Journal of Web Engineering, Vol. 11, No. 4 (2012) 290-316

WEBFDM: A SITUATIONAL METHOD FOR THE DEVELOPMENT OF WEB APPLICATIONS

ADELAIDE BIANCHINI, ASCANDER SUÁREZ, CARLOS A. PÉREZ D.

Universidad Simón Bolívar, Caracas

abianc@usb.ve, suarez@usb.ve, caperez@usb.ve

Received August 10, 2011
Revised June 28, 2012

Several methodologies have been proposed to improve the quality of Web application development in the last decade. Some proposals provide techniques and mechanisms to specify the product model; others are focused on process development models. However, few approaches have suggested methods adapted to different situations and development circumstances. Besides, some industrial and academic methods are not flexible enough to react according to the different situations and projects conditions to be developed. These conditions may include application type and complexity, models to be created, development team characteristics, technological resources among others. This paper presents WEBFDM, a method grounded on Situational Method Engineering principles for the development of web applications and a CASE Tool – COHESIÓN. The KANON framework, used to characterize Web development situations, is also described.

Key words: Web application development, Web Engineering, Situational Method Engineering
Communicated by: B. White & M. Bielikova

1 Introduction and motivation

The complexity of Web applications has grown significantly from static systems (Web sites) oriented to information content dissemination to dynamic systems like online transactions, e-banking, e-commerce applications, etc. Several approaches have been proposed over the past decade to improve the quality of Web application development. Among the best known OOWS [39], OO-H [10] [19], WebML [11], WSDM [12] and UWE [25] may be mentioned. These methodologies have promoted the growth of Web Engineering, a discipline that provides a systematic and disciplined approach for developing, documenting and maintaining Web applications.

Web Engineering is more complex than traditional Software Engineering since it raises new issues such as rich interface design (for presentation and interaction), hypertext modelling support and content modelling. Methods and models from Software Engineering are not expressive enough for the specific characteristics of Web applications [34]. Also, Web application development is characterized by [42]: i) small and multidisciplinary development teams, ii) many small software deployments (iterative and incremental process); iii) several stakeholders from diverse fields; iv) informal requirement specifications in multiples cases; and v) short deadlines to delivery of the final product.

There is common agreement about the steps or phases that methods follow in Web Engineering because all of them establish a typical process from requirements elicitation to implementation. Besides, the use of different techniques and procedures are proposed for each phase [40].

An agreement about the products generated and the process to be followed exists; the latter prescribes complex tasks rather than broad guidelines [36], [37]. For example, a very exhaustive requirement elicitation phase may be unnecessary in small-sized Web applications or with low complexity levels. In practice, most methods can be considered generic since they are used in any Web application development, different domains and in diverse situations. However, a rigid or generic method does not fit well every Web application domain [40]. Some proposals disregard aspects about process development models for shaping the development and product generation based on different situations. "A situation is the combination of circumstances at a given moment, possibly in a given organization" [21, p. 25]. A situation affects the way of working and product types to be delivered. It can be a combination of different domains, specific type of Web application, complexity level of the application, capacities of the working team, times and costs related to the development, and other attributes. In each new development, the designers have to adapt or extend, in some way, the method to support the new Web project [40].

The discipline to build project-specific method based on a situation is called Situational Method Engineering (SME). It proposes strategies to select methods, called method fragments or method chunks (from a repository), to create a new method based on characteristics of the development circumstances [8].

There is interest in the following research issues associated on the above observations: is it possible to adapt a Web application development methodology based on different situations using the principles of Situational Method Engineering and the foundations of Web Engineering? Which indicators or attributes are adequate to characterize a Web application development?

The basis of this research is that Web applications development requires flexible approaches that allow developers to configure the process and to generate the product models according to a set of attributes about the project characteristics and situation. In addition, flexible approaches have to guide developers in the creation of models and artefacts needed to specify the Web application to be implemented.

A situational methodology for the development of Web applications has been designed and developed – WEBFDM - to achieve a flexible approach [4]. The methodology has a CASE tool, called COHESION [5], which brings full support in the use of WEBFDM. Also the KANON framework, used to identify the characteristics of Web application development situations, is proposed.

This paper is organized as follows: in section 2, preliminary definitions necessary to describe the proposal are introduced. Related works are presented in section 3. An overview of the proposal is described in section 4. Finally, conclusions and future works are presented in section 5.

2 Preliminary definitions

There are numerous definitions of the term Web application. Some of them are about content centric Web sites and others about data centric applications or transactional Web applications. In this paper we will use the following definition of a Web application: "a software system based on technologies and

standards of the W3C that provides Web specific resources such as content and services through a user interface and delivery over the WWW." [24, p. 2]. Following, there are some definitions needed to introduce our approach.

2.1 Web application categories

Web applications are used in traditional fields. Advances in technology launch diverse new fields of use: in business, in information retrieval, learning, collaborative work, among others. Based on [24] Web applications are categorized in: document centric Web sites, interactive Web applications, transactional Web applications, workflow based Web applications, collaborative Web applications, social Web applications, portal-oriented Web applications, ubiquitous applications and semantic Web applications.

Depending on the Web application category, different resources and techniques are required for its development. Furthermore, Web application categories have intrinsic complexity levels. Whenever a new advanced category shows up, this development implies the evolution of functionality, new technology use and team experience, among others.

2.2 Web Engineering framework

The existing methods in Web Engineering follow the well-known framework proposed by [18] and [33] to model Web applications at different abstraction concepts. The framework also considers customization in all dimensions. The elements proposed by the framework are summarized in Table 1.

TABLE 1. MODELING DIMENSIONS FOR WEB APPLICATION. ADAPTED FROM [18] AND [33]

DIMENSIONS	DESCRIPTION AND DETAILS
Levels	• **Content**: represents the information structure and application logic underneath the Web application. • **Hypertext**: represents the structuring of the content into nodes and links between nodes. • **Presentation**: the user interfaces and interaction style.
Aspects	Aspects refer to the levels of the application logic **structure** as well as application logic **behavior**. The relevance of the structure and behavior models depends on the type of Web application to implement.
Phases	Concept about a general modelling process approach for the development of Web applications. In general, the methodologies most mature propose the following phases: requirements elicitation and analysis, conceptual design, implementation, testing and validation, deployment, etc.
Customization	• It considers the context, e.g., users' preferences, device characteristics, or bandwidth restrictions, and allows adapting the Web application accordingly. • It influences all three Web modelling dimensions of content, hypertext and presentation with respect to structure and behavior. • All phases of the development process should take into account adaptation (customization), and evolution: in all levels, all aspects and during all phases.

The framework represents different concerns to be considered during Web development process in a comprehensible manner.

2.3 Situational Method Engineering

As stated before, Situational Method Engineering (SME) is the discipline to build a project-specific method based on a given situation. It proposes strategies to select method building-blocks (from a repository), to create a new method based on characteristics of the development circumstances [8]. The most important advantages of applying SME are the following: i) building or creating a method base that allows a method definition in a specific domain, reusing validated building-blocks; ii) adapting, in a simple way, project specific based methods considering the constraints and project characteristics [39]; iii) providing techniques based on the reuse of existing building-blocks in the construction of new methods, more flexible and better adapted to the development situation [30].

The SME is not oriented to acquire ready-to-use-methods from a supplier, but it is focused on the in-house construction for a specific organization or a specific methodological approach to carry out a project.

In the reviewed literature, different terms are used to denote the method building-blocks that are the basis of situational method creation approaches. The term method fragment is used to comprise a product (models, artefacts) perspective and a process perspective (activities, techniques). It is also known as method chunk and method component. In this paper we follow the concept method fragment, which is characterized by a product perspective (includes artefacts and models) and a process perspective (includes activities, procedure and roles). With regard to the method construction, the approaches found in the literature and most used by researchers in the SME discipline are summarized in table 2.

TABLE 2. APPROACHES IN SITUATIONAL METHOD ENGINEERING, ADAPTED FROM [22].

APPROACH	DESCRIPTION
Assembly-based	A new method is constructed from fragments of existing methods. Fragments can be categorized into product and process fragments. It follows the reuse strategy.
Extension-based	The required method is created from an existing method, and extends through patterns to complement or fill gaps in the existing method. The SME defines strategies and guidelines to carry out the extension.
Paradigm-based	It is grounded on the idea that the new method can be obtained either by abstracting from an existing method or by instantiating a meta-model. It is also called evolution-based approach.

Situational Method Engineering approaches differ in the creation process in several aspects. The most notably according to [3] are:

a. The point of departure that indicates what kind of source a SME approach is using to create the new method. Some points of departure are: a portfolio of methods, a portfolio of fragments from different methods, and a single method called base method.

b. The granularity of method fragments used to create the new method. A method fragment can reside on diverse granularity layers [9]: i) method, i.e. the complete method; ii) stage or phase, which addresses a life-cycle segment; iii) model, which represents an abstract aspect of the system, like the Data Model; iv) Diagram used to represent a view of a model; v) "concept, which addresses the concepts and associations of the method fragment on the Diagram level, for example an Entity" [9, p. 384]. Granularity influences the complexity and flexibility of the new method created: if the granularity of the method fragments is high, the flexibility in producing new methods is low.

