals, *Cybernetics and Control Theory,* **10,** 707-710.
8. Lipton, R. J., Marr, T. G. and Welsh, J. D. (1989) Computational approaches to discovering semantics in molecular biology, Proc. IEEE.
9. Myers, R., Wilson, R. C. and Hancock, E. R. (2000) Bayesian graph edit distance, *Ieee Transactions on Pattern Analysis and Machine Intelligence,* **22,** 628-635.
10. van der Aalst, W. M. P. and Jablonski, S. (2000) Dealing with workflow change: identification of issues and solutions, *Computer Systems Science and Engineering,* **15,** 267-276.
11. W3C (2004), Vol. 2004 W3C Technical Report.
12. Wagner, R. A. and Fischer, M. J. (1974) The String-to-String Correction Problem, *Journal of the ACM (JACM),* **21,** 168-173.
13. Wang, Y., DeWitt, D.J., and Cai, J-Y. (2003) X-Diff: An Effective Change Detection Algorithm for XML Documents, Proceedings of the 19th International Conference on Data Engineering. Bangalore, India
14. WMC-Specification (2002) Workflow Management Coalition.
15. Yeoh, M. L., Chung, P. W. H., Anumba, C. J., El-Hamalawi, A. and Motawa, I. A. (2003) Supporting Construction Processes Using Workflow Technologies, Proceedings of the 2nd International Conference on Innovation in AEC. Loughborough, UK
16. Zhuge, H. (2002) A process matching approach for flexible workflow process reuse, *Information and Software Technology,* **44,** 445-450.