Journal of Web Engineering, Vol. 10, No.2 (2011) 087-108

ENTERPRISE FRAMEWORKS FOR DATA INTENSIVE WEB APPLICATIONS: AN END-USER DEVELOPMENT, MODEL BASED APPROACH

FRANCA GARZOTTO

HOC- Hypermedia Open Center

Department of Electronics and Information, Politecnico di Milano (Italy)

garzotto@elet.polimi.it

Received June 20, 2009
Revised May 18, 2011

This paper investigates *enterprise frameworks* in the context of *data intensive web applications*, and proposes an approach that integrates the paradigms of End User Development and Model Based development. An enterprise framework denotes a reusable, semi-complete application "skeleton" that can be easily adapted to produce custom software products in a specific business sector. Traditionally, it is conceived as a *tool for expert software developers.* In contrast, we propose to regard enterprise frameworks as tools that enable *domain experts* to develop for data intensive web applications in a given field without the need of technological training or support by expert programmers. We propose the adoption of a *model-driven* process for framework-enabled development, based on *conceptual models* that are appropriate for the framework domain and domain experts can understand, adapt, and customize. We discuss requirements for and benefits of combining the two paradigms, and exemplify our approach presenting CHEF, an enterprise framework for data-intensive *multichannel* web applications in the domain of *cultural heritage* and *cultural tourism*. CHEF has been developed in the context of a wide international initiative called MEDINA and has been intensively evaluated in this and other projects.

Key words: Data Intensive Web Application, Enterprise Framework, Web Model, Web Design, End-User Development, Meta Design
Communicated by: D. Schwabe & G. Rossi

1 Introduction and Motivations

The concept of *enterprise framework* specializes the notion of *application framework* [7][8] [31][39], and denotes a reusable, semi-complete application "skeleton" that can be easily adapted to produce custom software products in a *specific business sector"* [6]. Like application frameworks, enterprise frameworks provide powerful means to support *reuse*, and reduce the burden of re-creating and re-validating common design or implementation solutions to recurring application requirements. While many examples of application frameworks can be found in the literature, documented exemplars of enterprise frameworks are relatively few [3] [18][19] [43], and address broad application areas such as telecommunication, avionics, manufacturing, business, and financial engineering.

This paper investigates enterprise frameworks supporting the development of *data intensive web applications*, referred to as "Data Intensive Web Enterprise Frameworks", or *WEFs* for short. We propose an approach to WEFs that integrates two paradigms: *Model Based Development* and *End User Development*.

In a *model-based* approach [1][4][5][13][14][40], development activities are driven by the definition of a *conceptual model* of the data intensive web application to be build. At *design time*, an *application model* is created using proper modelling primitives and notations. The application model captures the general information, navigation, and lay-out properties of the application under construction abstracting from any implementation detail. According to the general principle of separation on concerns, the application model is typically composed of three separate *sub-models*. The *information model* describes the types of information objects that the end user perceives in the application (*information schema*). The *navigation model* describes the navigation structures (*navigation schema*), which enable users to navigate across the information objects of the different types. The *presentation model* describes the visual lay-out structures (*presentation schema*) for rendering information and navigation objects on the screen. At *instantiation time*, the different sub-models or schemas are *instantiated* with multimedia contents, links, and lay-out properties, of the final application. Model based approaches have been proved to be more effective for supporting usability, reuse, and consistency of information architecture, navigation structures, and interface than a "page-by-page" development carried on without any previous conceptual design [14]. As such, model based approaches have been widely adopted in web application frameworks ([30][37][41]). Still, they have been seldom exploited for *domain-specific* web frameworks, i.e., enterprise frameworks [16][17][18][19].

End User Development (*EUD*) approaches shift the control of design, development, and evolution of an application from skilled ICT professionals to end users and promote the idea of building systems that enable non programmers "not only to personalize computer applications and to write programs, but to design new computer based applications without ever seeing the underlying program code" [42]. "The dominating design goal of a EUD system is to compensate for a discrepancy between domain specialists' expertise and the development task to be performed" [44]. The adoption of the EUD paradigm for WEFs implies the need of transforming them from *tools for expert software developers* to *tools for people* who are *not technologists* but are the key owners of problems in a domain, i.e., *domain experts*, offering them opportunities for extensions and modifications. As EUD WEFs are meta-tools (i.e., "applications for building applications"), they involve two categories of "end users", who might not be the same population: i) domain experts, who use the framework functionality for development purposes; ii) end users of the resulting applications, who search and navigate contents for domain specific tasks. For this reason, and because EUD WEFs involve development processes at two levels - building the design space (by ICT professionals) and building applications using the design space (by domain experts) - this class of systems involve a *Meta-Design* approach as defined in the EUD community [9][10][11].

The rest of this paper discusses the design requirements induced by the combined adoption of a EUD approach and a model based approach induces on WEFs (section 2). Then it describes an example of WEF that meet these requirements, named *CHEF*, an enterprise framework for data-intensive *multichannel* web applications in the domain of *cultural heritage* and *cultural tourisms*. CHEF allows cultural heritage professionals or cultural tourism specialists to *design* high quality, content-rich web sites, to build them in a relatively simple way, and to make them available on *different devices*, e.g., conventional PCs and handheld devices, without learning any specific implementation technology.

The rest of the paper is organized as follows. Section 3 discusses CHEF at conceptual level, describing its modeling characteristics, and introduces its functionality and interface. Section 4 presents CHEF software architecture. Section 5 outlines the empirical evaluation of CHEF. Section 6 frames our work in the current state of the art, and section 7 draws the conclusions.

2 General Requirements for EUD, Model Based WEFS

The combination of the *EUD paradigm* with a *model-based* approach induces a number of functional and non functional requirements for WEFs:

1) To foster the adoption of a framework by non programmers, this should be (obviously) *easy to use*, hiding the complexity of the implementation underlying the tool and the products developed with the tool; it should be *fast to learn*, and *quick* in allowing developers to publish a web site once all contents, links, and lay-out features are designed and instantiated. Everything from building hypermedia structures to inserting or updating multimedia contents and links and publishing the result must be intuitive and must require few clicks.
2) The framework should include a set of *design functionalities* that enable domain experts to produce the *design specifications* (i.e., the "schemas" that compose the "application model") for the system under development, at the three conventional levels of data intensive web application design: information design, navigation design, and presentation design. The underlying design language (i.e., set of concepts and primitives for creating design specifications) should be *simple* enough for domain experts, and *close to their mental model and their view of the domain.*
3) To support *learning by example* and facilitate *design reuse*, a *reference model* of information, navigation, and presentation design should be provided, which must be easily understandable by domain experts, specific for the business sector of the framework, and flexible enough to accommodate the modeling requirements of a large spectrum of applications in this sector. The framework should enable domain experts to reuse a reference information model tout-court, simply instantiating it without any design customization. Domain experts should also have the possibility of customizing the reference model at some degree, and the customization task, if needed, should not require any implementation effort. More precisely, concerning navigation design domain experts should be enabled to either reuse the built-in navigation patterns offered by the reference navigation model, or to customize them, or to adopt different navigation patterns provided by the framework, without any programming effort. Concerning presentation design, any web application typically has, at different degrees, its own lay-out requirements (e.g., colors, logos, backgrounds, button arrangements, etc.) and customization can be a complex activity. A EUD framework should support a repertoire of high-quality page lay-out specifications among which developers can choose the ones to adopt tout-court or the ones to customize.
4) Once application schemas are specified, the framework should create a *schema-compliant instantiation environment* (semi) automatically or in a very simple way. The instatiation environment should enable developers to populate the customized schemas with information, navigation, and presentation contents consistently with schema specifications. The interfaces for the above operations should be simple and intuitive, and should enable smooth transitions among different instantiation tasks.
5) The success of a EUD environment depends on a fine balance between effective software tools and effective *process support* [37]. Therefore the *simplicity* and *effectiveness of the development process* are equally important as the effectiveness and usability of the software tool per se. The adoption of a model based approach for a WEF makes, in principle, the overall development process more systematic and effective, since it supports a clear distinction between design and instantiation tasks and promotes a design process that proceeds along different levels of abstraction and complexity. Still, a model based process might not be obvious to domain experts and must be well understood by them. While expert developers know by experience which activities are required to design and build a good web

