(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 June 2002 (06.06.2002)

PCT

(10) International Publication Number
WO 02/45405 A2

(51) International Patent Classification⁷: H04N

(21) International Application Number: PCT/US01/44286

(22) International Filing Date:
28 November 2001 (28.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0029110.4 29 November 2000 (29.11.2000) GB

(71) Applicant *(for all designated States except BB, US)*: **AGENCY.COM LIMITED** [GB/GB]; 8 Crinan Street, Battlebridge Basin, London N1 9SQ (GB).

(71) Applicant *(for BB only)*: **AGENCY.COM LTD.** [US/US]; 20 Exchange Place, New York, NY 10005 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CHONGTAY, Rocio, A.** [DK/DK]; Bremensgade 17, 2.th., DK-2300 Copenhagen S. (DK). **BRUUN, Jesper** [DK/DK]; Kastanie Alle 26B, st., Vanloese (DK). **KOCH, Martin** [DK/DK]; Klokkerhoejen 14, 4.th., Copenhagen NV (DK). **ANDERSEN, Peter, Holst** [DK/DK]; Stenkrogen 17, Lyngby (DK). **LASSEN, Lars, K.** [DK/DK]; L1. Kongensgade 4, 2.th., Copenhagen K (DK). **SPORK, Maz** [DK/DK]; Solbejergvej 26, 2.tv., Fredriksberg C. (DK). **LAWAETZ, Sten** [DK/DK]; Frederiksdalsvej 8E, 2.tv., Virum (DK).

(74) Agents: **ALTMILLER, John, C.** et al.; Kenyon & Kenyon, Suite 700, 1500 K Street, N.W., Washington, DC 20005 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** METHOD AND APPARATUS FOR BUILDING A PLATFORM SPECIFIC DATA SERVICE

(57) **Abstract:** Methods and apparatus are disclosed for building a data service for transmission to a selected type of data display platform, such as a particular type of interactive television set top box. Such data services generally comprise a data component and a software application component. The software application component executes on a data display platform to display elements of the data component, in response to commands generated by a user. In the present disclosure an application data model is used to build a software application component which is able to access data configured to conform to the application data model. Service content data in a first configuration generic to a variety of data display platform types is then converted to a second configuration conforming to the application data model, in order to provide the data component. The use of a content data model intermediate to the raw formats of the service content data and the application data model is also disclosed.

METHOD AND APPARATUS FOR BUILDING A PLATFORM SPECIFIC DATA SERVICE

The present invention relates to methods and apparatus for building a platform specific data service for transmission to and use on data display platforms of a type selected from a plurality of data display platform types.

In particular, but not exclusively, the invention relates to methods and apparatus for building interactive television data services for broadcast to and use on television set top boxes.

Referring to figure 1, there is shown a schematic of a typical interactive television system, in which data flows are shown by arrows. A service provider 10 receives changeable data service content, such as the latest news, weather or television schedules, from a content provider 12 and integrates the data service content with appropriate software application components to create one or more data services. The service provider 10 forwards the one or more data services to a network operator 14. The network operator encodes the data services as appropriate and broadcasts them as a series of modules contained in a broadcast stream across one or more of a variety of broadcast paths 16 such as terrestrial digital television, satellite television or cable television broadcast paths. Typically, a single data service is broadcast within a single television channel also carrying one or more audio streams and one or more video streams. Alternatively, a data service may be broadcast as the sole element of a television channel.

The broadcast data services are received by a large number of set top boxes 18. In each set top box 18 the signal of the interactive television channel chosen by the user of the box is decoded and audio and video signals, if present, are displayed on a television set 20. One or more selected modules of the data service broadcast in the chosen channel are loaded into memory in the set top box 18 as they are received and software application components thus loaded are executed to provide the data service to the user of the set top box, using the television set 20 as the display device.

The user of a set top box 18 is able to interact with and control a software application component executing on the set top box by means of a remote control unit 22 or keyboard device, for example in order to browse the data service content provided with the data service. Typically, a return data path 24 is also provided, for example by using the public telephone network 26, to enable the set top box to transfer data back to the service provider 10 or another site, and to provide an additional and user-selective data path to each set top box 18. The return data path 24 enables users to send email, carry out commercial transactions and so on. Data services may be partly or entirely transmitted to users over the return data path 24 or some other user-selective path, if required.

A number of different types of set top box are currently in existence, each providing different hardware capabilities, for example in terms of memory size, screen display hardware, data decoding facilities and so on. Any one set top box may be loaded or preloaded with one or more of several different proprietary operating systems such as OpenTV, Mediahighway or Betanova (Dbox), or versions of such operating systems, each providing a different software environment to software application components executing on the set top boxes, for example by way of differences in interpreters and run time libraries. Other systems such as Microsoft TV and Liberate operate under a pull-type schema where the user selects the material to be transmitted, for example over a cable network.

The variety of different set top box and other data display environments presently in use makes it difficult to provide the same data service over a variety of different networks, while making optimum use of the facilities available in each type of set top box.

A further difficulty lies in providing the same or a similar data service to a wider range of platforms, for example to personal digital assistants (PDAs), wireless application protocol (WAP) equipped mobile telephones and world wide web (WWW) browsers, using a common data service building system or methodology in order to maintain consistent performance, maximise reuse of data service content, and minimise costs.

Other differences in data service requirements may relate to, for example, data transmission and bundling requirements imposed by network operators, and the requirement of providing the same data service to a number of geographical regions having different language, currency or information needs. Network operators may further impose data service constraints relating to commercial, aesthetic and various other types of policy.

Consistently reliable performance, and robustness to problems such as computer crashes and power supply failures, is very important in the provision of consumer data services, and in particular for television services. Attempts to provide the same data service to a wide range of set top boxes, to a number of different network operators, to a number of different user groups, and perhaps to data display platforms other than interactive television set top boxes, or any combination of the above, is likely to lead to an increase in problems with spurious data service content and inconsistencies between data and software application components, leading to unacceptable data service reliability.

The present invention seeks to address these and other problems of the related prior art. According to a first aspect of the invention there is provided a method of building a platform specific data service for transmission to and operation on data display platforms of a type selected from a plurality of data display platform types, the platform of selected type having data service requirements different from those of other ones of said plurality of platform types,

the platform specific service comprising a software application component and a data component,

the software application component being operable to cause a platform of said selected type to display elements of said data component in response to commands generated by said user,

the method comprising the steps of:

providing an application data model;

using said application data model to build said software application component to be operable to access data configured to conform to said application data model;

providing service content data in a first configuration generic to said plurality of platform types;

converting said content data from said first configuration to a second configuration conforming to said application data model; and

using said content data in said second configuration to provide said data component.

Either or both of said software application component and said data component may be specific to said selected platform type.

Multiple instances of a software application component may be built for transmission to multiple selected types of data display platforms. These platforms may typically be interactive television set top boxes having various hardware and operating system resources and set ups, receiving television channel broadcasts containing video, audio and data streams. However, other types of data display platforms may be used such as pagers, mobile telephones, personal digital assistants and the like. Different platform types may also be distinguished by geographical or language requirements. For example, a number of platform types providing predominantly or exactly the same software environment and hardware facilities may be distinguished by the geographical requirements of the users, such as the need for local news and weather information, information in one of several natural languages and so on. Platform types may also be distinguished by differences in network operator or broadcast medium constraints, such as data bundling, repeat rate, network operating policy and other constraints.

