Value Added Web:
Integrating WWW with a TINA Service Management platform

Aart T. van Halteren, Lambert J.M. Nieuwenhuis, Mike R. Schenk, Maarten Wegdam
KPN Research
PO Box 15000
9700 CD Groningen
The Netherlands
{A.T.vanHalteren, L.J.M.Nieuwenhuis, M.R.Schenk, M.Wegdam}@research.kpn.com

Abstract **- One of the most spectacular developments of this decade is the enormous growth of the Internet. One of the most popular services of the Internet is the World Wide Web (WWW). It may be expected that the Web will be used to provide more sophisticated services, e.g., Video on Demand. Customers will be prepared to pay for such services, because of the exclusive content and the quality of the (broadband) transport network needed to transfer the information. Consequently, we need a way to manage these services, without violating the ease of use provided by current WWW. In this paper we present a solution based on TINA's Business Model. We introduce Value Added Web (VAW), which is an integration of the WWW with TINA Service Management. This combination adds the benefits of the TINA Business Model to the WWW. A VAW session appears as a normal WWW session, except that it allows charging for specific content and the setup of connections with an agreed Quality of Service. The VAW Business Model assumes that users only have a direct relation with a Retailer and that the Retailer is responsible for charging. This paper describes the rationale behind VAW and the design and implementation of a prototype of VAW.**

I. INTRODUCTION

The complexity of the telecommunications market is growing. Legislators require telecommunication service providers to open up their business and allow other players to operate on the market. In addition, traditional telecommunication services such as telephony are gradually outdated by new and advanced services. In particular, Internet technology has a major impact on society and influences the business of telecommunication providers.

TINA [1] identifies a number of business roles relevant for an open telecommunication market that can be used to reduce the complexity of this market. An important aspect of the TINA approach, is the identification of interfaces between the business roles.

Implementation of a TINA Service Management platform depends on the availability of object middleware in the public infrastructure and in the End-user equipment. Today, CORBA [2] and Java technologies are applied in many trials and 'proof-of-concept' projects. However, object middleware is not that widespread used as web technologies such as HTTP and HTML.

In this paper, we describe the integration of web services with a TINA platform, which results in a best-of-breed combination of the TINA business benefits and the widely used WWW. Our approach leads to a Value Added Web (VAW), which creates new business opportunities for Content Providers and Retailers on the Internet and a higher service level for End-users.

Our observations are based on a Value Added Web prototype, which is available on the *EURESCOM Services Platform* (ESP). The ESP is described in a separate paper [3]. We describe the design of the Value Added Web prototype and how we have combined Internet technologies (HTTP, HTML) with object middleware (CORBA, Java).

A. Paper structure

In Section II we explain the VAW Business Model, and compare it to the TINA and Internet Business Models. Section III explains the current Internet and object middleware technologies, and serves as an introduction to Section IV, which explains the design of VAW. Section V describes and evaluates the implementation of our prototype. The paper ends in Section VI with conclusions and future work.

II. TOWARDS AN ENHANCED BUSINESS MODEL

This section first gives a brief overview of the TINA and Internet Business Model. It then compares these models to the VAW Business Models and explains the benefits of the VAW Business Model.

A. TINA Business Model

The TINA Consortium (TINA-C) has defined a Business Model for telecommunications. This model specifies several business roles and business relationships. The roles are the

0-7803-5785-X/99/$10.00 © 1999 IEEE

'consumer role', the '3pty service provider' role, the 'connectivity provider' and the 'Retailer'. The 5th role 'broker' is left out, because is not essential for this paper. The model is depicted in Fig. 1. The relations between the roles are specified by Reference Points, which are described in OMG IDL and informal semantic descriptions.

The TINA business model enables the Consumer to obtain many services from many Service Providers through a single Retailer. This is essential, because the Consumer only has to trust a single Retailer in order to have access to a wide range of services.

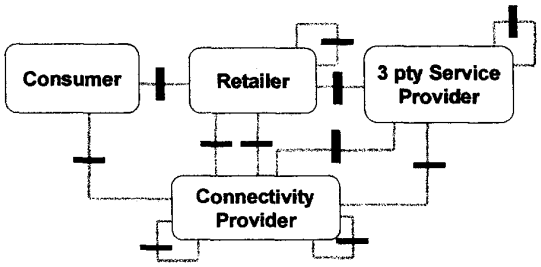


Fig. 1. Simplified TINA Business Model

B. Internet Business Model

The Business Model for the World Wide Web looks like the one shown in Fig. 2. Each End-user has a business relation with an Internet Service Provider (ISP) to get access to the Internet and services such as the WWW. Likewise, the Content Provider has a similar relationship with an ISP to get access to the Internet and provide WWW content. The ISP may also provide web space to the Content Provider. While an ISP has a formal relationship to other ISPs thus creating the global Internet, this relationship merely covers IP connectivity and does not provide a formal retailing relationship. Currently the WWW does not provide end-to-end Quality of Service, because ISPs do not have the mechanisms to set-up an end-to-end connection between a Content Provider and an End-user.