application, and how to organize them, the same is typically not true for non web professionals. Involving "domain experts as developers" leads to wider definition of WEFs. They should be regarded them not only as software tools but also as "packages" that include software *and* a usable *methodological support*, providing *guidelines* on the development process that capture *process expertise* and make it *reusable* by domain experts.

3 CHEF - An Example of Model Based EUD WEF

CHEF (Cultural Heritage Enterprise Framework) is an enterprise framework for *multichannel* data intensive web sites in the domains of cultural heritage and cultural tourism that implements the principles of End User Development and Model Based Development. CHEF allows cultural heritage or cultural tourism specialists to create *"by-reuse"* high quality, content-rich web sites in a model driven fashion and in a simple way, making them available on *different devices*, e.g., conventional PCs, PDAs, or mobile phones, without learning any specific implementation technology.

CHEF has been developed in the context of a large international initiative called MEDINA - "MEDIterraneaN by INternet Access [25]. The goal of this project is to build a multimedia *multichannel* web "Portal on Mediterranean Cultural Tourism. MEDINA purpose is to exploit ICT technology to promote cultural tourism in the Mediterranean area, especially in less developed countries. It aims at encouraging innovative ways of doing tourism, through the deep understanding of local civilizations and the promotion of destinations that are usually not mentioned in standard "all inclusive" packages proposed by tour operators. To achieve these goals, the project built of a "federation" of eight *National Web Sites* (for Morocco, Tunisia, Algeria, Cyprus, Malta, Lebanon, Syria, Jordan) under the umbrella of a *Mediterranean Portal* [20][25]. Each National Web Site addresses the material and immaterial heritage of a specific country and pinpoints the relationship with the culture of other Mediterranean nations. The Portal focuses on the common cultural roots of the different regions, highlights the cross-influences of the different civilizations born in the Mediterranean basin, and acts as a "gateway" towards the National Web Sites. Linked to the portal is a web site called "A Day in...", composed by a set of multimedia "hyperstories" that explore the life and cultural in some fascinating but less known Mediterranean towns [20].

CHEF was intensively used by the cultural experts of Ministries of Tourism and Culture, Museums and Cultural Institutions, Cultural Tourism Associations, National Tourism Agencies in twelve Mediterranean countries (Italy, Greece, France, Morocco, Tunisia, Algeria, Cyprus, Malta, Lebanon, Palestinian Authority, Syria, Jordan). Their effort represents approximately 90% of the resources needed to develop the overall "MEDINA system" (composed of the Portal, the 8 National Web Sites, and the "A Day in…" section). Programmers and graphic designers accounts for 10% of the overall resources.

MEDINA domain experts contributed to the elicitation of CHEF requirements and the validation, testing, and overall evaluation of the intermediate prototypes and of the final system (see section 5). They intensively used the framework, which was later adopted in a number of projects in cultural heritage involving small museums o cultural institutions in Italy and abroad.

CHEF was designed to meet the requirements discussed in the previous section. In addition, CHEF exhibits a number of characteristics that are induced by the multichannel nature of its target applications:

1) It provides an *integrated* environment to support design *for different delivery devices*, helping users to describe which application "properties" (e.g., types of information objects,

navigation tasks, and interface elements) are *common to all devices*, and which are *device specific*. The design process is carried on through multiple *customizations* of a general design space, by means of a simple visual interface that does not require any technical know how.

2) It supports a *flexible workflow*, enabling smooth transitions among different design and instantiation tasks for the same device and for different devices.

3) It provides an integrated environment for delivering the final application on stationary and mobile devices, hiding the implementation complexity of page generation and the plethora of technical details related to each device.

3.1 CHEF Modeling

Application design is the difficult part of any model-based development process and therein lies the real challenge for an EUD model based WEF: supporting abstract conceptual thinking – a task that domain experts oftentimes are not used to. To address this issue, the modelling apparatus of CHEF is not only specific for data intensive web applications in the field of cultural heritage/cultural tourism, but it has been defined with, and validated by a large team of domain experts. In CHEF, application modeling is a *design-by-reuse activity* that consists of the instantiation of a Meta Model or a customization of a Cultural Model, as discussed in the rest of this section.

3.1.1 CHEF Meta-Model

The CHEF meta-model, hereinafter called CHEF-MM, provides a high-level semantic language in which constructs directly reflect some general abstractions within the target domain, and can be used to describe the entities of a specific application in cultural heritage or cultural tourism regardless the characteristics of a specific device. CHEF-MM is inspired by the modeling primitives of existing hypermedia/web design languages HDM [13] [14] and IDM [4]. With respect to its ancestors, CHEF-MM can be regarded as a generalization of these models that uses simpler concepts and a terminology that is more appropriate for culture and tourism experts, while attempting to be accommodate the requirements of a wide spectrum of cultural or tourism applications on different delivery devices.

CHEF-MM provides a *schema definition language* for the three kinds of schemas resulting from a model-based design process. Figure 1 sketches a high level UML representation of the key primitives of CHEF-MM, where CHEF-MM primitives are represented as (meta)classes. Table 1 groups these primitives by design activity.

Cultural Axis (*Axis*, for short).

This class defines the *types* of "topics" that are of interest for end users of the application under development. An Axis[a] defines a multimedia data structure composed by the following elements: Title, Subtitle, Short Description, Description (textual attributes); *Images* (list of cardinality N of structured multimedia attributes <Large size image, caption>); *Small images* (list of cardinality N of structured multimedia attributes <Small size image, caption>); *Audios* (list of cardinality N of structured multimedia attributes <audio, (optional) transcript>); *Videos* (list of cardinality N of structured multimedia attributes <video, (optional) transcript>); *URLS* (list of cardinality N of structured multimedia attributes <url, comment>). The last attribute represents links to web sources external to the application under development. To address the need for multi-linguism (a

[a] The term "Axis" was proposed by MEDINA cultural tourism experts

frequent requirement of cultural applications) textual and audio attributes are indeed substructured, to allow designers define multiple "versions", in different languages, of the same attribute.

