Computer Science Journal of Moldova, vol.**11**, no.3(33), 2003

NeoSite: A simple Content Management System

Oleg Burlaca

Abstract

This article describes NeoSite: a content management system that facilitates clear separation between content, logic and design. The system has a modular architecture and it's user interface is modeled (customized) using XML. The content is represented as a tree, whose nodes are called "datacells". The definition of a datacell is composed from: a list of attributes, the relationship specification with other nodes, interface modeling instructions. The system was conceived for web site implementation by programmers and information architects, after that, site maintenance is shifted to content authors, managers, etc.

Keywords: content management system, user interface, xml, usability.

1 Introduction

The activity of today's organizations can't be detached from information technologies. Web sites are used to promote companies and their products, offer services and information, facilitate communication. Web content needs to be managed quickly at a qualitative level. The problem of content management automation has arised. There are tools for creating web pages and even web sites, but it takes time to learn how to use them.

The site designer should be freed from creation and maintenance of HTML pages. When the site structure is settled down, page generation

should be automated. The system will take care about page cross-link validation in case of reorganization.

Most good ideas are easily understood. Usually they appear because of a need, and are accepted quickly. The delicate moment constitutes the transition from conceptual acceptance to practical implementation. A Content Management System is a great idea, easily accepted, but hard to accomplish.

There are a lot of perspectives on what "content management" really is [1]. Meta Group, for example, defines content management as "... a complex blend of functionality, including the acquisition, management, assembly, review and approval, effective publishing, retention and security of information bound for any of an organization's ... Internet, intranet, or extranet venues". But analyst views are always to be taken with a grain of salt, as they are typically the result of an individual analyst or two that is then influenced by the firm's marketing needs. These are specialized definitions that will fail to encompass future CMS. In addition to this, substantial overlaps exist with document management systems, knowledge managements systems, enterprise application integration systems, e-commerce systems and portals. A more abstract approach:

"A set of tasks and processes for managing content throughout its life from creation to archive" [2]

"An emerging field that's concerned with making information easier to manage and access inside corporate environments" [3]

In reality a CMS is a concept rather than a product. It is a concept that embraces a set of processes that will stay at the foundation of the next generation of web sites where content authors will have more privileges, duties and responsibility than designers and developers. The main purpose of a CMS is: enhanced integration and automatization of processes that contribute to efficient dissemination of information on the Internet.

Despite a wide spectrum of CMS [4], there is no dominant proprietary content management solution, especially for smaller sites. A

survey [5] conducted in January 2003, identified the major problems when designing for or implementing content management software:

- commercial software too expensive and required too much time to implement
- difficult to integrate with other systems
- determining requirements
- migrating old content
- training authors and editors

In fact, 61% of companies who have already deployed Web content management software still rely on manual processes to update their sites [6]. Users would like:

- Smaller and less complex systems
- Documentation should provide concrete ideas for how to structure project and make maximum use of features
- Scalable user interface: features can be removed/added as required
- Better UI for content creation and management of workflow.

A CMS implementation will be a success only if it's accepted by content authors, designers, managers and administrators. Today's requirements comprise integration with other systems that means new types of users. To be accepted, a system should offer a handy interface for all of its users. This point of view is sustained by "Chaos Report" [17] appeared in 1994, which concluded that the highest-ranking factor contributing to failed system implementations is lack of user involvement. Users care most about the interface. Developers tend to dedicate most of the time to the technological aspects of the system and often don't take into account its future users. Shortly, they care about how the system works, but not about those who'll use it.

2 Related Work

Tiramisu [7] is a declarative web site management system decoupled from concrete implementation tools. The designer defines site structure and content in a declarative way, after that, for each part of the site, the implementation tools are specified. The longer term goal of the project is to develop a federated architecture in which it is easy to incorporate new web site development and management tools. A key element of this architecture is the definition of a standard API to web-site management tools. The federative architecture has similarities to NeoSite GUI architecture based on modules that can interact with each other. Thus, new modules can benefit from services offered by existing ones.

Site Manager [8] is a 3D visualization tool for web sites. It displays the entire hyperlink structure of your Web site in a three-dimensional sphere that can be rotated and zoomed in/out to more closely examine a specific document's link hierarchy and still see how this "close-up" fits into the entire Web site.