An End-user can have a separate direct relation with each Content Provider, and vice-versa. But since that the content on the WWW is growing rapidly, users have the need for a personalized view on Web content. A Portal, offering users a personal starting point for accessing the Web fulfills this need. Often the Portal brings End-users with similar interests into contact with each other, thus creating a Community.

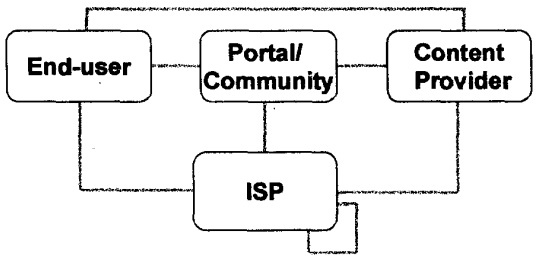


Fig. 2. Simplified WWW Business Model

In this business model, financial transactions can be rather risky because the End-user does not necessarily trust all Content Providers. For example, if a Content Provider wants to charge a user for the retrieval of certain content, this is usually done through the End-user's credit card number, which involves risks for the End-user and the Content Provider. In addition, financial transactions and billing for WWW services can become rather complex for the End-user who has to maintain a relation with many parties, such as a Portal, an ISP and several Content Providers.

This Business Model clarifies that the shortcomings of the World Wide Web in creating a global marketplace cannot only be solved by technical means. An enhancement to the Business Model is required.

C. The VAW Business Model

VAW applies the TINA Business Model to the WWW. In the VAW model, added value is achieved by introducing a Retailer, which acts as an intermediate party between End-users, Connectivity Providers and Content Providers. Fig. 3 shows these business roles and their business relations. In the model, End-users are enabled to retrieve information from the Web as they are for the Internet model.

The Connectivity Provider is responsible for providing network connections between all the other parties. Content Providers offer not only traditional Web content but can also offer more exclusive VAW content. VAW content is characterised by additional attributes, such as cost or specific connectivity requirements (e.g. up to date stock exchange information or an MPEG movie that requires a certain minimum bandwidth). The Retailer has a direct relation with the End-user, the Connectivity Provider and the Content Provider. The Retailer maintains the direct relationship with the End-users through subscriptions and the handling of the billing of obtained services. By administrating and managing the service sessions, the Retailer can register the used services and charge the End-user accordingly. The Retailer is also responsible for communicating the required Quality of Service (QoS) of the network connections to the Connectivity Provider.

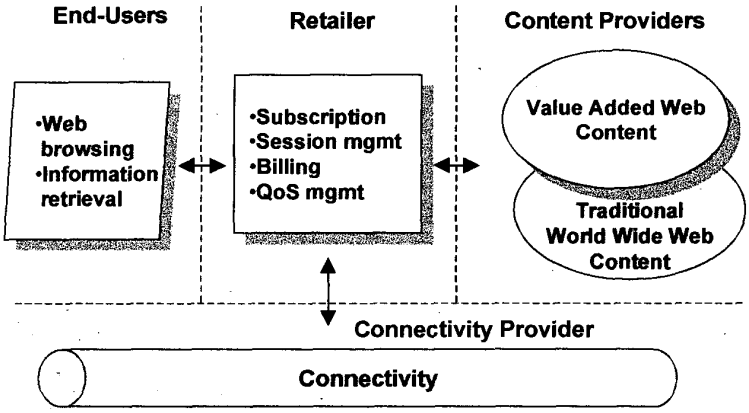


Fig. 3. VAW Business Model

D. Business benefits

The VAW Business Model provides several benefits to the involved parties. Content Providers can charge for valuable content, without dealing with complex billing processes and customer care. Content Providers can keep their existing (Internet) technologies and software. End-users can still use the traditional web, with its browser based user interface and its best-effort services, but they can additionally buy value-added services when required. This includes retrieval of exclusive content, which is currently not on the web because there is no easy way to charge for it. Furthermore, End-users only need to have a direct formal relationship with one actor (the Retailer) instead of formal relationships with all Content Providers. This means that the End-user will only get one bill and has to identify him or herself only once. The Retailer can benefit in this model, by adding an extra charge to the content charge or by creating package deals by combining information from several Content Providers.

III. INTERNET AND OBJECT MIDDLEWARE TECHNOLOGIES

This section compares Internet technologies with object middleware technologies. Internet technologies such as TCP/IP, HTTP and HTML form the basic ingredients for the WWW. Object middleware technologies such as CORBA, IDL, Java and IIOP form the basic ingredients for the TINA Service Management platform [3]. Comparing these technologies allows us to make design choices for the implementation of a VAW prototype. The comparison is made from the perspective of information retrieval systems.

