Distributed Parallel Multi-Channel Publishing System

Wei Li, Framkom, Rebecca Dählin, Framkom

Abstract
With more and more electronic media appearing and being popularly accepted, the publishing industry is facing a greater challenge than anytime else before. The only solution to cope with these problems is to extend the traditional publishing to cover both print publishing and electronic publishing business. This paper intends to propose an integrated system solution to assist this transition. Distributed Parallel Multi-channel Parallel Publishing, as one of the outcomes of DIP (Digital Information Processing) project, sponsored by STI (Swedish IT Institute) aims to propose a comprehensive solution to cover both paper-based and electronic media publishing, based on advanced distributed software system architecture and file-related up-to-date standards. In the DIP project, the new standard of news industry, NewsML is adopted as the authoring, storing and transforming format. IDF (Job Definition Format) is partly employed to monitor and assist the control of the entire paper-based publishing process. On the aspect of software system architecture, Java-based distributed network technology, Java, acts as the infrastructure of the system.

0. Introduction and Motivation
Nowadays, with the rapid evolution of Internet and web technology, publishing industry is experiencing a revolution on all steps of its processes. One publication has to be delivered parallely in different transferring formats as well as display layouts, according to various accessing client devices, such as web browsers, PDAs, mobile phones and PDF format for printers, etc. This gives birth to multi-channel publishing.

Multi-channel publishing refers to a new publishing model, which leverages the advantages of XML and XSL technologies to separate the content and layout of a publication, allows information to be authored with standardized templates, so that the same information can be re-purposed and redistributed across those multiple delivery channels with or without the same “look and feel” as the original print document. This maximizes the document’s value and makes it easier to update and reuse. Multi-channel publishing will change the present publishing model.

However, there are still some necessary studies to be carried out in order to achieve a highly automated multi-channel publishing system: how to describe and store the publication resource, how to deliver content and layouts according to different client devices; how to bridge the gap between the traditional printing publishing and electronic media publishing process, as well as the consideration on the scalability issue and some other attentions on the software system architecture.

This paper intends to provide an integrated system solution to cover both paper-based and electronic media publishing, based on the study of DIP (Digital Information Processing) project, sponsored by STI (Swedish IT Institute). In the following of the paper, section 2 gives a analysis of the current problems existing in the publishing industry. Section 3 briefly introduces some concepts and terminology, including popular web content description standards XML specification and advanced middleware technologies. Section 4 elaborates in depth on the essence of these new technologies and points out the relevant
issues and approaches in a pragmatic way. Then section 5 lays out the concepts and technologies under the specific scenario of multi-channel publishing, where a comprehensive solution is illustrated together with some up to date publishing industry standards. And the last section concludes the whole work and present some future perspectives on it.

1. Problems in Publishing Industry

The history of the printing industry officially starts back in 1439 when Johann Gutenberg first experimented in using lead figures to print books. But some people argue that the art of printing is much older in that Chinese people have used screen-printing technology for more than 1500 years. Either way the printing industry is old and have distinguished heritage. However when entering the internet and web epoch, the status of this old-lined industry looks no longer steady enough:

The traditional paper-based newspaper product is losing ground on the benefit of digital products. The editions are constantly decreasing and so is the biggest income for the newspaper producers - advertisement fees. In the near future news publishers have to find a way to charge the readers for information and companies advertising on any output medium.

Publishing materials include text, pictures, videos and audio data. These materials are highly required for reuse and sharing. However, due to various reasons, such as centralized storage, copyrights, ownership of the publishing materials, it is quite difficult to share and reuse the materials. Thus, it causes a big waste of these valuable resources.

Other problems that the newspaper producers struggle with are of a more organizational nature. Today every publishing medium means extra work. One has to rewrite or adjust the material to be suitable for the specific output medium. Most times the technical knowledge of the reporter is too limited to fulfill this and it is also hard for newspaper company to afford the cost for every transformation. It is most desirable that this adjustment be automated as much as possible.

All these problems force the publishing industry modernize its way of work, and the burgeoning computer technologies provide the possibility to update the old-fashioned work model.

