CBiX: A Model for Content-based Billing in XML Environments

by

Peter de Villiers

CBiX: A Model for Content-based Billing in XML Environments

by

Peter de Villiers

Dissertation

submitted in fulfillment
of the requirements
for the degree

Magister Technologiae

in

Information Technology

in the

Faculty of Computer Studies

of the

Port Elizabeth Technikon

Promoter: Prof. Reinhardt A. Botha

December 2003

Declaration

I, Peter de Villiers, hereby declare that:

- The work in this dissertation is my own work.
- All sources used or referred to have been documented and recognized.
- This dissertation has not previously been submitted in full or partial fulfillment of the requirements for an equivalent or higher qualification at any other recognized education institute.

Peter de Villiers

Abstract

The new global economy is based on knowledge and information. Furthermore, the Internet is facilitating new forms of revenue generation of which one recognized potential source is content delivery over the Internet. One aspect that is critical to ensuring a content-based revenue stream is billing. While there are a number of content-based billing systems commercially available, as far as can be determined these products are not based on a common model that can ensure interoperability and communication between the billing systems.

This dissertation addresses the need for a content-based billing model by developing the CBiX (**C**ontent-based **B**illing **i**n **X**ML Environments) model. This model, developed in a phased approach as a family of billing models, incorporates three aspects. The first aspect is access control. The second aspect is pricing, in the form of document, element and inherited element level pricing for content. The third aspect is XML as the platform for information exchange.

The nature of the Internet facilitates information interchange, flexible web business models and flexible pricing. These facts, coupled with CBiX being concerned with billing for content over the Internet, leads to a number of decisions regarding the model:

- The CBiX model has to incorporate flexible pricing. Therefore pricing is evolved through the development of the family of models from document level pricing to element level pricing to inherited element level pricing.

- The CBiX model has to be based on a platform for information interchange that enables content delivery. XML provides a broad family of standards that is widely supported and creating the next generation

Internet. XML is therefore selected as the environment for information exchange for CBiX.

- The CBiX model requires a form of access control that can provide access to content based on user properties. Credential-based Access Control is therefore selected as the method of access control for CBiX, whereby authorization is granted based on a set of user credentials.

Furthermore, this dissertation reports on the development of a prototype. This serves a dual purpose: firstly, to assist the author in understanding the technologies and principles involved; secondly, to illustrate CBiX_0 and therefore present a proof-of-concept of at least the base model.

The CBiX model provides a base to guide and assist developers with regards to the issues involved with developing a billing system for XML-based environments.

Acknowledgements

I would like to thank our heavenly Father for the blessings he has bestowed on me, for guiding me and helping me every step of the way. I would further like to thank the following people, without who this research undertaking would have been a long, lonely road:

- My parents, for their love, support and encouragement.
- Jacqueline, for her love and encouragement, keeping me motivated, and for being there for me when the long hours were needed.
- Greg, for his support and help proofreading the academic papers and dissertation.
- My promoter, Professor Reinhardt Botha, for his guidance, advice, support and encouragement.
- The Port Elizabeth Technikon for support and financial assistance.

The financial assistance of the Department of Labour (DoL) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the DoL.

Contents

List of Figures

Part I

Background

Chapter 1

Introduction

The global economy has made a transition from an economy with an industrial focus to being an economy based on knowledge and information. This new paradigm is enabled by the use of Information and Communication Technologies (ICT).

The increasing pace of technological innovations in the field of ICT has given rise to new ways of communicating, learning and conducting business. The Internet has facilitated the establishment of a "borderless" environment for communications and the electronic delivery of certain services. This is known as electronic business (e-business). Electronic business is the key to doing business in the new global economy that is based on knowledge and information (South African Department of Communications, 2000).

E-business applications include e-commerce, e-learning, gaming, online communities and TV-quality streaming media for communication with employees and partners. All of these applications are enabled through the web (Cisco World, 2000).

Web pages are created with the Hypertext Markup Language (HTML) that tells the browser how to display the information. HTML provides a simple, common language to create web pages that could easily be viewed by just about anyone more or less as intended. The problem with HTML is that it is a limited language, and you have trouble effectively separating content and presentation (Hampton, 2002).

Presentation and content is effectively separated by the Extensible Markup Language (XML) (W3C, 2002a), which describes the information itself. By encoding information in a standardized way XML can help make the web much more useful and versatile. For example, if you have financial assets scattered across a range of companies, it is possible to access this information at each companies web site, display it on your PC and perform transactions, but it is virtually impossible to consolidate all this information into a single spreadsheet on your PC. With XML this is possible. XML thus promises to transform the structure of the web, replacing HTML with a stronger more flexible and extensible architecture by separating presentation and content (Nokia, 2000).

"Content is the defining essential. Those with content will set the pace." says Sandy Climan, Managing Director, Entertainment Media Ventures (Nokia, 2002).

This kind of realization led to there being a change in emphasis to not only provide traditional Internet transactions and services, but content-based services as well. This change coupled with the Third Generation (3G) broadband wireless networks, will cause the telecommunications operator's position in the value chain to change.

The introduction of content-based services offers telecommunications operators the opportunity to span a much broader section of the value chain than before. Establishing a mobile portal and active partnerships with content providers, internet services providers, application service providers, and financial institutions will be crucial for operators wanting to increase profits in 3G networks (Nokia, 2002).

This study is therefore primarily motivated by the realization that content is an important source of revenue.

1.1 Motivation for this study

In addition to the aforementioned comments, several other realizations motivate and support this study.

The realization that billing is critical to increasing revenues

The key to increased revenues is the introduction of content-based services. With the introduction of these services telecommunications operators will have the opportunity to span a much broader section of the value chain. In order to do this effectively they will have to make effective use of their existing assets, such as their subscriber base. They will also have to form new partnerships in order to create attractive applications and provide quality content. However, the key to increasing revenues will be implementing efficient billing systems (Nokia, 2002).

Part of an efficient billing system is effective access control to the system. This lead to the following realization.

The realization that billing and access control share common ground

Billing and access control both essentially consist of three phases. The first phase in both instances is identification and authentication of the client. In both cases the next phase is access or an attempt to access the service. The third phase, however, differs slightly. In billing the next phase consists of the actual monetary transaction or payment for the received content/service. In traditional access control the third phase consists of either permission being granted or denied for the requested service (Sandhu, Coyne, Fenstein, & Youman, 1996).

Furthermore, Bullock and Benford (1999) identify three properties that the implementation of an access control service should adhere to. The access control service should be unobtrusive, simple and maintainable. The author contends that these three principles also apply to a billing system.

Billing and access control therefore share common ground. Both require identification and authentication and determination of access to the requested service. Access control and billing differ in the final phase with billing focussing on payment and traditional access control on permission or denial. Both billing and access control systems should be unobtrusive to the user. However, due to a billing system requiring a financial transaction it will require a certain amount of user interaction. The remainder of the operation of the billing system such as the determination of which content a user may access, should be unobtrusive. Furthermore, both an access control service and a billing system must be simple, and easy to maintain.

Because of the above mentioned requirements of simplicity and ease of maintenance a billing system would be most effective on a platform with a flexible and extensible architecture.

The realization that XML provides a flexible and extensible architecture

Hypertext Markup Language (HTML) is the language for publishing hypertext on the World Wide Web. It is a non-proprietary format based upon Standard Generalised Markup Language (SGML), and can be created and processed by a wide range of tools. HTML uses tags to structure text.

The next generation HTML, eXtensible Hypertext Markup Language (XHTML) (Pemberton, Altheim, et al., 2001) is a family of document types. These document types are eXtensible Markup Language (XML) (Bray, Paoli, Sperberg-McQueen, & Maler, 2000) based, and are ultimately designed to work in conjunction with XML based user agents. XHTML documents are XML conforming. As such they can easily be viewed, edited and validated with standard XML tools. XHTML documents also have the ability to utilize applications based on either the HTML Document Object Model (DOM) or the XML DOM (Le Hgaret et al., 2002).

Because XML separates content and presentation, an XML file doesn't contain any presentation information. The presentation information is contained in a style sheet. This allows the same document to be displayed on different media, and it also enables users to view the document according to their preferences and abilities, just by modifying the style sheet. The stylesheet is created using The Extensible Stylesheet Language (XSL) (Froumentin, 2002a). XSL uses XML notation. In XSL, the formatting object tree can be radically different from the source tree, and inheritance of formatting properties is on the formatting object tree. XSL is targeted at XML, in particular highly-structured, data-rich documents that require extensive formatting (W3C, 2002c).

Therefore the web is an environment that XML is helping to make more flexible and extensible by promoting compatibility (W3C, 2002b). This has a great potential impact on the next generation web.

The realization that XML has a great potential impact on the next generation web

XML is often used for structured documents such as XHTML, rendering or transformation languages like Extensible Stylesheet Language Transformations (XSLT) (Clark, 1999), and as the basis for extensible network protocols like the Simple Object Access Protocol (SOAP) (Gudgin, Hadley, Moreau, & Nielsen, 2001).

Another example of the use of XML in network protocol design is Composite Capabilities/Preference Profiles (CC/PP) (Ohto & Hjelm, 1999). One important characteristic of the next generation Web is the proliferation of types of client access device, often by the same user, allowing people to access email with a desktop computer, a laptop computer, a handheld device, or a Wireless Application Protocol (WAP) (Hjelm, Martin, & King, 1998) enabled cell phone. Each of these devices has a different set of capabilities. To help Web servers keep pace with heterogenous clients, the CC/PP protocol provides a way for clients and servers to exchange information about the client's capabilities and preferences.

The CC/PP working group chose the Resource Description Framework (RDF) (Miller, Swick, & Brickley, 2000) to describe device capabilities. A RDF description is an XML document containing a description of the client's capabilities. The RDF description provides a framework within which diverse clients can tell servers exactly what their capabilities are. This framework allows servers to adapt to a client's needs, thus creating a more dynamic web (Martin, 2002).

Therefore XML is key to creating a more dynamic web through various standards that are using its structured nature as a basis.

The realization that the structured nature of XML facilitates access control

XML imposes strict rules to the structure of XML documents. An XML document that adheres to these rules is said to be well-formed. In addition, the structure of XML documents may be specified by an XML Schema (Fallside, 2001). These facts already ensure a greater level of integrity in XML documents than HTML documents.

The separation of content and presentation in XML documents yields further security opportunities. Because of the separation of content and presentation focus can be given to protecting the information content without being concerned with the presentation of information. Since the tags that define the contents can be created by the author of the document, specific semantics can be associated with each tag that can be used to interpret the document and its security requirements (Botha, 2001).

For example, suppose there is an online catalog document written in XML that lists available goods sold on the Internet. Because the content is separated from the presentation, access control policies could be applied to the information. One access control policy could be that only premium members can view the special discount price information in the document. When a regular member views the catalog, any information provided for the premium members would be hidden. Access control whereby access is only allowed to certain parts of a document for certain users, is called fine-grained access control. Fine-grained access control is made possible by the fact that the tags defining the document are defined by the author of the document (Damiani, De Capitani di Vimercati, Paraboschi, & Samarati, 2002) .

XML has a flexible and extensible architecture. It has a great potential impact on the next generation web. XML's structured nature also facilitates access control. Access control and billing share common ground and billing is key to increasing revenues. This prompted an investigation into existing billing products, leading to the following realization.

The realization that existing products are proprietary

Software such as Apogee Networks' NetCountant (Apogee Networks, Inc., 2002b) software is available for content-based Internet billing. Using this for an e-zine, for example, can charge users for accessing certain stories, without requiring a monthly subscription fee.

Current content-based billing such as this works very well in cases where the information is structured as in the case of an e-zine where it is easy to divide content into separate stories. Financial information is another example of information that is already highly structured and easily broken into smaller billable sections (Emigh, 2000).

Information that is based on already existing information such as a story/article or financial information is generally more structured and specific. Specific information is generally better catered for by existing billing software, while unstructured information such as dynamically generated information is much harder to handle.

There are a host of other products available with similar features to Apogee Networks' NetCountant such as TeleKnowledges Total-e (Allot, 2000), Nokia's Charging Center and Charging Gateway (Nokia, 2001) and Pinpoint Networks' Fuel platform (Pinpoint Networks, 2002).

Although all these products provide billing solutions for varying content, they were, as far as could readily be established, aimed at specific information types and of a proprietary nature. As such, these billing solutions are not able to easily communicate amongst each other.

These realizations provided motivation for undertaking the proposed research.

1.2 Problem Statement

There are a number of products available that provide for billing. However, these products are not based on a model that is based on a standardized environment that can enable information interchange between solutions developed by different developers. Furthermore, as far as can be determined these products do not facilitate access control that can allow a user access to content over the Internet based on user properties that can be identified as relevant for security purposes. Upon investigation it was realized that a problem exists in that there is no generic content-based billing model available for XML environments that can provide access based on user properties.

To address this problem a number of sub-problems first have to be investigated:

- What is the importance of content-based billing?
- How is semi-structured information exposed in XML environments?
- What is effective access control for content-based billing?
- How is access control done in XML environments?

1.3 Objective

The objective of this study is to define a model for content-based billing within XML environments. This model must be developed in such a way that it can then be used as a basis for designing flexible and secure content-based billing within any environment based on the XML standard.

As part of the process of developing the model a number of sub-objectives must be addressed:

- Establish the need for and elements of content-based billing.
- A method to expose semi-structured information in XML environments must be identified.
- The position of access control in a billing model must be clarified and the relationship between access control and billing must be established.
- Effective access control for XML environments must be identified.

1.4 Methodology

The primary methodology followed in undertaking this research is the development of a model, supported by a thorough literature study.

The literature study will be used to form the basis of the background chapters. Firstly, the literature study focusses on web business models and the value of content to determine the importance of content-based billing and clarify aspects of billing. Secondly, the literature study focusses on XML, related standards, and the workings of XML, to determine how to expose semi-structured information in XML environments. Thirdly, the literature study focusses on access control to determine the relationship between access control and billing, to position access control within a billing model, and provide secure access control for a billing system using XML as an environment.

Following the literature study a model for content-based billing in XML is developed and formally defined. The formal notation selected in this dissertation is the Z notation. To assist in the formal definition of the content-based billing model for XML environments, a prototype is developed. The prototype is developed to clarify concepts, technologies and principles related to

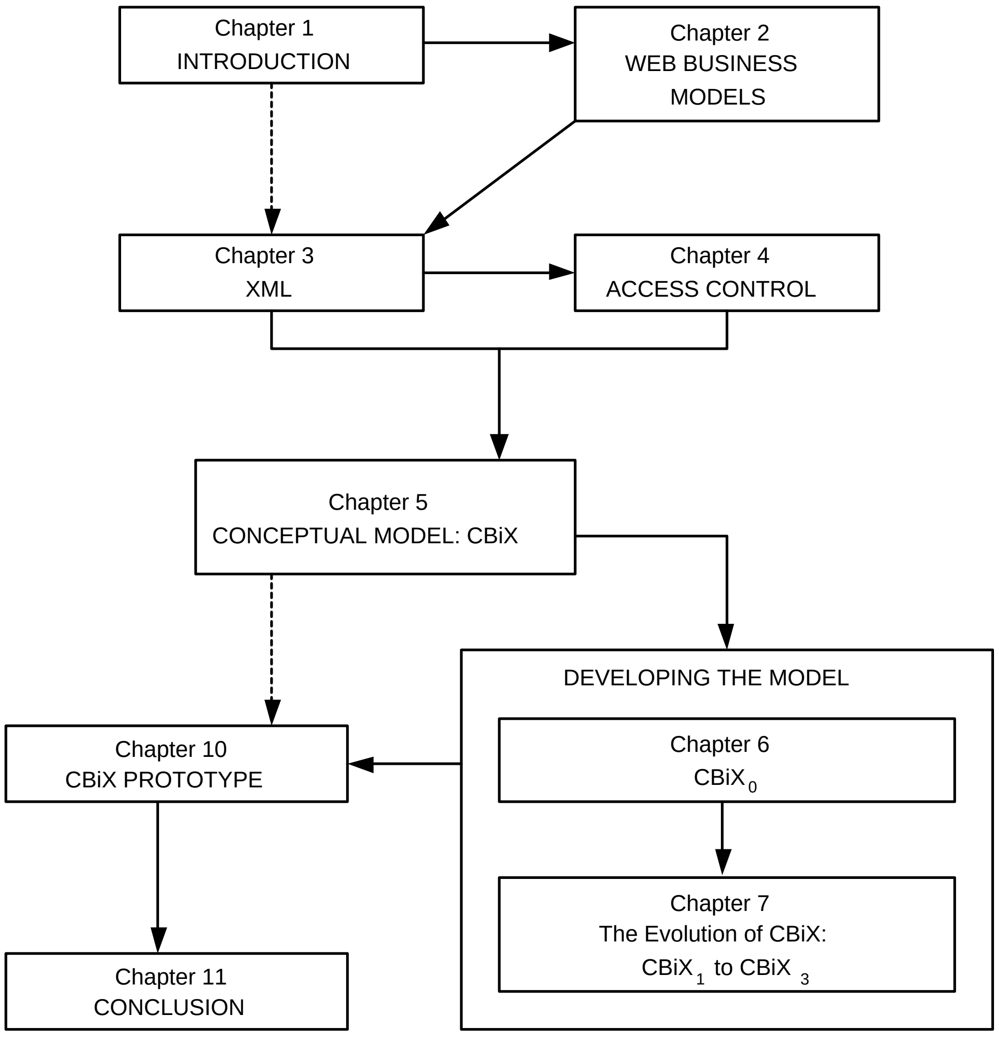


Figure 1.1: Lay-out of dissertation

a content-based billing model in XML environments, as well as to provide proof-of-concept for the content-based billing model that has been developed.

The results of this research are reported in the form of this dissertation. The next section describes the layout of this dissertation.

1.5 Layout of the Dissertation

The lay-out of the dissertation is depicted in figure 1.1.

Chapter 2 investigates the importance of content for generating revenue, followed by an investigation into the various business models for the Internet. Next, this chapter discusses various price strategies and maps them to the mentioned web business models. Following an investigation of the pricing strategies, the need for a common environment will be established.

Chapter3 provides specific motivation for the choice of XML as a common environment for information delivery on the Internet. Chapter 3 also discusses some related standards that are XML-based. These standards are key to enhancing the power and flexibility of XML as an environment for information delivery. A common environment for information delivery on the Internet that is facilitated by XML, is key to fulfilling CBiX's requirements of power and flexibility.

Chapter 4 identifies three common forms of access control. Chapter 4 then focusses on the specific needs of CBiX and investigates credential-based access control as a means of access control for CBiX . The technologies that are available to implement credential-based access control in a billing system for XML environments are then investigated. Lastly access control in XML is investigated as it has been selected as the common environment for CBiX.

Chapter 5 delimits the scope of the research and makes note of issues that are not addressed by the CBiX billing model. Following the establishment of the scope of CBiX this chapter establishes the conceptual foundation of CBiX. Next this chapter provides a conceptual overview of the development cycle of the CBiX billing model as well as describing a common architecture to support CBiX.

Chapters 6 and 7 are dedicated to the development of the content-based billing model for XML environments. Chapter 6 formally defines the first in the CBiX family of models, CBiX_0, the base model. CBiX_0 is used by the rest of the CBiX family of models as a basis for further development. Chapter 7 then extends the base model by defining the rest of the family of content-based billing models. Chapter 7 firstly defines CBiX_1 which builds on the functionality of CBiX_0 by adding inheritance. Chapter 7 then develops CBiX_2 in parallel to CBiX_1. CBiX_2 enhances the CBiX family of billing models by developing the document level pricing defined in CBiX_0 into element level pricing. Chapter 7 lastly defines CBiX_3, the comprehensive model, which combines the features of CBiX_1 and CBiX_2 into a comprehensive billing model.

Chapter 8 reports on the development of a prototype. The prototype serves a dual purpose: firstly, it assists the author in understanding the technologies and principles involved; secondly, it serves to illustrate CBiX_0 and therefore represents a proof-of-concept of at least the base model.

Chapter 9 concludes this dissertation and proposes further research.

Chapter 2

Web Business Models

The Internet has played a leading role in enabling the new global economy based on knowledge and information. The Internet has also enabled the generation of new revenue streams. The importance of the Internet in generating revenue, coupled with the realization that billing is critical to increasing revenues has led to this chapter firstly investigating the history of commerce on the Internet. Once it has been established where we are today with regards to commerce on the Internet, the importance of content for generating revenue will be investigated. Following this will be an investigation into the various business models for the Internet. Next, this chapter discusses various price strategies and maps them to the mentioned web business models. Following an investigation of the pricing strategies, the need for a common environment will be established. Part of the investigation into a common environment delves into existing billing systems, followed by discussion of UDDI (Universal Description, Discovery, and Integration) (Bellwood, Clment, & Riegen, 2003) to show the importance of a common environment.

2.1 The History of Commerce on the Internet

To date there have been four phases of commerce on the Internet as depicted in figure 2.1 on the following page, starting with brochureware and ending in e-Enterprise. Although each of these phases has a beginning none of them has an ending. Even today some companies may be performing brochureware

Phase 1	Brochure ware	Companies realize the potential of the Internet. Sites with static documents and simple multimedia are established, providing customers with company and product information.
Phase 2	e-Commerce	Business to Consumer web models are developed. This leads to buying and selling on the Internet as well as provision for online bill payment. Customer care and management functionality is offered over the Internet.
Phase 3	e-Business	Business to Business transactions take priority. e-Applications are developed to facilitate communication between business partners.
Phase 4	e-Enterprise	Convergence between e-Commerce and e-Business. Companies are using the Internet to extend the valus chain and combine their brick and mortar assets with the speed of the Internet.

Figure 2.1: The Four Phases of Commerce on the Internet

over the Internet while others are full-fledged e-Enterprises.

The first phase of commerce, brochureware, began with companies realizing the value of the Internet as a medium to reach out to their customers. This led to the Internet becoming the key medium for global marketing that provided a low-cost repository of product and service information. According to Forrester Research by the end of 1999 34 percent of Fortune 500 companies had a web presence (Hoque, 2000). A year later this had risen to over 80 percent. Most of the sites in this phase consist of static documents with simple multimedia, with customer interaction consisting of reading the information and viewing the images.

The next phase is the e-Commerce phase. This phase saw the development of the new B-to-C (Business to Consumer) business models for the buying and selling to consumers online. This led to the development of e-Tailers such as Amazon.com and Auction sites such as eBay. Companies also started offering Customer Care and Customer Management abilities over the Internet. Other companies such as AT&T allowed customers to pay their bills on-line. This phase was characterized by companies striving to create a brand name for its .com initiative, to market to each consumer individually, to personalize the information and transactions for each consumer, and to create a community environment that welcomes back consumers repeatedly (Hoque, 2000).

Initiatives in the following phase, e-Business, have focussed on business-to-business (B2B) applications that allow transactions and interaction between the company and its business customers and partners over the Internet. Within this phase the development of new e-Application categories and functionalities are developed such as Virtual Marketplaces, Procurement and Resource Management, Extended Value Chain, and Customer Relationship Management. The e-Business phase is characterized by initiatives that are proactively focussed on the organization's core competencies and whose business model is orientated toward process aggregation. For ventures in the phase to be successful the organization must not only have agile applications and the ability to integrate data and applications, but the organization must also be structured in a way to support the business model of the e-Business initiative (Kerrigan, Roegner, Swinford, & Zawada, 2002; Sairamesh, Mohan, Kumar, Hasson, & Bender, 2002).

The final phase is the e-Enterprise. This phase is characterized by a convergence between e-Commerce and e-Business as companies find that their markets contain both B2C and B2B segments. These companies are implementing common functionalities across different e-Application categories in order to serve both market segments. In e-Enterprise, the whole value chain – from procurement on the supply side to consumer retailing and customer management on the demand side – happens by combining traditional bricks-and-mortar assets with the efficiency of cybermediation. This phase allows companies to use their Internet business models to leverage their existing asset base to become more agile and effective.

As all these phases evolve they are allowing companies to use technology to reengineer themselves rather than simply automate business processes. With the emergence of the World Wide Web and standards such as Secure Electronic Transaction (SET) and the eXtensible Markup Language (XML), managers are beginning to turn their attention to reengineering business processes to unite buyers, suppliers, and trading partners in dynamic, real-time information sharing partnerships. This allows companies to move to a customer-centric business model by placing the customer in control. The result is a fundamental shift in the focus of the business from optimizing and refining internal processes and strategies to refocussing the core of the enterprise outward to business partners (Hoque, 2000).

One major shift in business strategies is the realization that content has an important role to play in increasing revenues.

2.2 The Importance of Content

"Content is the defining essential. Those with content will set the pace." says Sandy Climan, Managing Director, Entertainment Media Ventures (Nokia, 2002). IBM has also stated that content is your biggest asset and that by improving on the distribution and management of your content you can add value to your assets (IBM, 2002).

Cisco Systems has also realized the importance of content, and has therefore formed the Content Alliance to push the market evolution of content delivery systems (Cisco World, 2000). This is achieved by developing open standards and protocols to advance content networking and deliver key technologies for a number of emerging areas, including content peering. A critical technology for the widespread adoption of content delivery services, content peering permits the CDNs (content delivery networks) of multiple service providers to work in cooperation. Greg Howard, principal analyst at The HTRC Group has said that interoperability will play a key role for successful content peering, which is an essential next step in order for content providers to work together to offer a greater Content Delivery Network footprint. CDNs enable enterprises and dot.coms to roll out accelerated e-business applications including e-commerce, e-learning, gaming, online communities and TV-quality streaming media for communication with employees and partners. CDNs also allow service providers to distribute content closer to the end user and overcome issues such as network bandwidth availability, distance or latency obstacles, origin server scalability, and congestion issues during peak usage periods. Content delivery solutions will increase the value of customers' networks by enabling them to offer new interoperable and profitable value-added services (Cisco World, 2000).

These kinds of realizations and initiatives have led to there being a change in emphasis to not only provide traditional Internet transactions and services, but content-based services as well. This change coupled with the Third Generation (3G) broadband wireless networks, will cause the telecommunications operator's position in the value chain to change.

The introduction of content-based services offers operators the opportunity to span a much broader section of the value chain than before. Establishing mobile portals and active partnerships with content providers, internet services providers, application service providers, and financial institutions will be crucial for operators wanting to increase profits in 3G networks. However, the key to increasing revenues will be implementing efficient billing systems (Nokia, 2002).

Today's market leaders require a complete, end-to-end billing and settlement solution that enables them to accelerate their speed-to-market, product differentiation, and realize high revenue gain and profit margins. Service providers need real-time billing and settlement solutions that are capable of identifying, defining, measuring, rating, processing, billing, and settling billions of events per day. The billing system must be able to support the convergence of different service offerings into one invoice. (Apogee Networks, Inc., 2002b)

There are many different web business models that can be adopted by a company, and there are also many different pricing strategies that the business can follow. The choices of the business in this respect has to form part of an effective billing system.

2.3 Web Business Models

Revenue justifications allow a business to reach new customers, expand existing partnerships or build new ones, and expose existing offerings to new delivery channels. This category of adoption motivators are the sweet spot for potential service providers (Gebauer, 2000). Figure 2.2 on the next page shows a mapping of the Web Business Models that are discussed below to the phases of commerce on the Internet. Service Providers are e-Enterprises that perform an electronic service such as providing content to customers.

From figure 2.2 it can be determined that there are five potential revenue models for Service Providers: transactional, membership/subscription, lease/license, business partnership, and registration. However, not all of the potential revenue models are suitable for every phase of commerce on the Internet, as indicated by the mapping of the revenue models to the phases of commerce in figure 2.2. Due to the nature of brochureware, none of these potential revenue models apply to this phase.

The first of the potential revenue models is the **transactional model**. The transactional model refers to the pay-per-click or fee-for-use model. It

		Phases of Commerce on the Net			
		Brochureware	e-Commerce	e-Business	e-Enterprise
Web Business Models	The Transactional Models		Yes		Yes
	The Membership or Subscription Model		Yes		Yes
	The Lease or License Model			Yes	Yes
	The Business Partnership Model			Yes	Yes
	The Registration Model		Yes		Yes

Figure 2.2: The Four Phases of Commerce on the Internet mapped to Web Business Models

is the most primitive of all the models. Once a business relationship exists between two trading partners (a service requestor and a service provider), the provider needs to determine how it will obtain agreement on terms for usage of the service. One possible method is the pay-as-you-go approach. Here the notion of a charge per transaction is possible using payment instruments like credit cards. Now it is possible for Web services providers to address this revenue method via the service interface they provide, but that is a business consideration that must be made at design time. Another possible method for establishing terms for usage is that of the out-of-band relationship. Simply stated, the terms of the business relationship are agreed upon prior to use of the service. Since the technology of e-contracts is not yet mature, a dependency on an established business relationship prior to the use of the service must exist. Using this method, the service provider needs to just audit the usage of a service and bill for it on a periodic basis. In either case, the two trading partners must establish a mutual agreement associated with the fee for services rendered. The transactional model is a relatively primitive and straightforward model and as indicated on figure 2.2 it can be used in the e-commerce phase as it is not ideally suited to the business to business type of commerce found in the e-business phase. As the e-enterprise phase is the convergence between e-commerce and e-business, the transactional model can also apply to the e-enterprise (Gisolfi, 2002; Rappa, 2002; Gebauer, 2000).

The second potential revenue model is the **membership or subscription model**. This revenue model pertains to the establishment of an user account with specific terms for usage. A user may register for usage based on a period regardless of quantity or based on quantity alone. A service provider may create tiers of membership levels to allow for the targeting of specific classifications of users. Similar to the transactional model, the service provider must decide whether its service interface will deal with the aspects of managing this revenue model or if it will manage their revenue through out-of-band relationships. Figure 2.2 depicts that just as with the transactional model, the membership or subscription revenue model is also applicable to both the e-commerce and e-enterprise phases (Gebauer, 2000; Gisolfi, 2002; Rappa, 2002).

The third potential revenue model is the **lease or license model**. This revenue model would be common amongst larger business partners who would require large volumes of usage and expect a more customized agreement. Here, a service provider may charge by volume of transactions or maybe the volume of requesting "components" (seats) within the service requestor. In this case an out-of-band relationship is a given assumption. As indicated on figure 2.2 this model will occur in the e-business phase as well as the e-enterprise phase as the lease or license model is aimed at business partners that require large volumes and greater customization (Gisolfi, 2002; Rappa, 2002).

The fourth potential revenue model is the **business partnership model** This revenue model is a new concept brought about by the dot.com hype in 2000. Essentially, this model refers to the establishment of terms through an out-of-band relationship or prior usage-based system, on the bartering of services, equity, or even a percentage of gross revenue of the requestor. As indicated on figure 2.2 on page 20 this revenue model, as its very name suggests, is most applicable to the e-business and e-enterprise phases (Sairamesh et al., 2002).