The creation of a new method must meet quality requirements [9, p.391, 392] like:

a. Completeness: the new method contains all the method fragments that are referred to by other fragments in the method.

b. Consistency that includes among others: i) precedence consistency, requiring that fragments be placed in the right order in the new method; ii) granularity consistency that imposes that the granularity layer of related fragments to be similar.

An interesting state of the art review about SME approaches and terminology is provided by [22].

2.4 Web Engineering meets Situational Method Engineering

A Web application development methodology may include different methods to address the development. Furthermore, the target of each method is related to the concerns explained in the Web engineering Framework proposed by [18] and [33]. Therefore, a method is responsible for the content model (structure) during the design phase, and another method is responsible for the hypertext structure, behavior, and so on.

Considering the principles of Situational Method Engineering, a Web application development methodology can be viewed as a collection of method fragments needed to carry out the development, where different method fragments are required based on the characteristics of the Web application to be developed. This interpretation is shown in figure 1, where a single box represents the method fragment used in a level element considering the aspect to be modelled. For instance, the fragment method used to model the content structure, or the method fragment used to model the Hypertext Structure in a Web application.

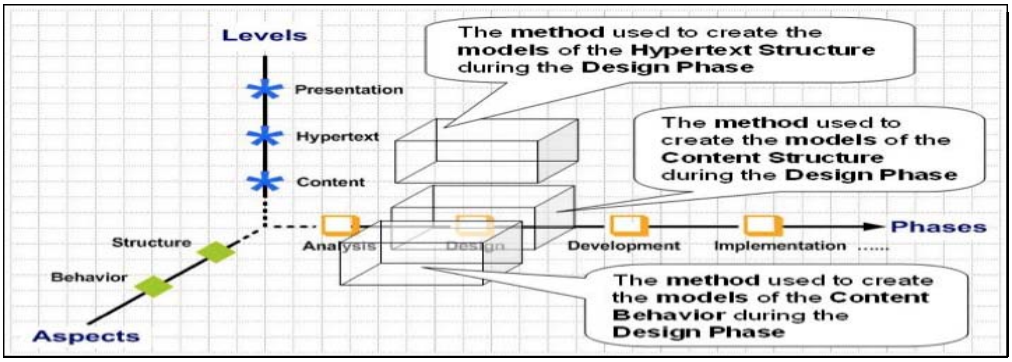

Figure 1. Modeling dimensions for Web application based on Situational Method Engineering.

The research presented in this article takes levels and aspects into account as the most appropriate concerns for adaptation through a situational method because these concepts are inherently related to the product characteristics and the development situation. However, some adaptation can be carried out in the development phases.

3 Related works

Upon reviewing the literature, we observed that few proposals considered the possibilities of adaptation based on a development situation and/or process' flexibility. In this section related works in adaptation and flexible processes, the use of SME principles in Web application development, and approaches developed for characterizing Web applications are presented.

3.1 About adaptation process and flexibility in Web application development

According to [20], different projects often demand different degrees of formality and accuracy based on their quality requirements, functionality and complexity. In the mentioned research two criteria among others, which assess process flexibility, are defined: (i) Does the process allow techniques selection for each task? (ii) Does the process allow the selection of activities, tasks and techniques to be used in a project, depending on the specific characteristics of the project?

A meta-process that uses and observes some characteristics about the Web project to be developed is proposed in [15]. It also promotes the transition from an agile method to a disciplined one.

According to [13] methods have to rely on empirical and flexible process models that can be easily adapted to fit process requirements and to be put in practice without disturbing the project goals or compromising its success. It is also stated "… that Web projects require such flexible or empirical process models that allow developers for shaping the development life cycle according to their changing needs and the process model is even more important in industrial settings than the model expressiveness" [13, p. 1376]. In the work the ADM method is described. It proposes no dependencies amongst each phase, and can be sequenced as required by the development situation. The basis of the flexible process is the fact that in each ADM phase the elaboration of mandatory, recommended or optional models is suggested.

3.2 Using Situational Method Engineering in Web development and the characterization of Web application development

Some works have been carried out using Situational Method Engineering in conjunction with Web Engineering approaches to improve Web applications development. In [38] and [41] it is stated that Web application development methods do not cover the specific needs of a method for Web Content Management implementations and present WEM (Web Engineering Method) as an assembly-based Situational Method Engineering approach applied to develop a new design method for the CMS domain.

The OOWS method meta-model is defined with the purpose of applying Situational Method Engineering in order to improve the development of CMS (Content Management System), because the

OOWS method "lacks expressiveness to define Web Applications in some domains as CMS" [40, p. 227]. They apply the same approach of WEM to OOWS.

The Method Association approach proposed by [29] selects and constructs methods from five model-driven Web modelling approaches to fit the project domain.

In accordance with [28] the most appropriate approach for Web application development is a construction of an organization-specific base method with the use of reusable method fragments (components) and the adaptation of the base method, in order to support specific project characteristics. Besides, a Method Engineering framework that includes a process for method construction and a repository for methods, method components, configurations, rules and development situations characteristics storage is proposed.

In [15] the principles of Situational Method are not used. Instead, some indicators are listed to characterize a situation: user number of the application, member number of the teamwork, documentation costs, among others. These indicators are used to measure and identify the development complexity.

In [3] an industrial experience in Slovenia shows a practice-driven approach called Process Configuration. This approach serves to create an effective tailoring approach to produce project-specific methods from existing ones, taking into account project circumstances and situations.

A collection of factors to characterize Web applications and the construction of new methods based on existing method fragments, grounded on Situational Method Engineering is proposed in [28]. Three types of guidance for developers are also provided: (i) the selection of the most appropriate design process-model; (ii) the selection of the most appropriate method fragments, and (iii) the application of selected method fragments.

In [27], it is stated that the method fragments selection is a fundamental process in the creation of new methods. A typology of software engineering projects characteristics and four dimensions are considered as a key contribution: organization (project importance, impact, size, innovation level); human (conflict resistance, experience, clarity, stakeholder number among others); application domain (accuracy of formalism, relationships, integration with other systems, complexity, and application type) and development strategy (project organization, development strategies, goals number, etc.).

In the approach described in [36] the factors identified to characterize a Web application intended are: application type, complexity, similarity with other applications, problem clarification and designer experience. In a study about practical applications of Web Engineering conducted by [16], there is an interesting proposal to categorize the complexity of Web applications through requirements elicitation.

Another approach used to characterize Web applications is described in [24]. The proposal is based on the ISO/IEC 9126-1 standard used for the evaluation of software quality characteristics. This standard considers three dimensions: product, usage and development with their corresponding attributes. The authors also include the Web application Evolution as a characteristic that governs all three dimensions. In [2] the same approach is used, although to identify the risks in Web projects development.

4 Proposal description

All mature methods in Web Engineering provide techniques and mechanisms to specify product models including content, presentation (interface and interaction) and hypertext. However, some aspects about process and product development models for shaping the development process according to different situations are not taken into account. One reason associated with the flexibility of a method is the possibility of adaptation to different development situations. "In Web application development it is impossible to adhere to a rigid, predefined project plan. It is vital to react flexibly to changing conditions" [24, p. 16]. In the following sections the core elements of the proposal are described: WEBFDM, a methodology to design and develop Web applications; the COHESIÓN CASE Tool used to support WebFDM and the KANON Framework used to characterize a Web application development situation in order to obtain a situational Web engineering method.

4.1 WEBFDM and COHESIÓN CASE Tool

The fundamental aspect related to this proposal is WEBFDM, -Web application Flexible Development Methodology- based on the MVC architectural pattern [6]. In this pattern, the Controller can be represented by a graph, which we call "MVC-Graph". The MVC pattern usage favours the separation of concerns in Web application development mainly based on the framework described in section 2.2. The team designer can distribute development efforts to some extent so that, if the implementation changes in one part of the Web application, then changes are not required in other parts.

