SIGNS OF IT SOLUTIONS FAILURE: REASONS AND A PROPOSED SOLUTION

MAJED ABUSAFIYA
NEW MEXICO TECH
majed@nmt.edu

SUBHASISH MAZUMDAR
NEW MEXICO TECH
mazumdar@nmt.edu

ABSTRACT

In spite of the huge advancement in information technologies and the commitment to adopt them, still large deal of the organization's information and business processes activities exist outside the scope of automated management. In this paper we will show why current IT solutions fail to completely cover information and business processes within the organization. We will propose new document based information and business process models that cover information and business processes within and outside the scope of current IT solutions. These models can be then the underlying model for a new IT solution that provides a wider scope of automated management than current IT solutions.

Keywords: Keywords: Information technology, Organization, Automated Management, Business process modeling, Document.

INTRODUCTION

Information Technology and Information Technology Solutions

Information technologies [8] include any software or hardware that supports the management of the two very important tightly associated components of any organization: *information* and *business processes*. An *information technology solution* is the adoption of available *information technologies* to provide better management of information and business processes. Information technologies are the tools with which we develop IT solutions. Two main categories of *information technologies* that are most adopted and effective in improving the performance of the organization: *database management systems* [7] in the context of information management and the less popu-

lar *workflow management systems* [13,10] in the context of business process management[1,9]. These information technologies have powerful *engines* that provide automated management of *well-defined strictly-structured templates* that follow what we call the *information technology underlying model*. To develop an IT solution based on these information technologies, such *templates* need to be designed so that our information (and business processes) can be *restructured* into these templates. Once information (and business processes) is placed in these templates, the automated management services provided by the information technology engine can be used. We will refer to this approach of automated management as the *restructuring-based automated management approach*.

For example, to utilize a relational database management information technology to develop a relational database IT solution, we design templates (i.e. relations or tables) that follow the underlying model (i.e. rela-

tional model) of this information technology into which we can restructure our information. In the context of business process management, to utilize information technology like WFMS we should be able to define business process templates within which we can map our business process instances to be managed by the workflow management engine.

The process of developing an IT solution based on these technologies is a typical software engineering process. We need to specify the organization's information and business process management needs using special modeling languages, for example UML [6,12] or ER-based [7]. Afterwards we develop *design models* that are the link between analysis and the adopted IT underlying model. For example, for ER-based analysis model and a target a relational database management information technology, our design model will represent the relational model for this ER-analysis model. In the *implementation* phase, we define the *templates* based on the design models using the definition services provides by the information technology. Finally the IT solution is deployed by mapping the information (or business process instances) into these templates. Afterwards, this information technology engine becomes our main interface to manage this information (or business processes). So, developing an IT solution is projecting the reality of the organization into strict well-defined well-structured templates (that follow the underlying model of) and managed by the engines of the adopted information technologies. The order of these steps depends on the development process adopted (e.g. waterfall, iterative). See Figure 1.
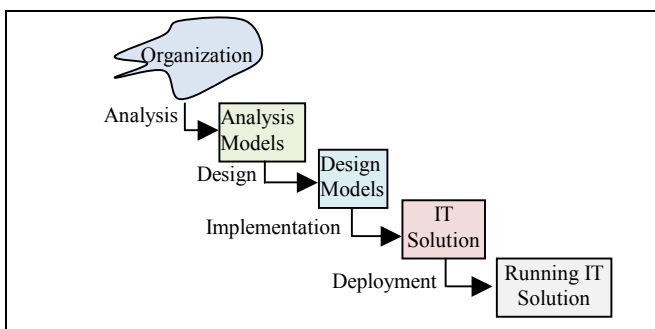

Figure 1: IT Solution Development as a Software Engineering Process

For example, a student record relation schema within relational database management system will define a template and the students' information will represent information to be restructured into this template. Figure 2

shows this idea where circles represent information (or business process activities) within the organization before deploying the IT solution and afterwards. To utilize these information technologies, we should be able to structure our information (or business processes) into strict templates. Note that in Figure 2 there are some information objects (or business processes activities) still outside the scope of these IT solutions (This will be discussed latter).
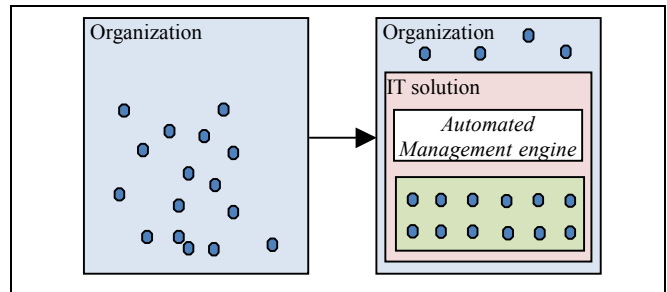

Figure 2: The restructuring Approach of Current IT solutions

Evolving Nature of Organizations

Organizations are dynamic entities. They show continuous unpredictable behavior. They continuously change their business rules, policies, and organizational structure. Organizations have to take corrective steps to deal with changes within and outside the organization and to achieve its strategic goals. Also, managerial activities like controlling, directing, coordinating, supervising, and reporting result in unpredictable actions. The effect of these changes is mainly seen in the behavior of business processes. New business processes may be created and existing ones may be eliminated. There can also be a change on the default behavior of the existing business processes resulting from a change in their definition. In addition, instances of business processes often show unpredictable behavior due to two main reasons: (1) some business processes are vague and hence their instances behavior is unpredictable. One example is business processes requiring human intelligence where there could be general rules but a detailed description of steps cannot be specified. Managerial processes, writing reports and decisions making fall in this category, (2) it is impossible to come up with a complete business process definition that considers every possible scenario that may happen. This includes errors in human handling, unexpected input, unexpected change in the organizational structure and the

need to deal with special cases. Business process instances deviate from their specified behavior to deal with these scenarios. In the rest of this paper, when we refer to *unpredictable behavior* of business processes we refer to all possibilities we mentioned above: change in the default behavior (i.e. change in business process definition), belonging to a vaguely or newly defined business processes, or because they deviate to deal with unexpected scenarios.

Failure of current IT solutions

A success factor of an IT solution is how wide the scope of information and business processes it covers. In reality, information and business processes are not completely covered by the current IT solutions mainly because:

Firstly, not all information (or business processes) is considered for mapping into the automated management of the IT solution. This is because (a) this information is considered less important for the main functioning of organization, or (b) there is a lot of overhead and cost involved in restructuring such information into templates that follows the underlying model of the target information technology (for example informational content in mail correspondences of the admission office).

Similarly, some business processes are considered less important and is not considered for automation. Moreover, some business processes are vague and hence such templates cannot be defined. Even for precisely defined business processes, frequent deviation from default behavior results in having them completely or partially outside the scope of automated management. In addition to that, some business process activities are manual and should stay outside the scope of automated management (e.g. receiving student application in a paper form).

Secondly, the organization's information and business process management requirements deviate from those managed by the adopted IT solution, however IT solutions, once deployed, are hard to modify: IT solutions are designed based on the current *well-defined* information and business process management needs. Moreover, IT solutions once deployed, they are hard to change. On the other hand, the organization's information and management requirements continuously deviate from the original requirements upon which the IT solutions were built. With time, the deployed IT solution will match less with the current information and business process management requirements of the organization. This change results in new information and business process activities emerging outside the scope of the deployed IT solutions. For example, suppose a registration system in some university that was developed based on the policy that a teaching assistant should pay his tuition fees before registration. Later a change in policy was adopted where it allowed payment through installments where a special form needs to be signed to authorize these installments. Such a change in policy resulted in a change in the registration process and information (a new form needs to be signed and processed).