2. Concepts and Terminology

2.1 XML and related Specifications

2.1.1 XML

Extensible Markup Language (XML) is a meta-markup language that provides a format for describing structured data. XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them [1]. XML documents are designed to be easy for computer processing, as well as being human-readable and reasonably clear. This facilitates more precise declarations of content and more meaningful search results across multiple platforms. In addition, XML is enabling a new generation of web-based data viewing and manipulation applications.
2.1.2 DTD and XML Schema

XML provides a mechanism to define and check the structure of XML documents, with constraints on the storage layout and logical structure could be imposed on XML document. By far, there are two standards DTD (Document Type Definition) and XML Schema for this purpose [2]:

- DTD is used to define the legal building blocks of a XML document. It defines the document structure with a list of legal elements. It uses document-orient syntax and can be declared inline in XML document, or as an external reference. XML schema is a set of constraints for describing the structure of domain information. It describes an internal and inter-document structure for the whole class of documents. The Schema provides a set of validation rules for the organization of XML documents.

XML can use DTD or XML Schema to validate XML documents. Compared with DTD’s deficits: more document-oriented and not easily extensible, XML Schema can be a advanced replacer. However, XML Schema specification is released recently, most of the XML documents at present are still using DTD for construct and validation.

2.1.3 XSLT (Extensible Stylesheet Language Transformation)

XSLT enables and empowers interoperability between XML documents and other web documents. It provides mechanism for converting a XML document to other web document, e.g. HTML document [3]. More commonly, it is used now to convert a XML document to its XHTML [4] counterpart for displaying the document in a standard web browser. Moreover, it can also be used to merge logic (script codes) part into a web page when converting a XML document to a XHTML document. This feature can be used to separate the logic description from the content of web pages.

2.2 Middleware and Mobile Agent

Middleware is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware is essential to migrating mainframe applications to Client/Server applications and ability of communication across heterogeneous platforms. During 1990s, it evolves to support interoperability in terms of object-orient component technology. There are 3 widely-publicized middleware models: Object Management Groups’ CORBA [5]; Microsoft’s COM/DCOM [6] and Sun’s Java-specific RMI [7]. Now, more middleware platforms are written in Java, which support more than one model and can easily talk with different models to re-use the existing software legacy. Middleware technology makes distributed application easily constructed and gives more advanced features such as load balancing to the software architecture.

The increasing development of distributed systems requires more and more efficient technologies to ensure the access and the management of the network resources. In this context, mobile agent technology is considered an enhancement of distributed technologies as it provides powerful and efficient mechanisms to develop applications for distributed and heterogeneous systems.

Mobile agent technology has become popular primarily because of the efficient way it provides for the access and the manipulation of remote information. The mobile agent architecture considers that local interaction is more efficient than the remote one. The solution proposed by mobile agents is for software to migrate to remote hosts, where information is stored, so that the user’s request can be executed locally. It means that agents can take decisions or fulfill previously designed operations in an autonomous way.
on the remote site, even if the home host becomes temporary unavailable. This feature, called mobile logic, gives flexibility to the design of distributed systems.

2.2.1 Message System and Publish/Subscribe mechanism

Messaging System, built from loosely coupled components communicating through messages, is used to build highly reliable, scalable, and flexible distributed applications. A messaging system allows separate, uncoupled applications to reliably communicate asynchronously.

A Publish/Subscribe messaging system is a typical model of Message System. It supports an event driven model where information consumers and producers participate in the transmission of messages. Producers "publish" events, while consumers "subscribe" to events of interest, and consume the events. Producers associate messages with a specific topic, and the messaging system routes messages to consumers based on the topics the consumers register interest in [8]. In Java application, message can be encapsulated as a Java object, which can wrap some possible behavior of the message data, and the delivery of the message is entitled the sense of mobile code. So far, Publish/Subscribe mechanism has been natively implemented in JMS and other middlewares.

