INDIAN INSTITUTE OF MANAGEMENT CALCUTTA

WORKING PAPER SERIES

WPS No. 628/ September 2008

Workflows as UML Activity Diagrams: Analytical Methods for Control-Flow Verification

by

M Hema Sundari
Doctoral student, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700104, India

Anup K. Sen
Professor, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700104, India

and

Amitava Bagchi
Professor, Indian Institute of Science, Education and Research, Kolkata, IIT Kharagpur Extension Centre, HC Block, Sector III Saltlake, Kolkata – 700106, India

Workflows as UML Activity Diagrams: Analytical Methods for Control-Flow Verification

M Hema Sundari Anup K Sen
Indian Institute of Management Calcutta
Joka, D H Road, Kolkata – 700 104, INDIA
{hema, sen}@iimcal.ac.in

Amitava Bagchi
Indian Institute of Science, Educ & Research
Salt Lake HC III. Kolkata – 700 106, INDIA
bagchi@iimcal.ac.in

Abstract

A workflow is a directed graph that depicts a business process. It consists of tasks linked together by AND and XOR connectors, and is usually drawn as a Petri Net or a UML Activity Diagram; the latter notation has gained in popularity in recent years. Workflows sometimes suffer from behavioral defects such as deadlock, lack of synchronization, and interminable looping. Here we present new analytical results useful for verifying the control flow in workflows represented as UML Activity Diagrams. We make use of the notion of 'corresponding pair' and focus on loop-free structures, allowing connectors to form non-nested and overlapping patterns. The analysis can be extended to workflows with loops. To clarify the significance of our results, we provide illustrations of theoretical and practical interest.

Keywords: Workflow, Business process, Verification, Control flow, Analytical method

1. Introduction: A workflow is used by a business organization to perform a specific business function, and is usually depicted as a directed graph consisting of tasks linked together by AND and XOR connectors. A newly created workflow might contain defects that cause control flow errors such as deadlock, lack of synchronization and interminable looping, and dataflow errors such as missing data, redundant data and lost data. It is therefore necessary to verify it before employing it in practice. The verification problem is known to be difficult; no existing algorithm or software package does a completely satisfactory job.

Workflows are often represented as Petri Nets [2]; their control flow can be verified by the Woflan package [10]. In recent years, other graphical formalisms such as UML Activity Diagrams [4], BPMN [7] and Metagraphs [3] have gained in popularity. UML Activity Diagrams have the advantage of being convenient for both control flow and data flow analysis [8]. New analytical results that pertain to this formalism for detecting control flow errors in non-nested

(unstructured) workflows are of topical interest. Our approach here has two distinctive features: a) we provide, for the first time, control-flow verification procedures based on analytical results for a formalism other than Petri Nets; b) our analysis makes use of the interesting notion of 'corresponding pair' [5,6]. We focus on the loop-free case, but our results can be extended to workflows that contain loops and those that employ connectors other than AND and XOR [1].

1.1 Illustrative Example: MHA Inc. manufactures modern household appliances. It wants to increase its production capacity and build a new factory within its premises. It floats a tender inviting construction firms to apply for the project. A firm responds to the tender call, but since the processing of the application by MHA might take time, it formulates in parallel a budget estimate and plan of work. These are submitted to MHA and are considered for approval once the tender application has been accepted. The UML Activity Diagram *G* for handling the applications is shown in Figure 1. *G* is non-nested and has two case instances, *i.e.*, control flows in *G* in two different ways depending on the choice of the outgoing path at the XOR split. In the first, which contains tasks a, b, d and e, the firm's tender application is rejected. The AND join never gets activated because task c is not part of the case instance and does not get performed. Although tasks b and d get done unnecessarily, this gives a realistic picture of how the firm would function. The second relates to the acceptance of the tender application and consists of tasks a, b, c, d and f; here the AND join does get activated.

