A Stakeholders Centered Approach for Conceptual Modeling of Communication-Intensive Applications

Vito Perrone
HOC (Hypermedia Open Center)
Politecnico di Milano, Italy
perrone@elet.polimi.it

Davide Bolchini
TEC-Lab
University of Lugano, Switzerland
davide.bolchini@lu.unisi.ch

Paolo Paolini
HOC (Hypermedia Open Center)
Politecnico di Milano, Italy
paolini@elet.polimi.it

ABSTRACT
To be successful, any engineering product should accomplish the needs and expectations of its potential stakeholders. Similarly, design models should be defined taking into account goals and requirements of their users, i.e. the practitioners who daily conceive, develop and deploy applications. Neglecting stakeholders' needs can bring to lack of attention towards these engineering products (design models) while fitness to requirements can drastically increase their acceptability in the real world. This paper focuses on the domain of Communication and Interaction Intensive applications (C&II applications) by presenting a suite of two conceptual models (namely IDM and E-WOOD) belonging to a more comprehensive methodological framework addressing the analysis and design of such a kind of applications. The focus of the paper is not on the presentation of the methods but on highlighting their fitness to the requirements of the potential adopters of such methods. To this end, the overall framework has been defined on the basis of an accurate analysis of potential stakeholders' goals and requirements gained from our training experience to professional designers and from adoption of our previous conceptual methods in several real-life projects.

Categories and Subject Descriptors
D.2.10 [Design]:*Methodologies;* D.2.1 Requirements/Specifications: *Methodologies*

General Terms
Design, Documentation, Human Factors

Keywords
Web application design, design usability, documentation usability, UML, design requirements.

1. INTRODUCTION
Industrial stakeholders are still reluctant to use academic models and methodologies for the analysis and design of interactive applications [[1]]. As a matter of fact, they do not see (or are not educated to see) the actual benefit (if any) of the proposed models (and the corresponding methodologies) developed within the academic research arena.

SIGDOC'05, September 21-23, 2005, Coventry, United Kingdom.

This is particularly evident in the domain of Communication and Interaction Intensive applications (C&II applications). These are applications (including the most part of complex web applications) whose main asset is the content to communicate to the users. Cultural-heritage web sites, educational web sites, institutional web sites, promotional and corporate web applications, and even a large part of e-commerce web sites are just a few examples of domains in which sites are designed first and foremost as means to communicate content and also as a tool for accomplishing operations and transactional tasks. In such domains, stakeholders need to address communication goals, i.e. they wish to use the site to get across structured messages and content to a variety of users. In turn, potential users have their own goals with the respect of the application; they expect to find usable site architecture by which learning to be engaged, to retrieve information and execute operations and processes.

In spite of the abundant offer of models and methods, coping with the analysis and design phases within the overall lifecycle of these applications, various factors hinder the adoption of systematic approaches for modeling. These factors can be summarized according to some important criteria offered by the Diffusion Theory [27]:

a) cost/benefit ratio is not clear. Costs involved in learning the methods, using them efficiently, properly training project teams, and granting the transfer of knowledge across projects are often considerable and not justifiable with respect to the actual impact on the quality of the final application. Success cases have to demonstrate that the use of a given design was the main driver for the quality of the application.

b) Complexity. Even before coping with the cost issue, conceptual tools appear too hard to understand and frighten the potential adopters [[14]] because they are overly complex. Usability (which surely facilitates adoption) should replace expressiveness (which do not guarantee quality design).

c) Triability. It is often the case that methods and models are offered and presented as "one block", without providing the conditions and the opportunities to "try" them on a limited basis before adoption. If it works (i.e. if only a small subset of the method is worth the effort) then other components can be easily adopted.

d) Relative advantage. Why should a stakeholder adopt a given method instead of another, among the dozens available in the research arena? Are the scopes and the modelling boundaries of each method clearly defined? Are there comprehensive comparative studies or experience-based ranking of the methods available? This information is important for stakeholder to decide upon which method to invest in. The lack of this information fail

to attract stakeholders towards academic method: "there are so many, where should I start?"
Modelling conceptual tools, especially those dedicated to the design phase should possess a number of essential requirements to be effectively and efficiently adopted by professionals on the field. **Lightweight** design processes and **usability** are being recognized as key factors for the success of a design method. Successful conceptual tools should be:

- **easy to teach** to anyone (from students to practitioners). Professionals, especially, do not have time and resources to invest for learning new methodologies; one of the success factors of "Entity Relationship" (probably the most successful design model, ever) stems from the fact that it was very easy to transmit its basic concepts, both in academia and professional environment;
- **effective for brainstorming**, i.e. for generating and discussing ideas among developers, with stakeholders, and with potential users. It is of little use to have a design model capable of representing only fully developed solutions;
- **requiring little time to write down design ideas**: analysts and designers do not like to spend too many resources in preliminary activities;
- enabling to move, smoothly, **from a general design, to more detailed** design, without need for excessive reworking and without need for completeness;
- enabling a **intuitive mapping between analysis, design and implementation artefacts**. Implementers often complain that design specifications are hard to code so they are forced to do as their thinks fit.

These factors are considered in our stakeholder-centered approach, where design models are crafted taking into account the needs, goals, roles, attitudes and expectations of the stakeholders of the model. By stakeholders in this case we mean those who can gain direct benefit from the adoption and the use of the modelling method, being them the developers, the designers, the project managers, or other relevant actors.

This paper introduce our so called stakeholder-centered approach for the development of C&II applications focusing on a suit of two conceptual modelling tools (namely IDM and E-WOOD) which try to best correspond to the needs of the different stakeholders of a design model. The features of these design models (one focussing more towards analysis and the other more towards detailed design) represent an important step forward towards the deployment of modelling techniques that can be efficiently and effectively adopted by practitioners.

