Flexible Component Architecture for Information WEB Portals

Łukasz Dutka[1,2] and Jacek Kitowski[1,2]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Cracow, Poland
[2] ACC CYFRONET AGH, ul. Nawojki 11, 30-950 Krakow, Poland
{dutka,kito}@uci.agh.edu.pl

Abstract. In this paper a proposition of Component-Expert Architecture for designing information Web portals is presented. Taking advantage of the expert system approach, it introduces a sophisticated method of components management, allowing to reduce complication of software used to build the portals. Experience of practical implementation is reported.

1 Introduction

Portals serve a wide range of uses in the enterprise – from internal self-service and knowledge management portals to external business-to-business catalogs and e-commerce platforms. They are about aggregating information and content (both static and dynamic), integrating functionality from other applications, and providing unifying services such as user access control, personalization and a common user interface [1].

In the nearest future the portals would be one of the hottest topics. The American Library Association has published the most important Information Technology trends [3] and three of them are very closely related to the portals. One of these three issues, "User Centered Design", is very close to the topic of this paper.

Traditional portal applications implement popular PHP, ASP, J2EE and JSP technologies. These technologies are difficult to apply directly for large-scale Web information systems, since they are page oriented. Contrary, the information portals are driven by information and by the functional layer (a user interface). Hence, the architecture should be information and user oriented. This has lead to development of portlets (with two main initiatives, Java Specification Request 168 – JSR168 and Web Services for Remote Portals – WSRP), which are visible active components that end users see within their portal pages [1,2]. Each portlet can be built independently of the others, once a portlet is developed and tested, it is stored into the container (the portlet catalog). By browsing the container, the end user can then select a portlet and place it into one of their own portal pages. Technically, a portlet is a piece of code, that runs on a portal server and provides content to be embedded into portal pages [2].

User selection of the portlets from the container could be supported by a kind of expert system, making the process both more efficient and flexible. For this

© Springer-Verlag Berlin Heidelberg 2003

purpose the Component-Expert Architecture (CEA) [4,5] is proposed, which is based on the popular component approach and makes use of a rule-based expert system (e.g. [6,7,8]) for portlets selection.

The component programming model, in which the components are versionable, programmable and fast making the software easier to write and reuse, provides wide choice in services, tools, languages and applications [9,10,11,12]. Many models and component architectures have been developed, see for example [13,14,15,16,17,18,19]. The proposed automatic selection of components makes the model more efficient.

The paper describes the Component-Expert Architecture and its application for designing Web portals with an example of practical implementation.

2 Basics of Component-Expert Architecture

A general view of the architecture is shown in Fig. 1. In CEA the portal is represented by a set of simple components, which are managed and selected (from a given collection) by the expert system. The components serve as portlets, although neither JSR168 nor WSRP standards are maintained. During the selection, the expert system takes into account many factors, as a kind of reader facilities, reader preferences, features of the presented information, etc. The programmers do not manage components, they just specify in detail the purpose of the components and register them in the container. Both, the components (which are intended to present the information) and the user interface, are easily connected with the information [4,5].

A CEA component (see Fig. 1) consists of the header and the component code.

The header is defined by a component type and by an attributes list (Fig. 1). It describes the general purpose of the component (defined by its type, TIDx) and its specialization (defined by a list of attributes, SPECy). Examples of the types for Web portal problems are: MainMenu for components that attend to horizontal menu on pages, PresentationInShortForm for components presenting particular information in a list form or FullPresentation for components presenting the whole information about a particular data object.

The specialization represents the detailed purpose for which a particular component is the most appropriate. The specialization is used by the rule-based expert system for selection of one component of the required type from the component collection according to the context of component invocation, defining the requested functionality (this context is also called 'call-environment' in the paper).

The logical component structure (presented in Fig. 1) can be realized by almost all programming languages including script languages, being very popular in the Web domain. This functionality is also easy to developing in the environments that make use of commercial component standards, like COM [12], NET [15] or JavaBeans [16,17], and commercial UML notation (see Fig. 2).

