Building Hypermedia Artifacts by the systematic use of the Flexible Process Model

Luis Antonio **OLSINA**
Computer Department, Engineering School, UNLPam; also Ciencias Exactas School, UNLP- Argentina
E-mail olsinal@unlpin.edu.ar - **TelFax** (+54) 302 24711

Abstract

Most of the current hypermedia model life cycles focus in analysis and design issues, ignoring crucial tasks and activities of hypermedia projects. Others do not take care of basic Software Engineering concepts such as planning, physical and logical modeling, validation and quality assurance, among other issues. In this paper we propose an integrated software process model, called Flexible Process Model, useful in building hypermedia artifacts. This strategy, when instanciated in a specific project, implies a systematic use of model-based constructors, both logical and physical models.
The main benefits of this process model are: a) it covers all the principal phases and tasks of a hypermedia project; b) this clear break down can contribute fairly to project planning and can help to establish milestones and metrics; c) it fosters a positive balance by a systematic use of logical and physical modeling; d) it facilitates human communication; e) it promote process improvement and standardization.
Therefore, we will discuss and represent, in a medium level of granularity, the phases, tasks and activities, mainly in the dynamic modeling phase. Also we will present some perspectives, stressing the functional, methodological and behavioral perspectives of the three-phased Flexible Process Model. Finally, we will discuss related works and concluding remarks.

Keywords: Flexible Process Model, Hypermedia Project, Physical and Logical Modeling, Object-Oriented Technology.

1. Introduction

The development of software artifacts is a process of model building and this is reached by a systematic use of strategies. Indeed, the discipline of Software Engineering faces new challenges with the development of hypermedia artifacts. We have thought that the study of new development processes to build hypermedia applications is an indispensable research activity for the nowadays-technological challenges.

We define a software process model as an appropriate strategy to abstract, organize, execute and control (by means of heuristics, methods, techniques and tools), the different phases, tasks, activities, artifacts and resources of the hypermedia project to reach the desired goals. A software process model must answer issues such as, what to do, how to do it, when and where will it be done, who will do it and what dependencies there will exist.

We can benefit greatly from established logical models and researches that put emphasis in objects models [Booch 96, Goldberg et al 95, Jacobson 94, Rumbaugh et al 91], as from models and principles of hypermedia design discussed, for instance in [Isakowitz et al 95, Nanard et al 95, Schwabe et al 96, Thüring et al 95]. Likewise, contributions in the world of physical modeling have come from [Boehm 88, Connell et al 95, Davis 92, Nanard et al 95], who in some cases apply cycles with prototyping and in few cases use O-O methods and techniques [Connell et al 95, Nanard et al 95].

It is worth stressing that most of the existing hypermedia development processes focus chiefly in analysis and design tasks leaving aside the definition of activities that are fundamental in an integrated process model that considers Software Engineering principles such as project planning, quality assurance strategy and logical and physical modeling. Generally no one treats the concept of physical modeling as a fundamental strategy to guide the development of hypermedia project that feeds different tasks in the way that we propose it. We apply a set of strategies to create and evolve at the same time both logical and physical models. Among the first we can develop plan model as well as requirement, conceptual, navigational, abstract interfaces and validation models. Among the second, we can build sketches and software prototypes as physical models.

So that our integrated approach consists of a Flexible Process Model (FPM) [Olsina 96, 97a, 97c] to support the whole development cycle of hypermedia artifacts in which the physical models are built with O-O flexible prototyping strategy [Olsina 97b] and the used logical models to develop conceptual, navigational and abstract interfaces model respond to O-O Hypermedia Design Method (OOHDM) [Rossi 96, Schwabe et al 96]. Also we apply a requirement model similar to Jacobson's model [Jacobson et al 92] and other modeling constructors.

We consider that the logical models improve the consistency of the specifications and provide well-defined guides throughout the development cycle but the use of appropriate prototyping strategy and experimental feedback loop are also of vital importance for the hypermedia process.