The remainder of the paper is as follows. Section 2 discusses the relevant achievements on the field of design models and methods for the development of C&II applications. The scope of this section is not to encompass an exhaustive bibliography, but rather to quote some of the several approaches from where we "borrowed" ideas. Section 3 illustrates the elements of a stakeholder-centered approach to design models, discussing the requirements for a modelling method supporting the analysis phase and for one supporting the proper design phase. Section 4 demonstrates how the proposed models are a "good enough" solution to correspond to the requirements of the stakeholders and explains the essential features of the models through some examples. Conclusions and future works are discussed in Section 5.

2. APPLICATION DESIGN: RELATED WORKS AND MOTIVATIONS

Along the last ten years a number of methods have been proposed for supporting the design of Web applications. In this section we briefly resume the main characteristics of these methods from two perspectives – the academia and the industry – and considering their role in the analysis and design phases.

Looking at the academic community, some of the most known existing methodologies are HDM [[4]], W2000 [[18]], OO-HDM [[6]], WebML [[8]], UWE [[9]], WSDM [[7]], OO-H [[11]], and many others. Roughly speaking, they specify the design of a Web application at the conceptual level, neglecting technological aspects and constraints. Besides technical (minor) differences, these methods share lots of common features. All of them are based upon an information-navigation paradigm to describe the user interaction, recognize the importance of the semantics as guidance for conceiving the application design and share the fundamental principle of *separation of concerns.* On the other hand, they differ in terms of the proposed design primitives, notation and support tools.

Let us consider the key features of one of these methods (W2000), arguing that the considerations valid for it can be easily generalized to many of the existing approaches. According to W2000 [[18]] terminology, the design of a Web application is divided into four dimensions: *Information and Access Structures design*, defining the basic conceptual information units (entities) as perceived by the user, the navigational infrastructure in terms of semantics associations between entities, and access structures (navigational paths enabling users to locate and reach the content of interest); *Operations and Business Process design,* defining operations (e.g. "add to shopping cart") and processes (e.g. "check-out", "registration"); *Navigation design,* defining the navigation network allowing users browse information and access structures and execute operations and processes; *Presentation design,* defining the page structure in terms of lay-out aspects and graphical elements and the page organization and navigation.
If properly used, current academic methods unleash the potential of enabling designers to conceive high quality (usable and effective) applications. However, these methods suffer, of some deficiencies which contribute to a poor acceptance from the industrial environment [[1]],[[3]]. These limits can be summarized as follows: (1) providing sophisticated and semantically rich primitives often *takes too much effort and time to learn and start using the methods*; (2) *modelling purpose is only badly or vaguely specified* with the respect of the overall development process. It is often claimed that models are intended as support tool during the early analysis activities, but then these models are also used to automatically generate the running application [[11]] [[8]]; (3) *cumbersome design documents* are generally produced as output of the design activities. These documents risk being hard to read and use both during the analysis activities and the following implementation ones; (4) *proprietary concepts and notations* are generally proposed (except a few cases like [[9]]) by each method, thus increasing the learning time and the consequent negative perception of practitioners [[14]]; (5) *ad-hoc and in-house made support tools* are generally proposed instead of commercial ones.

With regards to the methods proposed by the industrial world, UML [[13]] is definitively considered the standard de-facto in the design practice. Referring to the Web application domain, the only recognized method coming from the industrial environment is the one proposed by Conallen in [[10]]: the Web Application Extension (WAE). WAE, like other UML native methods, adopts

an implementation oriented approach, in that most of the modelling primitives directly abstract from concrete implementation artefacts[1]. Due to this characteristic, they are quite easy to understand and use by technicians for supporting the software design activities and broadly supported by commercial tools. On the other hand, concerning C&II applications, it is known [[12]] that UML lacks of proper semantics for supporting the design of communication and navigation aspects both during the analysis and design phases.

Finally, the topic of explicitly considering stakeholders and their requirements for shaping a suitable design method has been barely fronted by existing approaches. In most of examined literature when a new modelling method is proposed, the well-known and high level software engineering principles are, at most, cited. For example in [[12]] it is argued that the next generation of OO methods "*…should be sufficiently user-friendly to all kinds of possible stakeholders. That is, for all stakeholders of any model, its relevant parts expressed in the modeling language, must be understandable, must be clear even. For the modeler as well as for all other persons involved in the modeling activity, any model must be expressive, precise and clear as well*". However, besides these well known software engineering principles, we also advocate that, due to the diversity of all possible stakeholders, the lack of an explicit consideration of what every potential stakeholder expects by the modeling method could be one of the main reasons of the existing gap between current proposals and industry practice.

3. STAKEHOLDERS CENTERED CONCEPTUAL DESIGN

In [[16]] Mylopulos discusses about the role of conceptual models in the system development lifecycle observing that they provide semantic terms that are used both to support reasoning in the analysis activities and to define a user oriented solution in the design activities. In particular, they act as a bridge towards a more technical design of the system, usually known as *logical design*. It is thus clear that they play a central role in the whole software engineering life cycle. Given the scenario sketched above, we defined our approach for the conceptual modeling of C&II applications starting by a fine analysis of stakeholders and their needs for both the facets of a conceptual model, that is, analysis tool and bridge towards the software design and implementation. The next two paragraphs describe the result of our analysis.

3.2 Requirements for a modeling method in the analysis phase

After a requirements analysis activity, where the needs and the goals of the various stakeholders of the application have been elicited and gathered, projects should smoothly to a highly-iterative phase in which possible design ideas are devised, negotiated and discussed at the light of the requirements.
As better explained by the next section, this phase lays between the traditional "requirements analysis" and "design" activities, as it is an activity needed for discovering and assess requirements but, at the same time, it features creativity and generation of ideas typical of design activities. We will call this phase "Requirements Design". It is also in fact different from the proper "design" phase, where detailed solutions are refined, specified and consistently organized to provide a structured input for the technical implementation.