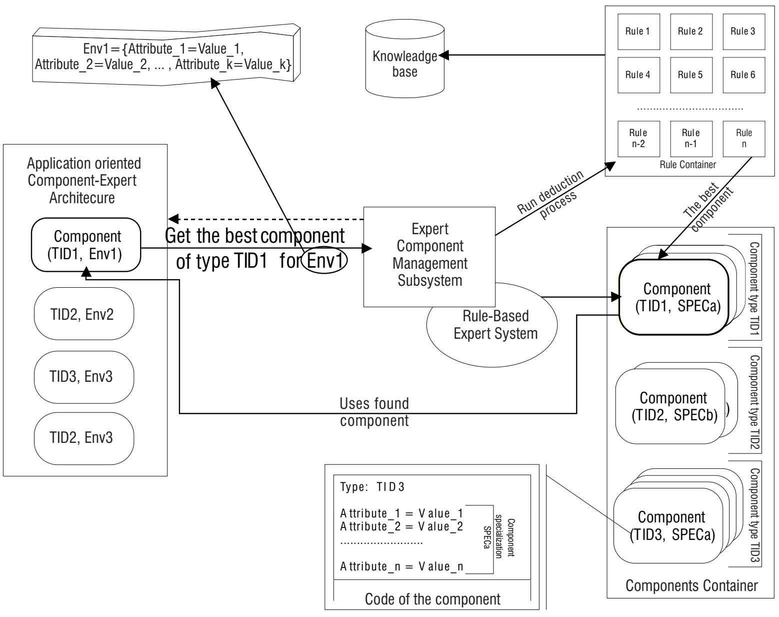

Fig. 1. Logical scheme of CEA

The functionality of the header can be realized by introducing a dedicated interface – ISpecification, which returns the type of the component and its specialization in a form of the list of attributes and their values.

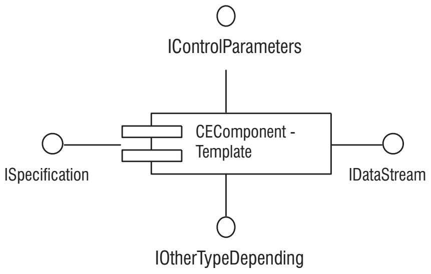

Fig. 2. UML Model of component for CEA

The component input stream supplies control parameters for controlling the component. The output parameters stream fetches low-volume results of the execution and can be used by other components. Both are defined as operations of IControlParameters interface. The data stream is intended to pass large amount of data – usually HTML/XML or a binary stream, using IDataStream interface. IOtherComponentDepending interface depends on the component type.

The code is the central part of the component for data processing and generates the output stream. The functionality of the streams can be obtained by introducing other interfaces. Due to generality of CEA description, just the logical component model is used.

Taking MainMenu component as a further example, there can be a bunch of components of this particular type, which are suited for different client browsers (e.g., one for HTML 4.0 client, another for the old version of Netscape Navigator, etc.), for different portal sections (e.g., for homepage, for sport or news sections) or for different languages. Each particular component performs a similar task, but for a different context. There can be universal components, which can work for almost every context or very specialized components for particular applications, e.g., a component specialized just for one section of a portal, one particular browser and one language of the interface.

The components are selected by the expert system based on the context of component invocation, i.e., call-environment, ENVx. The call-environment is composed of a list of attributes and their values (see Fig. 1). It defines requirements or requests, which should be fulfilled by a selected component. Such a message composed of the component type and the requirements is passed to the Expert Component Management Subsystem, which prepares the call to the expert system (see Fig. 1).

The expert system obtains a pair: the type of the searched component and the call-environment. Its goal is to find (for the given type) a component with such specialization, that is most suitable for the call-environment. In the proposed model, the specialization and call-environment are the lists of attributes with values. The specialization defines the scope of use of each component, while the call-environment describes requirements requested for the selected component.

The rules of the expert system are used in typical form, i.e., `if <log-expr> then <action> else <action>`. They are constructed by the programmer during the development stage of the system. The external Knowledge Base (cf. Fig. 1) represents the source of information presented in portal pages. Its supplementary task is to support the system with the additional knowledge useful for making the system decisions more accurate and the whole solution more flexible. For example, it could be information on preferences of a user (like browsers capabilities or the language).

The rules have to be prepared in a way that selection of the best component for a given context is made unambiguously. If not fulfilled, a general purpose component is taken arbitrary and further optimization is needed by rules refinement. Components should rather be constructed as independent parts of the portal; if necessary they could interact.

The operation of the CEA system is summarized as follows (see Fig. 1):

1. The program requires service the TIDx type component for the ENVy call-environment. It requests from the Expert Component Management Subsystem a handle to the particular component, most suitable for ENVy.
2. The Expert Component Management Subsystem parses obtained information and using the Expert System it searches the best component for ENVy.