The final potential revenue model introduced in this chapter is the **registration model**. This revenue model would apply more readily to a Yellow Pages type of business. Here the concept of collecting revenue based on a pay-to-be-seen concept is feasible. The premise stands that if a service provider wants to be published they will be willing to pay a registration fee. As indicated on figure 2.2 the registration model will be associated with the e-commerce phase as this is not a revenue model for large business partners with high transaction volumes. The registration model will therefore also apply to the e-enterprise phase.

A model for content-based billing would be different from the above mentioned business models, however it will incorporate some of the aspects of these models such as the possibility of creating pricing tiers for various subscribers (Gisolfi, 2002; Rappa, 2002).

A major factor in the success of any business model is the ability for the business implementing that model to interact with its suppliers, customers and trading partners. Another important facet for a business model is the pricing strategy of that model.

2.4 Pricing

There are two major trends in electronic commerce that are causing a shift from fixed to dynamic pricing. First, the Internet has reduced the transaction costs associated with dynamic pricing. This has primarily been achieved by eliminating the need for people to be physically present in time and space to participate in a market as well as considerably reducing the menu costs. In the physical world changing a price incurs huge costs, while the same task in electronic commerce is reduced to a database update. Second, price uncertainty and demand volatility have risen and the Internet has increased the number of customers, competitors, and the amount and timeliness of information. In addition, the increased use of flexible pricing itself leads to increased price uncertainty. As a result of these trends, businesses are finding that using a single fixed price in volatile Internet markets is often ineffective and inefficient (Bichler et al., 2002).

Because the Internet has enabled an increasing shift towards dynamic pricing many different pricing strategies have emerged. Four pricing strategies that can be identified from Chang and Petr (2001) and Odlyzko (2001) are Flat-rate, Usage-based, Session-based and Per incident.

2.4.1 Flat-rate strategy

Flat-rate is currently the most common approach for a paid content site. In this strategy the user pays a flat fee on a monthly or annual basis. This allows the user to then download as much content as he wishes. A flat-rate strategy is often used to capture market share as it is very attractive to customers. In the long run this strategy does not generate the most revenues for the content provider as it allows users unlimited usage driving up the content providers costs without increasing revenues (Odlyzko, 2001; Chang & Petr, 2001).

2.4.2 Usage-based strategy

Usage-based is the most common alternative to a flat-rate strategy. In this strategy the user is billed according to usage of the service. Usage can be defined as either an amount of time or traffic volume. For a content provider the usage-based strategy can be applied to the volume of the content downloaded. The content can be charged per unit for example an article or image can be classified as a unit with an attached price. Charging can be performed as a combination of flat-rate and per unit. An example of this would be to allow a certain amount of downloads at a flat-rate, and once this limit is exceeded charging per unit. Charging can also be performed on a tiered basis with the first article costing X, the following article costing Y, the next Z, and so forth (Odlyzko, 2001; Chang & Petr, 2001).

2.4.3 Session-based strategy

A session-based strategy is more logical for streaming media and/or special Internet events. Rather than being charged for a given amount of time or volume, the customer is charged for a single session that may vary in length and volume depending on the particular content stream. Again, depending on the content providers business model, customers may be able to buy several sessions for a single price, and then purchase additional individual sessions for another price. Also, several executions of the same Internet event may not incur additional charges for a specified duration depending on the content provider's business model (Odlyzko, 2001; Chang & Petr, 2001).

2.4.4 Per incident strategy

With a per incident strategy, the customer is charged based on the occurrence of a specific incident or user action such as clicking on a particular hyperlink. This approach is particularly applicable to content purchases such as MP3s, online articles or e-books. Once again, depending on the content provider's business model, customers may be offered multiple incidents for one flat rate with additional incidents at an additional charge or they may be offered discounts on their nth incident. This approach is also used for measuring clicks per million (CPM) for advertisements (Chang & Petr, 2001).

There are many parameters that can affect a content or service provider's price strategy. It may be that a premium level of service is available for a certain price, or for users of other linked services. Sometimes certain pricing may apply to users who sign up by a certain date, while those who sign up later pay a different price. Sometimes geography may enter into the equation. Sometimes an individual users affiliation with a corporation or other organization may affect pricing (Chang & Petr, 2001; Odlyzko, 2001).

To be successful and aggressive in providing content-based services the ability to rapidly activate and de-activate pricing schemes is crucial. This requires a billing system that allows new pricing strategies to be implemented in near real-time (Apogee Networks, Inc., 2002a).

Figure 2.3 on the next page shows a mapping of the discussed pricing structures to the web business models that were discussed in the previous section. This illustrates that for most web business models more than one pricing strategy can be followed. This allows the company to choose the pricing strategy that will generate the most revenue for its business model. Furthermore, it can be observed from the mapping in figure 2.3 that the usage-based pricing strategy is the most common pricing strategy and can be applied to all but the registration web business model. The mapping also illustrates that all the pricing strategies can be applied to both the e-commerce and e-business phases of commerce on the Internet. Therefore, all the pricing strategies are also applicable to the e-enterprise phase. Figure 2.3 also shows the pricing strategies mapped to the CBiX billing model. For the CBiX billing model the most appropriate pricing strategies to support, exposing the CBiX model to a wide range of web business models, will be the flat-rate and usage-based pricing strategies. However, making provision for all the pricing strategies discussed would allow a business using any of the discussed business models to utilize the CBiX model.

2.5 The Need for a common environment

Flexible pricing strategies, and the ability to cater for various web business models are crucial components of a powerful and flexible billing system. One of the features of the Internet that is enabling the introduction of flexible pricing strategies and the ability to cater for various web business models is the amount and timeliness of information (Bichler et al., 2002). The amount of information used by businesses to enable their pricing strategies and web business models requires communication between business partners. Communication is greatly enhanced and made more efficient when enabled by a

		Pricing Strategies			
		Flat-rate	Usage-based	Session-based	Per Incident
Web Business Models	The Transactional Models		Yes		Yes
	The Membership or Subscription Model	Yes	Yes		
	The Lease or License Model		Yes	Yes	
	The Business Partnership Model	Yes	Yes	Yes	Yes
	The Registration Model				Yes
CBiX	Content-based Billing Model	Yes	Yes		

Figure 2.3: Pricing Strategies mapped to Web Business Models

common environment. The need for a common environment has been recognized by industry leaders such as Ariba, Microsoft and IBM.

These companies consider a common environment to be so important that they have joined with 36 other technology companies to create a global Internet standard designed to accelerate e-commerce. This is called Universal Description, Discovery, and Integration (UDDI) (Bellwood et al., 2003).

UDDI is open, platform-neutral, and is based on XML. Vice President of Product Marketing for Ariba, Nick Solinger has said that the fact that UDDI is XML-based means that it is grounded on practical technology that can enable it to become part of the internet infrastructure.

John Swainson, IBM General Manager and UDDI spokesperson, has said that the UDDI project is an example of the inclusiveness and cooperation needed to drive greater efficiencies in e-business transactions and integration through open standards. He has also said that without universal communication and information sharing there is no growth (Gisolfi, 2002).

Standards such as UDDI are being developed to allow companies to communicate more efficiently.

Because of the emphasis by industry leaders on the need for a common environment for information interchange this dissertation is focussing on the development of a standard content-based billing model that promotes the use of a common environment. Developers can then use this model as a basis for development being assured that individual products of business partners can correctly communicate billing information.

2.6 Conclusion

Commerce on the Internet has changed over the past few years, shifting from brochureware to e-Enterprise. This shift in focus has led to companies reengineering themselves to make the most of the available Internet technologies to not only improve automated business processes, but to develop new business models to use their existing assets and create partnerships to create new streams of revenues.

One major new stream of revenue can be found in delivering content to consumers. To effectively generate revenue from content will require the implementation of efficient billing systems. However, at present there is no standard model for the implementation of a billing system for content. As part of development of the proposed model existing web business models were investigated.

Revenue justifications are the primary reason for business to explore new business models and reengineer their existing business model if necessary. There are five easily identified business models. These are transactional, membership/subscription, lease/license, business partnership, and registration.

Various pricing strategies can be implemented for content and service billing. The pricing strategy that is chosen by the business would depend on the business model followed. A model for content-based billing should cater for all the pricing strategies, allowing a business to tailor their billing system to their billing model. This would result in a highly complex pricing component to the model, requiring extensive research out of the scope of this dissertation. Instead, the proposed model will only support a flat-rate strategy and a usage-based strategy. In the flat rate strategy a user pays a flat price for a document or element containing content. In a usage-based strategy the user saves by having inherited element level pricing and paying a reduced price for purchasing a content element and its child elements.

A need of all web business models is cooperation and information sharing between all the concerned parties. Extended value chains further emphasize the need for cooperation and communication between all business parties.

The development of UDDI to facilitate cooperation and communication between business parties also illustrates the importance of a common environment. The basis for the development of the UDDI standard is XML.

Because of the need for a common environment and because of XML being used as a platform for the development of standards such as UDDI which promote a common environment, XML will be investigated as an environment for a model for content-based billing. Chapter 3 will focus on why XML is the choice for a common environment and how it promotes information sharing and delivery.

Chapter 3

eXtensible Markup Language

Chapter 2 explored the evolution of business over the Internet. For this evolution to be effective a common environment is required for information delivery. Currently HTML (Hyper Text Markup Language) is the most widely used method of information delivery on the Internet. This chapter therefore investigates HTML and points out some of its shortcomings.

XML (eXtensible Markup Language) is investigated as a solution to the problems with HTML. This chapter argues that XML, being a powerful and flexible platform for information delivery, forms a solid foundation for a common environment for information delivery on the Internet. This chapter also discusses some of the related standards that are XML based. These standards are key to enhancing the power and flexibility of XML as an environment for information delivery. A common environment for information delivery on the Internet that is facilitated by XML, is key to fulfilling the proposed model's requirements of power and flexibility.

3.1 HTML

HTML is the lingua franca for publishing on the Internet, instructing the web browser how to display the information on web pages. HTML provides a simple, common language to create web pages that can be easily viewed by just about anyone, more or less as intended. HTML is non-proprietary and can be created and processed by a wide range of tools from plain text editors to sophisticated WYSIWYG (What You See Is What You Get) authoring tools (W3C, 2002b).

```
1  <html>
2   ...
3  <body>
4  <p align="center"><b><font size="6">EDU ONLINE</font></b></p>
5  <p align="center">&nbsp;</p>
6  <p align="center"><b><font size="4">Please Login</font></b></p>
7  <p align="center">&nbsp;</p>
8  <p align="center">&nbsp;</p>
9  <p align="center">&nbsp;</p>
10 <p align="center"><b>Username: </b>
11 <input type="text" name="textUserName" size="20"></p>
12 <p align="center"><b>Password:&nbsp; </b>
13 <!--webbot bot="Validation" b-value-required="TRUE" i-minimum-length="6" -->
14 <input type="password" name="textPassword" size="20"></p>
15 ...
16 </body>
17 </html>
```

Figure 3.1: HTML file snippet

Figure 3.1 shows a snippet from an HTML file generated by a popular authoring tool. Line 4 instructs the browser to display “EDU ONLINE” centered on the web page and in a large font. “EDU ONLINE” is intended as a heading for the web page, but this is not clearly expressed on line 4, instead the heading is combined with presentation information. This demonstrates that HTML does not separate content (the heading) and presentation (the size, position, colour etc. of the heading). The lack of effective separation of content and presentation is one of the primary problems with HTML (Hampton, 2002).

HTML is a presentation language: its tags, the identifiers put before and after text blocks, can only be used to describe the look and feel of the document. This way, a web browser can figure out how the presentation of the document should look, but nothing more. HTML was never designed for tasks other than presenting information on a screen (Willaert, 2001).

The previously discussed example from figure 3.1 illustrates that by using a presentation-only markup language such as HTML, much of the structure and meaning of the data is lost.

Another problem with HTML is that both the tag semantics and the tag set are fixed. An `<h1>` tag is always a first level heading. Line 4 from figure 3.1 could have been written to take advantage of the `<h1>` tag as follows:

`<h1>EDU ONLINE</h1>`

This would make “EDU ONLINE” the first level heading on the web page. A tag such as `<title>` does not exist and can’t be defined by the web site developer to be used to better describe the “EDU ONLINE” heading. The lack of the ability to define your own tags is a limitation of HTML as the set of predefined tags such as `<h1>` is small. The W3C, in conjunction with browser vendors and the WWW community, is constantly working to extend the definition of HTML to allow new tags to keep pace with changing technology and to bring variations in presentation to the Web. However, these changes are always rigidly confined by what the browser vendors have implemented and by the fact that backward compatibility is paramount. For people who want to disseminate information widely, features supported by only the latest releases of web browsers are not useful (Shneiderman, 1997).

Cascading Style Sheets (CSS) (Lie & Bos, 1999) is an attempt at removing some of the presentation from HTML documents. CSS is a simple style sheet language that allows authors and readers to attach style (e.g. fonts, colors and spacing) to HTML documents. One of the fundamental features of CSS is that style sheets cascade; authors can attach a preferred style sheet, while the reader may have a personal style sheet to adjust for human or technological handicaps. The rules for resolving conflicts between different style sheets are defined in the W3C CSS specification. To design a style sheet requires a little HTML knowledge and some basic desktop publishing terminology. A CSS style sheet consists of a number of CSS rules. A CSS rule consists of two main parts: the selector that identifies the element the style is applied to, and the declaration which consists of a property and a value. An example of a CSS rule is:

`h1 { color: blue }`

This simple CSS rule has `h1` as the selector identifying the element `<h1>` in the HTML document as the target. The declaration identifies that the property to be affected is the colour of element `<h1>`, and sets that colour to blue. The selector part is the link between the HTML document and the style sheet, with all HTML element types being possible selectors (Lie & Bos, 1999).

As discussed, HTML has some fundamental limitations that prevent it from being a common environment for information delivery that is both

```
1  <eduonline>
2          <exercise>
3              <title status="public" code="001">
4                  XML Tutorial 1
5              </title>
6              <body status="public">
7                  This is an introductory tutorial for
8                  the XML (eXtensible Markup Language).
9              </body>
10         </exercise>
11         <solution>
12             <title status="private" code="001">
13                 Solution to XML Tutorial 1
14             </title>
15             <price>
16                 123.06
17             </price>
18             <body status="private">
19                 This is the solution to XML Tutorial 1.
20                 You will have to buy this solution.
21             </body>
22         </solution>
23 </eduonline>
```

Figure 3.2: An XML document

powerful and flexible. To address the limitations of HTML the W3C has introduced XML.

3.2 XML (eXtensible Markup Language)

Presentation and content is effectively separated by the eXtensible Markup Language (XML) (W3C, 2002a), which describes the information itself. XML is a meta-markup language that provides a way to create extensible formats for describing structured data in electronic documents and for expressing rules about the data. XML specifies neither semantics nor a tag set. All of the semantics of an XML document will either be defined by the applications that process them or by stylesheets, and the structure of an XML document is specified by a DTD or XML schema that is associated with the document.

Figure 3.2 shows an XML document demonstrating the separation of content and presentation in XML. The XML document uses custom tags that contain content and no presentation information. For example, lines 2 to 10 describe an exercise that has a title and a body, defined by the

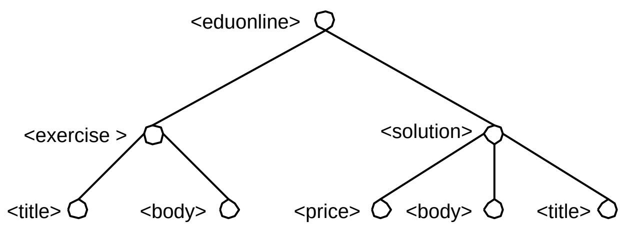


Figure 3.3: The XML Document Tree

`<title></title>` tag pair and the `<body></body>` tag pair. Neither of these tag pairs contain any presentation information, only content.

The presentation information for the XML document in figure 3.2 on the facing page is contained in a separate stylesheet. This is how content and presentation is separated in XML. The separation of content and presentation in XML allows the same document to be displayed on different media, and it also enables users to view the document according to their preferences and abilities, just by modifying the style sheet.

XML has a hierarchical structure that can be mapped as a document tree. Figure 3.3 shows the document tree for the XML document shown in figure 3.2 on the facing page. The fact that the structured nature of an XML document allows it to be mapped as a document tree is used extensively by the stylesheet. The stylesheet is created using the eXtensible Stylesheet Language (XSL) (Froumentin, 2002a). XML is often used for structured documents such as XHTML (Pemberton et al., 2001), rendering or transformation languages such as eXtensible Stylesheet Language Transformations (XSLT) (Clark, 1999), and as the basis for extensible network protocols such as SOAP (Gudgin et al., 2001).

3.2.1 Well-formed XML

The structured nature of XML also allows for greater security than HTML. This is achieved by XML imposing strict rules to the structure of XML documents. An XML document that adheres to these rules is said to be well-formed.

Figure 3.4 on the next page shows a snippet from a well-formed XML document. In the snippet every element consists of both a start and end tag, for example, the element tag pair `<username></username>` (lines 5 and

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"href="contents.xsl"?>
<eduonline>
 ...
    <username value="Peter">
        <credential name="student" value="true">
            <institution name="PE Tech"/>
        </credential>
        <credential name="creditworthy" value="true">
            <payment>
                <creditcard>
                    <holderName>
                        Peter de Villiers
                    </holderName>
                    <number>
                        17846869984546
                    </number>
                    <expire>
                        22 July 2005
                    </expire>
                </creditcard>
            </payment>
        </credential>
        <credential name="age" value="23"/>
    </username>
...
</eduonline>
```

Figure 3.4: Well-formed XML document snippet

25). In addition to properly closed elements the XML document has only one root element, `<eduonline>` (line 3).

A well-formed XML document is one that has only one root element, and its child elements are properly nested. All elements must also have an opening and closing tag. An example of well nested elements are the `<credential>` elements on lines 6 to 8, 9 to 23, and line 24. A document can only be well-formed if it obeys the syntax of XML, therefore a document that includes sequences of markup characters that cannot be parsed or are invalid cannot be well-formed.

In addition to these rules a document must meet the conditions specified in Appendix A to be considered to be well-formed (Walsh, 1998).

By definition, if a document is not well-formed, it is not XML. This means that there is no such thing as an XML document which is not well-formed, and XML processors are not required to do anything with such documents (Willaert, 2001).

A document being well-formed is not a sufficient condition to prove the document's integrity. To determine if a document has integrity in terms of structure (well-formedness) as well as content means that it has to be validated.

3.2.2 Valid XML

A well-formed document is valid only if it contains a proper document type declaration and if the document obeys the constraints of that declaration (element sequence and nesting is valid, required attributes are provided, attribute values are of the correct type, etc.). The XML specification identifies all of the criteria in detail (Walsh, 1998).

Validity is important to show the document's integrity in terms of its content. This can be done in two places. Firstly when the data is first entered into a system, the data entry program can enforce obedience to the rules described in a schema. Secondly when a document comes from an external source a parser can compare it to a schema and make sure the document conforms.

The advantages of having well-formed and valid documents has also impacted the HTML community with the development of XHTML (Pember-

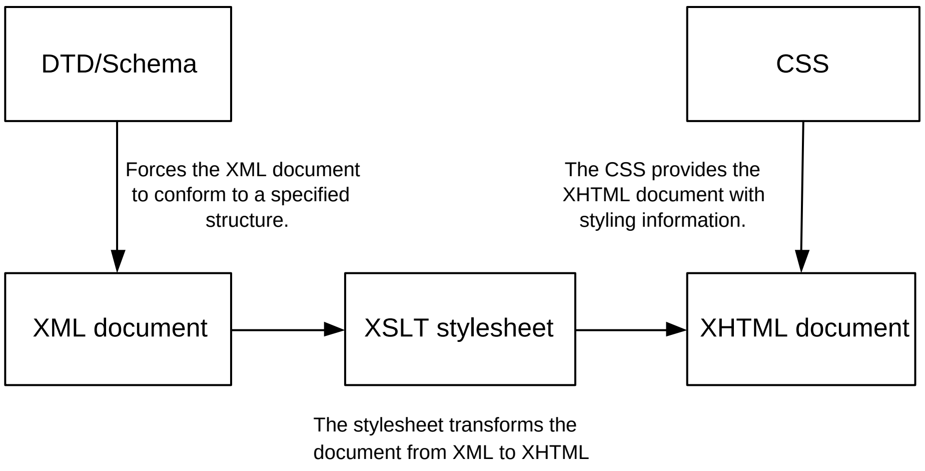

DTD/Schema
CSS
Forces the XML document to conform to a specified structure.
The CSS provides the XHTML document with styling information.
XML document
XSLT stylesheet
XHTML document
The stylesheet transforms the document from XML to XHTML

Figure 3.5: The Transformation of XML to XHTML

ton et al., 2001). XHTML is based on XML and is designed to be XML-conforming, allowing for the display of XML content.

XHTML is a family of document types that extend the HTML 4 document types. They are XML based, and are designed to work in conjunction with XML-based user agents.

XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools. XHTML documents can utilize applications (e.g. scripts and applets) that rely upon either the HTML Document Object Model or the XML Document Object Model (Le Hgaret et al., 2002).

Figure 3.5 shows that an XML document which conforms to a DTD or XML schema (Fallside, 2001) can be transformed to an XHTML document using an external stylesheet written with XSLT. Figure 3.5 also shows that a CSS can still be applied to the XHTML document.

In figure 3.6 on the facing page the XML family of standards are shown. These standards are at present all W3C recommendations. From figure 3.6 on the next page it can be seen that there are many standards that are based on XML. The rest of the chapter is going to discuss the DTDs and XML schemas that are used to ensure that the XML document is properly structured, as well as XSL, XSLT, XPath (Clark & DeRose, 1999) and XLink (DeRose, Maler, & Orchard, 2001) which are all standards involved with the transformation and presentation of XML documents.

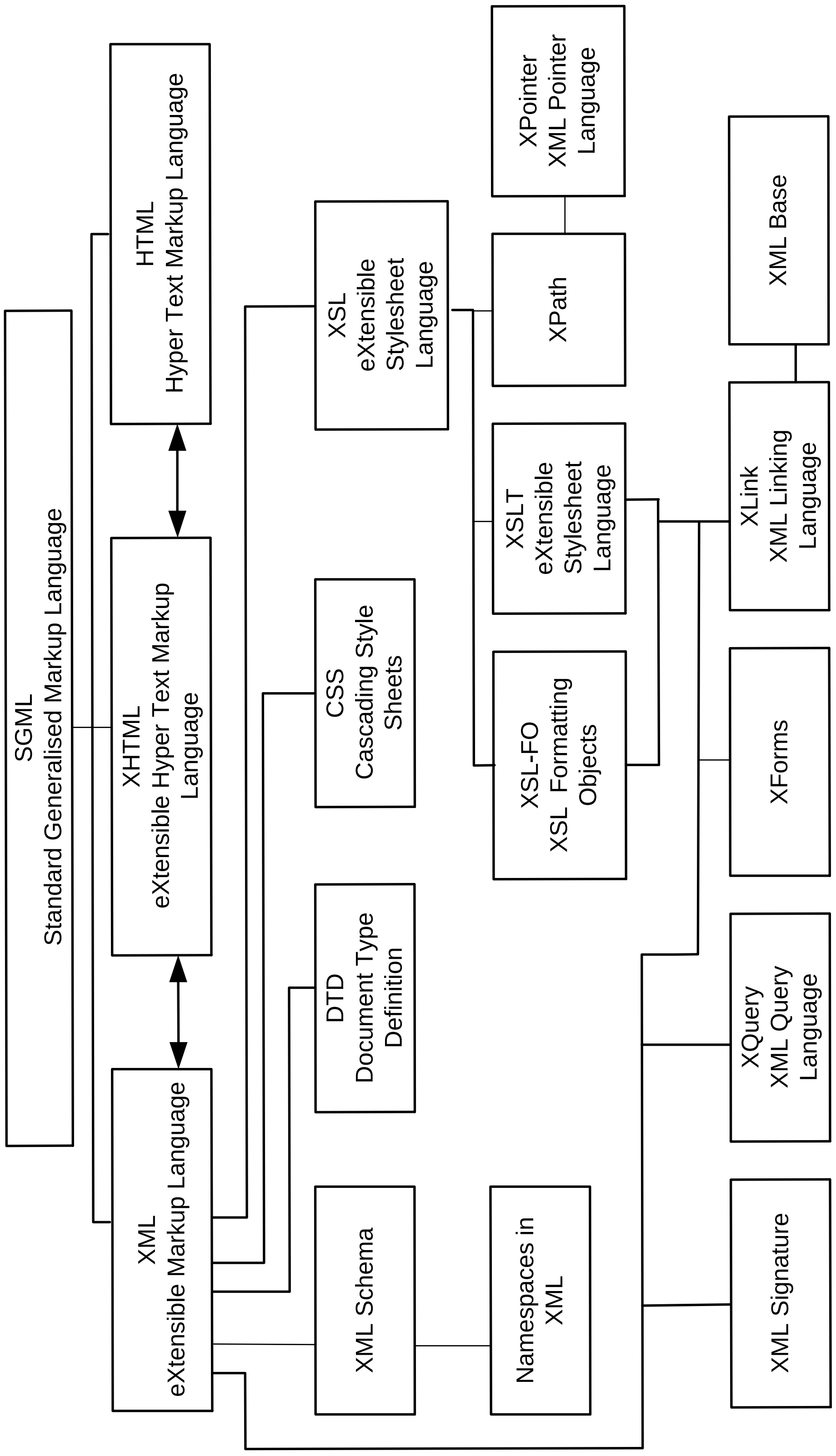


Figure 3.6: The XML family of standards hierarchy tree based on (Willaert, 2001)

3.3 DTDs (Document Type Definitions)

DTDs are a means for developers to define the structure of the content of an XML document. A DTD lists the elements, attributes, entities, and notations that can be used in a document, as well as their possible relationships to one another. A DTD specifies a set of rules for the structure of a document.

If the developer is using a DTD to validate an XML document, the XML document must specify the DTD it is valid with respect to. The DTD can be included in the XML document it describes, or the XML document can link to the DTD at an external URL. External DTDs located at an URL can be shared by different documents and Web sites. If the DTD is not directly included in the document but is linked in from an external source, changes made to the DTD automatically apply to all documents using that DTD. However, backward compatibility is not guaranteed when a DTD is modified. Incompatible changes can thus invalidate documents.

DTDs provide a means for applications, organizations, and interest groups to agree upon, document, and enforce adherence to markup standards. DTDs also help ensure that different people and programs can read each others files.

Furthermore, a DTD shows how the different elements of a document are arranged. A DTD shows the generic structure of a document separate from the actual data in the individual document instances. This allows styling and formatting of the underlying structure to take place without the destruction of the structure (Harold, 2001).

DTDs do however have a few drawbacks. Firstly, DTDs are not very readable. This makes them not only hard to write, but hard to understand, which in turn makes work for designers more difficult, and promotes the formation of legacy DTDs. This leads to organizations being afraid to modify existing DTDs as they have become too complicated. Secondly, DTDs do not allow differentiation between different primary data types (text strings, real or integer numbers) (Willaert, 2001).

Figure 3.7 on the facing page shows a snippet from a simple XML document. In order to ensure that the XML document in figure 3.7 is properly structured a DTD is used. Figure 3.8 shows a DTD snippet that is used to ensure that the XML snippet in figure 3.7 is properly structured.

An attempt at reading the DTD snippet in figure 3.8 illustrates the pri-

```
 ...
   <username value="Peter">
       <credential name="student" value="true">
           <institution name="PE Tech"/>
       </credential>
       <credential name="creditworthy" value="true">
           <payment>
               <creditcard>
                   <holderName>
                       Peter de Villiers
                   </holderName>
                   <number>
                       17846869984546
                   </number>
                   <expire>
                       22 July 2005
                   </expire>
               </creditcard>
           </payment>
       </credential>
       <credential name="age" value="23"/>
   </username>
...
```

Figure 3.7: XML document snippet

```
 ...
   <!ELEMENT username (credential*)>
   <!ELEMENT credential (institution)?,(payment?),(EMPTY)>
   <!ELEMENT institution EMPTY>
   <!ELEMENT payment (creditcard)>
   <!ELEMENT creditcard (holdername,number,expire)>
   <!ELEMENT holdername (#PCDATA)>
   <!ELEMENT number (#PCDATA)>
   <!ELEMENT expire (#PCDATA)>

   <!ATTLIST username value CDATA Guest #REQUIRED>
   <!ATTLIST credential name CDATA Guest #REQUIRED>
   <!ATTLIST credential value CDATA true #REQUIRED>
   <!ATTLIST institution name CDATA Guest #REQUIRED>
   <!ATTLIST creditcard name CDATA Guest #REQUIRED>
...
```

Figure 3.8: DTD snippet

mary problem facing designers attempting to use DTDs, which is their lack of readability. An example of this is line 3. Line 3 states that the `<credential>` element must either be empty, or it can contain an `<institution>` element or a `<payment>` element. Furthermore, lines 12 and 13 define attributes for the`<credential>` element as well as default values for these attributes and whether they are required or not.

However, because none of the newer standards still rely on DTDs, they are expected to be replaced by the newer, more powerful XML Schema Language (Willaert, 2001).

3.4 XML Schema

The purpose of a schema is to define a class of XML documents. XML Schemas (Fallside, 2001) are expressed in XML syntax, which allows for easy access and manipulation, but is also extremely verbose. However, this verbosity allows for increased functionality and makes XML schemas a lot more readable than DTDs. In contrast with DTDs, XML schemas allow the use of Namespaces.

Figure 3.7 on the page before shows an XML snippet from a simple XML document. There are two methods of ensuring that the XML snippet in figure 3.7 is properly structured. The first method is by using the DTD snippet shown in figure 3.8 on the preceding page. As previously discussed in the section covering DTDs this DTD snippet has readability issues. The issue of readability in DTDs is addressed by XML Schemas. Figure 3.9 on the next page illustrates the enhanced readability of an XML Schema snippet. Using an XML Schema is the second method that can be used to ensure that an XML document is properly structured.

The XML schema snippet in figure 3.9 provides the same functions as the DTD snippet in figure 3.8. However, it is much more verbose as it is expressed in an XML syntax, enhancing readability. The XML schema snippet also has the possibility for greater functionality than the DTD as XML Schema allows for the definition of complex and simple data types, as well as providing access to built-in simple data types.

An XML Schema consists of components such as type definitions and element declarations. These can be used to assess the validity of well-formed

