Organizations today face increasing pressures to integrate their processes across disparate divisions and functional units, in order to remove inefficiencies as well as to enhance manageability. Process integration involves two major types of changes to process structure: (1) synthesizing processes from separate but interdependent subprocesses, and (2) decomposing aggregate processes into distinct subprocesses that are more manageable. We present an approach to facilitate this type of synthesis and decomposition through formal analysis of process structure using a mathematical structure called a metagraph.

1. Introduction

An important purpose of organizations is to implement processes such as order fulfillment, processing of applications (e.g., loan applications), new product development, and responding to existing or impending emergencies, such as the loss of a key account (Fleisch and Osterle 2000). A process is defined as a collection of tasks that transform a given set of inputs into a desired set of outputs. The inputs and outputs may be informational, such as documents (e.g., loan applications), or physical (e.g., raw materials or subassemblies), and the tasks may be information processing tasks such as credit checks, or physical tasks such as machining or shipment.

The tasks used to realize a process under a particular set of conditions (e.g., the result of a credit check) is defined as a workflow (Basu and Blanning 2000). Thus, a process may include several alternative workflows. Furthermore, it may happen that while certain workflows in a process will produce an appropriate output (e.g., a loan decision), others may not execute properly, because of poor design or unforeseen circumstances.

For example, let the process be the approval of a travel request. Consider two possible conditions: one when the requested travel cost is less than a given threshold and one when it is more. There are thus two workflows for this process, each corresponding to one of these conditions. For instance, the workflow in the second case may include a review and approval by a department chairman, in addition to the tasks needed for the first case. If there are other conditions that affect the specific tasks to be performed, such as whether the travel is domestic or foreign, the process could have several additional workflows.

The growing application of computer and communication technology has led to the rise of organizations that integrate the activities of separate organizations and components of organizations to produce a whole that exceeds the sum of its parts (Davidow and Malone 1992). Lack of process integration in such organizations can result in a number of managerial problems. For example, the same work might be repeated in multiple tasks (possibly across multiple subprocesses), and there may be no coordinated management of several disparate subprocesses that could better be managed as a single composite
process. In addition, a combined process (single- or multiorganizational) may become overly complex and thus cumbersome to manage if it contains too many tasks, resources, or other components.

The structural changes to a business process caused by interorganizational linkages can be characterized in terms of two basic types of changes: synthesis and decomposition. Synthesis involves the combination of multiple processes into a single composite process. It can be useful as a means to remove redundancy of tasks and to enhance manageability. For example, a packaging and handling process within a firm such as Ingram Book Company could be combined with an order-taking and online sales process at a firm such as Amazon.com and a shipping and tracking process within a firm such as FedEx to make up a single order fulfillment process, say for Amazon.com. Note that synthesis involves more than simply linking the component processes together. For instance, before linking the different firms, Amazon.com may have an order status page on its Web site. Once the processes are synthesized, this page may have to be modified, and may link to FedEx’s tracking process in a way that is different from FedEx’s tracking page on its own Web site. Leaving the original processes in place would result not only in possible confusion, but also potential redundancies and inefficiencies.

On the other hand, process decomposition identifies components of a process that could be better organized as a separate subprocess, thus simplifying the structure of the original process. Note that synthesis and decomposition do not necessarily undo each other. For instance, a possible two-stage approach to process integration is (1) to synthesize disparate processes to remove redundancy and inconsistency, and then (2) to decompose the synthesized process to enhance manageability through process simplification and delegation. In our book industry example, each of the partner firms may have a subprocess for interactively acquiring customer data. In the combined process, it may be useful to extract this data entry process as a separate process, that not only feeds the relevant tasks between the component processes in order fulfillment, but possibly also feeds other business processes in one or more of the firms.

Since the processes involved may be fairly complex, involving many tasks, information elements, and resources, it is important that the result of any synthesis and decomposition not be a collection of incomplete, incompatible, or wasteful processes (Curtis et al. 1992). Effective process integration requires that (a) each individual workflow within a process remain integral and adequate, (b) there is a single workflow for each situation, and (c) the interactions between the workflows in multiple interacting processes remain effective, even if the processes may be instantiated as distributed workflows (Kwan and Balasubramanian 1997, Tan and Harker 1999).

In addition to these structural issues, there are several other types of issues in process synthesis and decomposition. For instance, a variety of legal issues may arise, especially when decomposition results in outsourcing parts of the process, such as intellectual property protection, accountability, and liability. Human resource management issues are relevant too, especially when decomposition or synthesis results in the transfer of parts of the process to new functional areas of the organization (e.g., from manufacturing to distribution). Yet another category of issues consists of security concerns. While all of these types of issues are important, we limit our attention here to structural issues in the design of processes, primarily those involving information flows within a process and between a process and its subprocesses.

In this paper we identify several structural problems that could arise in the synthesis and decomposition of processes and their constituent workflows, and show how these problems can be analyzed using formal properties of a graph-theoretic construct called a metagraph. We start in §2 by briefly reviewing metagraphs and introducing three properties of metagraphs that are useful in our process analysis: full connectivity, nonredundancy, and independence. In §3 we define well-structured processes, and show how the above metagraph properties can be used to test whether a process is well structured. Then in §4 we explain how these properties may be applied to the synthesis and decomposition of processes. Finally, §5 presents some directions for future research.
2. Metagraph Properties

Metagraphs (Basu and Blanning 1994a, b; 1998; 2000), are graphical structures in which edges represent directed relationships between sets of elements. They extend both directed graphs (by allowing multiple elements in vertices) and hypergraphs (by including directionality in edges). Thus, they support important and well-known graph theoretic properties such as connectivity, while modeling the features of systems more naturally than traditional graph-theoretic constructs such as simple graphs, digraphs, and hypergraphs.

We start by reviewing some basic notions about metagraphs. Consider a set \( X \), called a generating set. Each member \( x \in X \) is called an element (or sometimes an information element). A metagraph is an ordered pair \( S = \langle X, E \rangle \), in which \( E \) is a set of edges. Each edge \( e \in E \) is an ordered pair \( e = \langle V_e, W_e \rangle \), in which \( V_e \subseteq X \) is the invertex of \( e \) and \( W_e \subseteq X \) is the outvertex of \( e \). We require that for each edge \( V_e \cup W_e \neq \emptyset \). The pure inputs of a metagraph are those elements that do not occur in any outvertex (i.e., \( PI = \bigcup V_e \setminus W_e \)), while the pure outputs are those that do not appear in any invertex (i.e., \( PO = \bigcup W_e \setminus V_e \)). A simple example of a metagraph is shown in Figure 1, in which there are four elements \( a, b, c, \) and \( d \), two propositional elements \( p_1 \) and \( p_2 \), and four edges \( e_1, \ldots, e_4 \).

An important issue in metagraphs is connectivity between elements. A simple path is a sequence of edges connecting a pair of elements. For example, \( \langle e_1, e_2 \rangle \) is a simple path connecting \( a \) to \( b \), as is \( \langle e_1, e_4 \rangle \). However, in many cases it is necessary to connect sets of elements, rather than pairs of elements, and this requires a more complex construct, called a metapath. A metapath is a set, rather than a sequence, of edges connecting one set of elements to another set of elements. Just as there may be several simple paths connecting two elements, there may be several metapaths connecting two sets of elements. We say that a metapath connecting two sets of elements is edge dominant if there is no other metapath connecting the same two sets of elements that is a proper subset of the first metapath.