XXL [9] – interactive development system for building user interfaces which is based on the concept of textual and visual equivalence. The XXL Builder provides three views of the interfaces that are being developed: the text view, the graph view and the widget view. The text view shows the corresponding descriptions in the source code. The graph view is an iconic representation that is equivalent to the text view. These two views constitute the dual (textual and visual) "abstract" specification of the UI while the widget view can be seen as the "result" of this specification.

Works mentioned above concentrates either on inner part (databases, declarative languages for site structure specification) or on external aspects (information visualization, using XML in user interface design) of development tools. The goal of the proposed system is to combine these two aspects in order to obtain a flexible and usable system.

Twingle [10] - a tool for content authors to work with networked information. The project is focused on improving three parts of the content management experience: Locating, Creating, and Collaborat-

ing. Each of these will be simpler, faster, and more powerful for average people that work with WCM (web content management) systems.

XcoP [11] - XML content repository based on an object-relational databases management system and improves content management of XML documents. Documents are called "fragments". A fragment consists of textual contents and may further comprise a set of other fragments. Fragments are stored as table rows and could be processed partially.

Newtenberg Engine [12] – a framework for modeling and generating web sites based on "boxes". A "box" is a way of describing a view over one or more "eidox". An "eidox" is the basic unit of content and is an abstraction of a document that includes the content, its properties and the generative history of the content.

XcoP's Fragment, Newtenberg Engine's Eidox and NeoSite's datacell are basically the same thing – an atom of information that cannot be splitted any more.

3 NeoSite overview

NeoSite is a client/server application that leverages content management within a website. Is promotes a good structuring of information and automates site maintenance. The system is designed for small to medium organizations. Basically, its users will be content authors and providers. If the information is well structured and has a high degree of granularity, then HTML knowledge is not required.

The system is not an integrated and independent solution for web site development, but rather a framework which is used to construct a unique solution based on site's specific requirements.

The system's motto is: A flexible system based on plugins that offers simple and usable interfaces for effective and efficient content management.

3.1 Important features

- Content management based on small, interconnected pieces of information (content cells)
- A content cell is defined by it's properties and it's relationship with other content cells
- High degree of cell interconnection
- Handy management (good interfaces) of information cells
- Doesn't need special/external viewers and editors
- Separation between content, design and application logic

3.2 Technical aspects

All parts of the system, except client application, was implemented using free software. MySql was chosen as database server: connectivity, speed and security has made MySql a standard de facto for web applications. Server side scripts and applications were written in Perl: the most used language for CGI scripts. The web server, and it's no surprising, is Apache, because it's free and is the most popular web server today: a Netcraft survey (www.netcraft.com/survey) showed that it runs on 55% of web servers. The client application was implemented in Borland Delphi and runs on all Windows platforms (9x, NT, 2000, XP).

3.3 Client application vs. Web interface

Web interfaces to content management systems are the most popular today. There advantages are: handiness and simplicity, there is no need to install special software on the client side. However, web browser interface elements and possibilities are limited, compared to common applications. Web interfaces fail to offer a clear and intuitive model for interacting with big volumes of information. In addition, an interface must also operate well in adverse conditions (poor internet

connection, rapid transfer and complex data processing), that is a difficult achievable goal for such interfaces. The system is intended for a high rate of editing, adding and interconnecting of information. These activities require more rapid and democratic interfaces, that's why a client application is more desirable in this case.

3.4 From concept to implementation

Determined objectives could be achieved only through a scalable, flexible and modular concept. From information structuring to some implementations for dynamic page generation, every aspect influences the accomplishment of a system that will be easy to support and develop in the future. The more rigorous and, at the same time flexible, is the initial concept the more smoothly will go future development. Abstractions identified at the inception phase of the project are essential in order to achieve a flexible system.

4 System architecture

The activity of most computer programs could be divided into three phases: obtain initial data (input), process it (logic) and present the result (output). These stages coincide with the layers from Model-View-Controller (MVC) framework. The MVC paradigm originated in the Smalltalk-80 system to promote a layered approach when developing graphical user interfaces. Today MVC is also used in structured development of complex systems, particularly in web applications [13].

MVC is the central concept of NeoSite's architecture that promotes clear separation between data, presentation and logic. It becomes possible to separate in time and space the processes of content management, template design and script development.

NeoSite's components are: database, user interface, tools for authoring, visualization and searching of content, templates, scripts for processing, crunching and presentation the content using templates.