```
<xs:schema xmlns:xs="http://www.w3.org/2000/10/XMLSchema">
...
    <xs:element name="username">
        <xs:complexType content="elementOnly">
            <xs:sequence>
                <xs:element name="credential" type="credentialType"
                minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="value" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="credentialType" content="elementOnly">
        <xs:sequence>
            <xs:element name="institution" type="institutionType"
            minOccurs="0"/>
            <xs:element name="payment" type="paymentType"
            minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="institutionType" content="elementOnly">
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="paymentType" content="elementOnly">
        <xs:sequence>
            <xs:element name="creditcard" type="creditcardType"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="creditcardType" content="elementOnly">
        <xs:sequence>
            <xs:element name="holdername" type="holderType"/>
            <xs:element name="number" type="numberType"/>
            <xs:element name="expire" type="expireType"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="holderType" base="xs:string"/>

    <xs:complexType name="numberType" base="xs:string"/>

    <xs:complexType name="expireType" base="xs:string"/>
...
</xs:schema>
```

Figure 3.9: XML Schema snippet

elements and attribute items and furthermore may specify augmentations to those items and their descendants.

Each of the elements in the schema has a prefix “xs:” which is associated with the XML Schema namespace through the declaration, `xmlns:xs=http://www.w3.org/2001/XMLSchema`, that appears in the schema element as seen on line 1 of the XML schema snippet in figure 3.9 on the preceding page. The prefix “xs:” is used by convention to denote the XML Schema namespace, although any prefix can be used. The same prefix, and hence the same association, also appears on the names of built-in simple types, e.g. “xs:string”. The purpose of the association is to identify the elements and simple types as belonging to the vocabulary of the XML Schema language rather than the vocabulary of the schema author.

In XML Schema, there is a basic difference between complex types which allow elements in their content and may carry attributes, and simple types which cannot have element content and cannot carry attributes. There is also a major distinction between definitions which create new types (both simple and complex) and declarations which enable elements and attributes with specific names and types (both simple and complex) to appear in document instances. Line 13 of the XML schema snippet in figure 3.9 on the page before creates a new type, while lines 14 to 21 of the XML schema snippet enable elements and attributes with specific names and types to appear in document instances.

New complex types are defined using the `<complexType>` element and such definitions typically contain a set of element declarations, element references, and attribute declarations. A new complex type is defined in the XML schema snippet in figure 3.9 on lines 13 to 21. Lines 15/16 of the XML schema snippet declare an element that can occur zero or more times. Line 20 of the XML schema snippet contains an attribute declaration. The declarations are not themselves types, but rather an association between a name and the constraints which govern the appearance of that name in XML documents governed by the associated schema. Elements are declared using the “element” element, and attributes are declared using the “attribute” element.

For example, if a complex type is defined, and within this definition we see two element declarations and two attribute declaration. Then when any element within a XML document is declared as that complex type, that element must consist of two elements and two attributes. These elements must also be called by the same name as specified in the schema and must also appear in the same sequence as specified by the schema.

XML schema allows for the definition of how many times an element has to occur, allowing for the specification of the minimum and maximum occurrences. XML schema allows attributes and elements to be declared with default values, making provision for global attributes which can be defined for assigning properties to all elements. XML schema defines element content and makes provision for mixed element content and empty content. XML schema provides for list types and union types. XML schema supports schema inheritance which allows borrowing from a schema, but overriding it where new features are needed. XML schema provides support for relationships which includes specifying their cardinality, and elements which must be unique or may have null values can be used. XML schema also provides better documentation through annotations for the benefit of human and application readers (Fallside, 2001).

All the above-mentioned features of XML Schema make it powerful and flexible. Therefore XML schema would seem to be the choice for providing XML documents with a valid schema to conform to, in contrast to DTDs which are limited and lack XML Schema's features.

The use of XML schemas to define an XML document's structure ensures conformity to a specified structure by that XML document. This enhances the ability to effectively separate content and presentation as an XSL stylesheet can easily be developed to present content that conforms to the structure specified by the XML schema.

3.5 XSL

XSL (Froumentin, 2002a) is a language for expressing stylesheets consisting of three parts. The first part is XSL Transformations (XSLT) (Clark, 1999) which is a language for transforming XML documents. The second part is the XML Path Language (XPath) (Clark & DeRose, 1999) which is an expression language used by XSLT to access or refer to parts of an XML document. The third part is XSL Formatting Objects (XSL-FO) which is an XML vocabulary for specifying formatting semantics (Froumentin, 2002a). In this section focus will be given to XSLT and XPath, followed by an investigation into XLink as a supporting specification for linking within XML documents.

XSL uses an XML notation. An XSL style sheet is, like CSS (Cascading Style Sheet), a file that describes how to display an XML document of a given type. In XSL, the formatting object tree can be radically different from the source tree, and not the source tree. Inheritance of formatting properties is on the formatting object tree. XSL is targeted at XML, in particular highly-structured, data-rich documents that require extensive formatting (W3C, 2002c).

3.5.1 XSLT

XSLT is designed for use as part of XSL, and is a transformation language for XML documents. However, XSLT has also been designed to be used independently of XSL. XSLT was originally intended to perform complex styling operations, but it is now used as a general purpose XML processing language. XSLT is thus widely used for purposes other than XSL, like generating XHTML web pages from XML data.

Styling requires a source XML document, containing the information that the style sheet will display and the style sheet itself which describes how to display a document of a given type.

Figure 3.10 on the facing page shows an XML document snippet. This XML document snippet doesn't contain any presentation information, which is instead contained in a separate stylesheet. Separating the document's content and the document's styling information allows the same document to be displayed on different media (like screen, paper, cell phone), and it also enables users to view the document according to their preferences and abilities, just by modifying the stylesheet.

An XSLT stylesheet contains a set of template rules. A template rule has two parts: a pattern which is matched against nodes in the source tree and a template which can be instantiated to form part of the result tree.

Figure 3.11 on the next page shows an XSLT stylesheet snippet that contains the presentation information that is used to transform the XML document snippet in figure 3.10.

The XSLT stylesheet can be used to transform any instance of the DTD or XML schema it was designed for. The first rule shown on line 2 of the XSLT

```
 ...
   <username value="Peter">
       <credential name="student" value="true">
           <institution name="PE Tech"/>
       </credential>
       <credential name="creditworthy" value="true">
           <payment>
               <creditcard>
                   <holderName>
                       Peter de Villiers
                   </holderName>
                   <number>
                       17846869984546
                   </number>
                   <expire>
                       22 July 2005
                   </expire>
               </creditcard>
           </payment>
       </credential>
       <credential name="age" value="23"/>
   </username>
...
```

Figure 3.10: XML document snippet

```
 ...
  <xsl:template match="username">
       <xsl:apply-templates select="credential"/>
  </xsl:template>