3. The Expert System gets information about all TIDx type components from the Components Container.
4. Based on the rule-based knowledge of the expert system and on the external Knowledge Base, as well as on the specialization of components of the required type, the Expert System takes a decision, which component is most suitable for ENVy.
5. If the Expert System finds the appropriate component then the Expert Component Management Subsystem returns a handle to the component, otherwise an error message is returned.
6. The program exploits the component referencing the handle.

By analysing log files the system is able to keep the track of deduction.

3 Application of Component-Expert Architecture for Information Web Portals

The CEA can be directly implemented for the Web portals. The Web services are provided by the WWW servers, which assume that the services are divided into pages. Each page is identified by URL, e.g., `http://www.domainname.com/pagename?attr1=val1&attr2=val2`. The HTTP page name could be considered as a type of component; the list of parameters could be identified with the call-environment. The HTTP pages return the HTML stream, which is the data stream for the component model. The URL can include also control parameters, which are not taken into account during the component selection process, but are passed to the selected component directly using the input parameter stream. In the presented implementation, these attributes are distinguished using some special prefix, i.e., `@` in our case. The call of `http://domainname/page?attr1=val1& attr2=val2&@attr3=val3` defines a request for the best `page` type component for the call-environment determined by `attr1=val1` and `attr2=val2`. This call-environment can be expanded by parameters passed in the HTTP request but not mentioned in the URL notation (e.g., type of browser, operating system, screen size, etc). Additionally, the call-environment can further be extended by information stored in cookies or in sessions identified by cookies. In practice, all methods of call-environment expansion are used.

Because during the run time the components may call other components, it is easy to divide a complicated portal structure into a set of small independent components, which are invoked hierarchically using CEA technique.

An important issue connected with the information portals should be a selection of components depending on information, which they are presenting. Assuming that the portal consists of several panels (realized by components) and taking, say, 'LeftPanel' as further example, the detailed kind of information which will be presented using this panel is only hardly known during development stage. It may be for example, a list of news, a weather forecast or a kind of marketing information. Different client environments could be used (e.g., PDA, PC or cellular phones) and different information repositories (e.g., RDBMS, OODBS, LDAP, ISAM, XML databases) implemented. The selection of the way of information presentation is the problem for which CEA comes into operation.

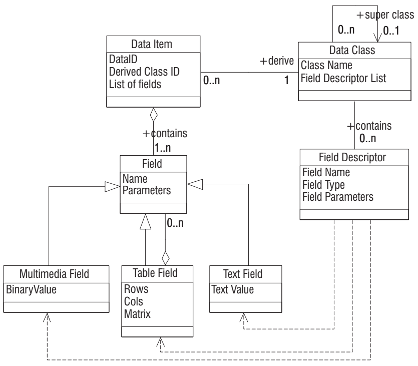

Fig. 3. UML Model of data

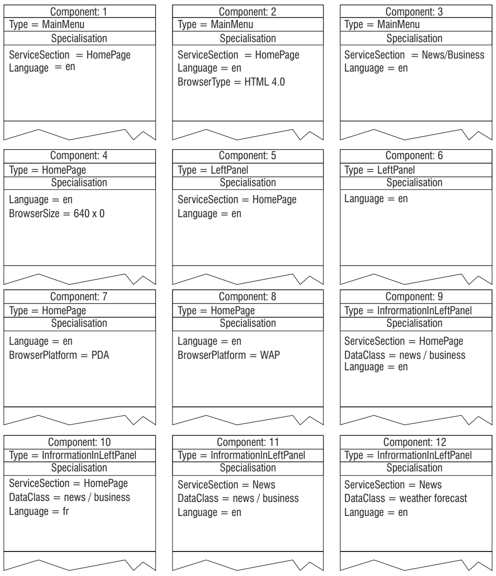

Fig. 4. Sample set of components

In Fig. 3 a simplified information structure for the portals is presented [8]. Each data item has a globally unique identifier – DataID and is derived from a data class. The data class describes the internal structure of each derived data item, which fields are included in the data item. Each data class may have one super-class and many sub-classes. The super-class concept is intended to group data classes in a hierarchical tree, which allow to organize information in a structural form. Each data item based on the data class consists of the fields, which contain appropriate values.