What conceptual model can support this activity? To start answering this question, we firstly have to understand who the main stakeholders of such a modeling method are. In the following, different types of stakeholders for such "design" model are illustrated. These are intended to be "roles" of possible stakeholders or users of the design model. Therefore, a given person in the same project may play one or more of the following roles.

Analysts: are responsible for the traditional activity of "requirements analysis". They have to manage the proper elicitation, organization, specification, refinement and analysis of the requirements for the application. Given the volatile nature of the requirements, and being no clear-cut boundary between requirements and design, they need to ***see the impact of their requirements on the actual design of the application with little effort and time***. The modeling method should enable to *quickly turn ideas into possible solutions (A1)*. Models should be suited to *support communicating these ideas among other analysts and client counterparts (A2)* and *stimulate the discussion of both ideas and possible alternative solutions (A3)*. Moreover, *relationships between early design decisions and requirements should be mastered (A4)*.

Project Managers: are in charge of monitoring the proper development of the project. They should be able to ***master the requirements and design picture*** in order to easily infer the expected costs and effort needed for the application management and enhancement over time. They needs for a method that *allows tracking decisions performed passing from the analysis to design activities (A5)*. Moreover, to effectively master the whole development activities, the method should also enable to *pass from the whole picture to single details (A6)*.

Decision Makers (Opinion Makers): represent that part of the client (external to the clients but able to influence its decisions) who have the actual decision and contractual power on the project (who may understand better the client organization's work and needs and thus influence the project development). They, or their delegates, ***are the primary communication partners of the project team*** (represented by analysts or designers) should be involved, on a regular basis, all along the project the lifecycle. All these stakeholders are typically non technical thus the models should *communicate ideas, scenarios, goals and requirements (A8)* using concepts *easy to understand by non computer experts (A7)*. Being these stakeholders responsible of the system quality and effectiveness with the respect of the whole client organization, they also need to *get a clear, even if in-the-large, picture of the main communication and operative features (A9)* of the application and to be able to *intervene on the design to make suggestions (A10)*.

Domain/Content Experts: represent an important source of knowledge and expertise about the specific topics, contents and services the application is supposed to offer. Communication strategies are significantly affected by the content that is either available or producible; therefore domain experts can be crucial to shape the communication effectiveness, to understand the levels of details by which the content has to be conveyed, and the users to whom this content is addressed. Being the content still the most important asset of such a kind of applications, domain experts are important partners in the analysis and design activities because

[1] Examples of WAE's primitives are server side and client side pages, applets, Java Scripts, ActiveX controls, frames, etc. They are obtained by stereotyping UML classes.

they may be those who will actually provide the contents to the project team. In particular, in this phase they need models that can *provide a clear idea of the quantity and quality of the content to be produced and managed (A11)*.

Designers: are in charge of system design but in this phase they work closely with analysts on the bridge between requirements and design. Their main goal is to *master the complexity of the application design and to anticipate its impact on the subsequent development activities (A12)*.

3.2 Requirements for a modeling method in the design phase

Conceptual models are used at the beginning of the overall *design* activities, as intended in the software engineering discipline, which will finally lead to the detailed specification of the software modules to be coded. In this phase, the main goal of conceptual models is to clearly define the solution (application to-be) characteristics, even if still avoiding implementation details. Models should be well structured and their primitives should gain concreteness. Some of the previous stakeholders are still present in this phase, even if their goals can change, but newer must be considered.

Designers: are, as said above, in charge of the system design. Depending on the reference community, the terminology adopted within a company, the kind of application, and so on, different professional figures (e.g. information architects, interaction and usability experts, and so on) might be attributed to play this role. Usually several designers work both in the analysis and design phases thus first goal is to ***ease the communication with the analysis activities and among different designers in the design activities***. For the former, some form of *guidance* should be provided *to support the passage from the early solution devised in the analysis activities to the actual design of the system (R1)*. This *mapping should compromise between rigour* – to enable some form of automatic passage – *and flexibility* – to not constraint choices designers have to perform in the design phase *(R2)*. In this phase, they have to design models very close to the application to-be, thus inevitably these models are rich in details and the specification is often composed by several heterogeneous diagrams representing different application concerns. To master the overall design complexity (avoiding naive designers feel lost) the method should *provide an explicit framing strategy (R3)*. Furthermore, model drawing is time-consuming activity that needs proper tool support. In order to be used in professional environments, support tools should adhere to the commercial standards. Since building such tools is an expensive activity, new modeling methods should be defined so that *existing commercial tools can be exploited (R4)*.

Usability experts and Graphical designers*:* depending on project parameters like those mentioned above, these roles could be attributed to designers or other professionals with non technical skills. However, in C&II applications these aspects are taking more and more importance and require specific competences. Whatever is the case, these figures are interested in carefully ***defining and reviewing usability and graphical aspects*** of the application to-be, thus *concerns impacting usability and layout/graphical aspects should be explicitly modeled and made easy to access (R5)*. These experts are used to analyze and discuss about usability and graphical concerns by means of mock-up or other similar representations that closely reproduce the application to be. Thus, to achieve an effective communication with usability and graphical experts, *models should* also *look as close as possible to the actual application (R6)*.