To deal with these deviations and changes, an ad hoc quick solution outside the scope of IT solution is needed. This quick ad hoc solution is usually managed through *documents*. Documents are the cheap fast flexible ad hoc solution to handle a specific change or unpredictable deviation that cannot be handled by the deployed IT solution. Documents can easily be created with any content and structure, communicated and interchanged. They can be used to keep track of information and business process activities existing outside the scope of automated management. Information in these documents cannot just be merged into the existing IT solutions due to their unpredictable structure and content. So these documents become part of the organization information base *beside* information within the scope of IT solutions. They will also be the target of activities of deviating business processes instances and even part of default business process behavior when a business process definition itself changes or new business processes are defined. Also documents exist within the organization because they are the default information containers and part of this is usually neglected in the process of IT solution engineering. Moreover, some business process activities are document-based in nature and can only be done through documents, like receiving paper admission application, writing a report, replying to a received e-mail. Also coordination and communication activities are mainly done through documents like emails, memos, letters, etc. Thus, there will always be a minimum margin of documents within the organization. So, documents and document-based activities are the *main component* of information and business process activities existing outside the scope of the deployed IT solutions.

With the continuous evolution of the organization, the volume of documents and document based activities increases. It is known that documents are hard to manage (updating, querying about, storing and retrieving). Imagine, for example, the effort needed to query about information within paper admission applications or finding electronic version of a memo that was sent to the department last month regarding Alex's admission application. Also because of this continuous change, the information and business process requirements of the organization will continuously deviate from those upon which the IT solution was developed reaching to a point where an *IT solution reengineering* is needed to have better coverage

of the current information and business process management requirements and to reduce the volume of documents and document based activities (Figure 3).
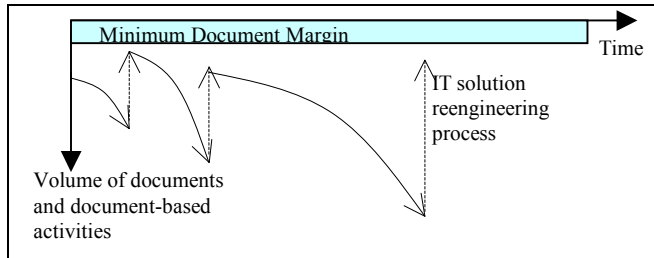

Figure 3: Document Volume Increases with Time and Triggers IT Solution Reengineering

Failure of the current IT solutions can be seen at the point of their deployment where they do not completely cover the whole information and business processes within the organization. Not all information and business processes within the organization can easily be mapped into models that follow the well defined strictly structured underlying models of adopted information technologies. Also with time, the organization's information and business process management requirements deviate from those on which the IT solution was originally designed, while the deployed IT solutions cannot be easily adjusted.

Also current IT solutions fail in providing automated management for an important component of the organization's information and business processes which is documents. Restructuring-based automated management approach works towards the elimination of documents and document-based activities by restructuring their informational content into strict templates. Meanwhile, documents form a very important component that will always be there. Failure of current IT solutions to cover documents means that the informational content of these documents and business process activities dealing with these documents will stay outside the scope of automated management.

Moreover, current IT solutions fail to keep track of the organization's unpredictable deviating behavior of its business processes. Deviating business processes do not comply with the strictly defined templates managed by the adopted information technology and hence should be managed partially or completely outside the scope of the IT solution. Another factor that makes keeping track of business process instances harder is the unpopularity of

business process management information technologies. In addition to their expensive cost of purchase, installation and deployment, the deviating nature of business processes makes such IT solutions less attractive. In reality, we find that IT solution engineers find it enough to develop information management solutions like databases and build application layers on the top of them, for example *two-tier* and *three-tier* architectures, instead of adopting a business process management based information technology (like WFMS). In developing these layers, only parts of business processes that affect the information within automated management are considered. For example, when a student is registered through a database management system, only the final step is captured excluding the prior activities, like communicating with the advisor.

If we have a closer look, we find that IT solutions underlying models are designed following the underlying models of the adopted information technologies. We have seen that these models are strictly structured, well-defined, hard to adjust and accepts only information and business process instances that follow the designed model. This results in having important portion of the organization's information and business process activities (that is, document and document based activities) to exist outside the scope of automated management.

SOLUTION

The main idea of the proposed solution is to design IT solutions with more flexible underlying model than the strict inflexible underlying models of the adopted information technologies. Since documents are important component of the organization information base that we cannot completely eliminate, we need an IT solution underlying model that accepts the existence of documents and the deviating business process behavior while keeping track of them. These flexible underlying models can be efficiently queried by adopting recent information technologies.

Information within the organization can be viewed to exist in two spaces: information within automated management (like databases) and information outside the scope of automated management which is in documents. These two spaces are tightly associated because (1) information in one space may depend on information in the other space. For example a paper document may be generated by printing the student database record. A student database record may be filled from a received paper admission application and (2) a business process instance may span the two spaces. For example, consider admission business process that starts with receiving a student admission application in paper form, and involves

creating a database record creation and sending a memo to the department. Any IT solution should integrate the two spaces to have a better management of the information base of the organization.

We propose a *document-based information model* for the organization information base. This model is document based because it views information in *both spaces* as a set of documents and it captures this information base by maintaining an up-to-date document image of these documents. This model is a synchronous model of the information base of the organization in the sense that a change on the organization information base should result in a similar change on this information model.

Based on this information model, we propose a business process definition model. This model defines business process activities as sequences of document operations (including views over information within the deployed IT solutions). This view of business processes is justified due to the tight association between information and business processes. Almost all activities are associated with some information access. Even those activities that are not associated with information access can be made so by logging. Based on this business process model, we define a *business process instances model*. This model captures the up-to-date state of a business process instance by keeping track of the information access operations happened as part of this instance. Since the proposed information model is a synchronous document-based image of the information base of the organization, we can keep track of the business process instances by keeping track of their document operations happens on the proposed information model.

We define two symmetric worlds (models) (1) the *reality models* corresponding to the real information base (viewed as a set of documents) and real business processes instances and (2) the *capture models* which will be the up-to-date image of the reality models. For the capture models to be valid, they should always be *synchronized* with the reality model. This means that a change on the states of documents and business process instances in reality should be captured by a corresponding change on the capture models.

The reality models are defined to (1) refer to real documents and business process instances of the organization and (2) show the correspondence between the reality and the capture models. This will help in checking the correctness of the capture models.

We propose the capture models (D and P) that keep track of the reality models (D^* and P^*). D^* will represent the real information base in the two information spaces viewed as a set documents. P^* is the reality model of business processes instances of the organization. For example, in Figure 4, we can view a thread of document operations happened as part of some admission business process instance. In the reality side, there is a sequence of D^* document operations includes receiving a paper-form admission application, creating student record through a database view, creating a credit transfer record for the student through a database view, sending an e-mail to the department and finally the admission office creates a record for accepted transfer credits through a database view. This sequence of document operations composes the so far finished part of p^* in P^*. In the capture model side, there is a corresponding document *document(d*)* in D for the document d^* in D^* accessed by p^*. *document(d*)* is an up-to-date image of d^*. The sequence of document operations representing p^* is captured by *process(p*)* in P.
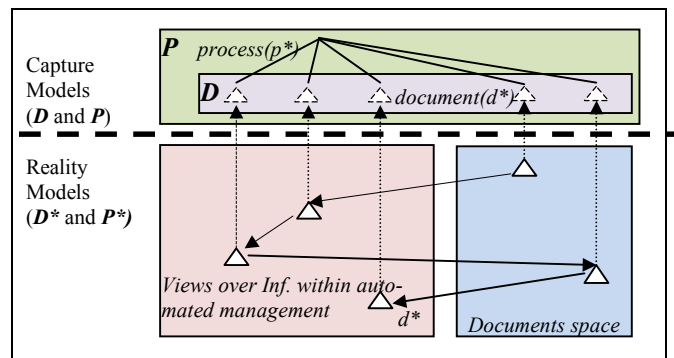

Figure 4: Relation between Reality and Capture Models

The capture models D and P will be the *underlying models* for an IT solution that covers information and business processes within and outside the scope of the deployed IT solutions. This IT solution should utilize recent technologies to automate the capture of changes on D^* (and changes on P^*) and automate the update of D and P so that both models represents an up-to-date capture of D^* and P^*. It should also provide the automated querying of D and P models.