  <xsl:template match="credential">
        <xsl:if test="@name='creditworthy'">
           <xsl:choose>
               <xsl:when test="@value = 'true'">
                   <xsl:apply-templates select="/eduonline/contents"/>
               </xsl:when>
               <xsl:otherwise>
                  This user is not creditworthy.
               </xsl:otherwise>
           </xsl:choose>
       </xsl:if>
  </xsl:template>
...
```

Figure 3.11: XSLT stylesheet snippet

stylesheet snippet in figure 3.11 on the preceding page, matches a `<username>` element in the XML snippet. Within this context the template that matches any `<credential>` elements within the XML snippet in figure 3.10 on the page before is then applied. This is performed by the code on line 3 of the XSLT stylesheet snippet in figure 3.11 on the preceding page. The next step within the template matching a `<credential>` element (starting on line 6 of the XSLT stylesheet snippet in figure 3.11) is to test if the name of the matched credential is "creditworthy". This test is performed on line 7 of the XSLT stylesheet snippet in figure 3.11. If the test evaluates to true the value attribute of the `<credential>` element is evaluated (lines 8 and 9 of the XSLT stylesheet snippet in figure 3.11). If the value of the attribute evaluates to true then a new template is applied (line 10 of the XSLT stylesheet snippet in figure 3.11). Otherwise a message is displayed (line 13 of the XSLT stylesheet snippet in figure 3.11) (W3C, 2002c).

As discussed, a transformation expressed in XSLT describes rules for transforming a source tree into a result tree. The transformation is achieved by associating patterns with templates. A pattern is matched against elements in the source tree. A template is then instantiated to create part of the result tree. The result tree is separate from the source tree and the structure of the result tree can be completely different from the structure of the source tree. In constructing the result tree, elements from the source tree can be filtered and reordered, and arbitrary structure can also be added.

XSLT makes use of the expression language defined by XPath for selecting elements for processing, for conditional processing and for generating text (Clark, 1999).

3.5.2 XPath

Addressing XML documents occurs by using an XPath (Clark & DeRose, 1999) expression. In addition to addressing functions, XPath provides basic facilities for the manipulation of strings, numbers and booleans. XPath operates on the abstract, logical structure of an XML document (the document tree), rather than its surface syntax. XPath also has a natural subset that can be used for matching (testing whether or not a node matches a pattern).

XPath works by modelling an XML document as a tree of nodes as depicted in figure 3.12 on the next page. The nodes can be different types,

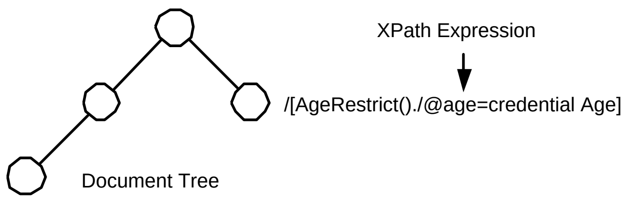


Figure 3.12: XPath modelling of a document tree

including element nodes, attribute nodes and text nodes. XPath defines a way to compute a string-value for each type of node. Some types of nodes can also have names. The primary syntactic construct in XPath is the XPath expression as shown in figure 3.12.

An XPath expression is evaluated to yield an object. This object is a part of the document tree which can consist of one or more nodes. Because of this ability to identify a specific part of the document tree, access control can be performed on the document tree. XPath expression evaluation occurs with respect to a context. XSLT and XPointer (DeRose, Maler, & Daniel Jr., 2001) specify how the context is determined for XPath expressions used in XSLT and XPointer respectively.

Another specification that makes use of XPath is the XML Linking Language (XLink) (DeRose et al., 2001).

3.6 XLink (XML Linking language)

HTML linking allows for a link to be made to a resource in one direction. To make a link you just need write access to the starting point, not to the resource you are linking to. This introduces some robustness through tolerance of broken links as lost resources do not jeopardize the system, but at the same time makes it hard to keep links up to date.

With HTML there is no granularity of links. In general a link can only be to a complete entity. You can however point to an anchor ID within a document, but this only tells the browser to move the view to that point. You can only specify a destination spot, not a paragraph, table or section, within the linked document.

XLink (DeRose et al., 2001) solves the problems with HTML linking in two ways. Firstly, the definition of the link can be separated from its starting point and put into a link element, which also contains the reference to the destination point.

Secondly, the XPointer language is used to address into XML resources and to specify a destination region. In addition, some new options are provided for the link traversal behaviour, such as when to traverse the link and what exactly happens then. XLink consists only of attributes, making it an enabling vocabulary, meant to be incorporated in your own vocabulary. XLink's namespace is `http://www.w3.org/1999/xlink`.

In XLink, there are two types of link elements: HTML-style simple links and link elements that contain several kinds of participants, and one or more arcs. The participants can be resources of any type, either XML or non-XML, and are either contained inside the link element (local resources) or are referenced by a URI (locators or remote resources). If the resource is an XML document, XPointer expressions can be used. One extended link element can have more than two participants, which removes an important restriction of HTML links. The actual connection between the participants is defined by the arcs, which indicate the starting and ending point(s) of the respective resources. In addition, metadata can be attached to these arcs and participants, containing data that is readable by humans or processors (Willaert, 2001).

3.7 Conclusion

XML is key to creating a more dynamic web through various standards that are using its structured nature as a basis.

XML has been chosen as the platform for information delivery for CBiX because XML has a flexible and extensible architecture that separates content and presentation, has to be well-formed and uses XML schemas or DTDs to validate the structure of the XML document's content. Furthermore XML has a great potential impact on the next generation web with a large number of formatting, communication and security standards being developed based on XML and supported by major players in ICT. Lastly, XML's structured nature and the ability to prune the document tree as needed facilitates access control and allows users to view only the information they are permitted to view.

Due to the separation of content and presentation in XML, focus can be given to protecting the information content without being concerned with the presentation of information (Botha & Eloff, 2002) which is especially important for a billing model where security is of paramount importance.

In chapter 4 access control will be discussed with the aim to choosing the best method of providing access control for the proposed model.

Chapter 4

Access Control

Chapter 3 provided an introduction into the workings of XML, as well as the benefits it offers over HTML. One benefit of XML is that it allows for greater security to be implemented. This is especially important for a billing system, due to the financial implications.

This chapter firstly investigates the access control decision, dividing access control into three phases for closer examination. Thereafter the chapter identifies a number of access control administrative paradigms, which fundamentally differ with regards to access resolution. Following an introduction into the various administrative paradigms of access control this chapter investigates the synergy between access control and billing by focussing on the three identified phases. This chapter selects an access control administrative paradigm suited to the proposed model. Following a choice of an administrative paradigm, the technologies that are available to implement the selected administrative paradigm, credential-based access control, in a billing system for XML environments are then investigated. Lastly access control for an XML environment is investigated as XML has been selected as the common environment for the proposed model.

4.1 The Access Control Decision

Access Control is typically implemented to answer the fundamental question of who can access what resources, and what determines whether or not access can be granted to the selected resources.

According to Sandhu and Samarati (1994) the purpose of access control is to limit the actions or operations that a legitimate user of a computer system can perform. Access control allows a user access to resources based on authorizations specified for a user.

To further examine Access Control, it can be divided into three phases. For the purposes of discussion the three phases have been named:

- Phase 1: Identification and Authentication
- Phase 2: Attempt to Access a Resource
- Phase 3: Access Resolution

Consider each in turn.

4.1.1 Phase 1: Identification and Authentication

Identification and Authentication is the first phase identified for access control. In this phase the user is identified and authenticated. According to Sandhu and Samarati (1996) user-to-computer authentication can be based on one or more of the following. Firstly, something the user knows such as a password. Secondly, something the user possesses such as a cryptographic token or smart card. Thirdly, something the user is, exhibited in a biometric signature such as a fingerprint or voiceprint.

After Identification and Authentication have taken place, a session is established to create an environment in which the user can attempt to perform actions. This leads to the second identified phase.

4.1.2 Phase 2: Attempt to Access a Resource

This phase is an attempt to access a resource. In access control the user's session permissions will determine which resources the user can view.

In the case of access control in an operating system the user can view the resources on the connected network. The resources as well as the nature of the access control requirements would differ depending on the environment that is protected. Workflow systems, for example, differ from operating systems in the method they use to provide a user with a view of resources he/she can access. In a workflow system the user can only view the tasks that he has to perform (Botha, 2001), whereas in an operating system a user can view all the resources, both those he/she can and cannot access.

Once a user attempts to access a resource, access resolution has to occur.

4.1.3 Phase 3: Access Resolution

Access resolution is used to determine if a user has permission to access a selected resource. Access resolution differs depending on the access control mechanism selected to secure the resources the user is attempting to access. At this point in the three phases it becomes necessary to examine various access control paradigms, as access control can take on the form of a number of administrative paradigms, with each paradigm handling access resolution in a different manner.

4.2 Access Control Paradigms

Access control can take on the form of several administrative paradigms. From Sandhu and Samarati (1996) three main administrative paradigms can be identified. These are Discretionary Access Control (DAC), lattice-based access control, also known as Mandatory Access Control (MAC) and Role-based Access Control (RBAC). Furthermore the author maintains that a fourth access control administrative paradigm can be identified from Samarati (2002) namely Credential-based Access Control.

Consider each in turn.

4.2.1 Discretionary Access Control

The first of the identified administrative paradigms of access control is DAC. If a DAC approach is followed the owner of the data specifies which users are authorized to access the data. If a user wishes to gain access to the data the user's identity and associated access control privileges must meet the requirements specified by the author of the data. The advantage of DAC is its flexibility, making DAC suitable for a variety of systems and applications. The disadvantage of DAC is that data can be freely copied by a user that is authorized to access the data. Access can then be obtained to the copied data by users without the authorizations that the data's owner specified (Department of Defense National Computer Security Center, 1985).

4.2.2 Mandatory Access Control

The second identified administrative paradigm of access control is MAC. If a MAC approach is followed each user and object in the system is assigned a security level. The security level associated with an object determines the sensitivity of the information contained within the object, and the security level associated with the user determines the user's level of trust. Two of the principles of MAC are read down and write up. The read down principle allows a user access to objects with the same or lower security level as that user's security level. The write up principle allows a user to write an object at a higher security level than is assigned to the user, but not write objects at a lower security level than is assigned to the user. If an user wishes to communicate with others of a lower security level, then that user has to sign on at a lower security level (Department of Defense National Computer Security Center, 1985). Therefore, for access resolution in MAC the user has to have an equal or greater security level than the requested object for access to be granted.

4.2.3 Role-based Access Control

The third administrative paradigm of access control identified by Sandhu and Samarati (1996) is Role-based Access Control (RBAC). RBAC limits the user's access to information based on the role of the user. A role is a job function within an organization that describes the authority and responsibility conferred on a user assigned to the role (Sandhu et al., 1996; Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001). The user's role therefore defines the set of actions and responsibilities that the user may undertake. For RBAC, instead of having to specify all the authorizations for each user, a user is simply assigned a role, and access authorizations for objects are specified in terms of roles. For access resolution in RBAC the various permissions assigned to the possibly multiple roles of the user have to be resolved. These possibly conflicting role permissions are resolved by determining the highest set of permissions according to the role hierarchy. For example, a person's one role may give him permission to access information available only to managers, and another role gives him access to information available to clerks. Therefore after resolution of the roles this user will get the highest set of permissions which would be access to both managerial and clerk information (Sandhu & Samarati, 1994).

4.2.4 Credential-based Access Control

Credential-based access control is the fourth access control paradigm identified from Samarati (2002). In credential-based access control a user is granted or denied access based on a number of user properties or credentials. In credential-based access control the user's credential set has to match the required set of credentials to access a restricted object.

A credential is a set of properties concerning a user that are relevant for security purposes. Authorizations are expressed by specifying the user receiving the authorizations in terms of conditions against user credentials. User credentials thus represent a way to support access control based on user qualifications and profiles. For example, credentials could be properties such as the user's age and credit rating. By using credentials one can directly formulate policies such as "The user can only have the option to buy an article from the online journal if he does not have a negative credit rating".

Conflict resolution within credential-based access control would differ from RBAC. Within RBAC these conflicts are typically resolved by obtaining the highest set of permissions according to the role hierarchy. Credential conflict resolution is a logical "and", in that if one condition is false, the expression is false. For example, if the material to be viewed is restricted to users over the age of 18 and a user's credentials indicate that he has a good credit rating, but he is under 18 then access is denied as he fails the condition of being over 18 (Samarati, 2002).

One possible manner of practically performing credential-based access control is by using digital certificates to present the system with user credentials, which grants the user access to the system unobtrusively. The digital certificates represent statements from trusted certification authorities. These statements ensure that the user credentials of the certificate's holder are valid. The credential-based access control system grants or denies access based on the properties in the supplied digital certificate (Samarati, 2002; Housley, Ford, Polk, & Solo, 1999; Biskup, 2002).

While each of the mentioned administrative paradigms approach access control in a different manner, these administrative paradigms are not orthogonal paradigms. Osborn, Sandhu, and Munawer (1996) demonstrate this by proving that MAC and DAC can be implemented using RBAC. Therefore, just as MAC and DAC can be implemented using RBAC, credential-based access control could also forseeably be implemented using roles as a user's credentials.

In addition to the four administrative paradigms for access control, Bullock and Benford (1999) identify four basic requirements for collaborative access control models.

4.3 Access Control and Billing

The author has identified three of the four basic requirements (Bullock & Benford, 1999) as inherent to each of the four identified access control administrative paradigms.

The access control service should be *unobtrusive.* It should be implemented in such a way that it is unobtrusive to the user. The user should not have to request access, instead access should be granted in such a way that there is no interruption to the work of the user. The access control service should also be *simple.* It should be easy to understand how the service functions. The access control service should be *maintainable.* It should be easy to administer the system and change permissions as needed.

The author contends that to a certain extent these three principles also apply to a billing system. A billing system must be simple, and it must be easy to maintain. However, a billing system would differ from an access control service with respect to unobtrusiveness. By adhering to Nielsen (1994)'s usability heuristics the billing system would have to provide the user with appropriate feedback within a reasonable time. A billing system would also require feedback from the user due to the fact that it has a contractual nature requiring the user to pay for the supplied content.

There is, however, a synergy between access control and billing in that they both, to a certain extent, adhere to the three principles identified from the basic requirements for collaborative access control models by Bullock and Benford (1999). Further synergy between access control and billing can be demonstrated by dividing billing into the three identified phases and comparing access control and billing as depicted in figure 4.1.

With respect to the first phase, identification and authentication, in the case of billing there is a greater need for information classifying the user based on properties describing the user than on knowledge of the user's relationship to other users in a hierarchical organizational structure such as in RBAC. Therefore, for a billing system, the user is assigned credentials, as credentials represent a set of properties concerning a user that are relevant for security purposes.

In the second phase, an attempt to access a resource in the case of billing this phase would consist of the evaluation of an XPath expression (Clark & DeRose, 1999) which will determine what the user can view based on his credentials. The method to provide the user with a view in billing would relate more closely to workflow systems than to operating systems in that only the material designated as "free" or "public" would be initially viewable. For example, non-member users visiting an online journal will only be able to view the abstracts describing the articles in the journal that they can purchase. If a user decides to purchase the available content the third phase will take place; determining if permission to access the resource is allowed or denied.

The third phase for both access control and billing is access resolution. It is however in this phase that the most significant differences between access control and billing can be observed.

This phase differs with regards to billing in that not only does permission have to be granted or denied, but also a method would be required which will determine the content pricing and instantiate the necessary financial transaction needed to pay for the purchased resource or service. Damiani et al. (2002) have demonstrated that is possible to prune an XML document tree using XPath expressions to obtain the desired user view.

For a content-based billing system the actual identity of the user is not as intrinsic to providing access to information. Instead, a content-based billing system requires a system where the user can securely present information about that user that is relevant for security purposes. The presented infor-

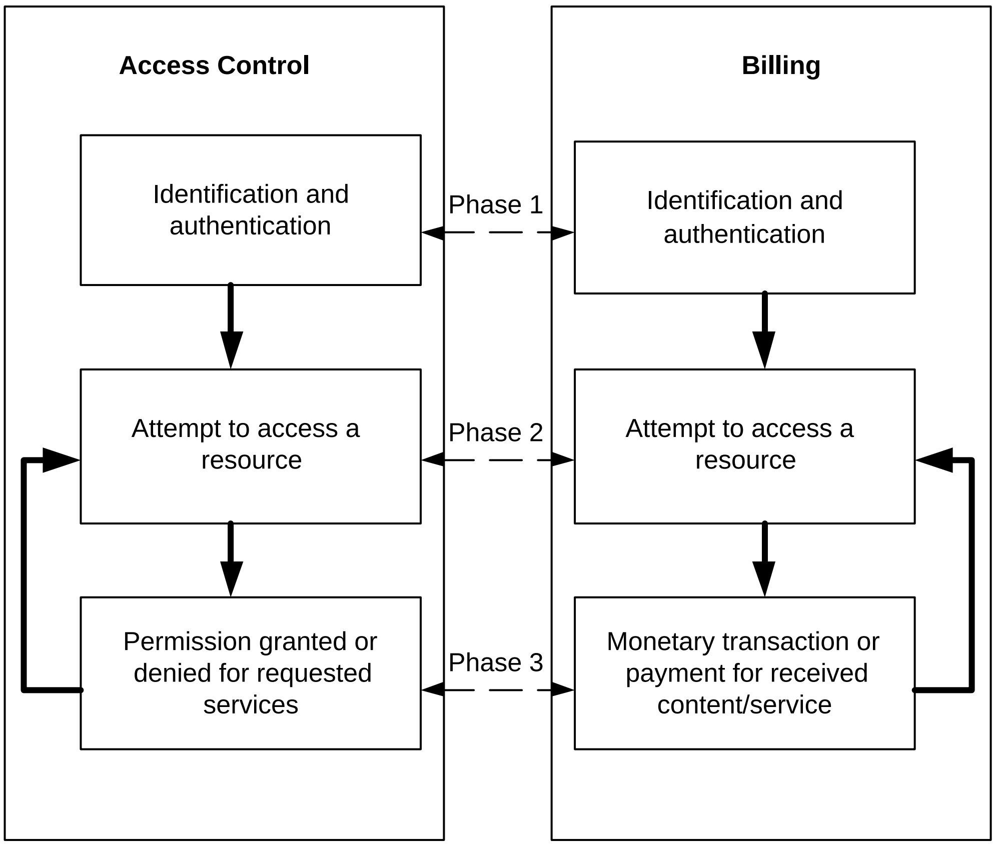

Access Control
Billing
Identification and authentication
Identification and authentication
Phase 1
Attempt to access a resource
Attempt to access a resource
Phase 2
Permission granted or denied for requested services
Monetary transaction or payment for received content/service
Phase 3

Figure 4.1: The Three Phases of Billing and Access Control

mation can then be used to provide access control. A content-based billing system therefore needs access information that once resolved can simply specify authorizations for XML content. The access information must also be able to be used by the content-based billing system to perform other functions such as content pricing.

Therefore, for the purposes of a content-based billing system, credential-based access control seems to provide a better possible solution for providing access control for XML content than any of the other identified access control administrative paradigms.

Not only does credential-based access control provide a means of specifying authorizations for XML content in terms of user credentials, but it also enables the content-based billing system to determine the pricing for content based on the user's credentials. Credential-based access control is therefore ideal for a web environment, specifically an XML environment as the power and flexibility of XML enables the implementation of access control within a billing system. There are technologies available to secure a credential-based access control system, specifically in an XML environment with XML technologies such as XML signature (Simon, Madsen, & Adams, 2001) and the XML Access Control Language (XACL) (Kudo & Hada, 2000b). The next section discusses these technologies in more detail.

4.4 Technologies to Secure a Credential-based Access Control System

It is important to protect the confidentiality of business messages. Equally important is ensuring the authenticity (ensuring that the identity of the sender can be determined by anyone), data integrity (ensuring that any alterations of the message content can be easily spotted by anyone) and non-repudiation (ensuring the origin or delivery of data in order to protect the sender against false denial by the recipient that the data has been received; or to protect the recipient against false denial by the sender that the data has been sent) of the messages. This kind of functionality is not totally catered for by existing Internet security technologies such as SSL (Secure Socket Layers) and security mechanisms such as username/password combinations.

The globally-recognized method for satisfying these requirements for secure business transactions is to use digital certificates to enable the encryption and digital signing of the exchanged data (Simon et al., 2001).

The term public key infrastructure (PKI) (Housley et al., 1999) is used to describe the processes, policies, and standards that govern the issuance, maintenance, and revocation of the certificates, public, and private keys that the encryption and signing operations require.

Public key cryptography allows users of an insecure network, like the Internet, to exchange data with confidence that it will be neither modified nor inappropriately accessed. This is accomplished by encrypting and decrypting the data using the information found in the public and private keys. Each participant in the exchange has such a pair of keys. They make the public key freely available to anyone wishing to communicate with them, and they keep the other key private and protected. Although the keys are mathematically related, if the cryptosystem has been designed and implemented securely, it is computationally infeasible to derive the private key from knowledge of the public key.

The nature of the relation between the public and private keys is such that a cryptographic transformation encoded with one key can only be reversed with the other. This enables confidentiality because a message encrypted with a public key can only be decrypted by the recipient with the corresponding private key. Even if the message is intercepted by a third party they won't be able to decrypt the message without having the private key.

Besides ensuring confidentiality, public and private keys enables authentication and data integrity. Together this supports non-repudiation. These features provide us with a method of digitally signing a message (Neuman & Tso, 1994).

To create a digital signature for a message, the data to be signed is transformed by an algorithm using the sender's private key. This can only be undone using the sender's public key. Therefore the recipient of the data can be confident of the origin of the data.

For signature verification to be meaningful, the verifier must have confidence that the public key does actually belong to the sender. A certificate, issued by a Certification Authority, asserts that the certificate's subject and that person's public key really belongs to the person that claims that public key as their own.

Largely due to the performance characteristics of public-key algorithms, the entire message data is typically not itself transformed directly with the private key. Instead a small unique thumbprint of the document, called a "hash" or "digest", is transformed. The recipient compares the hash that was sent to them with the hash they calculate. Any alteration of the original document would cause the hash to be changed. Additionally, by transforming the hash with their private key it then acts as a digital signature for the document. The recipient verifies the signature by taking a hash of the message and inputting it to a verification algorithm along with the signature that accompanied the message and the sender's public key. If the result is successful, the recipient can be confident of both the authenticity and integrity of the message (Simon et al., 2001; Netscape Netcenter, 2001).

For credential-based access control, specifically as pertaining to a content-based billing model, digital certificates provide a secure method to perform information exchange, authentication and non-repudiation while ensuring data integrity. The digital certificate of the user can provide the system with that user's credentials as well as their public key. The credentials in the digital certificate can be used to determine the user's access permissions, and the user's public key can be used to encrypt any content that needs to be delivered to that user.

The content-based billing model developed in this dissertation uses XML as a common environment for information exchange. Using XML as the common environment for the content-based billing model provides the opportunity to use XML specific technologies such as XML Key Management Specification (XKMS) (Froumentin, 2002b) and XML signature to provide credential-based access control within the content-based billing model.

Within an XML environment XML signature can be used to digitally sign XML data, allowing the user to present digital certificates containing his credentials in an XML format.

4.4.1 XML Signature

XML provides both challenges and opportunities for the application of encryption and digital signature operations to XML-encoded data. For example, in many workflow environments a person might wish to digitally sign only that part of the document that they take responsibility for. Older standards for digital signatures provide neither the syntax for capturing this sort of high-granularity signature nor mechanisms for expressing which portion a person wishes to sign.

XML signature (Eastlake, Reagle, & Solo, 2002) and XML encryption are two initiatives designed to take into account the special nature of XML data and take advantage of this.

XML signatures are digital signatures designed for use in XML transactions. The standard defines a schema for capturing the result of a digital signature operation applied to data. Like non-XML-aware digital signatures, XML signatures add authentication, data integrity, and support for non-repudiation to the data that they sign. However, unlike non-XML digital signature standards, XML signature has been designed to both account for and take advantage of the Internet and XML.

A fundamental feature of XML Signature is the ability to sign only specific portions of the XML tree rather than having to sign the complete document. This will be useful in situations where a document is authored at different times by different authors. Each author can sign only the elements of the document relevant to himself. This flexibility will also be used in situations where it is necessary to ensure the integrity of certain parts of the document while leaving other portions of the document open for editing. An example of this would be a signed XML form containing some default values that should not be changed being delivered to a user for completion. Once the user has completed the portions pertaining to him he can sign the section that he has completed ensuring that the original signature is still valid.

An XML signature can sign more than one type of resource. For example, a single XML signature might cover character-encoded data for example HTML, binary-encoded data for example a JPG, XML-encoded data, and a specific section of an XML file.

For signature validation to take place the data object that was signed must be available. The location of the original data object will be indicated by the XML signature. This can be done by referencing the data object with a URI within the XML signature. This will allow the XML signature to be either a child, parent, or sibling of the data object within the XML tree (Simon et al., 2001).

XML signature can therefore be used to digitally sign an XML document containing the user's credentials, ensuring their authenticity for the content-based billing model. Appendix B shows how to create and verify an XML signature. Hand in hand with XML signed user credentials is the need to provide access to specific XML data based on the user credentials.

4.5 Access Control in XML

Due to the separation of content and presentation, focus can be given to protecting the information content without being concerned with the presentation of information. Since the tags that define the contents can be created by the author of the document, specific semantics can be associated with each tag that can be used to interpret the document and its security requirements (Botha & Eloff, 2002).

Damiani et al. (2002) have demonstrated that is possible to prune an XML document tree using XPath expressions to obtain the desired user view.

For example, suppose there is an online catalog document written in XML that lists available goods sold on the Internet. As the content is separated from the presentation information, access control policies could be applied to the content only. One access control policy could be that only premium members can view the special discount price information in the XML content document. When a regular member views the catalog, any information provided for the premium members would be hidden.

This kind of access control is known as fine-grained access control, allowing certain parts of a document to be accessible to certain users only. This is made possible by the fact that the tags defining the document are defined by the author of the document (Damiani et al., 2002).

Figure 4.2 on the next page shows how the user view can be obtained by pruning the document tree. As shown in figure 4.2 the content-based billing system serves an XML content document represented as a document tree. Due to the separation of content and presentation an XSLT stylesheet is then applied to the XML content document to display the content in an appropriate form to the user. When the XSLT stylesheet is applied to the XML content document an XPath expression is evaluated by the processor to determine if the user's credentials match the requirements to access the content nodes in the XML document. In the case of figure 4.2 the user has to be older than 18 to access node 4 in the XML content document's document tree. As the user's credentials state that the user is only 17 the user is denied access to node 4. Therefore the XML content document is pruned to form the user view which consists of nodes 1,2 and 3 as these are the only content nodes within the document tree that the user may access according to his credentials (Damiani et al., 2002).

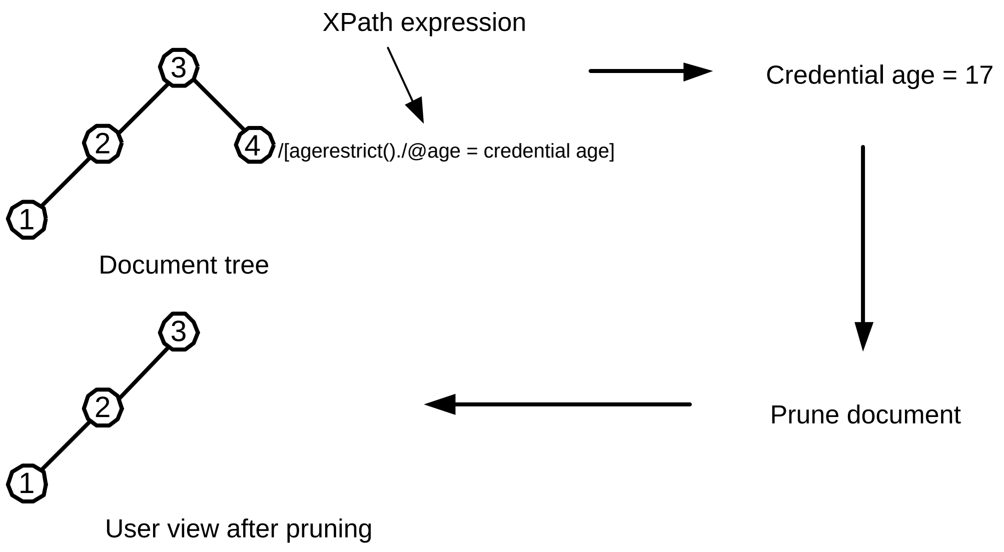


Figure 4.2: Pruning of the Document Tree

An emerging technology that could be used to control the user view, providing access to specified elements within the XML document is XML Access Control Language (XACL) (Kudo & Hada, 2000a).

4.5.1 XML Access Control Language (XACL)

XACL is an access control policy specification language that is a primary component of XML Access Control technology. Similar to existing policy languages, XACL is a language oriented around triplets of subject, object and action. The subject primitive can consist of user IDs, groups and/or roles. The object primitive is the secured data and the granularity of the object can be as fine as a single element within an XML document. The action primitive consists of four kinds of actions: read, write, create, and delete, and determines the actions that can be performed on the object primitive.

Moreover, XACL provides the notion of provisional actions that means provisions can be attached to the access decision. Suppose a “log provisional” action is specified in the access control rule of “Alice is allowed to read the salary field”. This basically means, “Alice is allowed to read the salary field, provided the access is logged.” The provisional authorization model provides more flexibility in specifying access control policies than is possible with traditional object-subject-action based semantics (Kudo & Hada, 2000b).

With this access control technology, the access control policies control how an XML document appears. The policies also insure the document is securely updated as specified by the system administrator. XACL could be used in the content-based billing model to provide access to specific elements within an XML document based on the user’s credentials which are supplied by a digitally signed XML document.

4.6 Conclusion

This chapter has discussed access control and identified three common forms of access control. Intrinsic to DAC, MAC and RBAC is the identity of the user. The content-based billing model does not require that the user is identified, instead it needs information about the user that can be used to provide access. Therefore credential-based access control was identified as a means of access control for the content-based billing model. This chapter then investigated digital certificates as a technology to provide credential-based access control. As the content-based billing model uses XML as a common environment for information exchange XML signature was identified as a technology that can be used to digitally sign an XML document that can be used to present the content-based billing system with the user’s credentials. Once the system has the user’s credentials it can use these credentials to provide access to XML content. Access to content is achieved by pruning the document tree so that the user’s view consists only of XML content that the user’s credential’s allow that user to view. As the user purchases content the user view is adjusted as needed. An XML technology that can be used to prune the user view as needed by the content-based billing model is XACL, which can provide access to specific elements within an XML document based on the user’s credentials.

The following chapter defines the scope of the content-based billing model and introduces the conceptual model, as well as outlining the further development of the content-based billing model.

Part II

CBiX : The Model

Chapter 5

Conceptual Model: CBiX

The importance of content for increasing revenues was highlighted in Chapter 2. It was argued that the key to increasing revenues through content is by implementing efficient billing systems. Next chapter 3 introduced XML as a platform for a billing model. Chapter 4 then investigated access control within a billing model and proposed that for a billing model credential-based access control would be more suited than traditional access control paradigms such as DAC, MAC or RBAC. Chapter 4 also introduced technologies to implement credential-based access control for the content-based billing model, and described how access control can be provided in an XML environment.

In this dissertation, the content-based billing model is hereafter referred to as CBiX (**C**ontent-based **B**illing **i**n **X**ML Environments).

This chapter provides the scope of the research and makes note of issues that were not addressed. Following the establishment of the scope of CBiX this chapter describes the conceptual foundation of CBiX. Next this chapter provides a conceptual overview of the development cycle of the CBiX billing model as well as describing a common architecture to support CBiX.

5.1 Scope of the Research

The CBiX billing model encompasses three main focus areas as depicted in figure 5.1. These are XML, Access Control and Web Business Models.

Figure 5.1 is then refined to show the specific components of Web Business Models, XML, and Access Control that are of interest. Web Business Models consists of two major areas of interest. The first is content-based

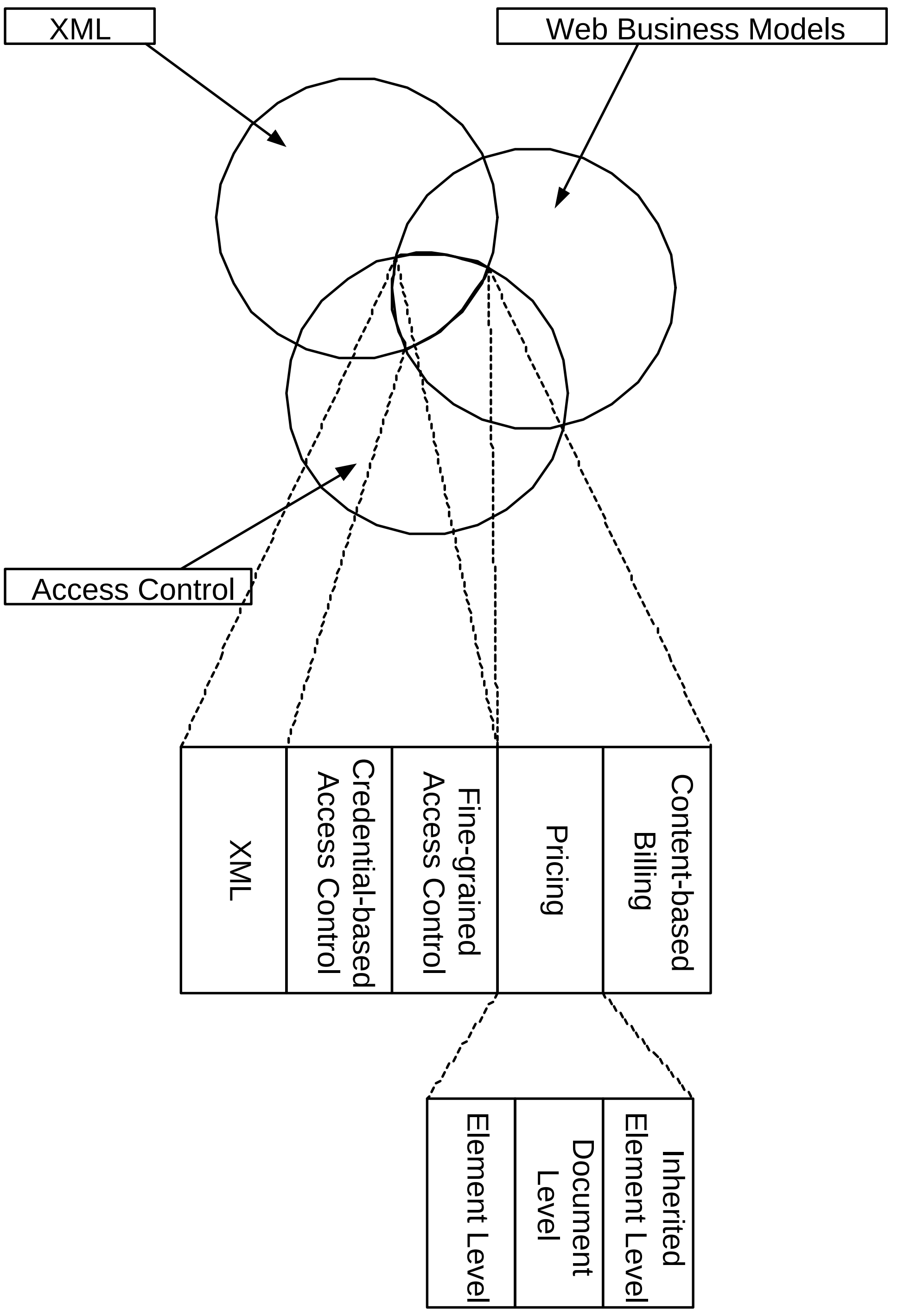


Figure 5.1: The Scope of the Research

billing which is the objective of the CBiX model. A billing model by its nature suggests a financial transaction. This leads to the second area of interest under Web Business Models, namely pricing. For the purposes of CBiX the area of pricing can be divided into three subsegments. The first, document level pricing, occurs where there exists a single price for the whole document. The second, element level pricing, occurs where there exists a price for each element within the whole document. The third subsegment is inherited element level pricing, where there exists a price for each element within the document, as well as a price for an element and all its child elements.

The second focus area is XML as the need for a common environment for communication is also shown by the focus given to supply chain integration by e-Enterprises. Extended value chain applications seek to extend the functionality of traditional Supply Chain Management (SCM) applications by integrating customer demand planning, supplier supply planning, and inbound/outbound logistical constraints to optimize performance and profits. In a typical SCM environment suppliers and customers are linked via expensive, tightly integrated extranets, SCM installations and Electronic Data Interchange (EDI). This technology is technologically complex and expensive, putting it out of reach of Small to Medium Enterprises (SMEs). In contrast extended value chains are Internet-based and use common standards such as XML as a basis. This allows information from customers, suppliers, and newly connected SMEs to be shared in a dynamic environment containing real-time business process facilities and shared data warehouses of information for decision support. This results in enterprises being able to bring inventory levels down to improve responsiveness to market conditions.

Fundamentally, extended value chain applications differ from traditional SCM systems through their ability to integrate easily and inexpensively with external suppliers, trading partners, and customers. This shows how important a common environment is for business taking place over the Internet (Hoque, 2000).

XML is the environment for the CBiX model. However, only those features of XML focussed on in Chapter 3 are used for CBiX.

