A Model for Office Document Processing and Collaboration in Cloud

Jiafei Wen, Shui Lam, and Xiaolong Wu
Computer Engineering and Computer Science, California State University Long Beach,
Long Beach, California, USA

Abstract - *Office software suite is one of the most widely used and well developed applications. The suite includes word processor, spreadsheet, presentation, and so on. With recent developments of high-speed network and distributed computing technologies, some office applications such as document processing, have turned web-based (e.g., Microsoft Office WAC) and even cloud-based (e.g., Google Docs). The initial and obvious benefits of moving office applications into the cloud for small and medium sized businesses are cost savings in the purchase, maintenance and upgrades on both software and hardware. But the more significant advantage of doing so is to enable users of real-time collaborative editing on a shared cloud-based document. We believe moving office applications into cloud is an inevitable trend in the development of office applications. A novel, efficient cloud-based document processing model: DPC is proposed in this paper. Detailed description and functionality of this model are first introduced. We then instantiate the model using a document based on ISO 29500 [1]. Finally, we also discuss security, reliability and synchronization issues.*

Keywords: Office document processing, cloud, collaborative editing, logic structure, DPC.

1 Introduction

Office documents produced nowadays can be very sophisticated. Besides text, they may contain complex formulas, graphic illustrations, video clips, and control information. Therefore, modern office document processing software systems are large, complex and powerful. They have evolved from simple text processors to systems that deal with lots of objects and complex tasks.

The office document processor market is huge. To compete for or maintain the market share, software developers continue to upgrade their products with added features and capabilities. The Microsoft Office Suite is a prime example, which has undergone frequent upgrades in recent years. This phenomenon is not always beneficial to the end-users in terms of cost. The frequent upgrades with more functions that demand more storage space and faster CPU mean that users would have to continue to invest money on both software and hardware for the purchase, maintenance, and upgrade, just to support routine document production and processing. The rise of cloud computing technology provides an alternative approach that businesses may adopt to address this problem.

Cloud computing is a fairly new technology developed in distributed computing., The technology enables services and storage facilities to be provided over the Internet, thereby allowing users to access the services and storage facilities anywhere in the world, any time, wherever the internet access is available. A service provided through cloud computing may be an application, a computing environment, an IT infrastructure, or even a business process. Document processing is an obvious candidate service to be offered in the cloud, so that users can create, maintain, and share their documents without installing a complex software suite or needing a powerful computer to support the tasks. As such, users can expect to save thousands of dollars on both hardware and software. Furthermore, a cloud provider would have the business and economic incentives to maintain and improve efficiency of its computing facilities through proper and timely upgrades. Though payments for the service will partially offset the cost savings on hardware and software upgrades, the added benefits of using the cloud to support document processing in a business by shifting user focus from application and hardware upgrades to innovative use of the latest functionalities for their document productions should not be underestimated.

The trend that the IT industry has adopted a strategic position of moving their document processors into cloud is quite obvious. First was Google with its cloud-based Google Docs [3], and more recently Microsoft that offered its Microsoft Office 365 [4]. Both of these services claim to emphasize the support of collaboration of document editing among users. However, a closer look at these products led us to believe that the collaborative editing in Google Docs and the co-authoring in Office 365 both lack a proper granularity. For example, Google Docs's support of fine-grained collaboration that allows multiple users edit one sentence, even one word, concurrently, have been found to have led to confusion and disorder among editing users. In fact, many users have complained about this kind of confusion in the Google Docs help forum [5]. Therefore,

this level of granularity may not be suitable for collaborative document editing to some extent.

Aiming to enable users to process their office document collaboratively by a proper granularity in the cloud, we propose a Document Processing in Cloud (DPC) model in this paper. This paper is divided into four parts: background of DPC model, DPC model proposing, DPC model instantiating and discussion.

2 Background

Most office documents created nowadays are XML-based with embedded structure that describes the document content. The structure of an XML-based document defines a hierarchy, with pieces of document at the root below which are parts that make up the document. These parts are referred to as objects and each object has its own functions. Some objects can be further subdivided into smaller parts are referred to as composite objects, while others that represent smallest parts in the structure of a documents are called basic objects. For example, a sentence in an office document may be marked up as an object named as "run" (a basic object), and a paragraph may also be marked up as an object named as "P", which is a composite object and would be defined as the parent of a "run" object in the XML structure.