In the rest of the paper we will present the reality models D^* and P^*. We will then present the capture models D and P. Afterwards; we will give the blueprints of an IT solution based on D and P as the underlying models. We will present the advantages of the proposed model and end the paper with concluding remarks.

REALITY MODELS

The definition of Document

A *document* is a presentable information container that has certain attributes that define its state and on which certain operations can be applied that change this state. Documents are presentable objects to the human where he can read and comprehend. A *document state* is defined by the values of its attributes. A document attribute is any information about the document (e.g. type, informational content, location, owner, last accessed). Also associations with other documents and components of the organization are attributes. For example, *copy-of* could be an attribute that associates a document with its original document. The attribute *owned-by* may associate a document with its owner. Documents are objects on which we can apply operations (e.g. annotate, view, update, print, mail, fax, etc). A document operation may generate new document (*copy* for example). We will use object oriented programming language notation to refer to a document attribute or operation. For example, *d.physicalLocation* represents the physical location attribute of document *d*. Also, *d.print()* refers to the print operation.

*D** Model

D* represents the actual information base of the organization viewed as a set of documents and the target of business process instances effect. *D** includes (1) *paper documents*: In spite of the advances in paper-less office and information technologies, paper documents still exist in every office within the organization. It is a frequent activity to print electronic documents for annotation, routing, personal use and signature. Sometimes only the original paper document is considered to be authentic. Paper documents are important for communication within/outside the organization, (2) *synthesized documents*: We will consider views over information-based IT solutions (like DBMS) to be documents as well, (3) *standalone documents*: these are electronic documents that are not synthesized documents (e.g. text files, WinWord files, spread sheets, emails, non-synthesized web pages, PDF files, audio, etc). Standalone documents are part of the document space and are not part of the IT solutions information space.

In this paper, we will focus on *D** documents that are accessed by some business process instance whether they are paper, synthesized or standalone. This is justified, because a document is of value if it is involved in some business process instance. Note that for synthesized documents, we may have infinite number of views over a database. We only consider those views that are accessed by some business process instance. In the rest of this paper, when we refer to a document, this includes views over information within IT solutions accessed by business processes except explicitly specified otherwise.

The reasons for adopting document view of the organization information base are: (1) it is a natural way of viewing information. A human views information in the form of documents. A business process instance activities deal with information in the form of documents, (2) documents are an important component of the organization information base, (3) document view is a homogeneous view where we can even consider views over information within automated management as documents, (4) informational content of documents can easily be transformed into an electronic semi-structured constructs which is supported by advanced information technologies [11].

Any information about *d** represents an attribute of *d**. This information can be *explicitly* represented by an attribute name and a value from some domain. For example, the information: "the owner of this document is *Alex*", can be represented by an attribute called *owner* and a corresponding value *Alex* whose domain is the set of participants of the organization. The most important attribute of the document is its informational content. When we refer to a *d** document operation, we actually refer to an actual operation applicable on *d** due to some business process. A document operation could be for example annotating a paper document, creating a new document, sending an e-mail or changing a database record.

Documents in *D** that have similar informational content and behave in similar manner due to business processes are assumed to be instances of some *document type*. A document type is a template that defines a general structure of the documents with similar informational content and operations (e.g. student applications). These types of *D** are not explicitly defined. A document *d** in *D** has a type and we will refer to that type as *type(d*)*.

*P** Model

P* is the set of actual business process instances of the organization. A business process instance *p** in *P** is a conceptual entity corresponds to the thread of activities of an enactment of some business process. Instances in *P** are tightly associated with *D**. A *p** in *P** affects *D** by applying operations on its documents. Although uncommon, there could be some activities without an access to *D** (*make a phone call* for example), such activities could be made so by logging. That is, requiring the participant carrying out that activity to log that he did that

activity on some document in *D**. By logging we can make every activity of *p** to have an effect on *D**.

Based on *D** and *P** models we can rephrase the problems with current IT solutions. IT solutions fail to cover the whole space *D** or *P**. There are some documents within *D** that are outside the scope of these IT solutions. Also there are some business process instances of *P** that are completely or partially outside the scope of the deployed IT solutions. The evolution of the organization and the unpredictable behavior in *P** results in adding new documents to *D** outside the scope of automated management and instances in *P** completely or partially outside the scope of automated management.
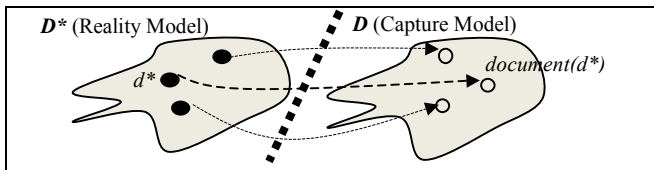
CAPTURE MODELS

D Model

D* (Reality Model)   *D* (Capture Model)

d*   *document(d*)*

Figure 5: The Capture of *D** Through *D*

D is the capture of the reality *D**. It is an up-to-date image of *D**. *D* maintains for *d** in *D** a corresponding document *document(d*)* that explicitly captures the up-to-date state of *d** (Figure 5).

document(d*) is a document that represents an explicit representation of the state of *d**. Information about *d** (or attributes of *d**) are explicitly represented by the corresponding attributes of *document(d*)*. This explicit representation of the state of *d** through *document(d*)* makes this information about *d** easier to map to (and hence be managed by) an automated management solution.

One flexible way to represent an attribute value is through semi-structured data [2]. An attribute value can be viewed as an information object which is composed from simpler information objects. Simplest information object could be a piece of text or a reference to another information object. For example, let *d** be a memo received by the admission office from the department to accept Alex. Assume that the attributes of *d** that we need to keep track of are the informational content and last access. This information can be captured by *document(d*)*

as shown in Figure 6. Note that *d** may have many other attributes but in *document(d*)* we only capture those attributes needed for information and business process management and may be the target for querying. How to represent an attribute value may vary. For example, in the capture of physical location, we can represent it by the department or by building. We will assume every attribute of *d** captured by *document(d*)* is also an attribute of *document(d*)*. For example *d*.informationalConent* is captured by *document(d*).informationalContent*.

From: CS Department
To: Admission Office
Subject: Alex Application
 	
The department decided to accept Alex. Please be informed.
 	
Thanks,
	<memo>
	 <informationalContent>
	  <from>
	   CS Department
	  </from>
	  <to> *A*dmission Office *</to>*
	  <subject>
	   Alex application
	  </subject>
	  <body>
	   admit Alex
	  </body>
	 </informationalContent>
	 <physicalLocation>
	  Admission Office
	 </physicalLocation>
	</memo>

Figure 6: *d** and *document(d*)*

document(d*) has the same document operations applicable on *d** with a similar effect. If an operation *o** is applicable (due to some business process) on *d**, a corresponding operation *o* should be defined on *document(d*)* such that it changes the attributes of *document(d*)* exactly as *o** changes the corresponding attributes of *d**. This is needed to keep *document(d*)* as an up-to-date capture of *d**.