A more detailed development of these concepts, including an algebraic framework for analyzing metagraphs, is summarized in Appendix A and is described in detail in a number of references (Basu and Blanning 1994a, b; 1998; 2000). To analyze how process synthesis and decomposition can factor in dependencies among information elements, tasks, and resources, we need to introduce some additional properties of conditional metagraphs, namely full connectivity, nonredundancy, and independence. These are discussed in the remainder of this section.

2.1. Connectivity and Redundancy

We now introduce two important properties of metagraphs. The first, connectivity (and especially full connectivity), determines the ability of a metagraph to connect certain input variables to certain output variables. The second property is redundancy—that is, a determination of whether there is more than one way to connect an input to an output. We begin with some definitions.

**Definition 1.** Given a conditional metagraph \( S = \langle X_p \cup X_v, E \rangle \), any two sets \( B \subseteq X_v \) and \( C \subseteq X_v \), and \( R \), a defined set of logic expressions over \( X_p \), let \( M(B, C, S) \) be the set of all edge-dominant metapaths from \( B \) to \( C \). An interpretation \( I(X_p, R) \) is an assignment of truth values to the propositions in \( X_p \) such that all the expressions in \( R \) evaluate to true. \( P \subseteq X_p \) denotes the set of true propositions in \( I(X_p, R) \) and \( Q \subseteq X_p \) denotes the set of false propositions in \( I(X_p, R) \).

To illustrate this, consider again the example metagraph in Figure 1. Let \( B = \{ a, c \} \), \( C = \{ d \} \), and \( R \) be the single logical expression \( (p_1 \lor p_2) \). Then \( M(B, C, S) = \{ \langle e_1, e_2, e_3 \rangle, \langle e_1, e_4 \rangle \} \).
We note from Appendix A that a context metagraph \(K(P, Q, S)\) corresponding to an interpretation \(I(X_p, R)\) is a simple metagraph (i.e., it has no propositions), since \(P \cup Q = X_p\).

Definition 2. Given a conditional metagraph \(S = \langle X_p \cup X_v, E \rangle\), any two sets \(B \subseteq X_v\) and \(C \subseteq X_v\), and \(R\), a set of logic expressions defined over \(X_p\):

1. \(B\) is said to be connected to \(C\) with respect to \(R\) if for some interpretation \(I(X_p, R)\), 
   \(|M(B, C, K(P, Q, S))| \geq 1\).

2. \(B\) is said to be fully connected to \(C\) with respect to \(R\) if for every interpretation \(I(X_p, R)\),
   \(|M(B, C, K(P, Q, S))| \geq 1\).

3. \(B\) is said to be nonredundantly connected to \(C\) with respect to \(R\) if for every interpretation \(I(X_p, R)\), 
   \(|M(B, C, K(P, Q, S))| \leq 1\).

It follows from (2) and (3) above that \(B\) is fully and nonredundantly connected to \(C\) if \(|M(B, C, K(P, Q, S))| = 1\) for every interpretation \(I(X_p, R)\).

Definition 3. A conditional metagraph \(S = \langle X_p \cup X_v, E \rangle\) is nonredundant with respect to \(R\), a set of logic expressions over \(X_p\), if in the context metagraph \(K(P, Q, S) = \langle X_p \cup X_v, E \rangle\) corresponding to any interpretation \(I(X_p, R)\), \(\forall x \in X_v\), we have 
\(|\{e \in E_x \mid x \in W_x\}| \leq 1\).

Informally, a metagraph is nonredundant if for every interpretation each element is in the outvertex of at most one edge. In other words, there is at most one way to determine the value of the element in each interpretation, so there is no ambiguity.

Given the algebraic representation of a conditional metagraph in terms of its adjacency matrix \(A\) and closure, \(A^+\), the set of metapaths from one set of elements \(B\) to another set of elements \(C\) can be identified using \(A^+\). The procedure is as follows:

1. \(A^+\) is reduced to the rows corresponding to \(B\) and the columns corresponding to \(C\) (since all edge-dominant metapaths from \(B\) to \(C\) can be constructed from this submatrix; Basu and Blanning 1994b).

2. For each interpretation, the corresponding reduced context metagraph is generated (as described in Basu and Blanning 1998).

3. All valid metapaths from \(B\) to \(C\) in the context are then constructed for that interpretation using the procedure specified in (Basu and Blanning 1994b).

The computational complexity of the above procedure depends upon the size of the proposition set \(X_p\). In general, the number of possible interpretations of a given set of propositions and set \(R\) of expressions is exponential in \(N\), the number of propositions (the worst-case complexity is \(2^N\)). However, in many practical situations the number of possible interpretations will be sufficiently small to render the procedure feasible. For instance, in the context of process modeling, the number of interpretations of \(R\) corresponds to the number of alternate workflows for the represented process, which is not likely to be very large for typical business processes. Also, \(R\) may have some special structure that can be exploited to restrict the search for interpretations. For example, in process modeling, \(R\) may contain a number of expressions of mutual exclusivity between complementary literals (e.g., \(X_p = \{\text{salaried}, \text{hourly}, \text{high-risk}, \text{low-risk}\}\) and \(R = \{\text{salaried} \oplus \text{hourly}, \text{(high-risk} \oplus \text{low-risk)}\}\).

Given a metagraph \(S\) with pure inputs \(PI\) and pure outputs \(PO\), we can test whether \(PI\) is fully connected to \(PO\) in \(S\) as follows:

1. Let \(S\) be a simple metagraph. Then \(S\) is fully connected if there exists any \(M(PI, PO)\).

2. Let \(S\) be a conditional metagraph, with a set of propositions \(X_p\) used as assumptions. Assume that all \(X_p\) are pure inputs. Then \(S\) is fully connected iff there exists a \(M(PI, PO)\) for every possible interpretation of \(X_p\). In effect, this implies that \(S\) is fully connected iff there exists a \(M(PI, PO)\) even when all \(X_p\) elements are false.

In addition, let \(R\) be a set of Horn clause assertions on \(X_p\). Again, \(S\) is fully connected iff there exists a \(M(PI, PO)\) even when all \(X_p\) elements are false. The only additional feature here is that even when the truth value of all propositions in \(X_p\) are not explicitly known, the unknown propositions that are heads of clauses in \(R\) can be inferred.

Consider the union \(S_2\) of two fully connected metagraphs \(S_1\) and \(S_2\) (i.e., \(S_3 = S_1 \cup S_2\)). Since \(PO_1\) can be reached from \(PI_1\) in all interpretations, and \(PO_2\) can be reached from \(PI_2\), then \(PO_1 \cup PO_2\) is reachable from \(PI_1 \cup PI_2\) in all interpretations. Then it follows that the pure inputs \(PI_3 \subseteq PI_1 \cup PI_2\), and the pure outputs \(PO_3 \subseteq PO_1 \cup PO_2\).
However, it is important to realize that the pure inputs and pure outputs of the combined metagraph need not be \(PI_1 \cup PI_2\) and \(PO_1 \cup PO_2\), respectively. Therefore, it does not necessarily follow that \(S_3\) is fully connected, as is demonstrated by the following example. Let \(S_1\) consist of the edge \((a, b), (c, d)\) and \(S_2\) consist of the edge \((d, f), (b, g)\), as shown in Figure 2. If \(S_3 = S_1 \cup S_2\), then \(PI_3 = \{a, f\}\) and \(PO_3 = \{c, g\}\), and there is no metapath from \(PI_3\) to \(PO_3\). Thus, \(S_3\) is cyclic, \(S_3\) is not fully connected. Although neither \(S_1\) nor \(S_2\) is cyclic, \(S_3\) is cyclic.