Television and/or data services may be transmitted or broadcast over conventional satellite, terrestrial radio or cable type media, or may be obtained by users over a data return path 24 or similar data link. Such a data link may be provided, for example, by telephone, data, cable TV or other telecommunications networks.

The use of a common application data model in the building of a software application component, at a service creation time, and in the configuring of transient service content data prior to broadcast or transmission as a data component at service operation time, enables the service provider to ensure that data components and software application components are compatible, leading to improved reliability of the service. Preferably, the application data model is used to carry out an explicit check that the service content data cast into the second configuration, for subsequent transmission or broadcast, is compatible with the software application to be transmitted.

According to a second aspect of the invention there is provided a method of building a data component of a platform specific data service for transmission to and operation on data display platforms of a type selected from a plurality of data display platform types, the platform of selected type having service requirements different from those of the other ones of said plurality of platform types

the platform specific data service typically comprising a software application component as well as said platform specific data component,

the software application component being operable to cause a platform of said chosen type to display elements of said data component in response to commands generated by said user,

the method comprising the steps of:

providing a content data model generic to said plurality of platform types;

providing an application data model representative of a configuration of the data component readable by the software component;

receiving service content data in a raw configuration;

rendering said service content data from said raw configuration into a content data model configuration conforming to said content data model;

rendering said service content data from said content data model configuration into an application data model configuration conforming to said application data model; and

using said service content data in said application data model configuration to provide said data component.

Either or both of the software application and data components may be specific to the selected platform type. The software application may be included in the data service transmitted to the platforms, or may be separately transmitted or already resident.

The service content data received in the raw configuration may generally be in one or more arbitrary formats such as plain text files containing delimited lists or paragraphs of text, binary graphics files and so on. These files may, for example, contain information such as recent news headlines and stories, weather information, television programme schedules, commercial catalogues and the like.

The content data model is used to define a structure into which the raw configuration service content data may conveniently be rendered and stored, but which is not limited to structures or configurations specific to particular platform types or families of platform types.

The service content data is then rendered into the application data model configuration, which conforms to the application data model, and hence is structured according to the requirements of the service software application.

This multi-staged reconfiguration process allows for increased flexibility in configuring the same content data for transmission to different platform types. It also allows for greater flexibility in defining the logical and physical boundaries between content providers who provide the raw content data and service providers who build the data service. For example, the content provider could take responsibility for ensuring that data forwarded to a service provider is already in a configuration conforming to the content data model, rather than the service provider carrying out this stage of reconfiguration. The step of rendering the content data into the configuration conforming to the data content model could thereby be carried out centrally before forwarding the data to a number of different service providers, such as interactive TV network operators, for rendering into the application data model configuration and for subsequent transmission to data display platforms of various types.

Advantageously, the step of rendering the content data from the content data model configuration to the application data model configuration may be carried out via an intermediate configuration. By choosing an industry standard intermediate configuration, such as the Extendable Markup Language (XML), the number of standard tools available for manipulation and rendering of the data into the application data model configuration may be increased. The intermediate configuration may also be chosen to make rendering of the content data into a pull-type configuration, to be made available on a pull-type service, more straightforward. For example, if XML is chosen for the intermediate configuration then translation into HTML (HyperText Markup Language) or a similar markup language for making the content data available on an Internet world wide web site or other pull-type service is facilitated.

According to a third aspect of the invention there is provided apparatus for building a platform specific data service for transmission to and operation on data display platforms of a type selected from a plurality of data display platform types, the platform of selected type having data service requirements different from those of other ones of said plurality of platform types, the platform specific data service comprising a data component,

the apparatus comprising:

a content acquirer operable to receive content data in a raw form and to reconfigure said raw form content data to conform to a content data model; and

a content instance builder operable to receive said content data model conformed content data and to reconfigure said content data to conform to an application data model to thereby provide said data component.

Preferably, the platform specific data service further comprises a software application component operable to cause a data display platform of the selected type to display elements of said data component, the apparatus further comprising an application builder adapted to receive said application data model and to build said software application so as to be operable to utilise said data component.

The apparatus may be provided by one or more computer systems programmed appropriately. Typically, the apparatus will be networked to one or more streaming devices and other hardware in order to broadcast or transmit the data service to the data display platforms. Alternatively, or additionally, the data service may be provided to selected data display platforms on request, for example over a telephone or other telecommunications network.

Multiple instances of the content instance builder may be provided, either simultaneously or as and when required, to build content data for different types of data display platform. Multiple instances of the application builder may similarly be provided. The content instance builder and application builder may, for example, take the form of particular processes or groups of processes executing under the control of an operating system.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying figures of which:

figure 1 is a schematic showing components of a typical interactive television system;

figure 2 shows a system for building one or more data services for broadcast to data display platforms;

figure 3 shows in more detail the application builder of figure 2;

figure 4 shows in more detail the content acquirer of figure 2;

figure 5 shows is more detail the content instance builder of figure 2;

figure 6 shows an example of a simple content data model;

figure 7 shows an example of content data configured to conform to the content data model of figure 6;

figure 8 shows the content data of figure 7 configured into an XML format; and

figure 9 shows an example of an application data model.

Referring now to figure 2, a preferred embodiment of the present invention provides a system for building one or more data services for broadcast to and use on one or more of a plurality of different types of data display platform as detailed above. A data service comprises one or more service software application components and one or more service data components.

Each data display platform type may provide a different platform software environment for execution of service software application components. For example, a platform type may be represented by an interactive television set top box running a particular version of a particular operating system such as OpenTV (trademark). Each software environment, in general, provides different facilities for execution of service software application components. Differences may exist, for example, in different language syntaxes accepted by software language interpreters, access to differing run time libraries, different physical or logical memory sizes, different television display resolutions, and different hardware data decoding and decryption facilities, to name but a few. Other variations in platform types, for example relating to language and geographical content requirements, and to data bundling and other network operator constraints, have been mentioned above.

In the system shown in figure 2, data service content is acquired from a content provider 102 by content acquirer 104 which configures the data service content to conform to a content data model 106. The content data model defines a general configuration of the data service content which is convenient for handling and storage by database server 108 to which it is passed by the content acquirer 104. The content data model 106 does not define a specific data structure as such, because the precise form of the data service content will vary from time to time as the data service content is updated. Rather, it defines a data type to which the particular instances of content data are to conform following reconfiguration by the content acquirer 104. The database server 108 stores the content data model configured data service content in database 110.

An application data model 112 defines a general configuration of the service data components to be broadcast in the data service for access and use by the service software application components executing on the data display platforms. The application data model 112 is of similar structure to the content data model 106 . The application data model 112 defines a generalised structure such as a tree of particular data types, preferably with an undefined numbers of branches of any one type providing flexibility.