We will develop this work as follow: first, we will describe the essential entities of the hypermedia project in the context of a flexible process model. Next, we will concentrate in the main phases, tasks, artifacts produced and process constructors utilized, mainly in the development phase. Next, in section four, we will present the behavioral view of the FPM, to finally discuss related works and some concluding remarks.

2. Main entities of a Hypermedia Project

First, we will present the main components of the hypermedia project in the context of the flexible process. Figure 1 depicts a diagram of classes, relationships and subsystems that abstracts a static view of the fundamental responsibilities and collaborations of the hypermedia project (here we use a similar notation that specified in [Rumbaugh et al 91]). It must take into account that classes and subsystem represents a general set of responsibilities and are not intended to be instanciated as in O-O program. We consider the following components:

- the *process model* component that abstracts and represents project entities like task, resource, role, agent, process constructor, artifact among others.
- the *task* entity that represents the basic management unit of any software project. A task can be planned, scheduled, enacted and monitored.
- *the goal and objectives* that represent the statements of all the outcomes that is desirable to be reached.
- *the process constructors* component such as strategies, methods, models and development criteria. In turn, this class inherits six other classes denominated: Plan Model, Requirement Method, Design Method, Aesthetic and Cognitive Criteria, Prototyping and Integration Strategy and Quality Assurance Strategy.
- *the resource* component, i.e. human, technological, etc. The project coordinator,

the architect, designers and users who are involved in the process, integrates the first; the second are to match software and hardware components, which automate tasks and activities. On the other hand, the project is subject to monetary, material, human and time restrictions.

- the *software artifact repository* component that acts as input and output in the three phases of the FPM. These are the exploration phase, the development phase and the operational phase shown in the next section.

