Document Management and Tracking System for Emergency Response Headquarters

Wataru Sendo[1], Norihisa Segawa[2], Jun Sawamoto[2], Eiji Sugino[2], Masato Yazawa[3], and Shinji Akitomi[4]

[1,2] Iwate Prefectural University (Graduate School), 152-52 Sugo, Takizawa, Iwate 020-0693, Japan
g231l021@s.iwate-pu.ac.jp, sega@acm.org, {sawamoto,sugino}@iwate-pu.ac.jp
[3] Mathematical Assist Design Laboratory, 1058-5 Yoshizawa-cho, Ota, Gunma 373-0019, Japan
yazawa@mail.wind.ne.jp
[4] Iwate Medical University, 19-1 Uchimaru, Morioka, Iwate 020-8506, Japan
shin-zi@pop12.odn.ne.jp

Abstract. One problem faced by an emergency response headquarters is that emergency response procedures are not adequately documented. This lack of documentation is a major problem in testing preparedness for future emergencies. However, it is difficult to document an emergency response while actually responding to an event. The authors have attempted to resolve this issue by building a content management system with a focus on documentation created during emergency responses and on documentation that can be printed out and attached.

Keywords: Disaster information, content management, operations support, automation, image processing.

1 Introduction

One issue in Japan currently is learning from the lessons of the Great East Japan Earthquake and Tsunami, and verifying that organizations are prepared for any future disasters. One extremely important mission for handling future disasters is verifying the decision-making and action processes and operations of an emergency response headquarters, which are key to an emergency response.

With respect to the verification of the operations at an emergency response headquarters, one current topic is that the minutes of the emergency response meetings are not recorded [1]. Given insufficient documentation, it is unclear at the time an instruction or command is given by the emergency response headquarters, what information that instruction or command was based on. The people involved in emergency response events are working to rectify this problem by creating minutes after the fact, but this relies on human memory, and as such is lacking in reliability.

One obstacle to documentation in an emergency response headquarters is the sheer amount of work that must be carried out in the initial outbreak stages of an emergency (within the first 72 h). The main responsibilities are gathering and sharing information, linking up with other relevant organizations and authorities, and issuing instructions and commands based on the information gathered. Information gathering often occurs by sending and receiving paper faxes, and sharing information with everyone often involves copying paper documents onto whiteboards or pasting them onto presentation panels. Paper documents created at an emergency response headquarters or gathered from relevant authorities contain information about the state of the affected area or parties that is required for deciding on an appropriate emergency response, and other information about the staffing of the relevant organizations and their emergency response. All of this forms not just the basis for the decision making and orders issued by the emergency response headquarters and the related organizations, but it is also considerably important material for verifying in future whether the response was appropriate on the basis of the information available at the time, and for verifying preparedness for future emergencies.

However, while responding to an emergency, it is difficult to make changes to the created documentation, or to entirely record when information was shared or how widely paper documents were distributed. Further, unlike email or electronic files, copying or sending paper documents (such as via a scanner, by fax, or by hand) does not leave any computer log, making it harder to trace after the fact.

In our research into this problem, we developed a content management and tracking system that can be set up at an emergency response headquarters in the event of a major disaster, such as an earthquake or tsunami.

2 Related Cases

Content management systems are widely used to manage content and versioning for documents created on computers. Well-known content management systems (CMSs) include Alfresco [2] and KnowledgeTree [3]. These systems are web-based, and system users access the server running the CMS via web browsers running on their client machines, using the browser to upload, manage, and otherwise operate on the document content.

Managing documentation by using a web browser from an emergency response headquarters differs from the normal workflow and thus, requires prior training. There is also the possibility of human error leading to failures in documentation management for areas differing from the normal workflow, such as forgetting to upload documentation. It would not be helpful to increase the user workload at an emergency response headquarters, where considerable output is already demanded of workers during an emergency response. What is needed is a system that reduces human error as much as possible, while also not introducing any impediments to user work.

It is also conceivable that paper documents posted and used in decision making at one point in time might have taken down and not used in decision making at a later point in time. A CMS must allow users to not just manage document content and versioning but also print documents and manage them after printing and posting in an emergency response headquarters.

3 System Concept

We built a CMS for recording document content and changes to this content as authored in an emergency response headquarters, and for tracking the status of document sharing within the headquarters over time. Figure 1 shows how a CMS can be used in an emergency response headquarters.