The construction of a web site consists of:

- Selecting, integrating and management of data that are going to be published on the site
- Work out the site structure: page types, information used on each of these pages and links between them.
- Design the look of the site (art design)

4.1 A relational-hierarchical model for content management

The organization of human brain is based on hierarchies. Information that is categorized, classified, organized, presented to the user as a hierarchy is understood more easily. In NeoSite, site content is represented as a tree, like in Windows Explorer. The proposed model has the hypertext idea at its roots. W.Weiland and B.Shneiderman define hypertext as a "technique for organizing textual information through a complex non-linear method in order to facilitate rapid exploration of a large quantity of knowledge". The tree is stored in database as a self-referenced table (see Figure 1).

A node represents an unitary piece of information and constitutes a row in the database table. It can be said that in the tree, relations between the nodes are implicit: the parent-child relation. For specifying explicit relationships a new table has introduced (see Figure 2).

It's obvious that such a primitive approach is not suited for large projects that abound in semi-structured information, where XML is appropriate. The impact of XML on data storage and presentation gave birth to XML native databases [14]. Sometimes, possibilities offered by a DBMS, even XML native, aren't sufficient for today needs. Today's market offers content management servers for XML document applications [15] and XML content repositories [11]. In [14] it is said that "content management systems are specialized models of XML native databases", which is a narrow-minded approach in my humble opinion.

```
CREATE TABLE tree (
id mediumint unsigned NOT NULL auto_increment,                    # node identifier
parent_id mediumint(8) unsigned NOT NULL,                         # parent node identifier
tip tinyint unsigned NOT NULL,                                    # node type
title varchar(255) NOT NULL,          # title that appears in client application treeview
... user_columns ...,                                             # node attributes
modified timestamp(14) NOT NULL,
pos smallint unsigned NOT NULL,                 # node's position relative to its siblings
total int(10) unsigned NOT NULL,                                  # number of descendant nodes
PRIMARY KEY (id),
KEY parent_id (parent_id),
KEY tip (tip)
)
```

Figure 1. Table that holds the content of the site

```
CREATE TABLE tree_related (
from_id mediumint unsigned NOT NULL,                              # reference node id
to_id mediumint unsigned NOT NULL,                                # referenced node id
... user_columns ...                                # the specification of the relation
pos smallint unsigned NOT NULL,                                   # position of the relation
PRIMARY KEY (from_id, to_id)
)
```

Figure 2. Table that holds explicit relations

The proposed model is very simple, but for small to medium websites (www.patria.md, www.mavr.md, www.monument.md for ex.) with highly structured information this works well.

4.2 Client application

The client application was implemented in Borland Delphi. Flexibility and reliableness are guaranteed by its modular architecture based on plugins. Plugins can be developed in parallel by a group of programmers, which increase productivity and independence. A plugin (module) performs a set of tasks and can request services from other plugins. A plugin is simply a DLL library, loaded/unloaded by the main program (i.e. the .exe file) when needed. All the functionality is concentrated in plugins, the main program serves as a plugin moderator.

```
// user wants to edit a node (press F2 or double click the node)
Main LoadPlugin('edGeneric')                    the EdGeneric.npl library is loaded
SiteTree GetTipImg (7)                          get the glyph for the node with tip==7
TaskBar NewTask (EditorForm)                    add a button in the TaskBar
TaskBar ActivateForm (EditorForm)
EdGeneric Edit(881)                             editing the node with id==881

// user updates the node title and presses the 'Save' button
TaskBar ChangeCaption (EditorForm, 'A Title')   update the the taskbar button
SiteTree OnItemSaved (881, 'New Title')         treeview are notified about changes

// building the node
Srv ReadCfg ('builder_script')                  which is the builder script ?
CGI Get('id=881&tbl=site_ro')                   invoking the server script with GET method

// user closes the node editing window
EdGeneric Close
SiteTree OnItemClosed (881)                     node editing is over
```

Figure 3. The interaction between client application plugins

The core client plugins are:

- SiteTree, data tree management (visualization, updating). Note: Tree nodes are fetched from db on request.
- Srv, a helper plugin which acts as a service provider
- EdGeneric, manages a "data cell", i.e. a single node from the SiteTree plugin
- CGI, offers services for interacting with the web server (export commands for invoking server scripts)
- SQL, executes sql queries and show results in a grid. It is used mainly by administrators.
- TmplTree, template management

Figure 3 shows the sequence of events/actions generated on editing, saving and building a data cell.