A. Internet technologies

From a technology perspective, the WWW is based on a few very simple and widespread technologies. The transport protocol is TCP/IP, the application protocol is HTTP and the content is formatted as HTML documents, see Fig. 4. These three technologies (TCP/IP, HTTP and HTML) have enabled End-users of the WWW to access huge amounts of information through web pages.

Some of the strong points of these Internet technologies are:

- The HTTP protocol [4] uses two basic messages (Request and Reply), which makes it simple and relatively easy to implement. The fact that HTTP engines have been implemented in many different languages for virtually every computing environment supports this.
- The HTML definitions [5],[6] are based on tagged data definitions for structuring HTML documents. In essence, an HTML document is a long string where data tags are used for structuring the document. This makes HTML documents platform independent and they can be authored and displayed on many computing environments. This makes content provisioning on the Internet relatively easy.
- The features of HTTP and HTML as described above have enabled wide support in both browsers and servers.

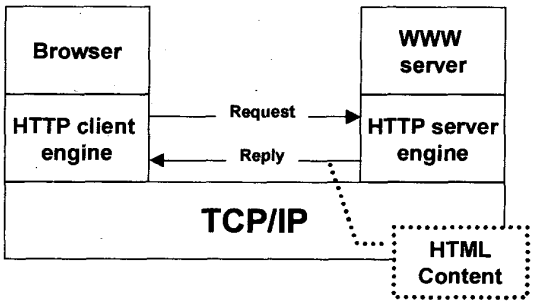


Fig. 4. WWW technologies

Web content can be authored, viewed and served from virtually every computing environment.

Some of the weak points of Internet technologies are:

- Information retrieved through HTTP is transported over TCP/IP. The implication is that Quality of Service (QoS) parameters, such as delay and throughput, are determined by factors that can not be influenced by the End-user. For example, the number of concurrent End-users on the Internet has a major impact on the QoS experienced by the End-user. In fact, End-users only have a best effort QoS.
- The current support for secure transactions is diverse. Several solutions exist for paying over the Internet, including DigiCash, CyberCash and Millicent [7]. Other mechanisms rely on a secure connection [8], [9] between the End-user and Content Provider using credit cards for payments. The diversity in technologies withholds Content Providers from investing in secure transaction solutions and reduces the e-commerce possibilities of the Web. Given the diversity in financial banking mechanisms worldwide, it is not expected that a universally accepted means of electronic money transfer will be available in the near future.
- HTTP servers respond to a client request without relating that request to previous or subsequent requests. HTTP does not provide support for session management. Ongoing work [10], [11] standardises the notion of cookies to allow clients and servers to exchange state information and to maintain a session. However, currently not every Web browser handles cookies in the same way, thus limiting the use as a generic means for identifying End-users.

Value Added Web eliminates the drawbacks of current Internet technologies, while taking advantage of the strong points. Before we go into the design of VAW, we evaluate the technological characteristics of object middleware.

B. Object middleware technologies

Object middleware enables the interworking of distributed objects, by hiding the heterogeneity of the underlying computing and network resources. We used CORBA [2] for our prototype, which is a standard for interoperability of distributed objects that is specified by the Object Management Group (OMG). The main vehicles for achieving interoperability are the Object Request Broker (ORB), the Internet Inter-ORB Protocol (IIOP) and the Interface Definition Language (IDL). The ORB is responsible for locating objects and facilitates interaction between distributed objects. Fig. 5 depicts the main components of CORBA that enable interoperability.

Some strong points of CORBA are:

- IDL is a declarative language for specifying interfaces in a programming language independent way. CORBA objects can be developed in many programming languages for many operating systems. This is supported by the fact that IDL compilers are available for most programming languages and operating systems.
- Interworking between objects is facilitated by the IIOP protocol. Each CORBA 2 compliant ORB implements this protocol and interoperability between products of CORBA vendors has been proven.
- ORBs implemented in Java are available on the market. Applications and applets written Java can use the ORB to communicate over the Internet to back-end services. It seems that CORBA/Java technology is a strong candidate for running object middleware on End-user equipment

Some weak points of CORBA technologies are:

- Although OMG-IDL is well a supported language for describing object interactions, CORBA does not provide a language for representing content. The lack of a standard way for representing content, which is comparable to HTML, requires applications to define

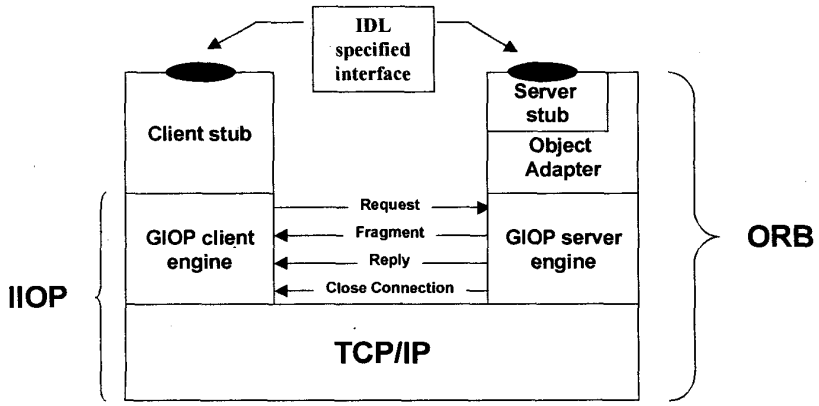


Fig. 5. CORBA interoperability components

their own way for representing content. Information retrieval systems based on pure CORBA technology have to build their own End-user application for representing content. There is no equivalent of a Web browser for CORBA objects.