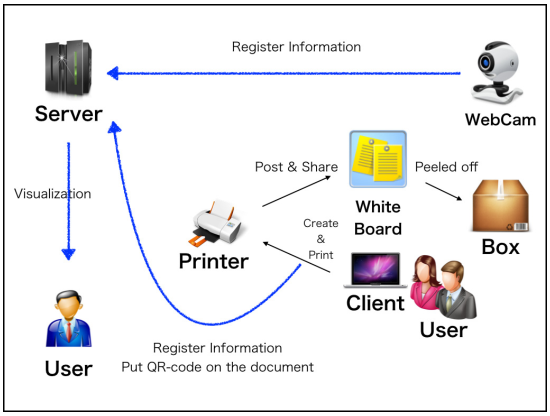


Fig. 1. CMS Overview

When users create documents or print them out for sharing, the CMS starts managing document data in the background. These document data are sent to the server, and making them visible via the web makes it possible to view and verify the information using a web browser. Document data managed by the system are as follows: Document Title, Content, Creator, Created, QR Code Link Document ID, Document Revision ID.

The CMS also adds a QR code to a document when a user prints it. Users can then manage date information for when a document was posted to a whiteboard or taken down again by using a web camera to scan the QR code. The information recorded for paper documents are as follows: QR Code Link, Data Posted, Data Removed(last date confirmed as posted).

The most significant feature of the CMS is that documents are managed automatically without requiring users to even be aware of the process. Users do not have to utilize any specialized tools other than the ones that they regularly use in their work.

These various types of data are also collected in real time, such that workers in areas with a functioning Internet infrastructure can monitor the situation at the emergency response headquarters and possibly help in decision making processes. However, some problems remain in terms of how to manage who can access which information. These issues will be dealt with in the future.

The next section provides a more detailed description of the system configuration and implementation.

4 System Implementation

This system consists of software used by the workers of an emergency response headquarters to create documents (the client), a server for storing the document data, and a web camera for monitoring documents posted within the headquarters.

The client is implemented so that users do not need to know any commands or operations other than those used normally when creating documents. The client software is installed on a MacBook Pro and includes PHP and the PEAR libraries, as well as shell scripts. Users can use Microsoft Office Word 2007 or later to create documents.

Figure 2 shows the processing flow for managing documents, with the client portion marked clearly.

When a user creates and prints a document, the system catches the printing event, and the document data are temporarily saved to the client machine's working directory as a PDF. No commands have yet been sent to the printer. The saved PDF document is then analyzed using the `pdftotext` feature of the open-source XPDF library for analyzing PDF files. A unique QR code identifying the document is then generated and added to the saved PDF. Once the QR code is applied, the print commands are sent to the printer. The PDF document and the URL obtained by reading the QR code are then registered to the server's database. These data are used to link the saved document with any paper copies posted on whiteboards or elsewhere.

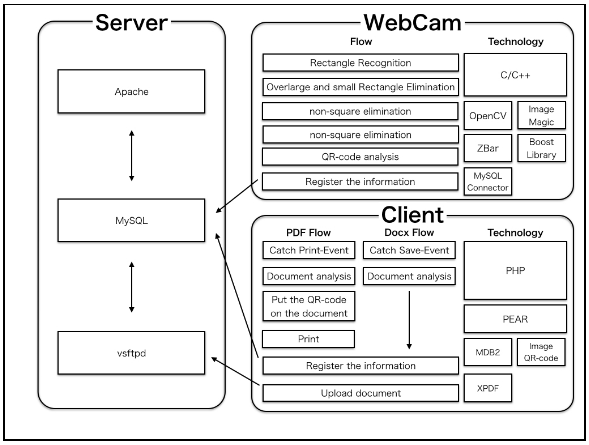
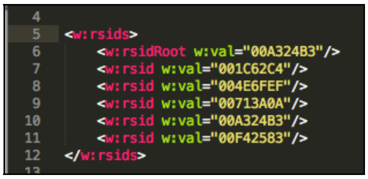


Fig. 2. Document Management Processing Flow

The user next saves the created document. Saving generates a copy of the document in the working directory. Word documents (with the filename extension of `*.docx`) use Office Open XML [4] as the internal data representation of the text and formatting information that make up the document. One notable aspect of this system is that a

Word file can be unzipped to access the collection of the Office Open XML files that make up the document. Analyzing this collection of files makes it possible to get the document text and to get a record of content revisions.