4.3 Templates

Templates are document "skeletons" that describes the structure of a page (or a set of pages). The template defines when and where the content should be presented. Templates are stored in the database and managed from the client application.

A web page can be generated from several templates, and vice versa, a template can generate a set of pages. A template can process itself recursively to build a hierarchical structure for example.

Templates are processed on the server side using the Template Toolkit [16] - a collection of open source perl modules, which collectively implement a flexible and powerful template processing system, designed for constructing presentation elements and formatting data.

An alternative for Template Toolkit would be the XSLT, XML, CSS combination. Despite their popularity, using these standards is not an easy task. Nevertheless, XML and XSLT can be used in Template Toolkit through a set of modules.

4.4 Scripts

Web pages are generated by server side scripts. The script's work usually goes through the following set of stages:

- information extraction from databases
- preprocessing obtained data
- dispatching data to templates for page generation

Dynamic pages are generated implicitly; the static ones are generated explicitly by calling the builder script from the client application. Script functionality can be splitted into several modules, plugged in by templates on request (see Figure 4).

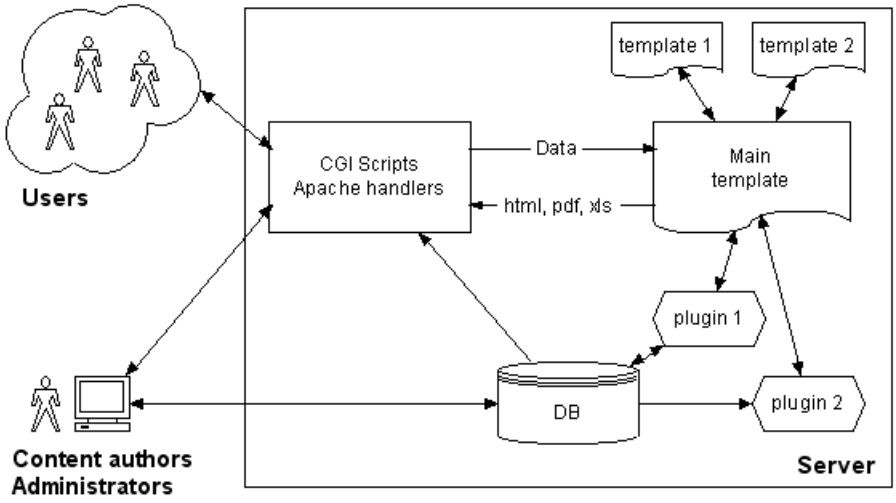

Figure 4. Server side implementation

5 DataCell

Content tree nodes are called 'datacells' (information cells). A cell represents a unique concept that has some textual information and a set of proprieties. The most important propriety of a cell is it's type. The cell type definition (see Figure 5) describes how a node will be displayed (icon, font, tab sheets), which db fields are used, and how this type of node relates to other types.

The key elements from the type definition are:

- **linkage,** fields from the *tree_related* database table used for describing the relation between two nodes
- **memos,** textual fields that require specialized editors with syntax highlighting
- **fields,** fields from the *tree* table: node proprieties

```
<tip>
   <has_props> N </has_props>
   <has_content> Y </has_content>
   <has_related> Y </has_related>
   <npl_editor> EdSql.npl </npl_editor>
   <npl_editor_debug> C:\NeoSite\Editors\EdSql.npl </npl_editor_debug>
   <shortcut> Ins </shortcut>
   <template> mon/item </template>
   <is_leaf> Y </is_leaf>

   <linkage>
     <rel_fields>
       <f> <title> Info </title> <name> info </name> </f>
     </rel_fields>
   </linkage>

   <memos>
     <f> <title> Ro </title> <db_field> content </db_field>
         <shortcut> Alt+R </shortcut> </f>
     <f> <title> En </title> <db_field> content_en </db_field>
        <shortcut> Alt+E </shortcut>  </f>
   </memos>

   <fields>
     <f> <name> id </name>  <readonly>Y</readonly> </f>
     <f> <name> filename </name> </f>
     <f> <name> thumb </name>  <editor>ImgEdit</editor>
         <initial_dir>C:\images\monuments</initial_dir>
     </f>
     <f> <name> stare </name>  <editor> ComboList </editor>
          <editor_items> <li>FB</li><li>B</li><li>S</li><li>PD</li><li>D</li>
         </editor_items>
     </f>
     <f> <name> recommend </name>  <editor>ComboList</editor>
         <editor_items> <li>N</li> <li>Y</li> </editor_items>
     </f>
     <f> <name> info </name>
         <editor>MemoEdit</editor> <height> 100 </height>
     </f>
   </fields>
</tip>
```

Figure 5. The definition of cell type

The most challenging part in the design of a web site using NeoSite is partitioning the content into datacells. Because this process is influenced by a lot of factors that sometimes are contradictious: granularity, dependencies, flexibility, scalability, performance, evolution, reusability degree.