Defining *document(d*)* for *d**: To support the automated construction of *document(d*)*, we define document types for *D* documents. Let *d** be an instance of *type(d*)*, define *type(document(d*))* as follows:
Define the attributes of type(document(d*)): Identify important information (attributes) about documents of *type(d*)* that we need to keep track of. To define an attribute we need to define an *attribute name* and define its *type*. Defining an *attribute type* means defining the informational structure of information objects that can represent a value for that attribute. Since we adopted the *semi-structured* data to represent information, we can define an attribute type by defining the structure of semi-

structured data that can represent a value for that attribute. This can be defined by defining the semi-structured data hierarchical structure relating composed elements with composing elements. The goal of defining structures for attribute values is to make these values subject to automated management (update, query, access, etc). For example, assume it is important to keep track of the last access information about documents of *type(d*)* which includes *date*, *time* and the person who did that operation. Assume that the last access of *d** was at 5:30PM on May 10th by George. This information (attribute value) can be captured by semi-structured data shown in Figure 7a. We can then define an attribute whose name is *lastAccess*. We can define a type for this attribute by defining the hierarchal structure of semi-structured data that may represent a value of that attribute (Figure 7b).


```
<lastAccess>                        lastAccess
    <time> 5:30PM </time>
    <date> May 10 </date>      time   date      by
    <by> George </by>
</lastAccess>                    <text> <text>  <text>

(a) lastAccess attribute value of d*    (b) attribute type
```
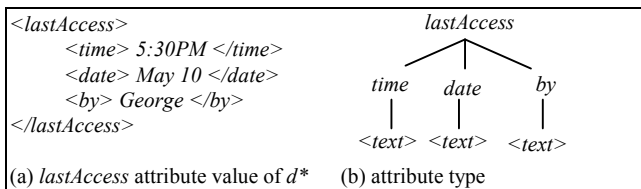
Figure 7: Attribute Value and Attribute Type

A special attention should be given to the definition of the informational content attribute of *type(document(d*))*. We need to define informational structure for the informational content attribute of *document(d*)* into which we can map the informational content of *d**. It is always better to map the whole informational content of *d** but this could be complex and involve a lot of overhead. If this mapping can be done in automated way (without human intervention, like synthesized documents) then a complete capture of the informational content of *d** into *document(d*)* can be chosen, otherwise only important informational content of *d** need to considered for mapping to reduce overhead. This is also justified because not all the informational content of *d** needs to keep track of. Informational content to consider for *document(d*)* is information required for the definition of document operations (and hence business processes, e.g. GPA for admission business process) or information that may be the target of querying for any purpose.

In addition to the definition of the *attribute type*, we need to define how an attribute value of *d** can be transformed to the corresponding attribute value of *document(d*)* matching the defined *attribute type*. That is, a *transformation definition* is needed. It is a transformation

because some processing may be needed to transform an attribute value of *d** to the corresponding attribute value of *type(document(d*))*. For example, not all information known about a given attribute of *d** need to be transformed (e.g. it may be enough to maintain *date* only for the *lastAccess* attribute). Also sometimes we may need to reformat the information known about *d** attribute to create the corresponding attribute value for *document(d*)*. For example, we may transform the statement of purpose of the admission application *d** into few words summarizing areas of interest for the applicant in *document(d*)* informational content.

To support document retrieval based querying of *d*'s* informational content, an *electronic copy* of *d** can be maintained as an attribute of *document(d*)* (e.g. scanned admission application, a copy of the sent memo, the student record in the database). Another important attribute needs to be defined for *type(document(d*))* is the *log* that maintains the history of document operations applied on *document(d*)*. The *log* attribute maintains a list of *log records*. When a document operation is applied on *document(d*)*, due to a document operation on *d**, a log record is created and added to the log list. A log record keeps track of information about the document operation (like operation name, parameters, time, effect, etc).

Define Document Operations of type(d*): If an operation *o** is applied on documents of *type(d*)* as part of a business process, define a corresponding operation *o* for *type(document(d*))*. *o* is defined such that it changes the attributes of *document(d*)* exactly as *o** affects the corresponding attributes of *d**.

For example, let *takeAdmissionDecision* be an operation that is applied on admission application document as part of admission business process. This operation, if applied on a document *d**, will update the informational content of *d** by annotating the word "accept" (if GPA>3.0) or "reject". This means that *takeAdmissionDecision* should also be an operation for *type(document(d*))* whose affect is to update the informational content and *any other affected attributes* (for example, last access, log, copy of *d**, etc) exactly as the corresponding *takeAdmissionDecision* operation affects the attributes of *d**. Note that *GPA* and *decision* are accessed as part of this operation, so they should be defined as information objects within the informational content structure of *type(document(d*))*. This is needed so that the operation can be defined. For example, the effect of *takeAdmissionDecision* on the informational content can be defined as follows (assuming that *getValue* and *setValue* are primitive operations that are already defined):

gpa=getValue(admApplication/gpa)

```
    if (gpa>3.0)
        setValue(admApplication/decision,"accept")
    else
        setValue(admApplication/decision,"reject")
```

Note that a composing informational object is referred to by a path within the hierarchy of the information object representing the attribute value.

Extending Other Types: We can ease the creation of new document types by *extending* an already defined type (similar to the concept of inheritance of object-orientation paradigm [6,12]). The new document type inherits the definitions of the attributes and operations of the extended document type. We can then define new attributes and operations or update the definitions of the inherited ones. This feature is very helpful to cope with the continuous document type generation within the organization. Newly generated types may be a slight incremental change of existing document types. Tightening informational content of a given generic document type to generate more specific type can easily be done through extension. Consider for example *Memo,* a generic document type, may be extended to generate *RequestDepartmentInputMemo* which is the type of memos that are sent to the department to request input about an appealing student.

Methodology to Build *document(d*)* for *d** : A methodology for constructing a document(d*) for d* is:

---
1 Bind document(d*) to a matching document type
2 If a matching type is not found
3   Create a new type for document(d*)
4   Bind document(d*) to that type
4 Instantiate document(d*) from d* and its type

Binding *document(d*)* to a document type is the first step towards the construction of *document(d*)*. In case such a matching type is not found, define a new type and then bind the *document(d*)* to that type. By binding *document(d*)* to a certain type, *document(d*)* will inherit all the attributes and operations defined in that type. Finally, *instantiation* is needed where attribute values should be extracted from *d** and transformed to the corresponding attribute values of *document(d*)*(could be manual or automated).

Synchronizing *D* with *D**

For *D* to be a capture of *D**, *D* should be in synchronization with *D**, meaning that *document(d*)* should maintain correct and up-to-date information about the state of *d**. This can be achieved by maintaining the following rule: if a document operation was applied on *d**, this operation should be captured and a corresponding

document operation should be applied on *document(d*)* such that *document(d*)* stays up-to-date capture of *d**.

The Correctness of *D* in Capturing *D**: The synchronization between *document(d*)* and *d** might be broken because an operation happened on *d** was not captured, synchronizing *document(d*)* with *d** was delayed, or error happened in updating *document(d*)*. We can check for inconsistencies between *d** and *document(d*)* by comparing them (could be manually). One way to find inconsistencies between *d** and *document(d*)* is to reconstruct *document(d*)* and compare the new *document(d*)* with the existing *document(d*)*. Remember that by constructing *document(d*)* from *d**, we immediately put *document(d*)* in synchronization with *d**. If there is a difference, it can be fixed by replacing the old *document(d*)* with the new *document(d*)*. That is, reconstruction is a straight forward solution to return to synchronization state. In addition to discovering discrepancies, another advantage of comparing the newly constructed *document(d*)* with already defined *document(d*)* is discovering changes happened on *d** that were not captured.