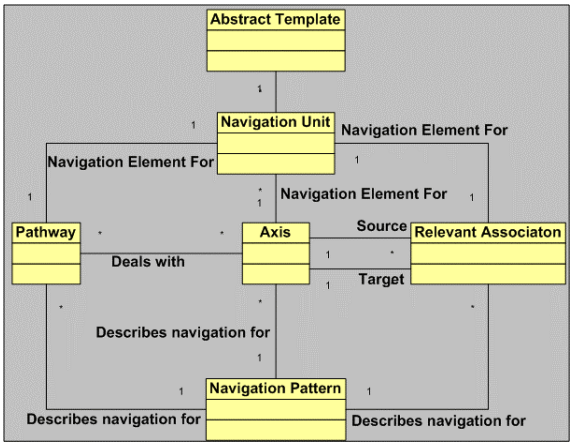

Figure 1. CHEF Meta-Model at a glance

Table 1 - CHEF Primitives grouped by Design Activity

Design activity	**CHEF-MM primitives**
Information modeling	*Cultural Axis, Relevant Association, Pathway*
Navigation modeling	*Navigation Unit , Navigation Pattern*
Presentation modeling	*Abstract Template*

Relevant Association

This class denotes *types* of relationships among topics, defined by their source and destination Axes, and the relationship cardinality.

Cultural Pathway (Pathway for short*)*

This class represents "groups of topics" about a given cultural theme that are relevant for the application under development – a kind of virtual itineraries about subjects of interest for the application end users. A Pathway is defined by a Theme attribute and a set of attributes (similar to Axis) for specifying, at design time, the types of contents of the Introductory section of a Pathway subject and the types of topics (i.e., Axes) around which the itinerary is built.

Navigation Unit

This class defines the "conceptual pages" that correspond to the topics of a given Axis, or the (instances of) a specific Pathway, or Relevant Association, and in the final application are rendered through a concrete interface. An instance of a Navigation Unit specifies which attributes of a given Axis, Pathway, or Relevant Association, must appear on a page of a given type.

Navigation Pattern

This class defines "abstract", generic navigation topologies [37] that describe how users move across a group of pages. CHEF-MM provides an extendible *pattern language* [12] that includes the most popular navigation patterns: *Index, GuidedTour, Index+GuidedTour, All-to-all, Automatic Guided Tour* [15]. For example, the *Index Pattern* defines a one-level tree topology, i.e., a set of bidirectional links from an assigned member of a group of objects to all other objects. The *Guided Tour* pattern defines a sequence topology, i.e., a set of bidirectional links from a member of a group of objects to the "next" and the "previous" ones. An Automatic Guided Tour has the same topology, but assumes that link transition is executed automatically (under the control of a dynamic, time based media, such as audio or video).

Abstract Template

This class provides an abstract representation of the lay-out properties of page types. An Abstract Template specifies the spatial position of the different media elements defined for a Navigation Unit, of the navigation elements (links) induced by its Navigation Pattern, and of the interaction elements that enable navigation across multiple pieces of the same media (e.g., multiple images) within the same page. An Abstract Template is general enough to accommodate, through simple customizations, the presentation requirements of any type of navigation unit defined according to CHEF-MM.

Information Schema, Navigation Schema, and Presentation Schema are defined in terms of the above primitives as follows:

- *Information Schema:* a set of Axes, Relevant Associations, and Pathways that instantiate the corresponding meta-classes in the Meta-Model.
- *Navigation Schema:* a set of Navigation Units (instances of the Navigation Unit meta-class) and Navigation Patterns *associated to* each Axis, Pathway, or Relevant Associations defined in the information schema.
- *Presentation Schema:* a set of instances of the Abstract Template class, called *Presentation Templates*. They instantiate the Abstract Template class with the actual lay-out properties of the different Navigation Units.

3.1.2 CHEF Cultural Model

The *CHEF Cultural Model* (or *CHEF-CM)* has been introduced to reduce the effort needed to instantiate the CHEF Meta-Model. Defined with a set of cultural experts, it offers an *application design "skeleton"*, i.e., a "reference" application model. CHEF-CM can be regarded as a *default instantiation* of the CHEF meta-model: It provides an Information Schema, a Navigation Schema, and a Presentation Schema. The latter is composed by a set of high quality Presentation Templates that can satisfy the visual requirements of wide spectrum of data intensive cultural applications delivered for stationary and mobile devices. These schemas represent default application design solutions that can be adopted tout-court or partially (integrating it with user-defined customizations of the Meta-Model) before filling it with multimedia contents and links at instantiation time.

In CHEF-CM, the set of built-in cultural Axes (instances of the Axis class of CHEF-MM) include: *material heritage, performing art, folk tradition, food&gastronomy, handicraft, (economic-historical-social) background, geographical info.* Built-in Pathways are groups of homogenous topics belonging to the same Axis.

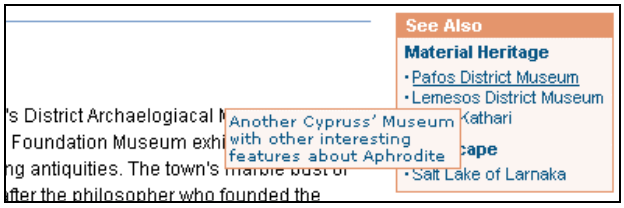

Figure 2. Display of a descriptor attribute explaining why another instance of Material Heritage - Pafos District Museum - is associated to the current one - Cyprus Archeological Museum (they both hold artifacts related to Aphrodite)

(a)

(b)

Figure 3. MEDINA National Web Site (Tunisia): A page corresponding to a topic of Axis "Material Heritage" – stationary device version (a) and PDA version (b)

A single Navigation Unit (instance of the Navigation Unit class in CHEF-MM) is specified for each built-in Axis and contains all the attributes defined for that Axis. The built-in instances *of* class *Relevant Associations* are the relationships "X-SEEALSO-Y [m,n]", where X and Y stand for the names of any built-in Axis and represent the arguments of a m-n relationship. The Navigation Pattern that defines navigation across "pages" related by X-SEE-ALSO-Y relationships is Index, while Index+Guided Tour is the Navigation Pattern for Pathways. In the stationary version, Axis specifications also include a *descriptor attribute* useful to explain why a cultural topic of a specific axis is linked to another one (see figure 2). The descriptor, the textual attributes (captions) required for images, and the video transcripts (see the specification on CHEF-MM) ensure that CHEF final applications fulfil some W3C Web Content Accessibility Guidelines (WCAG), i.e., are appropriate, in principle, also for visually impaired users.

The attributes defined for the various built-in Axes might be different in the information design of stationary and mobile devices, which also differ in terms of Relevant Semantic Associations and Pathways.

In the examples shown in figure 3, the information design of the stationary version differs from the mobile version as it includes multiple images, audio and video contents, a number of Relevant Associations, and several Pathways.

3.2 CHEF Development Process

To support the development process and the *reuse of process expertise*, CHEF is delivered with a *clear documentation* of developer's activities, which has been widely validated by a team of thirty-two framework users - domain experts from all over Mediterranean - during workshops organized within the MEDINA project, and along the development work of the overall project.

CHEF Development activities are organized along the two major roles that CHEF users can play and that we call *editorial designer* and *editorial author.*

3.2.1 Editorial Designer Activities

Editorial Designers operate at *design time.* Their job is to define the information, navigation, and presentation *schemas* for the application under development. They usually produce different specifications for each device, to meet the device specific requirements in terms of interaction style, pointing mechanisms, or display constraints. The Editorial Designer is responsible for the following activities: *Specification of the actual Cultural Axes, Specification of Relevant Associations, Pathway Specification, Specification of Navigation Units and Navigation Patterns, Specification of Presentation Templates*

Specification of the actual Cultural Axes

Axes can be selected from the CHEF Cultural Model, or can be created by instantiating the CHEF-MM Axis Class. Customization consists of selecting the proper combination of attributes and defining the cardinalities of list attributes for the Axis under definition. A major decision at this time concerns the language(s) in which (audio and textual) contents are offered in the final application.