Software designers and Implementers*:* define and implement the software modules that will actually realize, on the basis of the chosen system architecture, the application specified by the conceptual models. From our experience on the field, a recognized lack of ***existing conceptual models*** is that they ***require a considerable effort to be mapped into software artifacts***. Often, it is hard to understand which diagrams should be considered for obtaining a single software artifact and, most of times, several different diagrams must be composed. For example in the web domain, to design a server page, software designers have to refer to information models for the page data, operation and business process models for the business logic, navigation models for the navigation logic and presentation models for graphical and layout aspects. Software designers consider this activity being time consuming and, if not properly supported by tools, a possible source of mapping mistakes. On the basis of these considerations, models should *embody modeling primitives as closer as possible to concrete counterparts (R7)* and that *as less as possible diagrams should be considered to define a software component (R8)*. Also the *design documentation to be used for supporting the implementation activities should be contained and easy to read (R9)* (many cross-reference are considered highly annoying). Another highly desirable feature a modeling method should own, for these stakeholder types, is to *provide predefined mapping strategies (mapping patterns) towards the most known architectural patterns (R10)*. Finally, most of the interviewed software designers and implementers were already used to the UML and related CASE tools, thus they showed a remarkable preference in having conceptual models described in UML-like notation and following the UML philosophy, that is, modeling methods should *belong to the UML family (R11)*.

Product manager: this stakeholder type represents the most important client counterpart dealing with the application design, and act as interface of decision makers, opinion makers, clients and content/domain experts (described above). Product managers are usually in charge of assuring the envisioned application will be able to satisfy the client company expectations, but they also are responsible of a number of other specific tasks. Among others, one the most important is to set up the editorial chain. Their main, somehow opposite, goals are *to take the control of the overall application at a glance* and *to get details of specific aspects (related to their tasks)*. Desirable features for the method should be to *review models at different levels of detail (R12)*, to *embody most of the needed information to set up the editorial chain* (R13) and to *enable some form of requirements tracking (R14)*.

Final Users*:* this stakeholder category is the more important for tuning the application interaction even if it is also the less accessible for several reasons. In fact, they usually are not part of the client, are barely identifiable and their characteristics can vary remarkably. Nevertheless, gathering some feedback from potential users before the coding activities start can bring several advantages since modifying models is much less expensive than modifying code. From our experience [[24]], a discussion with users mediated by models is usually ineffective because they need to see and handle application as it were running. Application prototypes are much more effective in this development stage, thus *models should be easy to turn into prototypes (R15)*.

Testers and Evaluators: models produced in the design phase are also used by testers and evaluators once the application has been implemented. In these phases, models should provide the ground for setting up the testing or evaluation plan. Testers and evaluators need *different concerns to be evaluated being easily identifiable (R16)* in the implemented application. Moreover, *models should look very close to the implemented application (R17)* so that testers and evaluators can easily match the running product to the originating models.

4. OUR MODELING APPROACH (TO SATISFY THE REQUIREMENTS)

In this section we briefly introduce the whole methodological framework to better contextualize the proposed conceptual modelling methods. In all the section, we specify precise references to the requirements discussed above as it becomes necessary.

In Figure 1 the composing phases are shown. A different modelling method is proposed for each of them. As well as other software development processes, we assume that these phases should be executed in an iterative and incremental way, therefore the picture only purpose is to express the phases order within the whole process. Considering the entire development process of a web application, we can say the framework covers both the *analysis* and *design activities* [[23]]. Moreover, adopting the Jackson terminology [[15]], we distinguish between the *problem* and the *solution domains*. These dimensions, the process and the domain, are used to organize the following discussion.

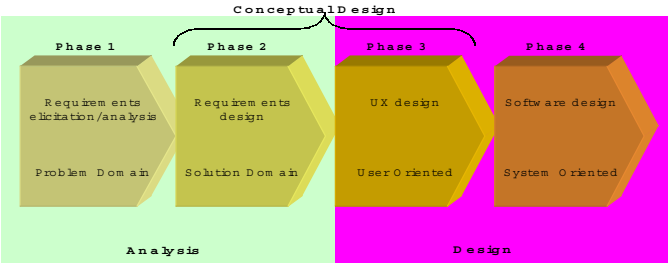

Figure 1: Phases in the development process of Web applications

The two left more phases are both achieved during the analysis activities. For supporting the **requirements elicitation and analysis** (phase 1) we propose AWARE [[1]], a goal-oriented method specially suited for web application requirements engineering. AWARE primitives include *goals* and *requirements* which definitively belong to the problem domain. However, in our experience, discussing with stakeholders (analysis) about needs and goals can be too abstract for a fruitful reasoning about relative importance of various goals and requirements and for eliciting new ones [[24]]. A first very high level solution, focusing on specific topics, can help validation and elicitation activities (e.g. interviews) enabling a more concrete discussion about the problem (*A2, A4*). We call this activity **Requirements Design** (phase 2) meaning that in this phase requirements take a more concrete form accomplishing a preliminary hop from the problem domain to the solution one. In this phase we use IDM [[22]]. Although in traditional SE approaches requirements are directly used for designing the software architecture (e.g. class diagrams, component diagrams, etc. using the UML terminology), in applications where the user interaction and the communication potential play crucial roles, the software design has to be postponed to the *user experience* design [[10]]. In this phase the application is designed as perceived by final users, neglecting how the software will be realized. Here, designers have to precisely define how users interact with the application to accomplish their tasks, taking care of the application usability and effectiveness with the respect of user requirements and quality expectations. In our framework, we achieve the concrete passage into the design phase by translating (*A2,A4,A5*) IDM models (phase 2) into E-WOOD ones (phase 3). IDM and E-WOOD, together, build up our approach to the conceptual design of C&II applications. Both methods take their foundations in W2000 [[19]], last heir of HDM [[4]] recognized as one of the first conceptual methods for web application design. As described in section 2, W2000, as well as other similar conceptual models, implements the *separation of concerns* principle by structuring the design in four dimensions. Both our methods keep this principle at the basis of their definitions but projecting the previous dimensions in a sole dimension for the sake of conciseness, for reducing the number of concepts to be learnt and references among diagrams (*A1,R8,R9*). The last step (phase 4) consists of a detailed design of the software that will be implemented to *realize* the desired user experience. This is generally called *logical design* of the system to-be. Passing from phase 3 to phase 4, a paradigm shift is achieved since, in phase 4, designers have to design the *system* that will *realize* the modelled user experiences. This passage is far to be straightforward and a number of trade-offs with the architectural constraints and various decisions have to be undertaken [[20]]. Models produced in this phase should specify a design easy to code. Here, we adopt the modelling method proposed by Conallen, namely WAE [[10]]. Our choice has been driven by two main reasons. First, it is already recognized in the industrial environment as the UML method for designing the software for web applications and a number of CASE tools already support its diagram drawing (e.g. Rational Rose, MS Visio). Second, as shown in paragraph 4.2, it is very easy and intuitive mapping WAE models upon E-WOOD as far as most of times, only one E-WOOD artefact is needed to define a set of related WAE artefacts (*R8,R9*).