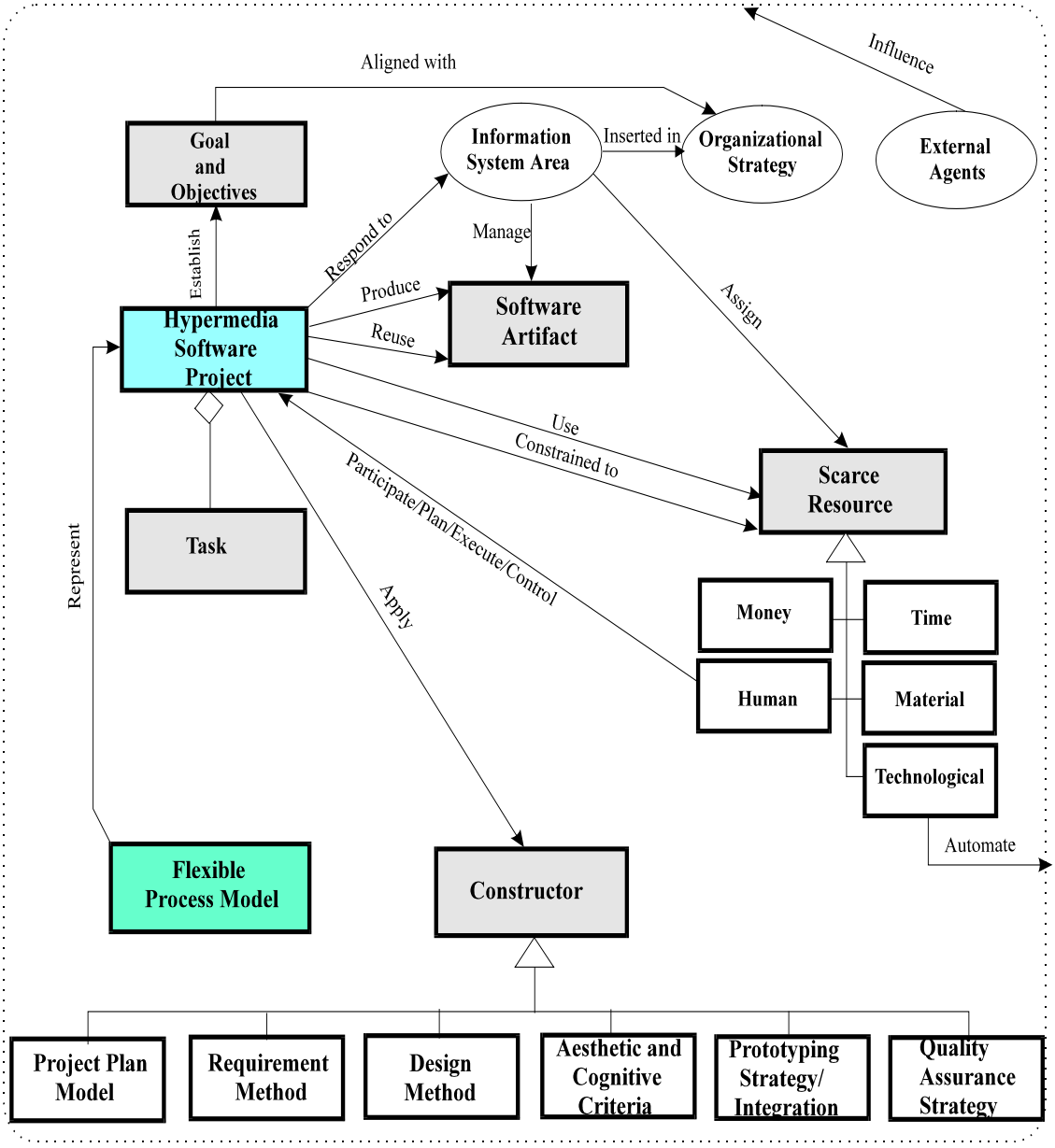

Fig. 1. Diagram of the Hypermedia Project as part of the Flexible Process Model.

We might distinguish a software process model from other types of process modeling because many of the issues represented are carried out by humans rather than by automated devices. Therefore, a software process model deals with those software projects phenomena that occur during creation, evolution, control and maintenance of software artifacts.

3. General view of Phases and Tasks of the Flexible Process Model.

Although we can abstract different perspectives in process modeling [Curtis et al 92], next, we will present a general view, i.e., rather a medium level of granularity of our proposed FPM, remarking functional, methodological and behavioral perspectives.

In figure 2 an ellipse represents a task or process; a process can abstract a set of sub-processes; a sub-process may be composed of one or more cohesive activities. An activity is associated to atomic actions. A role has assigned coherent activities and a role is "instanciated" with resources. The arrows represent input/output dependencies (for instance, a process produce an artifact that serves as input to one or more processes or a process can receives one or more messages).