The languages describing the call-environment and component specialization can be defined in different ways, but the list of attributes is sufficient in most cases. In the presented study, the following attributes have been adopted (cf. Fig. 4):

– `ServiceSection` – describes the part of the portal,
– `Language` – the language preferred by the reader,
– `DataID` – identifier of the particular data object,
– `DataClass` – describes a class of data objects (e.g. news, news/business or weather forecast),
– `SessionID` – identifier of the user session,
– `ReaderID` – identifier of the user,

– `BrowserVendor` – vendor of the browser (e.g. IE or Netscape),
– `BrowserModel` – particular browser model and/or browser category,
– `BrowserPlatform` – kind of the platform (e.g. PDA, PC or WAP),
– `BrowserSize` – resolution of a user screen.

`ReaderID` can be implemented for defining user preferences, making use of `BrowserVendor`, `BrowserModel`, `BrowserPlatform`, `BrowserSize`, `Language` and external Knowledge Base.

Assume, that for the sample set of components in Fig. 4 and for the `TID = HomePage`, application requires the best component for the call-environment `Env = ServiceSection = Homepage; ReaderID = 546AF; BrowserVendor = IE; BrowserModel = IE 5.01.1734; BrowserPlatform = PC; BrowserSize = 900 x 600`. In such a case the deduction process is the following:

1. Expert Component Management Subsystem searches for components with type `TID = HomePage` included in Components Container, i.e., components 4, 7 and 8 are considered.
2. Since components 7 and 8 are specialized for other `BrowserPlatform`, they are neglected.
3. Component 4 is rather a general one, however it requires 640 pixel size (horizontally) of `BrowserSize` (with any vertical resolution). In the current context this requirements is fulfilled because the user's browser is size 900 pixels.
   Component 4 is intended for the English language (`Language = en`), thus to decide if this component is a proper one, the expert system has to obtain the current user preferences from external Knowledge Base. If the user preferred language is `en` then component 4 fits the user requirements and should be selected as the best one. Otherwise, component 4 is not quite proper, but no other choice exists. The rules have to decide whether the `Language` is the critical requirement. In most cases using the component in another language is a better solution than returning the error, but it strongly depends on the case.

4 Current Experience and Implementation

The practical MS Windows implementation of CEA is presented in Fig. 5, with HTTP Microsoft IIS and the ASP technology exploited. About 50 rules were implemented. MS SQL Server was used for the component container. It kept the source of scripts, which was compiled into JScript language. The components were written using a special preprocessing language, allowing to specify types of components and their specializations. However, the main semantics came from JScript. The preprocessor of this language parsed the code and converted it into pure JScript. After conversion the code was put into the database, called the Components Container in Fig. 5, in which the Expert Subsystem was integrated. The language offered other features, like multilingual support and special constructions allowing to include large blocks of HTML code easily.

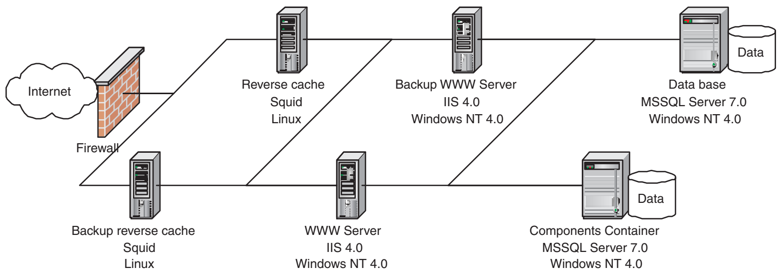

Fig. 5. Structure of the implemented system

Additionally, there was quite a large library of convenient functions for data accessing, and for manipulation of the call-environment and the parameter streams. This library offered a useful function to generate URL for pointing other components with the call-environment.

The discussed solution was built as a general tool for portal and e-shop development. In practice, it operated a hundred implementations of e-shops, e-catalogues and portals. The largest implementation – the portal – has been receiving a million hits per day. The component-expert architecture allowed a few programmers to develop the large solution in a short period of time. Modifications or extensions of the working solution were easy and cost-effective.