- The current version of IIOP uses seven basic messages, which makes it more complex then HTTP. The IIOP protocol is not as widely supported as HTTP.
- Internet technologies are much more used than CORBA technologies, and thus from a maturity perspective CORBA is less 'proven-technology' then Internet technology.
- Some Web browsers now support a Java implementation of CORBA. However, no Web browsers are available that implement the portable stub specification (Chapter 24 in [2]). Therefore, Java applets that want to take advantage of CORBA are not portable.

Based on our evaluation of Internet technologies and CORBA technologies, we now discuss the design of Value Added Web. Our aim is to end up with a best-of-breed solution.

IV. DESIGN OF VALUE ADDED WEB

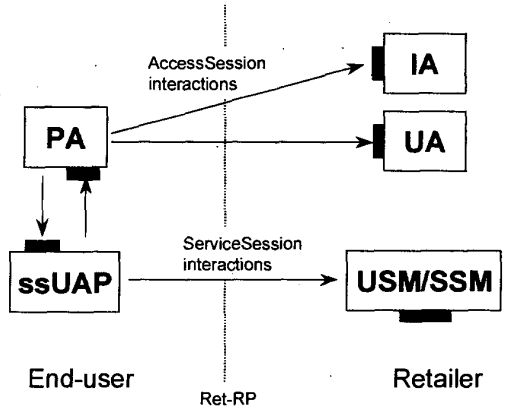
The design of VAW is based on three goals. These goals are:

- It should support the Business Model as explained in Section II, with the Retailer to act as a mediator between Content Providers and End-users.
- It should take into account the characteristics of the Internet and object middleware technologies as described in Section III. In particular for both the End-user and the Content Provider there should only be minor changes compared to the 'normal' Internet situation.
- It should fit into a TINA Service Management Platform, which is in this case the EURESCOM Services Platform [12],[13].

The remainder of this section describes the technical design of a prototype that demonstrates the feasibility of the proposed Business Model. Section A explains the computational objects that are used for service provisioning and which objects must be added to realise the VAW service. Section B describes the essence of our approach for integrating Internet technology with CORBA technology. Based on that approach, Section C describes how Content Providers can use VAW for content charging. Finally, Section D describes how a VAW service session is initiated from a TINA access session.

A. Multiprovider service provisioning

The TINA Service Architecture in general and more specifically the EURESCOM Services Platform allows End-users to choose between Retailers. This requires standard interfaces between the objects in the End-user domain and the Retailer domain. As a minimal set of standardisation, the access part of the Ret-RP was implemented. The service session related interactions were not standardised and left open to the implementers of a service. For a more detailed description of the EURESCOM Services Platform and interoperability issues, we refer to the literature [12]. The



- Fig. 6. Computational objects in End-user and Retailer domain

description is limited to the computational objects that are specific for the VAW service.

Fig. 6 shows the computational objects needed for service provisioning. In the Retailer domain the Initial Agent (IA) and the User Agent (UA) allow the End-user to set-up an access session with the Retailer. The User Session Manager (USM) and/or the Service Session Manager (SSM) deal with the service session related interactions. In the End-user domain, the Provider Agent (PA) initiates an access session and consequently presents the End-user with list of subscribed services offered by the Retailer. When the End-user starts a service, a service session User Application (ssUAP) is started, which communicates with the USM and/or the SSM.

The computational objects in the End-user domain are built as Java applets, using a Java implementation of CORBA. We decided on this technology constraint in order to have downloadable End-user software.

A new service can be added to the EURESCOM Services Platform by implementing the service specific objects, such as an ssUAP, a USM and an SSM.

B. VAW service sessions

To include the VAW service in the platform, we must design VAW specific version of the appropriate computational objects as described in the previous section. In addition, the design should take advantage of the strong points of CORBA and Internet technologies while trying to eliminate some of the weak characteristics of these technologies.

VAW uses HTTP for retrieving HTML content, so the End-user uses VAW through a normal Web browser interface. CORBA is used to manage to additional features such as content charging and initiating high-bandwidth connections. Our design requires a mechanism for intercepting HTTP requests that are initiated by the End-user. This is realised as an extension on an HTTP-proxy.