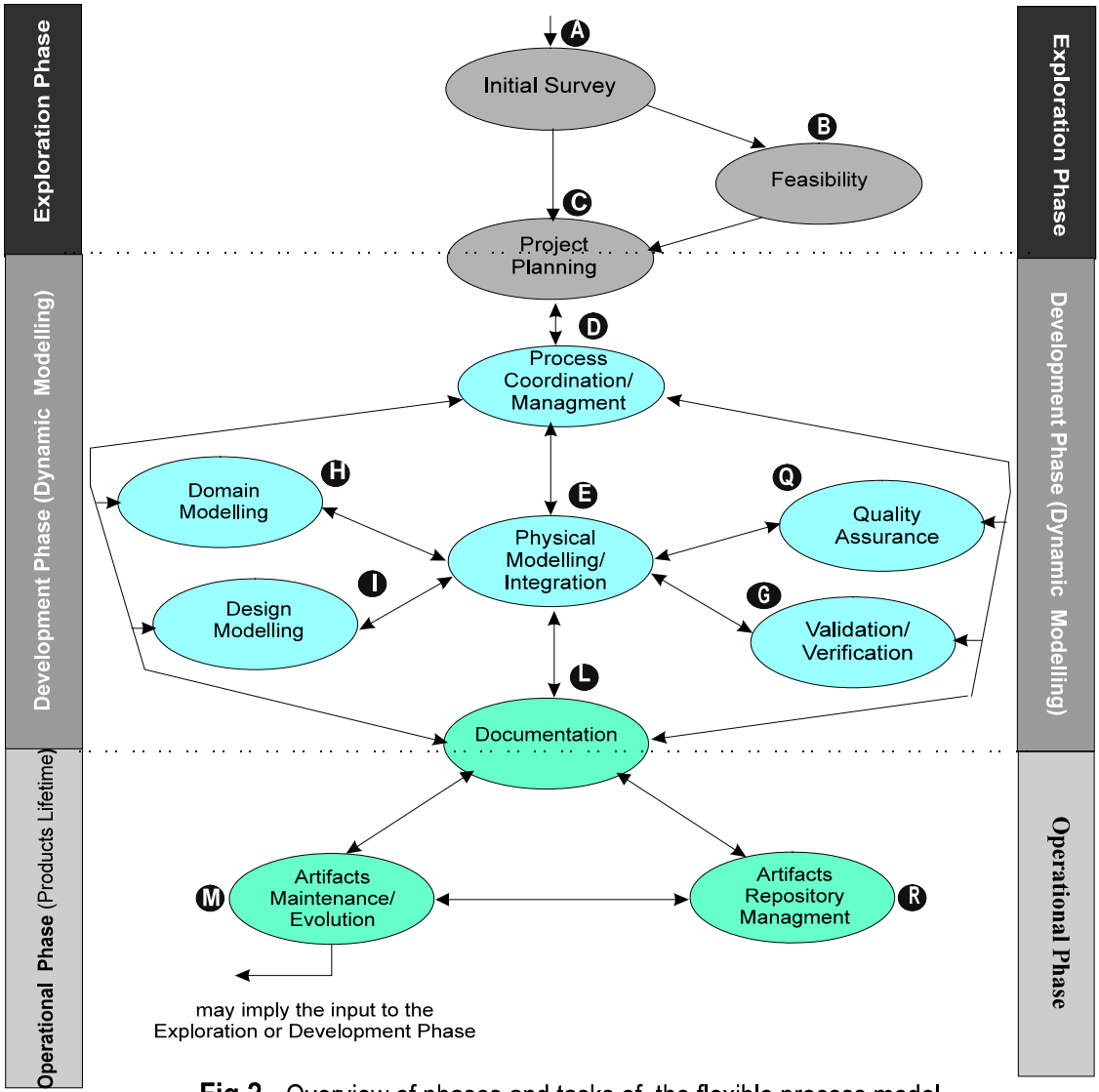

Fig.2 Overview of phases and tasks of the flexible process model

There are three general phases in the FPM. A first phase is called *exploration phase* wherein initial concepts and users' needs are elicited; next, if necessary, it is possible to carry out a feasibility study and, then, we can build a preliminary plan model. The second phase, *the development phase*, is the core of the dynamic modeling (we will concentrate on this phase). The third phase, called *operational phase*, essentially consists of products' configuration, maintenance and evolution. The latter could imply

the input to the exploration or development phase.

Prior to beginning the development process, initial insight about the problem and its alternative solutions must be highlighted. The **A**-task input may be at best the extension of an operative system or component with documented requirements or perhaps the construction of a new system or component (given the novel character that has the hypermedia discipline it is difficult to find documented applications). We use here some established techniques to capture the users' needs. For instance, it is possible to determine an initial set of action sequences that define the way in which distinct actors interact with the system and establish initial use cases. The output of this task is an initial requirement specification in natural language. From these descriptions we can later feed logical models and physical models so that we can extract information of how the users perform their tasks and we can recognize classes, attributes, transformations, navigational contexts or other building primitives.

It is necessary to make an initial model plan, by means of the **C** task. It will basically contain, from the preliminary requirement specification and, if necessary, from a feasibility study: a description of the goals and objectives of the hypermedia system to be built or extended, the foreseeable scope of the final product and deliverables, the developers and users involved and their respective responsibilities, the process model strategy to be applied, the selection of the working environment and tools and, also, the likelihood to establish metrics. As we can see, these activities embrace both phases, therefore the plan document (the output) should also include the date and revision number that will be useful for the project coordinator in subsequent iterations.