2.2.2 Jini

Jini network technology provides a simple infrastructure for delivering services in a network and for creating spontaneous interaction between programs that use these services regardless of their hardware/software implementation [9]. Any kind of network made up of services (applications, databases, servers, devices, information systems, mobile appliances, storage, printers, etc.) and clients (requesters of services) of those services can be easily assembled, disassembled, and maintained on the network using Jini Technology. Services can be added or removed from the network, and new clients can find existing services - all without administration.

Jini builds on the concept of a federation of devices and software components. A federation is a temporary union of Jini clients and services that come together to get work done, lookup service is a full-fledged Jini service. It records information about the various services on the network. Services in Jini publish themselves by registering their proxies with one or more lookup services. A client uses the Lookup service to locate the services it needs, download the service's proxy object and start interaction with the service itself. In addition, Lookup service takes charge of bootstrapping Jini network.

3. Issues and Approaches

3.1 XML and Data Exchange

XML DTD specifies in advance the kinds of XML data elements expected and the structural relationships between them. Normally it is used to check XML document's validation, however it can be used to construct XML documents as well.

Based on a DTD, an application can easily grasp the document structure and create XML documents dynamically to communicate with another application. The receiver application that understands the same DTD can use a XML parser [10] (support DOM or SAX) program to fetch the desired information from the document for further operation. If the receiver application cannot understand the DTD of the sender application, there needs a
transformation between these two applications: a XSLT can covert a XML document from one DTD to destination DTD. This processing can be deployed either in sender part or receiver part or an independent layer between them.

This is critically useful when data exchange takes place between different companies. So far, there are plenty of standards, which are used to model the similar facts in real world to store into computers, if all the data to be exchanged are XML-based, all the standards can be remixed intact and the difficulty of interchange between them can be decreased greatly.

The main work will be to define a XSLT transformation between different standards.

XML Schema can be a good substitute of DTD, since it uses the XML syntax then when reading the structure of document, a same XML parser can be use and a extra DTD parser is not necessary, furthermore, XML Schema give more exact type description than DTD.

3.2 XML Storage

The best way to store and retrieve XML format documents is to have a database that supports XML. However, different means can be applied to the storage regarding to the type of XML documents. XML documents fall into two broad categories: data-centric and document-centric.

Data-centric documents are those where XML is used as a data transport, e.g. sales orders and invoices. In these documents the order of sibling elements is often unimportant. They are constructed from known, regular sets of data and the content of each element is subject to update independently. Document-centric documents are those characterized by irregular structure and mixed content and their physical structure is important. Examples include written text such as articles and advertisements. Normally, document-centric XML document contains static content that will only be updated by replacing the entire document.

To store and retrieve the data in data-centric documents, one needs a database that is tuned for data storage, such as a relational or object-oriented database. With relational database, the structure of the data in the document will be mapped as tables in the database. To document-centric documents, store and retrieve a single, intact XML document as a CLOB or BLOB is a good strategy, since it contains static content and is delivered from the database as a whole. However, the feature of index, query and efficient retrieve are necessary to be taken into consideration.

3.3 E-Media Multi-Channel Publishing

The reason enabling XML the universal data standard is that a XML document can be easily converted to a new one conforming to another XML document structure (DTD or Schema) after a XSLT transformation. This is significantly useful for data exchange between departments or companies supporting different data standards. When under the scenario that different client devices and programs (e.g. web browsers and wap phones) support different formats as HTML, WML, etc. a corresponding XSLT transformation has to be applied to cater to client in its understandable format. For example, a conversion from HTML to WML to enable WAP phone users to read the content of web pages. This process, from the viewpoint of message itself, gives same content different presentations conforming to the different client devices and programs. This is the essential concept of multi-channel publishing.

In a more practical way, the raw content of information is constructed as a separate XML document. Every XSLT transformation for different presentations is defined in separate XSL layout stylesheet documents. When an HTTP request reaches, the web server determines the client type through an HTTP_USER_AGENT parameter in the request.
Then it retrieves the corresponding content document, chooses the matched XSL stylesheet and applies the stylesheet on content to produce and send the transformed document to client. The whole procedure can be fulfilled on the fly without the client noticing. Different clients receive the right content and layout to display on their accessing devices.