6 User interface

Human-computer interaction is achieved through interfaces. The more the interface is designed the more the user is satisfied. When an interface is well designed, the time needed to access the information decreases, and the user adapts much more quickly with the software.

A CMS implementation will be successfully only if: 1. is accepted by content authors; 2. web site users are satisfied. That's why usability is a key factor. Usability is a science and art of designing interactive systems that are: easy to learn, efficient in utilization, flexible, stable.

Some elements and aspects of the client application user interface are modeled through a set of XML documents stored in the database. In Figure 6, the 'MONUMENT.MD' subitem is added to the 'View' menu item. When clicked, the 'SiteTree.npl' plugin is loaded with the specified parameters from <params> section.

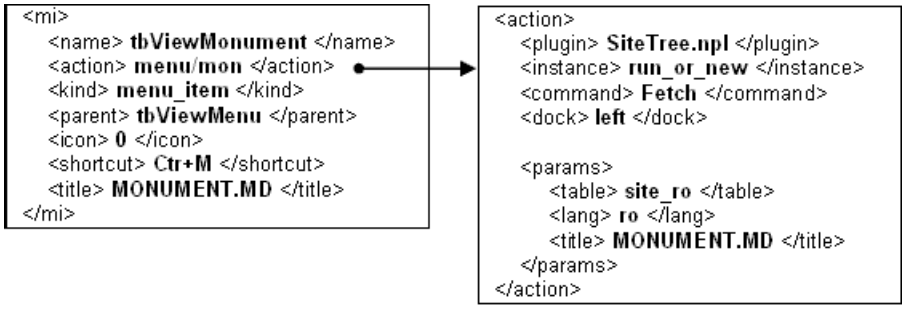

Figure 6. Modelling the interface using XML

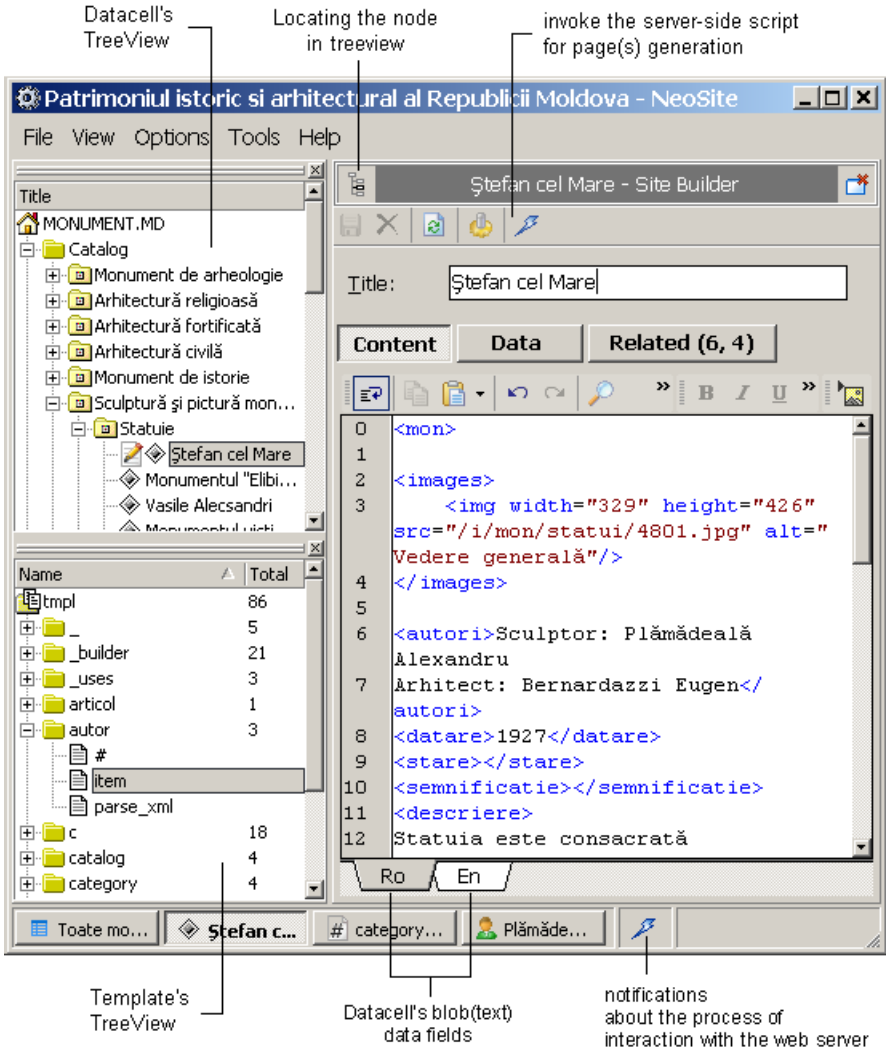

Figure 7. Client application user interface

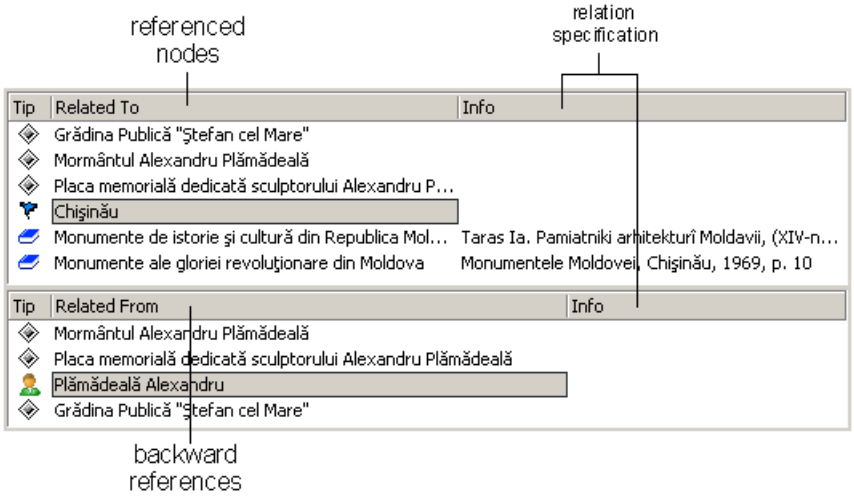