2. Analysis of the Loop-free Case: Let *G* be a loop-free workflow with XOR and AND split/join connectors. *G* has the following structural characteristics:

- All splits and joins are binary. This causes no loss in generality, since, for example, a ternary split (join) can always be broken up into two binary splits (joins).
- The START connector *s* has no incoming edge and one outgoing edge, and the END connector *r* has one incoming edge and no outgoing edge.
- Tasks can follow one another without intervening connectors, and a connector can follow another connector.
- *G* has no simple structural defects such as dangling nodes; there is a directed path from *s* to every node and from every node to *r*.

- Deadlock, lack of synchronization and inactive AND joins (see below) are the only errors that can arise when *G* is processed.

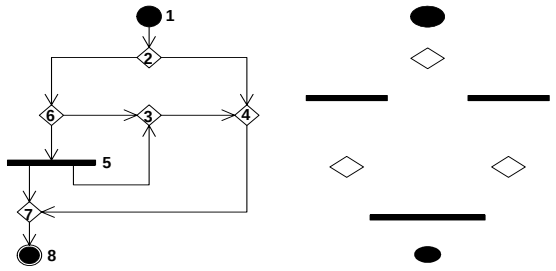
Definition 1: A *corresponding pair* (*cp*) is a pair (X,Y) of connectors in *G*, where X is a split connector, Y is a join connector, and there are two outgoing paths starting at X that meet at Y but not at any predecessor of Y. *CP* refers to the set of cps of a given workflow. A cp (X,Y) is a *strong corresponding pair* (*scp*) if there is no join connector Z that is a predecessor of Y in *G* such that (X,Z) is a cp. *SCP* refers to the set of scps in *G*. Different pairs of outgoing paths from the split connector X meet at different join connectors Y, Z, W, …, for the first time. Typically, one of these pairs defines an scp, while the others define cps. In overlapping structures (see Figures 3,4), two cps with X as the first element can both be scps. A *weak corresponding pair* (*wcp*) is a cp that is not an scp. *WCP* refers to the set of wcps in *G*. So *CP* = *SCP* $\cup$ *WCP*, and *SCP* $\cap$ *WCP* = Φ (empty set). A cp (X,Y) is *proper* if X and Y are connectors of the same type (either both XOR or both AND); otherwise it is *improper*.

These definitions generalize the ones given in [5, 6]; 'corresponding pair' there is the same as 'strong corresponding pair' here. We

other outgoing paths from connector 2, namely the paths 2-3-5 and 2-4-5, meet for the first time at connector 5. Since connector 4 is a predecessor of connector 5, (2,5) is in *WCP* and not in *SCP*. The cps (2,4) and (3,5) are proper, but the cp (2,5) is improper. Thus in this example *SCP* consists only of proper cps, but *WCP* contains only one cp and it is improper. □

Definition 2: A *lack of synchronization error* arises when both incoming edges at an XOR join are simultaneously active. A *deadlock error* arises when: i) not all incoming edges at an AND join are simultaneously active; and, ii) flow of control is halted at the AND join so it is not possible to reach the END node. In a *well-behaved* workflow, neither of these errors ever occurs. Sometimes a workflow is well-behaved but an AND join remains inactive in some case instance. From an operational point of view this is not as serious as a deadlock error and might even help to portray a realistic situation, as in Figure 1. But an *inactive AND join* has often been regarded as an undesirable feature as it can lead to unnecessary execution of tasks. A well-behaved workflow is *strongly well-behaved* if no inactive AND joins ever occur.

Example 2: A workflow with a single cp consisting of an AND split and an XOR join suffers from a lack of synchronization error, and one with a single cp consisting of an XOR split and an



AND join suffers from a deadlock error. In the loop-free workflow *G* of Figure 2, *SCP* = {(2,4), (5,7), (6,3)}, *WCP* = {(2,7), (6,7)}, and (5,7) is the only improper cp. There is lack of synchronization at connector 7 when the left outgoing paths are taken at connectors 2 and 6. If connector 4 is changed to an AND join, there is deadlock at 4. If 4 is changed back to an XOR join, and 5 is changed to an XOR split, *G* consists entirely of XOR connectors and is strongly well behaved, and all cps are proper. The workflow of Figure 1 is well behaved since control reaches from the START node to the END node in both case instances; it is not strongly well behaved since connector 4 is inactive in one case instance. □

Definitions 3: In a *nested* workflow, splits and joins occur in pairs as in nested balanced parentheses. A workflow that is not fully nested is said to be *non-nested*. If the two connectors in a nested pair are of different types, say (AND, XOR) or (XOR, AND), an error will occur. When they are of the same type, they are said to be *matched*

strongly well-behaved.