Specification of Relevant Associations

Associations can be selected from the CHEF Cultural Model, and/or created by instantiating Relevant Association Class of the CH Meta Mode (specifying Association name, actual Cultural Axes involved in the relationship, and relationship cardinality).

Pathway Specification

For each actual Pathway, the *Theme* and the *Axes* for its member objects must be defined. For example, the editorial may define a Pathway that have "Rituals" as Theme, and Folk Tradition and Cultural Heritage as Axes for pathway objects.

Specification of Navigation Units and Navigation Patterns

Each Axis must be decomposed into the corresponding *Navigation Units. Navigation Patterns* must be associated to each Axis, Relevant Association, and Pathway in the information schema. The editorial designer may *reuse* the default Navigation Patterns of CHEF Cultural Model, or select other patterns from a *Pattern Library*. As discussed in 3.1.1, CHEF-MM offers provides a rich pattern language that includes the most popular navigation patterns among which the designer make her selection. The language is extensible, but the inclusion of new patterns requires the intervention of programmers to implement and include them in the design environment.

Specification of Presentation Templates

For each Axis or Pathway in a specific device, the corresponding Presentation Templates must be defined. Presentation Templates, that instantiate the Abstract Template of CHEF-MM, can be obtained in two ways: i) by (visually) *mapping* content or navigation elements of Navigation Units to Meta Template components, and *decorating* them with visual or typographical properties (colour, shape, size, background,…); ii) by "reusing" existing Presentation Templates, available in the Presentation Template Library of CHEF, possibly adapting them to meet the requirements of the application under development (e.g., to reflect the visual solutions adopted in the "corporate image" of an institution). Figures 3 and 4 show examples of pages that adopt different lay-out templates.

Figure 4. MEDINA Project: A page for Axis "Food and Gastronomy" in "A Day in …" section of the MEDINA Portal

3.2.2 Editorial Author Activities

Editorial authors operate at *instantiation time*. They define:

- which instance(s) of the various Axes (defined at design time) must be available in the final application for each device;

- how they are related (consistently with the definition of Relevant Relationship objects)

- in which Pathways they occur.

Editorial authors define and update the values of the attributes of the various instances. The production of individual content pieces is largely performed outside the CHEF framework, using appropriate multimedia authoring tools. Only few textual attributes may be created on-the-fly in CHEF: titles, sub-titles, image captions, or "descriptors" in Relevant Associations (Figure 2).

3.3 CHEF Tools

The various user activities are supported by a sophisticated suite of web based *integrated visual* tools: *Customization Design Tool, Mockupper, Instantiator, Previewer, Static and Dynamic Generators.*

Customization Design Tool

It allows the editorial designer to define information, navigation, or presentation schemas. It generates a set of design specifications, and the *customization parameters* used by the Dynamic Generation Tool for automatic customization of the Instantiation Tool.

Mockupper

Invoked at design time from the Customization Design Tool, the Mockupper creates a *fictitious application* that reifies the design specifications defined by the Editorial Manager. It exploits examples of multimedia contents and links (pre-stored in CHEF at configuration time) to *automatically instantiate* the schemas of the current design, and, through the Dynamic Generator, to produce an interactive web based mock up. The Mockuper generates a very rough prototype that can serve different purposes: for training and "learning by example"; for sharing design understanding among the development team; for experimenting navigation, interaction, look&feel solutions; for discussing the requirements and design choices, and deciding how to adjust or improve them.

Instantiator

This is a *schema-driven data-entry tool* for the editorial author. It provides the proper data entry "forms" for each instantiation task and for each delivery device. Each data entry form is automatically *customized* (by the Dynamic Generator) according to the design schemas of the application under development, and includes the insert/update fields for the attributes specified in the design schemas of the application. Figures 5a, 5b, and 5c show instantiation forms customized for the Portal of Mediterranean Cultural Tourism in the MEDINA project.

Previewer

The *Previewer* is a feedback tool that allows editorial authors to immediately inspect the effects of their work as it proceeds during the instantiation activity. The Previewer can be invoked at any time from any data entry form, and allows editorial authors to see and navigate across the pages corresponding to all the instances they have created or updated so far. Since the Previewer exploits the capability of the Dynamic Generation Tool, which is also responsible for the generation of the final application, preview pages have the same lay-out and navigation capability as the final end-user application (on a desktop or mobile device). This tool provides an *immediate visual feedback* of design and instantiation tasks, which facilitates the creation of a shared understanding within the development team of what is achievable, and of the effects of different design and instantiation choices.

Generation Tools: Static and Dynamic Generators

CHEF supports both the dynamic generation of the application pages (through a *Dynamic Generator*) and the "batch" generation (through a *Static Generator*) of the entire hypermedia network, made of interlinked "static" pages. The behaviour of the Dynamic Generator is hidden to CHEF users and is outside their control, but is the core engine for the overall environment. It is exploited for web based delivery of the final application, for preview purposes (when invoked by the Previewer or the Mockupper), and for generating the data entry forms of the Instantiation Tool. In contrast, the *Static Generator* creates a *static HTML version of the entire application*. This static application is useful for off-line use of the application, e.g., when the application (either in its final version, or for demo purposes) is delivered on CD-ROM

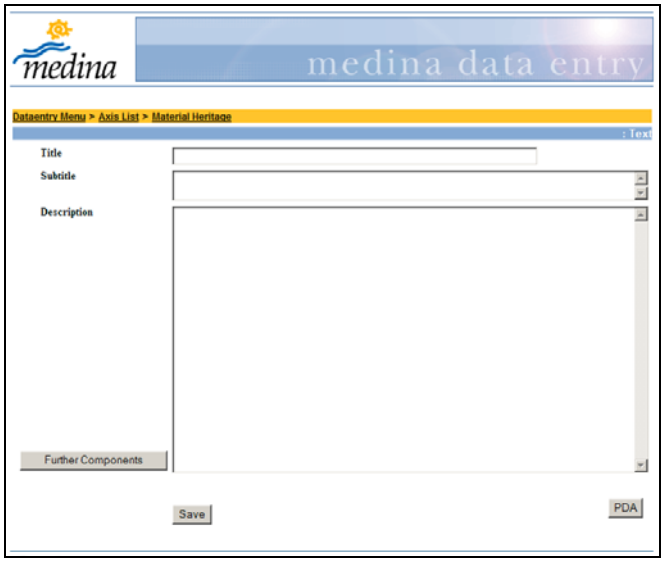

Figure 5a. Instantiator output for the MEDINA project: Customized Data Entry Form for Axis "Cultural Heritage"

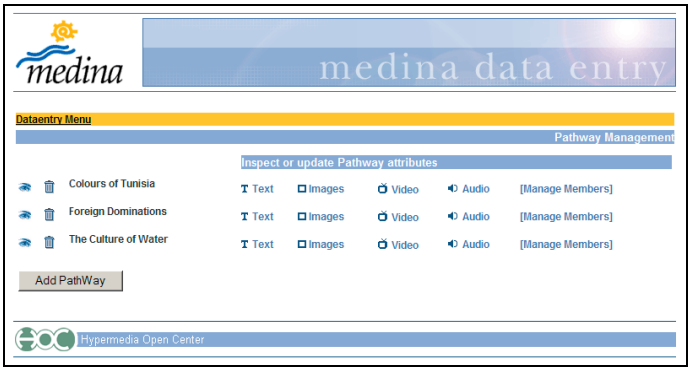

Figure 5b. Instantiator output for the MEDINA project: Customized Data Entry Form for Pathways