The hierarchical structure of an XML-based office document is the key requirement for our proposed DPC model. For most documents, the XML hierarchy consists of two categories of structures: geometric (layout) and logical [6]. A typical document is no doubt made up of pages, blocks, paragraphs, sentences, and so on that are defined as objects. When the subdivisions are based on the specific geometric layout of the document, such as page and block, the result is a geometric (layout) structure. When they are based on the human-perceptible meaning of the content, such as paragraph and section, the result is a logical structure. These two structures provide alternative but complementary views of the same document [7] and a mapping between these two structures may exist. Both structures are hierarchical in nature and can be represented by a tree as illustrated in Figure 1.

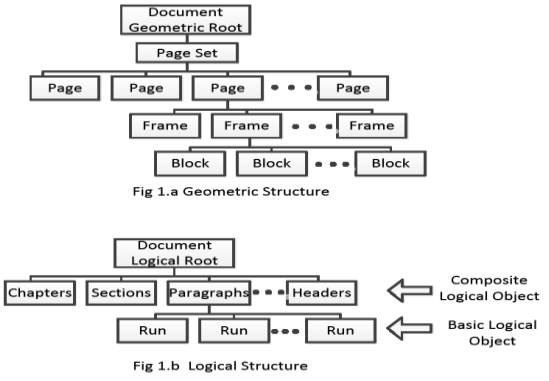

Figure 1. Two Structures.

Although both structures can represent a document comprehensively, we adopt the logical structure in developing our model for the following reasons: firstly, XML-based office documents normally use logical objects for markups; secondly, our proposed model is designed for the cloud. In a cloud, a task will be "divided" into several subtasks to be undertaken on different application servers. Then, each application server performs its assigned tasks using shared resources such as computing resources, data sources and application services. Finally, the results from all involved application servers will be collected and combined to produce the final solution of the given task [8]. For document processing, users tend to spend more time working on their documents based on the logical level, such as adding paragraph, than based on the geometric level. In light of these considerations we believe the logical structure of a document is the better choice for designing our model for document processing in a cloud environment.

In this paper, we use Microsoft Word to illustrate the development of our model. An overview of a logical structure of a Word document is shown in Figure 2.

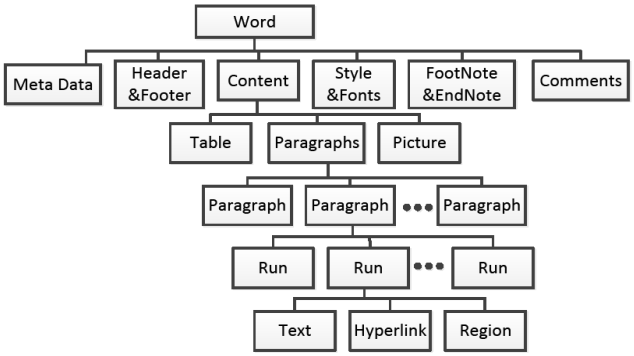

Figure 2. Logical structure tree overview of word document.

3 DPC Model

From the logical structure shown in Figure 2, we consider the non-leaf nodes in the tree as composite objects, and the leaf nodes as basic objects. More details of these two objects are discussed in the following paragraph. Our proposed DPC model is object-oriented based on these objects. It treats editable components in a document as distinct objects. DPC also gives users respective access to objects which are components of a whole document. As a result, multiple users working on one document can do collaborative editing at the same time. Our cloud-based model supports the distribution of individual objects to available processors in the cloud. Object-oriented DPC provides a more logical granularity for document processing collaboration. Instead of treating the content of a document as a string stream as Google Docs does, and which had been found to cause user confusion and disorder during concurrent editing among collaborators, we use

objects which can easily be distributed to different servers to be worked on by different users.

As an object-oriented model, our DPC model divides a document into objects. Each part of the document is assigned to a sub-task, and each subtask is a unit of work distribution to processors in the cloud. The sum of all edit tasks is the entire document. For example, if a user wants to edit a paragraph in a document, this paragraph object will be sent to a processor in the cloud and the editing subtask will be accomplished on that processor. The user performs the editing through remote access to the assigned processor. While this paragraph object is being edited by this particular user, other parts of the document are available and accessible to other users. Consequently, DPC enables multiple users to collaborate concurrently on the same document, and the collaboration is accomplished at the object level.