The third focus area is Access Control. In chapter 4 it was shown that traditional access control models are not ideally suited for a billing model such as CBiX. Therefore in figure 5.1 it is depicted that the focus of the access control component of CBiX is narrowed to incorporate credential-based access control and fine-grained access control. Credential-based access control provides access to content based on the users credentials. The users credentials determine which content he is allowed to view, what the price of the content is and what payment strategy will be followed. Fine-grained access control is achieved by pruning the user view to provide access to content on the element level of an XML content document as dictated by the user's credentials.

There are however issues that fall outside the scope of the research and as such not addressed by the CBiX model. One such issue is the actual financial transaction. How the financial transaction via credit card or electronic money occurs and which payment protocols are to be followed are not addressed by CBiX. The choice of payment protocol will depend on the personal preference of the company selling the content. Another issue not specifically dealt with in CBiX is how the user presents his credentials to the billing system. Digital certificates are discussed in chapter 4 as an option for presenting verifiable credentials to the billing system. The choice of a credential delivery mechanism will be dependant on the technology availability and the personal preference of the company selling the content.

5.2 CBiX : Conceptual Foundation

In chapter 4 the author has identified three properties that the implementation of an access control service in general should adhere to based on the four requirements for collaborative environments identified by Bullock and Benford (1999).

Furthermore, the author maintains that to a certain extent these three principles also apply to a billing system, leading to an identified synergy between access control and billing. The author further demonstrated synergy between access control and billing by dividing access control and billing into three phases as depicted in chapter 4.

Throughout the three identified phases, similarities can be seen between access control and billing. Firstly, both access control and billing initially require identification and authentication to take place. It was also argued that in the case of a billing system there is a need for an access control paradigm that provides access based on user properties that can be used to determine access. This has led to the decision to use credential-based access control as the administrative paradigm of access control for CBiX. Both access control and billing then require the establishment of a session. Secondly, both access control and billing allow for an attempt to access a resource to allow the user to see which resources he can attempt to access.

Finally, both access control and billing require access resolution. They differ with respect to how this is done. In the case of access control role conflicts are evaluated into the highest set of permissions. However, in the case of billing credentials are resolved as a logical "and" where if one condition evaluates to false the whole expression will be false. Billing also has an additional requirement in that it must make provision for a monetary transaction to occur.

Due to the identified synergy between access control and billing as well as the simplicity, generality and clarity of existing access control models such as the RBAC96 model (Sandhu et al., 1996) the author has decided that existing access control models can be used as a conceptual basis for developing a billing model. The methodology followed while developing access control models, specifically the RBAC96 model (Sandhu et al., 1996), has provided a solid foundation and guideline for the development of CBiX. This has resulted in the development of the CBiX family of billing models.

5.3 CBiX : A Family of Billing Models

The CBiX model for content-based billing in XML environments will be developed as a family of models as depicted in figure 5.2. This will result in 4 different models, with CBiX$_1$ and CBiX$_2$ being developed in parallel. The architectural concepts are used to describe the high level design of CBiX.

5.3.1 CBiX$_0$: The Base Model

CBiX$_0$ is the first in the family of models. This is the base model. This model incorporates all the minimum requirements for a basic billing model, allowing other models in the family to build upon CBiX$_0$'s functionality.

CBiX$_0$ firstly provides support for identification and authentication. Following this will be the assignment of credentials. Next CBiX$_0$ will handle

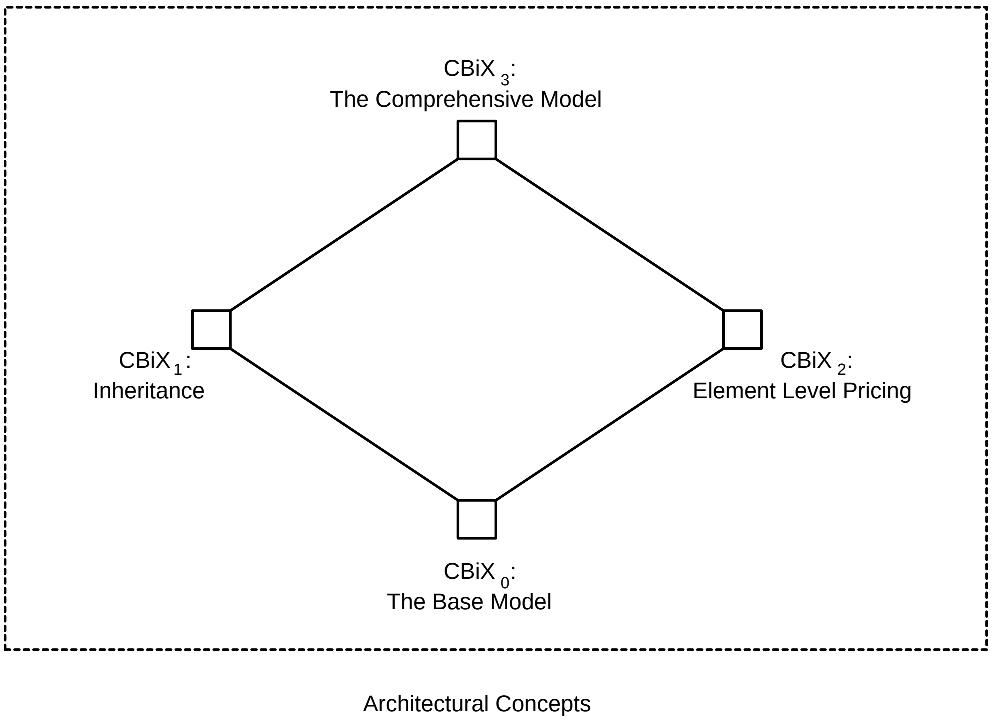


Figure 5.2: CBiX : A Family of Models

the establishment of a session. After a session has been established CBiX$_0$ will determine the user's view. CBiX$_0$ also provides support for pricing for content at the document level as depicted in figure 5.3. Figure 5.3 shows that in CBiX$_0$ there is one price for the whole content document. This price can vary depending on the user's credential set. Next CBiX$_0$ handles access resolution when the user attempts to access a resource.

CBiX$_0$ is developed as a basis for a more comprehensive model that would incorporate higher level features to make the model more commercially viable.

The next step is the concurrent development of CBiX$_1$ and CBiX$_2$.

5.3.2 CBiX$_1$: The Base Model with Inheritance

CBiX$_0$ by itself is a functional billing model, but it lacks certain features that would make is more useable and commercially viable.

CBiX$_1$ starts to address these needs by incorporating inheritance into the base functions it inherits from CBiX$_0$. This allows child elements within the

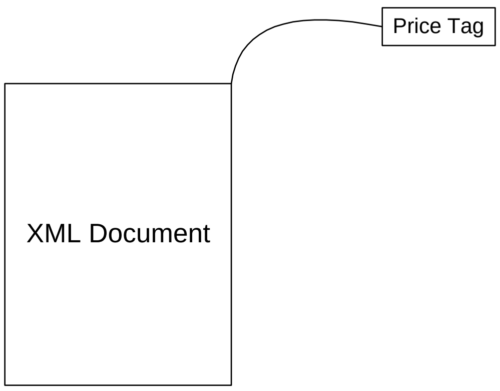


Figure 5.3: CBiX_0: The Base Model

user view to inherit access permissions from their parent elements. Figure 5.4 depicts the changes from CBiX_0. The document now contains content elements. Access can be given to each element in turn, and access rights can be inherited due to the added inheritance functionality added by CBiX_1. The document is still priced as a complete entity, with there being one price for the whole document.

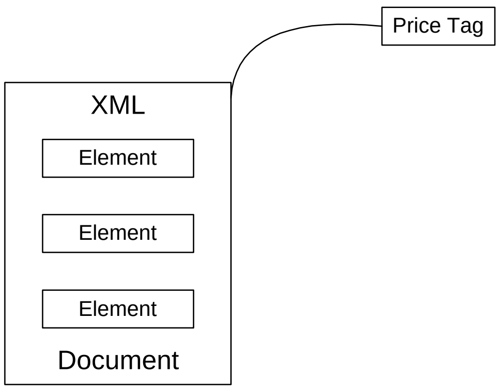


Figure 5.4: CBiX_1: The Base Model with Inheritance

Incorporating inheritance will give the CBiX model much greater functionality.

5.3.3 CBiX_2: The Base Model with Element Pricing

Various pricing structures for the Internet, and the importance of proper pricing is discussed in chapter 2. Because of the importance of pricing CBiX_2

evolves the document level pricing found in CBiX$_0$ into element level pricing, where each content element in a content document is priced individually. Figure 5.5 shows the evolution in pricing from CBiX$_0$, with an individual price being assigned to each element within the document. These prices can vary from user to user, depending on the user's credential set.

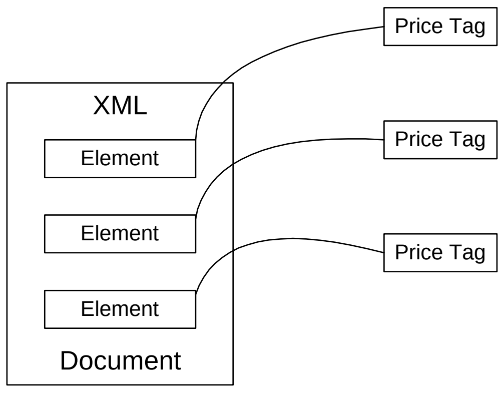


Figure 5.5: CBiX$_2$: The Base Model with Element Pricing

Incorporating element level pricing in CBiX$_2$ gives the CBiX billing model greater flexibility. This gives the model much greater commercial appeal, and enables users of the model to develop and implement more flexible and efficient solutions.

Following the concurrent development of CBiX$_1$ and CBiX$_2$, CBiX$_3$ is developed.

5.3.4 CBiX$_3$: The Comprehensive Model

CBiX$_3$ is the comprehensive model. This model brings together CBiX$_1$ and CBiX$_2$ to create a fully functioning billing model that incorporates all the features of CBiX$_0$ as well as including inheritance and inherited element level pricing. Figure 5.6 depicts the comprehensive model with each element having its own price, as well as there being a price for a number of elements where you pay one price for the selected element and all its child elements. This aspect is depicted on the figure by a circle surrounding the selected element and the inherited element, with both elements then having only one price.

This model provides an effective billing solution while also providing the

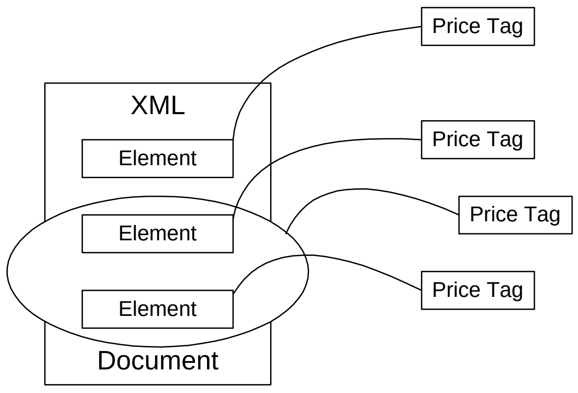


Figure 5.6: $CBiX_3$: The Comprehensive Model

user with features that allow for much greater flexibility in implementing a billing solution.

In addition to the family of billing models, there is an architecture that serves to describe the high-level design of the CBiX family of models. This architecture provides a high-level design and establishes basic concepts for the CBiX family of models.

5.4 The CBiX Architecture

The CBiX architecture is divided into two main sub-systems. The first sub-system is the *administrative backend.* This sub-system is used to transform the supplied content from a content provider into CBiX compliant content.

The second sub-system is the *user frontend.* This sub-system is the web component of the system that provides the user with content and bills the user for the provided content.

5.4.1 The Administrative Backend

The content provider provides the content. As depicted in figure 5.7 on the following page the provided content from the content provider consists of XML documents that contain the actual content, and XML schemas that validate the XML documents. Each XML schema represents a specific document class. A document class represents a specific type of content, for example, articles, presentations, exercise and solution sets, question papers. Each document class schema can validate multiple XML content documents.

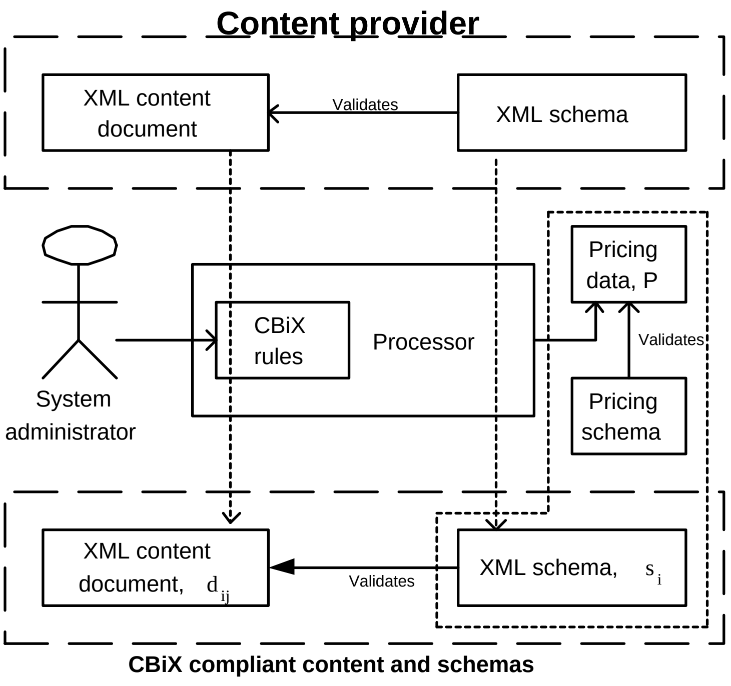


Figure 5.7: The Administrative Backend

The provided XML content document and document class schema is then parsed through a processor as depicted in figure 5.7 to make both the XML content documents and the document class schemas CBiX compliant. This results in a set S of document class schemas of which a document class schema s_i is shown on figure 5.7, and a set of D of XML content documents of which XML content document d_{ij} is shown on figure 5.7. Each XML content document consists of a number of elements.

Figure 5.7 shows that the system administrator chooses settings in the administration backend interface to set up the CBiX rules. These CBiX rules are used by the processor to modify the XML content documents into CBiX compliant XML content documents. The processor also modifies the document class schemas so that they are CBiX compliant and can correctly validate the CBiX compliant XML content documents.

The system administrator also uses the administrative backend interface to generate the pricing data and the validating pricing schema for a specific document class. This pricing data is used in conjunction with the user's credentials to generate a price for content in the content display area (CDA) of the user view. The CBiX compliant content documents, the CBiX compliant

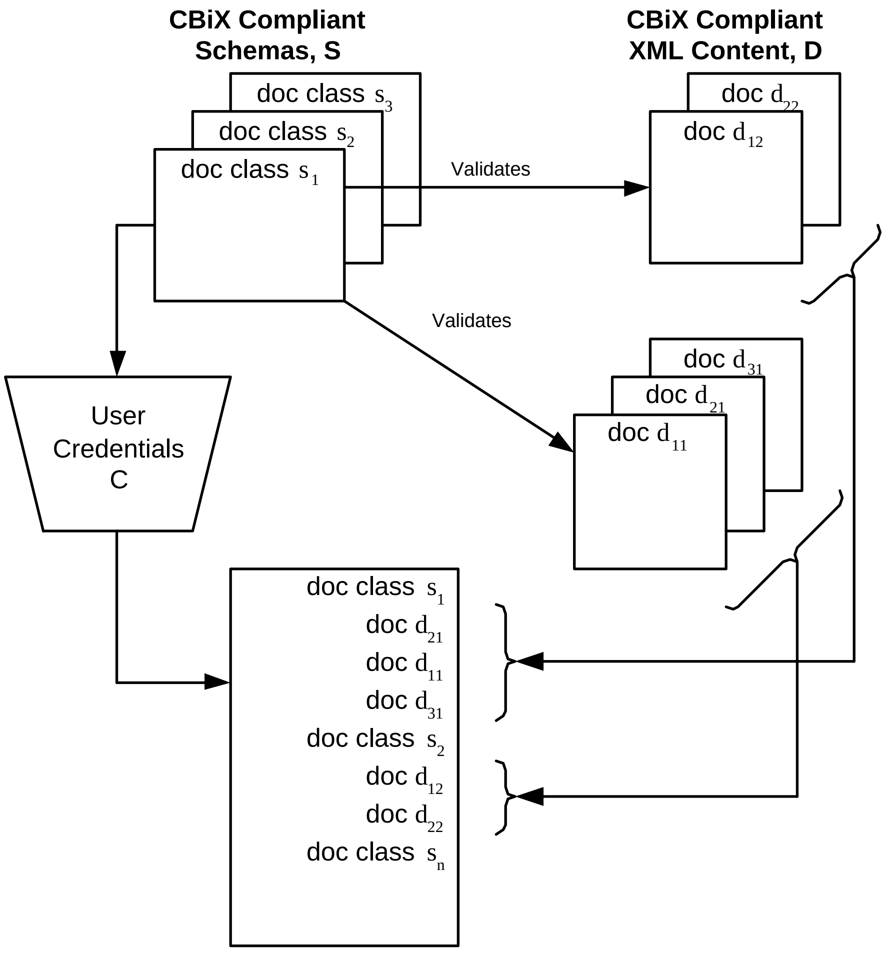


Figure 5.8: The User's Navigation Bar

document class schemas, the pricing data and the pricing schema are then used in the user interaction.

5.4.2 The User Frontend

The user presents his digital certificate containing his credentials on arrival at the CBiX web site. An XML document containing his relevant credentials is generated. This document validates against the XML schema for CBiX credentials.

The CBiX web site's user frontend allows the user to navigate to any content the user's credentials allow him to view or purchase by utilizing the links on the navigation bar of the web site. The remaining area of the web page is dedicated to the display of any public (free) content and content that

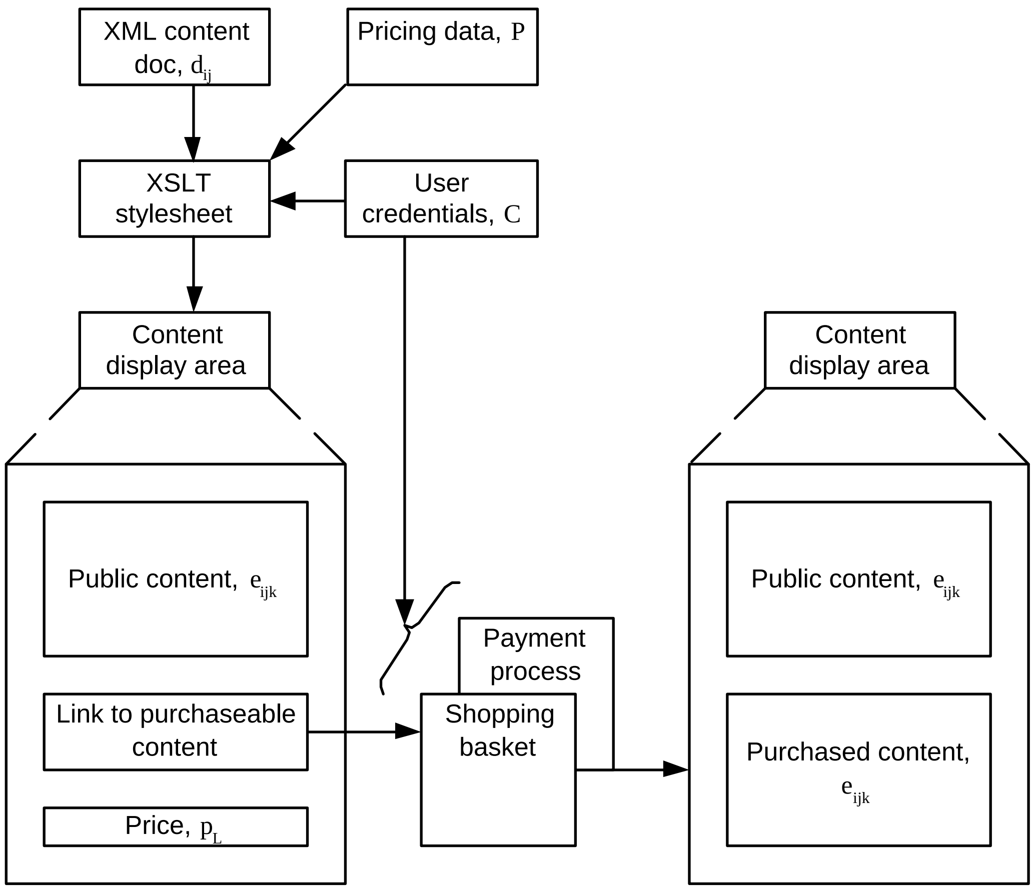

XML content doc, d_{ij}
Pricing data, P
XSLT stylesheet
User credentials, C
Content display area
Public content, e_{ijk}
Link to purchaseable content
Price, p_L
Payment process
Shopping basket
Content display area
Public content, e_{ijk}
Purchased content, e_{ijk}

Figure 5.9: The User's Content Display Area

the user has purchased. The user interface of the web site only allows the user to view options that his credentials allow him to view, eliminating any unnecessary or restricted information.

The *web-based user frontend* is depicted by two figures. Figure 5.8 on page 79 depicts the navigation bar section of the user view, and figure 5.9 on the facing page depicts the content display area (CDA) section of the user view.

The navigation bar menus, as depicted in figure 5.8, are generated from the set S of document class schemas and the set D of XML content documents. The document class headings s_1 to s_n are generated from the set S of document class schemas. Each document class represents a type of content. The document class heading will only be generated if the user's credentials allow the user to view that content. Nested under each document class heading are links to XML content documents. These are the XML content documents that are validated by the document class schema that was used to generate the document class heading that the links are nested under. For example, under the document class heading s_1 links to the XML content documents d_{11} to d_{i1} are found. The navigation bar as depicted in figure 5.8 is suited for display on a display device such as a PC monitor. The navigation bar's basic principles will remain the same, but its presentation will differ depending on the device the content is being delivered to. For example, a mobile phone or PDA have much smaller displays than a PC and therefore will require an alternatively structured navigation bar.

Once a user selects a link in the navigation bar the content display area (CDA), as depicted in figure 5.9, is used to display the content in the linked XML content document.

The XML content document, the user's credentials and the pricing data for the document class schema that the XML content document validates against are all processed by an XSLT stylesheet. The XSLT stylesheet is used to present the XML content in an appropriate format. Initially the CDA displays any public (free) content found in the XML content document, a link to the rest of the content in the XML content document that has to be purchased, and the price of the content that has to be purchased.

Once the link to the purchased content is followed, a purchasing mechanism is selected depending on the user's credentials. If the user has a positive credit rating as one of his credentials the content he wishes to purchase will be added to a shopping basket as he is a trusted customer. However, if the user has a negative credit rating, or has made no previous purchases from the site, he will have to enter into a payment process.

After the content has been added to the shopping basket, or paid for, depending on the user's credentials, the CDA is updated. The CDA now displays the public content, as well as the purchased content.

5.5 Conclusion

This chapter has defined the scope of CBiX and has also shown a synergy between access control and billing. The synergy between access control and billing has led to the adoption of the developmental methodologies of traditional forms of access control, with specific reference to Sandu's RBAC96 (Sandhu et al., 1996) model. This has led to the development of the CBiX family of billing models with an architecture to describe the high level design of CBiX .

The following chapter describes CBiX_0, the first in the family of models. Chapter 7 then focuses on the parallel development of CBiX_1 and CBiX_2, concluding the family of models with CBiX_3.

Chapter 6

CBiX$_0$: The Base Model

This chapter formally defines the first in the CBiX family of models, CBiX$_0$, the base model. The chapter commences by defining the common components of the CBiX model. This chapter then develops CBiX$_0$ by defining the functions used by each of the three major aspects of CBiX$_0$. All of the defined components and functions are used by the CBiX family of models as a basis for further development.

For the purpose of formalizing the CBiX model the author has chosen to adopt the Z notation as presented by Jacky (1994). This choice is due to the Z notation's wide use and easily understood formal notation. Complete and concise descriptions of the CBiX family of models can be achieved by using Z. However, the Z specifications presented here have not been verified by any formal means, nor does the author claim it to be a completely executable model. Nevertheless, it does reduce ambiguity and provide a sufficiently clear exposition of the model. The key aspects of the Z notation as used in this dissertation are summarized in appendix D.

6.1 Common Components

CBiX is concerned with the sale of XML content. This XML content is incarnated as XML documents that contain information worth selling. For the purposes of distinguishing these documents from other XML documents (such as XML schemas) we adopt the term XML content document, or, in short, content documents.

The type of content that is represented within an XML content document is determined by the schema that is used to validate that XML content document. The validating schema represents the document class for all the XML content documents that are validated by that specific schema. Consider, for example, a content provider that sells various academic materials. These materials may include books, articles, lecture notes, old exam papers and slide sets. Each of these types of materials have informal semantics associated with them through everyday English use. These semantics are formalized, structurally at least, by schema documents that define what the content documents would comprise. The "article" schema documents would thus be used to validate content documents claiming to be "articles". The document class can therefore be argued to be a basic type:

Definition 6.1.

$[DOCUMENTCLASS]$

Schemas without content documents based on those schemas would not be sensible (although for a limited time during administration it would be required). The DOCUMENTCLASS of content documents can represent many different types of content such as "articles", "presentations", "question papers".

A content document is defined in terms of its elements. At this stage we don't want to concern ourselves with the hierarchy of elements in an XML document; we merely view a document as a set of elements, each element having a specific element type. The relationship between a content document and its elements are formally defined as follows:

Definition 6.2.

$[ELEMENT]$

$DOCUMENT == \mathrm{seq}\, ELEMENT$

An element type is formally described as follows:

Definition 6.3.

$[ELEMENTTYPE]$

$GetType : ELEMENT \mathrm{dom}\, ELEMENTTYPE$

For example, every document that is an “article” could conceivably consist of multiple elements such as an image, text and a code snippet. Each of these different content elements are a different element type, with each “article” possibly containing a number of content elements of each element type. The GetType function is used to return the ELEMENTTYPE of a selected ELEMENT.

Just as every element type has an associated element, every element belongs to a specific content document. A content document will belong to a specific document class. The relationship between a content document and its describing document class is formally described as follows:

Definition 6.4.

$$\mathit{GetDocClass} : \mathit{DOCUMENT} \leftrightarrow \mathit{DOCUMENTCLASS}$$

GetDocClass is being used to return the DOCUMENTCLASS of a DOCUMENT. For example, if a content document contains content of the type “article”, it will be associated with an “article” DOCUMENTCLASS. GetDocClass is used in later functions to retrieve the DOCUMENTCLASS of a DOCUMENT.

Access control has to be performed on a content document as well as the content elements within that content document. In chapter 4 credential based access control was proposed as the preferred access control mechanism for CBiX. We therefore define a credential as being a basic type:

Definition 6.5.

$$[\mathit{CREDENTIAL}]$$

A credential is a user property that is used to perform access control. Each user has a set of credentials. These credentials determine what content the user is allowed to view. For example, a user who is 17 years old would not be allowed to view content that is age restricted for people 18 years and older. The users credentials also determine the flow of the payment process. For example, if a user’s credentials define that the user has a bad credit rating then the payment process could require that the user pays for the content before receiving it. If, on the other hand, the user has a good credit rating the user may be allowed to view or download the content first adding it to his shopping basket and only paying on completion of all his transactions. A shopping basket is defined as follows:

Definition 6.6.

$$SHOPPINGBASKET == \mathrm{seq}\, ELEMENT$$

A shopping basket is instantiated for a user when visiting the site to facilitate, and is simply a representation of a number of content elements that have been selected by the user for purchase.

Each of the above common architectural components can be instantiated as part of the CBiX components schema:

Definition 6.7.

CBiXComponents

$$S : \mathbb{P}\, DOCUMENTCLASS$$
$$E : \mathbb{P}\, ELEMENT$$
$$ET : \mathbb{P}\, ELEMENTTYPE$$
$$D : \mathbb{P}\, DOCUMENT$$
$$C : \mathbb{P}\, CREDENTIAL$$
$$SB : \mathbb{P}\, SHOPPINGBASKET$$

When a developer is using the CBiX billing model to develop a billing system, all the common components will be used within the billing system. The developer will also have to write functions to add, edit and delete the various objects associated with the common components. The author has not defined the add, edit and delete functions for the common components as this is deemed standard functionality that a developer can easily develop without the need for formal specifications. It also does not contribute to the essence of the model.

6.2 The Aspects of CBiX$_0$

CBiX$_0$ incorporates three major aspects: access control, XML, and pricing.

Consider each in turn.

6.2.1 Access Control

The access control aspect of CBiX$_0$ incorporates credential-based access control and fine-grained access control. The credential-based access control aspect utilizes user credentials to determine which content the user is allowed to access and what the price for the accessed content is. The fine-grained access control aspect is used to prune the XML document tree of the XML content document into the user view that is permitted by the user's credentials.

CBiX$_0$ firstly defines a function to determine if a user may access an element. The second defined function is used to assign a set of credentials to an element of a content document. The third function defined for the access control aspect returns a set of elements based on a credential set.

The Access function is used to determine whether a user can access a content element based on the user's credential set. The Access function schema is therefore defined as follows:

Definition 6.8.

Access

$$access : ELEMENT \leftrightarrow \mathbb{P}\, CREDENTIAL$$

For example, to access a content element with the credential requirement of (student), a user would need to have a credential within their credential set stating that they are a student.

The AssignCredential function assigns a set of credentials to an element in order to perform access control on the element. The AssignCredential function is defined as follows:

Definition 6.9.

$$\begin{array}{|l}
\hline
\textit{AssignCredential} \\
\Delta \textit{Access} \\
\textit{element}? : \textit{ELEMENT} \\
\textit{credentials}? : \mathbb{P}\,\textit{CREDENTIAL} \\
\hline
\textit{access}' = \textit{access} \oplus \{(\textit{element}?, \textit{credentials}?)\} \\
\hline
\end{array}$$

An example of the assigned credential set could be {student,WIH1000}. This set would be assigned to an element and used to determine access. Only if the user has both a student and WIH1000 credential in his credential set will the user be granted access to the element.

The GetElements function returns a set of elements. This set of elements consists of elements that the user may access according to the user's credential set. An element is added to the set of elements that is returned if the relevant user credentials match the credential set that is assigned to the element. The element is assigned credentials by the AssignCredential function. The GetElements function is defined as follows:

Definition 6.10.

$$\begin{array}{|l}
\hline
\textit{GetElements} \\
\textit{elements}! : \mathbb{P}\,\textit{ELEMENT} \\
\textit{credentials}? : \mathbb{P}\,\textit{CREDENTIAL} \\
\textit{Access} \\
\hline
\exists\, \textit{assignedcred} : \mathbb{P}\,\textit{credentials}? \bullet \\
\qquad \textit{elements}! = \textit{access} \,(\!|\, \textit{assignedcred} \,|\!) \\
\hline
\end{array}$$

For example, consider a user with a credential set of {student, WIH1000, DEV1000, full-time}. Also consider a set of three content elements. These content elements have the following assigned credential sets. The first content element, {student, WIH1000}, the second content element {student, DEV1000,full-time} and the third content element {student, WIH1000, DEV1000, part-time}. The GetElements function would only return content elements one and two for the user as the user's credential set does not meet the access requirements assigned to content element three.

So, in essence the GetElements function checks whether the required credentials for accessing an element is a subset of the user's set of credentials.

6.2.2 XML

The XML aspect of CBiX$_0$ consists of XML as the platform for information delivery. XML is used to deliver the content that is to be sold by the billing system. XML schemas are used to validate the XML content documents. All content is validated and conforms to CBiX$_0$'s requirements so that access control and content delivery can be performed correctly. The common components previously defined describe the XML function of CBiX$_0$.

6.2.3 Pricing

The pricing aspect of CBiX$_0$ consists of a document level pricing strategy. In this strategy each document class is assigned a price for a specified set of credentials. Therefore a user will pay a fixed price for a content document based on the document class of that document and the user's credential set. For example, if a document class of type "Exercises and Solutions" exists and is assigned a set price of R10 for user's that are students and R20 for users that are non-students, a user that is a student will therefore pay R10 for any content document based on that document class.

For the pricing aspect of CBiX$_0$ three functions are defined. The first of these functions, AddPrice, adds a price for a specific type of content and credential set into a Price set. The second defined function, GetPrice, is used to retrieve the price of a specific type of content for a user, depending on the user's credentials. The third defined function defines the axiomatic description of a user's ShoppingBasket, whereby the size of the ShoppingBasket is determined according to the user's credit rating credential.

Firstly, to enable the AddPrice function, the Price schema is defined. This schema defines the price data set that is used to store the relevant pricing data. The Price schema is defined as follows:

Definition 6.11.

Price

$$Price = (\mathbb{Z}, DOCUMENTCLASS, CREDENTIAL)$$

The price schema's price data set consists of a combination of a monetary amount, a DOCUMENTCLASS and a CREDENTIAL set.

The AddPrice function adds a price amount for a specific document class and set of credentials to a price data set. If the price amount for a specific document class and credential combination exists the amount is replaced. The function AddPrice is defined by the following schema:

Definition 6.12.

AddPrice

$\Delta PricingSchema$
$docclass? : DOCUMENTCLASS$
$credential? : \mathbb{P}\ CREDENTIAL$
$amount? : \mathbb{Z}$

$price' = price \oplus \{(amount?, docclass?, credential?)\}$

The AddPrice function is necessary to allow a number of prices to be added into a data set for a content document of a specific document class. Each price for a document class is stored with a credential combination and an amount. The user's credential set determines which price amount is retrieved for a selected content document which belongs to a specific document class.

The AddPrice schema populates the price data set. An example of an entry could be as follows: (R20, QuestionPaper, {student, WIH1000}). This entry can be interpreted as follows; a user will pay R20 for content with a DOCUMENTCLASS type of QuestionPaper, if the user has as part of his credential set the facts that he is a student and takes the course WIH1000.

The function GetPrice retrieves the price from the Price set for a specific content document as determined by the document class of the document and the user's credentials. The function *GetPrice* is defined by the following schema:

Definition 6.13.

$$\begin{array}{|l}
\textit{GetPrice} \\
\hline
\textit{amount}! : \mathbb{Z} \\
\textit{credentials}? : \mathbb{P}\ \textit{CREDENTIAL} \\
\textit{document}? : \textit{DOCUMENT} \\
\hline
\exists\ \textit{assignedcred} : \mathbb{P}\ \textit{credentials}? \bullet \\
\qquad (\textit{amount}!, \textit{GetDocClass}(\textit{document}?), \textit{assignedcred}) \in \textit{Price} \\
\hline
\end{array}$$

The GetPrice function retrieves the price amount from the price data set for a specific content document, as determined by the document class of the content document and the set of user credentials. The price amount for a document class and credential set combination is added into the price data set by the AddPrice function.

The GetPrice function works as follows. If the price data set holds the entry, (R20, QuestionPaper, {student,WIH1000}) and the user has the credential set {student, WIH1000, DEV1000}, the user will be granted access to content document having a DOCUMENTCLASS of QuestionPaper and pay R20 for the content document as the user has the required credentials.

A user's shopping basket has a size. To determine the size of the shopping basket the function receives a user credential representing the user's credit rating. A user's shopping basket SB_i, an element of SB, is therefore constrained by:

Definition 6.14.

$$\begin{array}{|l}
SB_i : SB \\
\textit{MAXVALUE} : \mathbb{Z} \\
\textit{cred} : \textit{CREDENTIAL} ::= \textit{goodcreditrating} \mid \textit{badcreditrating} \\
\hline
\textit{if}\ (\textit{cred} = \textit{badcreditrating})\ \textit{then}\ \#SB_i = 1\ \textit{else}\ \#SB_i = \textit{MAXVALUE}
\end{array}$$

The size of a user's shopping basket, SB_i, determines how many content elements SB_i can contain before the user has to pay for the content. If the user has a 'good credit rating' credential, SB_i has a size greater then 1 as the user is trusted to pay for the shopping basket. The maximum size of the shopping basket is then equivalent to a variable MAXVALUE. The

MAXVALUE variable is the maximum size of a shopping basket. The value of this variable is determined by the technical limitations associated with the selected shopping basket mechanism. If the user has a ‘bad credit rating’ credential the user’s shopping basket, SB_i, is limited to a size of 1 as this user is not trusted. This can be due to a number of causes such as payment problems in the past, or the fact that this is a first time user of the billing system, and therefore an unknown risk. A shopping basket with a size of 1 ensures that the shopping basket registers as being full after the addition of only one content element, thereby requiring the user to pay for the content before receiving it.

6.3 Conclusion

In this chapter CBiX$_0$ has been defined using Z notation. CBiX$_0$ is the base model for the CBiX family of models. All the other models in the family build on CBiX$_0$, further extending the defined common components and functions. As depicted in figure 6.1 CBiX$_0$ sells whole content documents at a time, with the price associated with the document varying according to a user’s credential set.

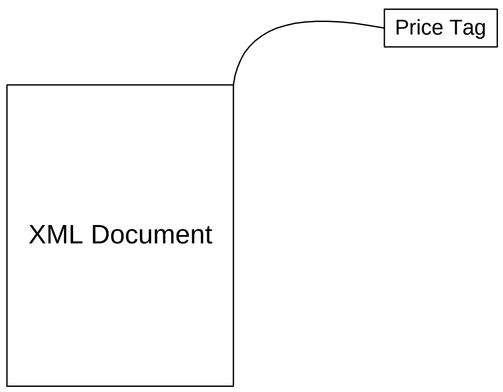


Figure 6.1: CBiX$_0$: The Base Model

Following the development of CBiX$_0$, CBiX$_1$ and CBiX$_2$ are developed in parallel. The next chapter extends CBiX$_0$ by defining CBiX$_1$, CBiX$_2$ and CBiX$_3$.

Chapter 7

CBiX$_1$ to CBiX$_3$: Extending the Model

Chapter 6 defined CBiX$_0$ which is the first in the CBiX family of billing models. CBiX$_0$ forms the basis for the further development of CBiX$_1$, CBiX$_2$ and CBiX$_3$.

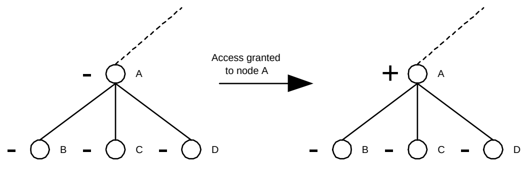
This chapter firstly defines CBiX$_1$ which builds on the functionality of CBiX$_0$ by adding inheritance. Following the definition of CBiX$_1$, this chapter defines CBiX$_2$, which is developed in parallel to CBiX$_1$. CBiX$_2$ enhances the CBiX family of billing models by developing the document level pricing defined in CBiX$_0$ into element level pricing. This chapter lastly defines CBiX$_3$, the comprehensive model, which combines the features of CBiX$_1$ and CBiX$_2$, redefining the CBiX$_1$ and CBiX$_2$ functions as necessary.

7.1 CBiX$_1$: The Base Model with Inheritance

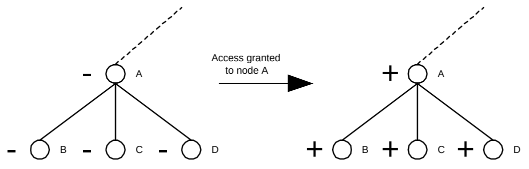
CBiX$_0$ by itself is a working billing model, but it lacks certain features that would make it more flexible. CBiX$_1$ is defined to build on the functionality of CBiX$_0$ by adding inheritance.

7.1.1 Inheritance

Inheritance in the context of CBiX allows child elements within a content document to inherit access permissions from parent elements. Figure 7.1(a) on the next page depicts the lack of inheritance in CBiX$_0$. In figure 7.1(a)



(a) CBiX$_0$: The Base Model



(b) CBiX$_1$: The Base Model with Inheritance

Figure 7.1: A Depiction of Access Control With and Without Inheritance

the user is granted access to node A. The user now has access rights to node A, but the user does not inherit the access rights to node A's child nodes (B, C and D), and is therefore denied access to nodes B, C and D. Figure 7.1(b) depicts the implementation of inheritance in CBiX$_1$. In figure 7.1(b) the user is granted access to node A. The user now inherits the access rights to node A's child nodes (B, C and D), granting the user access to nodes B, C and D. Incorporating this feature will extend the CBiX model's administrative powers. Figure 7.2 depicts the addition of inheritance to CBiX$_0$, where the document is now depicted as containing a number of content elements, with child elements being able to inherit access rights from their parent elements. The content document is still priced as a single entity, with the price being able to vary from user to user, depending on the user's credential set.

7.1.2 CBiX$_1$ Inheritance Functions

Within a content document child elements can inherit access rights from their parent elements. Child elements within the content document can have an overriding access permission set if needed.

The ElementHierarchy relation is defined to enable inheritance, describing the relationship between elements within a content document. Following the

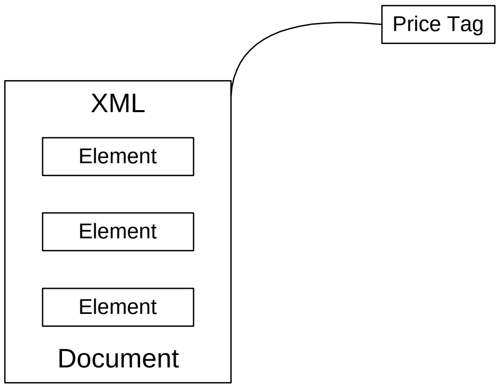


Figure 7.2: CBiX$_1$: The Base Model with Inheritance

definition of the ElementHierarchy relation, the GetAllDescendants schema is defined to return all the children of a specified element.

Elements are related to other elements within a document in an element hierarchy. The ElementHierarchy relation is defined in the following axiomatic description:

Definition 7.1.

$$\mathit{ElementHierarchy} : \mathit{ELEMENT} \leftrightarrow \mathit{ELEMENT}$$

$$\forall\, e : \mathit{ELEMENT} \bullet \#\mathit{ElementHierarchy}\,(e) = 1$$

To be able to get all of the descendants of an element we can refer to the transitive closure of the ElementHierarchy relation, which in Z notation is expressed as $\mathit{ElementHierarchy}^{+}$. The element hierarchy is needed to determine the relationship between two elements. If the element hierarchy is known then it can be determined which element is a parent of another element. The transitive closure over the ElementHierarchy relation is used in the GetAllDescendants schema.

All the descendants of a particular element that a user with a specific set of credentials may access can thus be described by the following GetAllDescendants schema:

Definition 7.2.

GetAllDescendants

$$Credentials? : \mathbb{P}\, CREDENTIALS$$

$$topelement? : ELEMENT$$

$$Elements! : \mathbb{P}\, ELEMENT$$

$$E_1 = ElementHierarchy^{+}(topelement?)$$

$$\exists\, Assignedcred : \mathbb{P}\, Credentials \bullet E_2 = access \,(\!|\, Assignedcred \,|\!)$$

$$Elements! = E_1 \bigcap E_2$$

This function takes the user's credential set and the selected element as inputs. It then returns a single element, if the element has no child elements, as allowed by the user's credential set. Otherwise, if the element has child elements, the child elements are returned as well, if the parent element may be accessed by the user according to the user's credential set. The transitive closure of the ElementHierarchy relation is used within the schema to get all the descendants of the element that the schema takes as input. The input credentials are the user's credential set and are compared to the credentials required by the selected element to determine if the user is allowed access.

7.2 CBiX$_2$: The Base Model with Element Pricing

CBiX$_2$ is developed in parallel with CBiX$_1$. CBiX$_2$ enhances the CBiX family of billing models by developing the document level pricing defined in CBiX$_0$ into element level pricing.

7.2.1 Element Pricing

Element level pricing allows the user to purchase specific elements within a content document instead of the whole content document as with the document level pricing strategy offered in CBiX$_0$.

With element pricing a price is assigned to an element type and credential set pair. Every element that is of a specific element type is priced according to its element type and the user's credential set. For example, if element type "Exercises and Solutions" exists, a price of R5 can be assigned to that element type for a credential set that specifies a user must be a student, and a price of R7 can be assigned to that element type for a credential set that specifies a user must be a non-student user.

Figure 7.3 depicts the evolution in the pricing aspect from CBiX$_0$. Now, instead of the whole document being assigned a price, each content element is assigned a price. As with pricing for the whole document, the price per element can vary from user to user depending on the user's credential set.

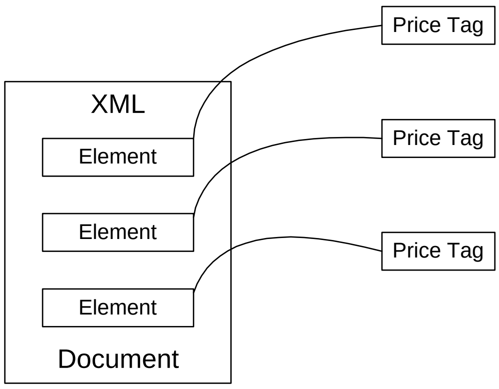


Figure 7.3: CBiX$_2$: The Base Model with Element Pricing

To upgrade document level pricing to element level pricing a number of CBiX$_0$ system functions need to be extended and redefined.

7.2.2 CBiX$_2$ System Functions

The ShoppingBasket function as defined in CBiX$_0$ remains unchanged for CBiX$_2$. The function AddPrice as defined in CBiX$_0$ is redefined to add the price for a specified content element type and a set of user credentials, instead of a price for a content document and user credential set combination. Furthermore, the function GetPrice as defined in CBiX$_0$ is also redefined for element level pricing.

Firstly, to enable the redefined AddPrice function, the Price schema must be redefined. This schema defines the price data set that is used to store the relevant pricing data. The Price schema is defined as follows:

Definition 7.3.

$$\begin{array}{|l}
\textit{Price} \\
\hline
\textit{Price} = (\mathbb{Z}, \textit{ELEMENTTYPE}, \textit{CREDENTIAL}) \\
\hline
\end{array}$$

The redefined price schema's price data set now consists of a combination of a monetary amount, an ELEMENTTYPE and a CREDENTIAL set.

The AddPrice function adds a price amount for a specific element type and a set of credentials to a price data set. If the price amount for a specific element type and credential combination exists the amount is replaced. The function AddPrice is defined in the following schema:

Definition 7.4.

$$\begin{array}{|l}
\textit{AddPrice} \\
\hline
\Delta \textit{PricingSchema} \\
\textit{elementtype}? : \textit{ELEMENTTYPE} \\
\textit{credential}? : \mathbb{P}\, \textit{CREDENTIAL} \\
\textit{amount}? : \mathbb{Z} \\
\hline
\textit{price}' = \textit{price} \oplus \{(\textit{amount}?, \textit{elementtype}?, \textit{credential}?)\} \\
\hline
\end{array}$$

The AddPrice function is necessary to allow a number of prices to be added into a data set for a single content element of a specific element type. Each price for an element type is stored with a credential combination and an amount. The user's credential set determines which price amount is retrieved for a selected element which belongs to a specific element type.

The AddPrice schema populates the price data set. An example of an entry could be as follows: (R5, ShortQuestion, {student, WIH1000}). This entry can be interpreted as follows; a user will pay R5 for a content element with an ELEMENTTYPE of ShortQuestion, if the user has as part of his credential set the facts that he is a student and takes the course WIH1000.

The function GetPrice, defined in CBiX$_0$ to return the price for a content document, is redefined for CBiX$_2$ to return the price for individual elements. The function GetPrice is defined by the following schema:

Definition 7.5.