The COHESIÓN CASE Tool [5] supports the WEBFDM methodology. COHESIÓN is a Web application developed with both WEBFDM and COHESIÓN itself. The generic development process in the methodology consists of the following phases (Figure 2):

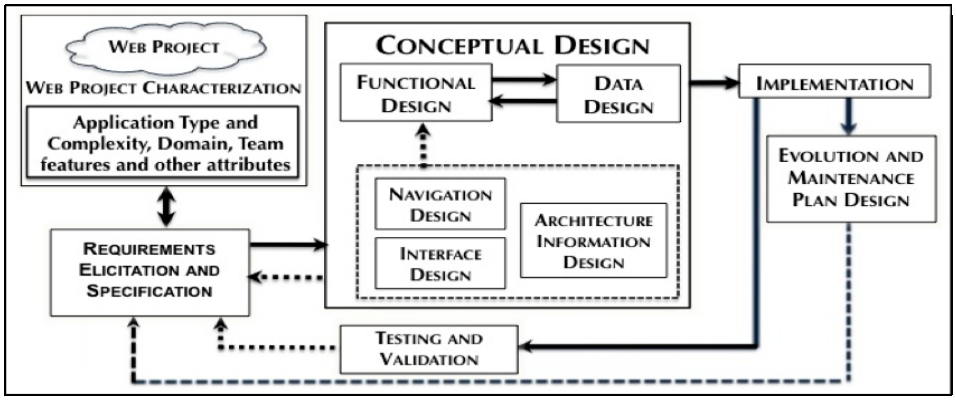

Figure 2. The generic development process in WebFDM

- Web Project Characterization. This phase includes the characterization of the Web application and the development situation. The situation is refined through the requirements.
- Requirements Elicitation and Specification. The functional requirements considered in WEBFDM are classified, using the approach proposed by [16] as: Data requirements (leading to the design of a representation model, i.e. a collection of persistent and interrelated ADTs), Interface

requirements, Navigation requirements, Transaction requirements (leading to the design of the logical organizations of the Web application in Use Cases), and Customization requirements (defining a set of actors and their corresponding transactions and functionalities).

- Conceptual Design. It includes Data Design, Functional Design (to accomplish the transaction requirements), Navigation Design, Interface Design and Information Architecture Design.
- Implementation.
- Testing and Validation.
- Evolution and Maintenance Plan Design.

The life-cycle model specifies the process sequence and products involved in the Web application development. WEBFDM follows the iterative and incremental process model, so in each increment the designer builds or enhances the products created. The current design is tested and the achieved products can be submitted to modifications or extensions.

The COHESIÓN CASE Tool follows the Model-Driven Development approach: it uses models and transformations of models in different phases of the development. There are two kinds of transformations in COHESIÓN [30] implemented by the following elements:

- The CRUD modeller, which is an endogenous, horizontal transformation which purpose is to automatically create all standard operations relating to an entity, i.e. Create, Read, Update and Delete;
- The Model Compiler Engine is a model transformer developed with the idea that the target can be defined (at least implicitly) using a meta-model. It is a template-based, exogenous, vertical, model-to-code transformation, which works hand-to-hand with a description language, CTDL (COHESIÓN Target Description Language). This allows the designer to describe a target's meta-model and generate the necessary code. This engine has three components: a set of templates, a description file relating each template to resources in the target, and a support class which allows the engine to accomplish the necessary transformations in order to construct valid implementations on the target. With this engine, four source code generators are implemented: the Web development frameworks Struts, Struts 2 (both under the Java programming language) [1], Django (under Python) [14] and a Portlet generator (powered by Struts). Another source code generator with the Symfony framework (in PHP) [35] as a target is under development.

The COHESIÓN CASE Tool is concurrent: different users can have access and modify the same design project.

4.2 Models proposed by WEBFDM (and supported by the COHESIÓN CASE Tool)

The models proposed by WEBFDM, and supported by COHESIÓN, are the following:

- Common models: Content (Data model representation), hypertext (navigation model), Presentation (interface design model).
- Feature models: Requirements analysis model (Use Case Model), Adaptation/Customization model (Actor identification based on roles), Services/Task model (Behavior Diagrams representing the CIM – Computer independent model)) and the Implementation model (PIM – platform independent model and PSM-platform specific model).

During the Conceptual design phase, the designer using COHESIÓN builds the common and feature models collection of the application using the requirements obtained in the Requirements Elicitation and Specification phase, as shown in figure 3. For example:

a) From the Data Requirements the designer with COHESIÓN support creates the Class Diagrams/Entities and Data Dictionary in order to obtain the Data Model and so implement the Data Base.

b) From the Transactional Requirements the designer identifies the Use Cases and with COHESIÓN support creates the service/task model, and so on.

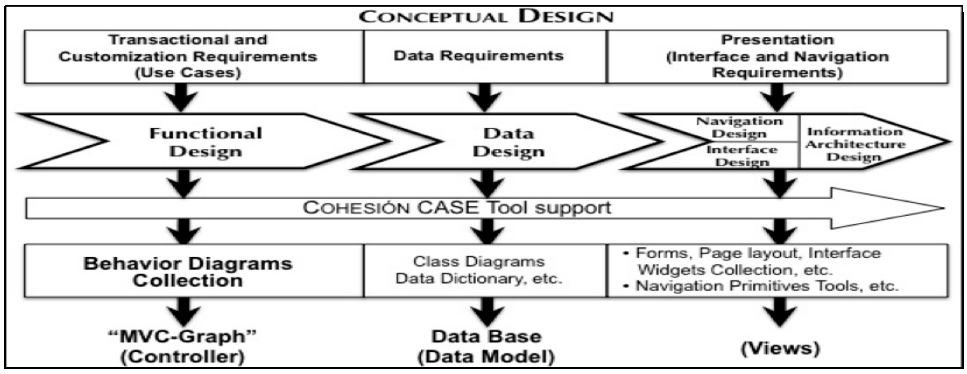

Figure 3. Conceptual design of a Web application modeled with COHESIÓN

Although UML has become a standard for modelling Web applications, UML is insufficient to model aspects such as user interfaces and business models in such applications [26]. Therefore, it became necessary to create an artefact that models key aspects such as user tasks and their relation to the different roles (actors) and data entities.

The fundamental artefact that represents both Services/Task model and Adaptation model is called Behavior Diagram [7]. A Behavior Diagram models temporal information, process workflow and relations between "user" requests and "system" responses in a Use Case. This artefact includes and summarizes all the information that can be retrieved from UML diagrams like Sequence Diagrams, Activity Diagrams and State Machine Diagrams.

The Behavior Diagrams Collection models the "MVC-Graph" of the Web application being designed, based on the MVC architectural pattern.

Let UC_i be a detailed Use Case in the Web application, and let b_i be the Behavior Diagram that models the UC_i, then the MVC-Graph = (B, A) with $B=\{b_1, b_2, ..., b_n\}$, and $A=\{(b_i, b_j) \mid \forall i, \exists j, b_i$ and $b_j \in B$, and there is at least one output from b_i that reaches $b_j\}$, A is the set of arcs in the MVC-Graph.

An example of a Behavior Diagram is shown in figure 4 where the typical Use Case "Search and View Content Article" in a Digital Journal Web Application is modelled with two types of actors: Visitor and Registered Users.

A Behavior Diagram has the following elements:

a) User Nodes: used to describe actions that users execute in their interaction with the system; each user node includes the views where the designer can attach forms (in the example, `SearchView` is a view and `SearchLayoutForm` is a form).

b) System Nodes: used to describe the actions that the system should execute in response to user requests (in the example `InitSearchAction, SearchAction`, etc.).

c) Arcs: used to represent communication between the user and the system. The arc labelled for example with "Use Case(m)", is directed to the outside of this behavior diagram and indicates the flow to another behavior diagram which models the use case "Use Case(m)".

d) Actors: used to define the type of actor who has access to an action (in the example there are two actors: REGISTERED USER and VISITOR). The designer has the mechanisms to specify which Actor, when, in which order, and under which conditions can perform each action.

e) Entities: used to define the data entities involved in the action.

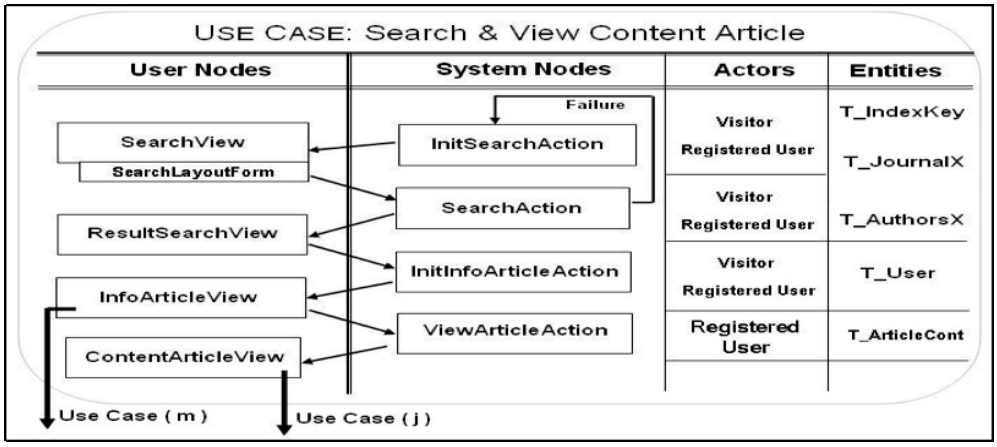

Figure 4. Behavior Diagram artefact

During the implementation phase, the designers can navigate across the Behavior Diagrams Collection (i.e. the MVC-Graph), which models all Use Cases (UC_1, UC_2, …, UC_n). With COHESIÓN support, it is possible to validate the transactional requirements of the Web application to be implemented. Finally, if the Web application will be implemented with Struts, then the COHESIÓN CASE Tool through the Model Compiler Engine transforms the Behavior Diagrams Collection and other objects, generates the `struts-config.xml` and creates other elements needed to implement the application. Furthermore, designers can include and edit all the methods needed in the actions, and refine forms and views for the final product, among other operations.