An application builder 114 builds a particular instance of the service software application components intended for execution on a selected data display platform type. The build is based on a set of application libraries 116 which may contain application software fragments generic to all or a range of platform types and software fragments which are specific to particular platform types or families of platform types. A particular instance of the service software application components may take advantage, for example, of the particular hardware and software capabilities of the data display platform type for which it is built, and will need to be compiled or built in a code appropriate for the interpreter or other run time environment provided on the target platform type.

The application builder 114 also makes use of application data model 112, to define the interface required to access elements of the service data component which conform to the application data model.

A content instance builder 118 is provided with data service content from the database server 108 and from this data builds platform specific service data components for broadcast or transmission with associated platform specific service software application components. Conversion of the content data for a particular data display platform type is achieved using a set of content conversion rules 120 which also define how to transform data stored in a configuration conforming to the content data model 106 into a configuration conforming to the application data model 112.

The platform specific data components built by the content instance builder 118 are checked for consistency with the application data model 112, to ensure that the data will be readable by the corresponding software application components. The data components are then processed for broadcast along with the corresponding service software application components and broadcast to data display platforms via one or more streaming devices 121 and other appropriate hardware.

The process of building a data service can be broadly divided into two time frames. At service creation time the application and content data models and the content rewriting rules are defined. The service software application components for each target platform type are also built at this time. At service operating time service content data is acquired, stored and rendered by the content instance builder 118 into one or more service data components conforming to the application data model and compatible with the software application component or components.

The content acquisition process, from provision by the content provider 102 to storage in the database 110, is independent of the particular platform type for which a data service is subsequently to be built, and consequently may be reusable or common to a family or wider range of data display platform types. The data service content remains in a platform independent form until it reaches the content instance builder 118 where it is rendered into a platform specific form using the platform specific content conversion rules 120. This arrangement is of significant benefit where the data service content frequently changes. Furthermore, plural content instance builders 118 may be provided to permit building of platform specific service data components for a plurality of different platform types and platform content requirements, each instance builder receiving data in a common format from the database 110.

Similarly, plural instances of the application builder 114 may be provided to build instances of the software application components for a number of different platform types. Compatibility of each platform specific data component with the corresponding platform specific software application components is assured by the use of the common application data model 112 which may be explicitly used to cross check the data component or components. Separately adapted application data models may also be provided, if required, for each of two or more different platform types of families of platform types.

The described arrangement allows the physical or logical boundary between the content provider 12 and the service provider 10 to be positioned as desired so that more or less of the content acquisition and handling process is carried out by the content provider 12. A particular content provider 12 may wish to gather, verify, and store data service content under its own control and to then forward the content to the service provider for content instance building. Alternatively, a content provider 12 may wish to provide data service content in a raw state and to require the service provider to carry out all the required precessing steps.

The described functional units and processes may be carried out on any suitable configuration of computer hardware and software. For example, the entire data service building process could be carried out on a single computer workstation, or could be distributed over a number of different computers situated in different locations.

The content acquirer 104, the database server 108, the application builder 114 and the content instance builder 118 may conveniently be provided by a number of software processes running under one or more operating systems on one or more computers. The application data model, content data model, content conversion rules and content data in various configurations may conveniently be provided by a number of computer readable data files accessible by the relevant processes.

The system shown in figure 2 will now be described in more detail. Referring first to figure 3, the application builder 114 is illustrated as a number of processes (indicated by ellipses) and a number of resources such as software libraries or similar data components (indicated by rectangles). Central to the application builder 114 is a platform specific compilation process 123 which, in general, is specific to a particular data display platform type or family of platform types. Typically, the compiler software needed to build service software application components to execute on a particular platform type will be provided by the proprietor of the operating system to be used on that platform. Such compilers generally accept source code in a commonly used language such as C or a variant thereof, and compile the source code into a proprietary language or byte code to be interpreted or executed by the operating system running on the target platform.

A number of resources are required to complete the compilation 123. Native platform libraries 122 will generally be provided as part of the software development kit which includes the compiler software. The native platform libraries 122 provide software fragments specific to particular platform types for carrying out general purpose functions such as reading data from the broadcast stream, writing data to the platform display, receiving input from user controls such as a remote control hand set and so on. Third party libraries 124 may be provided which provide additional software resources not provided by the native platform libraries 122. These additional resources might include, for example, software resources for providing communication over a return data path from the data display platform to a service provider.

A platform specific user interface object library 126 provides software for implementing particular user interface objects such as buttons, menus and text boxes for display and for interaction with the user of any particular platform type.

The application data model 112 is rendered into a compilable source code content retrieval library 130 by a retrieval library generator 132. The content retrieval library provides type casting of the data structures to be loaded into the data display platform from the broadcast service data components at run time, and also provides executable resources for initiating retrieval and storage in platform memory of individual elements of the service data components as and when required. Further general data and memory management resources may also be incorporated in the content retrieval library 130.

The application data model 112 is a resource that describes the structure of the content of the service data components that the software application components expect to receive at run time from the data modules of the broadcast stream. Essentially, the application data model 112 takes the form of a single tree of data type definitions. The tree is built on a single root structure and contains simple objects such as numbers, text strings and graphics objects, and aggregate objects that may in turn contain other objects, to form the structure of the tree. Structures defined in the tree are generally flexible enough to accommodate, for example, the absence of or a variable number of a particular substructure, for example through the use of arrays of flexible size.

Small objects such as integers may be type cast as themselves. However, in order to facilitate distribution of objects between different delivery units or modules in the broadcast data service, and to be able to fold the tree to refer to an identical object or subtree from different points in the same tree, larger objects are represented by references or handles. Such a reference may, for example, refer to an object in the tree which has not yet been loaded into the memory of the data display platform. The content retrieval library 130 provides resources for creating such handles and loading associated data items and for freeing up memory associated with handled data items when no longer required.

A platform specific application code resource 128 contains the software resources for controlling the layout of the display gadgets, the interaction between the gadgets and the data, and the general logical structure of the software application. The application code resource, while specific to a particular platform type or family of platform types, may none the less be partly generated from more generic user interface and navigation code specifications 134, although various other ways of generating this resource could clearly be used.

The compiled service application software component or components 136 are packaged into an appropriate serialised and modularised form by the compilation process 123, or by a separate serialisation and modularisation process. The application software for an interactive television broadcast will generally be packaged into one or more modules for broadcast to the data display platforms. A small stub module, containing application software for initialising the application may be broadcast more frequently than one or several more substantial modules which are loaded by the software in the stub module.

Referring now to figure 4, the content acquirer 104 is illustrated as a number of processes (indicated by ellipses) and a number of resources such as raw data and software libraries (indicated by rectangles). Central to the content acquirer 104 is an acquire content process 152. This process receives or retrieves raw service content data 150 and renders it into a configuration conforming to the content data model 106. The rendered data is then stored in content database 110. The raw content data 150 may be, for example, the text and images of news stories or weather bulletins, the names and times for television schedule listings and so on. The raw content data 150 may be actively retrieved by the acquire content process 152 or a related process over a network connection, or may be made locally available to the acquire content process 152, for example by a remote content provider 12.