**Theorem 1.** Given two fully connected metagraphs \(S_1\) and \(S_2\), their union \(S_3 = S_1 \cup S_2\) is also fully connected if it is acyclic.

**Proof.** Since \(S_3\) is acyclic, its elements can be organized in a partial order based on the existence of simple paths between elements (i.e., \(p\) precedes \(q\) if there is a simple path from \(p\) to \(q\)). It follows that the elements in \(PI_3\) are roots, and elements of \(PO_3\) are leaves of the precedence graph.

Consider an element \(x\) in \(PO_3\) that is not reachable from \(PI_3\) in some interpretation. Without loss of generality, assume that \(x\) is in \(PO_1\). Since \(x\) is reachable from some subset of \(PI_1\), say \(PI_{1r}\), then it must be true that \(PI_{1r} \setminus PI_3 \neq \emptyset\). Let \(PI_{1r} \setminus PI_3 = Y\). Then \(Y\) consists of elements that were in \(PI_1\), but are not in \(PI_3\). Thus, each element of \(Y\) must be either an internal element of \(S_2\) or a pure output of \(S_2\). The precedence graph to each such element has to ultimately end with roots that are in \(PI_{1r}\), say \(PI_{2r}\). If all these elements are in \(PI_3\), then \(x\) is reachable from \(PI_3\) and we are done. However, if \(PI_{1r} \setminus PI_3 = Z \neq \emptyset\), then, as before, these must be internal or pure outputs in \(S_1\). Since \(S_1\) is cyclic, \(Z\) is reachable from a subset of \(PI_1\) say \(PI_{1z}\), such that \(PI_{1z} \cap PI_3 = \emptyset\). Since both \(PI_1\) and \(PI_2\) are finite, these iterations must terminate, which proves the result. Q.E.D.

This theorem provides a two-step test for whether the combination of two fully connected metagraphs is also fully connected, as follows:

1. Compute \(A^*\) and examine its diagonal elements.
   If all of the diagonal cells are empty, then full connectivity still holds because the metagraph is acyclic.
2. Test whether there is any cycle containing elements either from \(PI_1\) and \(PO_3\), or \(PI_2\) and \(PO_1\). If not, then full connectivity still holds.

If neither condition above holds, then full connectivity cannot be assured. In such a case, the resulting metagraph has to be itself tested for full connectivity by looking for a metapath from its pure inputs to its pure outputs in each interpretation.

It is possible to determine whether a given metagraph is nonredundant in a given context, using the algebraic representation. The incidence matrix \(G\) can be adapted to the context by reducing it based on the elements in \(P\) (the true propositions) and \(Q\) (the false propositions), as described in Basu and Blanning (1998). Then, the resulting metagraph is nonredundant if each row has at most one “+1” entry. Otherwise, there are two or more edges (tasks) that produce the same output. As before, the complexity of checking whether the metagraph is nonredundant in all contexts depends upon the number of valid contexts, which in turn is determined by \(R\) (the remaining undetermined propositions).

### 2.2. Independent Submetagraphs

We now examine the issue of independence of a submetagraph contained within a larger metagraph.

**Definition 4.** A metagraph \(S' = \langle X', E' \rangle\) is said to be a submetagraph (SMG) of another metagraph \(S = \langle X, E \rangle\), (denoted by \(S' \subseteq S\)) if \(X' \subseteq X\) and \(E' \subseteq E\).

Note that the SMG relationship is defined by edges, not by elements. Thus, it is possible for \(S' \subseteq S\) even if \(X' = X\) as long as \(E' \subseteq E\).
Input Independence: A metagraph $S_1$ is an input independent SMG of a metagraph $S_2$ if every element of $S_1$ that is not a pure input is determined only by edges within $S_1$.

Output Independence: A metagraph $S_1$ is an output independent SMG of a metagraph $S_2$ if every element of $S_1$ that is not a pure output is used (i.e. as an input) only by edges within $S_1$.

Independence: A metagraph $S_1$ is an independent SMG (denoted ISMG) of a metagraph $S_2$ if it is both input independent and output independent.

Examples of input independence, output independence, and independence appear in Figure 3 and Table 1.

Theorem 2. Given two ISMGs $S_1$ and $S_2$ of a common containing metagraph $S$, then $S_3 = S_1 \cup S_2$ is an ISMG of $S$.

Proof. We prove this by contradiction. All elements that are not either pure inputs or pure outputs of $S_1$ or $S_2$ clearly cannot violate the independence of $S_3$. Let $x$ be a pure input of $S_1$ that violates the independence of $S_3$ by being in the output of some edge outside $S_3$. Clearly $x$ must be a pure output of $S_2$. By definition, though, every pure output of $S_2$ is determined only by edges within $S_2$, and thus $S_3$, which contradicts the claim about $x$. A similar argument holds if $x$ is a pure output of $S_1$ that is an input to some edge outside $S_3$. Thus the result is proved. Q.E.D.

Independence is desirable in the sense that any coordination issues involving an ISMG can be addressed solely in terms of its pure inputs and pure outputs, while this is not true for all SMGs in general.

Table 1  Independence of Some SMGs in Figure 3

<table>
<thead>
<tr>
<th>SMG</th>
<th>PI</th>
<th>PO</th>
<th>Other</th>
<th>Input Ind.</th>
<th>Output Ind.</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>${c_1, c_2}$</td>
<td>$a$</td>
<td>$h$</td>
<td>$d, k, l$</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>${c_1, c_2, c_3}$</td>
<td>$a$</td>
<td>$h$</td>
<td>$k, l$</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>${c_1, c_2, c_3}$</td>
<td>$c, d, h, q$</td>
<td>$l, n, p$</td>
<td>$k$</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

Definition 5. Two metagraphs are mutually independent if each is an ISMG in the metagraph formed by their union.

Theorem 3. If two edge-disjoint SMGs (no common edges) are both ISMGs of a single containing metagraph, then they are mutually independent.

Proof. Since the two SMGs are edge-disjoint and each is an ISMG, the only common elements can be "boundary elements" (i.e. pure inputs or pure outputs), and furthermore, the SMGs must be sequentially related, that is, $PI_1$ can overlap with $PO_2$ but not $PI_2$, and $PO_1$ can overlap with $PI_2$ but not $PO_2$ (the reason for this is that if $PI_1$ and $PI_2$ have any common elements, then because of the independence assumption, all edges containing the common elements have to be in both SMGs, which violates their edge-disjointedness). It then follows that they are each ISMGs of their union, which proves the result. Q.E.D.

Theorem 4. Given two ISMGs $S_1$ and $S_2$ of a containing metagraph $S$, the intersection $S_3 = S_1 \cap S_2$ is also an ISMG of $S$.

Proof. Each ISMG can be viewed as the union of a set of independent metapaths (each itself an ISMG of $S$ and $S_1$), each from some subset of pure inputs to a subset of pure outputs of that ISMG. By the definition of independence, if any edge $e$ occurs in both $S_1$ and $S_2$, every metapath $M(PI_1, PI_2)$ in $S_1$ containing $e$ must also occur in $S_2$. Otherwise, at least one edge in $M$ would violate the independence of $S_1$ and $S_2$. Thus $S_1 \cap S_2$ can also be viewed as the union of a set of independent metapaths from its pure inputs to its pure outputs, and is thus an ISMG of $S$. Q.E.D.