Both WEBFDM and the COHESIÓN CASE Tool are currently used to design and implement Web applications at Universidad Simón Bolívar (Caracas). Some examples of Web applications developed are summarized in Table 3.

TABLE 3. EXAMPLES OF WEB APPLICATIONS DEVELOPED WITH WEBFDM AND COHESIÓN

WEB APPLICATION	USERS	USE CASES	DB ENTITIES	REQUIREMENTS
Surveys System (Includes integration with Studies Control System)	+8,000	11	13	35
Services Management System	+300	47	22	59
Software Development Management System	+35	57	19	72

4.3 Method Fragments Collection in COHESIÓN.

According to the fact that any development project involves different characteristics and conditions, WEBFDM (through COHESIÓN support) mechanisms were included in order to make the methodology flexible. This implies that WEBFDM in any different development will be tailored for project conditions or characteristics, and an instance of the COHESIÓN CASE Tool will be set specifically for that development.

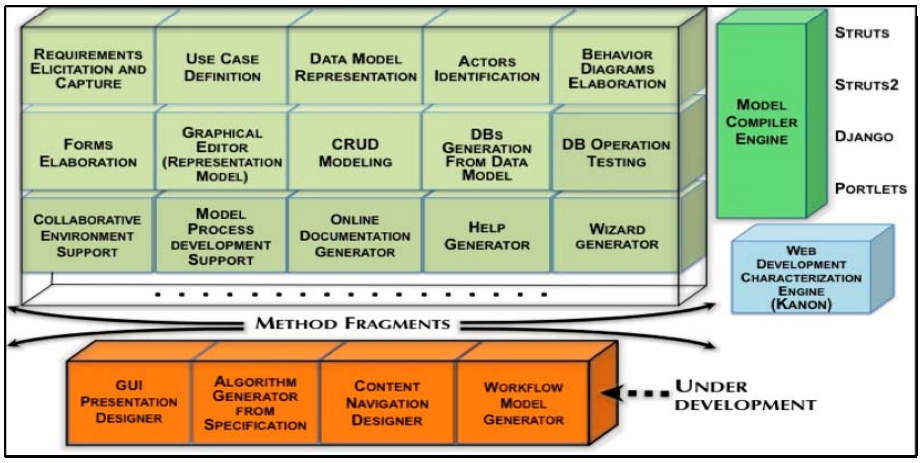

Figure 5. Method fragments collection of Cohesión CASE Tool

Considering potential benefits of Situational Method Engineering, WEBFDM was restructured as a method fragments collection. Each fragment resides at the Model layer based on the classification proposed by [9]. The current list of method fragments is shown in figure 5 and summarized in Table 4.

Each box represents a method fragment which can be arranged as a sequence based on the situation characteristics. Currently, WEBFDM with COHESIÓN support provides two categories of method fragments:

- Method fragments for modelling, design and implementation such as: Requirements Elicitation and Capture, Use Case Definition, Data Model Representation, Actors Identification, CRUD Modeling, Data Base Generation from the Data Model, Behavior Diagrams Elaboration, Forms Elaboration, Model Compiler Engine, among others [6].

- Method fragments for development support: Web development Characterization Engine, Collaborative Environment Support, Help Generator, DB Operation Testing, Wizard Generator and Model Process Development Support.

The Model Compiler Engine, already described, and the Web Development Characterization Engine are the main method fragments in COHESIÓN.

TABLE 4. METHOD FRAGMENTS COLLECTION IN COHESIÓN

METHOD FRAGMENT	ARTIFACTS PRODUCED
Requirements Elicitation and Capture (M)	Functional and non functional requirements
Use Case Definition	Use Cases Model
Data Model Representation (M)	Data Model, Data Dictionary
Actors Identification	Actor types definition
Forms Elaboration	Forms layout and definition
Behavior Diagrams Elaboration	Collection of Behavior Diagrams (MVC-Graph)
Graphical Editor (Model Representation)	Graphic representation of the Data Model
CRUD Modeling	Use Cases and Behavior Diagrams for CRUD operations
DBs Generation from Data Model (M)	Data Base
DB Operation Testing (S)	Set of test data for Data Base operations
Model Compiler Engine (M)	Transformation of models (PIM) to deployment platform (PSM)
Web Development Characterization Engine (M, S)	Collection of method fragments based on the results of KANON framework (new situational method)
Online Documentation (M, S)	The documentation of the development
Help Generator (S)	Help system to be embedded in the Web application
Wizard Generator (S)	Use Cases needed to implement the wizard
GUI Presentation Designer (*)	Set of widgets to create the rich GUI in the presentation level
Content Navigation Designer (*)	Tools to navigate the contents
Workflow Model Generator (*)	Workflow model
Model Process Development Support (S)	Process guidelines for designers
Collaborative Environment Support (S)	Activation of collaborative tools in COHESIÓN

(*) Under development; M = Mandatory fragment; S = fragment used to support the development process

Suppose that a designer has to develop an interactive Web application which requires only the creation and implementation of standard operations on a database (create, read, update and delete). Then, in this specific case, a minimal configuration of a new situational method based on WEBFDM and supported by COHESIÓN, should include the following fragments: Requirements Elicitation and Capture, Data Model Representation, Actors Identification, DBs Generator from Data Model, CRUD Modeler and Model Compiler Engine. When the method fragment "CRUD Modeling" is used the designer obtains the Use Cases and the respective Behavior Diagram Collection from COHESIÓN but s(he) does not have to use either the method fragment "Use Case Definition" or the method fragment "Behavior Diagrams Elaboration".

However, Web applications have more functionalities than the standard operations mentioned above. Therefore in this case the minimal configuration for a new method includes the following fragments: Requirements Elicitation and Capture, Data Model Representation, Actors Identification,

DBs Generator from Data Model, Use Case Definition, Forms Elaboration (for input/output interaction), Behavior Diagrams Elaboration and Model Compiler Engine. In both cases, the fragment Online Documentation stays always active because it generates all kinds of documents related to the developed Web application.

The method designer initiates the process for creating a new method using COHESIÓN through the “Web Development Characterization Engine” method fragment. This fragment is used to identify the development situation, the Web application characteristics and to select the method fragments in order to obtain a new situational method. This method fragment is described in the next section.

4.4. Web Development Characterization Engine – KANON Framework

Every new situational method to be created and supported by the COHESIÓN CASE Tool is defined by identifying its characteristics in the development situation. The situation is determined by various attributes associated with the final product, organizational development environment, types of users, technology and others. The "Web Development Characterization Engine" method fragment is responsible for identifying the characteristics of the Web application and the development situation. This fragment is anchored to a framework called KANON. This framework is based on the foundations of Web Engineering, which takes into account the fundamental elements about the development of Web applications described in section 2.2: levels (content, hypertext and presentation) and aspects (structure and behavior of each element on levels).

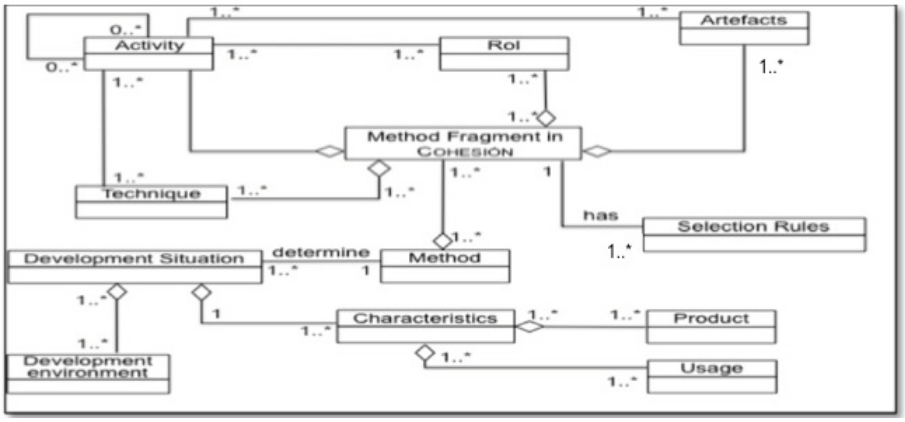

Figure 6. Metamodel of method fragment in COHESIÓN

The method construction approach followed by the "Web Development Characterization Engine” is assembly-based. The point of departure to create a new method is the repository of method fragments currently available in COHESIÓN (Table 4). The fragments collection provides enough details to be used in different types of Web applications. All method fragments are at the same low granularity layer in order to create the new method with high flexibility. This proposal does not consider methods from other approaches. If a new method fragment has to be included in COHESIÓN, it will be defined according to the metamodel shown in figure 6.