The acquire content process 152 operates according to executable code built by the acquire content compilation process 154. This compilation process combines a number of software resources including specific import code 156, an import database library 158 and a set of content data structures 160 for defining how content data is to be stored prior to commitment to the database 110. The content data structures and/or other code elements generated using the content data model also effectively define an interface between a particular content data model 106 and the import database library 158, and enable type checking in order to reduce the incidence of errors in the acquire content process 152. The importation and restructuring of the raw content data is made more straightforward and safe by these automatically generated interface and data structures.

The import database library 158 provides a number of general purpose data handling and conversion routines, and provides resources for casting data into the form required and for passing the data so cast to the content database 110.

The content data structures 160 are generated by a content data structure generator process 162 from the content data model 106. Conveniently, the generator process 162 may build the structures as a set of C++ classes. The content data model is written as one or more trees of C-language like type structures, the structures being chosen to provide a reasonably logical layout of the raw data, to facilitate storage in the content database 110 and to make conversion to other configurations, and in particular to a variety of different platform specific service data components convenient. Clearly, various coding structures other than C-language type schemes could equally be used.

Raw content data 150 is acquired, processed and stored in the content database 110 on a continual or periodic basis, for example as such data becomes available during service operation time.

Referring now to figure 5, the content instance builder 118 is illustrated as a number of processes (indicated by ellipses) and a number of resources such as content data and software libraries (indicated by rectangles). A content tree generator 180 receives or extracts content data from the content database 110, creates a content data tree 182 corresponding to the required data and embeds the actual content data into the content tree 182. The Extendable Markup Language (XML) provides a particularly suitable syntax and structure for the content tree, although formats other than XML could clearly be used, and structures other than trees could be used. However, XML is conveniently generalised and suitable for transformation and further processing of the content data as detailed below.

The content tree 182 may conveniently be passed to a markup language generator 184 to build pull-type content 186 for use in a pull-type data service context, for example provided by a hypertext transport protocol server or a wireless applications protocol server. In such a case, the markup language generator 184 would build an HTML (hypertext markup language) file from the content tree. Because XML is a generalised markup language, the reconfiguration of the content tree from XML to a variety of other markup languages such as HTML is relatively straightforward.

The content tree 182 is passed to a rewriting process 188 which processes the data from a data-oriented representation into a presentation oriented platform specific content tree 190 configured for use with a platform specific instance of the service software application components. The rewriting process 188 carries out a number of selection and conversion processes. These may include:

selection of content data items that are applicable to a particular data display platform type, eg Swedish text for a Swedish network operator;

collation of content data items for presentation on a particular platform type, for example so that content data items for presentation together are located in the same region of the platform specific content tree 190;

conversion of content data items for presentation on a particular platform type, typically involving a change to the original format, for example when breaking up text into lines and pages with reference to available font metrics and presentation space boundaries, converting graphic images to a format easily useable by a particular platform type and so on;

marking resources with platform type dependent format requirements for later conversion, if required, for example in respect of levels of data transiency, to facilitate division of the data into broadcast modules; and creating a navigation and presentation oriented markup of the content data.

The rewriting process 188 also carries out operations such as folding of the content tree so that multiple nodes can refer to a common data item. Content tree folding is an optimisation process necessary to avoid duplicate content data being broadcast.

From a transformational point of view, the rewriting process 188 rewrites the content tree 182, which conforms to the content data model 106, to the platform specific content tree 190, which conforms to the application data model 112. The rewriting is carried out by reference to a sequence of rewrite, or content conversion rules 120, each referring to a top-level variable declaration having a corresponding type declaration in the application data model 112. The rewriting process also makes use of a library of conversion functions 192, for example for converting between graphics formats, and a tree manipulation library 194. Each rewrite rule can be regarded as a transformation of a subtree conforming to a type in the content data model into a subtree conforming to a type in the application data model.

The rewriting process may be generalised further, applying the rewrite stage several times for content destined for the same target platform.

The rewrite rules are expressed using a syntax allowing reference to the roots of subtrees in the input content tree 182. The syntax provides for application of conversion functions to these roots in order to build a new platform specific content tree 190 which is type checked against the application data model. The conversion functions may refer directly to the actual content data, so that computations may be performed on the actual input. In this way the structure of the output platform specific content tree

190 may be dependent on the actual input content.

The application data model 112 is used by the rewriting process to ensure that the platform specific content tree 190 conforms to the application data model, and that it will thus be correctly readable by the corresponding service software application components. This cross checking of data compatibility is particularly important to ensure reliability in using the same underlying content data for broadcast to a number of different platform types.

More particularly, the application data model 112 ensures that the conversion of an instance of a content tree conforming to the content data model is type-safe. If the content tree is successfully converted by the rewriting process 188 then it is more likely that it can be used by the software application at run time on the target data display platform type without any type errors arising, since the same application data model is used in constructing the data retrieval elements of the software application.

The platform specific content tree 190 is passed to a serialisation and modularisation process 192 which takes account of the structure of the platform specific content tree 190 and preferences for data grouping expressed therein to assemble the data into broadcast ready modules for broadcast as one or more service data components, along with one or more service software application components. The broadcast ready modules are then routed via one or more streaming devices 121 to the target platform.

A very simple example of a content data model is shown in figure 6. The example relates to a product catalogue in two different presentation languages (Danish and English) for three different product groups (mobile telephones, jeans and coffee). A record is defined for each product group, and each group contains some information relevant to that particular group. For example, the product description for a pair of jeans consists of a price, a size and a colour. An object of type "Product" can be one of the types "Phone", "Jeans" or "Coffee". This is expressed by a sum type. Finally, there is an object called "Catalogue" which is a vector of products.

Figure 7 shows an example of a catalogue content data tree, configured to conform to the content data model of figure 6, containing one type of telephone, one type of African Coffee and one type of South American coffee.

Figure 8 shows the content tree of figure 7 marked up as an XML tree.

An example application data model is shown in figure 9.

CLAIMS

1. A method of building a platform specific data service for transmission to and operation on data display platforms of a type selected from a plurality of data display platform types, the platform of selected type having data service requirements different from those of other ones of said plurality of platform types,

the platform specific data service comprising a software application component and a data component,

the software application component being operable to cause a platform of said selected type to display elements of said data component in response to commands generated by said user,

the method comprising the steps of:

providing an application data model;

using said application data model to build said software application component to be operable to access data configured to conform to said application data model;

providing service content data in a first configuration generic to said plurality of platform types;

converting said content data from said first configuration to a second configuration conforming to said application data model; and

using said content data in said second configuration to provide said data component.

2. The method of claim 1 further comprising the step of using the application data model to check that said content data in said second configuration is compatible with said software application.

3. The method of either of claims 1 or 2 wherein said second configuration is specific to said selected

data display platform type.

4. The method of any preceding claim wherein said content data in said first configuration comprises content data in a tree configuration.

5. The method of claim 4 wherein said content data in said first configuration comprises content data in an XML (extendible markup language) configuration.

6. The method of either of claims 4 or 5 wherein the step of converting comprises the step of folding said tree configuration to eliminate duplicate instances of elements of said content data.

7. The method of any preceding claim wherein said step of converting includes the steps of:

providing a set of content conversion rules defining the required transformations to render content data in said first configuration into said second configuration; and

applying said set of content conversion rules to said content data in said first configuration.

8. The method of any preceding claim wherein the application data model is generic to said plurality of data display platform types.