As discussed in Section I, collaboration in a cloud environment requires proper level of granularity. In our DPC model, the proper granularity lies on a proper design of objects in a document. DPC designs objects based on three criteria: 1) objects must reflect the logical structure; 2), the design of an object must take into consideration the editing frequency of each part in the logical structure; 3), the design needs to comply with the mainstream office document standards. Based on these considerations, our model defines thirteen objects for document processing in a cloud environment, as shown in Figure 3.

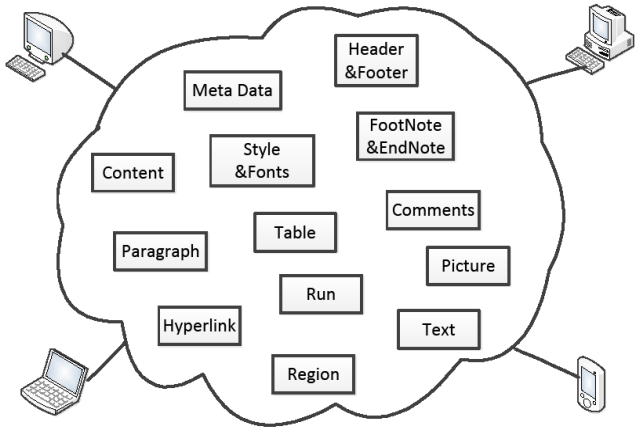

Figure 3. The proposed novel DPC model.

The thirteen objects in Figure 3 are also listed in Table 1. Nine of these objects are composite and four are basic. The composite objects are Content, Meta Data, Header&Footer, Style&Fonts, FootNote&EndNote, Comments, Paragraph, Table, and Run, and the basic objects are Hyperlink, Region, Text, and Picture. Since the objects are the basis of work assignment for editing subtasks, the definition of these objects determines the level of granularity of document processing in a cloud environment. Each object is atomic and cannot be edited by more than one user at any one time.

Table 1. DPC Objects description.

	Object Name	Object Description
Composite Objects	Content	Specifies document's properties.
	Meta Data	Specifies meta data.
	Header& Footer	Specifies headers and footers.
	Style& Fonts	Specifies styles and fonts setting.
	FootNote&End Note	Specifies foot note and end note.
	Comments	Specifies comments.
	Paragraph	Specifies paragraphs' properties.
	Table	Specifies tables.
	Run	Specifies runs' properties of content in the parent field.
Basic Objects	Hyperlink	Specifies hyperlinks.
	Region	Specifies regions.
	Text	Specifies literal text of runs which shall be displayed in the document.
	Picture	Specifies pictures.

Besides these thirteen objects, DPC also needs other components for a complete utilization in the cloud environment. We use the Z notation introduced by Spivey in [9] to define all eight components in our DPC model as follows:

DPC = {DOC, MIDDLEWARE, PROCESSORS《bag PROCESSOR》} (1)
DOC = {ROOT, ASSIST_INFO} (2)
PROCESSOR = { APPS《bag APP》} (3)
ROOT = {NONLEAF《bag COMPOSITE _OBJ》, LEAF《bag BASIC _OBJ》} (4)
COMPOSITE_OBJ = {OBJECT, DESCENDANT《bag ROOT》} (5)
BASIC_OBJ = {OBJECT} (6)
OBJECT = {NAME, ACCESS_PATH, ON_EDITING} (7)
ON_EDITING :: = Busy | Idle (8)

Formula 1 defines that DPC consists of three parts: DOC, MIDDLEWARE and PROCESSORS. Details of DOC and PROCESSORS will be discussed in the following paragraphs. MIDDLEWARE is the middleware [10] in the cloud, which is the software layer that "sits" between application servers (processors) and the resources (document and its pieces). MIDDLEWARE provides the link and passes data among resources and servers. In DPC, MIDDLEWARE is primarily responsible for reading/storing data from/to storage, generating objects from the division of the document, providing the link among objects processed by multiple distributed servers, and combing the results of sub-tasks to produce the final document.

Formula 2 specifies that DOC is the collection of all information contained in an office document. It is composed of two parts: ROOT and ASSIST_INFO. ROOT will be discussed in more detail below. ASSIST_INFO is the assistance information on the office document that is required for its processing, such as read-only sign.