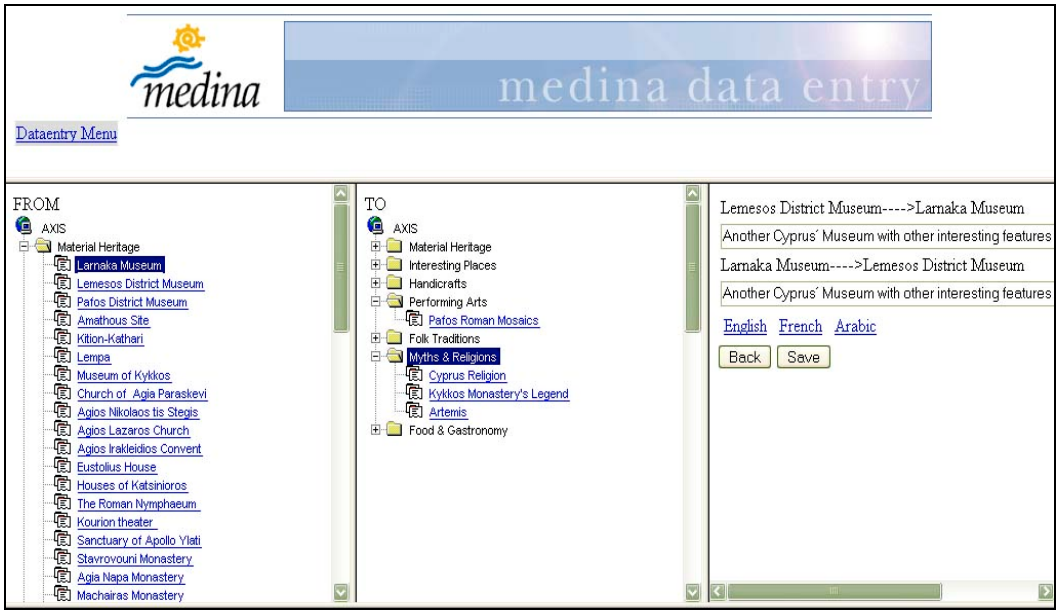

Figure 5c. Instantiator output for the MEDINA project: Customized Data Entry Form for Relevant Associations

4 CHEF Software Architecture

Figure 6 sketches the CHEF architecture that supports the functionality discussed in the previous section. Its main *logical* components comprise a Configuration Tool, a Static Compile, and a Dynamic Generator.

The *Configuration Tool* implements the functionality of the Customization Tool and creates the configuration parameters for the customized design schemas and data entry forms (stored in the *Application Profile* component).

The *Static Compiler* implements the Static Generation Tool. When the Static Generation Tool is invoked, the Static Compiler repeatedly calls the Dynamic Generator and *simulates* all possible page requests, storing the generated pages persistently (e.g., for off-line use).

The *Dynamic Generator* is made of a set of components (surrounded by grey background in the figure) that produce the web pages requested by the editorial manager during the instantiation activities, i.e., customized data entry forms or preview pages. The Dynamic Generator also builds dynamically the pages of the "final" application requested by end users, either on stationary or mobile devices. The architecture of the dynamic generation components exploits a well known approach in web engineering, separating the application business logic from its presentation and control logic; it is modelled according the *Model-View-Controller* (MVC) design pattern [21], which organizes application "objects" into three logical categories: Model objects, View objects, and Controller objects.

Model objects (collectively referred as "Model") represent application domain data and the business rules that govern access to and update of these data. *View* objects are responsible of rendering the contents of the Model and forwarding user commands to the Controller. *Controller* objects (collectively referred as "Controller") are responsible *to map user requests to operations* on

the Model, to execute them, to build the proper View, and to return to the client. In web applications, user "commands" appear as HTTP page requests. View objects typically correspond to HTML pages. Based on the page request, the results of the operations on the Model, and the Model state, the Controller generates the next HTML page.

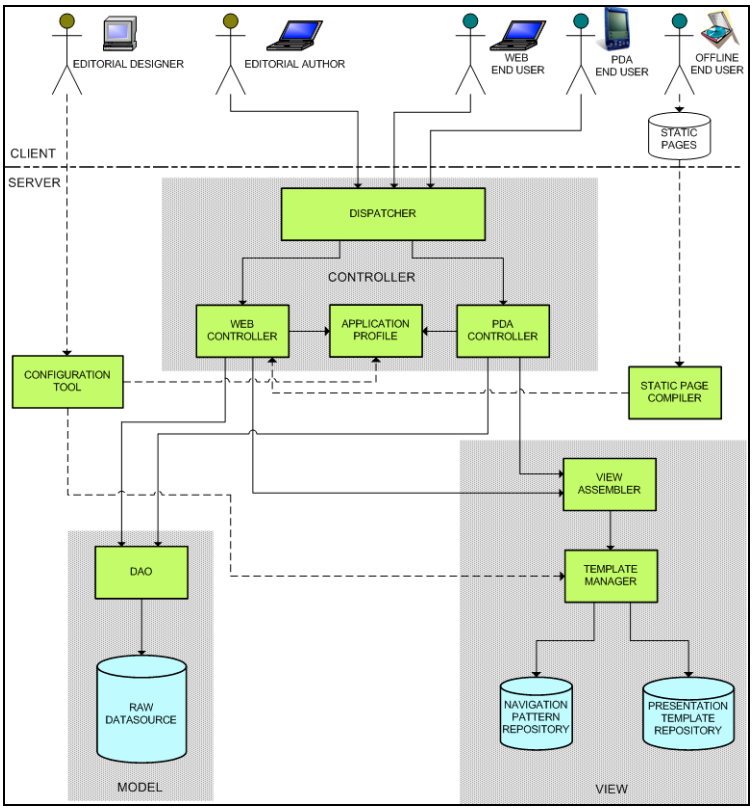

Figure 6. CHEF software architecture

The CHEF *Model* comprises the Raw Data Source (RDS) component and the Data Access Object (DAO). RDS is a data base storing the multimedia content pieces delivered by the application.

DAO stores the classes and objects that represent the definitions of information and navigation structures (schemas and instances), and provides the methods for two kinds of operations: *retrieval* of multimedia content pieces, from the underlying RDS, that must be presented in the next page to display; *validation* and *persistent storage* of the data entered by the framework user.To abstract from the implementation characteristics of the Model, DAO provides an interface that translates DAO operations into commands on the DBMS that implements the RDS. This approach allows different data bases to share the same Model, supports content reuse, and makes the development of different applications easier to implement, test, and maintain. The CHEF DAO is implemented using PHP [29] (the same open source implementation language used for the Controller components). It also exploits the Microsoft implementation [22] of ODBC, a standard

API specification that makes it possible to access relational data from any application, regardless of which DBMS is handling the data.

CHEF *View* corresponds to:

i) *HTML pages*, which are generated by the Controller and are viewed through the client browser;

ii) the Repository of *Navigation Patterns Specifications* and the *Repository of Presentation Templates* (the latter implemented as a repository of XTPL files [23] and associated CSS specifications [32]);

iii) The *Template manager*, which selects the appropriate Presentation Template and navigation pattern for the page to be displayed, based on the customization parameters of the current application and device; iv) The *View Assembler*, which is responsible of the composition of the HTML page to be displayed to the user.