9. The method of any preceding claim wherein said plurality of platform types comprises one or more interactive television set top box configurations.

10. The method of any preceding claim further comprising the steps of combining said software application component and said data component in a transmission to a plurality of said data display platforms of said selected type.

11. A transmission comprising a software application component and a data component built according to the method of any of claims 1 to 10.

12. Apparatus operable to carry out the method steps of any of claims 1 to 10.

13. A computer readable medium comprising computer program elements operable to carry out the method steps of any of claims 1 to 10 when executed on one or more computer systems.

14. A method of building a data component of a platform specific data service for transmission to and operation on data display platforms of a type selected from a plurality of data display platform types, the platform of selected type having data service requirements different from those of the other ones of said plurality of platform types,

the platform specific data service comprising a software application component and said data component,

the software application component being operable to cause a data display platform of said chosen type to display elements of said data component in response to commands generated by said user,

the method comprising the steps of:

providing a content data model generic to said plurality of platform types;

providing an application data model representative of a configuration of the data component readable by the software component;

receiving service content data in a raw configuration;

rendering said service content data from said raw configuration into a content data model configuration rendering said service content data from said content data model configuration into an application data model configuration conforming to said application data model; and

using said service content data in said application data model configuration to provide said data component.

15. The method claim 14 further comprising the step of storing in a database said service content data in said content data model configuration.

16. The method of either of claims 14 or 15 wherein said content data model defines a tree structure.

17. The method of any of claims 14 to 16 wherein the application data model defines a tree structure.

18. The method of any of claims 14 to 17 wherein the step of rendering said service content data from said content data model configuration to said application data model configuration comprises the steps of:

rendering said content data from said content data model configuration to an intermediate configuration; and

rendering said content data from said intermediate configuration to said application data model configuration.

19. The method of claim 18 wherein said intermediate data model configuration conforms to the Extendable Markup Language (XML).

20. The method of either of claims 18 or 19 further comprising the steps of:

rendering said content data in said intermediate configuration into a pull-type configuration; and making the content data in said pull-type configuration available on a server to third parties.

21. The method of claim 20 wherein said pull-type configuration conforms to a version of the Hypertext Markup Language (HTML) and said server is an HTML server connected to the Internet.

22. The method of any of claims 14 to 21 wherein said application data model is generic to said plurality of platform types.

23. The method of any of claims 14 to 22 wherein said plurality of platform types comprises one or more interactive television set top box configurations.

24. The method of any of claims 14 to 23 further comprising the step of combining said software application component and said data component in a transmission to a plurality of said data display platforms of said selected type.

25. A transmission comprising a software application component and a data component built according to the method of any of claims 14 to 24.

26. Apparatus operable to carry out the method steps of any of claims 14 to 24.

27. A computer readable medium comprising computer program elements operable to carry out the method steps of any of claims 14 to 24 when executed on a computer system.

28. Apparatus for building a platform specific data service for transmission to and operation on data display platforms of a type selected from a plurality

of data display platform types, the platform of selected type having data service requirements different from those of other ones of said plurality of platform types, the platform specific data service comprising a data component,

the apparatus comprising:

a content acquirer operable to receive content data in a raw form and to reconfigure said raw form content data to conform to a content data model; and

a content instance builder operable to receive said content data model conformed content data and to reconfigure said content data to conform to an application data model to thereby provide said data component.

29. Apparatus as claimed in claim 28, further comprising a database server and a database,

the database server being operable to accept said content data conformed to said content data model and to store said accepted content data in said database,

the database server being further operable to retrieve said stored content data from said database and to forward said retrieved data to said content instance builder.

30. Apparatus as claimed in either of claims 28 or 29 wherein said platform specific data service further comprises a software application component operable to cause a data display platform of the selected type to display elements of said data component,

the apparatus further comprising an application builder adapted to receive said application data model and to build said software application so as to be operable to utilise said data component.

31. Apparatus as claimed in claim 30 comprising a plurality of said application builders, each

application builder being operable to build a software application for use on a different selected type of said plurality of types of data display platform.

32. Apparatus as claimed in any of claims 28 to 31, comprising a plurality of said content instance builders receiving said content data conformed to said content data model, each content instance builder being operable to provide a data component for use on a different selected type of said plurality of types of data display platform.

33. A method of building a platform specific data service substantially as herein described with reference to figures 2 to 9 of the accompanying drawings.

34. A method of building a data component of a platform specific data service substantially as herein described with reference to figures 2 to 9 of the accompanying drawings.

35. Apparatus for building a platform specific data service substantially as herein described with reference to figures 2 to 9 of the accompanying drawings.

36. A method for building a platform specific data service, comprising:

assembling a platform-specific software application in accordance with an application data model using application software fragments stored in at least one application library; and

converting generic content data to platform-specific content data using content conversion rules that define how to transform the generic content data into a format that conforms with the application data model.

37. The method of claim 1, wherein converting generic content data to platform-specific data includes:

receiving raw content data;

generating a content data structure using a content data model;

embedding the raw data into the content data structure.

38. The method of claim 1, wherein converting generic content data into platform-specific data includes:

generating a generic content tree that conforms to the data content model;

retrieving a content conversion rule; and

rewriting at least part of the generic content tree in accordance with the content conversion rule.

39. The method of claim 2, wherein converting generic content data into platform-specific data includes:

generating a generic content tree that conforms to the data content model;

retrieving a content conversion rule; and

rewriting at least part of the content tree in accordance with the content conversion rule.