3.4 Remoto/Local in Jini Services

In most means of implementation within object-based distributed computing architecture (CORBA or COM/DCOM), a server publishes its services as interface for access from clients. The interface will be compiled to generate stub and skeleton working for communication between client and service implementation respectively. In other words, any client requesting for service has to bind a stub with him through which to interact with server-side Skeleton on behalf of service itself.

Jini service replaces the stub with service proxy, which can be treated as mobile object traveling from service to client machine via lookup service. In addition, the proxy can be customized to carry logic from server, which is much more advantageous than Stub/Skeleton generated to only take charge of conforming network communication and calls to predefined formats. With this property Jini services can be designed in flexible ways which can be categorized roughly as types below:

- A proxy runs entirely locally on client side.
- A proxy as stub to communicate back to the logic on the service.
- A proxy that has logic on both the client and service.

With this flexibility, one can design and deploy logic freely between the client and server sides, or more widely, over the Jini network. The Jini service can be treated as a mobile-service register in the first type, every service visitor will receive integrated application logic to run locally in run-time. The second type looks like the CORBA model, however, it can help to make reuse of legacy system services an easy thing. For example, to incorporate a legacy service into Jini network, the proxy can be implemented as a wrapping of the original stub object to move to client and interact with service directly with any originally specified communication protocol. The third type gives freedom to application logic deployment to make the logic run efficiently on the machine benefiting from data retrieve or computing processing. No matter what type the service is implemented actually, the client program can be a same one and the server (service proxy and service implementation) can switch among the different types described above as needed later.

4. Integrated Application Solution

The integration of XML and Jini technology gives revolutionary change on software system architecture. In our DIP project, we apply the concepts and technologies we discussed above to achieve a distributed parallel multi-channel publishing system.

4.1 System Goals

As a completed printing and publishing solution conforming to the up-to-date field standards, our system intends to achieve support for traditional printing publishing and electronic multi-channel publishing. Other than trying to reach multi-channel access from various client devices, streamlined printing workflow and authoring and maintenance of publications will also be taken into consideration intensively. Generally speaking, our publishing system will have the following features:
4.2 Publishing Industry Specifications

**NewsML**

NewsML [11], proposed by IPTC (International Press Telecommunications Council), is a XML encoding for news. It is intended to be used for the creation, transfer, delivery and archiving of news. NewsML is media independent, and allows equally for description and representation of the evening TV news and a simple textual story. Not only specially be set out to meet the use of electronic news production, archiving and delivery, NewsML also include features for paper-based publishing and other specific production environments.

**JDF (Job Definition Format)**

JDF is an extensible XML-based interchange data format proposed by four companies prominent in printing industry, which aims to unify the pre-press, press, and post-press aspects of any printing job. JDF provides a mechanism to explicitly specify the controls needed by each process in a printing process, which may be specific to the devices that execute the processes. JDF works together with a corresponding format Job Messaging Format (JMF). JMF provides the means for production components of a JDF workflow to communicate with system controllers and administrative components. It relays information about the progress of JDF jobs and gives MIS (Management Information System) the active ability to query devices about the status of processes.

4.3 System Design

Based on the understanding of an entire publishing procedure, process can be divided generally into 2 steps: authoring and publishing. Comprehensive analyses are placed on these 2 processes to meet the demands with their different features.

4.3.1 Authoring

To accomplish good multi-channel publishing and to be able to adjust content as well as layout according to media and/or the readers expressed wishes, the news story created by the journalist needs to be extended with additional information – meta data. This meta data can be divided into two main categories: layout- and content related information.
Layout related meta data concerns information that is useful when processing information for publishing. It could be the size of a text, the format of a picture but also more administrative information such as the author's name and what company owns the material.