Finally, the methodological framework also includes a number of guidelines on how to use every method within each phase and how to move forward and back between adjoining phases. Guidelines are informally described in terms of patterns [[25]] so providing an useful but flexible guidance (*A5,A4,R1,R2*). They also front specific design issues like the multi-user and multi-channel design. Lack of space prevents us to describe this aspect, but the complete set of guidelines can be found in [[20]].

4.1 IDM: supporting analysis with early solutions

IDM (Interactive Dialogue Model) [28] is a design model for interactive applications based on linguistic concepts of human dialogue. It bases on the interpretation of the interaction between the user and the application as a sort of dialogue (*A7*). It is simple to grasp, and effective in representing the most relevant features of the application in terms of content of the dialogue and dialogue moves (*A1,A9*). In fact, three simple design elements characterize IDM: “topic”, “relationship”, and “group of topic”. An interactive application may describe a “topic” (e.g. a “print”, or a “technique”); or it may allow the user to switch to a “related topic” (e.g. switching from a “print” to the “technique” used for it); or it may allow the user to start from a “group of topics” (e.g.

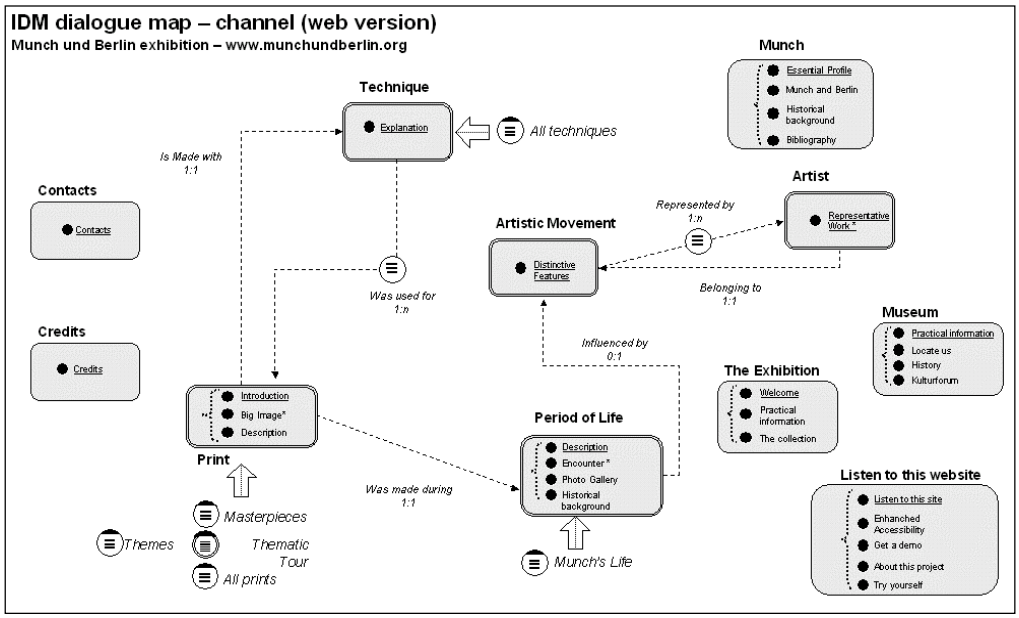

Figure 2: IDM dialogue map of www.munchundberlin.org.

"the masterpieces", or "the prints dealing with sickness") and then browse within the group. The above simple ideas have been translated into the following IDM design primitives (to understand the quoted example please refer to the schema in Figure 2 and to the website www.munchundberlin.org, design using IDM).

Topic: something that can be the subject of conversation between the user and the interactive application. "DRYPOINT" (a technique for prints), "THE SICK AN THE CHILD" (a print by Munch), "INTRODUCTION TO MUNCH" are example of topics, i.e. possible subjects of a dialogue between the user and the application.

Kind of Topic: the category of possible subjects of conversation. "Technique", "print" are kinds of topic. "DRYPOINT". is an example of "technique".

Change of Subject (or Relevant Relation): it determines how the dialogue can switch from a kind of topic to another one. "made with" is a possible change of subject relating any PRINT to one TECHNIQUE.

Group of Topics: it determines a specific group of topics, possible subject of conversation. MASTERPIECES is a specific group of PRINTS, while ALL_PRINTS is another, larger, group.

Multiple Group of Topic: it determines a family of group of topics. It could be nice, for example, to group the prints according to the themes, sources of inspiration for Munch. All the prints of the same theme are a group of topics; "prints by theme", overall, is a family of groups of topics (as many as there are themes). Each multiple group of topics has a corresponding "higher-level" group of topics (e.g. "all themes"), which allows to select the specific group of topics of interest (e.g. "prints about theme "sickness").

