Conceptual Modelling of Services in Multi/Cross Channel Web Applications

Franca Garzotto and Vito Perrone

HOC- Hypermedia Open Center
Department of Electronics and Information, Politecnico di Milano (Italy)
garzotto|perrone@elet.polimi.it

Abstract.
Web Applications are progressively becoming *multi-channel* and *cross-channel.* The "same" service should be made available in different delivery environments and devices. A user may invoke a service on one device, suspend it, and complete its execution in another one. In this paper we present MC2 a design framework for Multi/Cross Channel web services adopting a high-level, end user perspective and exploiting the notion of *context,* to characterize who, where and how an operation can be invoked. In this paper the term web service is intended in a broad sense as a service provided by a Web Application and for which the user is interested in.

Keywords: conceptual modelling, multi-channel, services on the Web, Web operations, context, UML, OCL.

1. Introduction

Modern web applications differ from original web sites from a number of aspects. In the past, we had read-only, mono-channel web sites. Today, web applications provide functionality beyond search and navigation, offering a variety of *services* that allow users modify the application state. They are progressively becoming *multi-channel:* users can work with the same application using various delivery channels (e.g., desktop PCs, laptops, mobile phones, PDAs, Web TVs, etc.), usually finding different contents and services in each different device. Finally, modern web applications are evolving towards being *cross-channel:* the effects of an operation invoked on one device can be perceived, by the same or other users, when working with another device; for complex, long-lived services, the user can perform some operations required by a service in one device, suspend them, and continue with another device.

As a consequence, the design of web applications is becoming more and more complex, and new models and methods must be explored to support the design activities. Design can be tackled at several levels of abstraction and from different perspectives. In this paper, we discuss MC2, a framework for conceptual modelling of Multi/Cross Channel web application services, that addresses design from a high-level, *end user perspective*, abstracting from any technological or implementation dependent aspects. We consider only services that involve some form of user interaction and we model only features that can be perceived by *end users.* In other words, we model services as "user

experiences" rather than "system experiences". The key aspects of MC2 are the following:

- We take into account the intrinsic *hypermedia* nature of services on the web. Users perceive the application as a hypertext network where navigation is the primary interaction paradigm. Services can be invoked only if the user is located in specific points of the hyperspace; users may navigate while using a service; user operations required by a service may have navigational effects, i.e., they may change the user position in the hyperspace.
- We decompose *services* into *"elementary" user interactions* that can be either operations or navigation steps and can be executed on a single device or on multiple devices, and may produce effects, which are perceived on multiple devices and by several users.
- We take into account the need of providing different "versions" of the same application depending on the *context*, i.e., on the characteristics of both the user invoking a service and the situation of use (including the device). We complement the service model with a *context model* and a *hyperview model*, which allows designer to specify how different users perceive services in different situations of use. Differently from most object-oriented web models, where operations are modelled as methods attached to information "objects", or to navigation or presentation "objects" (pages), we associate operations and services to contexts and hyperviews.
- We adopt a notation based on UML [1] to describe the various models. In particular, we adopt OCL [2] to model service constraints and effects, "extending" standard OCL with some predicates that explicitly relate to context and hyperviews.

In this paper, we shortly introduce the main concepts of the overall MC2 framework, and focus on the design of operations, which are the building blocks for designing services. To exemplify our approach, we specify several operations using the MC2 modelling elements.

2. The MC2 approach

MC2 is a modelling framework for designing web application services from a user perspective. It is composed of a *service model,* an *operation model*, a *context model*, and a *hyperview model*.

2.1 Services, interactions, and operations

In a user-centred perspective, we think of a service as a user *activity*: a (non linear) flow of *tasks* that the user performs within the application to achieve a given goal. Tasks can be progressively decomposed into subtasks up to elementary building blocks that we call *interactions.* The concept of "elementary" is subjective. It depends on various factors: the level of detail that the designer wants to achieve in the specification of interactions, the designer perception of what can be considered as an elementary task for the user (not for the system executing it). The minimal, necessary condition for considering an interaction as elementary is its atomicity: either all its effects are achieved, or none of them is.