TABLE 5. KANON DIMENSIONS FOR CHARACTERIZING WEB APPLICATION DEVELOPMENT SITUATIONS

DIMENSIONS	ATTRIBUTES DIMENSION	SOME VALUES/SETTINGS
PRODUCT	Category	Content driven , Interactive, Transactional, Workflow, Collaborative, Portal, Social web, Ubiquitous, Semantic Web, etc.
	Common Models: • Levels: Content, Hypertext, Presentation • Aspects: Structure Behavior	Content → data model, information architecture model, Hypertext → content navigation model, interaction model Presentation → interface model (forms and views contents) Structure → Data Model entities, Data Dictionary, data glossary, Class Diagrams, Information Architecture Taxonomies and Categories, Behavior → Operations, Methods, Rules, etc.
	Feature Models	• Requirements analysis, • Adaptation/customization model (Actors identification based on roles) • Service/Task model (Behavior Diagrams - MVC-Graph), • Implementation model
	Complexity	φ (Requirements number, quality and documentation of requirements, Criticality, Size/Scale, Domain, Natural Context, Technical context) → low, medium, high
	Domain	Commerce/Business (Tourism, Booking services, etc.), CRM, CMS, Health, LMS (Education), etc.
USAGE	Social context	Requirements analysis → users, stakeholders, personalization
	Technical context	Infrastructure: networks, devices, deployment platform
	Natural context	Non Functional Requirements → availability, security, etc. physical context → intranet, extranet, internet
DEVELOPMENT ENVIRONMENT	Development team and organization	Developers → skilled; multidisciplinary; Development environment →location; tool support
	Technical infrastructure	Innovative; novelty, etc.
	Process	Agile/disciplined approach; planning; resources; commitments; quality standard; time to delivery → phases and activities
	Integration with other systems	internal, external → Web Services development, database access, etc.
EVOLUTION	Considered as a new development situation.	

In the KANON framework, a Development Situation determines a new instance of the method to be created using the method fragments collection provided by COHESIÓN. As shown in the metamodel a new situational method is determined by a combination of the Web application characteristics

represented by the attributes present in Product and Usage, and the Development Environment attributes. The concept of Evolution in Web application is considered as another development situation. In Table 5 the dimensions and some attributes considered by KANON are summarized.

Each method fragment in COHESIÓN has selection rules, required artefacts/models (preconditions) and produced artefacts/models (postconditions).

In figure 7 is shown the description of "Use Case Definition" (a) and the "Data Model Representation" (b) method fragments. Following is described some rules of the method fragment descriptions:

- Each fragment has a proper type and it is used with a specific function within the methodology: The "Use Case Definition" fragment is optional and is used to accomplish Analysis activities. The "Data Model Representation" fragment is mandatory and is used for Design activities.
- The description includes information about methodology phase and activities where the method fragment is used: the "Use Case Definition" method fragment is used in the Requirement Elicitation phase to accomplish the Transaction Requirement Elicitation activity. On the other hand, the "Data Model Representation" fragment is used in the Conceptual Design phase, specifically during the Data Design activity.
- The rule "`application_category`" indicates if the method is required by the Web application categories listed. In the example, both fragment methods, "Use Case Definition" and "Data Model Representation" are necessary in developing interactive, transactional, workflow and collaborative Web applications. Each new advanced category is related to the evolution of functionality, new technology use, etc. The framework considers the Web application Category as an attribute in the dimension "Product". Different resources and techniques are required to develop Web applications, and different models are to be created because of Web application category dependence.
- Other concepts in the fragments description are related to levels and aspects (based on the Web Engineering Framework): the structure aspect of the content level is the main concerns of the "Data Model Representation" fragment. Meanwhile the behavior aspect of the hypertext level is the main objective of the "Use Case Definition" fragment. In both cases each fragment produces the related models as is shown as `<produced_artefact>` in the description.
- The complexity concept (or rule) indicates if the method fragment is necessary to be used by one or more complexity levels of Web applications. Based on the complexity level of a Web application some formalisms and additional methods are required to develop the application. Generally, a Web application with high complexity level requires major and diverse types of artefacts to model functionality, navigation, data, etc. In the other hand, a Web application with low complexity level requires fewer artefacts to be produced. Complexity is a combination of different factors like: Web application category, requirements number and quality, criticality of the application, size, domain, etc. This proposal considers low, medium and high complexity level. For example, the "CRUD Modeling" fragment may be used if complexity level of the Web application to be developed is low.

Method fragment description (and characterization) is carried out when a new method is included in the Method Fragment Repository in COHESIÓN. The designers can describe the new method.

```
<fragment>
   <id-fragment>
      UCD_Use_Case_Definition
   </id-fragment>
   <rule>
      <type>  fragment_type   </type>
      <value> Optional   </value>
   </rule>
   <rule>
      <type>  function  </type>
      <value> Analysis   </value>
   </rule>
   <rule>
      <type>  phase   </type>
      <value> Requirements_Elicitation  </value>
   </rule>
   <rule>
      <type>  activity   </type>
      <value> Trasnsaction_Requirements  </value>
   </rule>
   <rule>
      <type>  application_category   </type>
      <value> interactive </value>
      <value> transactional </value>
      <value> workflow </value>
      <value> collaborative </value>
   </rule>
   <rule>
      <type>  level   </type>
      <value> Hypertext   </value>
   </rule>
   <rule>
      <type>  aspect   </type>
      <value> Behavior   </value>
   </rule>
   <rule>
      <type>  complexity   </type>
      <values> low, medium, high   </values>
   </rule>
   ... ... ... ...
   <required_artefacts>
      <artefact>  Use_Case_Description  </artefact>
   </required_artefacts>
   <produced_artefacts>
      <artefact>  Use_Case_Model  </artefact>
   </produced_artefacts>
</fragment>
```

(a)

```
<fragment>
   <id-fragment>
      DMR_Data_ Model_Representation
   </id-fragment>
   <rule>
      <type>  fragment_type   </type>
      <value> Mandatory   </value>
   </rule>
   <rule>
      <type>  function  </type>
      <value> Design   </value>
   </rule>
   <rule>
      <type>  phase   </type>
      <value> Conceptual_Design   </value>
   </rule>
   <rule>
      <type>  activity   </type>
      <value> Data_Design   </value>
   <rule>
      <type>  application_category   </type>
      <value> interactive </value>
      <value> transactional </value>
      <value> workflow </value>
      <value> collaborative </value>
   </rule>
   <rule>
      <type>  level   </type>
      <value> Content   </value>
   </rule>
   <rule>
      <type>  aspect   </type>
      <value> Structure   </value>
   </rule>
   <rule>
      <type>  complexity   </type>
      <values> low, medium, high   </values>
   </rule>
   ... ... ... ...
   <required_artefacts>
      <artefact>
            data_requirements_documentation
      </artefact>
      <artefact>  data_model  </artefact>
   </required_artefacts>
   <produced_artefacts>
      <artefact>  data_model_entities  </artefact>
      <artefact>  data_dictionary   </artefact>
   </produced_artefacts>
</fragment>
```

(b)

Figure 7. Method Fragments Description

With respect to method fragments related to development support tasks the rules are associated to the Development Environment Dimension. The rules specify presence/absence (and in some cases values) of attributes considered in KANON. If the already specified selection conditions are satisfied, the fragment is a candidate for incorporation into the new method.

4.5 Method fragments selection process to assemble a new method

The "Web Development Characterization Engine" method fragment uses the resulting KANON attributes and creates different structures in order to select the method fragments of the new situational method. The algorithm implemented in the "Web Development Characterization Engine" fragment for the construction of the new method is shown next in Table 6 and figure 8.