Using the matrix representation of metagraphs, we have developed an algorithmic procedure to test for independence of a given SMG $S'$ in a metagraph.
S, shown in Appendix B. The procedure is quite straightforward, since the columns in A corresponding to all elements in \( S' \) other than the pure inputs have to be empty except for rows corresponding to elements in \( S' \); similarly all rows corresponding to elements in \( S' \) other than pure outputs have to be empty except for columns corresponding to elements in \( S' \).

3. Analyzing Process Structure

In \( \S\S 3 \) and 4, we show how the metagraph properties of full-connectivity, nonredundancy, and independence can be used to analyze the structure of processes and workflows. We start in this section by considering a single process, and discussing how its structure can be analyzed. We assume that the goal of the structural analysis is a well-structured process, as defined below.

**Definition 6.** A process is **well structured** if it is fully connected and nonredundant.

3.1. Existing Approaches to Workflow Representation

Several graph-theoretic approaches have been proposed to represent workflows and processes. A common method of modeling information flows in information systems is based on data flow diagrams (DFDs). A DFD is a digraph in which information flows are represented as edges, and tasks are represented as vertices, as opposed to the metagraph representation used in this paper, in which tasks are edges linking sets of information elements. Modeling of systems using DFDs is a part of many popular systems analysis methodologies. Interestingly, the features of a DFD can also be represented in the metagraph framework, using a construct called a Task Interaction Metagraph (TIM), in which information flows are also represented as edges and tasks as vertices. A TIM is the dual of the type of metagraph described in \( \S 2 \), and can be calculated from the primal metagraph, as shown in Basu and Blanning (2000). It has two advantages over a traditional DFD. First, since a TIM is a metagraph, the connectivity operations described in Appendix A and in the metagraph literature referenced here can be applied to a TIM. For example, a cycle in a TIM identifies tasks that may have to be repeated within a given workflow. Second, it combines its similarity to a DFD, with the additional ability to model many-to-many relationships and support algebraic analysis of such relationships.

Other examples of digraph representations of workflows include Kumar and Zhao (1999), in which dependencies between tasks and agents are represented as dependency graphs and If-Then rules are used to constrain these interactions. In Flowmark® (Leyman and Roller 1997), a workflow structure is represented by a digraph linking process initiation, intermediate information processing tasks (e.g., performing a credit check), and process outputs (e.g., accept credit, deny credit). These digraph representations, like DFDs, provide powerful diagrammatic views of business processes.

Another type of digraph used in workflow modeling is Petri nets (Peterson 1981, Murata 1989). A Petri net is a bipartite digraph in which there are two types of nodes: places and transitions. A place usually denotes an activity, and may be occupied by one or more tokens. When all of the places leading into a transition contain a token, the transition may fire, removing a token from each of the input places and placing a token in each of the output places. Petri nets offer a powerful tool for the representation and analysis of certain network-based discrete dynamic systems, and a special form of Petri nets, called Workflow Nets, have been proposed in the context of workflow management (Aalst 1998).

A variety of digraph representations are found in the Universal Modeling Language (UML), which allows different aspects of a system to be represented by different constructs (Fowler and Scott 1998). One such construct is activity diagrams, which are similar to Petri nets. Other constructs focus on such items as (1) the classes of users, documents, etc. involved in the system, (2) the functionality of the system as perceived by its users, and (3) the sequence of messages passed among users and system components. This allows an analyst to select an appropriate representation from the variety of available representations.

Digraph-based approaches such as these possess the virtue of simplicity, but they have limited expressive power. They describe very well the relationships between pairs of elements—such as tasks, agents, and
documents—but they do not easily represent relationships between sets of elements. Yet in many cases the components of workflow systems are better represented as set-to-set mappings than by point-to-point mappings. For example, the input to a task may consist of several documents, and the output may be several documents, as well. Problems arise when there are two or more combinations of documents, any of which is sufficient to invoke the task. This problem can be resolved with AND/OR digraphs, but the resulting visual representation is clumsy, and there is no established algebraic form for this structure.

A structure that attempts to overcome some of these problems is a higraph (Harel 1988). In a higraph the nodes are blobs, and the edges are blob-to-blob mappings. A blob consists of one or more information elements (i.e., documents) and/or one or more blobs. Thus, blobs are recursive structures—blobs can contain blobs, which can in turn contain other blobs, and so forth. Thus, higraphs are a very powerful construct for representing directed set-to-set mappings.

There has also been considerable progress in the development of computer-based tools for workflow analysis. One way in which workflow models may be implemented is by means of XML-based platforms. The Workflow Management Coalition has proposed a standard for describing and implementing workflow models called Wf-XML (WfMC 1999). A related standard based on so-called XML nets, which are structured along the lines of Petri nets, has been proposed for interorganizational workflows (Lenz and Oberweis 2001). In addition, Microsoft has developed an XML-based product called BizTalk© for e-commerce document messaging systems, and this has been implemented in the context of B2B contracts (Herring and Milosevic 2001). The value of the XML approach is that it provides for a flexible and adaptable interface between processes. However, it does not help deal with problems of analysis and redesign caused by workflow synthesis and decomposition.

3.2. Representing Workflows and Processes as Metagraphs

As mentioned in §1, a process is a collection of tasks or activities that converts a set of inputs into a set of outputs, and a workflow is a particular instantiation of a process, namely the collection of tasks that enact the process in a specific situation. Each task itself converts a set of inputs into a set of outputs. In this section, we briefly describe how metagraphs can be used to provide a black box representation of workflows and processes, based on the approach originally presented in (Basu and Blanning 2000).

The inputs and outputs of tasks, workflows, and processes can be viewed as relevant information elements. In the metagraph representation, each information element in a workflow can be represented as an element of the generating set \( X \), or more specifically, of \( X_v \), the set of variables in the generating set. A collection of information elements comprising a document (form or report) can then be represented as a vertex.

Each task is itself represented as an edge in the metagraph. Assumptions underlying each task can also be represented in terms of a set \( X_p \) of propositions in the generating set and by including the relevant propositions in the invertices of the task edges. Also, resources needed for each task can be represented in terms of a set \( X_r \) of resources included in the generating set. Then, the resources required in each task can be represented as additional inputs to the corresponding edge.

Each process can then be represented by a metagraph. More generally, a metagraph can be used to represent the tasks comprising a collection of possibly related (or overlapping) workflows comprising the process. Each workflow in turn is represented as a metapath from a source set of information elements comprising the inputs of the corresponding process to a target set comprising the process’s output. Since the specific outputs that result from different workflows within a single process could vary (e.g., a loan review might lead to either an approval, or a request for revision of the amount requested), the scope of the complete process can be captured by adding two “dummy” elements, denoting the start and end of the process. This is consistent with other approaches to workflow and process representation, such as Petri nets (Aalst 1998).

We now present an example in the area of loan processing for consumer loans. Consider two processes that are subprocesses of loan processing. The
first process determines the riskiness of a potential loan, based on the data in the loan application. A simplified form of this process is illustrated in Figure 4. There are six variables in the generating set. These are Account Data (AD), Loan Application (AP), Appraised Value (AV), Credit Rating (CR), Loan Amount (LA), and Loan Risk (LR). We note that this is a fully connected, nonredundant, acyclic metagraph. Since the edges from the Start vertex and to the End vertex (e' and e″, respectively) are only used to identify separate workflows, we will ignore them in the following discussion, except where significant.