The Assembler assembles the data retrieved by the DAO on the presentation template selected by the Template Manager, and composes the HTML page to return to the user. It uses XTemplate, a PHP compliant open source template engine [23] that creates HTML pages on the basis of the data sent by a PHP program, a page template (a XTML file) and a CSS.

The CHEF *Controller* includes:

i) The *Dispatcher*, which is responsible of forwarding user commands to the components that manage the business logics of a specific delivery device (desktop or handheld);

ii) The *Application Profiler*, which is responsible for storing the customization parameters of the application under development;

iii) A set of *Device-specific Controllers*, which store the business logics for the generation of device-specific pages. Based on the user command, the Controller identifies the proper "generation rules" (implemented in PHP) and launches their execution. The configuration data contain the parameters used by the generation rules to invoke the proper methods in the Model, and to identify the needed Presentation Templates and Navigation Patterns used by the Template Manager and View Assembler.

5 Evaluating CHEF

We carried on a systematic evaluation of CHEF within the MEDINA project, involving thirty-two domain experts from twelve different Mediterranean countries who used CHEF to produce the cultural tourism web sites of their countries. We used a mix of data collection techniques, including questionnaires, task-based user testing, and ethnographic studies, focusing on the evaluation of the following factors: *simplicity* and *efficiency* of the *development process*, *usability* of CHEF tools, and *overall satisfaction* of framework users.

Process simplicity was operazionalized in terms of two attributes:

- *process learnability*, measured in terms of *learning time* needed by domain experts to understand the overall CHEF design process and the different activities, and its *perceived difficulty*;
- *perceived difficulty* of *performing* the entire development process of a real medium size application.

Process efficiency was operazionalized in terms of person-hours needed, after the process and the framework functionality have been understood, to deliver a realistic application.

Usability was operazionalized in terms of various attributes [35]:

- *tool learnability*, i.e., the time needed by a domain expert having no experience in web technology, to become able to use the tool with a reasonable level of performance, after the development process has been understood
- *standard usability measures* such as performance, number of errors, number of help requests, and similar, during the execution of a set of assigned tasks.

Satisfaction was operazionalized in terms of global "appreciation" of the development "experience" by domain specialists, measured after the first version of the Medina Portal was completed (comprising 40% of the expected final product), and in terms of *prospective adoption*, i.e., declared intention of using the tool for future projects.

We measured *process learnability* and *framework usability* in the context of a 1.5 day tutorial/workshop carried on at our lab, and involving 17 tourism experts from different countries.

During the first day, our team explained the development process and engage participants in paper-based design of a preliminary tourism application concerning their country. They initially used the concepts of CHEF Cultural Model, and then were asked to make modifications and extensions to their drafts and apply the more general concepts of CHEF Meta Model. These activities lasted for the whole day. At the end, all participants had understood the CHEF design process and the activities involved. Thus we can estimate that the average time for *process learnability* is *one person/day* (involving training and practicing). This is apparently rather costly, but it is compensated by the advantages of the approach in terms of efficiency, as discussed later in this section and in "Conclusions". In addition, in spite of this relatively high learnability burden, 90% of the domain experts involved in the workshop perceived this overall learning process "reasonable simple" (in a five values scale – "very simple", "simple", "reasonably simple", "difficult", "very difficult"), finding the major difficulty in understanding the abstract concept of meta-modeling per se.

We investigated tool *usability* during the tutorial/workshop second day. We briefly explained how the CHEF framework works, for approximately 15 minutes, and then assigned to all participants a set of precise, small-sized tasks of increasing complexity (e.g., instantiate one Axis and one Relevant Association of the Cultural Model, create a Pathway, create a new Axis, create a new Relevant Association, define a different Navigation Pattern for a given Axis). The average *learning time* for learning the core functionality of the tool was 80 minutes. 87% of testers were able to complete all tasks in this span of time, with a minimum of 40 minute and a maximum of 90. After this hands-on session, they declared to feel comfortable with the tool and ready to start a true development process, working in groups (each one composed of 2-3 persons) and focusing on their National Web Sites for the rest of the day. During the first two hours of their work, an observer sat side by side the users and measured (without intervening in their activities) task performance, degree of completion, number of errors, and number of help requests. In 85% of the cases, users asked no help to observers. The percentage of "errors", in terms of "wrong" clicks, was quite low (in average, 14% of the overall set of interactions).

We measured *process efficiency*, *perceived difficulty of performing the entire development process*, and *satisfaction*, through a questionnaire submitted to all MEDINA national teams (32 domain experts overall) after they had completed the development of a preliminary version of their cultural tourism web application, which comprised at least 30 instances of 3 Axes, 50

instances of Relevant Associations, 2 pathways composed of a minimum of 10 "topics". The *average effort* amounted to 45-60 person-hours (in total) in 75% of the cases (i.e., in 6 of the 8 applications developed by the different teams). This time did not include the time needed to produce the cultural multimedia contents, as we asked participants to *reuse* those already available in the official web sites of their national tourism offices. It is interesting to observe that the development of such existing web sites had required from 6 person-months to 1.5 person-years, depending on the country; participants themselves were impressed by the limited time they invested to build their preliminary applications, and by their quality, which was judged "much nicer than the existing web sites using the same contents!" In terms of *satisfaction*, *87%* of the domain experts in our study declared to be "very satisfied" (in a 4 values scale in which "very satisfied" is the highest score) of the development experience. A high majority (76%) declared to have a "strong intention to continue using CHEF in future projects".

During the development period, we visited the local teams of five countries and carried on an ethnographic study aimed at collecting qualitative data on the development process in the actual contexts of work. The most interesting result of this study concerns the use of the features related to CHEF Cultural Model vs. the use of more advanced design capability of the framework. Initially, domain experts tended to reuse build-in constructs: they instantiated cultural Axes, Relevant Associations, Patterns and Layouts of CHEF cultural model, filling them with contents without introducing new design elements. As the work proceeded, domain experts discovered the need to include new classes of contents (and thus to define new Axes) and to introduce new Pathways that integrate topics from different Axes. In some cases they changed the built-in navigation patterns associated to Pathways in the CHEF Cultural Model, for example using Guided Tour only for short pathways, and Index+Guided Tour pattern for long ones. These observations highlight that for the purposes of simple applications, or in the initial phase of the development process, CHEF might provide more functionality than it is needed. As the amount of content to be included in an application grows, and domain experts gain more experience in design, new requirements are identified and advanced customization features become useful. This may suggest us to deliver the framework in two versions, a simple one, offering the core features and based on by the CHEF Cultural Model, and an advanced, more flexible, full-sized version, including all sophisticated taylorability functionality.

During the end of the project we carried on a final study to investigate the quality of the applications developed with CHEF by MEDINA partners. The quality of the artifacts created with the tool depends from a number of variables, including not only on the capability offered by the system but also the design skills of framework users (domain experts) and the amount of effort invested in content creation. Thus this indirect form of evaluation of the framework may be biased by a number of confounding factors which we can partially control for. We carried on an empirical assessment of 4 web sites included in the MEDINA portal that were comparable in terms of designer's skills, quantity and quality of multimedia contents. To investigate usability and end user appreciation of these applications, a survey was carried on involving 84 users (21 per application) aged 20-40, who had high-medium interest for cultural tourism and can be regarded as representative of the main target of these applications. A questionnaire was submitted after a free use of the web site for approximately 30 minutes. Students of our HCI and Web Design Master courses were responsible for recruiting users and collecting and analyzing data. They were motivated in performing this job to the best of their capabilities, as it counted towards their grade in the courses. The results of the survey were positive. 89% of respondents judged the application "very good" or "good". In terms of usability, 90% found the application "very easy to use", 87% judged contents "well organized" and "easy to search", and 89% considered its navigation "intuitive" and "predictable".