Formula 3 defines the PROCESSORS component in the cloud. It typically comprised of multiple document processors, which are application servers available in the given cloud. These application servers can be substantially heterogeneous, thereby allowing servers with various specialized capabilities to be used in processing a single document. Since these servers are being shared among many cloud users, the effectiveness of document processing is improved without incurring high costs. For example, users who may wish to use handwriting pad to draw pictures in their Microsoft Office document cannot do so without buying add-in software that supports such functions. This translates to additional financial and time burden. However, a cloud created to serve a large and diverse population of document processing users would likely have a variety of add-on capabilities, including hand-drawing tools, so that a user may access them at little extra costs. This advantage may be especially obvious for large businesses, which normally have greater demands for sophisticated documents that contain embedded objects or links to external databases and require complex add-ins for their correct and effective processing.

Formula 4 defines ROOT, which is the aggregation of all objects in the DPC division. It has two components: NONLEAF and LEAF objects. NONLEAF is a collection of COMPOSITE_OBJ (composite objects). LEAF is a collection of BASIC_OBJ (basic objects).

Formula 5 defines COMPOSITE_OBJ, which is an object with descendants. As a result, one constituent part of COMPOSITE_OBJ is DESCENDANT. Because of the recursiveness attribute of the tree structure, DESCENDANT is a collection of ROOT, defined in Formula 4. COMPOSITE_OBJ also has another component, OBJECT, to identify itself. OBJECT will be defined in Formula 7.

Formula 6 defines BASIC_OBJ, for basic objects, which are leaf nodes in the tree structure. It only has OBJECT item to identify itself.

Formula 7 defines OBJECT, which is used to identify all objects including COMPOSITE_OBJ and BASIC_OBJ in DPC. It has three constituent items: NAME (name of the object), ACCESS_PATH (accessing path to the object) and ON_EDITING (the sign of whether it is on editing).

ACCESS_PATH is an accessing path which guides users to reach the target object in the given cloud. Because objects in DPC are derived from the nodes in a logical tree through divisions, every object corresponds to a node in the logical tree. Meanwhile, as described by XML, each node in the logical tree has its own unique XPath [11], so we adopted XPath to be identifiers of the nodes. By extension, some of these XPath become the identifier of objects in DPC because of the corresponding relationship mentioned above. The identifier of an object is the accessing path (ACCESS_PATH defined in Formula 7) that guides user to get in that object. As a document is being divided, the XPath of each node is recorded, and some of them become the ACCESS_PATH of the objects in DPC by division. Then the ACCESS_PATH will be sent with its corresponding object to a target processor. Users get the target object through its ACCESS_PATH. For example, the ACCESS_PATH of the object of "the texts of the first run in second paragraph in document" is: /document/body/paragraph [2]/run [1]/text. Once a processor is selected for the object's processing, the ACCESS_PATH will become the ID of that object on the processor and will be sent to the processor along with its corresponding object. At the other end of the processing, ACCESS_PATH also plays a key role in combining results from servers to produce the final document by giving each result of a subtask its location on the final document. In the above example, when the result of this subtask needs to be combined with results of other subtasks, the ACCESS_PATH will provide its location in the final document, which is the second paragraph's first sentence content. Due to the different ACCESS_PATH leading to different server, different user can access different parts of the same document, accomplishing collaborations at the logical object level among distributed processors in the cloud.

Finally Formula 8 defines ON_EDITING, which is a sign and works as a write lock. When it is busy, the object is locked to other users. When a user gets the link to a target object, this user will have the editing right as long as the ON_EDITING sign of that object is IDLE. The object will set its ON_EDITING sign as "Busy" when there is someone editing it. As the objects in DPC are isolated with each other, there is no inheritance relationship between any two objects. So do the ON_EDITING signs, which means even though object A is the ancestor of object B in DPC, there is no inheritance relationship of ON_EDITING sign between A and B, because A and B are sent to different processor, editing one of them on a processor won't affect the other on other processor. This state of ON_EDITING is crucial for collaboration in a cloud, because how often it is set to "Busy", and the duration of the "Busy" state impacts the collaborative editing in cloud deeply. In our DPC model, these two conditions depend on the granularity of document processing. As mentioned above, the granularity is decided by the design of objects in DPC. If the basic object is designed as phrase rather than text of a run in DPC, the collaboration will get finer granularity. A finer granularity is not always good for users, as it will bring confusion to when several users are editing one phrase. Besides, doing so will cost more time to divide the document and combine the results from individual servers.

4 Instantiate DPC

To help clarify how our DPC model works, we instantiate it using a real world example, the instance, which is an example office document and used to instantiate DPC, is based on ISO 29500. The logical

structure of ISO 29500 [12] and its mapping with DPC is shown in Table 2.