Content related meta data is additional information that describes what the story is about and how it is structured. We identified three levels of content related meta data that must be considered to be able to automatically adjust news content. All news stories handle one or more subjects, for example sports, domestic, political statements or celebrity gossip. The topics give a first clue on what the story is about. Then the news story is build up according to a template. A common structure could be headline, byline, abstract, sub headline, text, picture, text to the picture and the name of the person taking the picture. The third and most exciting level of meta data is the weight added to identify the importance of different parts of the story. The weight could be added to a hole story, to the different components it is build up by or to specific paragraphs of sentences in the text.

Based on these three levels of content related meta data personalized news products can be produced automatically and without the journalist having to write different stories for different purposes. However it is impossible to totally avoid additional work for the journalist since it is only he who is capable of adding the meta data. Nobody else could decide what is the most important parts in the material.

In the DIP project we have developed a news editor program that will support the journalist when writing the story and above all when adding the meta data necessary. The NewsEditor is written in Java and is based on NewsML. It allows the journalist to add both layout related meta data as well as all three levels of content related meta data.

The story is composed by journalist and reviewed by editor who decides (with a importance value attached on every story) whether or when to publish it. When the story is written and the meta data is added all information is stored within a document-centric resource data base as a NewsML document. Once stored information could be shared among organizations or distributed to the readers through publisher/Subscriber technology. (see figure 1.)

![Image](image.jpg)

**Figure 1. Publishing Materials Exchange**

### 4.3.2 Publishing Process

With publishing content ready, publishing materials are prepared to publish in different media terms. In this step, most work is focused on layout design and publishing workflow. However, due to the inherent difference between different publishing means, We divide
this step into two procedures: Paper-based printing publishing and web-based e-media publishing, so that the appropriate architectures and technologies will be applied respectively to meet their diverse demands.

4.3.2.1 Printing Publishing

Paper-based printing publishing includes typically newspaper, magazine, pamphlet and brochure, which are more printer-related, and already have mature workflow. The workflow of paper-base printing publishing can be described generally as Create, Prepress, Press and Postpress steps. And furthermore each step can be resolved: Prepress as Trapping, Impression, Color Correction; Press as Ripping, Plate-making, Printing; and Postpress as Cutting, Folding, Binding.

All these processes have been considered and fully described in JDF. To make the print job fulfilled, JDF specification designed Job Messaging Format (JMF), specifically defined to describe communication between the controller and the work centers or devices within the production system.

The four components required to create, modify, route, interpret and execute a JDF job are named as agents, controllers, devices and machines [13] in the specification, based on which we will implement our JDF-based workflow system:

The overseer of the relationships between all of the units in a workflow is Management Information Systems, a macroscopic controller. It is responsible for dictating and monitoring the execution of all of the diverse aspects of the workflow. It must remain in contact with the actual production facilities. This can be accomplished typically using JMF messaging.

Referring to concepts and description in JDF specification, we design our printing publishing workflow system based on Jini infrastructure (see figure 2.), each device providing printing service is defined as an independent service in Jini network. When a print job begins, a federation of demanding services will be constructed dynamically following a predefined publishing processing sequence:

- A Job Composer defines the publishing job before the job begins to carried out, all the jobs will be saved in Workflow Database as JDF format. Job Composer can be treated as the implementation of the agent in JDF specification.

- A Job Dispatcher queries Workflow Database to acquire the Jobs to be fulfilled, and puts them into a Job queue. Every job in the queue will be loaded to execute one by one.

- Through Jini Lookup service, Job Dispatcher finds the required service available within Jini network, and receives corresponding service proxy to run locally interacting with remote service implementation.

- During the processing period, service implementation records status switches, and sends them back to store in Workflow Database, then Job Dispatcher can monitor the status of every process from Workflow Database. Here service implementation can be thought of as Devices in JDF and service proxy Controller.

- Considering that not all the processes could be done by computers and some physical equipments have no interface for computer control automatically. Some services can be implemented as computer terminals with which operators who take charge of these processes will check the processing status and input feedback manually as the result information to be converted into JDF and JMF format to save in Workflow Database. In this case, remote service can be implemented as an information input window program.
4.3.2.2 E-Media Publishing

Web-based Publishing is innately different from traditional printing publishing as it has many special features, such as no restricted size limits, rich media terms, video and audio. However, the background requires bulk of labor work of maintenance and updating. We target to give a solution to make this exhausting work flexible and automated.