The HTTP-proxy links browsing actions of the End-user to a TINA service session. The TINA service session is managed by VAW specific SSM called VAW-SSM. Fig. 7 shows the configuration of objects that we use to achieve our goal. In this figure we assume that the End-user has already started a VAW service session. We also assume that the End-user is browsing for content using a standard browser, which is configured to work through an HTTP-proxy.

The following steps show how the Retailer can intervene in the browsing actions of the End-User.

1. The browser sends an HTTP request to the HTTP-proxy.
2. The proxy notifies the SSM object that can then analyse the request and take appropriate action. For this scenario, we assume no special action is needed and it is only important to note that at this stage the Retailer has full control of the HTTP-request and transform or redirect it in whatever way is necessary.
3. Since no special action is needed, the default action is that the SSM informs the HTTP-proxy to forward the request to the WWW server.
4. The WWW server returns the content to the HTTP-proxy.
5. The HTTP-proxy sends an HTTP reply to the browser.

The next section describes what special actions a Retailer can undertake by describing what happens when a Content Provider wants to charge for content.

C. Content charging

VAW aims to make content charging very easy for Content Providers. Basically, the content can be stored on any WWW server. The Content Provider should only allow HTTP requests originating from the HTTP-proxy to the WWW server. Direct requests to the WWW server not originating from the HTTP-proxy should be blocked.

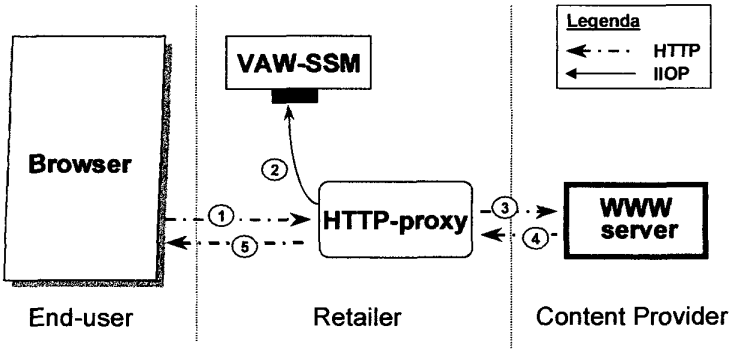


Fig. 7. Adding an SSM in the HTTP request chain

When a Content Provider wants to charge for content, an additional component called the ContentRepository is required. This component keeps a database with meta-information of all content that is offered by a provider. This meta-information includes the price of the content. The ContentRepository should be accessible by the Retailer and we use CORBA/IIOP for accessing the Repository.

Fig. 8 shows a slightly simplified operation sequence when an End-user requests VAW content, with a price tag. The following steps are involved:

1. HTTP request to HTTP-proxy.
2. The HTTP-proxy calls the SSM to check if the requested URL requires special action.
3. The SSM notes that the requested URL belongs to a VAW Content Provider and calls the Content Repository to retrieve content meta-information (i.e. cost).
4. The meta-information contains a price tag for the requested URL and the HTTP-proxy is informed that the End-user will be charged for the requested information.
5. The HTTP-proxy sends an HTTP reply containing a warning to the End-user about charging, including the charge for the requested information.
6. If the End-user agrees with the charge, an HTTP request containing an acknowledgement is sent from the browser to the HTTP-proxy.
7. The HTTP-proxy forwards the original HTTP-request (initiated in step 1) to the WWW server.
8. The HTTP-proxy receives the content.
9. The content is forwarded to the browser.

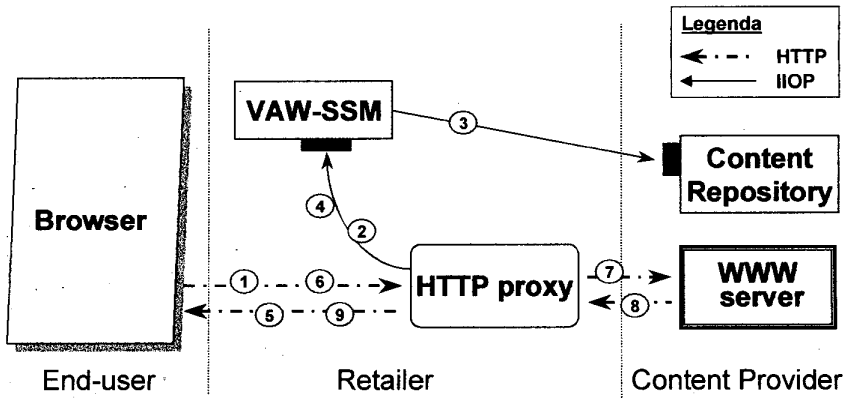


Fig. 8. Charging for content

The mechanism described above allows a Retailer to charge for content on behalf of the Content Provider. After step 6 in the scenario above, the VAW-SSM can send a billing event to a billing application so the charge is put on the bill of the End-user or use any other billing mechanism that End-user and Retailer have agreed upon.