In Figure 8, assume a document operation was applied on *d** generating *d*'*. There are two ways in having *document(d*)'* up-to-date with *d*'* (1) we know which operation that happened on *d** so we apply the corresponding operation on *document(d*)* and hence we have *d*'* and *document(d*)'* in synchronization. (2) For some reason we could not capture the operation happened on *d** giving *d*'*, so *document(d*)* is not updated accordingly and hence it becomes out of synchronization with *d*'*. In this case we can return to synchronization by reconstructing *document(d*)* from *d*'* .
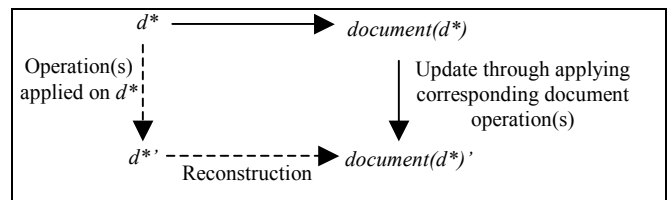

Figure 8: Putting *document(d*)* and *d** into Synchronization

P Model

D*, the information base of the organization, changes due to the execution of business process instances in *P**. *P** will be captured by *P*. If *p** is a business process instance in *P**, then *process(p*)* is a corresponding

business process instance is P (Figure 9). In this section, we will first present a business process definition model where a business process is defined as a sequence of document operations. We will then present the P model.
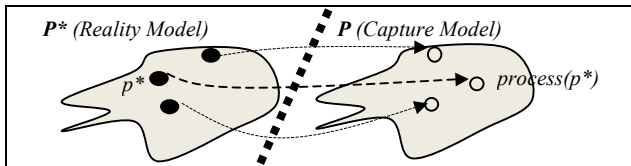

Figure 9: The Capture of $P*$ through P

Business Process Definition Model: A *business process definition* specifies the default behavior of instances of this business process. By defining a business process, we specify activities and logic flow among them. There have been many modeling approaches towards the business process definition. A typical business process definition model is a graph of activities with control flow among them.

One main problem of the current business process definition models is the neglect of the detailed information infrastructure on which the business processes are supposed to run and how they interact with it. There is a tendency to describe business processes abstracted from the information infrastructure context on which these business processes interact with. The goal of this abstraction is to have the definition independent from the information infrastructure context. This is one of the main reasons why business processes definition models are vague. Actually a business process cannot be defined precisely without defining its information infrastructure context. For example, given a transfer student admission application business process where the student credit transfer to be done by the registrar and a scholarship to be assigned by the financial aid office. For paper based information infrastructure the admission coordinator receives the transfer student application documents, makes two copies and sends them to the registrar and the financial aid office. She waits for answers from the two offices to inform the student. On the other hand suppose a shared database infrastructure, the admission office receives the application, fills the application into a shared database that makes this information available to the registrar and financial aid office. The admission office gets decisions made by checking the database. Note this business process cannot just be described precisely without defining the information infrastructure context. The abstraction of the informa-

tion infrastructure context makes the business process definition vague and hard to be precisely described.

We will go one step further in defining business processes by defining how they interact with the organization's information base. Since the information base of the organization is $D*$, this interaction can be viewed as a sequence of document operations over $D*$. Knowing that D is an explicit synchronous well-defined capture of $D*$, we will define business process interaction with $D*$ through defining how this interaction affects D. That is, we define a business process by defining its effect on D.

Based on the document view of the information base, a business process definition can be viewed as an object-oriented program whose objects are documents. In the object-oriented program definition we define classes, variables corresponding to objects of these classes and a sequence of calls of operations on these variables. This sequence of variable operation calls defines a sequence of transformations on the states of these variables until the program goal is achieved. A program instance is an enactment of the program definition, where variables are instantiated as objects in the memory with completely defined states, the sequence of operation calls are executed where the states of objects are updated and the goal of the program is achieved.

We can define a business process by defining an object oriented program whose classes are document types and whose variables (bound to document types) are document instances. A business process instance corresponds to the instance of a running object oriented program whose objects are document instances. A typical document operation of a business process will be modeled by a document operation call applied to some variable bound to a document type. For example, an operation like annotating a student admission application with "accept" can be represented by *application-Form.annotate("accept")*. This call of an operation on the variable *applicationForm* which is bound to the document type *ApplicationForm* models a step in the admission business process where the application form should be annotated with accept. The program may also use control flow constructs, (e.g. if-then, loops), concurrency structures (e.g. fork, join) and data structures (e.g. arrays, lists) to define the sequence of operation calls of the business process.

Methodology to Define a Business Process: To model a business process, a modeling language and a methodology are needed. Our modeling language constructs can be the typical constructs used in a typical object oriented programming language. For the methodology, it is very similar to object oriented analysis and de-

sign methodologies [6,12] where we define classes of objects (by defining attributes and operations applicable on objects of these classes). We also need to define how objects of these classes should interact with each other to achieve a desired goal. The difference is that (1) classes we define are document types and not reality classes (like student, course, etc), (2) our aim is not to define a software system to replace existing system or to reengineer the business process. Instead, our aim is to capture how business processes (viewed as sequences of document operations) should be carried out.

To define a business process, we start by defining document types involved in the business process (based on the proposed document type definition methodology). We need to study different scenarios of sequences of document operations happen due to the defined business process. To describe a scenario, we assume instances of documents (variables bound to document types) from the business process document type space and describe the sequence of document operations that should happen. Different scenarios correspond to the behavior of different instances of the business process. The main reason for having different scenarios is that the behavior of the business processes depends on the state of the documents involved (mainly the informational content) which is specified by the *business rules*. During this process we may continuously refine the document types to make sure that all needed operations are already defined. From these scenarios we can define a generic sequence that may include constructs like loops, if statements, etc. This generic sequence is very similar to an object-oriented program. For those activities that are not associated with a document operation, they can be made so by assuming that they are logged once they happen.

Business Process Instances Model *P:* Based on the proposed business process definition model, we propose business process instances model. The main idea of the proposed business process instances model is to keep track of the document operations happened as part of these instances. *P**, the set of actual business processes instances, will be captured by *P*. For every *p** in *P** there is a corresponding document *process(p*)* in *P*. *process(p*)* keeps track of the sequence of document operations happened on *D* due to the corresponding document operations happened on *D** as part of *p** (in addition to any important attributes of *p**). This is valid because *D* is a synchronous image for *D**.

process(p*) is represented as a document because: (1) it can be defined on the top of *D* which is a natural capture of the state of the business process according to the proposed process model, (2) it is flexible struc-

ture that can capture deviating *p**, (3) It can be easily updated to match *p** and (4) it can be managed in automated way (e.g. update, query). *process(p*)* is a document that is composed from *document(d*)* documents where *d** was accessed by *p**. Also *process(p*)* maintains the sequence of document operations happened on *D* due to *p** effect on *D**. This can be done by maintaining a log attribute for *process(d*)* (explained in detail in next section).

Synchronization between *P** and *P*: *D** and *D* are continuously changing due to *P**. For *P* to be a valid capture of *P**, *P* should be in synchronization with *P**. Part of the synchronization between *P** and *P* is the synchronization between *D** and *D*. That is, for a document operation happened as part of *p** on *d** a corresponding operation should happen on *document(d*)* to be up-to-date and then *process(p*)* need to be updated. *process(p*)* can be updated by (1) linking *document(d*)* to *process(p*)* if not already connected to *process(p*)*, (2) updating the *log* of the *process(p*)* to record that this operation happened as part of *process(p*)*.