Interactions in web application services have a heterogeneous nature: they can affect the navigation position of the user (and are typically called "navigation") or the presentation state of the objects on the screen (e.g., scrolling a page, playing a sound track); they can change the application state; or they can mix all these effects. In our terminology, an elementary interaction, that causes (among other affects) a modification of the application state (from the user point of view), is called *operation*. Defining precisely an application state is outside the purpose of a user-focused design model (and rather falls into the scope of system oriented methods). Informally speaking, we can say that the application state is defined by the state of all "objects" which describe the application at a logical level, from a system perspective.

As mentioned in the introduction, this paper focuses on operations modelling only. From a structural perspective, an operation is defined by the following ingredients, based on [3]:

- The conditions under which an operation can be invoked, or *pre-conditions*. They are "evaluated" before the operation execution starts.
- The input parameters, i.e., the operation *arguments*. Arguments can be either provided by the user (*user-arguments*) or can be "calculated" by the system (*system-arguments*). Arguments (and operation name) are described in the *operation signature*.
- The *effects* resulting from executing the operation, described by *post conditions*. They express application properties that must hold after completing the operation and are evaluated after the operation execution.
- The *synchronicity* of the operation. This property specifies whether the user can or cannot interact with the application while the operation is under execution. The operation is *synchronous* if no interactions can be performed until the synchronous operation is completed. The operation is *asynchronous* if other interactions can be invoked and executed in parallel with the operation (as far as they do not affect the execution of the synchronous operation).

The hardest problem for the operation designer is to model pre and post conditions. In multi-cross channel web applications, they are intrinsically complex for a number of reasons:

- Users do not perceive the entire application, but only a "view" on it – the "portion" of the application which better fits with their profile and their situations of use (which comprises, for example, device, location, temporal aspects, etc.). In particular, only some types of users using some specific devices can invoke an operation.
- Operations may have a variety of constraints on their execution and may involve many sophisticated effects. The user invoking the operation directly perceives some effects, other effects are perceived also (or only) by other users. The invoker may perceive some effects in the same situation of use where the operation has been invoked, others in a different situation of use. Some effects may even change the invoker profile or the characteristics of the situation of use (e.g., some device features), implicitly changing the "view" that the invoker perceives of the application. Finally, operations may affect the user navigational state, determining a user movement in the hyperspace.

To allow operation designers take all the above aspects into account, we introduce the notions of context and hyperview, and structure the operation design space in a number of dimensions, as discussed in the following sections.

2.2 Context

The notion of context is used to describe where an operation can be invoked and where it causes perceivable effects. In MC2, a context comprises the characteristics of *users* and of *situations of use*. In multi/cross channel web applications the situation of use often concerns just devices and their technological features. Depending on the application domain, it may involve additional aspects related to location (in its geographical or logical characteristics), time, etc. [4]

The MC2 operations model requires the definition of a Context Model, but it does not prescribe a specific context model. Since the modelling primitives exploit the distinction between user characteristics and situation of use, MC2 only prescribes that the chosen Context Model supports this distinction. Using the UML terminology, we can say that MC2 provides a Context Meta-model, comprising the User and the Situation of Use meta-classes. The actual classes describing the characteristics of the User and of the Situation of Use in a specific application, are defined by the designer according to the requirements of the actual application.

The following diagram provides an example of the Context Model that we will adopt for discussing the examp les in the rest of this paper.

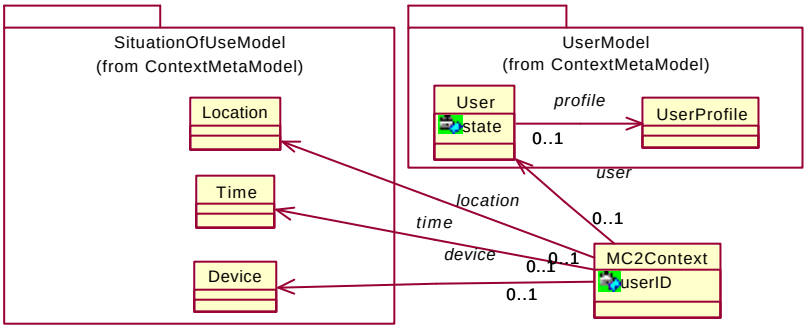

Fig. 1. An example of Context Model

2.3 Hyperview

A multi-cross channel web application should provide different user experiences to different users in different situations of use (e.g., devices, locations, etc.). To model the customisation capability of web applications, MC2 associates contexts to *hyperviews*. The notion of hyperview extends the concept of view in database, which can be "roughly defined as an interface between the user and the data stored in the data base which provides the user with a specific way of looking at the application" [7]. Differently from database views, a hyperview has a hypermedia nature: it involves not only information aspects, but also navigation and presentation elements.