Figure 8. Node relationship management

A special attention was devoted to content authoring tools. In spite of its usability, WYSIWIG (What You See Is What You Get) editors weren't used because it doesn't allow advanced HTML programming, and the produced source code a larger that usual because of needless specifications.

The relationship between nodes are established using a drag& drop operation: the referred node is dragged to the 'Related To' section from the node editing window (see Figure 8).

The interface for editing node properties is built based on the <fields> section from the node type definition. This section specifies the editors (ComboList, ImageEditor, NumericEditor) used for updating the fields. In Figure 9 you can see that the 'semnificatie' property is edited using a ComboList editor.

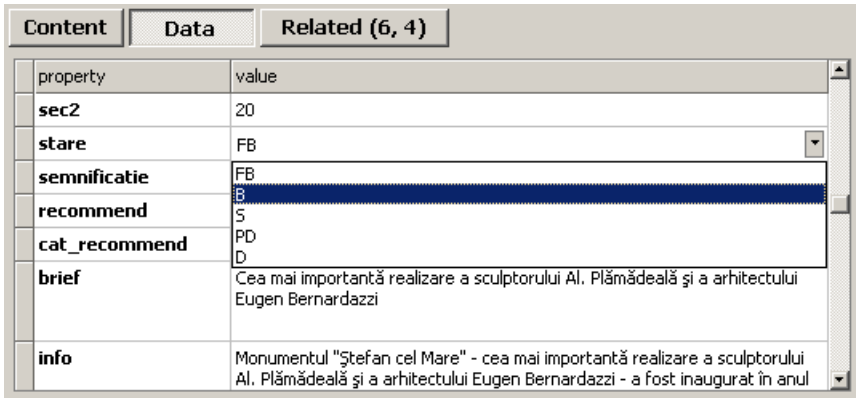

Figure 9. Editing datacell properties

7 Conclusions

Effective web site content management can be done using relatively simple instruments and limited resources. This article introduces NeoSite: a content management system for small to medium websites.