Proof: Let (X,Y) be an improper cp in *SCP*. Suppose (X,Y) is of type (AND,XOR), and consider a workflow instance that contains the connectors X and Y. If both incoming paths at Y are active then there is a lack of synchronization error at Y. An attempt at merging the signals in the two paths by means of a path (W,Z) of type (XOR,AND), as shown in Figure 5, makes Z inactive when the right hand outgoing path is selected at W; moreover, (X,Y) ceases to be in *SCP*. If W is an XOR split at any other position, or W is an AND split belonging to some other instance, Z is inactive and *G* is not strongly well-behaved. When (X,Y) is of type (XOR,AND), the only way to make *G* strongly well-behaved is by means of an overlap as shown in Figure 6, where

Suppose there is an edge (W,Z) as shown, so that (X,Y) is not in *SCP*.If W is of type AND, then *SCP* contains an improper cp (W,Y) of type (AND,XOR). If W is of type XOR and Z is of type AND, there is either no error or an inactive AND join. So both W and Z must be of type XOR. But then the cp (X,Z) is of type (AND, XOR) and is contained in *SCP*. Note that if there is a connector Y' between W and Z, then Y' can only be an AND join; this will lead to an inactive AND join in some case instances. □

Example 5: Even when there is a cp of type (AND,XOR) in *SCP*, no lack of synchronization error might result. In Figure 7, the cps (4,11) and (6,10) only cause an inactive AND join. □

Stronger results sometimes hold when a loop-free workflow can be drawn on the plane.

Theorem 4: Let the workflow *G* be loop-free, planar and strongly well-behaved. Then every cp in *CP* is proper. If *G* is also reduced, then the connectors in *G* are all of the same type (either all XOR or all AND).

Proof: Since *G* is loop-free and planar, it can be drawn on the plane with no crossovers. It will then have a mesh-like structure with polygonal regions (see Figure 8). *G* is also strongly well-behaved, so all nested pairs in *G* must be matched; we can therefore assume *G* is reduced. In Figure 8, the X's are splits and the Y's are joins. It is easily checked that if a single Y is of type AND, then all splits and joins must be of type AND; otherwise there is an error or an inactive AND join. Similarly, if a sing then th-yth pol7.25c7 fe spla1dgur(7p ng thI0.2lTD0.0003 Tc9 T-23.005906 G)Tj

SCP are proper, W must be an AND split and Z an XOR join. This immediately implies that when the left outgoing path is taken at X, there will be a deadlock error at Y. The same error occurs when there is more than one such path (W, Z) in parallel. The only way to resolve the problem would be by an overlap as shown in Figure 6, but *G* is by assumption planar.

b) Let (X,Y) in Figure 5 represent an improper cp of type (AND,XOR). Then W must be an XOR split and Z an AND join. If we take the right path at W, an inactive AND join occurs at Z.□

Theorem 6: Let the workflow *G* be loop-free and planar.

a) If *SCP* contains an improper cp of type (XOR,AND), then *G* suffers from deadlock error or has an inactive AND join.

b) If *SCP* contains an improper cp of type (AND,XOR), then *G* suffers from lack of synchronization error or has an inactive AND join.

Proof: See the proofs of

each split and join to be either AND or XOR, there are a total of 64 cases. Of these, one has AND connectors only and another has XOR connectors only. Five of the remaining 62 are well behaved and the remaining 57 are erroneous (Figure 4 shows one of them), but not all are distinct. Figure 3 is the standard example of a strongly well behaved overlapping structure. □

2.2 Verification Procedure for Loop-free Workflows: We now proceed to the verification procedure for loop-free workflows, assuming that the workflow, if it is overlapping, has the basic structure of three splits and three joins as shown in Figures 3 and 4.