TABLE 6. MAIN ALGORITHM OF THE FRAGMENT "WEB DEVELOPMENT CHARACTERIZATION ENGINE"

- m represents models/artefacts and f represents fragments –

{ Create_Categories-Models(Kanon_Attributes); `– Step 1 –`

Create_Fragments-Models(Kanon_Attributes); `– Step 2 –`

Set $c_i \leftarrow$ *category_of_the_web_application;* `– Step 3 –`

Set $A_0 \leftarrow \{ m \mid m$ *is required in web application category* $c_i \}$*;*
Set $F_0 \leftarrow \{ f \mid f$ *produce* $m, m \in A_0\}$*;*
Set $i \leftarrow 0$*;*

Repeat `– Step 4 –`
Set $i \leftarrow i+1$*;*
Set $A_i \leftarrow A_{i-1} \cup \{ m \mid m$ *is required by* $f, f \in F_{i-1}\}$*;*
Set $F_i \leftarrow F_{i-1} \cup \{ f \mid m$ *is produced by* $f, m \in A_{i-1}\}$*;*
until $F_{i-1} = F_i$*;*
}

– The final F_i *is the method fragments Collection and* A_i *is the artefacts / models Collection which models the application –*

The most significant structures are:

a. The fragment descriptions that reside in the Method Fragments Repository.

b. Categories-Artefacts/Models matrix in order to distinguish the artefacts (models) needed based on the Web application category to be developed and constructed using the rules in the fragments description.

c. The Fragments-Artefact/Models matrix that represents the artefacts (models) produced by a fragment (value = 1), and the artefacts to be used o required by a fragment (value = -1). A method fragment can use the same model in different iterations so, in this case, the value of the relation is equal to 2. These values are recovered from the method fragments description.

The process guarantees the completeness requirement because the final fragment collection contains all the method fragments that are referred to by other fragments in the method based on [9]. The precedence consistency is assured because all models and fragments are placed in the right order in the new method. Such precedence shows the artefacts and methods sequence needed and it is arranged as a graph to guide the designers, to show the critical path in the development and the process to be followed. This process is oriented to generate the most flexible method without incrementing the complexity in the assembled new method. As stated before, all fragments are at the same granularity layer.

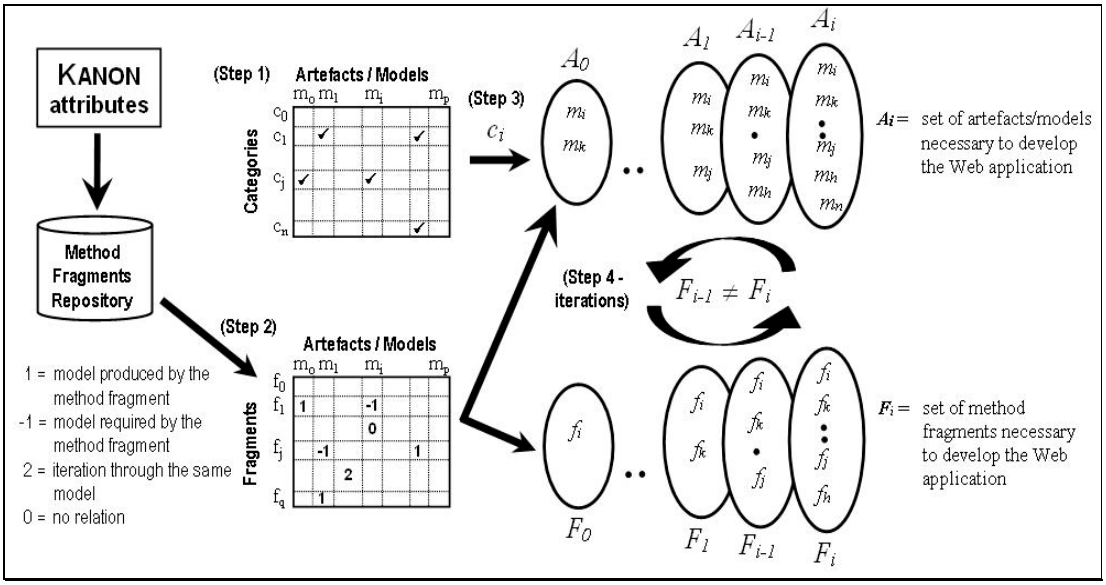

Figure 8. General process for selecting method fragment Collection

4.6 Using KANON and the Method Fragment "Web Development Characterization Engine"

In this section the KANON framework operation is described. The example is a Web application to be developed for a Consultant Firm. Some significant requirements about the Web application to be developed are listed in Table 7.

TABLE 7. "CONSULTANT FIRM WEB APPLICATION" EXAMPLE

GENERAL PURPOSE	The Consultant firm needs a Web application so that its consultants can record the hours worked on each project in which they are involve, worldwide.
REQUIREMENTS	
Data Requirements	a) Each consultant has a user and password to login to the Web application. b) The data to be recorded are: project number, date, hours spent, and working detail (activities carried out).

	c) All input data has to be validated.
Transactional Requirements	a) Each consultant can create and edit new records, and has access only to his corresponding projects. b) Each consultant can print a report of his recording activities in each project. c) Each transaction due by each consultant in the Web application has to update the Payroll System (consultant) and the Account Management System (project customers) of the firm.
Interface Requirements	a) The interaction style is text based, including standard text form for data capture. b) The consultant can use the Web application through his computer and/or Smartphone, anywhere, anytime.
Non Functional Requirements	a) The application has to be online in one week. b) The consultants are directly involved in the design. c) Security (projects and consultants information) and availability (24/7) issues have to be considered. d) Only 4 members of the teamwork are available and are located in the same site.

Once KANON determines the development situation the results are summarized in Table 8. Finally, the method fragments selected by the "Web Development Characterization Engine" method fragment, used to create the new method for the above example are shown in Figure 9.

TABLE 8. USING KANON FRAMEWORK

DIMENSIONS	RESULTS FROM KANON
PRODUCT	• **Category**: Transactional; • **Models** to be created: Data Model, Presentation Model, Service/Task Model through Behavior Diagrams, Actor types definition; • **Complexity**: Low, no critical. **Domain**: e-business
USAGE	• **Social context**: Users and stakeholders known; • **Technical context**: dual deployment platform (Smartphone and PC); • **Natural context**: availability (24/7), extranet
DEVELOPMENT ENVIRONMENT	• **Teamwork**: experienced, small group (4 members); • **Location**: same location; • **Tool support**: available for Struts2 and Django; • **Technical infrastructure**: DC (don't care); • **Process**: agile; **time to delivery**: 1 week; **model process**: Agile, rapid prototyping; • **Integration**: with legacy systems (internal).

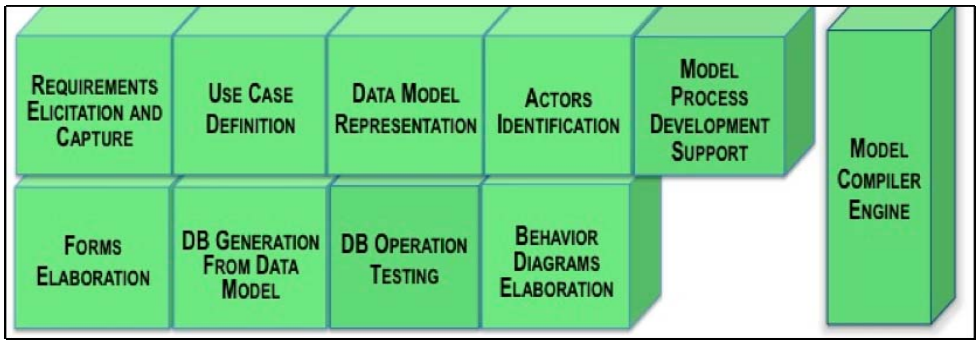

Figure 9. Method Fragments Collection for the "Consultant Firm Web Application"

The method fragments are turned on in the COHESIÓN CASE Tool. With a graphical mechanism which is under development, a graph that guides the designers will be displayed. The graph shows the models to be created and the activities to be carried out to develop the application. The graph also shows the method fragments used for support in the COHESIÓN CASE Tool. The graph representing the process development to be carried out is shown in figure 10: the rectangles represent the method fragments activities, and the ovals represent the artefacts generated by each fragment.

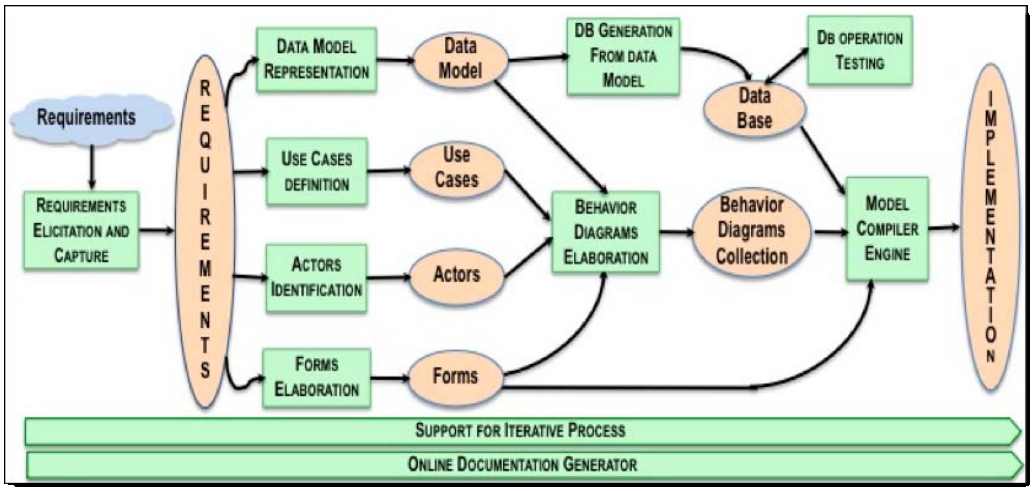

Figure 10. New situational method for the example

If a new requirement comes out during the development, for example "the members of the teamwork are located in different sites", the attributes of the Development Environment dimension and the overall Development Situation change. In this case, the "Web Development Characterization Engine" fragment will also select the method fragment "Collaborative Environment Support" (see Figure 5) to bring all support to the teamwork in order to use this functionality.