The above list of terms and concepts (including other advanced primitives which are not presented here for lack of space) has a number of advantages over most of the current design models and methods:

- The number of concepts is short, and therefore easy to teach (and to learn);
- Despite their limited number, the concepts are expressive enough for describing the concept of most C&II applications;
- Concepts (and terms) relate to the dialogue experience, rather than to informatics, therefore they can be more effectively conveyed to people without a computer science or engineering background (*A2,A7*);
- Concepts are of the proper "level" to allow an in depth comparison between requirements and design decisions (if
- requirements have been explicitly stated, of course) (*A3,A4,A5*).
- Design primitives represents clear concepts but none predefined structrure is imposed to enrich their description. Besides the graphical diagrams, they can be described at the needed level of detail adding information about, for example, editorial or architectural aspects (*A8,A11,A12*). The framework includes some guidelines about how to describe every concept type in order to improve the matching with the user experience and software design.

Figure 2 shows the visual representation of an IDM design schema (the *dialogue map*), namely the one used for the design of the website www.munchundberlin.org, dedicated to the the temporary exhibition of Edvard Munch's printed hosted at the National State Museum in Berlin during summer 2003.

A number of further considerations can be derived from this diagram.

- The schema is quite simple and it does not take too much time to write it down (any common editor tool or even paper and pencil may fit) (*A1*).
- The schema can be used to brainstorm, debate alternatives, and discuss preliminary decisions (*A8*).
- Due to its conciseness and intuitiveness, it is easy to make suggestions about alternative solutions (*A3,A9,A10*).
- The schema conveys the basic interaction ideas, without commitment to a specific software system (it could be a web application, a vocal interaction system, a mobile application and so on).

4.2 E-WOOD: modeling the user experience

Our proposal for designing the user experience, called E-WOOD, has been defined as a UML extension. UML has been chosen as modelling language to meet *R11*, while the extension mechanism has been preferred to defining a metamodel in order to exploit easily existing commercial tools (*R4*). Our model extends an existing proposal for designing the user experience, that is, the UX [[10]] since, as shown in the Conallen's book, mapping WAE models upon UX ones is easy and intuitive (*R8,R10,R17*). UX's high level primitives are *screen* and *links*, and an application is merely considered as made up of a number of screens connected by links. Typically, a set of WAE artefacts are mapped upon a screen by means of *realization* associations (stereotyped as <<build>>), specifying which logical elements (WAE models) build the various parts of the screen (contents and links). Our main goal in extending the UX has been to add the needed semantics (extracted by the W2000 primitives) to enable the separation of concerns impacting the application usability, its functionalities and the whole quality (*R5,R13,R16*). In E-WOOD different concerns are specified in different views and by introducing specific design concepts. These concepts have been defined extending standard *class* and *association* elements in terms of *stereotype, semantic description, constraints, tags* properties. An additional property (*mapping constraints*) has been also introduced to specify mapping constraints between IDM and E-WOOD models (*R1,R2*). As well as in UX, E-WOOD high level primitives are *screens* and *links*. Screens can aggregate both content and input forms; links can be used to perform a simple navigation among pages or to provide inputs to operations and processes. E-WOOD models are thus very close to the application to-be (*R6,R7,R17*) and easy to turn into prototypes or mock-ups (*R15*). Keeping these basic primitives we have also preserved the proven mapping capabilities towards the WAE (*R10,R8*).

The introduced semantics is also used to define a framing strategy (*R3*) which helps designers organize the overall design activities, fosters reuse and make design documentation more readable (*R9*). The framing strategy mostly reflects the W2000's design dimensions. E-WOOD proposes to organize the design of the overall application in five views. Each view includes several diagrams and makes use of specific stereotyped classes. Due to the lack of space, in the following we only describe three out of five views to show the philosophy behind our method and how we have tried to accomplish the above stated requirements. The complete specification can be found in [[20]].

The *Template View* is used to define common contents and links of page sets. Examples of common contents could be the copyright information, the company logo and so on, whilst examples of common links could be those connecting to the home page or to the various site's sections (like those on the bottom of many web sites). Typically the template design involves the graphical designers who are in charge of the application look-and-fell (*R5*). The basic primitive used in these diagrams is the <<Screen Template>>, an abstract class used as place-holder for content and links belonging to a set of screens. Layout contents (both information and graphical elements) and common links are modelled respectively by means of <<Layout Content>> and <<Landmark link>> primitives. The *Structural View* is used to define pages enabling users explore information concerning the domain entities or IDM' topics. <<Content>> classes are aggregated to screen classes and models portions of the whole topic information. <<Structural link>>s are used to model the navigation achieved across pages belonging to the same topic. For example, as depicted in Figure 3 (a), the overall information concerning the "Print" entity are organized in three pages (Introduction, Big Image and Description) which are connected by means of bi-directional links originating from the "Introduction" page. Each IDM topic is mapped on a number of content classes (and relative pages) equivalent to the number of its dialog acts. Content classes are then enriched by a fine-grain definition of data slots which can be used as input for setting up the editorial chain (*R13*). Content classes contain a Boolean tagged value called *entry point* whose purpose is to specify whether that portion of the content can be used as starting point for exploring the entity information. Following our framework guidelines, such pages should include, at least, a minimal set of entity attributes that can be used by the user to understand what the entity instance talks about. Information organization, kind of navigation and entry points are concerns usually discussed with communication and usability experts (*R5,R16*) taking in mind that when users navigate these pages are clearly interested in improving their knowledge about the entity.