The mere responsibility of the Content Provider is to stock the Content Repository with the correct meta-information and give the Retailer access. The Retailer handles billing of individual End-users. Billing between the Content Provider and the Retailer is based on an aggregation of all the content charges.

D. Initiating a VAW session

In our prototype, we have an instance of an HTTP-proxy and a SSM object for each service session. This means that the HTTP-proxy should invoke the appropriate SSM based on the service session id of the End-user. In the previous sections, we have assumed that the HTTP-proxy knows which SSM to address. The remainder of this section describes how this relation is established when an End-user initiates a VAW session.

Our basic assumption for this design is that the HTTP requests from the browsers contain some form of identification of the End-user. This identification could be a well-known HTTP header field, which is included with every HTTP request by the browser. Our current implementation uses the IP address of the End-user machine as a means for identification. This is not a bullet-proof solution, and in Section V.B we describe other means for identification. For now, we refer to the identification tag of the End-user as the VAW-id.

The following steps are needed to initiate a VAW session (see Fig. 9 for a visual representation of these steps):

1. The user selects VAW from a list of subscribed services and as a result of that the PA creates new browser window with the URL for the ssUAP. This is a standard step required for initiating a service session in the EURESCOM Services Platform.
2. HTTP request to get the ssUAP. This is a request for HTML page that includes a tag for the ssUAP applet. The VAW-id is not included in the HTTP request or the VAW-id is invalid (i.e. not known by the HTTP-proxy). The HTTP-proxy generates a new VAW-id and stores this.
3. The HTTP request is forwarded to the WWW-server
4. The WWW server returns the content (this is a page with the ssUAP applet and some parameters to this applet). The HTTP-proxy adds the VAW-id, generated in step 2, as a parameter to the ssUAP applet by modifying the HTML content.
5. The content is returned to the browser who starts the ssUAP applet.
6. The ssUAP calls "createssUAPfinished". This method is part of the intra-domain reference point described in [14].
7. The PA calls "startServiceInit". This method is also part of the intra-domain reference point described in [14].
8. The ssUAP calls "startService" on the PA (this call returns after step 12). This call obtains the VAW-id from the applet parameters in the HTML page and forward the VAW-id as a ServiceProperty parameter of the startService call
9. The PA calls "startService" on the UA (forwarding the ServiceProperty parameter from step 8). This is a standard call defined in the Ret-RP [15].

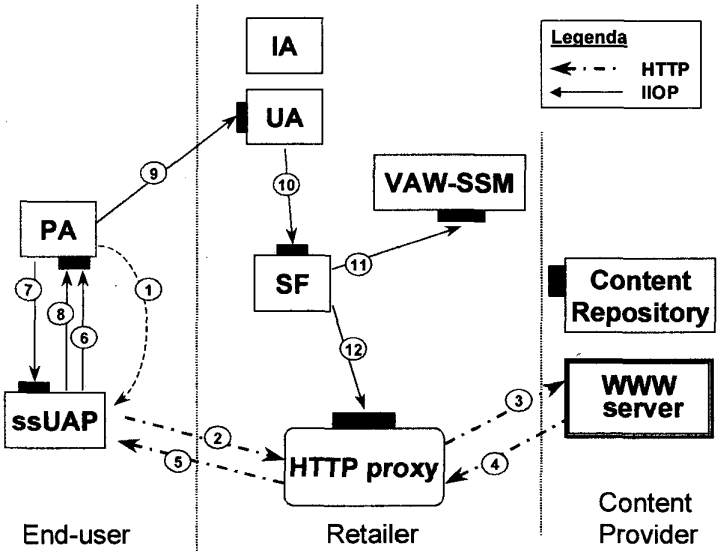


Fig. 9. Initiating a VAW session

10. The UA calls "createSSM" on the ServiceFactory (SF) of VAW.
11. The SF creates a VAW-SSM
12. The SF informs the HTTP-proxy about the IOR of the VAW-SSM for a particular VAW-id. The HTTP-proxy stores this in a table and uses this for subsequent requests.

These steps conclude the initiation of a VAW service session. The End-user can now transparently browse the WWW and whenever content from a VAW Content Provider is requested, the HTTP-proxy addresses the VAW-SSM that belongs to the service session of the End-user.

V. PROTOTYPE IMPLEMENTATION

Several components were implemented to realise the VAW prototype. We built the PA and VAW-ssUAP with Java (jdk1.1x) and OrbixWeb 3.0x. The IA, UA, SF, Content Repository and VAW-SSM were al built using C++ and OrbixOTM 1.0x.

The HTTP-proxy was built using a HTTP-proxy called Muffin [16]. This proxy is implemented in Java and supports HTTP/0.9, HTTP/1.0, HTTP/1.1 and SSL. Essentially, Muffin is a proxy server that manages user supplied filters. These filters can filter HTTP messages and possibly modify or redirect these messages. We have implemented a specific *VAW filter* that checks for HTTP requests directed at VAW Content Providers. If needed the VAW filter contacts VAW-SSM instances through standard CORBA calls. Requests for unregistered providers are forwarded without contacting the SSM.