The second process determines whether the loan should be approved, and a simplified form of this process is illustrated by S₂ in Figure 5. There are six variables and two propositions in the generating set. The variables are Appraised Value (AV), Comparables Basis (CB), Loan Amount (LA), Loan Decision (LD), Loan Risk (LR), and Property Data (PD). The two propositions are whether the risk is low (?RL) and whether the risk is high (?RH). A loan decision can be made only if the risk is low—that is if ?RL is true. For any value of LR either ?RL or else ?RH is true. Note that the variable LA representing the loan amount can change in situations where the loan risk is marginal. That is, rather than rejecting the application, the bank computes a reduced loan amount and considers that instead. This is also a fully connected, nonredundant, acyclic metagraph.

The above discussion illustrates the use of metagraphs for workflow representation. It is worth examining how this approach relates to some of the existing approaches discussed in §3.1. With respect to digraph approaches, the ability to map relationships between sets of elements is so central to representing tasks that the advantages of metagraphs are quite obvious. With respect to derivative approaches such as Petri Nets, the comparison is more interesting, since Petri nets enable m:n relationships to be modeled by partitioning nodes into places and transitions. However, a single metagraph can represent multiple, m:n relationships corresponding to multiple workflows, which is difficult to represent in a single Petri net, as shown in (Basu and Blanning 1992). Similarly, higraphs and state-charts also allow complex m:n relationships to be modeled, and thus offer an alternative to metagraphs. However, if one were to represent higraphs algebraically, it would be necessary to vary the size and composition of the generating set, depending on the contents of the edge set, since the generating set would contain not only information elements but also blobs. Thus, the construction of a new edge might change the generating set, even if the information elements were to remain
the same. This significantly complicates the algebraic procedures needed for testing and exploiting connectivity at the set level. Metagraphs provide a simpler structure that is effective as long as relationships do not have to be defined on a recursive generating set, and this problem does not arise in business process modeling.

In the following two subsections we will use the examples from this section to examine the analysis of process synthesis and decomposition.

3.3. Analyzing Process Structure Using Metagraphs

The metagraph representation of a process can be used to determine whether the process satisfies the definition of well-structuredness presented earlier in this section. This amounts to checking whether the process is fully connected and nonredundant.

Note that each process is represented as a metagraph, in which each complete workflow is a metapath from the “Start” vertex to the “End” vertex. If the process is well structured, then there should be exactly one applicable workflow in each possible interpretation. This means that there should be exactly one corresponding metapath in each interpretation, as well. This can be checked by enumerating the possible interpretations, and in each case, using the algorithms in (Basu and Blanning 1994a) to find all applicable metapaths \( M(\text{Start}, \text{End}) \) in the process metagraph. If for some interpretation there is no such metapath, then the workflow corresponding to that interpretation has to be modified and/or augmented with tasks to achieve connectivity. Similarly, if there are multiple metapaths in any interpretation, then appropriate tasks from the process have to be modified and/or removed to eliminate the multideterminacy.

For example, the Loan Risk process represented by the metagraph \( S_1 \) illustrated in Figure 4 is a simple example of such a process. The pure inputs are \( AC, AP, AV, \) and \( LA \), and the pure output is \( LR \), with \( CR \) as an intermediate element. Since there are no propositions, there is only one interpretation, and there is only one metapath from the pure inputs to the pure output. Thus, the process is nonredundant; it is also acyclic.

Recognizing that there may be multiple processes within an organization, the management of each pro-
cess is simplified if the only interactions of the process with the rest of the organization are through its inputs and outputs. This is not true of all processes. For instance, a control process such as auditing and accounting, has to interact implicitly, with processes such as product creation and order fulfillment at all stages. However, whenever possible, it may also be desirable to have each process correspond to an independent submetagraph. When the process metagraph for such a process is not independent, then, depending upon whether the cause is lack of input and/or output independence, the corresponding tasks have to be reconfigured to restore independence.

4. Analyzing Process Synthesis and Decomposition

As discussed in §1, the restructuring of business processes in organizations can lead to a variety of problems as well as opportunities. In §4, we identify some of the relevant issues, give a brief overview of how metagraphs can be used to represent workflows and processes, and then examine how metagraph analysis based on the properties introduced in the previous section can be used to address some of these issues.

Consider the situation where a business process is composed in terms of existing processes within a collection of collaborating organizations. While each of the component processes is well structured,1 it is possible that the composite process formed by the synthesis of these components is not. For example, there may be situations where the same task is performed in different ways in different parts of the combined organization, and this only becomes evident when the pieces are put together. In cases where the resulting process consists of only a few tasks, such problems may be discernible through inspection or informal analysis. However, in large, complex processes, the problems may be subtler.

Another issue is that a synthesized process may have some special structure that warrants redesign of some sort. This may consist of a decomposition of the aggregate by removing some of the tasks. An important consideration is whether or not the tasks to be removed are independent of the aggregate. Yet another consideration is whether the resources associated with these tasks are independent of the collective set of resources appearing in the aggregate set of tasks. In other words, the factors in such decomposition decisions include not only strategic considerations of core competencies, but also operational considerations of impact on remaining operations and units. Ideally, transitions from internal (hierarchical) to market transactions are easiest when the divested process has well-defined and controllable interfaces with the rest of the organization. For example, in a loan application process for a bank, we may find that a specific subprocess such as risk assessment involves tasks (e.g., risk level computation) and resources (e.g., risk analysts) that have no overlap with other subprocesses. In such a case, it would be safe to consider identifying this subprocess, either to manage it separately, or possibly even to outsource its execution.

4.1. Analysis of Process Synthesis

We now examine how the metagraph representation described in §3 can be used to analyze the effect of combining several well-structured processes. In metagraph terms this requires that the resulting metagraph be well structured. In other words, the conditional metagraph representing the process should be fully connected, so that a metapath exists from “Start” to “End” for all interpretations of the conditional metagraph. In addition, the metagraph should not be redundant, so that in each interpretation of the conditional metagraph, the outvertices of the edges should be pair-wise disjoint.

We assume that the synthesis of two or more workflows is the union of these workflows—that is, it is the union of the tasks that make up the workflows. Therefore, it would be useful to know whether the property of full connectivity, if present in the separate workflows, is preserved by the union operation. Otherwise there will be at least one interpretation for which it is not possible to complete the process. What this means is that if the intended outcomes are decisions, then it will not be possible in some cases to arrive at those

1 We restrict our attention to situations where each component process is well structured, and focus on the problems that might arise in the synthesis.
decisions. Similarly, if the process outputs are to be used in other, subsequent, processes, then these processes will not have the necessary inputs.

The metagraph representation of workflows, along with Theorems 1 and 2, can be helpful in addressing these issues. With regard to the first issue, full connectivity, we can see from Theorem 1 that full connectivity is preserved if the union of the two workflows is acyclic. Actually, full connectivity may hold even if the component workflows are cyclic, as long as the union introduces no new cycles spanning both components. However, in this case the theorem does not apply, and the combined metagraph has to be tested itself for full connectivity.