Step 1: Using a graph algorithm for detecting loops, determine whether the workflow *G* is loop-free. If yes, proceed to Step 2, else exit.

Step 2: If *G* contains mismatched nested pairs then *G* is not well-behaved; give error message and exit. Else collapse the nested pairs, get reduced workflow *G'* and proceed to next step.

```
/* input: a loop-free workflow G */
{  initialize SCP, WCP, CP to empty sets;
   for every split node X in workflow G do {
      find every join Y reachable from X along two
      outgoing paths such  that the two paths meet at Y for
      the first time;  let XCJ be this set of joins;
      determine the subset of joins XSJ in XCJ which have
      no  predecessors in XCJ;
      XCP = { (X,Y) | Y ∈ XCJ };
      XSCP = { (X,Y) | Y ∈ XSJ };
      XWCP = XCP –( }; )Tj-11.6235m
```

Step 3: Determine *SCP* and *WCP* for *G'* using the algorithm *findSCP&WCP* (see Table 2).

Step 4: Check whether there is a split connector X in *G'* such that two different cps (X,Z) and (X,W) are both contained in *SCP*. If yes, then *G'* must be overlapping; compare against known cases and see if it is (strongly) well-behaved. Otherwise, *G'* is planar. Using Table 1, find out whether *G'* is (strongly) well-behaved. In either case, output an appropriate message and exit.

3. Workflows with Loops: The analysis can be extended to workflows with loops. In this

case, it is not possible to define a consistent predecessor-successor relationship between connectors, so *SCP* can not be defined. However, all the cps in CP can be identified. Such a workflow will be valid only when each loop contains an XOR split for control to flow out and an XOR join for control to flow in. In addition, AND joins in loops must satisfy certain conditions, which are not specified here owing to lack of space.

4. Conclusion: We have derived analytical results and described simple verification procedures for loop-free workflows consisting of AND and XOR connectors. These methods are adequate for verifying most situations that arise in practice or are cited in the technical literature. Various other types of connectors have been proposed in [1], *e.g.*, inclusive ORs. The theoretical results and verification procedures can be extended to these connectors, as well as to workflows with loops. Since both the control flow and the data flow must be verified in a workflow, it would be advisable to model a workflow as a Document Driven Activity Diagram [8, 9]. The ultimate objective is to make use of the notion of 'corresponding pair' and develop a comprehensive verification procedure capable of detecting both control flow and data flow errors.

References:

1. Aalst W V P, Hofstede A H M Kiepuszewski B, Barros A P, Workflow Patterns, *Distributed and Parallel Databases 2003*, 14(1): 5-51, 2003.

2. Aalst W V P, Hee K v, *Workflow Management: Models, Methods, and Systems*, The MIT Press, 2002.

3. Basu A, Blanning R W, A formal approach to workflow analysis, *Info Sys Res* 11(1): 17-36, 2000.

4. Booch G, Rumbaugh J, Jacobson I, *The UML Reference Manual*, Addison-Wesley, 1998.

5. Kiepuszewski B, Hofstede A H M, Bussler C, On Structured Workflow Modeling, *Proc CAiSE'2000*, LNCS vol 1797, Springer Verlag, 2000.

6. Liu R, Kumar A, An Analysis and Taxonomy of Unstructured Workflows, *BPM 2005*, LNCS 3649, Springer-Verlag, 268-284, 2005.

7. OMG, *Business Process Modeling Notation Specification, BPMN 1.0,* Feb 2006.

8. Sun Sherry X, Zhao J L, Nunamaker J F, Sheng O, Liu R, Formulating the Data Flow Perspective for Business Process Management, *Info Sys Res*, 17(4): 374-391, 2006.

9. Wang J, Kumar A, A framework for document-driven workflow systems, *BPM 2005*, *Lecture Notes in Computer Science*, vol 3649, Springer Verlag, 285-301, 2005.

10. Verbeek H M W, Basten T, Aalst W V P, Diagnosing Workflow Processes using Woflan, *The Computer Journal*, 44(4): 246-279, 2001.