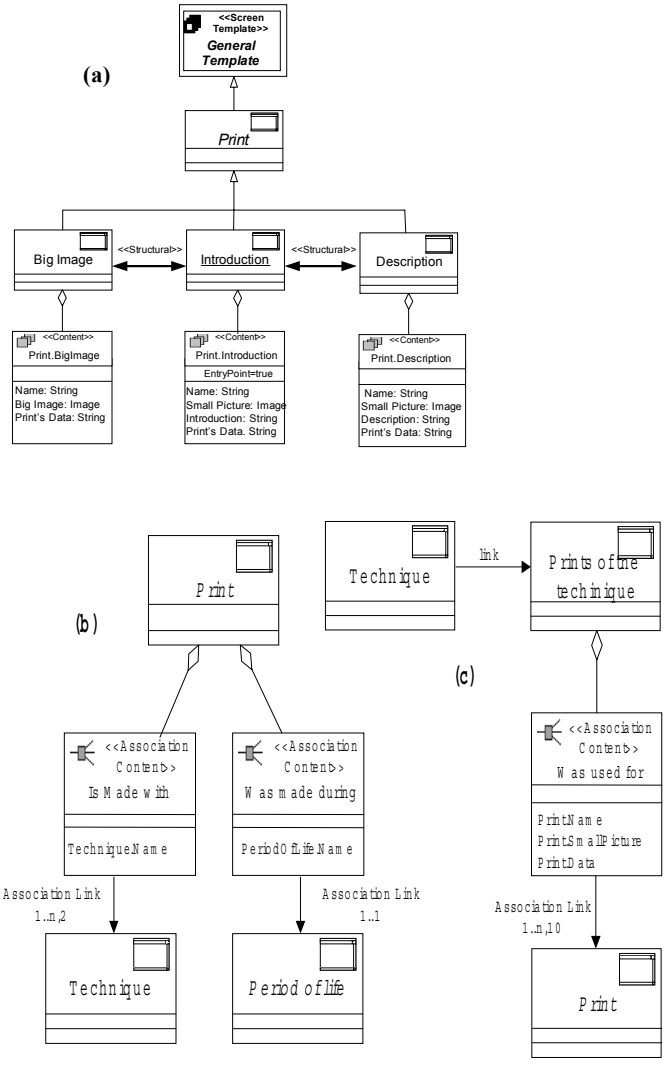

Figure 3: a) Structural view for the "Print" topic; b) and c) Association views

In the *Association View* designers specify how to pass from a discovered interesting topic to a related one (*relevant relation* in

IDM). For this part of the user experience design, it is very important to carefully decide how to allow users understand which of the possible target topic instances they are actually interested in. This aspect is called, in the HCI community, *information scent* and is one of the factors strongly impacting the application usability. In E-WOOD we use to this purpose the <<Association Content>> (Figure 3 (b) and (c)). In (b) these information are integrated in all the "Print"'s pages (it is aggregated to the abstract page representing the common features of all the structural pages) and <<Association link>>s connect these pages to the target one. In (c), the "Technique" page includes a <<Link>> association which brings to new page "Prints of the Technique" whose only purpose is to list the possible target "Prints" which have been produced using the source "Technique". From this page a <<Association link>> point to the destination pages. The <<Association link>> primitive includes a tagged value that specifies the association *multiplicity* in terms of *min, max* and *expected* values. In particular, the expected multiplicity provides a useful indication about how many instances of the target entity are in general addressed by the association. This information can be used for taking some design choices like attaching the <<Association content>> to the source page or defining a new ad-hoc page (the two possible solutions shown above). Having *max* or *expected* cardinality very small, our guidelines suggest aggregating the <<Association Content>> to the source pages, while in case the expected number grows up, we suggest the other solution.

Similarly to the structural and association views, the *Access View* and *Operation/Business Process View* are used to design respectively pages supporting access structures (e.g. book categories in an e-commerce web site) and operations and business processes (e.g. the "add to shopping cart" operation or the "check out" process). Besides these main views, we also propose a *Navigational Map View* that summarizes the main navigational features of the entire application. Our guidelines suggest how to choice candidate pages, among the overall defined in other views, to be included in the navigational map. Switching from the navigational map to the detailed design of contained screens it allows *R12* being accomplished. Finally, following our guideline, the navigational map looks very similar to the IDM dialog map making easier referring back to analysis artefacts (*A5,A12,R1*).

5. CONCLUSIONS AND FUTURE WORKS

We all know that the existing literature about conceptual methods addressing the design of Web applications is (over)abundant. On the other hand, we also know that a remarkable gap between theory and practice still exists [[1]],[[3]]. *Which are the reasons behind the poor acceptance by the practitioners*? Starting from these considerations, in this paper we have claimed that a possible reason could be that existing proposals have failed short in neglecting stakeholders' goals and expectations. In this light and focusing on the development of C&II applications, we have carefully analyzed the environment where a design method should operate identifying which are the potential stakeholder types and their goals and requirements. On the basis of this analysis, instead of inventing new design methods, we have reused or extended the best, in our view, of current approaches both in the academic and industrial communities. The approach covers both analysis and design activities and consists of four phases, executed in an iterative and incremental way. Defining it, we put in practice most of our experience achieved working on the field with conceptual design methods for Web applications [[3]], [[24]]. By this experience we realized that stakeholders' needs and expectations in a conceptual model vary when moving from the analysis towards the implementation in the development lifecycle.
Within this framework, our design methods IDM and E-WOOD have been defined taking explicitly into account their users and their audiences in order to improve their effectiveness on the field. IDM is a lightweight method that should be used to sketch the main aspects of very early stage solutions for supporting the problem understanding in the analysis phase, while E-WOOD should be used to design a complete solution, from the user point of view (user experience), that is, to specify what the implementation should realize.

IDM is a good candidate to meet the requirements of the stakeholders in the analysis phase, since it offers easy-to-learn tools to master the delicate process of passing from requirements to design and enables to capture the complexity of the user-application dialogue with a limited number of concepts, which are familiar also to people without a technical background.

E-WOOD is a UML profile that enables to specify the user experience in terms of pages and links but that embodies semantics enabling different stakeholders reason about crucial concerns heavily impacting the application usability and effectiveness, that is, its perceived quality. Moreover, due to its definition, E-WOOD can exploit existing commercial tools for supporting the model drawing and perfectly match an existing and already affirmed, among practitioners, method for designing the software modules of a C&II application.