Table 2. ISO 29500 mapping with DPC.

ISO 29500 (Word Processing)	Objects in DPC
|------ XXX.docx	
|------ Metadata	Meta Data
|------ Document	Content
|------ P	Paragraph
|------ Run	Run
|------ Text	Text
|------ Hyperlink	Hyperlink
|------ Region	Region
|------ Drawing/VML	Picture
|------ Revision	Content
|------ Region	Region
|------ SectPr	Paragraph
|------ SectPrchange	Paragraph
|------ Revision	Content
|------ Region	Region
|------ Table	Table
|------ P	Paragraph
|------ SectPr	Content
|------ SectPrchange	Content
|------ DocumentRels	Content
|------ Settings	Content
|------ Styles	Style&Fonts
|------ FontProps	
|------ Endnotes	FootNote&EndNote
|------ EndnotesRels	
|------ Footnotes	
|------ FootnotesRels	
|------ Header	Header&Footer
------ HeaderRels	
|------ Footer	
|------ FooterRels	
|------ Comments	Comments
|------ CommentRels	

An example document, based on the instance standard, named as "the old man and sea" is used here to illustrate the mechanism of DPC. The display of this example document is shown as Figure 4.

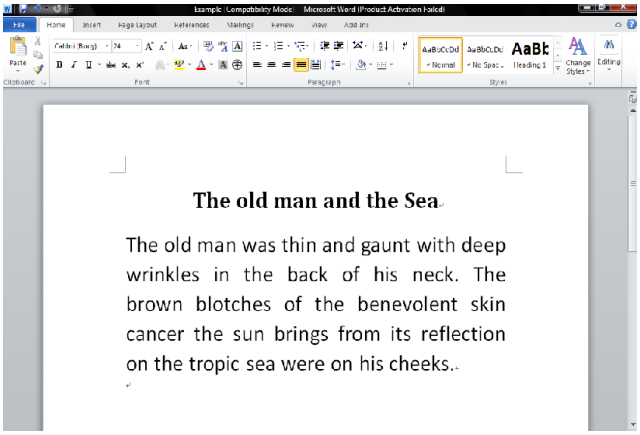

Figure 4. Display of example document.