```
GetPrice
  amount! : ℤ
  credentials? : ℙ CREDENTIAL
  element? : ELEMENT
  ───
  ∃ assignedcred : ℙ credentials? •
      (amount!, GetType(element?), assignedcred) ∈ Price
```

The GetPrice function retrieves the price amount from the price data set for a specific element, as determined by the element type of the element and the set of user credentials. The price amount for an element type and credential set combination was added into the price data set by the AddPrice function.

The GetPrice function works as follows. If the price data set holds the entry, (R5, ShortQuestion, {student, WIH1000}) and the user has the credential set {student, WIH1000, DEV1000}, the user will be granted access to element of ELEMENTTYPE ShortQuestion, and pay R5 for the element as the user has the required credentials.

7.3 CBiX$_3$: The Comprehensive Model

CBiX$_3$, the comprehensive model, combines the features of CBiX$_1$ and CBiX$_2$, redefining the CBiX$_1$ and CBiX$_2$ functions as necessary.

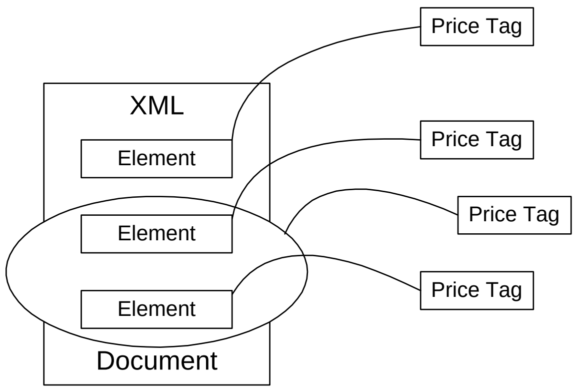


Figure 7.4: CBiX$_3$: The Comprehensive Model

Figure 7.4 depicts CBiX$_3$. CBiX$_3$ incorporates inheritance developed in CBiX$_1$ as shown by the elements within the content document. CBiX$_3$ also contains pricing for each element as developed by CBiX$_2$. CBiX$_3$ however also evolves the pricing aspect into inherited element level pricing. The circle on the figure depicts that a user can either pay a price for an element, or pay an alternative price for an element and all its child elements.

CBiX$_3$, the comprehensive model, as with CBiX$_0$, the base model, incorporates three major aspects: Access Control, XML and Pricing.

7.3.1 Access Control

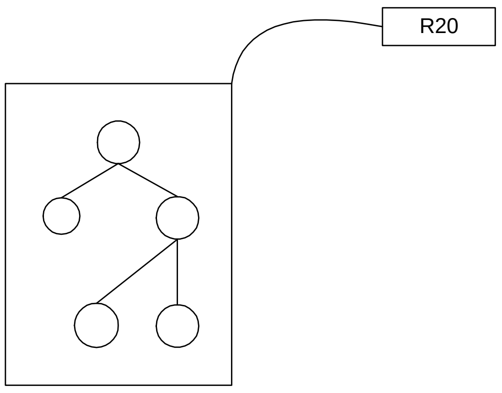
The access control aspect of CBiX$_3$ is based on the access control aspect of CBiX$_0$. CBiX$_3$ further enhances the access control aspect by incorporating the inheritance functions defined in CBiX$_1$ and replacing the GetElements function from CBiX$_0$ with the redefined GetElements function from CBiX$_1$.

7.3.2 XML

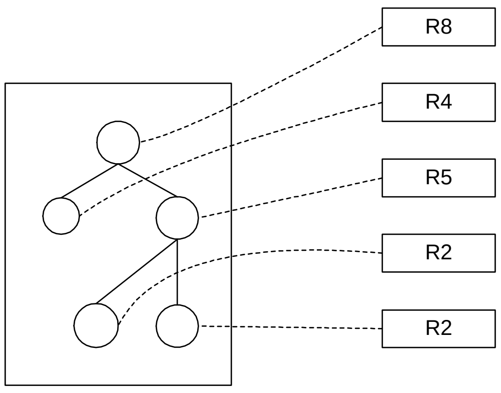
The functions that are used to define the XML aspect in CBiX$_0$ are used by CBiX$_3$, without extending or redefining these functions.

7.3.3 Pricing

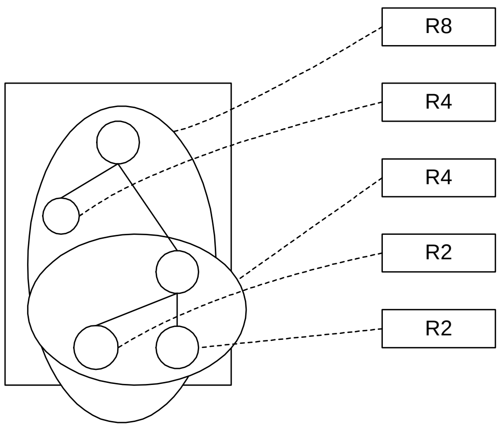
The pricing aspect of CBiX$_0$ consists of document level pricing as depicted in figure 7.5(a) on the next page. In document level pricing the user pays a set price for the entire content document as determined by the price set for that content document's document class and the user's credential set combination. For example, as depicted in figure 7.5(a), a user with a credential set specifying that the user is a student could pay R20 for the whole content document, while a user with a different credential set pays a different price for the whole content document. Document level pricing evolves into element level pricing in CBiX$_2$. Figure 7.5(b) on the facing page depicts the element level pricing of CBiX$_2$ where a user pays a set price for each element within a certain content document. This is determined by the price set for the selected element's element type and a credential set combination. Therefore, as depicted in figure 7.5(b), for every element, a user with a specific credential set, will pay the set price associated with the content element where the element type of the content element and credential set of the user yields a match.



(a) CBiX$_0$: Document Level Pricing



(b) CBiX$_2$: Element Level Pricing



(c) CBiX$_3$: Inherited Element Level Pricing

Figure 7.5: The Evolution of Pricing in CBiX

CBiX$_3$ further enhances the pricing aspect by combining the element level pricing of CBiX$_2$ with inheritance from CBiX$_1$. As depicted in figure 7.5(c) the result is that the user can purchase a single content element for that content element's price or purchase the content element and all its children content elements, paying a discounted price. For example, in figure 7.5(c), a user that wishes to purchase the top content element will pay the price of R8 and receive that content element and all its child elements. The user could instead wish to purchase a lower level element within the content document and would pay the specified price at the selected element for that element and all it child elements. The inherited price is the sum of the prices for the selected content element's child elements.

CBiX$_3$ retains all the pricing functions from CBiX$_2$, and redefines the GetPrice function to enable inherited element pricing.

The function GetPrice is defined by the following schema:

Definition 7.6.