Fig. 10 shows the output window (for logging purposes) of the VAW filter just after start-up. The window shows the names of registered servers of VAW Content Providers. Each HTTP request is checked against this set.

Our current implementation of the VAW filter uses the IP number of the browser to relate an HTTP request to a service session. The VAW filter uses OrbixWeb to communicate to the VAW-SSM.

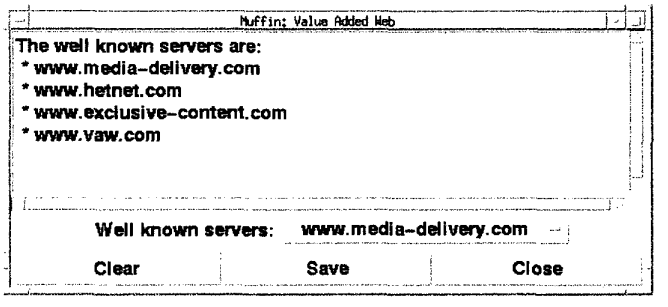


Fig. 10. GUI for Value Added Web filter

A. Performance and scalability

The performance and scalability issues of the VAW depend on the used service platform, and on some VAW specific issues. Although we have not done elaborate measurements, we can make some statements on the VAW specific performance and scalability.

The HTTP proxy we used (the Muffin) is a pure Java implementation, with the well-known Java performance

problems. Re-implementing it with a compiled language, using a sophisticated JIT compiler or a Java-to-native code compiler should increase the performance.

Since the proxy is kept almost stateless, the problem of the scalability of the proxy becomes similar to the scalability of 'normal' HTTP proxy's. Keep in mind that the only overhead for a non-charged HTTP request is the check if the HTPP request is for a 'well known server'. When the HTTP-proxy becomes a bottleneck, it can also be distributed over a pool of machines.

The service specific components like the SSM objects can be distributed over a number of machines, possibly using load-balancing mechanisms. Many scenarios are possible for scaling VAW to a large number of users with acceptable response time.

B. Evaluation of the prototype

A working prototype of VAW is integrated with a TINA compliant Service Management platform. There are however several improvements and variations we are considering.

First of all in the current implementation the HTTP-proxy identifies the service session by the IP number. This limits the number of service sessions: i.e. only one service session per client machine and is a great security hole. We find this an undesirable limitation and are considering other forms of identification. A possible solution would be to use cookies [10] [11]. However, in the cookie mechanism a cookie is specific for a domain and path combination and is not automatically included in every HTTP request. The cookie mechanism is therefore not a suitable solution. A better solution would be to implement a customized browser and define a specific HTTP-header field that contains the VAW-id and is included in every HTTP-request. A possible browser extension we are considering is an integration of the ssUAP with the browser, instead of starting a separate browser window with an ssUAP applet. A third reason for building a customized browser is we can then use IIOP instead of HTTP/HTML for the confirmation of a content charges (see step 5 and 6 of Section IV.C).

Security is an important issue for services like VAW. VAW depends for its security on the general security of the service platform and also on some VAW specific issues. This platform security is a separate issue that is outside the scope for this paper. The VAW specific security issues have mainly to do with the communication between the browser and the HTTP-proxy. E.g. it is very important that others can not guess the VAW-id. Some of these issues could be resolved by state-of-the-art HTTP security solutions, others are still open for investigation.

For our current prototype we limit ourselves to HTTP, and did not consider other Internet protocols. This means that some content, for example content retrieved through ftp or a streaming protocol, can not be charged as VAW content.

It should be possible to have much more sophisticated charging schemes than simply paying for a certain HTTP-request. One can think of subscription discounts for regular users, differentiating costs for different group of End-users etc. Even a loyalty program could be created, e.g. by handing out some form of 'frequent browsing miles'.

There are other possibilities to implement the Content Provider functionality. One possibility would be to send the meta-information along with the reply of the HTTP request from the WWW server to the HTTP proxy, for example as a reply header. Although this would eliminate some IIOP calls, this might require more processing by the HTTP-proxy making it 'fatter' and less scalable.

Currently the ContentRepository is located at the ContentProvider, because the ContentProvider is responsible for determining the content meta-information. However, for performance or system management reasons it could just as well be located at the Retailer. The ContentProvider would only get a specific interface for modifying the meta-information.

The TINA Business Model specifies a reference point between the Retailer and the 3rd Party Service Provider. This relation between Retailer and 3rd Party Service Provider in the pure TINA Business Model is quite similar to the relation between the Retailer and Content Provider in the VAW Business Model. A possible extension could be to make the VAW Content Provider a more TINA like service provider, including support for access sessions between the Retailer and Service Provider. However, this requires additional effort from Content Providers and may withhold them from providing VAW content. The benefit of our current design is that a Content Provider can keep using common Internet technologies, and can thus very easily enter the market.