In addition, the synthesized process may contain some redundancy that may need to be identified and eliminated. Nonredundancy would mean that for each interpretation no two outvertices share any common elements (i.e., that all outvertices are pair-wise disjoint), which is straightforward. Consider the synthesis of the two processes $S_1$ and $S_2$ in our example to produce a new process $S_3 = S_1 \cup S_2$, which is illustrated in the metagraph of Figure 6 (the labels $r_1$–$r_3$ will be explained later). We note that $S_3$ remains nonredundant but is no longer fully connected, since there is an interpretation ($\neg RH = TRUE \& \neg RL = FALSE$) for which there is no metapath from Start to End. $S_3$ is nonredundant because of the exclusive disjunction constraint on $\neg RL$ and $\neg RH$. In addition, the synthesis introduces a cycle that leads to $S_3$ being not fully connected. The cycle consists of $LA$, $LR$, and $\neg RH$, and can be identified by nonempty diagonal members of the closure matrix $A^*$. Detection of a cycle can alert process designers to possible problems. However, the presence of a cycle does not necessarily imply a problem with the process. In our example, if we specify an initial value for $LA$ and then iterate through the evaluation process until a value for $LA$ is reached for which $\neg RL = TRUE$, the process ultimately exits the cycle. For example, we can use $PD$ to calculate an initial value for $LA$ and then use $LA$, $CR$, and $AV$ to calculate $LR$. If the value of $LR$ is too high so that $\neg RH = TRUE$, we reduce $LA$ using $e_6$ (for instance, the task $e_6$ might reduce the loan amount by a fixed percentage, say 20%), and the revised loan is evaluated. If $\neg RH = FALSE$ (and therefore $\neg RL = TRUE$), then the process terminates with a loan approval decision. Otherwise, $e_6$ is used again to further reduce the $LA$ and the cycle repeats (the cycle always terminates since the risk of a loan amount near or at zero is implicitly low).

Figure 6 Combined Metagraph $S_3 = S_1 \cup S_2$

![Diagram](image_url)

Note: Legend for $r_i$: $r_1$: Risk Analyst; $r_2$: Loan Officer; $r_3$: Loan Clerk.
We have learned from this example that the synthesis of well-structured processes can lead to a composite process that is not well structured because the synthesis operation may introduce a cycle into the composite process. On the other hand, the introduction of cycles need not always prevent the composite process from being well structured.

4.2. Analysis of Process Decomposition

Once a synthesized workflow has been created, it may be desirable to decompose the aggregate by removing some of the activities and treating them as separate workflows. For example, we may create a separate management structure for these workflows or we may outsource them. There are many managerial and economic reasons for process decomposition and/or outsourcing, which are beyond the scope of this paper. However, regardless of the motivation, it is important to ensure that such changes do not disrupt the overall process(es), and we can address this problem using the property of submetagraph independence. The result of decomposition would be that the remaining metagraph would be smaller, and potentially simpler in structure, which in turn would result in a process that is simpler and easier to manage.

Note that it may be useful to extract either a single workflow in some cases, or an entire process in other cases, from a containing process. For example, it may be useful to outsource risk assessment of a loan in a particular situation (e.g., when the customer does not have an account at the bank); alternatively, it may be appropriate to outsource the risk assessment process as a whole (i.e., the union of all the workflows making up the risk assessment process). We assume that a workflow can safely—that is, without disrupting any other workflow—be extracted from a containing process if the metagraph representing the workflow is independent of the containing metagraph. Then it follows that in order to extract an entire process or sub-process we would need to consider whether we can extract the union of all of its component workflows.

There are two issues here. The first is whether the union of two independent workflows—that is, two separate workflows that are each independent of the entire aggregate of workflows—will also be independent of the entire aggregate. If that were not the case, then it would be necessary to consider each workflow in turn, and extract it only if it were independent of the current aggregate.

The second issue is the intersection of two workflows. We may wish to create a new and smaller workflow by pulling out the tasks common to two workflows. If the two workflows are independent, the question is whether independence will be preserved in the intersection.

The metagraph representation of workflows, along with Theorems 2, 3, and 4 can be helpful in addressing these issues. With regard to independence of the union of two independent workflows, we can see from Theorem 2 that independence is preserved. In other words, bundling (merging) two independent subprocesses for decomposition will result in an aggregate independent subprocess. In addition, according to Theorem 3 the subprocesses will be mutually independent. We note, however, that redundancy is not necessarily preserved under these conditions. In other words, each workflow might contain procedures for nonredundantly calculating the same information element, but the resulting union could be redundant.

With regard to the intersection of subprocesses, we can see from Theorem 4 that one can create a new workflow by taking all activities common to two independent workflows and decompose the resulting set of common activities. In this case the independence property will be preserved. However, as we have seen, the property of full connectivity will not necessarily be preserved.

The concept of independence also provides guidelines for process decomposition. Consider a process that is a candidate for decomposition. There are two issues here. The first is whether a workflow, as represented by a submetagraph, is independent of the aggregate process—that is, whether it is an ISMG. The second is whether the collection of resources for this submetagraph, in the form of the corresponding resource interaction metagraphs (RIMs) is also an ISMG of the aggregate RIM. There are four possibilities, as illustrated in Table 2.

The first possibility, termed decomposable organization in the table, occurs when both the submetagraph
and the RIM are ISMGs. In this case the corresponding workflow is a good candidate for decomposition. Of course, there will be other criteria for treating the decomposed workflow separately. There are economic and managerial issues that must be considered here, as well as traditions and issues of corporate culture. However, this joint independence suggests that there are no structural (or process-specific) impediments to decomposition.

The second possibility, termed matrix organization in the table, occurs when the submetagraph is an ISMG but the RIM is not an ISMG. In this case the tasks are separable but the process resources are shared with other parts of the resource aggregate. This suggests the use of a matrix structure in which the decomposed sets of tasks (corresponding to the ISMGs) are managed by separate project managers. Resource managers would assign the resources centrally to the various projects.

The third possibility, termed modular workgroups in the table, occurs when the submetagraph is not an ISMG but the RIM is an ISMG. In this case the tasks contained in the subprocess use specialized resources but interact with other tasks outside the subprocess, and the resources can be organized in a module (e.g., a workgroup). Thus, even though the tasks performed within the subprocess require interaction in the form of inputs and outputs with other tasks outside the subprocess, the module/workgroup itself requires less coordination with external resources, since only certain resources interact with other resources.

Table 2  Independence of Submetagraph and RIM

<table>
<thead>
<tr>
<th>Submetagraph</th>
<th>RIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>Decomposable organization</td>
</tr>
<tr>
<td>Dependent</td>
<td>Modular workgroups</td>
</tr>
</tbody>
</table>

The fourth possibility, termed monolithic organization in the table, occurs when neither the submetagraph nor its RIM is an ISMG. In this case the organization cannot be decomposed for structural reasons. Of course, there may be compelling reasons to decompose because of personalities, culture, economics, geographical locations, traditions, or other reasons. However, process managers should realize that there are structural arguments against decomposition.

We note that these independence conditions must hold for all interpretations of the conditional metagraph representing the process in question. However, in some cases there may be only partial independence—that is, independence in some interpretations but not in others. In this case, process managers and analysts may select the organizational alternatives suggested above for most instances and be prepared to override them in the special circumstances.