```
GetPrice
  $amount! : \mathbb{Z}$
  $credentials? : \mathbb{P}\ CREDENTIAL$
  $selectedelement? : ELEMENT$
  ----------------------------------------
  $\exists\ assignedcred : \mathbb{P}\ credentials? \bullet$
  $(amount!, temp, GetType(selectedelement), assignedcred) \in Price$

  $if(amount! = NULL)$
      $E_1 = ElementHierarchy(selectedelement?)$
          $\forall\ e \in E_1$
              $GetPrice(amount', e, assignedcred)$
              $amount! = amount' + amount!$
```

The function GetPrice takes the user's credential set as well as the selected element as input. The price amount is retrieved from the price data set according to the selected element's element type and the user's credential set. The returned amount is either the price amount for the selected element only; or the amount is NULL. If the amount is NULL the ElementHierarchy relation defined in CBiX$_1$ is used to get the child elements of the selected element. For each of the child elements the GetPrice function is called recursively. Each child element's amount is then also retrieved. This operation continues until there are no more NULL amounts. If an element does not have a NULL amount, then this element has no child elements. The sum of all the child element's price amounts form the inherited price, amount!, for the selected element.

For example, assume a content element tree with a parent element, A. A has two child elements, B and C. Furthermore, C has a child element, D. The parent element, A, has a NULL value associated with it. Child element

B is valued at R5 and child element C has a NULL value. C's child element, D, is valued at R9. Figure 7.6 depicts the mentioned content element tree structure.

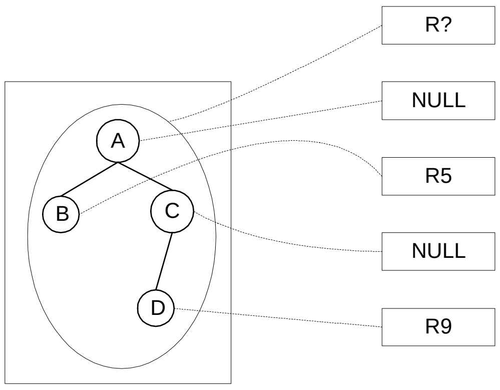


Figure 7.6: GetPrice function example: content element tree

Assume the user selects the parent element, A, of this construct, and that the user's credentials allow access to the selected element, A.

The GetPrice function begins by attempting to retrieve a price amount for the content element A. The retrieved amount is NULL. As A's amount is NULL the InheritedElement axiomatic description is used to retrieve the child elements of A.

The retrieved child elements of A are, B and C. For each of these elements the GetPrice function is recursively called, with that element being the selected element. Firstly, GetPrice is called with B as the selected element. The amount retrieved for element B is R5. As the value is not NULL, GetPrice does not drill deeper below element B. Next, GetPrice is then called with element C as the selected element. Element C's retrieved price amount is NULL. Therefore, the InheritedElement axiomatic description is once again used to return all the child elements of element C. This will return element D. GetPrice is then recursively called with element D as the selected element. A price amount of R9 is then retrieved. The total amount returned for element A and all its descendants is therefore R14.

7.4 Conclusion

This chapter defines CBiX$_1$ to CBiX$_3$ to extend the CBiX family of models.

This chapter firstly defined CBiX$_1$ which is based on CBiX$_0$ and is developed in parallel with CBiX$_2$. CBiX$_1$ builds on CBiX$_0$ by incorporating inheritance, which allows a child element to inherit the access permissions of its parent element within a content document.

Following the definition of CBiX$_1$, CBiX$_2$ is defined, which further enhances CBiX$_0$ by extending the pricing aspect of CBiX$_0$. CBiX$_2$ is defined to incorporate element pricing into the CBiX family of billing models. This is achieved by firstly redefining the AddPrice function to add a price to the price data set for a specified element type and credential set combination, and secondly by redefining the GetPrice function of CBiX$_0$ to return a price for a specific element instead of the whole content document. The Shopping-Basket function of the pricing aspect from CBiX$_0$ remains in place.

This chapter concludes the CBiX family of billing models by defining CBiX$_3$, the comprehensive model. CBiX$_3$ uses CBiX$_0$ as a basis for development, building the features of both CBiX$_1$ and CBiX$_2$ onto CBiX$_0$. The result is a comprehensive billing model that incorporates inheritance for controlling access to content elements within a content document, as well as inherited element level pricing. To achieve inherited level pricing, the element level pricing aspect developed in CBiX$_2$ is extended to take advantage of the principle of inheritance as defined by CBiX$_1$.

The following chapter describes the prototype that was developed to assist the author in understanding the technologies and principles of CBiX, and as a proof of concept for CBiX$_0$.

Part III

Epilogue

Chapter 8

Prototype Development

Chapter 6 introduced the first in the family of models, CBiX$_0$, the base model. All the other models in the CBiX family are built upon the functions and common components that are defined by CBiX$_0$, simply extending and modifying the defined set of functions and common components as needed.

To provide clarity with regards to the technologies involved in developing a content-based billing system, and as a proof of concept, the author has developed a prototype based on CBiX$_0$. This chapter begins by clarifying the motivation for developing a prototype and the expected insights. Following the motivation for the prototype any assumptions, choices and explicit omissions in the prototype are mentioned. The next section of the chapter describes the working of the prototype, focussing first on the administrative backend component of the prototype and then on the web-based user frontend component of the prototype. Following the description of the prototype any developmental issues that could arise when developing the prototype into a complete billing system are addressed.

8.1 Motivation for the Prototype

The main motivation for the development of the prototype was as a proof of concept to demonstrate that a billing system based on CBiX is viable. The prototype was not only used as a platform to test the key concepts of CBiX, but also to motivate and substantiate the choices made whilst developing CBiX by demonstrating that the model is functional.

The prototype was also developed to give the author further insights into the refinement of the CBiX family of models and give the author ideas as to how CBiX can be further improved and enhanced.

For the prototype the author has however made a number of assumptions and specific omissions that would have to be implemented in a full system.

8.2 Prototype Assumptions, Choices and Explicit Omissions

The assumptions, choices and explicit omissions that were made do not affect the viability of the prototype as they represent components that would add unnecessary complexity to a prototype.

8.2.1 Prototype Assumptions

It is assumed that there is a mechanism in place that receives the user's credentials on arrival at the web site. Furthermore it is assumed that these credentials are encapsulated in an XML file, or can easily be represented as an XML file, which is then used by the billing system.

It is also assumed that there is a payment mechanism in place that can securely handle the payment for the purchased content. As payment mechanisms and shopping baskets are readily available technologies the author has chosen not to develop and implement these mechanisms.

8.2.2 Developmental Choices

The prototype's web-based user frontend uses XSLT stylesheets to transform all the XML content into HTML that the user can view. This is achieved by applying the XSLT stylesheets to the appropriate XML content documents depending on the user's selections and the user's credentials. The benefit of using XSLT to transform the XML content documents is that it allows the user view to easily be customized for the individual user's preferences as well as for display on any media. When implementing a full system, depending on the developer's needs, XSLT might prove to be to slow and other means of displaying the XML content would have to be investigated.

Another developmental choice was to develop the administrative backend in Microsoft Visual Basic 6.0. Each developer can choose the programming language that best suits their needs and experience when developing the administrative backend.

8.2.3 Explicit Omissions

The prototype has explicitly omitted a mechnism for receiving user credentials. The author has simply represented the credential mechanism by presenting the credentials as an XML file for each user.

The prototype has also explicitly omitted the payment mechanism and the shopping basket in the web-based user frontend. The developer can simply choose the technology that best suits the billing system's needs and slot in the selected payment mechanism and shopping basket. The author has instead chosen to simulate the payment process and the shopping basket by using a message box to inform the user that one of the mechanisms has been entered into depending on the user's credentials. The user simply has to accept the message by clicking the "OK" button to accept that either the payment has taken place, or that content has been added to a shopping basket.

8.3 The Prototype

The prototype consists of two main components. The first component is the administrative backend. The second component is the web-based user frontend. The following two sections address each component in detail.

8.3.1 The Administrative Backend

The administrative backend is a stand alone tool developed to perform all the administrative functions necessary for the correct operation of the billing system.

The startup screen is a large menu bar depicted in figure 8.1 on the next page. From this menu bar the administrator can navigate to any of the administrative functions he has to perform. Before the administrator can access certain functions he has to first have performed other functions. Any functions that can not be carried out at a certain point in time are disabled.

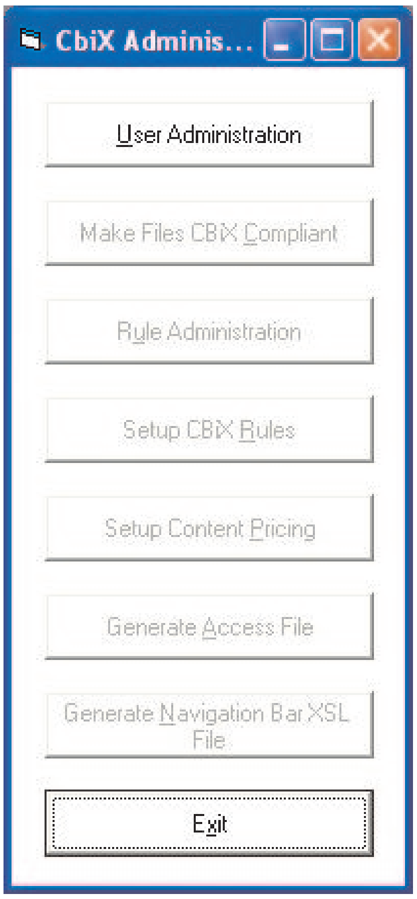


Figure 8.1: The Administrative Backend's Main Menu

This description of the administrative backend follows the steps needed to properly set up the billing system.

Step 1 Click “User Administration”. A window will open allowing the administrator to maintain the list of usertypes that are recognized by the billing system. This list of usertypes is used when setting up the pricing data and determines the price a user will pay for content depending on that user’s usertype as identified by the user’s credentials.

Step 2 Click “Make Files CBiX Compliant”. A window will open, allowing the administrator to select the directory containing the XML content documents that have been provided by the content provider. The administrator then has to select the document class XML schema that validates the XML content documents in the selected directory. Once the “Make Files CBiX Compliant” button has been clicked the XML content documents are validated against their validating document class schema. The XML content documents and their validating document class XML schema are processed and modified to make them compliant with CBiX. The CBiX compliant XML content documents and document class XML schema are then put in a new directory. The XSLT stylesheets are then generated. The public XSLT stylesheet is used to display the public content and a link to the private content with the corresponding price of the private content. The private XSLT stylesheet is used to display the private content once payment has taken place. For each document class schema there is a public/private XSLT stylesheet pair used for the display of that type of content.

Step 3 Click “Rule Administration”. A window will open allowing the administrator to maintain the list of CBiX rules. The rule list consists of two columns, with the first column containing the name of the rule and the second column containing the condition of the rule. This is where the administrator enters the condition against which the user’s credentials are matched. For example, an age rule can be set up and the administrator can specify that the user’s age credential has to equal or exceed the age specified for a specific content element. Rule administration has to be performed before the rules can be set up for the content.

Step 4 Click “Setup CBiX Rules”. A window opens that allows the administrator to set up the rules used by the billing system to determine whether a user can access content based on the CBiX rule conditions versus the user’s credentials. The rules are obtained from the database of rules and displayed as columns in the window along with all the XML content documents. This allows the administrator to specify the access restriction values for each rule for each XML content document. For example, if one of the rules is that the user’s age should equal or exceed the specified age restriction value, then the administrator inserts the age requirement for a specific content document under the age column in the appropriate row.

Step 5 Click “Setup Content Pricing”. A window opens allowing the administrator to select the type of content. This is obtained from a database of the content types as obtained from the document class schemas. The administrator then selects the usertype which is also obtained from a database of usertypes that is set up during user administration. The price for the selected content and usertype combination is entered by the administrator. Once the “Accept Price” button is clicked the price is stored in a database. This is similar in function to the AddPrice function defined in definition 6.12 on page 90. In the AddPrice function an amount is added to the price data set as defined in the Price Schema, definition 6.11 on page 89, for a specific content type (DOCUMENTCLASS) and CREDENTIAL set (usertype). When the “Generate Price File” button is clicked the pricing XML file is generated. The content is priced according to the price file. For a specific content type, a specific user (determined by the user’s credentials) will receive the specified price. If a price exists in the database for an existing content type and usertype combination this price is overwritten with the new price that is entered. A new price file must then be generated for the changed prices to take effect.

Step 6 Click “Generate Access File”. The access file that is used by the web-based user frontend is then generated. The access file is used to track the physical locations of the XML content documents and their public/private XSLT stylesheet pairs. The access file also contains the restrictions of the CBiX rules for the XML content documents. The XML content documents and document class schemas need to be made CBiX compliant and the CBiX rules have to be set up before the access file can be generated.

Step 7 Click "Generate Navigation Bar XSL File". You need to have set up the CBiX rules before you can do this. The navigation bar XSLT stylesheet is used by the web-based user frontend to generate the navigation bar. The XSLT stylesheet checks the access file against the user's credentials for each of the CBiX rules to determine which content the user is allowed access to, and consequently displayed in the navigation bar.

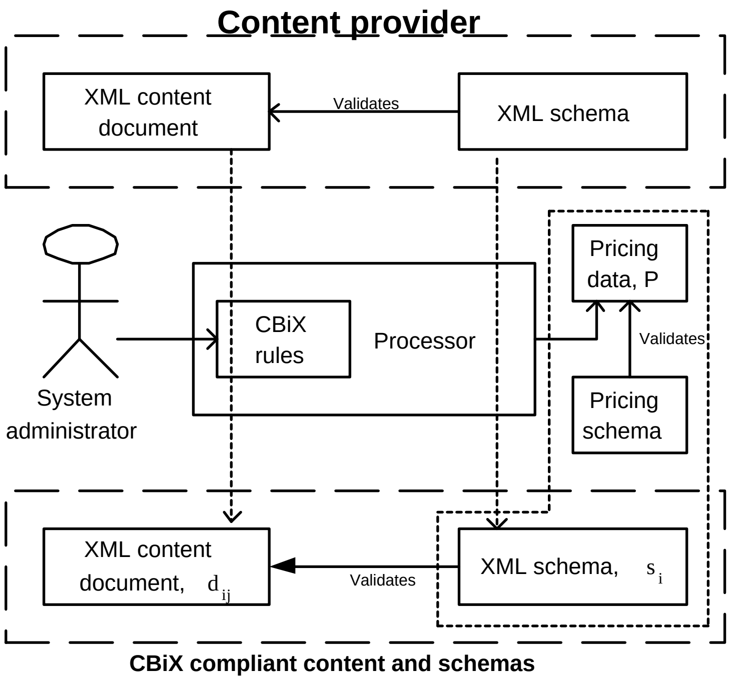


Figure 8.2: The Administrative Backend (repeated from figure 5.7)

Figure 8.2 depicts the conceptual administrative backend of the architecture to support CBiX as described in chapter 5. The administrative backend of the architecture maps onto the administrative backend of the prototye. Just as with the architecture depicted by figure 8.2, the prototype has rules and pricing that have to be set up. The prototype also sets up users and generates an access file and the navigation bar XSL stylesheet, which is not depicted on figure 8.2. Finally, as depicted on figure 8.2, the prototype also receives XML content from a content provider, validates the XML content according to the associated XML schemas and makes the XML content CBiX

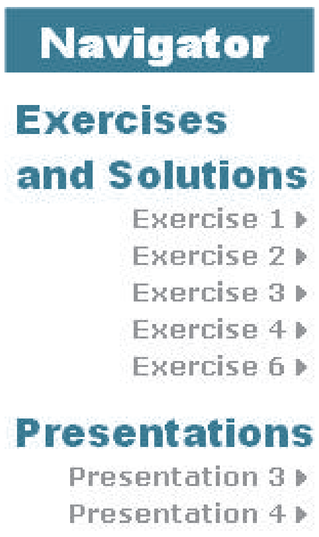


Figure 8.3: The Navigation Bar

compliant.

Once the administrative backend has prepared the content for use, the web-based user frontend is used to present the content to the users.

8.3.2 The Web-based User Frontend

The web-based user frontend is used to deliver the content to the user. The web-based user frontend can be divided into two major components. The first of these components is the navigation bar.

The Navigation Bar

The navigation bar is generated using the XSLT stylesheet that is generated in the administrative backend, described in step 7. The XSLT stylesheet used to generate the navigation bar evaluates each XML content document's access restrictions, which are set up in the administrative backend by setting up CBiX rules, against the user's credentials. Only content that the user is allowed to access is displayed in the navigation bar.

Figure 8.3 shows a sample navigation bar. This navigation bar represents two content types, exercises and solutions and presentations. Each content type forms a major heading. Under each major heading are the links to the content documents that the logged in user's credentials allow that user to

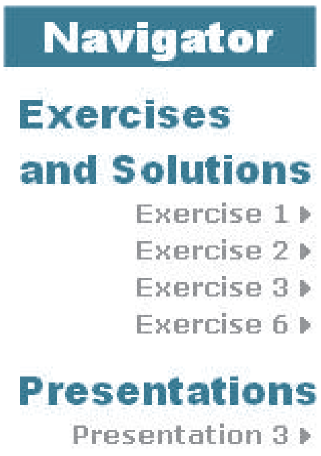


Figure 8.4: The Alternative Navigation Bar

access. Figure 8.4 beneath that shows another user's navigation bar with the links to Exercise 4 and Presentation 4 omitted due to the user's credentials, compared with the navigation bar represented in figure 8.3.

Figure 8.5 represents the conceptual navigation bar developed in the user frontend in chapter 5. Just as in the prototype, figure 8.5 depicts how the user's credentials determine which document class schemas and their corresponding XML content documents form the navigation bar.

Once the user clicks on a link to an XML content document the second component of the web-based user frontend, the Content Display Area (CDA) is used to display the selected content.

The Content Display Area

Once the user selects an XML content document for display, the CDA is updated. The CDA will initially display any free content, designated as public, in the selected XML content document, a link to the purchasable, designated as private content, and the price for the purchasable content as depicted in figure 8.6 on page 118. The price that is displayed is retrieved from the price data set that is defined in definition 6.11 on page 89, by the GetPrice function which is defined in definition 6.13 on page 91. The GetPrice function retrieves the price for a specific content type (as determined by the DOCUMENTCLASS) and user credential set. This display is achieved by applying the appropriate public XSLT stylesheet to the selected XML content docu-

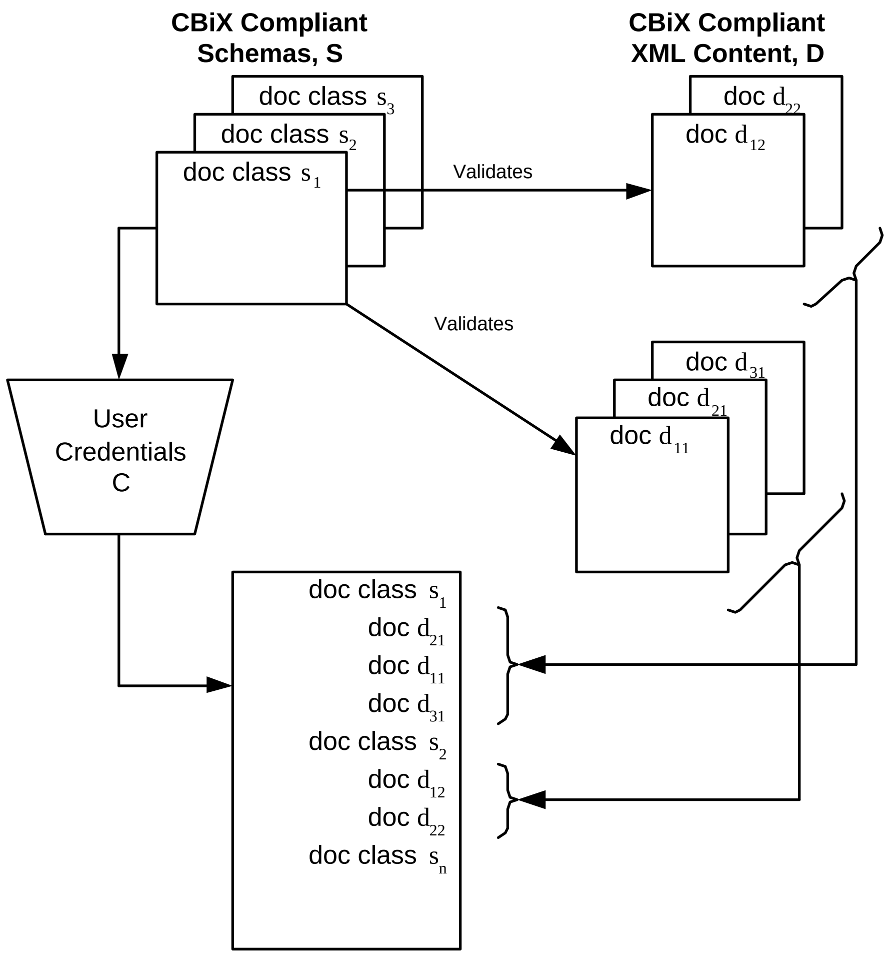


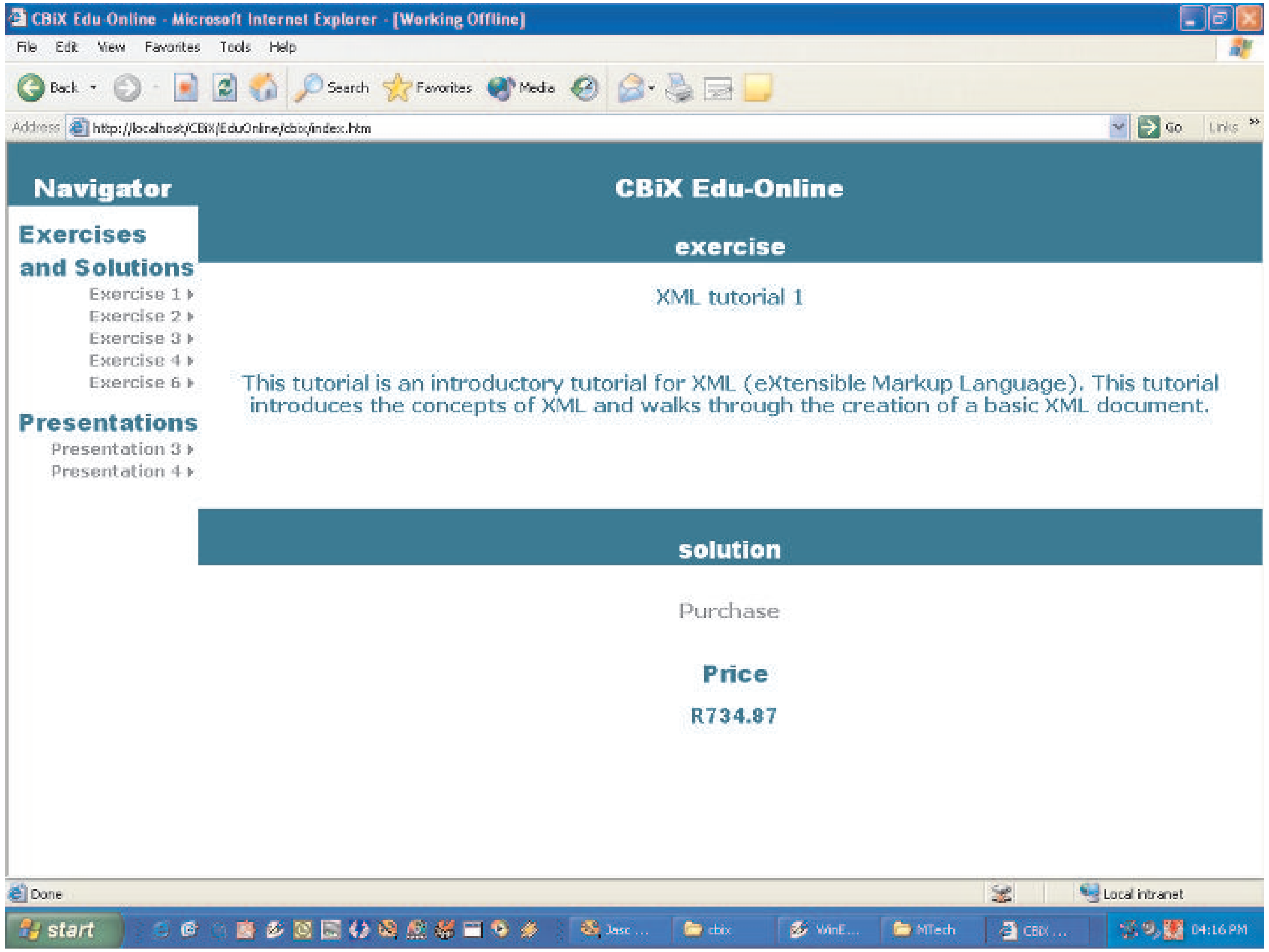
Figure 8.5: The User's Navigation Bar (repeated from figure 5.8)

ment. The association between the selected XML content document and an appropriate XSLT stylesheet is made in the access file that was generated in the administrative backend in step 3. The public XSLT stylesheet for the selected content document was generated for that content document's document class schema when the files where made CBiX compliant in step 3 of the administrative backend.

Once the user follows the link to purchase the private content, the user's credit rating credential is checked. If the user has a positive credit rating, the user is allowed to add the selected content to that user's shopping basket, as depicted in figure 8.7 on page 119. The user can then pay for the shopping basket at the end of the session. However, if the user has a negative credit rating, that user's shopping basket will be limited to a size allowing only one content element. The user will then have to enter into a payment process and pay for the selected content as depicted in figure 8.8 on page 120. The working of the shopping basket is defined in the axiomatic description, definition D.2 on page 148. This definition limits the size of the shopping basket according to the status of the user's credit rating credential. In the prototype the shopping basket is not implemented, but instead just simulated. The payment process also does not take place, but is instead simulated. As depicted in figure 8.7 and figure 8.8 the user simply clicks the OK button in a message box to accept adding content to the shopping basket, or to pay for the content.

Once the content has either been added to the shopping basket or paid for, the CDA is updated. The updated CDA is achieved by applying the private stylesheet that is associated with the XML content document in the access file that is generated in the administrative backend in step 3. This XSLT stylesheet causes the CDA to display both the public content and the purchased content from the XML content document as depicted in figure 8.9 on page 121.

Figure 8.10 on page 122 depicts the conceptual content display area and purchase process developed in the architecture in chapter 5 on page 69. Just as in the prototype the user's credentials along with the selected XML document and pricing data are parsed by an XSLT stylesheet to generate the content display area. After the payment process takes place, as determined by the user's credentials, the content display area is updated to display the


CBiX Edu-Online - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/CBiX/EduOnline/cbix/index.htm
Go Links
Done
Local intranet
start
Jasc ... cbix WinE... MTech CBiX ...
04:16 PM

Navigator

Exercises and Solutions

Exercise 1

Exercise 2

Exercise 3

Exercise 4

Exercise 6

Presentations

Presentation 3

Presentation 4

CBiX Edu-Online

exercise

XML tutorial 1

This tutorial is an introductory tutorial for XML (eXtensible Markup Language). This tutorial introduces the concepts of XML and walks through the creation of a basic XML document.

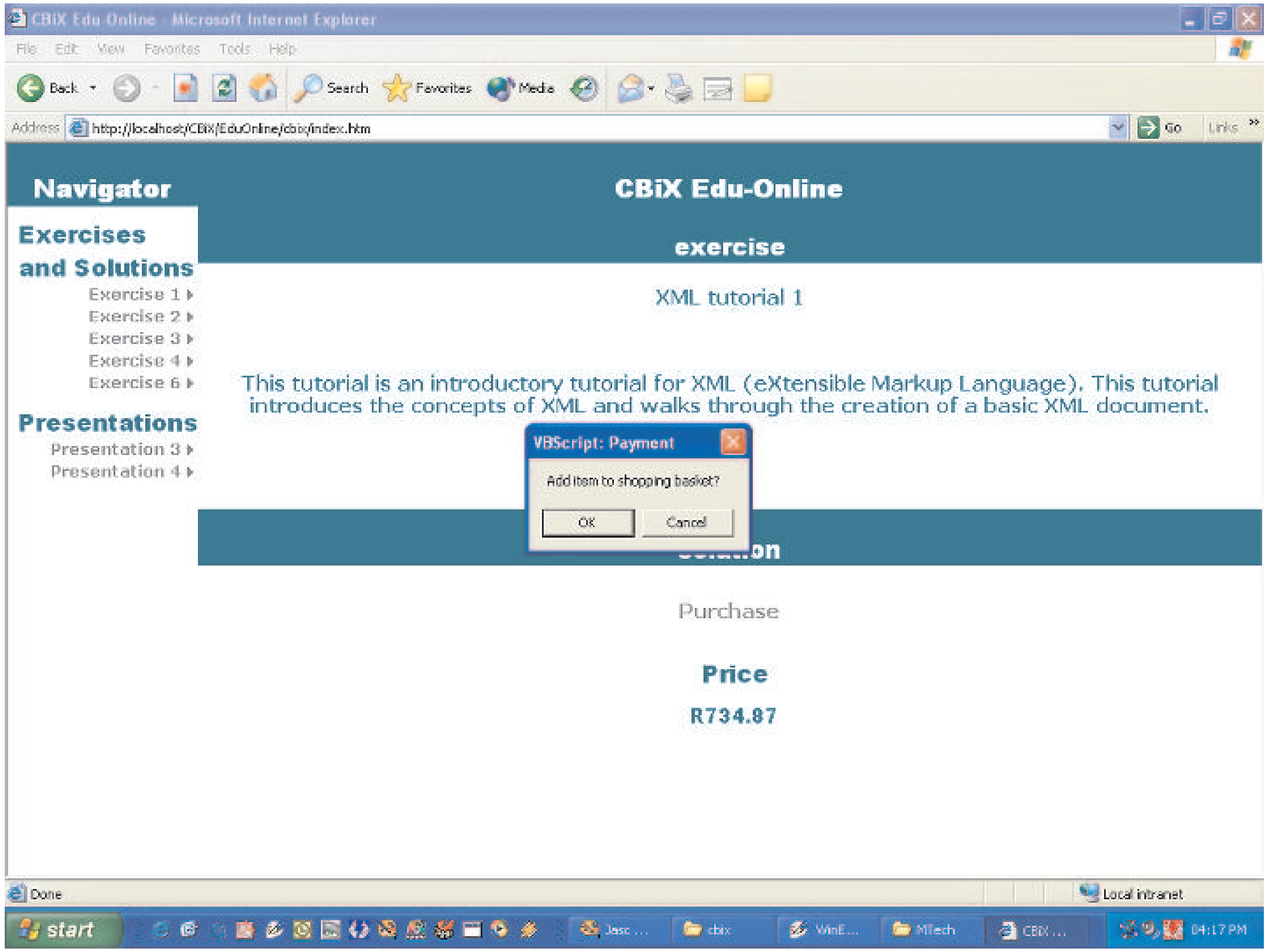
solution

Purchase

Price

R734.87

Figure 8.6: The Public View



CBiX Edu-Online

exercise

XML tutorial 1

This tutorial is an introductory tutorial for XML (eXtensible Markup Language). This tutorial introduces the concepts of XML and walks through the creation of a basic XML document.

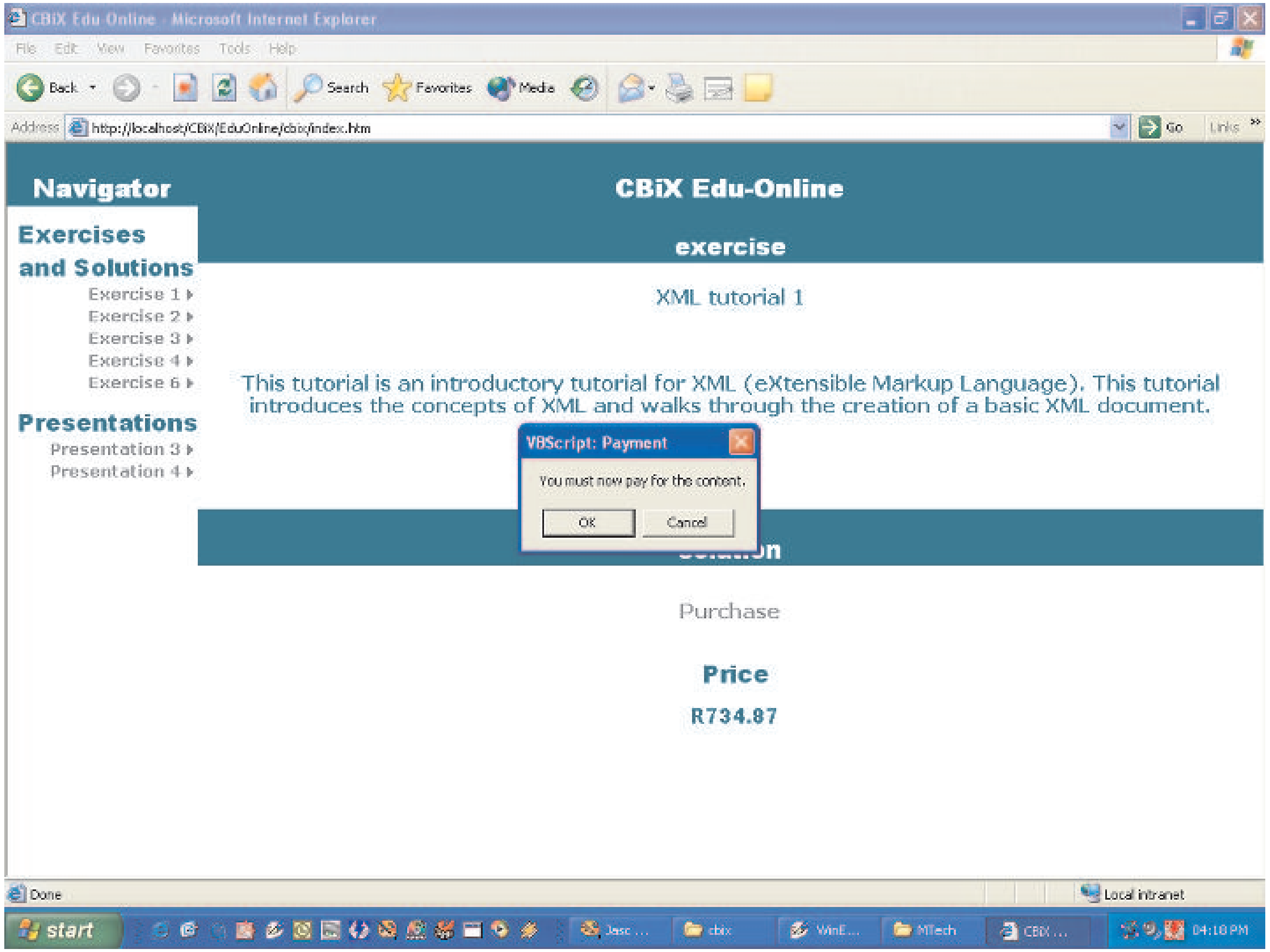
Add item to shopping basket?

Purchase

Price

R734.87

Figure 8.7: The Positive Credit Rating Procedure


CBiX Edu Online - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/CBiX/EduOnline/cbix/index.htm
Go Links

Navigator

Exercises and Solutions

Exercise 1
Exercise 2
Exercise 3
Exercise 4
Exercise 6

Presentations

Presentation 3
Presentation 4

CBiX Edu-Online

exercise

XML tutorial 1

This tutorial is an introductory tutorial for XML (eXtensible Markup Language). This tutorial introduces the concepts of XML and walks through the creation of a basic XML document.

VBScript: Payment

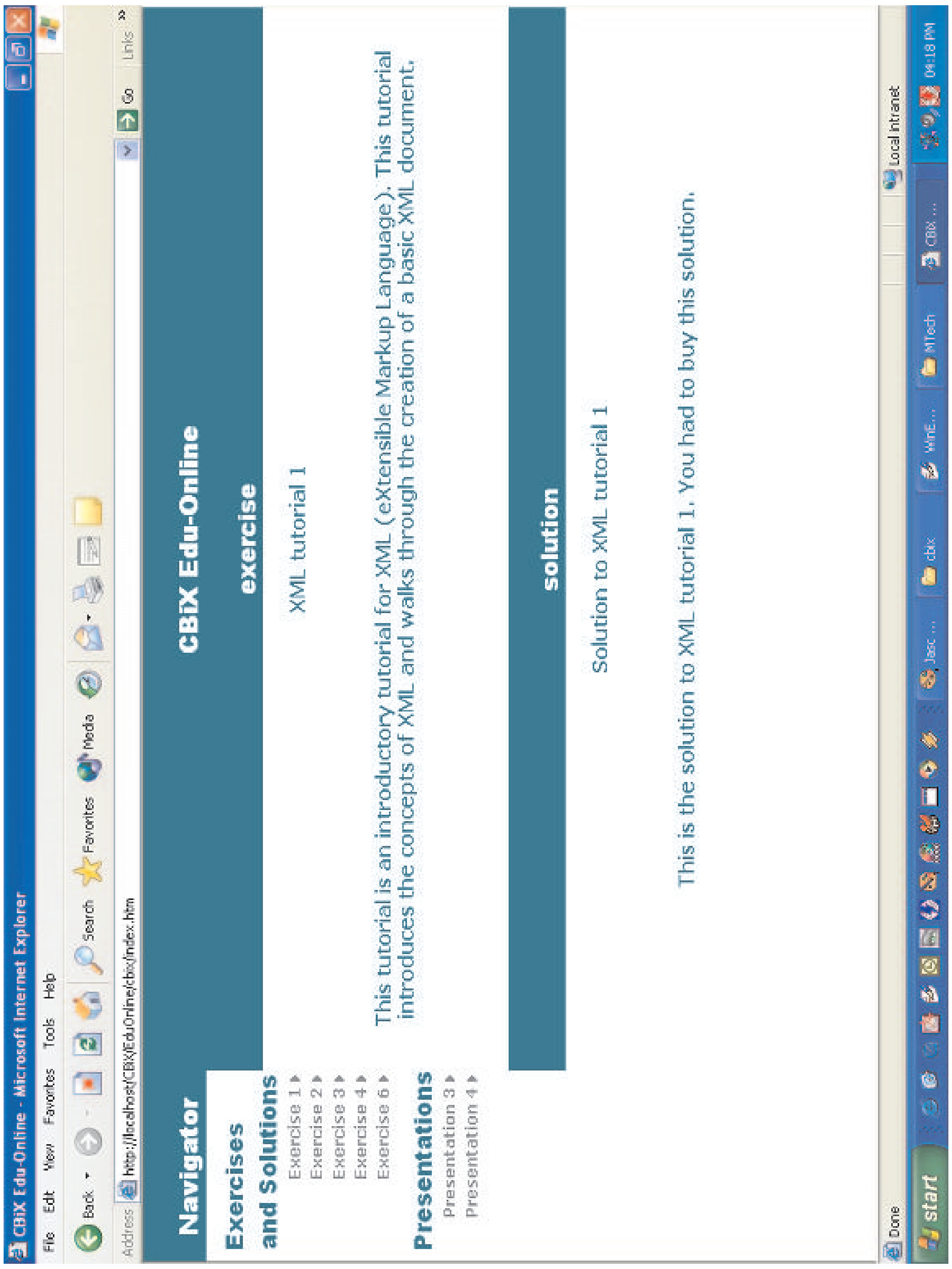
You must now pay for the content.

Purchase

Price

R734.87

Figure 8.8: The Negative Credit Rating Procedure


CBIX Edu-Online - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/CBIX/EduOnline/cbix/index.htm
Go Links
Done
Local intranet
start
04:18 PM

Navigator

Exercises and Solutions

Exercise 1
Exercise 2
Exercise 3
Exercise 4
Exercise 6

Presentations

Presentation 3
Presentation 4

CBIX Edu-Online

exercise

XML tutorial 1

This tutorial is an introductory tutorial for XML (eXtensible Markup Language). This tutorial introduces the concepts of XML and walks through the creation of a basic XML document.

solution

Solution to XML tutorial 1

This is the solution to XML tutorial 1. You had to buy this solution.

Figure 8.9: The View Once the Content has been Purchased

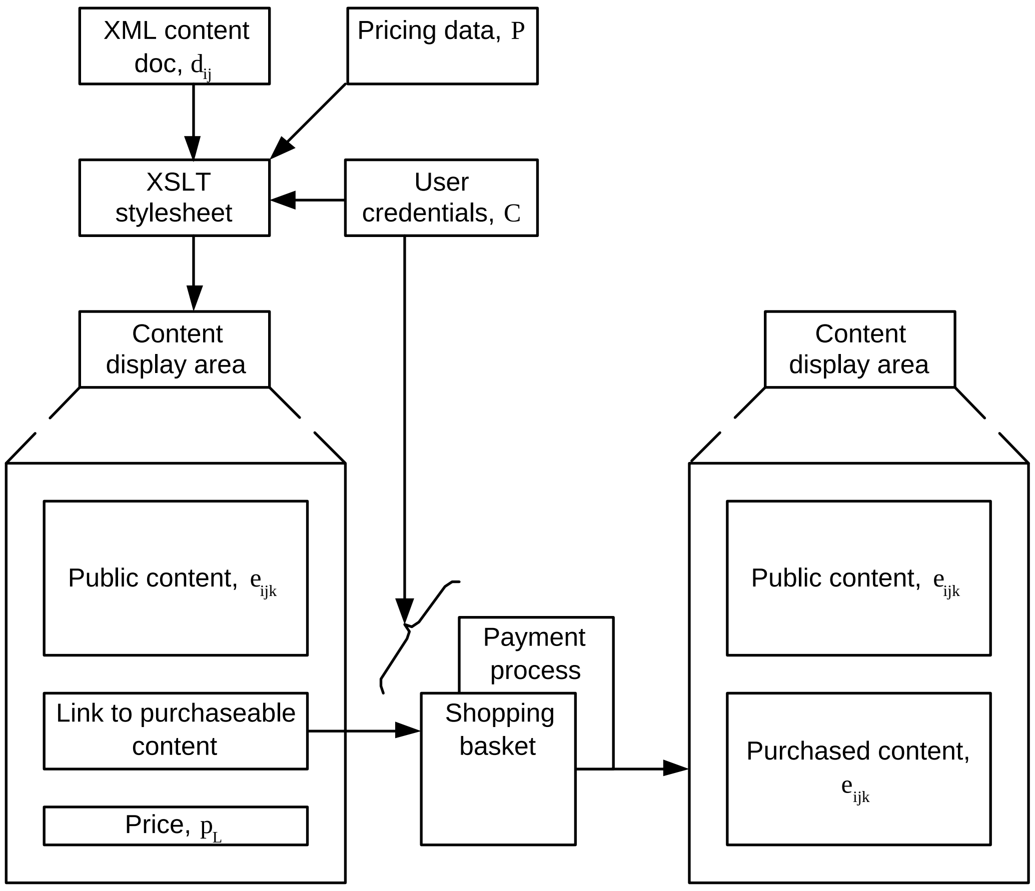

XML content doc, d_{ij}
Pricing data, P
XSLT stylesheet
User credentials, C
Content display area
Public content, e_{ijk}
Link to purchaseable content
Price, p_L
Payment process
Shopping basket
Content display area
Public content, e_{ijk}
Purchased content, e_{ijk}

Figure 8.10: The User's Content Display Area (repeated from figure 5.9)

purchased content.

The prototype, consisting of the administrative frontend and the web-based user frontend, forms a proof of concept for CBiX$_0$.

8.4 Developmental Issues

There are however a few developmental issues that would need to be addressed to evolve the prototype into a complete billing system.

The first issue that is not addressed in the prototype is the technical implementation of user credentials. The question is how the users will present their credentials to the billing system. The choice of technology to implement the credential subsystem will be up to the developer of the billing system who will need to investigate the available technologies and decide which technology to implement. In chapter 4 the author has identified digital certificates and public/private key technology as possible technologies that can be used to deliver the user's credentials to the billing system, providing the system with credentials that can be trusted, as well as a mechanism for the billing system to securely deliver the purchased content to the user by encrypting the content with the user's public key.

The second issue is the payment subsystem. The author has not specified or developed the exact payment mechanism that should be used. There are a wide variety of payment options available and the developer of a full system can choose whichever payment option suits the business best. This could include a wide range of options such as electronic money, and credit or debit cards.

The third issue is that the shopping basket component of the purchasing process is not specified or developed in the prototype. The developer will be able to choose from a wide range of shopping baskets available for purchase, and is free to choose any shopping basket as long as it can be integrated into the billing system.

The fourth issue is that no security has been implemented in the administrative backend. For a live system this issue would have to be addressed as only authorized content administrators should have access to the administrative backend.

The fifth issue not addressed is element and inherited element level pricing. Element and inherited element level pricing is not implemented in this prototype as the prototype is based on CBiX$_0$. In a full system, depending on the requirements of the billing system that has been developed, the developer would probably need to incorporate dynamic pricing from CBiX$_2$. The most likely model that developer's will use is CBiX$_3$, which is the comprehensive model, incorporating all the features from both CBiX$_1$ and CBiX$_2$ as well as inherited element level pricing.

The final issue that is not addressed in the prototype is how the navigation bar will be implemented in different environments. The navigation bar as developed for the prototype is optimized for a large screen display such as a PC monitor. If the content is being delivered to a device such as a mobile phone or a PDA, the navigation bars structuring would have to change and be optimized for the target device. The principle of the navigation bar would however remain the same.

8.5 Conclusion

The prototype demonstrates that a billing system based on CBiX is viable and that the concepts of CBiX can be implemented in a billing system.

The prototype has made certain assumptions and choices, and explicitly omitted certain components such as the delivery of credentials to the billing system and the payment subsystem. These assumptions, choices and explicit omissions where made to avoid unnecessary complexity in the prototype, and to allow the developer of a complete billing system to choose the technologies to perform these functions that would best suit the needs of the developer.

The prototype consists of two components. The first component is the administrative backend. This tool allows the content administrator to prepare the received content and make this content CBiX compliant and ready to be used by the billing system. The administrative backend tool also generates the access file containing the CBiX rule restrictions, content document and public/private XSLT stylesheet locations. Furthermore, the administrative backend tool enables rules to be specified for the content. Finally, the administrative backend tool generates the pricing data for the content and the navigation bar XSLT stylesheet. The second component is the web-based user frontend. This component allows the user to navigate to the content that he/she wishes to purchase using the navigation bar. The user will only be able to navigate to content that the user's credentials allow that user to access. Once the user has selected the content, any public content is displayed in the CDA along with a link to the content that has to be purchased, as well as its price. If the user decides to purchase the content by clicking the link, a payment process is entered, with the user either paying for the content because of a negative credit rating according to the user's credentials, or adding the content to a shopping basket because of a positive credit rating according to the user's credentials. Once payment has taken place, or the content added to the shopping basket the CDA is updated to display the public content as well as the purchased content.

This chapter has also mentioned a number of issues that need to be addressed by a developer wishing to develop a complete billing system based on CBiX.

This dissertation will conclude in the next chapter by summarizing the contributions of the research as seen by the author.

Chapter 9

Conclusion

This dissertation was based on the premise that the global economy has made a transition from one with an industrial focus, to one based on knowledge and information. A key enabler of the new global economy is the Internet. The Internet provides the means for entirely new revenue streams such as those brought about by e-commerce, e-business, and lately, e-enterprise.

Further investigation into the Internet and new potential sources of revenue generation via the Internet brought about the realization that content is one potential source of revenue generation.

This prompted further investigation into content as a source of revenue. It was discovered that billing content properly is the key to effectively leveraging content as a revenue source.

Investigation into billing followed. From this investigation further conclusions could be made. Firstly, it was realized that billing by its very nature requires security, and therefore has a possible relationship with access control. Secondly, it was realized that content-based billing would require a flexible and extensible environment. Thirdly, it was realized that XML as an Internet standard provides a flexible and extensible environment for information delivery, and has a great potential impact on the next generation web. Finally, it was realized that although there are a number of content-based billing products commercially available, these products are, as far as could be determined, proprietary. These realizations contributed to providing the motivation for undertaking the research.

The primary problem identified was that there was no generic content-based billing model for XML environments that could provide a "template" for developers wishing to develop a content-based billing system.

The objective of the research was therefore to define a model for content-based billing in XML environments. Furthermore, the model had to form a basis to develop a flexible and secure content-based billing system within any environment based on the XML standard.

As part of the process of developing the model a number of sub-objectives had to be addressed with regards to billing, access control and XML.

Firstly, the need for and elements of content-based billing had to be established. Secondly, XML had to be investigated, and a method to expose content in XML environments had to be highlighted. Thirdly, the role of access control needed to be clarified and the relationship between access control and billing had to be established. Finally, effective access control for XML environments needed to be explored.

Now consider how the research met the stated objectives.

9.1 Revisiting the Objectives

The CBiX model was developed to address the main objective of this research. To support the definition of CBiX the research had to address a number of sub-objectives.

9.1.1 Content-based Billing: Needs and Elements

Addressing this sub-objective entailed firstly establishing the importance of the Internet in the new global economy, followed by the evolution of commerce on the Internet. It was found that content is a potential source of revenue that is garnering increased interest. This resulted in further investigation into the various web business models used on the Internet, and the applicability of these models to CBiX. Future work needs to investigate the impact of varied business models on the requirement of a billing system, including how a billing system should cater for a company that changes/evolves web business models over time. The investigation of business models in chapter 2 highlighted the importance of revenue justifications for the selection of a web business model. Hand in hand with revenue justification comes pricing. Pricing is intrinsic to a billing system, which has a primary function of enabling the sale of content. However, the Internet allows for great scope with regards to pricing strategies due to its flexible, real-time nature. An investigation into the applicability of various pricing strategies to CBiX was undertaken as CBiX is a billing model and therefore has to perform some type of pricing. Even though a number of pricing strategies were investigated, CBiX simply implements pricing for content that is document or element based. In the CBiX model the pricing strategy is evolved from document-level pricing to element-level pricing to inherited element-level pricing. Further investigation into pricing strategies and how true dynamic pricing can be incorporated into the billing model needs to be undertaken. Another need for an Internet-based billing system is interoperability and the ability to exchange information easily. This leads to the investigation of XML as an environment for CBiX.

9.1.2 XML: The Common Environment

XML is readily acknowledged as the platform for the next generation Internet. Furthermore, XML has a large family of standards based upon it. Investigation into XML and the XML family of standards in chapter 3 paints a picture of a powerful common environment for information interchange. XML by its very nature is highly flexible and has an inherent ability to facilitate information interchange, and provide mechanisms for access control. The investigation into XML also provided an insight into XML schemas that can be used to ensure that the provided content is valid. XSLT, another XML based standard, together with XPath, provides a powerful tool for parsing and displaying content in a myriad of formats. This feature is particularly useful to enable delivery of content to a variety of devices. This research does not address the delivery of various forms of content such as streaming content. Instead focus is given to document based content. Further research also needs to be undertaken into displaying and formatting content for a variety of devices, and the usability issues involved.

The financial nature of a billing system requires that access control be investigated.

9.1.3 Access Control

Access control is critical to a billing system due to its financial nature and therefore inherent security needs. It is also critical to ensure that content is only accessed by authorized users, and that those users then pay for the purchased content.

The first step was to investigate and understand the access control decision. This led to the identification of four administrative access control paradigms, namely; MAC, DAC, RBAC and credential-based access control.

Understanding of the requirements of each administrative access control paradigm as well as the three identified requirements for a collaborative access control model prompted an investigation into the needs of a billing system with regards to access control as well as looking at the synergy between access control and billing.

The synergy between access control and billing was therefore explored in chapter 4. It was shown that except for an essential difference in terms of the financial aspect of billing, access control and billing share common ground. It was also found that billing requires an access control paradigm that provides access control based on user properties, and not necessarily their role in a hierarchical organizational structure. Therefore credential-based access control presents a better possible solution for a billing system than the other administrative paradigms. Although a user's credentials could possibly be related to the user's role, it is not necessarily the case.

The selection of the credential-based access control paradigm for CBiX led to an investigation into the mechanisms of implementing credential-based access control. Furthermore, possible technologies to secure a credential-based access control system had to be investigated.

A number of possible technologies were identified such as XML signature and XACL. Further research needs to be undertaken into how credential-based access control can be practically applied in a billing system. Further research is also required to determine which technology combination is optimal to properly secure a billing system using credential-based access control.

Addressing these sub-objectives presented a platform of research upon which to develop the CBiX billing model.

9.1.4 CBiX

The CBiX billing model incorporates three aspects identified in chapter 5; access control, pricing and XML. Credential-based access control forms the access control component of CBiX, with XML document tree pruning being used to determine the user's permitted view. A developmental issue not specifically dealt with in CBiX is how the user presents his credentials to the billing system. Various technologies such as digital certificates and XML signature are however explored.

The pricing aspect of CBiX consists of an evolutionary approach to pricing. The initial strategy followed is document-level pricing whereby a user pays a set price for a whole document. This evolves into element-level pricing whereby the user pays a set price for each element within a document. The final step of the evolution is inherited element-level pricing, whereby the user pays a price for the selected element and all its child elements. An omission, not dealt with in CBiX , is the actual financial transaction. How the financial transaction occurs, via credit card or electronic money, and which payment protocols are to be followed are not addressed by CBiX.

The XML aspect of CBiX is that the model uses XML as the common environment for information interchange and content delivery.

The synergy between access control and billing led to the adoption of a development strategy for the CBiX model similar to that used to develop the RBAC96 (Sandhu et al., 1996) family of models. Therefore CBiX was developed in chapters 6 and 7 as a family of billing models; with a base model, and each of the following models building on the base model. The CBiX family of models is formally defined using the Z notation as explained in appendix D. The author has not defined all the common functions a billing system would need such as Add, Edit and Delete functions. These functions are explicitly undefined as this is standard functionality that a developer can easily develop without the need for formal specifications. Furthermore, these functions do not contribute to the essence of the model.

In addition to the family of models a prototype was developed. The prototype helped clarify the technologies involved with the development of a billing system based on CBiX , and it provided a proof of concept for the base model, CBiX_0.

9.2 Future Work

During the research a number of issues arose that are not addressed.

Firstly, further investigation needs to be undertaken into the various web business models on the Internet, and how the choice of a business model can impact revenues. It is a critical component for the successful undertaking of a business initiative to select the correct business model to maximize profitability and sustainability. Research needs to be undertaken into how a billing system can cater for an evolving business model, so that a business has the ability to change or evolve their business model over time as needed.

Secondly, in depth research needs to be undertaken into the various pricing strategies available on the Internet. The impact of pricing strategy selection combined with various web business models needs to be clearly understood. Further research needs to be undertaken to address the limited pricing strategies made available by the CBiX model. A clear understanding is needed to determine how a billing system can efficiently support multiple pricing strategies that can evolve over time.

Thirdly, further investigation into secure content delivery for XML environments also has to be undertaken. This research did not make any in-depth investigation into various forms of content delivery, such as streaming content, with the model focussing on the delivery of document based content. Hand in hand with an investigation into content delivery of various types of content there has to be clarity on how to effectively price the content.

Finally, from the development of the prototype, a further area of research can be identified. In the prototype a navigation bar is developed. However, this navigation bar is optimized for a large screen display such as a PC monitor. Research needs to be undertaken to determine how content can be navigated on a variety of devices such as mobile phones or personal digital assistants. Included in such research is the requirement for extensive usability studies into human computer interaction for a variety of devices. The ability to securely deliver content to a variety of devices will then also have to be investigated, taking into account factors a such as device size and connection speed.

9.3 Final Word

In conclusion, this dissertation has developed a family of billing models for XML environments, promoting a platform for interoperability and information exchange. CBiX provides developers with a flexible content-based billing model that is secure and operates in an XML environment. There is however a possibility for further research that was out of the scope of this dissertation as discussed under future work.

Nevertheless, the author believes that the study contributes to the understanding of content-based billing in XML environments. The author also wants to express the hope that this study will fuel further interest in this domain of discourse.

Part IV

Appendices

Appendix A

Well-formed XML

- Markup constructs such as parameter entity references are only allowed in specific places.
- No parameter entity used in the document may consist of only part of a markup declaration.
- No attribute may appear more than once on the same start tag.
- String attribute values cannot contain references to external entities.
- Non-empty tags must be properly nested.
- Parameter entities must be declared before they are used.
- All entities except the following: amp, lt, gt, apos, and quot must be declared.
- A binary entity cannot be referenced in the flow of content, it can only be used in an attribute declared as ENTITY or ENTITIES.
- Neither text nor parameter entities are allowed to be recursive, directly or indirectly.

Appendix B

XML Signature

This appendix, extracted from Simon et al. (2001), shows how to create an XML signature, followed by a short section on how to verify the XML signature. Please refer to the XML Signature specification (Eastlake et al., 2002) at the W3C for additional information.

B.1 How to Create an XML Signature

Here is a quick overview of how to create an XML signature in six easy steps.

The first step is to determine which resources are to be signed. This will take the form of identifying the resources through a Uniform Resource Identifier (URI).

For example, `http://www.abccompany.com/index.html` would reference an HTML page on the Web, `http://www.abccompany.com/logo.gif` would reference a GIF image on the Web, `http://www.abccompany.com/xml/po.xml` would reference an XML file on the Web, and `http://www.abccompany.com/xml/po.xml#sender1` would reference a specific element in an XML file on the Web.

The next step is to calculate the digest of each resource (the content that is to be signed). In XML signatures, each referenced resource is specified through a `<Reference>` element and its digest (calculated on the identified resource and not the `<Reference>` element itself) is placed in a `<DigestValue>` child element as shown in figure...