Figure 3 shows the portion of an XML file internal to a Word document that is required to manage document revisions.



```
<w:rsids>
    <w:rsidRoot w:val="00A324B3"/>
    <w:rsid w:val="001C62C4"/>
    <w:rsid w:val="004E6FEF"/>
    <w:rsid w:val="00713A0A"/>
    <w:rsid w:val="00A324B3"/>
    <w:rsid w:val="00F42583"/>
</w:rsids>
```

Fig. 3. XML for Managing Revisions

In Figure 3, the `rsidRoot` element contains a string value that is automatically assigned when a document is first created. The `rsid` elements contain strings that are automatically assigned when the document is edited to add or delete content. For instance, a given document A would have a random value assigned to the `rsidRoot` element when the document was created. For example, let us use a value of `0`. After document A was edited, the `rsidRoot` value would still be `0`, and a new `rsid` value—again random in real life, but here let us use `1`—would be assigned. Irrespective of the number of times the document may have been revised, the `rsidRoot` value clearly identifies this document as document A. Now, assume that this document A has new content added and is then saved as document B. In this case as well, the `rsidRoot` value would be `0`, and there would be an `rsid` value of `1`, as well as a newly assigned `rsid` value of `2`. In other words, no matter what additions, deletions, name changes, or file moves may have occurred for document A, we can discern the original document by looking at the value of the `rsidRoot` element. We can look at the `rsid` values to see how many times the file was revised, i.e., the version of the file.

After analysis, the document title, text, creation date, creator, `rsidRoot`, and number of `rsid` elements are all stored in the server database.

The server regularly searches for DOCX documents with text that matches that of the stored PDF files. DOCX files that match are associated with the URL of the QR code in the PDF file.

This paper has described the process whereby a document is printed and then saved. This system would also work well if the order were instead to save the file and then print. Consequently, users can create documents just as normal without having to worry about how the system works.

Printed documents may be posted to whiteboards for sharing within the headquarters. This system uses a web camera to monitor and record when a document is posted and when it is removed. We used the Buffalo BSW180ABK [5] web camera in the normal video mode for monitoring. To recognize QR codes appearing in the video captured by the web camera, we used C++, the image processing library OpenCV, the barcode and QR code analysis library ZBar, the Boost library, and the ImageMagick library.

The processing flow is shown in the "WebCam" portion of Figure 2.

The first step is to use OpenCV to extract a single frame of the video. Next, we detect any rectangles in the image. In this system, the identifier that we want to recognize is a QR code; therefore, detection excludes any shapes that are not squares or that are too small. We then use ZBar to analyze any barcodes in the collection of found rectangles. If the analysis results in a URL from a QR code that has been detected for the first time, the first and last posting date and time for that document are recorded in the database as the current date and time. If the document has been detected before, the last posting date and time are updated to the current date and time. We can calculate the length of time that a document has been shared within the headquarters by finding the difference between dates and times of the first and last posting. This makes it possible to know that this document was used for emergency response decision making.

Lastly, we process the data stored on the server to make them viewable via the web. For server software, we use the Apache web server, vsftpd for file transfer, and MySQL for the database, and we use the HTML, CSS, PHP. CakePHP framework, and Java-Script as our development languages. The web view shows the stored data along a time axis and provides a search tool and other features.

5 Conclusion

In this paper, we proposed a CMS system built to avoid any impediments to user workflows, as a means of overcoming the difficulties faced in recording documentation in an emergency response headquarters.

In the future, we intend to conduct tests using this CMS in regular daily work and other situations to ascertain whether document content and document postings are correctly managed.

References

1. No Meeting Minutes for Ten Earthquake Response Meetings: Shoddy Document Management. Nikkei. (January 27, 2012), http://www.nikkei.com/article/DGXNASFS2700A_X20C12A1MM0000/ (in Japanese)
2. Alfresco, http://www.alfresco.com/
3. KnowledgeTree, http://www.knowledgetree.com/
4. Office Open XML, http://msdn.microsoft.com/en-us/library/office/aa338205%28v=office.12%29.aspx
5. BSW180ABK, http://buffalo.jp/products/catalog/supply/multimedia/webcamera/60/bsw180a/