6 Related Work

The idea of exploiting the concept of application framework for hypermedia dates back to the 80ies, with the pioneer work of Meyrowitz's Intermedia [33], and it has been later explored by many other research and development teams. Most of existing data intensive web frameworks built in academic or industry environments [30][37][40][41] exploit a model based approach. In some cases, they are conceived primarily as design tools (e.g., [37][41]). In other cases (e.g., [30]), they support the translation of conceptual designs of data-intensive web applications into effective software components. Still, these frameworks do not address a specific business domain, thus cannot be regarded as true *enterprise* frameworks, and do not adopt a EUD philosophy, being rather conceived as *tools for expert programmers*.

The many commercial Web Content Management Systems (WCMSs) created to support the development of data intensive web applications (e.g. [24]) can be customized to specific business sectors and can be configured to meet the requirements of a specific domain and so become enterprise frameworks. WCMSs are powerful tools in terms of functionality for distributed multimedia information management and web publishing. Still, they are not model-based, nor they explicitly support design activities and design reuse. From an end user perspective, the main different between commercial WCMSs and frameworks like CHEF relays in the degree of EUD support, and hence in the complexity for tool users. In CHEF, customization is performed by domain experts at design time, and requires a limited training (as discussed in the previous section). A WCMS requires advanced programming expertise for configuration and to put the customized system in the work place. It is only at this point that domain experts come into play, mainly using the system for data entry tasks, with limited opportunity for design extensions and changes.

Attempts to embrace a EUD approach in the web development context are limited, in spite of the increasing attention paid to this approach in academia and industry [42]. [36] provides a useful survey of existing web development tools in the research and market arena, pinpointing that they are mainly designed for ICT professionals but are less appropriate for users without programming expertise. These authors also empirically investigate the requirements of web development from and end user development perspective. These results informed the design of a prototype tool named Click. Click is a very preliminary proof-of-concept system that supports generic web application building; it is not domain specific nor has framework-like characteristics.

CBEADS [18][19] is a toolkit that supports the EUD paradigm and can be used to rapidly develop and change web applications in response to changing needs. It has some methodological similarities with CHEF, but also a number of significant differences. CBEADS an enterprise framework for developing *stationary* (i.e., non multichannel) web based applications that support business processes. CBEADS regards the application development process as an instantiation of a meta-model, the modelling features of which consider functions and views to be provided to different categories of business application. In this respect, CBEADS developers have built a design space in the Meta Design sense [9][10][11]. Still, CBEADS does not fully embrace the Meta Design philosophy in that its intended end users - business analysts - are users of *both* the framework *and* the final applications, i.e., use the framework for building applications for themselves. CHEF targets *data intensive* web *multichannel* applications, thus it puts more emphasis on content design and navigation, and on multiplatform delivery. The domains of applications built with CHEF are tourism and cultural heritage, and framework users are domain experts who design applications for others (e.g., tourists).

Existing environments developed in the EUD community do not adopt the concept of framework (at least, not explicitly), although some of them can be regarded as exemplars of this class of systems. TERESA [2][34], for example, can be viewed as a framework for multi-device user interfaces in a generic domain. TERESA is an authoring environment "for developing transformation-based ubiquitous interfaces, providing semiautomatic support … to generate suitable implementations for various platforms" [34]. Like CHEF, TERESA aims at enabling non computer expert designers to focus on the relevant logical aspects of interactive applications, rather than low level implementation details. It adopts a model based approach, distinguishing among multiple levels of abstraction in multi-device interface design, but it does not consider the design requirements of interfaces for specific business sectors, nor addresses issues related to information and navigation design.

In the domain of cultural heritage hypermedia, a system that can be regarded as an enterprise framework that implements the EUD paradigm is PACHIDERM [28][38]. PACHIDERM was built by a US consortium lead by NMC and San Francisco Museum of Modern Art and involving a number of museums and universities. It is an authoring and publishing tool for web based museum hypermedia that provides a set of high-quality built-in "educational-oriented" templates, which domain experts can fill with multimedia contents and links. Still, PACHIDERM supports a page-by-page development process only; it is not model based nor does it promote a Meta Design development process based on design customization and design schemas instantiation at different levels of abstractions.

CHEF is largely built upon the experience gained with a previous, much simpler WEF, developed in the first phase of the MEDINA project [16][17], and extends the original framework in two main directions: i) by introducing more systematically a meta-design approach, thus supporting the design of more flexible and general web structures, and ii) by addressing multi-device application delivery.

7 Conclusions

The main contribution of this paper is to promote the combined adoption of the End User Development and Model Based paradigms to enterprise frameworks for data intensive web applications (WEFs), discussing the implications of this approach in terms of design requirements for this class of tools, and benefits that can be achieved.

Any framework holds a *reusability* property that can yield substantial improvements in the productivity of a development team, and can enhance the quality, performance, and reliability of the final product [7][8][17][41]. In addition, a framework based approach enables the development of applications that share a common design from the user interface level to the implementation level, ensures their technical *interoperability* and enforces the harmonization of the User Experience (UX) in different web sites built using the same framework. This characteristic is crucial, for example, for organizations who must build a *set* of coordinated content intensive web applications (e.g., within a web portal) and need to provide a "coordinated image" and a coherent UX across all applications in terms of information architecture, navigation, interaction and navigation.

A Model Based approach may increase the conceptual complexity of the development process, since it requires abstract thinking at multiple levels both by framework developers and framework end users. Still, we believe it is a fundamental pre-condition for a systematic framework implementation. From the perspective of framework users, it fosters a design-by reuse method of work, in which proved design solutions can be used over and over again, and yields to benefits in terms of organization of the development process and, ultimately, usability of the final application.

Typically, an enterprise framework involves domain professionals only as *informants* of the framework design, but not as framework *end users*. Because of the limited ICT expertise, the use of the framework by these stakeholders is typically mediated by programmers or restricted to data entry. Still, the adoption of the EUD paradigm, with a strong focus on domain experts, seems to be a natural evolution of enterprise frameworks in non technological business sectors. Domain experts are the true "owners of problems", and should become protagonists of the whole development process of their web applications, breaking them free from the dependency on programmers or the need of understanding software architectures, databases, and programming languages [9].

EUD enterprise frameworks are not only productivity tools but also a means for supporting *appropriation processes* and *sustainability*, leading to more economical ICT investments [44]. By introducing a EUD WEF in an organization, in–house teams of domain experts can not only tailor web applications to their needs, but also maintain them along the time and make them evolve, at a sound cost, avoiding the need for hiring programmers or for expensive outsourcing.

In an arena where the web has become the main channel to communicate with "customers", to inform them, to promote a brand in the global society, the market for frameworks that exploit the above characteristics is potentially huge. Consider for example the vast universe of *small organizations*, like cultural institutions, schools, charity organizations, or little non ICT companies, which may want to exploit the web for communication, promotion, or educational purposes, but are bound to limited in-house technological know how and scarce financial resources. Rather than spending on implementation, they can empower domain experts and enable them to invest on activities such as definition, reuse, and adaptation of good conceptual design solutions, design of the proper "message", and creation of high quality contents, which contributes to create "added-value" in the final products.