The three key processes of web publishing are Content design, Layout design and Logic design, which are in fact independent of each other, although most present solutions mix them together. We will treat them separately, since it is close to the fact that the technicians who work on a certain part of these three are hard and also not necessary to understand the technology knowledge of other two parts. Then they can just utilize their specialized skills focusing on problem solving in their domains.

In our integrated publishing solution, the materials are stored in the resource database. The database can output the NewsML format data stream, based on which, there are only two steps left, Layout design and Logic design. (see figure 3.)
The Publishing Material Database is the legacy of printing publishing solution. As described in former parts, all the content is saved as NewsML format and easy to convert to other XML-based format standards.

Layout and Logic Design are separated to fulfill, and any third-party assistant tools e.g. XSL Editor and Script Builder can be used to aid design, there is no limit on these extra extensions.

Normally, to each different output medium, a layout stylesheet file will be designed and saved separately, e.g. a same news content will have a NewsWML.xsl for mobile phone display, a NewsXHTML.xsl for web browser, and some extra stylesheets for other accessing means. These stylesheets can be used as predefined templates for all the same structured NewsML documents and the same mechanism to applying logic template.

In Information Publishing part, the e-media multi-channel publishing technology is employed as discussed in 4.3. Client’s web request is passed on by web server to a servlet. The servlet checks the client device type, retrieves corresponding content, layout and logic templates from database and applies the XSLT transformation to produce different final document to deliver to client devices for presentation. The whole process can be fulfilled on the fly. To provide a better access performance when request amount is large, the static logic and content parts can be transformed in advance to store as semi-final version at a predefined time interval, then only one XSLT-stylesheet is needed when accessing.

At present, the supporting access means include wap phones, PDAs, web browsers, PDF reader, VosML and more media are to be supported soon.

5. Conclusion and Perspective

XML is becoming the universal data standard, and Java is a platform independent network programming language. Jini infrastructure walks further more, extending the Java application "write once run everywhere" mode to "write once run from everywhere". A skilful combination of use of these two advanced technologies can empower great functionality and flexibility to software system architecture.

Distributed Parallel Multi-Channel Publishing is an integrated application of XML and Jini technology on publishing domain. Within this specific scenario, we fulfill a publication content management system consisting of authoring tool, XML encoding NewsML-based storage and multi-channel clients access, applying Java-based Jini infrastructure to trace and monitor printing workflow defined in JDF. On e-media publishing aspect, we fit in various devices for access of news publication. There are still some meaningful work to increase the value of the solution, such as to make the workflow more flexible and efficient means, a fine loading algorithm worth study, the alarm and status message should be able to send to operator’s portable devices collaborating with each other in Jini network especially when manual manipulation is urgent. Also, many concepts and approaches adopted in our project can be adapted to other scenarios. For example this solution can be adopted in a ubiquitous computing environment. Jini supporting cooperation between all the portable devices, XML-based message transferring over Jini federation. Along this road, XML-based distributed software system architecture will be a prior choice for various software applications.
6. References

http://www.w3.org/TR/2000/REC-xml-20001006

http://www.w3.org/2001/XMLSchema

http://www.w3.org/Style/XSL/

http://www.w3.org/TR/xhtml11/

[5] OMG. CORBA 2.4.2 specification, 2001-02-01
http://www.omg.org/cpsicanservices/formal/01-02-01

http://www.microsoft.com/com/


[8] Sun Microsystems. JAVVE MESSAGE SERVICE API

[9] Sun Microsystems. Java network technology
http://www.sun.com/jn/index.html

[10] Apache XML Project
http://xml.apache.org/

http://www.rtc.org/vn/NewsML/index.html


[18] IFRA, "IFRATrack in practice", describing several publishing operations in which IFRATrack helps staff meet ever-increasing demands for speed, correct quality, high service level, safe deliveries and cost efficiency, 1999.


http://www.doc.ic.ac.uk/research/ubcc.html/