We illustrate how an analysis of resource interactions can augment the analysis of metagraph independence for process decomposition, using our loan evaluation example in Figure 6. The resources used in the various tasks are indicated in the figure below each edge, and there are three resources—\( r_1 \) a risk analyst, \( r_2 \) a loan officer, and \( r_3 \) a loan processing clerk. The RIM corresponding to Figure 6 is shown in Figure 7. The edge labels in Figure 7 indicate the nature of the resource interactions. For instance, the label \((CR)\langle e_1, e_2 \rangle\) on the edge from \( \{r_2, r_3\} \) to \( \{r_1\} \) indicates that the loan officer and loan clerk compute \( CR \) in the task \( e_1 \) and provide it to the risk analyst for use in edge \( e_2 \).

From the RIM, it is apparent that the risk analyst \( (r_1) \) only works alone, while the other resources work both alone and with each other. Thus, a potential candidate for decomposition is the set of edges using \( r_1 \). We can identify these edges from either the RIM (source edges in labels of all edges emanating from \( r_1 \)), or from the process metagraph. The edges are \( e_2, e_4, \) and \( e_5 \). What next needs to be examined is whether these three edges form an ISMG.

If \( e_2, e_4, \) and \( e_5 \) are extracted as a separate subprocess, we get the metagraph shown in Figure 8 where the extracted subprocess is represented by the new edge \( e' \). Unfortunately, this is not an ISMG,
since LR violates output dependency. However, this only occurs in one of the two possible interpretations of the process (i.e., when ?RH = TRUE and therefore, ?RL = FALSE), and thus represents a case of partial independence. In other words, for all low-risk cases (?RL = TRUE), the risk assessment subprocess represented by e2, e4, and e5 can be extracted as a separate process, since e6 does not apply and thus e' corresponds to an ISMG. However, for high-risk cases (?RH = TRUE), the fact that LR is involved in the subprocess represented by e' is lost, so that S' is not an accurate representation of the workflow.

4.3. Additional Issues in Synthesis and Decomposition

We have addressed only the most basic issues in process synthesis and decomposition. However, there are other issues that arise in this area as well. One is the hierarchical integration of processes. That is, there are processes, such as budgeting, that take place at the strategic, tactical, and operational levels of orga-
nizations. The processes at each level are decomposed into subprocesses at the next lower level. To some extent the lower-level subprocesses should be independent so that people in the various functions (such as purchasing, manufacturing, and marketing) can focus on their areas of specialization. On the one hand, this leads to “stove-piping,” in which all interfunctional coordination takes place at the top. On the other hand, requiring complete intertask communication may lead to chaos. It may be desirable to modify the definition and theorems concerning independence presented here so as to provide additional guidance in this special case.

Processes are often modified after they have been implemented. For example, processes may be initially independent or redundant but are then modified so that they become dependent or redundant. If the modifications are minor (e.g., the addition of a single task or the creation of a single redundant element), then it may be possible to modify the results presented here so that they are still applicable to some degree. This would correspond to a graphical form of sensitivity analysis.

Process synthesis and decomposition also raise issues of timing, scheduling, capacity planning for resources, and redesign of coordination and control mechanisms. Since our representation model does not include attributes such as task durations and costs or resource capacities, such analyses are beyond the scope of this paper. Similarly, we also do not address issues such as the human resource, security, and legal implications of process change, as pointed out in §1. In addition, practical implementation of these results requires the development of a practical user interface that would shield the user from the computational details of the algorithms. The algorithms are necessary for these approaches, but it is not necessary for the user to be aware of them.

5. Conclusion
In this paper we have examined the synthesis and decomposition of workflows. We began by developing three important metagraph concepts: full connectivity, independence, and redundancy, and the derivative notion of well-structuredness. We then derived several results concerning the ways in which these concepts would be preserved by metagraph union and intersection operations. Finally, we examined the application of these concepts and results to workflow synthesis and decomposition.

The issues we examined were structural issues—that is, issues involving the relationships among tasks, and therefore among workflows, and the relationships among resources associated with these tasks. Metagraphs are especially useful in examining structural issues because notions of connectivity, as realized in the concepts of full connectivity, independence, and redundancy, are well modeled with metagraphs. Of course, they also exist in other graphical structures such as digraphs and hypergraphs, but metagraphs capture the directionality properties of digraphs, as well as the set-oriented properties of hypergraphs, in a very convenient fashion. Conditional metagraphs also allow us to treat edge attributes (in this case assumptions) as elements in the invertices of the edges. In addition, metagraphs also allow us to treat other edge attributes (in this case the resources) as edge labels and to model the relationships among the edge labels. Since this model (i.e., the RIM) is also a metagraph, we can use the same graphical structure both to examine relationships among tasks (and therefore workflows) and also among resources.

There are at least four productive areas for future research. First, we can extend our work on qualitative edge attributes to quantitative attributes. For example, we may attach edge labels representing time, cost, and performance (e.g., quality or reliability) and to calculate these attributes for an entire workflow. This would be the metagraph version of traditional PERT/CPM analyses of time, cost, and performance. The closure of the adjacency matrix \(A^*\) may be helpful in performing these calculations.

A second area is the hierarchical abstraction of workflows. The notion of hierarchical abstraction of metagraphs has been addressed in some of the literature referenced in Appendix A. In this case we would be interested in hierarchical abstraction of independent submetagraphs and the conditions under which certain properties of interest would persist or be introduced as one moves up the hierarchy.

A third area is the application of this work to three organizational contexts of growing importance today. The first is organizations that are linked electronically
to their trading partners. The second is organizations that allow their members to communicate and participate regardless of their geographical location by means of telework systems. The third is organizations in which members work in teams, many of which are temporary, to address specific current issues. It may be that the structure presented here can be applied, with modifications, to these organizations, as well.

A fourth area is the extension of this work to a formal theory of process-based organizations. Process-oriented frameworks for organizational design are beginning to appear in the literature (Malone et al. 1999, Mackenzie 2000), and the analytical structure presented here may be helpful in developing such structures and may suggest methods of integrating components of organizations and virtual organizations. Since the analytical framework presented here is directed to the synthesis and decomposition of both processes and workflows, a workflow being an enactment of a process, we may expect that it may be of help in the synthesis and decomposition of process-based organizations.

Finally, it may be possible to use the work presented and the results of future research to develop metagraph-based CASE tools for workflow analysis. Thus, metagraphs not only offer qualitative insights into workflow analysis, but may also offer a foundation for quantitative analyses, hierarchical abstraction, and computer-based tools. Yet another important area is the implementation of systems that support workflow and process integration. In this area, the potential use of tools such as XML, which has been used to implement Petri net models of processes (Lenz and Oberweis 2001) is particularly promising.

Acknowledgment
This work was supported by the Dean’s Fund for Faculty Research at the Owen Graduate School of Management, Vanderbilt University.

Appendix A. A Brief Review of Connectivity in Metagraphs
Given two elements \( a, b \in X \), a simple path from \( a \) to \( b \) is a sequence of edges \( h(a, b) = \langle e_1, \ldots, e_j \rangle \), such that \( a \in V_j \), \( b \in W_j \), and \( e_j \cap e_k \neq \emptyset \) for \( j = 1, \ldots, k-1 \). The co-input of \( a \) in \( h(a, b) \) is \( \bigcup_j V_j \setminus \bigcup_j W_j \). The co-output of \( b \) in \( h(a, b) \) is \( \bigcup_j W_j \setminus \bigcup_j V_j \). The set of all the other outputs produced by edges in the path is \( \text{co-output}(a, b) \) from an element \( a \) to itself.