We take the composite object "document" in this example as an example object, which is described in XML as:

```
<w:document
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:pPr>
        <w:pStyle w:val="a5"/>
      </w:pPr>
      <w:r>
        <w:t>The old man and the Sea</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:pPr>
        <w:spacing w:beforeLines="100"/>
        <w:rPr>
          <w:rFonts w:hint="east Asia"/>
        </w:rPr>
      </w:pPr>
      <w:r>
        <w:t> The old man was thin and gaunt with deep wrinkles in the back of his neck. The brown blotches of the benevolent skin cancer the sun brings from its reflection on the tropic sea were on his cheeks.
        </w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

The composite object "document" contains five composite objects: content, two paragraphs and two runs, and two basic objects, which are two texts. Each composite object has the "w:pPr" attribute to represent the paragraph properties. Basic objects "w:r" represent the text of the paragraph. These five objects will be sent to five servers in a cloud with their own XPath as their accessing path shown in Table 3.

Table 3. XPath of object in instant.

Object	XPath
document	../w:document
First paragraph	../w:document/w:body/w:p[1]
First paragraph's properties	../w:document/w:body/w:p[1]/ w:pPr
First paragraph's text	../w:document/w:body/w:p[1]/w:r/w:t
Second paragraph	../w:document/w:body/w:p[2]
Second paragraph's properties	../w:document/w:body/w:p[2] / w:pPr
Second paragraph's text	../w:document/w:body/w:p[2] /w:r/w:t

In DPC, a paragraph is a composite object, and text is a basic object, and each of them has an ON_EDITING sign. As there is no inheritance relationship between paragraph and text, so their ON_EDITING signs do not have the inheritance relationship either. In this case, if a user edits the property of the first paragraph, the text of this paragraph is still available to other users, while the ON_EDITING sign of the first paragraph is BUSY. An overview of this example document processed in cloud based on DPC is shown as Figure 5.

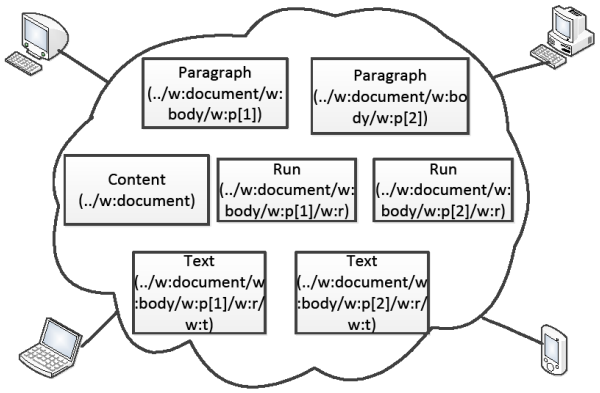

Figure 5. Overview of example document in cloud based on DPC.

Accessing path is used to lead user to get the sub-part of the office document, which is the key concept of collaboration editing in cloud. Through accessing path, multiple users can get access to the part they want to edit separately. For example, if user wants to edit the text of the first paragraph, the middleware in cloud will lead user to the processor which marked this object's accessing path as "../w:document/w:body/w:p[1]/w:r/w:t", and then user will accomplish their editing on that processor.

5 Discussion

The first problem of moving office document processing into the cloud environment using DPC is the security problem. As DPC divides a document into smaller pieces and sends them to servers that may be geographically disperse, how to ensure the security of the document would be a legitimate question. The best solution so far is to send pieces of the document to authorized servers with secure channels. The limitation of this method is that it will narrow the range of application servers in a cloud for document processing.

As different parts of a document are being processed in different processors separately, a second problem of concern is reliability. If one of the servers crashes while processing, the final document cannot be integrated and produced, because the crashed processor will not return the result of its part. A solution for this problem is to back up the document before it is divided in storage. If server crash does occur, the backup copy of those pieces on crashed server will be used to constitute the final document. Synchronization is a common problem encountered in all distributed computing, including cloud computing. For DPC, the user who opens a document first in a cloud is regarded as the owner of the document; other users collaborate on this document in two steps: (a), obtain the permission from the owner; (b), save their work before the owner closes the document, otherwise their work will not be saved. Meanwhile, for DPC, synchronization is on the logical object level which is a proper granularity for document processing.

6 Conclusion

As people realize that cloud computing will reshape the IT industry, some even predict that one day it will become the 5th utility after water, electricity, gas and telephone [13]. Moving document processing into the cloud is a logical and strategically wise choice. This movement is not only based on cost-saving considerations, but also on the merits that include easier and more effective collaboration in document processing. Aiming to accomplish this goal completely, we have proposed a DPC model to enable users process their office document collaboratively in a cloud environment at a proper level of granularity so that concurrent editing tasks are more effectively isolated, thereby eliminate any confusion that may occur as a result of collaboration at a finer granularity.

7 References

[1] ISO/IEC 29500, Information technology – Office Open XML file formats. ICS: 35.240.30; 35.060.

[2] Cloud Computing Definition by National Institute of Standards and Technology (NIST): http://csrc.nist.gov/groups/SNS/cloud-computing/.

[3] Create document, spreadsheet and presentations on line. http://www.google.com/google-d-s/intl/en/tour1.html

[4] About Office 365. http://office365.microsoft.com/en-US/online-services.aspx.

[5] Sharing.http://www.google.tm/support/forum/p/Google+Docs/label?lid=7d298e0b22c2d291&hl=en

[6] ISO 8613: Information Processing-Text and Office Systems-Office Document Architecture (ODA) and Interchange Format, International Organization for Standardization, 1989. ICS: 35.240.20.

[7] Yuan Yan Tang, Chang De Yan, and Suen, C.Y. Document Processing for Automatic Knowledge Acquisition. IEEE Transactions on Knowledge and Data Engineering, Vol. 6, February 1994. Pages 3-21.

[8] I. Foster and C. Kesselman (editors).The Grid Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, USA, 1999. Pages 220-221.

[9] J. Michael Spivey (1992). The Z Notation: A reference manual (2nd edition). Prentice Hall International Series in Computer Science.

[10] Middleware. http://en.wikipedia.org/wiki/Middleware#Use_of_middleware.

[11] XML Path Language (XPath) 2.0. W3C Recommendation 23 January 2007, http://www.w3.org/TR/xpath20/, February 2009.

[12] Qian Wu, Ning Li, and Chunyan Fang. Comparison and conversion between word processing format UOF and OOXML. Application Research of Computers, Vol. 26, February 2009.

[13] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I.Brabdic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Readily for Delivering Computing as the 5th Utility. Future Generation Computer Systems, Vol. 25, Issue 6, June 2009, Pages 599-616.