```
<Reference URI="http://www.abccompany.com/news/2000/03_27_00.htm">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
```

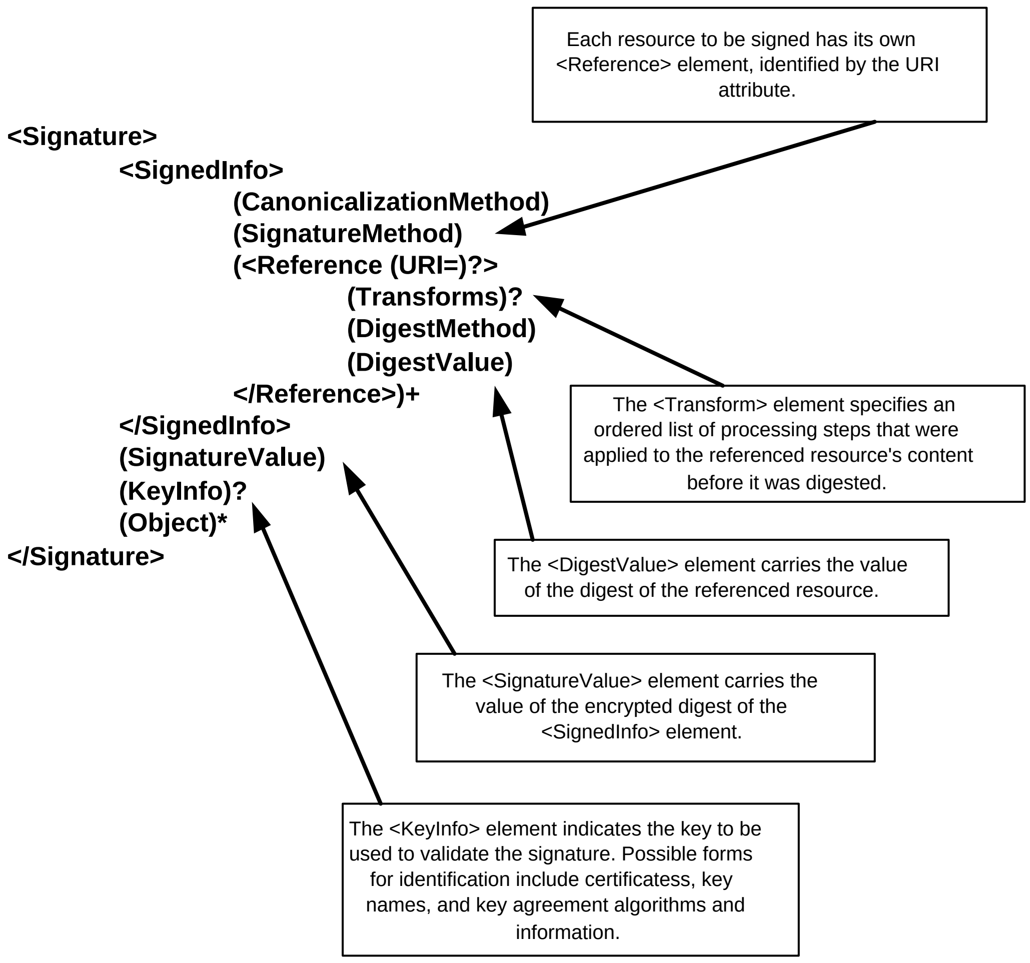

Each resource to be signed has its own <Reference> element, identified by the URI attribute.
<Signature>
<SignedInfo>
(CanonicalizationMethod)
(SignatureMethod)
(<Reference (URI=)?>
(Transforms)?
(DigestMethod)
(DigestValue)
</Reference>)+
</SignedInfo>
(SignatureValue)
(KeyInfo)?
(Object)*
</Signature>
The <Transform> element specifies an ordered list of processing steps that were applied to the referenced resource's content before it was digested.
The <DigestValue> element carries the value of the digest of the referenced resource.
The <SignatureValue> element carries the value of the encrypted digest of the <SignedInfo> element.
The <KeyInfo> element indicates the key to be used to validate the signature. Possible forms for identification include certificatess, key names, and key agreement algorithms and information.

Figure B.1: The Components of an XML Signature (Simon et al., 2001)

```
</Reference> <Reference
URI="http://www.w3.org/TR/2000/WD-xmldsig-core-20000228/signature-example.xml">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
</Reference>
```

The `<DigestMethod>` element identifies the algorithm used to calculate the digest.

The third step is to collect the `<Reference>` elements (with their associated digests) within a `<SignedInfo>` element like

```
<SignedInfo Id="foobar">
 <CanonicalizationMethod
  Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
 <Reference URI="http://www.abccompany.com/news/2000/03_27_00.htm">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
 </Reference>
 <Reference
 URI="http://www.w3.org/TR/2000/WD-xmldsig-core-20000228/signature-example.xml">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
 </Reference>
</SignedInfo>
```

The `<CanonicalizationMethod>` element indicates the algorithm that was used to canonize the `<SignedInfo>` element. Different data streams with the same XML information set may have different textual representations, e.g. differing as to whitespace. To help prevent inaccurate verification results, XML information sets must first be canonized before extracting their bit representation for signature processing. The `<SignatureMethod>` element identifies the algorithm used to produce the signature value.

Step four involves calculating the digest of the `<SignedInfo>` element, signing that digest and putting the signature value in a `<SignatureValue>` element.

```
<SignatureValue>MC0E LE=</SignatureValue>
```

In Step five you add key information. If keying information is to be included, place it in a `<KeyInfo>` element. Here the keying information contains the X.509 certificate for the sender, which would include the public key needed for signature verification.

```
<KeyInfo>
  <X509Data>
      <X509SubjectName>
          CN=Ed Simon,O=XMLSec Inc.,ST=OTTAWA,C=CA
      </X509SubjectName>
      <X509Certificate>MIID5jCCA0+gA...lVN</X509Certificate>
  </X509Data>
</KeyInfo>
```

Lastly, in step six, place the `<SignedInfo>`, `<SignatureValue>`, and `<KeyInfo>` elements into a `<Signature>` element. The `<Signature>` element comprises the XML signature.

```
<?xml version="1.0" encoding="UTF-8"?> <Signature
xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo Id="foobar">
  <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <SignatureMethod
    Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
  <Reference URI="http://www.abccompany.com/news/2000/03_27_00.htm">
     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
  </Reference>
  <Reference
  URI="http://www.w3.org/TR/2000/WD-xmldsig-core-20000228/signature-example.xml">
     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
  </Reference>
 </SignedInfo>
 <SignatureValue>MC0E~LE=</SignatureValue>
 <KeyInfo>
   <X509Data>
     <X509SubjectName>
        CN=Ed Simon,O=XMLSec Inc.,ST=OTTAWA,C=CA
     </X509SubjectName>
     <X509Certificate>
        MIID5jCCA0+gA...lVN
     </X509Certificate>
   </X509Data>
 </KeyInfo>
</Signature>
```

B.2 Verifying an XML Signature

To verify the signature of the `<SignedInfo>` element recalculate the digest of the `<SignedInfo>` element (using the digest algorithm specified in the `<SignatureMethod>` element) and use the public verification key to verify that the value of the `<SignatureValue>` element is correct for the digest of the `<SignedInfo>` element.

If this step passes, recalculate the digests of the references contained within the `<SignedInfo>` element and compare them to the digest values expressed in each `<Reference>` element's corresponding `<DigestValue>` element (Simon et al., 2001).

Appendix C

Academic Papers

The author has produced two academic papers based on the research to date. Both of the papers are available on the accompanying CD.

The first paper entitled, "Towards a Content-based Billing Model": The Synergy between Access Control and Billing, was presented at the Information Security South Africa (ISSA2002) conference in Johannesburg, June 2002. The paper was published on the accompanying conference proceedings CD.

The second paper entitled, "An Architecture for Selling XML Content", was presented at the Emerging Applications for Mobile and Wireless Access (MobEA) workshop, co-located with the WWW2003 conference, May 20-24, Budapest, Hungary. The paper was published in the proceedings of the MobEA workshop.

Appendix D

The Z Notation

According to Jacky (1994) Z is a set of conventions for presenting mathematical text, chosen to make it convenient to use simple mathematics to describe computing systems. Z is a model-based notation. It includes two notations. The first is the notation of ordinary discrete mathematics. The second notation is the Z paragraph which provides structures to the mathematical text.

This appendix gives a brief explanation of the Z notation that has been used to formally define the CBiX family of billing models. The appendix begins by defining the various Z paragraph constructs, utilizing examples from the development of the CBiX family of models. Followed the definition of the Z paragraph constructs is an explanation of the Z symbols used in the development of the CBiX family of models. These Z symbols are related to their equivalent set theory mathematical terms. Finally the appendix mentions expressions used in Z and clarifies the naming conventions used during the development of CBiX.

D.1 The Z Paragraph Constructs

In the dissertation three Z paragraph constructs are used. The first paragraph construct is the plain Z environment. This construct is used to declare a basic type.

An example from the development of CBiX is as follows:

Definition D.1.

$$[DOCUMENTCLASS]$$

In this example a basic type called a DOCUMENTCLASS has been declared. This basic type can now be instantiated and used in other Z constructs.

The second Z construct used is the axiomatic description. Axiomatic descriptions are used to describe functions in the development of the CBiX family of models.

An example of an axiomatic description from the development of CBiX is as follows:

Definition D.2.

$$\begin{array}{|l}
SB_i : SB \\
MAXVALUE : \mathbb{Z} \\
cred : CREDENTIAL ::= goodcreditrating \mid badcreditrating \\
\hline
if\ (cred = badcreditrating)\ then\ \#SB_i = 1\ else\ \#SB_i = MAXVALUE
\end{array}$$

The axiomatic description has two parts. The first part above the line contains the declarations. In the declaration section variables, constraints and instantiated basic types are found. The second part below the line is the predicate containing the function constraints. In this example the relevant credential is constrained to being either a good credit rating or a bad credit rating. Following the determination of the credit rating of the user, the size of the shopping basket is appropriately constrained.

The third environment used during the development of the CBiX family of billing models is the schema. Schemas are used to describe states, changes and operations within a system. A schema can be compared to a procedure in programming languages.

An example of a schema from the development of CBiX is as follows:

Definition D.3.

$$\begin{array}{|l}
\hline
\textit{GetPrice} \\
\textit{amount} : \mathbb{Z} \\
\textit{credentials?} : \mathbb{P}\ \textit{CREDENTIAL} \\
\textit{document?} : \textit{DOCUMENT} \\
\hline
\exists\ \textit{assignedcred} : \mathbb{P}\ \textit{credentials?} \bullet \\
\qquad (\textit{amount}, \textit{DocClass}(\textit{document?}), \textit{assignedcred}) \in \textit{Price} \\
\hline
\end{array}$$

A schema differs from the other constructs in that it has a name. This enables a schema to be called so that the schema's contents can be referred to. Just like an axiomatic description, a schema has a declaration part and a predicate part. The declaration part contains all the variable declarations. These variable declarations are local to the schema and consist of both input and output variables. The predicate part is called an invariant as it is always true. The predicate describes properties using the declared variables that are always true.

Within the Z paragraph constructs there a number of symbols used, some of which represent standard set theory mathematics.

D.2 Z Symbols

The following Z symbols are used within the Z paragraph constructs used to formally describe the CBiX family of billing models.

The Δ symbol is used in a schema to tell us that there is going to be a change in state caused by the schema.

For example, in the following schema the Δ symbol indicates a change with the PricingSchema.

Definition D.4.

$$\begin{array}{|l}
\textit{AddPrice} \\
\hline
\Delta \textit{PricingSchema} \\
\textit{docclass}? : \textit{DOCUMENTCLASS} \\
\textit{credential}? : \mathbb{P}\, \textit{CREDENTIAL} \\
\textit{amount}? : \mathbb{Z} \\
\hline
\textit{price}' = \textit{price} \oplus \{(\textit{amount}?, \textit{docclass}?, \textit{credential}?)\} \\
\hline
\end{array}$$

The $\mathbb{Z}$ symbol represents the maximal numeric type in Z as rational numbers, real numbers and complex numbers are not built in.

The $\mathbb{P}$ symbol means set of. In the following example $\textit{credentials}?\ \mathbb{P}\ \textit{CREDENTIAL}$ means that the variable credentials is a subset of the powerset of the basic type CREDENTIAL.

Definition D.5.

$$\begin{array}{|l}
\textit{GetPrice} \\
\hline
\textit{amount} : \mathbb{Z} \\
\textit{credentials}? : \mathbb{P}\, \textit{CREDENTIAL} \\
\textit{document}? : \textit{DOCUMENT} \\
\hline
\exists\, \textit{assignedcred} : \mathbb{P}\, \textit{credentials}? \bullet \\
\qquad (\textit{amount}, \textit{DocClass}(\textit{document}?), \textit{assignedcred}) \in \textit{Price} \\
\hline
\end{array}$$

The $\leftrightarrow$ symbol is used to show binary relations. In the following example a relationship is indicated between an ELEMENT and a set of CREDENTIAL. This relationship indicated that access to the element is determined by the relationship of the credential set associated with that element.

Definition D.6.

$$\begin{array}{|l}
\textit{Access} \\
\hline
\textit{access} : \textit{ELEMENT} \leftrightarrow \mathbb{P}\, \textit{CREDENTIAL} \\
\hline
\end{array}$$

Thick brackets, $(\!|\ ...\ |\!)$, are used to encapsulate the second argument in an expression. In the following example the secondary argument encapsulated in the brackets pertains to the assigned credential, ensuring that only elements are returned that have access granted according to the element's assigned credentials.

Definition D.7.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{GetElements}} \\
\hline
\mathit{elements}! : \mathbb{P}\,\mathit{ELEMENT} \\
\mathit{credentials}? : \mathbb{P}\,\mathit{CREDENTIAL} \\
\mathit{Access} \\
\hline
\exists\,\mathit{assignedcred} : \mathbb{P}\,\mathit{credentials}? \bullet \\
\qquad \mathit{elements}! = \mathit{access}\,(\!|\,\mathit{assignedcred}\,|\!) \\
\hline
\end{array}$$

The seq symbol represents a sequence of elements. In the following example this symbol is used to indicate that an XML content document consists of a number of content elements.

Definition D.8.

$$[\mathit{ELEMENT}]$$

$$\mathit{DOCUMENT} == \mathrm{seq}\,\mathit{ELEMENT}$$

The $\bullet$ symbol is a delimiter that separates a declaration and predicate within an expression as in the following example.

Definition D.9.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{GetPrice}} \\
\hline
\mathit{amount} : \mathbb{Z} \\
\mathit{credentials}? : \mathbb{P}\,\mathit{CREDENTIAL} \\
\mathit{element}? : \mathit{ELEMENT} \\
\hline
\exists\,\mathit{assignedcred} : \mathbb{P}\,\mathit{credentials}? \bullet \\
\qquad (\mathit{amount}, \mathit{GetType}(\mathit{element}?), \mathit{assignedcred}) \in \mathit{Price} \\
\hline
\end{array}$$

The exclamation symbol, !, used after a variable, means that the variable is an output variable, while a question mark symbol, ?, used after a variable means that the variable is an input variable. The following example has both input and output variables.

Definition D.10.

$$
\begin{array}{|l}
\hline
\textit{GetElements} \\
\textit{elements}! : \mathbb{P}\,\textit{ELEMENT} \\
\textit{credentials}? : \mathbb{P}\,\textit{CREDENTIAL} \\
\textit{Access} \\
\hline
\exists\, \textit{assignedcred} : \mathbb{P}\,\textit{credentials}? \bullet \\
\qquad \textit{elements}! = \textit{access} \,(\!|\, \textit{assignedcred} \,|\!)\, \\
\hline
\end{array}
$$

The $\oplus$ symbol is the overriding operator. In the following example the operator is used to override any existing access associations for an element and credential set.

Definition D.11.

$$
\begin{array}{|l}
\hline
\textit{AssignCredential} \\
\Delta \textit{Access} \\
\textit{element}? : \textit{ELEMENT} \\
\textit{credentials}? : \mathbb{P}\,\textit{CREDENTIAL} \\
\hline
\textit{access}' = \textit{access} \oplus \{(\textit{element}?, \textit{credentials}?)\} \\
\hline
\end{array}
$$

The symbols mentioned above are by no means all the symbols available for use in Z. In addition to using symbols, Z also makes provision for expressions.

D.3 Expressions in Z

Z supports a large number of expressions commonly used in standard programming languages. Examples of these are the commonly supported conditional expression constructs such as, "if" expressions and "while" and "for" loops.

D.4 Naming Conventions

The following naming conventions are used during the development of CBiX.

The names of the Z constructs such as schemas have the first letter of each word capitalized, with the name being formed as one word. For example; GetPrice, AddPrice, GetChildren, Price.

Basic types are one word and formed using capital letters. For example; DOCUMENTCLASS, ELEMENTTYPE, ELEMENT.

Variables used within the constructs such as schemas and axiomatic descriptions are one word and formed using small letter. For example; price, assignedcred.

References

Allot (Ed.). (2000). *Allot and Teleknowledge join forces to offer comprehensive service level management and QOS-based billing solution.* Available from: http://www.allot.com/html/march_27_2000.shtm: Allot.

Apogee Networks, Inc. (2002a). *Bridging the abyss.* Available from: http://www.apogeenetworks.com/swc.cfm?sr_urh=Home.products.NCC.NCC_Case%_Studies.

Apogee Networks, Inc. (Ed.). (2002b). *NetCountant Billing & Settlement.* Available from: http://www.apogeenetworks.com/swc.cfm?sr_urh=Home.products.NCBS: Apogee Networks, Inc.

Bellwood, T., Clment, L., & Riegen, C. von (Eds.). (2003). UDDI version 3.0.1 UDDI spec technical committee specification.

Bichler, M., Kalagnanam, J., Katircioglu, K., King, A. J., Lawrence, R. D., Lee, H. S., Lin, G. Y., & Lu, Y. (2002). Applications of flexible pricing in business-to-business electronic commerce. *IBM Systems Journal*, *41*.

Biskup, J. (2002). Credential-based access control roots and a perspective. In *Proceedings of ISSI - Informationssysteme und Sicherheit.* Fachbereich Informatik, Universitt Dortmund.

Botha, R. A. (2001). *CoSAWoE A Model for Context-sensitive Access Control in Workflow Environments.* Unpublished doctoral dissertation, Johannesburg, South Africa.

Botha, R. A., & Eloff, J. (2002). An Access Control Architecture for XML documents in workflow environments. *South African Computer Journal*(28), 3–10.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., & Maler, E. (Eds.). (2000). *Extensible Markup Language (XML) 1.0 (Second Edition).* Available from: `http://www.w3.org/TR/REC-xml`: W3C Recommendation.

Bullock, A., & Benford, S. (1999). An access control framework for multi-user collaborative environments. In *Proceedings of GROUP'99* (pp. 140–149). Phoenix, Arizona.

Chang, X., & Petr, D. W. (2001). A survey of pricing for integrated service networks. *Computer Communications*, *24*, 1808-1818.

Cisco World (Ed.). (2000). *Standards-Based Group Seeks Advancement of Content Networking.* Available from: `http://www.ciscoworldmagazine.com/backissues/back2001.shtml`: Cisco World.

Clark, J. (Ed.). (1999). *XSL Transformations.* Available from: `http://www.w3.org/TR/1999/REC-xslt-19991116`: W3C Recommendation.

Clark, J., & DeRose, S. (Eds.). (1999). *XML Path Language (XPath) Version 1.0.* Available from: `http://www.w3.org/TR/1999/REC-xpath-19991116`: W3C Recommendation.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., & Samarati, P. (2002). Role-based access control models. *ACM Transactions on Information and System Security (TISSEC)*, *5*(2), 169–202.

Department of Defense National Computer Security Center. (1985). Department of defense trusted computer systems evaluation criteria. *DOD 5200.28-STD.*

DeRose, S., Maler, E., & Daniel Jr., R. (Eds.). (2001). *XML Pointer Language (XPointer) Version 1.0.* Available from: `http://www.w3.org/TR/2001/CR-xptr-20010911/`: W3C Candidate Recommendation.

DeRose, S., Maler, E., & Orchard, D. (Eds.). (2001). *XML Linking Language (XLink) Version 1.0.* Available from: `http://www.w3.org/TR/2001/REC-xlink-20010627/`: W3C Recommendation.

Eastlake, D., Reagle, J., & Solo, D. (Eds.). (2002). *(Extensible Markup Language) XML-Signature Syntax and Processing.* Available from: `http://www.ietf.org/rfc/rfc3275.txt`: W3C Memo.

Emigh, J. (Ed.). (2000). *How Will You Price Content.* Available from: `http://www.inktomi.com/content/ContentBridge-Press.htm`: Content Bridge.

Fallside, D. C. (Ed.). (2001). *XML Schema Part 0: Primer.* Available from: `http://www.w3.org/TR/2001/REC-xmlschema-0-20010502`: W3C Recommendation.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, *4*, 224-274.

Froumentin, M. (Ed.). (2002a). *The Extensible Stylesheet Language (XSL).* Available from: `http://www.w3.org/Style/XSL/`: W3C.

Froumentin, M. (Ed.). (2002b). *The XML Key Management System.* Available from: `http://www.w3.org/TR/xkms2/`: W3C.

Gebauer, J. (2000). *Ba 147 fundamentals of e-business.* Fisher Center for IT and Marketplace Transformation, Haas School of Business.

Gisolfi, D. (Ed.). (2002). *Web services architect, Part 2: Models for dynamic e-business.* Available from: `http://www-106.ibm.com/developerworks/webservices/library/ws-arc2.html?%dwzone=webservices`: IBM Developer Works.

Gudgin, M., Hadley, M., Moreau, J., & Nielsen, H. F. (Eds.). (2001). *SOAP Version 1.2.* Available from: `http://www.w3.org/TR/2001/WD-soap12-20010709/`: W3C Working Draft.

Hampton, K. (Ed.). (2002). *Web Content Validation with XML::Schematron.* Available from: `http://www.xml.com/pub/a/2000/11/29/schemas/part1.html`: XML.com.

Harold, E. R. (2001). *XML Bible, Second Edition.* Hungry Minds, Inc.

Hjelm, J., Martin, B., & King, P. (Eds.). (1998). *WAP Forum - W3C Cooperation White Paper.* Available from: http://www.w3.org/TR/1998/NOTE-WAP-19981030: W3C Note.

Hoque, F. (2000). *e-Enterprise Business Models, Architecture, and Components.* Cambridge University Press.

Housley, R., Ford, W., Polk, W., & Solo, D. (1999). *Internet X.509 Public Key Infrastructure Certificate and CRL Profile.* Available from: http://www.ietf.org/rfc/rfc2459.txt.

IBM (Ed.). (2002). *The power and magic of Digital Content Creation: IBM makes it real.* Available from: http://www-1.ibm.com/industries/media/pdf/DCCBrochure.pdf: IBM.

Jacky, J. (1994). *The way of z: Practical programming with formal methods.* Cambridge University Press.

Kerrigan, R., Roegner, E. V., Swinford, D. D., & Zawada, C. C. (2002). *B2basics.* McKinsey & Company.

Kudo, M., & Hada, S. (2000a). Applications of flexible pricing in business-to-business electronic commerce. *ACM 1-58113-203-4/00/0011.*

Kudo, M., & Hada, S. (Eds.). (2000b). *XML Access Control.* Available from: http://www.trl.ibm.com/projects/xml/xacl/xmlac-proposal.html: IBM Research.

Le Hgaret, P., et al. (Eds.). (2002). *Document Object Model (DOM).* Available from: http://www.w3.org/DOM/: W3C DOM WG.

Lie, H. W., & Bos, B. (Eds.). (1999). *Cascading Style Sheets, level 1.* Available from: http://www.w3.org/TR/1999/REC-CSS1-19990111: W3C Recommendation.

Martin, D. (Ed.). (2002). *How Would You Like That Served.* Available from: http://www.xml.com/pub/au/16: XML.com.

Miller, E., Swick, R., & Brickley, D. (Eds.). (2000). *Resource Description Framework (RDF).* Available from: http://www.w3.org/TR/2000/CR-rdf-schema-20000327: W3C Candidate Recommendation.

Netscape Netcenter (Ed.). (2001). *How Digital Certificates Work.* Available from: `http://wp.netscape.com/security/techbriefs/certificates/howcerts.html?c%p=stbmid`: Netscape Netcenter.

Neuman, B. C., & Tso, T. (1994). Role-based access control models. *IEEE Communications*, *32*(9), 33-38.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of ACM CHI'94 Conference* (pp. 152–158). Boston, MA.

Nokia (Ed.). (2000). *Calling the Next Generation - Nokia Terminals and applications for the 3G market.* Available from: `http://www.nokia.com/press/background/pdf/dec004.pdf`: Nokia.

Nokia (Ed.). (2001). *Nokia Charging Solutions for ensuring mobile data revenues.* Available from: `http://www.nokia.com/Nokia%20%Chargin%20Solutions%20NCC.pdf`: Nokia.

Nokia (Ed.). (2002). *3G Market Overview.* Available from: `http://www.Nokia.com/3G`: Nokia.

Odlyzko, A. (2001). Internet pricing and the history of communications. *Computer Networks*, *36*, 493-517.

Ohto, H., & Hjelm, J. (Eds.). (1999). *CC/PP exchange protocol based on HTTP Extension Framework.* Available from: `http://www.w3.org/TR/NOTE-CCPPexchange`: W3C Note.

Osborn, S., Sandhu, R., & Munawer, Q. (1996). Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM*, *28*(1).

Pemberton, S., Altheim, M., et al. (Eds.). (2001). *XHTML 1.0: The Extensible HyperText Markup Language (Second Edition).* Available from: `http://www.w3.org/TR/2001/WD-xhtml1-20011004`: W3C Working Draft.

Pinpoint Networks (Ed.). (2002). *FUEL Mobile Application Platform.* Available from: `http://www.pinpoint.com/solutions/fuel.html`: Pinpoint Networks.

Rappa, M. (2002). *Business models on the web.* Available from: `http://digitalenterprise.org/models/models.html`: digitalenterprise.org.

Sairamesh, J., Mohan, R., Kumar, M., Hasson, L., & Bender, C. (2002). A platform for business-to-business sell-side, private exchanges and marketplaces. *IBM Systems Journal*, *41*.

Samarati, P. (2002). Enriching access control to support credential-based specifications. *GI Jahrestagung*, 114–119.

Sandhu, R. S., Coyne, E. J., Fenstein, H. L., & Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, *29*(2), 38–47.

Sandhu, R. S., & Samarati, P. (1994). Access control: Principles and practice. *IEEE Communications Magazine*, 40–48.

Sandhu, R. S., & Samarati, P. (1996). Authentication, access control, and audit. *ACM Computing Surveys*, *28*(1).

Shneiderman, B. (1997). Designing information-abundant websites: Issues and recommendations.

Simon, E., Madsen, P., & Adams, C. (Eds.). (2001). *An Introduction to XML Digital Signatures.* Available from: `http://www.xml.com/pub/a/2001/08/08/xmldsig.html`: XML.com.

South African Department of Communications (Ed.). (2000). *A Green Paper on Electronic Commerce for South Africa.* Available from: `http://www.ecomm-debate.co.za/greenpaper/`: South African Department of Communications.

W3C (Ed.). (2002a). *Extensible Markup Language (XML).* Available from: `http://www.w3.org/XML/`: W3C.

W3C (Ed.). (2002b). *Hyper Text Markup Language Home Page.* Available from: `http://www.w3c.org/Markup`: W3C.

W3C (Ed.). (2002c). *What is XSL?* Available from: `http://www.w3.org/Style/XSL/WhatIsXSL.html`: W3C.

Walsh, N. (Ed.). (1998). *Validity.* Available from: `http://www.xml.com/pub/a/98/10/guide3.html`: XML.com.

Willaert, F. (2001). *XML-based Frameworks and Standards for B2B Ecommerce.* Unpublished doctoral dissertation, Katholieke Universiteit Leuven, Departement Toegepaste Economische Wetenschappen.