The paper has presented a system, called CHEF, which implements this vision and exemplifies it. CHEF provides a tested *user-friendly, easy-to-use, visual* environment where also inexperienced developers, and domain experts in particular, can *design "by reuse"* their applications, *instantiate* the design schemas with the proper multimedia contents, and *generate* high quality data intensive web applications for different delivery platforms, both stationary and mobile. We uses CHEF in the MEDINA project and in a number of other projects in the tourism and culture domain, e.g., for creating web sites for museums or art exhibitions[b], and could empirically prove the benefits that can be achieved by combining the general characteristics of any enterprise framework with the advantages derived by the paradigms of End User Development and Model Based Development.

CHEF paved the ground for a number of ongoing and future activities. We are applying the concepts illustrated in the paper to other application domains, building new frameworks that adapt CHEF implementation architecture and meta-model to the characteristic of other application contexts (e.g., e-learning). We are also carrying on a number of long term studies to measure more precisely the economic aspects of model based EUD WEDs, and to compare costs, complexity, and organizational impact of this kind of systems against other development tools (e.g., WCMSs.)

Acknowledgments

The author is grateful to all partners of MEDINA project, Luca Megale, and the whole development team at HOC Lab.

[1] References to these web sites can be found in [27]

References

1. Atzeni P., Mecca G., Merialdo P. , Design and Implementation of Data-Intensive Web Sites. Proc. Conference On Extended Database Technology (EDBT) 1998, 436-450, 1998
2. Berti S., Paterno' F., Santoro C., Natural Development of Ubiquitous Interfaces. Comm. of the ACM 47 (49), 2004, 47-52,
3. Birrer E. T. , Frameworks in the financial engineering domain: An experience report. Proc. European Conference on Object-Oriented Programming (ECOOP) 1993, Springer LNCS 707/1993, 21-35
4. Bolchini D., Paolini P., Interaction Dialogue Model: A Design Technique for Multichannel Applications. IEEE Trans. Multimedia, 8 (3), 529-541, IEEE 2006
5. Ceri S., Fraternali P., Bongio A., Brambilla M., Comai S., Matera M., Designing Data-Intensive Web Applications**,** Morgan-Kaufmann, 2002
6. Codenie W, De Hondt K, Steyaert P, Vercammen A. From Custom Applications to Domain-Specific Frameworks. Comm. of the ACM , 40 (10), 1997, 70-77
7. Fayad M.E., Schmidt D.C. , Object Oriented Application Frameworks. Comm. of the ACM , 40 (10) 1997, 32-38
8. Fayad M. E., Introduction to the computing surveys' electronic symposium on object-oriented application frameworks. ACM Computing Surveys, 32 (1), 2000, 2-9
9. Fisher G., Giaccardi E., Ye Y., Sutcliffe A.G., Mehandjiev N., Meta-design: A Manifesto for End-User Development. Comm. of the ACM, 47(49), 2004, 33-37
10. Fischer,G., Giaccardi, E., Meta-design: A Framework for the Future of End User Development. End User Development, H. Lieberman et al. (eds), Kluwer Academic 2006, 427-457
11. Fischer G., Nakakoji K., Yunwen Y., Metadesign: Guidelines for Supporting Domain Experts in Software Development. IEEE Software, 26 (5) 2009, 37-44
12. Gamma, E, Helm, R., Johnson, R. and Vlissides, J. Design Patterns: Elements of Reusable Software Architecture. Addison-Wesley, 1995
13. Garzotto F., Paolini P., Schwabe D., HDM - A Model-Based Approach to Hypertext Application Design. ACM Trans. on Information Systems , 11 (1) 1993, 1-26
14. Garzotto F., Paolini P., Mainetti L., Hypermedia Design, Analysis, and Evaluation Issues. Comm. of the ACM , 38 (8) 1995, 74-86
15. Garzotto F., Paolini P., Bolchini D., Valenti S. "Modeling by patterns" of Web Applications ProcWWWCM'99. LNCS 1727/1999, Springer 1999, 293-306
16. Garzotto F., Megale L. Towards Enterprise Frameworks for Networked Hypermedia: a Case-Study in Cultural Tourism". Proc. ACM Hypertext'05, ACM 2005, 257 - 266
17. Garzotto F. A User-friendly Enterprise Framework for Data Intensive Web Applications. Proc. International Conf. on Information Reuse and Integration (IRI) 2005. IEEE 2005, 415-420
18. Ginige A., Liang X., Marmaridis M., Ginige A., De Silva B. Smart Tools to Support Meta-design Paradigm for Developing Web Based Business Applications. Proc. Web Engineering 2007, LNCS 4607/2007, Springer 2007, 521-525
19. Ginige A., De Silva B., CBEADS: A Framework to support Meta-Design Paradigm". Stephanis C. (Ed.), Universal Access to HCI, Part I, HCII 2007, LNCS 4554/2007, Spirnger 2007, 107-116
20. http://hoc.elet.polimi.it/medina/home.html
21. http://java.sun.com/blueprints/patterns/MVC-detailed.html
22. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbc_part_1.asp
23. http://sourceforge.net/projects/xtpl/
24. http://www.cmsmatrix.org/

25. http://www.medinaportal.net/
26. http://www.medinaproject.net
27. http://hoc.elet.polimi.it
28. http://www.pachiderm.org.index.html
29. http://www.php.net/
30. http://www.webratio.com
31. Johnson R.E., "Frameworks= Components + Patterns". Comm. of the ACM , 40 (10), 1997, 39-42
32. Meyer E.A., Cascading Stylesheets - The Definitive guide, 2° ed, O'Relly Ed., 2004
33. Meyrowitz N., Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. Proc. OOPSALA'86, ACM 1986, 186-201
34. Mori G, Paterno' F., Santoro C., Design and Development of Multidevice User Interface through Multiple Logical Descriptions. IEEE Trans. on Software Engineering 30 (8), 2004, 507-520
35. Nielsen. J., Designing Web Usability: The Practice of Simplicity, New Riders Publishing, 2000
36. Rode J., Beth Rosson M., Perez Quinones M.A., End User Development of Web Applications. Lieberman H. (Ed.), End User Development, 161-182, Springer 2006
37. Rossi G., Garrido A., Schwabe D., Navigating Between Objects: Lessons from an Object-Oriented Framework Perspective. ACM Computing Surveys, 32 (1), ACM 2000, article 30
38. Samis P., Making Sense of Modern Art at five. Proc. Museums and the Web 2004, Archives and Museums Informatics 2004, 20-28
39. Schmid, H.A., Systematic Framework Design. Comm. of the ACM , 40 (10), 1997, 48-51
40. Schwabe D, Rossi G., An Object Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems, 4 (4), J. Wiley, 1998, 207-225
41. Schwabe, D., Rossi G., Emeraldo L., Lyardet F., Web Design Frameworks: An approach to improve reuse in Web Applications. Proc. WWW99 Web Engineering Workshop, LNCS 2016/2001, Springer 2001, 335-352,
42. Sutcliffe A.G., Mehandjiev N. End User Development: Introduction to the Special Issue. Comm. of the ACM, 47 (49), 2004, 31-32
43. Turau V. , A framework for automatic generation of web-based data entry applications based on XML. Proc.SAC'02 - Symposium on Applied computing , ACM 2002, 1121-1126
44. Volker W. Jarke M. , The economics of end user development. Comm. of the ACM 47 (49), 2004, 41-42