For example, given a business process instance *p** and as part of *p**, the following sequence of document operations $d_1^*.o_1$, $d_2^*.o_2$, $d_3^*.o3$ happened. To keep *D* synchronized with *D**, these operations should be captured and corresponding operations should be applied on *document(d_1*)*, *document(d_2*)*, and *document(d_3*)*. *process(p*)* should be updated as shown Figure 10.
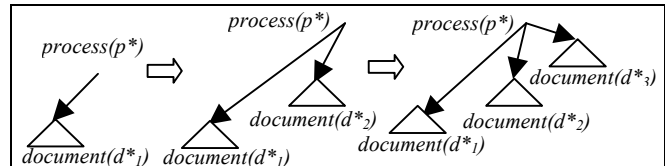

Figure 10: Evolution of *process(p*)* due to Document Operations

Also as a result of capturing these operations, a log record will be added to the corresponding *log* attributes for the corresponding document operation. The *log* attribute of *process(p*)* should be updated as well to keep track of this sequence of document operations by maintaining pointer to these log records (Figure 11). The main requirement to make this synchronization possible is to be able to bind a document operation on *d** to the right business process instance *p** (and hence a document operation on *document(d*)* to *process(p*)*).
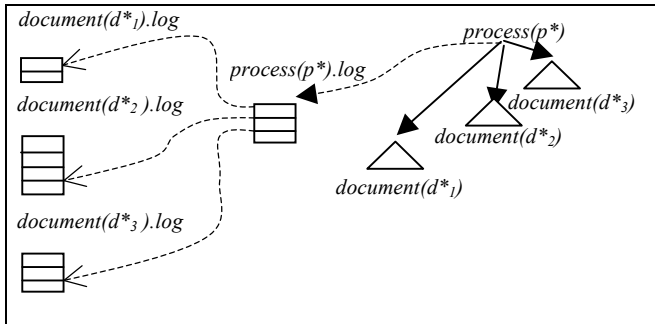
Figure 11: Updating *log* Attribute of *process(p*)*

The Correctness of *P* in Capturing *P*: *process(p*)* could be out of synchronization with *p** due to two main reasons: (1) failure to bind a document operation on *d** to the right *p** and hence operation on *document(d*)* to the wrong instance in *P*. This should be avoided, and (2) a document operation on *d** happened as part of *p** was not captured at all. For example, given the activity "*sign application*" that was not captured, then *process(p*)* will not match *p**. This even includes logging document operations through which we make non-document-based activities to have a document effect. For example, given an activity like "*make a phone call*", to make it document based, the participant is required log it once done. If the participant forgot to do so, *process(p*)* will not be in a matching state as *p**. The difference between the two examples is that for the sign application example, we have a document based discrepancy, that is *process(p*)* does not match *p** because a discrepancy happened between *d** and *document(d*)* due to an uncaptured operation that was part of *p**. We can fix this discrepancies by comparing the document space of *p** and *process(p*)*. If not synchronized, we can update the unmatching *document(d*)* and *process(p*)* to match *d** and *p**. For the phone call example, we don't have a reference document from which we can discover discrepancies and fix them. In this case, we should follow different approaches, for example asking the participant and taking what he says by faith. We can also adopt some technologies that may automate logging non-document-based activities to avoid errors or to discover discrepancies between *p** and *process(p*)*, for example, we can return to the phone logs.

BLUEPRINTS OF THE PROPOSED IT SOLUTION

D and ***P*** models can be the underlying models for a new IT solution that has a wider coverage of information and business process instances than that of the current IT solutions. Based on the flexible homogenous document-based view of information within the organization, we propose *document-based automated management approach*. In this approach we view information within the organization as a set of documents and design an IT solution that manages this view. This solution does not replace existing IT solutions within the organization. Instead, it will reuse them to cover a wider range of information and business processes instances. The main requirements of this document-based automated management IT solution is to provide automation needed to maintain ***D*** and ***P*** models up-to-date and to efficiently make information maintained in these underlying models efficiently queriable.

Note that in the restructuring based automated management approach, only the most important easy to restructure information is restructured into strictly structured templates following the underlying model of the adopted information technologies. It will result in the two spaces in Figure 4. This information becomes then the main source of information and the target of business processes effect. Information that was not considered for re-structuring (or was generated to deal with the change) will stay in its default containers (that is, paper or stand-alone documents) composing the document space. Note that the identity of documents considered for restructuring is not maintained; only their informational content was considered. In the context of business process automated management, not all business processes are covered. Only business processes that can be modeled following the underlying models of the adopted business process management information technology are covered. Business processes that are not well-defined will stay outside the scope of these solutions in addition to un-automatable and deviating parts of business process instances.

On the other hand, in the *document-based automation approach*, there is no restructuring of information to eliminate the document space. Information within the organization is viewed as a set of documents (including information within the deployed IT solutions). We keep track of documents state and not only their informational content. The states of documents are explicitly maintained (Figure 12). From business process point of view, this approach will also keep track of business process instances spanning the two spaces.

Each approach has its own advantages and disadvantages. Restructuring approach provides better automated management on the account of flexibility, scope of coverage, and preserving document identity. Documents considered for restructuring are neglected or even eliminated while their informational content is maintained within strictly structured templates efficiently managed by the adopted information technologies engines. Once these templates are built they only accept information that can be restructured into these templates. They are hard to fix and change once deployed. For the document-based automation, it has the advantage of flexibility on the account of better automation. It accepts documents and document-based activities. It keeps track of documents and business processes instances and provides automated querying of this information.

(a) restructuring-based approach   (b) document-based approach
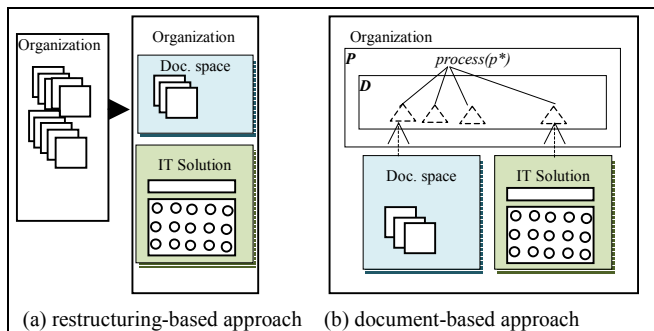
Figure 12: Restructuring-based and Document-based Automated Management Approaches

To have a better information and business process management through the organization as a whole, both approaches are needed. Although restructuring approach has better automated management it cannot cover the whole information and business processes within the organization. Document-based approach keeps track of the documents and business processes instances in both spaces. This information about the states of business processes and documents can be kept track of and queried in automated way using recent information technologies. The document-based automation is not an alternative of restructuring approach. Instead, it is an extension to it.

In the following we will present the blueprints of the proposed IT solution. This IT solution should provide automation needed to define, maintain and query *D* and *P* models. It is composed from three main components: *definition, synchronization and querying components*.

Definition Component

The IT solution should provide automated definition capabilities that allow defining *document(d*)* for a given *d** based the proposed methodology with minimum human intervention. This requires detecting the creation of *d**, binding to a document type (defining a new type if a matching type was not found) and instantiating *document(d*)*. If any of these steps can be automated, then the definition component should be designed to provide this automation. If any information needed to construct *document(d*)* cannot be gotten in automated way, then human intervention is needed where the definition component should provide user interfaces to get this information from the user. It should also provide needed interfaces to define business processes. It should maintain consistencies between the business process definitions and document type definitions (for example a document type referenced in a business process definition should be already defined).

Synchronization Component

Synchronization component is needed to maintain the *D* and *P* as up-to-date capture of *D** and *P**. This synchronization should be automated as far as possible to reduce overhead and errors. The IT solution should be supported by recent technologies (software or hardware) that automate the capture of document operations on *D**. Once a document operation *d** happens as part of *p**, this operation should be captured, *document(d*)* should be identified and a corresponding operation should be applied on *document(d*)*. It should also bind the captured document operation to the right business process instance *process(p*)* in *P*. Biding can be done by utilizing any information available to narrow the set of business process instances to which the captured document operation should belong. If the available information was not enough to bind the document operation, the synchronization component should provide interfaces needed for the human to refine the binding process.

Querying Component

The querying component should provide automation needed to query *D* and *P*. One natural representation of documents in *D* and *P* is using XML which is supported by recent automated management technologies [11]. The main advantage of XML as a representation language for *D* and *P* documents is that the hierarchical structure adopted for attribute values matches the structure of XML documents.