5 Conclusions and Future Work

Most important profits from the Component-Expert technology used for information Web portals are similar to those coming from portlets implementation, like expanding system properties, ease of programming in heterogeneous environment and internal simplicity of the components code. Some additional features are of interest:

– *Minimizing programmer responsibility for component choice.* In large-scale systems the programmers are not able to have the complete knowledge of the elements of the whole system. The programmer selects only a component type and creates a call-environment, while the component selection is done automatically. Many choices, which in traditional systems must be written in the code of components are omitted. They are written only once in the Expert System by the expert not by the component programmer.
– *Increase efficiency of programming process and releasing the first working system version.* In this architecture the components are internally simple, the programmers are independent, the system can start up based on a simple set of universal components. This implies that the programming process will be shorter and start up of system will be extremely quick in comparison with the classical approach.

- *Ease of solving new problems.* Extension of attributes and rules sets in the Expert System enables one to solve new kinds of problems, while previous applications will still be operational.

Similar to the traditional component technology, in the component-expert concept the application is divided into independent components. However, introduction of the components types, their specializations and managing them by the expert system makes the programmers more independent and constitutes a new paradigm in program creation and development. Most of the large-scale systems are characterized by gradual complication and continuous evolution. Intelligent management of components makes them more reusable.

At present, reusing portal by the same user needs repeated deduction. Although in most cases data caching results in improvement of the system performance, in the future learning procedure will be implemented to further system optimization. Next development will be concentrated also on implementation of semantic Web languages, like RDF, for both information content and component description.

The concept of the component-expert architecture seems to be useful for those large applications, which can be split into independent parts – just components. Examples of such applications are: grid projects (e.g. [4]), contemporary application interfaces with environment personalization, management applications, platforms for organisationally mobile employees, etc. This technology can be used also for applications, that use script languages as well as for applications written in compiled languages.

Acknowledgements. The work described in this paper was supported in part by the European Union through the IST-2001-32243 project "CrossGrid". AGH grant is also acknowledged.

References

1. Margulius, D.L.: Plug and Play Portlets, `www.infoworld.com/articles/fe/xml/02/04/29/020429feportaltci.xml`
2. —, WebSphere Portal for Multiplatforms, `www-3.ibm.com/software/webservers/portal/portlet.html`
3. —, Top Tech Trends - June 2002 ALA ANNUAL, American Library Association `http://www.lita.org/committe/toptech/annual02.htm`
4. Dutka, L., and Kitowski, J.: Application of Component-Expert Technology for Selection of Data-Handlers in CrossGrid, in: D. Kranzlmüller, P. Kacsuk, J. Dongarra, J. Volkert (Eds.), Proc. 9th European PVM/MPI Users' Group Meeting, Sept.29-Oct.2, 2002, Linz, Austria, Lect.Notes on Comput.Sci., vol.2474, pp. 25–32, Springer (2002).
5. Dutka, L., and Kitowski, J.: Expert technology in information systems development using component methodology, in: Proc. of Methods and Computer Systems in Science and Engng. Conf. Cracow, Nov.19-21, 2001, pp. 199–204, ONT, Cracow (2001) (in Polish).

6. Pharm, D.T. (Ed.): Expert System in Engineering, Springer-Verlag (1988).
7. Leondes, C.T. (Ed.): Fuzzy Logic and Expert Systems Applications, Academic Press (1998).
8. Vlahavas, I., and Bassiliades, N.: Parallel, Object-Oriented and Active Knowledge Base Systems, Kluwer Academic Publishers (1998).
9. Allen, P., and Frost, S.: Component-Based Development for Enterprise Systems: Applying the Select Perspective, Cambridge Univ. Press (1998).
10. Herzum, P., and Sims, O.: Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise, John Wiley & Sons (1999).
11. Brown, A.W.: Large-Scale, Component Based Development, Prentice Hall (2000).
12. Box, D.: Essential COM, Addison-Wesley (1998).
13. Rock-Evans, R.: DCOM Explained, Digital Press (1998).
14. Slama, D., Garbis, J., and Russell, P.: Enterprise Corba, Prentice Hall (1999).
15. Platt, D.S.: Introducing Microsoft .NET, Microsoft Press (2001).
16. Monson-Haefel, R.: Enterprise JavaBeans, O'Reilly & Associates (2001).
17. Sarang, P.G., et al.: Professional EJB, Wrox Press, Inc. (2001).
18. Roff, A.T.: ADO: ActiveX Data Objects, O'Reilly & Associates (2001).
19. Iribarne, L., Troya, J.M., and Vallecillo, A.: Selecting Software Components with Multiple Interfaces, Proc. of the 28th Euromicro Conf., Dortmund, Sept. 2002, pp. 26–32, IEEE Comput. Society Press (2002).