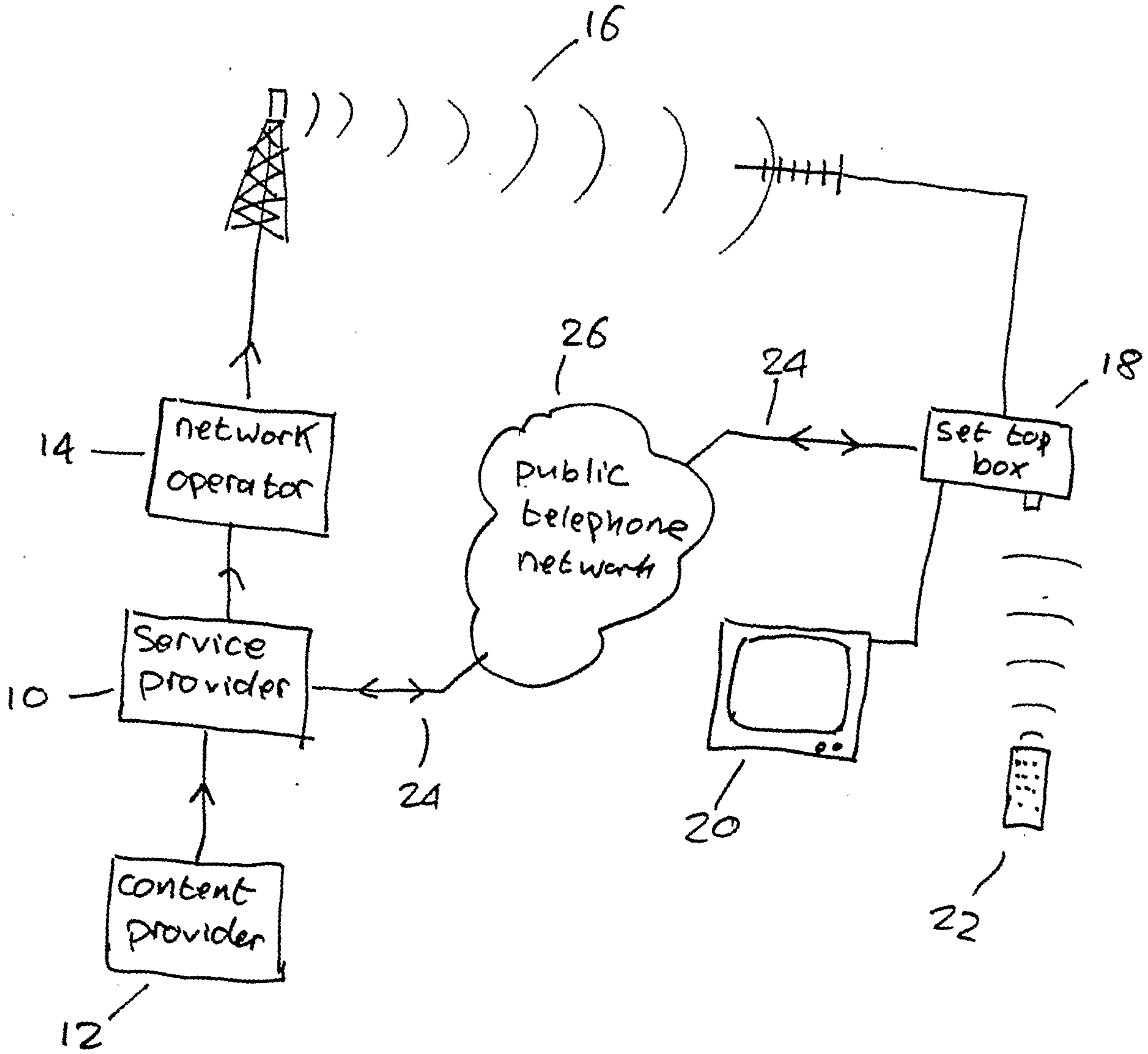
16
26
24
18
14
network operator
public telephone network
set top box
Service provider
10
24
20
22
content provider
12

Figure 1

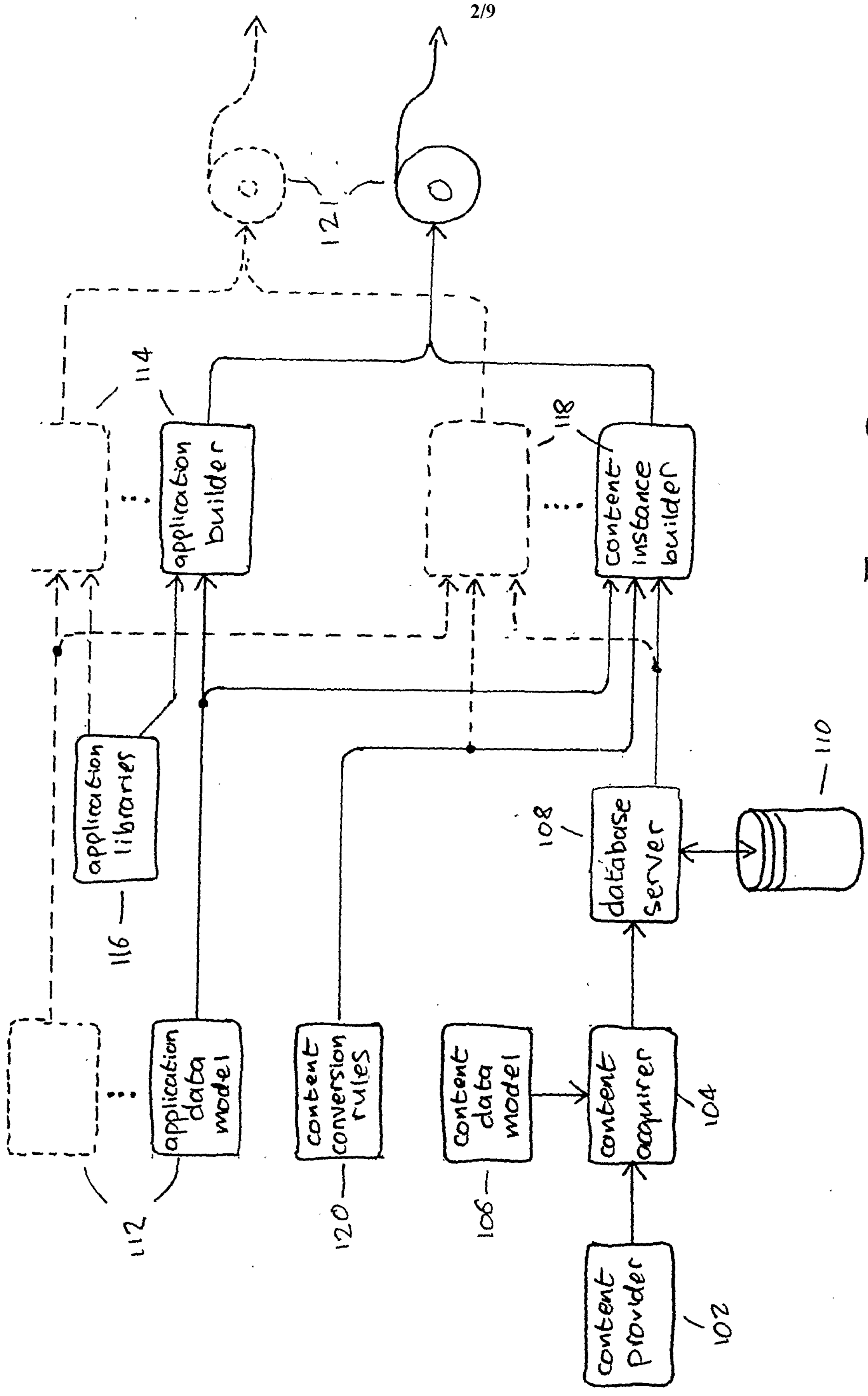
content provider
102
content acquirer
104
content data model
106
database server
108
110
content conversion rules
120
application data model
112
application libraries
116
application builder
114
content instance builder
118
121

Figure 2.