Comparing to other CMS's, NeoSite misses features like versioning or workflow support for now. Our system will never beat a commercial solution like Microsoft CMS because of limited human, time and financial resources. This system is more like a test platform for methods and ideas that, in my opinion, could enhance cms usability by providing a crystal-clear model for managing the content. When we developed this system and used it to implement a series of web sites we asked ourselves why such a simple approach, based on a tree with explicit node relationship, is so effective. We realized that, involuntary, We've used the hypertext paradigm and applied it to the problem of data management (adding, updating). The question now is: can this model be further enhanced to leverage the processes of adding, updating, relating the content?

Meanwhile, we should keep an eye on today trends in cms field. It's important to mention that the primary use of computing technology continues to evolve to be more for communication than for data crunch-

ing [18]. Communication means lots of users (humans and software) with its own specific needs. It's obvious that custom development + mission critical = expensive [18]. Vendors will have to expand their solutions to maintain business models that require large sales. Open Source software will surely influence the evolution of enterprise content management. Compelling evidence in this direction is the advent of the Zope CMS [19] - one of the leading open-source Web application servers and content management systems. Zope has won many converts in the last two years, and a vibrant development community has developed all over the world.

The purpose of an information system is: 1. satisfied clients 2. users acceptance.

Successful implementation of web sites showed that the NeoSite system accomplishes its objectives.

8 Future Work

NeoSite stores and displays data as a tree. If there are a large volume of content (the tree has a lot of nodes), a single method for data management becomes inconvenient. The system has to offer complementary tools for data visualization and editing. In other words, it would be appropriate to have more perspectives on the same content.

The system was conceived to be used by people. In the future, content providers could become: other CMS's, intelligent agents, email system etc. The system will be administrated directly: by users, and indirectly: by other software through a communication protocol.

References

[1] *What is Content Management*, The Gilbane Report, Vol. 8, No. 8, October 2000. www.gilbane.com

[2] Ovum, http://www.ovum.com/

[3] Louis Rosenfeld, *Content Management and Information Architecture*, CMS Watch, http://www.cmswatch.com/

[4] Paul Browning, Mike Lowndes. *JISC TechWatch Report: Content Management Systems*, September 2001

[5] *The Problems with CMS*, The Asilomar Institute for Information Architecture (AIfIA), http://aifia.org/pg/the_problems_with_cms.php

[6] "Web Content Management: Covering the Essentials, Avoiding Overspending", Jupiter Research 2003, http://www.jupiterresearch.com

[7] C.Anderson, A. Levy, *Declarative web-site management with Tiramisu*, ACM International Workshop on the Web and Databases (WebDB'99), June 1999

[8] *Site Manager - Software for Web Publishing and Site Management*, http://www.sgi.com/software/sitemgr.html

[9] Eric Lecolinet, *XXL: A Dual Approach for Building User Interfaces,* ACM Symposium on User Interface Software and Technology 1996. http://www.inf.enst.fr/∼elc/XXL/index.html

[10] *Twingle - a tool for content authors to work with networked information*, OSCOM http://www.oscom.org/Projects/Twingle

[11] B. Surjanto, N. Ritter, H. Loeser, *XML Content Management based on Object-Relational Database Technology*, Web Information Systems Engineering 2000

[12] Carlos Castillo, *A Framework for the design and implementation on web sites*, IADIS WWW/Internet 2002, http://citeseer.nj.nec.com/castillo02framework.html

[13] Alan Knight, Naci Dai. *Objects and the Web.* IEEE Software, March/April 2002.

[14] Ronald Bourret, *XML and Databases*, http://www.rpbourret.com/xml/XMLAndDatabases.htm

[15] T. Arnold-Moore, M. Fuller, *Architecture of a Content Management Server for XML Document Applications*, Web Information Systems Engineering 2000, http://citeseer.nj.nec.com/arnold-moore00architecture.html

[16] Andy Wardley, *Building and Managing Web Sites with the Template Toolkit*, Canon Research Centre Europe Ltd., http://template-toolkit.org/tpc4/

[17] The Standish Group, *The CHAOS Report*, http://www.standishgroup.com/sample_research/chaos_1994_1.php

[18] *The top 10 trends in content management*, The Gilbane Report, Vol. 10, No. 4, October 2002. www.gilbane.com

[19] Zope CMS, www.zope.org