The concept of a simple path is insufficient to describe connectivity in a metagraph. There are two reasons for this. First, since edge inverters and outvertices may contain multiple elements, it may be possible to connect a source element to a target element without any co-inputs by invoking parallel edges. Second, since metagraph edges connect sets of elements, it may also be useful to define connectivity between sets of elements, and simple paths are inadequate for this purpose.

To address this, the concept of a metapath is used. A metapath is a set (rather than a sequence) of edges that connects a set of source elements to a set of target elements. Intuitively, a metapath from a source set \( B \subseteq X \) to a target set \( C \subseteq X \) is a set of edges such that, starting with only the elements in \( B \), all the elements in \( C \) can be determined using these edges in some feasible (possibly parallel) order of instantiation. In other words a metapath \( M(B, C) \) is a set of edges \( E \subseteq E \) such that (1) each \( e \in E \) is on a simple path from some element in \( B \) to some element in \( C \), (2) \( (\bigcup_j V_j \setminus \bigcup_j W_j) \subseteq B \), and (3) \( C \subseteq \bigcup_j W_j \).

A metapath from a source \( B \) to a target \( C \) is edge dominant if the set of edges in the metapath is not strictly contained in the set of edges in any other metapath from \( B \) to \( C \). Edge dominance is not unique—that is, for some \( B \) and \( C \) there may be more than one edge-dominant metapath from \( B \) to \( C \).

Even though a metapath is defined as a set of edges, the sequential dependencies between its edges can be extracted and utilized, based on an algebraic representation of the metagraph. The structure of a metagraph can be represented by an adjacency matrix. Operations such as addition, multiplication, and transitive closure can be performed on this matrix. Analysis of the metagraph, including identification of paths and metapaths, can then be stated in terms of these operations and related procedures. Examples of analyses include identification of cycles, identification of edge-dominant metapaths, and identification of bridges (edges that are on all of the metapaths between a given source and target).

The adjacency matrix \( A \) of a metagraph is a square matrix with one row and one column for each element in the generating set \( X \). The \( ij \)th element of \( A \), denoted \( a_{ij} \), is a set of triples, one for each edge \( e \) connecting \( x_i \) to \( x_j \). Each triple is of the form \((C_l, C_{O_j}, e)\), in which \( C_l \) is the co-input of \( x_i \) in \( e \) and \( C_{O_j} \) is the co-output of \( x_j \) in \( e \). There is an algebra defined for metagraph adjacency matrices. Given adjacency matrices \( A_1 \) and \( A_2 \), defined for two metagraphs that have the same generating set, these matrices can be added and multiplied with the result in each case being another matrix over the same generating set (Basu and Blanning 1994b). Intuitively, \( A_1 + A_2 \) represents the adjacency matrix of the union of the two metagraphs, while \( A_1 \cdot A_2 \) represents all paths of length two, where the first edge is from the first metagraph and the second edge is from the second metagraph.

Using multiplication, we can also compute powers of an adjacency matrix. The \( n \)th power of \( A \) is denoted \( A^n \). The \( ij \)th element
of \( A' \), denoted \( a_{ij}' \), is a set of triples, one for each simple path \( h(x_i, x_j) \) of length \( n \) connecting \( x_i \) to \( x_j \). Each triple is of the form \( (C_{ij}, CO_{ij}, h) \), in which \( h \) denotes the sequence of edges comprising the path, \( C_{ij} \) is the co-input of \( x_i \) in \( h \) and \( CO_{ij} \) is the co-output of \( x_i \) in \( h \). The closure of \( A \), denoted \( A' = A + A^2 + \cdots \) (Basu and Blanning 1994b), represents all simple paths of any length in the metagraph. The \( ij \)th element of \( A' \), denoted \( a_{ij}' \), is a set of triples, one for each simple path \( h(x_i, x_j) \) of any length connecting \( x_i \) to \( x_j \). The multiplication operator allows any cycle to be traversed only once.

The incidence matrix \( G \) of a metagraph has one row for each element in the generating set and one column for each edge. The \( ij \)th component of \( G, g_{ij} \), is \( -1 \) if \( x_i \) is in the invertex of \( e_j \), it is \( +1 \) if \( x_i \) is in the outvertex of \( e_j \), and it is \( 0 \) otherwise.

Metagraph edges may be constrained by assumptions. An assumption is a proposition— that is, a statement that may be either true or false—which appears in the invertex of an edge and must be true for the edge to be used in a metapath. Propositions appear in the generating set along with elements representing other types of variables. A conditional metagraph is a form of \( S = \langle X, X', E \rangle \), in which \( X \) is a set of propositions and \( X' \) is a set of variables (i.e., the remaining elements). Note that a simple metagraph is a specialization of a conditional metagraph in which \( X' = \emptyset \).

The values of different propositions can be used to specify alternative contexts for a conditional metagraph. The context of a conditional metagraph \( S \) with respect to a set of true propositions \( P \) and false propositions \( Q \), denoted \( K(P, Q, S) \), is a conditional metagraph in which (1) any proposition in \( P \) is deleted and (2) any edge containing a proposition in \( Q \) is deleted. If any edge now has a null invertex or outvertex, then that edge is also deleted. The context operation is a useful abstraction on metagraphs, since it avoids the need to consider edges that cannot be used under the stated conditions.

Propositions are not the only attributes that can be attached to edges. Another type of attribute is a resource. Resources may be people, roles (i.e., classes of people, such as financial analysts), and physical resources (e.g., workstations in general or a specific workstation). A resource interaction metagraph, or RIM, is formed as follows. Each element in the RIM is a resource. Therefore, the invertices and outvertices of the edges in the RIM are sets of resources. Assume that an outvertex of one edge in the original metagraph has a non-null intersection with the invertex of another edge in the original metagraph, and that each of these two edges has resources associated with it. Then there is an edge in the RIM in which the resources of the first edge in the original metagraph are in the invertex and the resources of the second edge in the original metagraph are in the outvertex.

**Appendix B. Algorithm for Identifying an ISMG in a Metagraph**

The following algorithm determines whether a given submetagraph \( S(X', E) \) is independent within a given metagraph \( S(X, E) \) with adjacency matrix \( A \):

**Procedure Check-Independence \( (S', A) \)**

Let \( P' (PO') \) be the set of pure inputs (pure outputs) of \( S \) [generated using Procedure PIPO below]

\[
\text{For } i = 1, \ldots |X| \\
\text{For } j = 1, \ldots |X| \\
\text{If } [(x_i \in \{X \setminus P' \} \text{ and } x_j \notin X') \text{ or } (x_i \notin \{X \setminus PO' \} \text{ and } x_j \notin X')] \\
\text{Then } S' \text{ is not independent in } S; \text{STOP.} \\
\text{Next } j \\
\text{Next } i \\
\text{STOP.}
\]

The following algorithm identifies the sets \( PI' \) and \( PO' \) used in procedure Check-Independence:

**Procedure PIPO \( (S', A) \)**

Let \( PI' = PO' = \emptyset \);

For each \( x_i \in X' \)

\[
\text{If } a_{ii} = \emptyset \forall x_i \in X' \text{ Then } PO' = PO' \cup x_i; \\
\text{If } a_{ij} = \emptyset \forall x_i \in X' \text{ Then } PI' = PI' \cup x_i; \\
\text{Repeat; } \\
\text{Return } PI', PO'; \\
\text{STOP.}
\]

Note that these are polynomial procedures that are guaranteed to terminate.

**References**


---


---


---


---

---