If we model operations from a user perspective, pre and post conditions will predicate about properties of the various hyperviews and not about the entire application. Thus it is intuitive that the design of operations requires primitives concerning hyperviews, i.e., it

requires a Hyperview Model. MC2 is largely independent from the specific hyperview model chosen by the designer[1], but prescribes some requirements on its primitives:

- As for data base views, there must be a clear distinction between *hyperview definition* and *hyperview state*.
- A hyperview definition describes the *design* of the application in a specific set of contexts. It must include a *hyperview design schema* and a *materialization function*[2].
- Hyperview design schemas are associated to contexts: given a context (instance of the chosen Context Model), there must be only one hyperview design schema, which corresponds to that context (while the opposite is, in general, false.)
- An Information Schema, a Navigation Schema, and a Presentation Schema must compose the hyperview design schema. They describe, respectively, the information types, navigation structures (nodes and links), and presentation structures (pages) that are available in a given context (i.e., for a given user in a given situation of use). Information, navigation, and publishing schemas are not independent: the navigation schema must be complemented with a *navigation mapping* which defines how information structures are mapped into node and link structures; the presentation schema must be complemented with a *presentation mapping* which describe how nodes and links fit into pages. With these assumptions, any update to the hyperview state can always be expressed as an update on its information structures only.
- The materialization function defines the dependency among hyperview states and application states. A *hyperview state* is an *instantiation* of the hyperview schema, and an application state is an instantiation of the application "schema". The latter is defined by the state of all "objects" which describe the application from a system perspective. Therefore the materialization function specifies how we obtain a hyperview state from a given application state. The model for defining application schemas is chosen by the application designer when he defines his specific hyperview model[3].

The MC2 operation model allows designers to specify operations independently from how hyperview schemas are defined as far as they satisfy the above assumptions. In fact, the semantics of operation primitives relies upon them, as we will discuss in the next section.

3. Designing Pre and Post Conditions

The concepts of context, hyperview, and hyperview state, allow us to identify some design dimensions that can help designers to structure pre and post conditions.

---

[1] This is similar to databases, where general properties of view update properties can be established using an abstract view model [8].

[2] In relational database views, the hyperview schema is described by a relational schema, and the materialization mapping is described by a query on the underlying data base [7]

[3] This is similar to databases, where any view specification language relies upon a specific data base model [7].

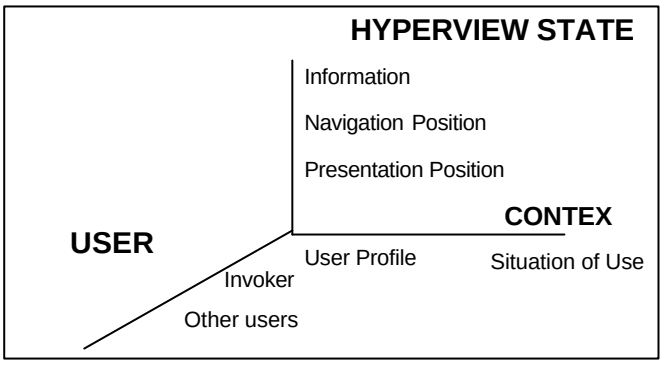

Fig. 2. Design Dimensions for MC2 operations

Two design dimensions concern context and hyperview state. Each one is further decomposed in a number of sub-dimensions: for Context, User profile and Situation of Use; for Hyperview State, Information, Navigation Position, and Presentation Position. The Information sub-dimension in Hyperview State concerns the information elements of a Hyperviews State, i.e., the instances of the Hyperview Information Schema. We do not need to consider the instances of the Navigation and Presentation Schemas since, by assumption, they can be "derived" from Navigation and Presentation Mappings of a Hyperview Definition, referred to the invoker position in the hyperspace of the hyperview state. The Navigation Position, and Presentation Position sub-dimensions refer to the invoker position in the hyperspace of a given hyperview state.

Since our design focus is the user, we need to introduce a User Dimension, and to distinguish two main perspectives under which an operation can be designed: the perspective of the user who invokes the operation (hereafter called *invoker*) and the perspective of *other users* working with the application.

In the rest of this section, we will discuss how designer can use the MC2 design space to reason on how to describe pre and post conditions.