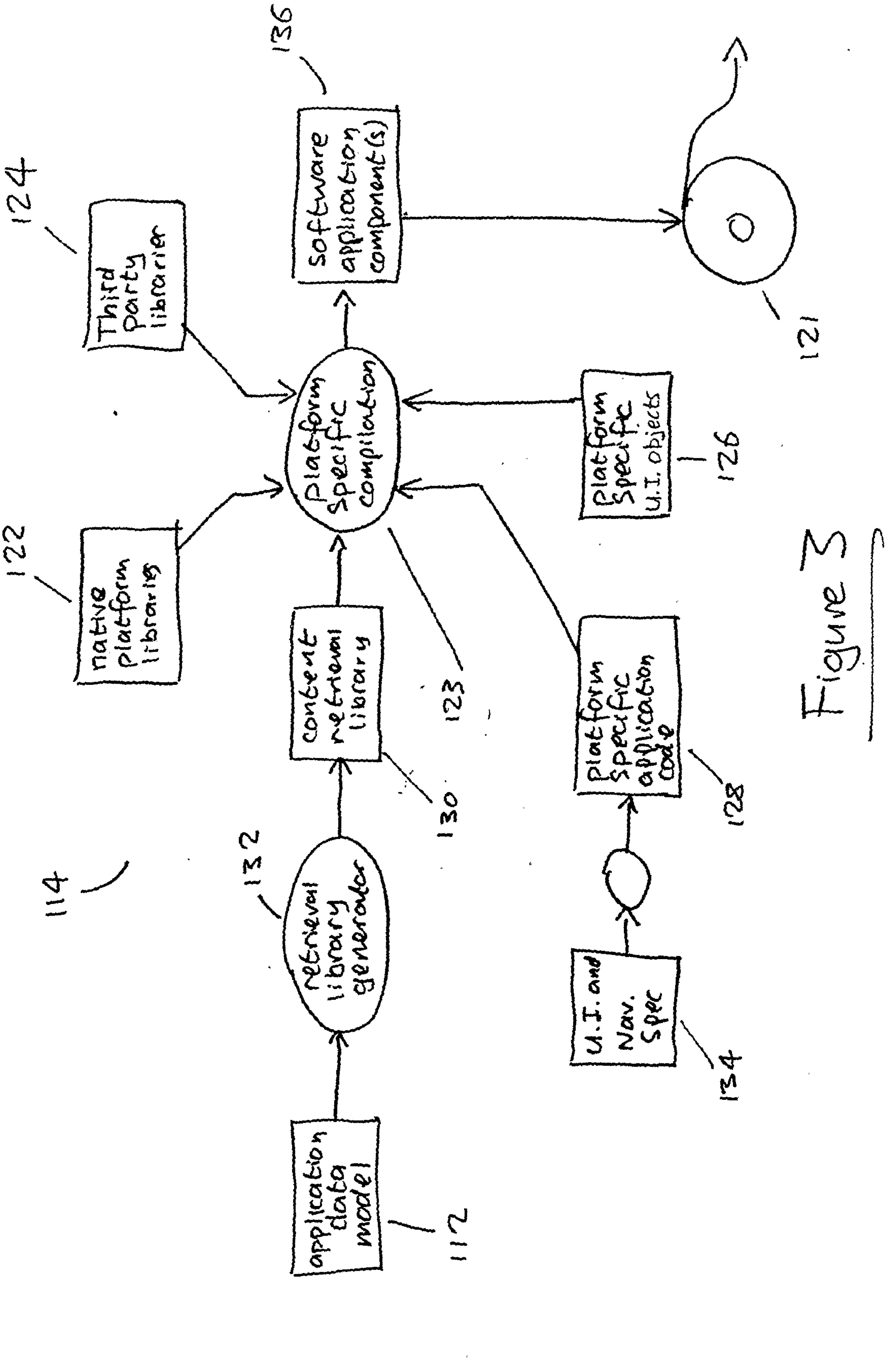
114
112
application data model
132
retrieval library generator
content retrieval library
130
122
native platform librarirs
123
platform specific compilation
124
Third party librarier
136
software application component(s)
121
126
platform specific U.I. objects
platform specific application code
128
U.I. and Nav. Spec
134

Figure 3

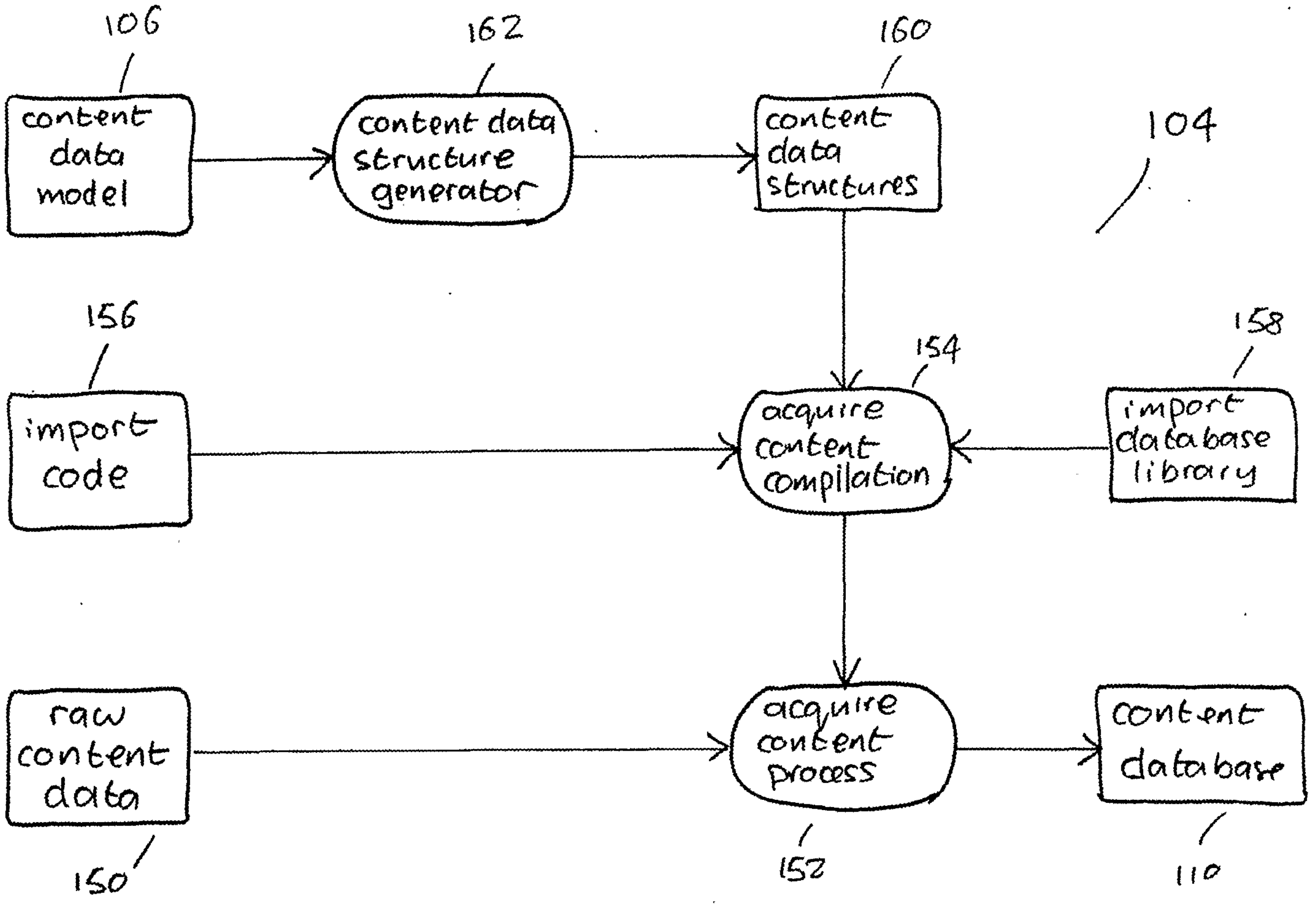
106
content data model
162
content data structure generator
160
content data structures
104
156
import code
154
acquire content compilation
158
import database library
raw content data
150
acquire content process
152
content database
110

Figure 4.