ADVANTAGES OF THE PROPOSED IT SOLUTION

Advantages of the proposed document-based IT solution include: (1) *Better real-time capture of the state of the organization*: It provides an up-to-date capture of information and business process instance states. This includes non-synthesized documents and ill-behaving business process instances. Since automated querying is provided, it makes this information easily available to whoever needs it. For participants, it means better performance and for managers better monitoring and decision making. (2) *Implementability and feasibility*: The IT solution built upon this model is incrementally implementable and does not require radical reengineering of IT solutions running within the organization; instead it reuses them. Initialization costs include investing resources needed to capture document operations within the organization. This includes for example using standalone document development environment, implementing triggers in available databases, providing paper document management infrastructure [3,4,5] (e.g. sensors, reading and scanning machines, etc) and automated logging system. It also requires integration with the deployed IT solutions and minor re-engineering of business processes to keep the captured model up to date (for example, constructing *document(d*)* for *d** and logging non-document based activities). Recent information technologies can be used to manage the underlying models. (3) *A new modeling approach for the organization's information base and business processes*. It is more comprehensive, homogenous, flexible, captures types and instances of both documents and business processes. It matches the human and business process view of information. Modeling a business process through its informational effect has many advantages: (a) *A precise and formal business process definition modeling approach*. (b) *Capturing business process instances*: The model keeps track of business process instances even if they are deviating or vaguely defined by capturing their informational effect within and outside the scope of automated management. (5) *It integrates business process and information dimensions of the organization*. We can find what effect a business process instance made on the information base of the organization. We can also associate a given informational access or change to its business process instance. This integration makes the information base of the organization to look like a virtual database and business processes to be as transactions on this database. This integrated view also can be used as a basis for the enterprise modeling and integration. (6) *Extended scope of automation to include documents*: it has many advan-tages: (a) automating business process activities involves documents. Extraction of information needed is faster. For example, if we maintain automated information about student applications documents within a file cabinet, we can check if there is an application for *Alex* by an automated querying without a manual search. (b) *Better control of business process involving documents:* We can enforce constraints specified by business rules through enforcing constraints on documents. For example, the admission application *d** should be rejected if GPA is less than 3.0. These business rules can be checked automatically when *document(d*)* is updated. We can discover *undesirable behavior and violations* (for example a paper document exists in wrong place, student with GPA less than 3.0 was admitted). We can also support *alerts* (For example, a software that reminds that admission coordinator that a given paper application is about to reach its processing deadline). (c) It also gives documents properties of information within automated management: like control, recovery, log, and automated queriability.

CONCLUSIONS

In this paper we pointed out the reasons of failure of current IT solutions to completely cover information and business processes within the organization. We saw that designing an IT solution requires a design of underlying model into which we can map the information and business processes of the organization. Current IT solutions are designed based on restructuring approach were we design IT solution underlying models that follow the underlying model of the adopted information technologies (e.g. relational model). Since the underlying models of the adopted information technologies are strictly-structured hard to adjust, this means that the underlying models of these IT solutions will have the same properties. On the other hand, not all information and business processes within the organization satisfy these requirements. Only the most important easy to restructure information and well-defined business processes are considered when deigning the underlying model of the restructuring based IT solution. Information and business processes not considered in the IT solution underlying model design will stay outside the scope of automated management. In addition to that, the organization's information and business processes requirements are continuously changing and hence have to be managed outside the scope of these IT solutions. We showed that documents and document based activities compose that part of the organization's information base and business processes existing outside the scope of current IT solutions. We also argued that we should accept the fact that documents will continue to

exist and they cannot be completely eliminated. This resulted in a gap between information and business processes within the scope of automated management and the actual information and business processes of the organization.

What is needed is a new IT solution with more flexible underlying model that accepts the existence of documents while keeping track of them. In this paper we proposed *D* and *P* which are more flexible models that can be the underlying model for a new IT solution that covers information within/outside automated management provided by restructuring based IT solutions. This means a better capture and management of the of the organization information and business processes as a whole. The proposed IT solution follows document-based automated management approach were it utilizes the restructuring based IT solutions and provide automation needed to maintain the underlying models (*D* and *P*) up-to-date and to provide automated querying needed of information maintained in these models.

As future work, we need to engineer the definition, synchronization and querying components of the proposed IT solution. One main challenge for the proposed IT solution is capturing document operations on *D**. Creative software and hardware information technologies are needed to achieve this for paper [3,4,5], standalone and synthesized documents. Binding a document operation to the right business process instance is another challenge. This is not easy especially due to unpredictable behavior of business processes. Also, smart solutions and techniques are needed to provide efficient automated querying of the *D* and *P* models.

REFERENCES

[1] Aalst, W.M.P. van der., Hofstede, A.H.M. ter. and Weske M. "Business Process Management: A Survey," *Lecture Notes in Computer Science,* Volume 2678, 2003, pp. 1-12.

[2] Abiteboul, S., Buneman, P. and Suciu, D., *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann, San Francisco, CA, 1999.

[3] AbuSafiya, M. and Mazumdar, S. "Accommodating paper in document databases," *Proceedings of the ACM symposium on Document engineering*, Wisconsin, USA 2004, pp. 155-162.

[4] Andreoli, J. M., Castellani, S., Grasso, A., Meunier, J.-L. , Muehlenbrock, M., O'Neill, J., Ragnet, F., Roulland, F., and Snowdon, D. "Augmenting Offices with Ubiquitous Sensing," *Proceedings of Smart Object Conference*, Grenoble, France, 2003.

[5] Arregui, D., Fernstrom, C., Pacull, F., Rondeau, G., Willamowski, J., Crochon, E., and Favre-Reguillon. F., "Paper-based Communicating Objects in the Future Office," *Proceedings of Smart Object Conference*, Grenoble, France, 2003.

[6] Booch, G., Rumbaugh, J. and Jacobson, J., *The Unified Modeling Language User Guide*, Addison-Wesley, Redwood City, CA, 1999.

[7] Date, C.J., *An Introduction to Database Systems*, Addison Wesley, Boston, MA, 2003.

[8] Gelinas, U., Sutton, S. and Fedorowicz, J., *Business Processes and Information Technologies,* Thomson Learning, London, UK, 2004.

[9] Karagiannis, D. B. "BPMS: business process management systems," *ACM SIGOIS Bulletin archive,* Volume 16, Issue 1, 1995, pp.10-13.

[10] Lawrence, P., *Workflow Handbook 1997*, Workflow Management Coalition. John Wiley and Sons, New York, 1997.

[11] McHugh, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J. "Lore: A Database Management System for Semistructured Data," *ACM SIGMOD Record*, Volume 26, Number 3, 1997, pp. 54-66.

[12] Object Management Group "UML Home Page," http://www.uml.org, December 2007.

[13] Workflow Management Coalition "Workflow reference model", http://www.wfmc.org. June 2007.

AUTHORS' BIOGRAPHIES

Majed AbuSafiya is a PhD student in Computer Science Department, New Mexico Tech, New Mexico.

Subhasish Mazumdar is an Associate Professor of Computer Science at the New Mexico Institute of Mining and Technology (New Mexico Tech). He received his B.Tech (Honors) and M.E. (Distinction) in Electronics and Electrical Communication Engineering from the Indian Institute of Technology, Kharagpur and the Indian Institute of Science, Bangalore respectively; his M.S. and Ph.D. degrees in Computer Science were from the University of Massachusetts at Amherst. His current research interests include integrity of distributed and mobile databases, document management, and the use of conceptual modeling in software development. For his research he has received support from the National Science Foundation and the Sandia National Laboratory. He is a member of the ACM, IEEE Computer Society, and Sigma Xi.