Let us start with pre-conditions, considering the invoker perspective. In our model, an operation can be invoked in hyperviews which are associated, via their design schema, to a specific set of contexts (e.g., given devices, user types, etc..). Thus the first question for the designer of an operation could be: "*What are the characteristics of the invoker and of his situations of use?*" The second question concerns the actual hyperview state where the invoker invokes the operation: *"What are the constraints on the information provided by the "current" hyperview state?"* (For example, an item must be in stock in order to "buy" it.) The third question is related to the position where the invoker is located in the hyperspace corresponding to the current hyperview state: *"On which nodes or pages can the operation be invoked? Is there any constraint on the presentation state of page elements"* For example, the user can only "put in the shopping bag" an item from a page containing a "Product" node with the product description. If this product is a video and the page offers a video preview, the "add" operation cannot be invoked while the video is being played.

If we now consider (again for pre-conditions) the "Other Users" perspective, the designer could reason about contexts, wondering: *"Are there any constraints for the*

operation execution that involve users with different characteristics and/or situations of use from those of the invoker? " For example, the designer may want to avoid a user in a forum sending an individual chat message to another user who is not "connected" at the time he starts writing the message. Similarly, the designer should reason about hyperview states of other users *"Are there any conditions that are specific of the hyperviews of other users"?* For example, the designer may want to avoid a mobile phone user sending a voice message to another mobile user if his voice mailing box is "almost full", i.e., it has space for one extra message only.

Let us now consider operation effects, i.e., post-conditions, again starting from the invoker perspective. An operation may change the context, so that after the operation execution the user may find him in a context different from the one where the operation has been invoked. Thus the designer should wonder: *"After the operation execution, does the user find himself in the same context as before the execution, or in a different one? If different, what is changed (the user profile, the device characteristics or other features of the situation of use)?* After an authentication operation (submit login and password, for example), the user state in his profile becomes "authenticated". Notice that if the new context is associated to a different hyperview schema, the user may also be located automatically on a different hyperview state (corresponding to the different hyperview schema). But this effect is implicit, and does not require any specification by the designer.

When we consider hyperview states for the invoker, we should distinguish between *local* and *non-local* hyperview states. The local hyperview state is the one where the user finds himself after executing the operation. For the reasons discussed above, it may correspond to the same hyperview schema or to a different one, depending whether the context has been changed or not by the operation. The modifications of the local hyperview state may concern the information state, or the invoker position (or both aspects). For example, "add to chart" affects the local hyperview state (without modifying the context) since a new product is added to the user shopping chart, but may also "move" the user to the page showing the updated shopping chart (as it happens for example in www.amazon.com). Non-local hyperview states are associated to contexts for the same user, but in different situations of use. The designer should wonder: *"Will the invoker perceive operation effects when he will use a different device or, in general, he will be in a different situation of use?"*

If we now consider "other users", the two main designer's questions are: *"Does the operation affects the context of other users, and/or their hyperview states?"*

4. Specification of Operations in MC2.

In this section we discuss how all the concepts introduced in the previous sections can be formally specified using OCL (Object Constraint Language), the formal language offered by UML for expressing constraints and invariant conditions on the system being modelled.

4.1 The MC2 reference model for specifying operations.

The following diagram describes how the various primitives involved in the MC2 Operation Model are mutually related.