VI. CONCLUSIONS AND FUTURE WORK

Value Added Web integrates the WWW with the TINA Service Architecture. VAW combines the benefits of the TINA Service Architecture with the benefits of the WWW. The combination leads to a more advanced Business Model then the current Internet Business Model. VAW adds the benefits of the TINA Business Model to the WWW.

For the End-user a VAW session appears as a normal WWW session. We have demonstrated how End Users can be charged for valuable content in a transparent way. Billing for this content is handled in a uniform way and is independent

of the Content Provider. The VAW Business Model assumes that End-users only have a direct relation with a Retailer and that the Retailer is responsible for charging and billing. This eliminates the need for Content Providers to handle charging for every End-user, and for End-users to establish a separate (financial) relation with every, possible non-trustworthy, Content Provider.

From a technology perspective the VAW prototype demonstrates how Internet technologies and object middleware (CORBA, Java) can be integrated to provide End-users with information retrieval services. Our prototype implementation demonstrates the feasibility of a hybrid Internet/CORBA solution.

Potentially, VAW can be used to initiate dedicated connections. End-users that are willing to pay for high-bandwidth connections with QoS guarantees can be catered for. For this scenario the Retailer must establish a relation with one or more Connectivity Providers and define an interface for connection establishment. The current prototype does not yet support the set-up of dedicated streams to get a guaranteed QoS. We did not consider this a high priority because of the technology constraint of our service platform, which was limited to IP over ISDN. In case of other network environments, for example with ADSL and ATM networks, it should be possible to set-up dedicated connections.

With e-commerce over the Internet becoming more and more popular, the possibilities to use the TINA architecture to provide an e-commerce platform should be examined. The VAW concept could be extended for such a purpose, since it already provides billing and integration with the WWW. Also charging for the purchase of products at different 'shops' is not such a big step if 'shops' are simply considered as a special kind of Content Provider, and products as special kind of WWW content.

The OMG is working on the Telecommunication Service Access and Subscription Service[17], which resembles the Ret Reference Point. The VAW design and prototype could most likely be migrated to this new specification.

Finally, future work on VAW could include an investigation on the impact of HTTP-NG on our design considerations. The direction that ongoing work on HTTP-NG is taking, makes HTTP-NG a possible alternative for IIOP, in particular for interactions between End-users and a Retailer. This is for further study.

ACKNOWLEDGMENTS

The work discussed has been performed within the context of EURESCOM project P715, funded by EURESCOM. The authors would like to thank participants of P715 and colleagues within their company for their collaboration in this work, and for ideas, comments and fruitful discussions.

REFERENCES

[1] Yates, M., Wataru, T., Demoudem, L., Jansson, R., and Mulder, H. "*TINA Business Model and Reference Points*". 20-5-1997. TINA-C.

[2] OMG. "*The Common Object Request Broker: Architecture and Specification*", Version 2.2. 1998.

[3] Nieuwenhuis, L. J. M and Halteren, A. T. van. "*EURESCOM Services Platform*". TINA'99

[4] IETF. "*Hypertext Transfer Protocol – HTTP/1.1*". RFC 2068. 1997.

[5] W3C. "*HTML 3.2 Reference Specification*", W3C Recommendation . 14-1-1997.

[6] W3C. "*HTML 4.0 Specification*", W3C Recommendation . 24-4-1998.

[7] W3C. "*W3C Security Resources, http://www.w3.org/Security/*". 1999.

[8] Farrell, S. "*An Internet AttributeCertificate Profile for Authorization*". 20-8-1998. IETF, Transport Layer Security (TLS)

[9] Rescorla, E. "*HTTP Over TLS*". 1998. IETF, http://search.ietf.org/internet-drafts/draft-ietf-tls-https-01.txt. 1999

[10] Netscape. "*http://home.netscape.com/newsref/std/cookie_spec.html*". 1999.

[11] W3C. "*HTTP State Management Mechanism*". RFC 2109. 1997.

[12] Fisher, M. A., Egelhaaf, C., and Rana, S. "*Interoperability in a Multiple-Provider Telecommunications Environment*", Proceedings of EDOC'98. 1998.

[13] Rana, S., Fisher, M. A., and Egelhaaf, C. "*Implementation and Interoperability experiences with TINA Service Management Specifications*". TINA'99

[14] Egelhaaf, C. and Fisher, M. A. "*Intra-domain reference point specification*". 1998. Deliverable of EURESCOM P715

[15] TINA-C. "*Ret reference Point Specification*". 1998. TINA baseline, Version 1.1

[16] Boyns, M. "*http://muffin.doit.org/*". 1999.

[17] OMG. "*Telecommunication Service Access and Subscription*". 1998. Request for Proposals, OMG Document/98-09-07