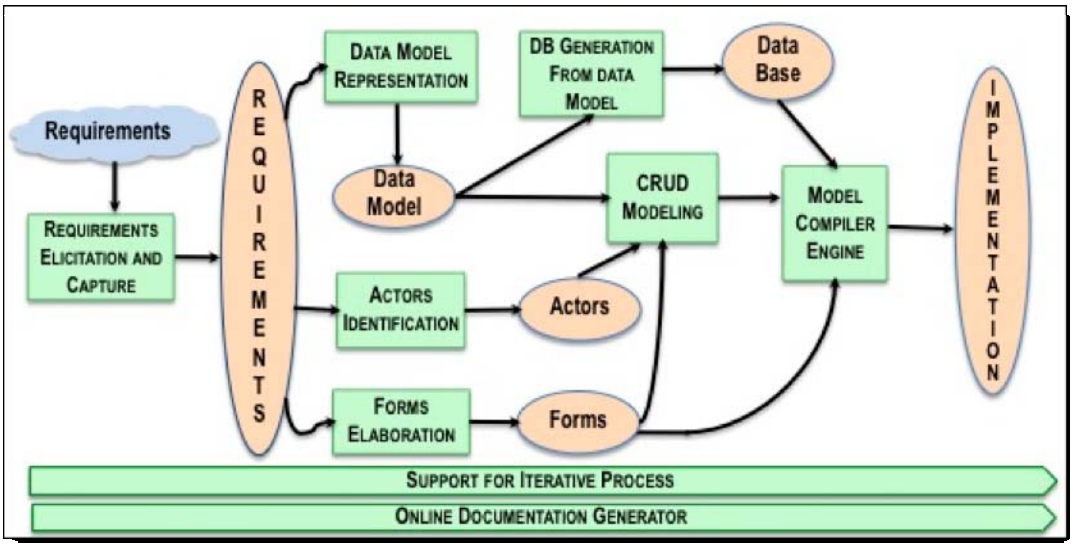

Figure 11. Situational method for a Web application with CRUD functionalities

Suppose the designers have to develop another Web application only with CRUD functionalities. Once KANON recognizes the attributes, the selection process carried out by the "Web Development Characterization Engine" generates the new situational method shown in figure 11 as a graph.

The proposal presented, WEBFDM and COHESIÓN, is used to develop applications aimed at solving administrative and academic needs of our institution.

5 Conclusion and future works

The aim of this research is twofold: i) To define WEBFDM methodology as a method fragments collection that follows the framework proposed by [18] and [33] in order to achieve a methodology that covers all topics needed for Web application development. All concepts are based on the foundations of Web Engineering and the principles of Situational Method Engineering; ii) To identify useful factors to characterize a Web application development situation. The factors, organized in dimensions are the core of the KANON framework, which is used to characterize the Web application to be developed. The dimensions and attributes collection identified determine the combination of method fragments needed to create the new method adjusted to the current development situation. KANON is the core of the "Web Development Characterization Engine" method fragment and has been implemented in COHESIÓN to produce a new situational method.

WEBFDM with COHESIÓN support will incorporate the complete features and configuration mechanisms necessary to make the methodology a flexible, model-driven development of Web applications with the CASE Tool support. Furthermore, it is quite able to shape the development

process and the generation of models and products, based on a proper situation identified by the KANON framework.

There are two additional contributions related to the modelling aspects of the proposal: the MVC-Graph concept and the novel artefact called "Behavior Diagram" that models the interaction between user and system and is used to declare the entities and actors involved in a Use Case. This artefact includes and summarizes all the information that can be retrieved from UML diagrams like Sequence Diagrams, Activity Diagrams and State Machine Diagrams. The MVC-Graph is the behavior diagrams collection and models interactions and functionalities of the whole Web application and can be tested and validated using the COHESIÓN CASE Tool.

The COHESIÓN CASE Tool supports WebFDM, leads the developer in the creation of the Behavior Diagrams and generates codes that run in Struts, Struts2, Django and Portlets. Because COHESIÓN is model-driven, the decisions about the deployment platform (PSM) can be delayed until the implementation phase is reached.

In addition, the CASE Tool generates documents about Use Case Model, Data Model, Data Dictionary and other objects that are useful to consolidate the documentation of all functionalities of the Web application developed. In the evolution and maintenance phase, the designer can review and edit those objects and documents and incorporate new functionalities. If so, COHESIÓN will transform the edited models through the model-driven fragment "Model Compiler Engine" and generate the new implementation in the deployment platform.

Currently, different method fragments are being developed to complete some concerns like presentation and navigation. The method fragment "GUI Presentation Designer" will be integrated to COHESIÓN to support the design of the presentation level of Web application, so the designer can define rich interface mechanisms to be incorporated in the application. The "Content Navigation Designer" will include a collection of navigation tools like Index, Trails, TOC, Maps, etc., which can be incorporated in Web applications, based on the results of the Information Architecture activity which is part of the Conceptual Design phase.

In the literature review phase, several Web Engineering approaches were found in which the principles of Situational Method Engineering are used. Some of them are oriented towards creating new methods based on a specific domain, as in [40] and [41]. In [28], [29], [36] and [37] authors stress defining new method creating strategies. Useful are the works presented in [27], [28], [36] and [37] leading towards the definition of development characterization factors. In the case of method guideline generation, the approaches described in [36] and [37] stand out.

From our point of view, our proposal attempts to fulfil some gaps in the treatment of these issues, as explained next.

KANON includes factors associated to Web Engineering concerns like hypertext and presentation model, which are not considered in the proposal of [27]. Then KANON extends the project characteristics.

Our proposal also extends the factors identified by [28], [36] and [37]. All works considered only the following application types: kiosk, i.e. content driven Web application, Web Information

Systems (WIS), adaptive application and e-commerce applications. KANON considers a wider variety of Web applications, from content-driven to ubiquitous, semantic Web and portal applications.

The KANON framework proposed in this paper includes new attributes and a different organization of the dimensions used by [24]. The new attributes are related to Domain and Category-Models relationship.

The method construction approach followed by the proposal is assembly-based and is implemented in the "Web Development Characterization Engine" fragment. The point of departure to create a new method is the repository of method fragments currently available in COHESIÓN. The fragments provide enough details and strengths to be used in different types of Web applications. All method fragments are at the same granularity layer in order to create the new method with high flexibility and low complexity, and the requirement about granularity consistency as stated by [9] is guaranteed.

The process proposed to construct a new method assures the completeness requirement because the final fragment collection contains all the method fragments that are referred to by other fragments in the method as mentioned in [9].

The "Web Development Characterization Engine" fragment generates the set of models/artefacts and the set of methods fragments that may be used to fulfil the development of the Web application. Both sets can be arranged like a DAG (Directed Acyclic Graph) in order to guide the designers, to show the critical path in the development and the process to be followed.

The precedence consistency requirement is assured because all models and fragments are placed in the right order in the new method based on [9]. The graph can help project managers during planning activities, scheduling and resources assignment.

The KANON framework is being validated through the characterization of different Web applications that we have developed in our university. The main differences during the validation process are related to application categories, requirements number, size and complexity level, among others.

The next step in this research is to extend WEBFDM and COHESIÓN including new method fragments aimed at improving the functionality of the CASE tool. KANON will be revised in order to incorporate new attributes and mechanisms to characterize Web application development, since technology, usage and platform deployment are in continuous change and evolution. An additional extension in WEBFDM will be the inclusion of project planning management and risks identification in Web application development using concepts from the approach proposed by [2] and [15].

References

1. Apache. STRUTS open-source framework. 2011. Available at http://struts.apache.org
2. Al-Rousan, T., Sulaiman, S. and Abdul Salam, R. Risk Analysis and Web Project management. Journal of Software, Vol 4, Nº 6, pp. 614-621. August 2009.
3. Bajec, M., Vavpotic, D. and Krisper, M. Practice-driven approach for creating projecy-specific software development methods. Journal of Information and Software Technology. Vol. 49, Issue 4, pp. 345-365. 2007.