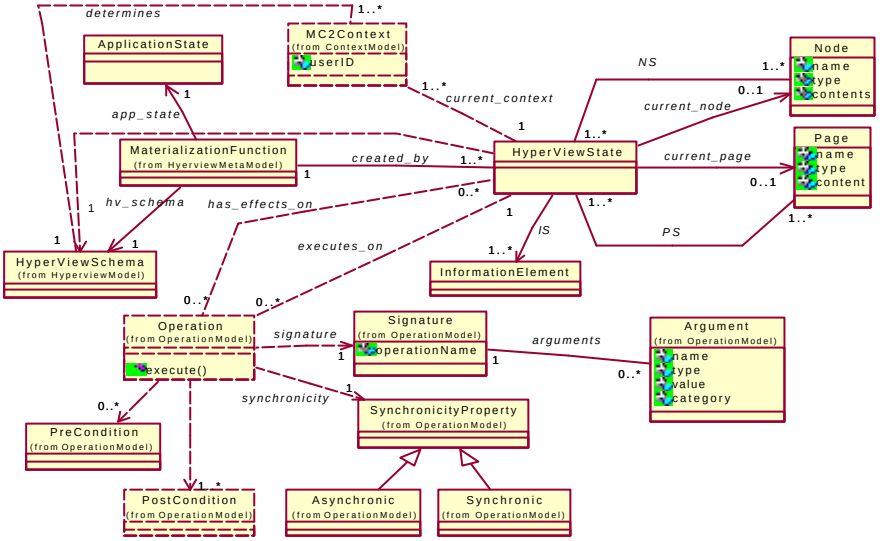

Fig. 3. The MC2 reference model for specifying operations.

We will use the UML class diagram in figure 3 as a "reference model" to define the OCL expressions needed to describe pre and post conditions. We extend the standard OCL package with some functions, which can be used by the designer as macros both in pre and post conditions: they simplify the operation specifications and make them more readable.

They concern the *current hyperview* – which defines the hyperview state where the invoker finds himself before or after the operation execution, and the *current node* and *current page* where the user is located before or after the operation execution. These extensions, defined below, are prefixed with *MC2cl* (that is, MC2 OCL) to distinguish them from reserved OCL terms or other terms mentioned in the class model of figure 3.

- `MC2clCurrentView(c:HyperViewContext):HyperViewState`

  `MC2clCurrentView (c) ≅`

  `op.executes_on` **and** `op.executes_on.current_context = c`

In the "context of operation op*", this function returns the hyperview state for context* c.

- `MC2clCurrentNode(): Node`

  `MC2clCurrentNode ≅ op.executes_on.current_node`

In the "context of operation op*", this function returns the node where the user is located*

- `MC2clCurrentPage(): Page`

  `MC2clCurrentPage ≅ op.executes_on.current_page`

"In the context of operation op this function returns the page where the user is located"

4.2 Examples of operations specifications

In this section, we exemplify how operations can be modelled using the model presented in section 3, providing the OCL specification of two operations.

4.2.1 Confirm Reservation.

This example is a scenario where the user can buy tickets (e.g., for a theatre ticket, a sport match, etc.) first by reserving them and then by confirming the purchase (in order to fire the ticket emission procedure). In the following OCL specification, we describe the operation of confirming a reservation. We suppose that the user has made the reservation and now wants to confirm the reservation. Its effects, from the user perspective, include: changing the reservation state to "confirmed", and sending to the user mobile an SMS message, reminding him to pick them up the ticket at the box office no later than a given time on the performance date. This is an example of a cross-channel operation affecting two different channels of the invoker.

```
Context ConfirmReservation::execute()
   inv:     self.signature.operationName = "confirm_reservation"
            Let arg1:Argument = arg1.name = "sms",
                                arg1.type = "Text",
                                arg1.category = "system-provided"
                      "The sms message is generated by the system."
                                arg1.value -> oclIsTypeOf(Text)
            self.signature.arguments -> includes(arg1)
            Let arg2:Argument = arg2.name = "res",
                               arg2.type = "Reservation",
                               arg2.category = "user-provided"
                               arg2.value -> oclIsTypeOf(Reservation)
            self.signature.arguments -> includes(arg2)
            self.synchronicity = SynchronicityProperty::synchronous
   pre:
            Let C: HyperViewContext = HyperViewContext.allInstances ->
                  any(C.device = PC or C.device = Mobile)
            MC2clCurrentView(C) and
            IF (C.device = PC) THEN
                MC2clCurrentNode = PCReservationIndex and
                PCReservationIndex.type = "PCReservationIndexType" and
                PCReservationIndex.contents -> includes(arg2.value)
            IF (C.device = Mobile) THEN
                MC2clCurrentNode = MobReservationIndex and
                MobRservationIndex.type = "MobReservationIndexType" and
                MobReservationIndex.contents -> includes(arg2.value)
   post:
            "For this first post condition, we assume that if the current
            view is not specified, then is the same as for the pre
            conditions hyper view."
            self.executes_on.IS.Reservation.allInstances[arg2.value].state
            = "confirmed" and
            MC2clCurrentPage = HomePage and
            HomePage.type = HomePageType
            Let C':HyperViewContext = HyperViewContext.allInstances ->
                  any(C'.device = Mobile and C'.userID = C.userID)
            MC2clCurrentView(C') and
            [4]self.executes_on.IS.smsList -> includes(arg1.value)
```

[4] This is an example of a post condition affecting "other context" of the invoker's hyperview state.