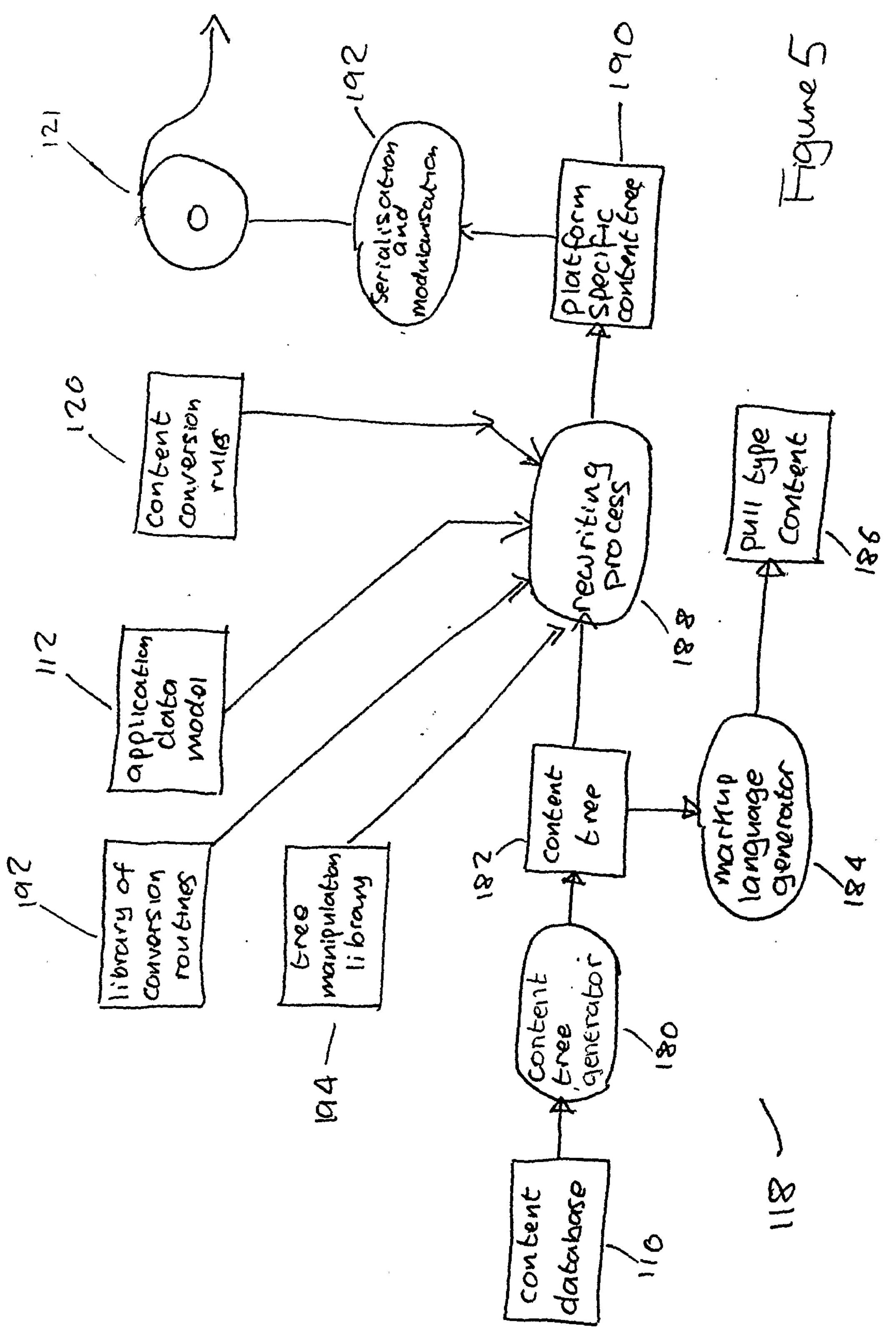
content database
110
content tree generator
180
content tree
182
markup language generator
184
pull type content
186
rewriting process
188
platform specific content tree
190
serialisation and modularisation
192
O
121
library of conversion routines
192
tree manipulation library
194
application data model
112
content conversion rules
120
118

Figure 5

```
data_model_name = test;
data_model_version = 1.0;

struct PresentationString
{
    danish : String;
    english : String;
};

struct Phone
{
    price : Int;
    weight : Int;
};

struct Jeans
{
    price : Int;
    size : Int;
    colour : PresentationString;
};

struct Coffee
{
    price : Int;
    country : PresentationString;
};

type Product = Phone + Jeans + Coffee;

type PList = vector of Product;

catalogue : PList;
```

Figure 6

```
catalogue =
Plist
{
    Phone
    {
        price = 42,
        weight = 10
    },

    Jeans
    {
        price = 50,
        size = 12,
        colour = PresentationString
                 {
                     danish = "sort",
                     english = "black"
                 }
    },

    Coffee
    {
        price = 12,
        country = PresentationString
                  {
                      danish = "Afrika",
                      english = "Africa"
                  }
    },

    Coffee
    {
        price = 24,
        country = PresentationString
                  {
                      danish = "Sydamerika",
                      english = "South America"
                  }
    },
}
```

Figure 7

```
<test>
    <catalogue>
    <Plist>

        <Phone>
            <price><Int>42</Int></price>
            <weight><Int>10</Int></weight>
        </Phone>

        <Jeans>
            <price><Int>50</Int></price>
            <size><Int>12</Int></size>
            <colour>
            <PresentationString>
                <danish><String>"sort"</String></danish>
                <english><String>"black"</String></english>
            </PresentationString>
            </colour>
        </Jeans>

        <Coffee>
            <price><Int>12</Int></price>
            <country>
            <PresentationString>
                <danish><String>"Afrika"</String></danish>
                <english><String>"Africa"</String></english>
            </PresentationString>
        </Coffee>

        <Coffee>
            <price><Int>24</Int></price>
            <country>
            <PresentationString>
                <danish><String>"Sydamerika"</String></danish>
                <english><String>"South America"</String></english>
            </PresentationString>
        </Coffee>

    </Plist>
    </catalogue>
</test>
```

Figure 8

```
data_model_name = test;
data_model_version = 1.0;

type IntList = vector of Int;
type StringList = vector of String;
type BlobList = vector of Blob;
type PageList = vector of StringList;

struct DifferentPrices
{
    francs: Int;
    kroner: Int;
    sterling: Int;
    usdollars: Int;
};

struct NewCoffee1
{
    pricelist : DifferentPrices;
    newcountry : String;
};

type NewCoffeeList1 = vector of NewCoffee1;

struct NewCoffee2
{
    price: Int;
    newcountry: String;
};

type NewCoffeeList2 = vector of NewCoffee2;

coffeelist1 : NewCoffeeList1;
coffeelist2 : NewCoffeeList2;

intlist: IntList;
stringlist: StringList;
bloblist: BlobList;

textpages: PageList;
```

Figure 9.