4. Bianchini, A., Ortega, M. and Suárez, A. Una Metodología de Diseño de Aplicaciones Web bajo el Patrón MVC. Jornadas Chilenas de Computación - 2005. Actas XIII Encuentro Chileno de Computación 2005. Valdivia, Chile, November 2005.
5. Bianchini, A., Blanch, R., Ortega, M. and Suárez, A. Diseño de una herramienta para el desarrollo de aplicaciones Web basadas en Struts. Proceedings IADIS Conferencia Iberoamericana WWW/Internet. Vila Real, Portugal, December 2007.
6. Bianchini, A. WEBFDM: A Web Application Flexible Development Methodology. Proceedings IADIS International Conference on WWW/Internet ICWI 2010 - Doctoral Consortium, pp 427-431. Timisoara, October 2010.
7. Bianchini, A., Ortega, M. and Suárez, A. MVC Based Behavior Diagrams as Key Artifacts for Web Application Design. Proceeding IADIS Information Systems IS 2011, pp 137- 144. Avila, Spain, March 2011.
8. Brinkkemper S. Method engineering: engineering of information systems development methods and tools. Information & Software Technology, 38(4), pp. 275-280, 1996.
9. Brinkkemper, S., Saeki, M. and Harmsen, F. Assembly Techniques for Method Engineering. Proceedings CAiSE´98. Pernici and Thanos, Editors. Lecture Notes in Computer Science 1413, pp. 381-400. Springer Verlag. 1998.
10. Cachero, C. OO-H: Una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales. 2003. Available at http://www.dlsi.ua.es/~ccachero/pTesis.htm
11. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M. Designing Data-Intensive Web Applications, Morgan-Kaufmann, 2003.
12. De Troyer, O., Casteleyn, S., and Plessers, P.: Using ORM to Model Web Systems, Proceedings of International Workshop on Object-Role Modeling. Cyprus. 2005.
13. Díaz, P. and Aedo, I. Towards Efficient Web Engineering Approaches Through Flexible Process Models. The Journal of Systems and Software. Nº 80, pp. 1375-1389. 2007.
14. Django Project. Django: The Web framework for perfectionists with deadlines. 2001. Available at http://www.djangoproject.com
15. Engels, G., Lohmann, M., and Wagner, A. The Web Application Development Process. In Kappel, G., Pröll, B., Reich, S., Retschitzegger W. (Editors): Web Engineering - The Discipline of Systematic Development of Web Applications, pp. 197-218. John Wiley and Sons Ltd. 2006.
16. Escalona, M.J., Koch, N. Requirements Engineering for Web Applications – A Comparative Study. Journal of Web Engineering, Vol. 2, No. 3 pp. 193-212. Rinton Press. 2004.
17. Escalona, M. J., Gutierrez, J. J., Villadiego, D., León, A. and Torres, J. Practical Experiences in Web Engineering. Advances in Information Systems Development, pp. 421-433. 2007.
18. Fraternali, P. Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. ACM Computing Surveys, 31 (3), pp. 227-263. September 1999.
19. Garrigos, F., I. A-OOH: Extending Web Application Design with Dynamic Personalization. European PhD Thesis. Universidad de Alicante. 2008. Available at: www.dlsi.ua.es/~igarrigos/tesisIreneGarrigos.pdf
20. Gericke, A., Fill, H., Karagiannis, D. and Winter, R. Situational Method Engineering for Governance, Risk and Compliance Information. Proceedings 4th International Conference on Design Science Research in Information Systems and Technology, DESRIST'09. ACM Press. Article no. 24. Malvern, PA. 2009.
21. Harmsen, A. F. Situational Method Engineering. Doctoral Dissertation, University of Twente. Published and distributed by Moret Ernst & Young Management Consultants 1997.
22. Henderson-Sellers, B., Serour, M., McBride, T., Gonzalez-Perez, C. and Dagher, L. Process Construction and Customization. Journal of Universal Computer Science, Vol. 10, Nº. 4, pp. 326-358, 2004.

23. Henderson-Sellers, B. and Ralyté, J. Situational Method Engineering: State-of-the-Art Review. Journal of Universal Computer Science, Vol. 16, Nº. 3, pp. 424-478. 2010.
24. Kappel, G., Pröll, B., Reich, S. and Retschitzegger, W. An Introduction to Web Engineering. Chapter 1, In Web Engineering: The Discipline of Systematic development of Web Application. John Wiley and Sons, pp. 1-17. 2006.
25. Koch, N., Knapp, A., Zhang, G. and Baumeister, H. UML Based Web Engineering: An Approach Based on Standards. In Web Engineering: Modelling and Implementing Web Applications. HCI Series, vol. 12, chapter 7, pp. 157-191, Springer-Verlag. 2007.
26. Kong X., Liu L., and Lowe D. Separation of Concerns: a Web Application Architecture Framework. Journal of Digital Information. Vol. 6, no. 2. 2005. Available at: http://journals.tdl.org/jodi/article/view/69
27. Kornyshova, E., Deneckere, R., and Salinesi, C. Method Chunks Selection by Multicriteria Techniques: an Extension o f the Assembly- based Approach. In IFIP Vol. 244, Situational Method Engineering: Fundamentals and Experiences. Eds. Ralyte, J., Brinkkemper, S., Henderson- Sellers B., pp. 64-78. 2007.
28. Kraiem, N., Selmi, S. and Ben Ghezala H. A Situational Approach for Web Applications Design. IJCSI International Journal of Computer Science Issues. Vol. 7, Issue 3, No. 1, pp. 37-51. May 2010.
29. Lahajnar, S. A Framework for Situational Web Methods Engineering. Proceeding of 7th International Conference on Web Engineering ICWE 2007. Springer-Verlag Berlin, pp. 569-574. 2007.
30. Luinenburg, L., Jansen, S., Sourer, J., van der Weerd, I. and Brinkkemper, S. Designing Web Content Management Systems Using the Method Association Approach. Proceedings 4th International Workshop on Model-driven Web Engineering MDWE 2008.
31. Pérez D., C. A. Traducción Dirigida por Modelos para el Desarrollo de Aplicaciones Web. Computer Science Master Thesis. Universidad Simón Bolívar. 2011. Available at: http://www.ldc.usb.ve/~caperez/thesis/book.pdf
32. Ralyté, J. Requirements Definition for the Situational Method Engineering. Proceedings of the IFIP TC8/WG8.1 Working Conference on Engineering Information Systems in the Internet Context (EISIC'02), Kanazawa, Japan. C. Rolland, S. Brinkkemper , M. Saeki (Eds.), Kluwer Academic Publishers, pp. 127-152. September 2002.
33. Retschitzegger, W. and Schwinger, W. Towards Modeling of DataWeb Applications: A Requirement's Perspective. Proceedings of the Americas Conference on Information Systems (AMCIS 2000), Long Beach, CA. August 2000.
34. Schwinger, W. and Koch, N. Modeling Web applications. In Kappel, G., Pröll, B., Reich, S. and Retschitzegger, W. (Eds), Web Engineering – Systematic Development of Web Applications, John Wiley and Sons, NY, pp. 39-64. 2006.
35. Sensio Labs. Symfony. 2011. Available at http://symfony.com/
36. Selmi, S., Kraiem, N. and Ben Ghezala, H. Toward a comprehension view of Web Engineering. Proceedings of International Conference in Web Engineering ICWE 2005, LNCS 3579. Springer-Verlag, Berlin Heidelberg, pp. 19-29. Sidney, Australia. 2005.
37. Selmi, S., Kraiem, N. and Ben Ghezala, H. Guidance in Web Applications Design. Proceedings of Model Driven Information Systems Engineering: Enterprise, User and System Models MoDISE-EUS 2008, pp. 114-125. Montpellier, France. 2008.
38. Sourer, J., Kupers, T., Helms, R. and Brinkkemper, S. Model-Driven Web Engineering for the Automated Configuration of Web Content Management Systems. Proceeding of International Conference of Web Engineering. Spain. Springer-Verlag, pp. 121-135. June 2009.
39. Valverde, F., Valderas, P. and Fonts, J. OOWS Suite: Un Entorno de desarrollo para Aplicaciones Web basado en MDA. In IDEAS 2007, X Workshop Iberoamericano de Ingeniería

de Requisitos y Ambientes Software. Losavio, F., Travassos, G., Pelechano, V., Díaz, I., Matteo, A., Editors; pp. 253-266. Venezuela. 2007.

40. Vlaanderen, K., Valverde, F. and Pastor, O. Model-Driven Web Engineering in the CMS Domain: A Preliminary Research Applying SME. Lecture Notes in Business Information Processing, Volume 19, Part 4, 226-237, 2009.
41. Weerd van de, I. Brinkkemper, S., Souer, J., and Versendaal, J. A Situational Implementation Method for Web-based Content Management System Applications: Method Engineering and Validation in Practice. Software Process: Improvement and Practice 11(5), pp. 521-538. 2006.
42. Ziemer S. Trade-offs for Web application development: understanding and improving current industrial practices. Doctoral theses. Norwegian University of Science and Technology. 2009.