4.2.2 Submit Review.

This last example consider an operation which updates the hyperview state both of the invoker and of other users, and where side effects on other users hyperviews must be explicitly specified by the designer. The precondition must predicate explicitly about the user profile, since a Program Committee Member can submit the review of a conference paper only. The fact that a Program Committee member can review only the papers assigned to him can be left implicit, since it is derived from the context property of being "Program Committee Member". The operation affects the invoker's hyperview state (adding a new review to his list) and the hyperviews of other Program Committee members, but only if they have submitted a review for the same paper. This is an example of a cross-context operation affecting the invoker and other users.

```
Context SubmitReview::execute()
   inv:  self.signature.operationName = "submit_review"
            Let arg1:Argument = arg1.name = "p",
                                arg1.type = "Paper",
                                arg1.category = "user-provided",
                                arg1.value -> oclIsTypeOf(Paper)
            self.signature.arguments -> includes(arg1)
            Let arg2:Argument = arg2.name = "r",
                                arg2.type = "Review",
                                arg2.category = "user-provided"
                                arg2.value -> oclIsTypeOf(Review)
            self.signature.arguments -> includes(arg2)
            self.synchronicity = SynchronicityProperty::synchronous
   pre:
           Let C:HyperViewContext = HyperViewContext.allInstances ->
         any(c | c.device = PC and
                 c.user.state = "authenticated" and
                 c.user.type = "PCMember")
           MC2clCurrentView(C) and
           MC2clCurrentNode = ReviewsIndex and
           ReviewsIndex.type = "ReviewsIndexType"
   post:
           self.executes_on.IS.Paper.allInstances[arg1.value].reviews ->
           includes(arg2.value) and
           IF (self.executes_on.IS.Review.allInstances -> exists(r|
               arg1.value.reviews -> includes(r)))
           THEN
               self.executes_on.IS.Paper.allInstances[arg1.value].
               reviews -> includes(r)
            Let C':HyperViewContext = HyperViewContext.allInstances ->
                  any(c | c.user.type = "PCMember" and
                          c.device = PC and
                          c.userID ≠ C.userID)
           MC2clCurrentView(C')
           IF (self.executes_on.IS.myPapers -> includes(arg1.value) and
               Review.allInstances -> exists(r |
               self.executes_on.IS.myReviews -> includes(r) and
               self.executes_on.IS.Paper.allInstances[arg1.value].
                    reviews -> includes(r)))
           THEN
               [5]self.executes_on.IS.Paper.allInstances[arg1.value].
```

[5] This is an example of a post condition affecting the hyperview state of "other users".

```
reviews -> includes(arg2.value)
```

5. Related Works and Conclusions

The problem of designing complex services in multi-cross web applications can be affronted with different perspectives and levels of abstraction. Usually current design methods front the problem from a technologic and implementation viewpoint. We have instead addressed services design from a complementary perspective, adopting a user-oriented point of view, which abstracts from any implementation aspect. In this paper, we have focused on the conceptual modelling of operations, which are the building blocks for designing services, proposing a design approach which explicitly takes into account two key features that characterize multi-cross channel web applications: the hypermedia nature and the role of the context where the application is used.

Recently, the issue of modelling hypermedia and operational features in a unifying framework has been acknowledged by the web engineering community as relevant for design. An attempt to address both kinds of features in a UML framework is Conallen's work reported in [9]. This approach adopts an implementation perspective. It aims at proposing a workable solution for *implementing* web applications; it privileges client-server interactions and architectural issues, and models web applications in terms of client and server pages, links, applets, frames, etc. A number of "pure" hypermedia models have been extended in order to include operation design. The latest versions of OOHDM [10] and OO-H [11], which both use standard "objects" to describe nodes, include operations as attachments to navigable hypermedia elements. ADM-2 (the latest version of the Araneus Data Model) [14] extends the notion of “link” to model both navigational and non-navigational “user actions”, where a non navigational user action determine either a database update, or a transition from a “phase” of the system to another one, or an invocation of an external module. WebML [13] introduces the concept of “operational unit” to allow designers include visual elements on the pages associated to insert-delete-update operations. The latest version of WSDM [12] introduces "functional modelling" as an explicit task within conceptual modelling (complementary to information and navigation modelling). It decomposes complex operational activities into elementary operations, modelled in terms of some basic operational primitives (e.g., " input data", "add/remove information elements").