The approach has been applied in several design and reverse design case studies and industrial projects. Its transferability in industrial environments has been also experimented in two projects in cooperation with two Italian software companies (in the context of the GENESIS-D projects [[20]]). From these first experiences a number of considerations can be drawn out. Compared to W2000, we have noticed a significant decrease of the required learning time. Practitioners were able to use both methods after a short but intensive course (2-3 days). They drew IDM models using paper and pencil, while used VISIO™ stencils for designing E-WOOD and WAE models. In all the achieved experiences, we spent, with E-WOOD, on the average one third of the time required by W2000 to produce the same level of detail in the specification of several application designs. This has to be summed to the time required for manually drawing IDM models which is, however, very contained. Compared to UX, we obtained several advantages mostly due to the introduced semantics. Models are more expressive and easy to be revisited; the framing strategy enables a suitable organization of the overall design activities; a number of well know design patterns, developed in the web engineering community, can be exploited to produce quality applications.

Finally, concerning future works, we are working in two main directions: (i) enriching the framework with guidelines and patterns for fronting specific aspects like the multi-channel design and the mapping of E-WOOD models upon the most known software architectures (JAVA and MS.NET); (ii) defining and implementing a complete set of supporting tools. Concerning the second point, we have already realized an ECLIPSE [[25]] add-in for supporting the model drawing of each method belonging to the framework. In the next months we aim at defining further add-in modules that should support the semi-automatic translation of models belonging to contiguous phases so as to enable

requirements and design tracking and the automatic generation of mock-ups or throw-away navigational prototypes.

REFERENCES

[1] Barry, C.; Lang, M. A Survey of Multimedia and Web Development Techniques and Methodology Usage. IEEE Multimedia. 8(3) - 2001, pag. 52-60

[2] Bolchini, D., Paolini, P. Goal-Driven Requirements Analysis for Hypermedia-intensive Web Applications. Requirements Engineering Journal, Special Issue RE03, Springer 2003.

[3] Garzotto, F., Perrone, V. On the Acceptability of Conceptual Design Models for Web Applications. In Pro. of ER'03 Workshops, (IWCMQ'03), October 2003 , Chicago, USA.

[4] Garzotto F., Paolini P. HDM- A Model-Based Approach to Hypertext Application Design. In ACM Transactions on Information Systems, Vol. 11, No1 January 1993, p1-26.

[5] Isakowitz T, Stohr EA., Balasubramanian P. (1995) RMM: A design Methodology for Structured Hypermedia Design. In Communications of the ACM Vol.38 No8 August pp 34-44.

[6] Schwabe, D., Rossi, G. An Object Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems, 4 (4), J. Wiley, 1998

[7] De Troyer, O., Leune, C., WSDM: a User-Centered Design Method for Web Sites, in Proceedings 7th International Wolrd Wide Web Conference, Brisbane, 1997.

[8] Ceri, S., et al., Designing Data-intensive Web Applications, Morgan Kaufmann, 2002.

[9] Hennicker, R., Koch, N. A UML-based Methodology for Hypermedia Design. In volume 1939 of LN in Computer Science, York, England, October 2000. Springer Verlag.

[10] Conallen, J., Building Web Applications with UML (second edition), Addison-Wesley, 2003.

[11] Gómez, J., Cachero, C., Pastor, O., Conceptual Modeling of Device-Independent Web Applications, IEEE Multimedia, April-June 2001 (Vol. 8, No. 2).

[12] Engels, G., Groenewegen, L. Object-oriented modeling: a roadmap. In A. Finkelstein, editor, "The Future of Software Engineering", Special Volume published in conjunction with ICSE 2000, 2000.

[13] OMG, Object Management Group: Unified Modeling Language (UML), version 1.5

[14] Kaindl, H., et al. Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda. Requirement Engineering journal 7(3): 113-123 (2002)

[15] Jackson, M. The World and the Machine. In Proceedings of ICSE-17; ACM Press, 1995.

[16] Mylopoulos, J., Information Modeling in the Time of the Revolution. Information Systems, Vol. 23, 1998

[17] Bolchini, D., Paolini, P., Dialogue-based Design for Multichannel Interactions. In Proc. of IWWOST04 workshop held in conjunction with ICWE'04, München, Germany.

[18] Baresi, L., Garzotto, F., Paolini, P. From Web Sites to Web Applications: New Issues for Conceptual Modelling. In Proc. WWW Conceptual Modeling Conf, October, 2000.

[19] Baresi L., Garzotto, F., Paolini, P., Perrone, V. Hypermedia and Operation Design. Deliverable D7, European IST project UWA (Ubiquitous Web Applications), www.uwa-project.org

[20] Balconi, A., Mainetti, L., Paolini, P. Perrone, V.: GENESIS-D: Formal specification of the conceptual and logical models. Politecnico of Milan, deliverable D2.2, project Genesis-D (October 2004).

[21] Rogers, E., Diffusion of Innovation, MT Press, 1995.

[22] Bolchini, D., et al. IDM – A User-Centred Model Shaping User Interaction as a Dialogue. In proc. HCII 2005 International Conference on Human-Computer Interaction, Las Vegas, USA, 2005.

[23] Ghezzi, C., Jazayeri, M., Mandrioli, D.: Fundamentals of Software Engineering. Prentice-Hall, 1991.

[24] Perrone, V., Bolchini, D., Designing Communication Intensive Web Applications: Experience and Lessons from a Real Case. In proc. of WER 2004, 9-10 Dec. 2004 - Tandil, Argentina. To appear in a special issue on Req. Engineering of the Journal of Computer Science & Technology, autumn 2005.

[25] Gamma, E, Helm, R., Johnson, R. and Vlissides, J. Design Patterns: Elements of Reusable Software Architecture. Addison-Wesley, 1995

[26] Eclipse consortium. Eclipse – Home page. www.eclipse.org/.