APPENDIX

We will give an example to show how to define a business process. Consider the *receive admission application business process* which starts by receiving student's documents. The student folder should be gotten from the set of student's folders. If his folder is found, the received documents are added to his folder, otherwise a new folder is created and documents are added. The folder is checked for completeness (i.e. contains an application form and a transcript). If so, his GPA is checked. If less than 3.0, a rejection letter is sent to him. If the GPA requirement is satisfied, the folder is sent to the department. If the application is not complete, a letter is sent requesting the missing document.

From the description the following set of activities can be identified: (1) add documents to the student folder, (2) check if the application is complete, (3) check GPA requirement, (4) send rejection letter, (5) mail folder to department, and (6) request missing document. Note that there is no general rule for specifying the activities of a given business process. We considered very simple business process and activities for the sake of simplicity and space. Figure 13 shows the business process using activity diagram and the corresponding model as a program.

DocumentList *documentList*	
ApplicationForm *applicationForm*	
Transcript *transcript*	
Folders *folders*	
Folder *folder*	
RejectionLetter *rejectionLetter;*	
RequestLetter *requestLetter*	
boolean *complete*	
boolean *lessThanThree*	

ReceiveAdmissionApplicationBP(){
   addDocumentsToFolder()
   complete=folder.isComplete()
   if *(complete)*
     lessThanThree=gpaLessThanThree()
     if *(lessThanThree)*
       rejectStudent()
    else
       mailToDepartment()
  else
    requestDocument()
}

(a) Business process as an activity Diagram

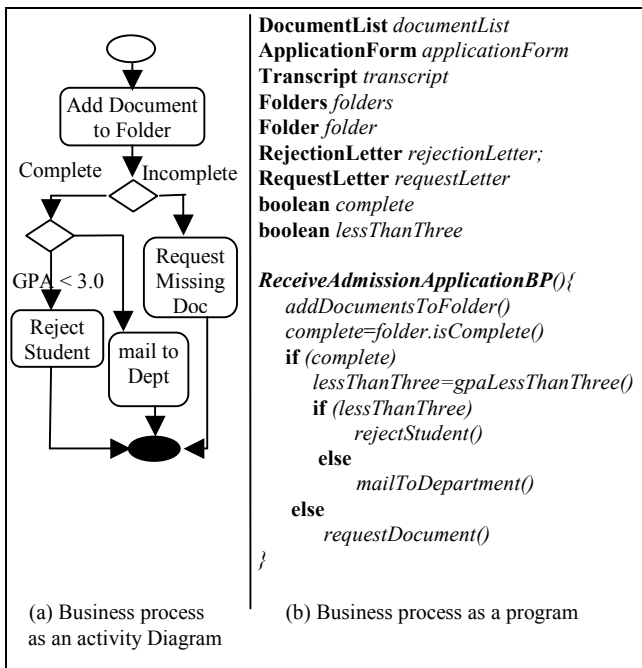(b) Business process as a program

Figure 13: Receive Admission Application Business Process

Once activities are identified, specify document operations carried out in each of them. For every document operation, define a document operation call by defining a corresponding document variable and a corresponding operation call on this variable. To define a document variable choose a name and bind it with a corresponding document type (**D** document types). If this document type is not defined, define that type following the methodology proposed. If that operation is not already defined in the corresponding document type, define that document operation. Finally, specify the logic flow among these document operation calls.

For example, a definition of *addDocumentsToFolder* activity is shown below. We will refer to the document variable with the name of its document type but with its first letter as a lower case

```
/* folder is the folder of the student*/
/* folders: set of folders viewed as a document*/
/* documentList: documents received*/

documentList= new DocumentList()
                              /* receive documents */
foler=folders.getFolder(documentList.getStudentName())
                              /* get student folder*/
if (folder=null)                      /* not found*/
     folder= new Folder()        /* create folder*/
folder.addDocuments(documentList)
                        /*add documents to folder*/
```

Note that *documentList* corresponds to the input to this activity and the instantiation of that variable corresponds to the receiving of student documents. For every identified document operation, we associated that document with a variable and bound that variable to a matching document type. Note that in the English description, when referring to a document we refer to a *d** document. But in the business process definition we are using the type definition of *document(d*)*. Remember that when we define a document, we need to define how the attributes of the document are changed due to the application of that document operation. In this example, we will show only the effect on the informational content attributes. Table 1 shows the definition of *getFolder* and *addDocuments* operations.

Note that activities are nothing but a functional abstraction of a set of document operations. Input to activities can be defined in two ways: by passing document variables as parameters or by assuming that document variables are available to the activity as a global variable.

We have chosen the second approach in this example. The detailed specification of defined activities can be seen in Table 2. Document operations defined for corresponding types are shown in Table 3.

The detailed specification of these operations can be specified by defining their effect on the attributes of the document. We adopt the semi-structured approach to define the documents informational content. The informational content structure of the above documents can be seen in table-4. In addition to these document operations, there is a set of primitive document operations that are defined for documents (e.g. *getField* and *setField*).

Table 1: *getFolder* and *addDocuments* Operations

Document Operation	Specification

folders.getFoler(studentName)	**for** *(folder* **in** *folders)*
	  if *(studentName=*
	      folder.getField(/applicationForm/studentName))
	   return *folder*
	return *null*

folder.addDocuments(documentList)	**for** *(document* **in** *documentList)* *folder. addDocument(document)*

Table 2: Activities and Corresponding Document Operations

Activity	Specification
applicatonComplete	**return** *folder.isComplete()*
gpaLessThanThree	*folder.gpaLessThanThree()*
rejectStudent	*stdName= folder.getField("/applicationForm/StudentName")* *address= folder.getField("/applicationForm/address")* *rejectionLetter=* **new** *RejectionLetter(stdName)* *rejectionLetter.mail(address)*
mailToDepartment	*department= folder.getField(/applicationForm/department)* *folder.mail(department)*
requestDocument	*missingDoc = folder.getMissingDocument()* *address = folder. getField(/applicationForm/studentAddress)* *requestLetter=* **new** *RequestLetter(missingDoc)* *requestLetter.mail(address)*

Table 3: Document Operations

Document Operation	Document Type	Detailed description
isComplete	*Folder*	**if** *(folder.contains(applicationForm)* **and** *folder.contains(transcript))* **return** *true* **else** **return** *false*
getField(path)	*all types*	Takes a field path in the informational content structure as a parameter and returns the field value
mail(address)	*RejectionLetter, RequestLetter, Folder*	Mail the document to specified address
contains(child)	*Folder*	Returns true if there is child for the information content with the name *child*
gpaLessThanThree()	*Folder*	*gpa=folder.getField(/transcript/gpa)* **if** *(gpa<3.0)* **return** *true* **else** **return** *false*
addDocument(doc)	*Folder*	adds *doc* as a child for *folder* informational content
getMissingDocument()	*Folder*	**if** (**not** *folder.contains(applicationForm))* **return** *"applicationForm"* **if** (**not** *folder.contains(Transcript)*) **return** "Transcript"

Table 4: Informational Content Structure of Documents Involved

Document Types	Informational Content Structure
ApplicationForm	*applicationForm* → *studentName address department* *studentName* → *<text>* *address* → *<text>* *department* → *<text>*
Transcript	*transcript* → *studentName gpa* *studentName* → *<text>* *gpa* → *<text>*
Folders	*folders* → *folder** (* *means zero or more)*

Folder	*folder* → *applicationForm transcript* |
	     applicationForm |
	     transcript (| *means* **or**)

RequestLetter	*requestLetter* → *studentName address missingDocument* *missingDocument* → *"applicationForm"* | *" transcript"*
RejectionLetter	*rejectionLetter* → *studentName address rejection* *rejection* → *<text>*