To our knowledge, the MC2 approach discussed in this paper is the first attempt to provide a conceptual modelling framework for web operations which integrate hypermedia aspects and context related aspects, both needed in multi-cross channel web applications. MC2 models operations as first order objects, and the operation model relays upon a context model and a hyperview model. The invocation constraints of an operation (modelled as pre conditions), and the operation effects (modelled as post conditions) are described as properties of contexts and hyperview states, where hyperview states comprise information, navigation, and presentation structures and provide a customized view of the application which is tuned to the needs of a specific context.

The true complexity is to specify post conditions on hyperview states, since an operation may trigger up side effects that either a user may perceive when working in a different situation of use, or other users may perceive. The number of hyperview states the designer needs to consider is potentially exponential. Still, not all side effects must be specified explicitly. We must consider that hyperview states result from the materialization of a common, shared application state. If two hyperview states, which

correspond (in the same point of time) to different contexts where the operation can be invoked, share a portion of the hyperview state, and the operation invocation is constrained by properties of the shared portion, these properties need to be stated explicitly for one hyperview only, since they implicitly hold for the other one.

In a similar way, many operation "side effects" (i.e., effects that the invoker may perceive in a different situation of use, or that other users can perceive) can be left implicit. The specification should include only the effects on the current hyperview where the invoker is located after operation execution, and side effects on those information, navigation, and presentation structures that are not shared by the current hyperview.

Some future work lines we have identified are:

- Studying how operation specifications from a user perspective can be translated into operations specifications from a design perspective.
- Integrating the operation model into a service model. Differently from operations, services may have a transactional nature.

7. References.

[1]. G. Booch, I. Jacobson, and J. Rumbaugh: "The Unified Modeling Language User Guide", The Addison-Wesley Object Technology Series, 1998.

[2]. Object Management Group. Object constraints language specification, February 2001.

[3]. L. Baresi, G. Denaro, L. Mainetti, and P. Paolini. "Assertions to Better Specify the Amazon Bug" In Proceedings of 14th International Conference on Software Engineeering and Knowledge Engineering. Ischia (Italy), July 2002.

[4]. G. Kappel, W. Retschitzegger, W. Schwinger, Modeling Customizable Web Applications - A Requirement's Perspective, International Conference on Digital Libraries: Research and Practice (ICDL), Koyoto, Japan, November 2000.

[5]. Deliverable D7: "Hypermedia and Operation Design: Model, Notation and Tool Architecture." http://www.uwaproject.org/

[6]. Deliverable D9: "Customization Design: Model, Notation and Tool Architecture." http://www.uwaproject.org/

[7]. F. Bancilhon, N. Spyratos, "Update Semantics of Relational Views" ACM Transactions on Database Systems, Vol. 6 No. 4, December 1981, Pages 557-575.

[8]. G. Gottlob, P. Paolini, R. Zicari, "Properties and Update Semantics of Consistent Views" ACM Transactions on Database Systems, Vol. 13 No. 4, December 1988, Pages 486-524

[9]. J. Conallen: "Modeling Web Application Architectures with UML", Communications of the ACM, 42:10, Oct. 1999, pp. 63-70.

[10]. Schwabe D., Rossi G, "Developing Hypermedia Applications using OOHDM". In Proceedings of Workshop on Hypermedia Development Process, Methods and Models. Hypertext 98. 1998

[11]. C. Cachero, J. Gómez, O. Pastor. "Object-Oriented Conceptual Modeling of Web Application Interfaces: the OO-HMethod Presentation Abstract Model". Greenwhich, London September 2000. ECWEB'00, LNCS 1875, Springer-Verlag

[12]. O. de Troyer, Sven Casteleyen. "The Conference review system with WSDM." 2001. IWWOST http://citeseer.nj.nec.com/detroyer01conference.html

[13]. Bongio, S. Ceri, P. Fraternali, A. Maurino. "Modeling Data Entry and Operations in WebML". Proc. WebDB 2000, Dallas, TX, USA, May 2000.

[14]. Paolo Atzeni, Alessio Parente. "Specification of Web applications with ADM-2 Specification of Web applications with ADM-2 – Schemes" 2001. IWWOST, Valencia, Spain.