As we can see, there are mainly five components mutually necessary in the development phase, namely: the **D** task that is the central process which coordinates and controls other processes and activities, the **E-G** component, that corresponds to the construction and validation tasks centered in prototyping strategy, the component made up by the **H-I** processes that accomplish logical modeling and implementation-independent specification activities, the **L** task which comprises documentation and fosters reuse, and the **Q** task of quality assurance, which promotes ultimately the users' satisfaction.

3.1. The Development Phase

In this part we will focus in the development phase, its subtasks and produced models and we do not deal with roles and assigned resources -the organizational perspective of the FPM. This phase is the core of the dynamic modeling process yielding both logical and physical models. In figure 3 we have made a process break down for development phase. The primary operation center of this phase is the **D** process that works as a pivot and could be functionally summarized in this few words: plan-do-check-act. The chief role assigned to this task is that of the project manager. It essentially coordinates, i.e. sends and receives messages to processes such as planning, prototyping, logical modeling, documenting, testing and quality assurance among others. Next, we will show for the **H**, **I**, **J** and **K** processes, their own activities and their produced logical models and artifacts that feed a documentation task.

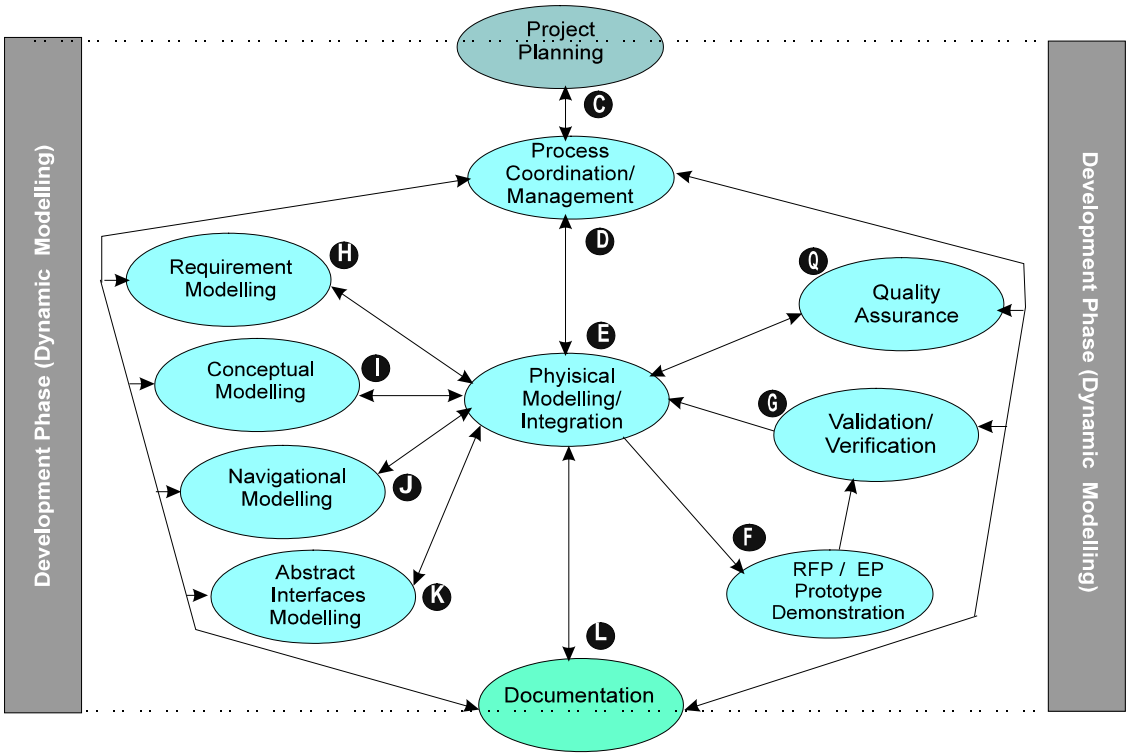

Fig.3 The main tasks of the Development Phase

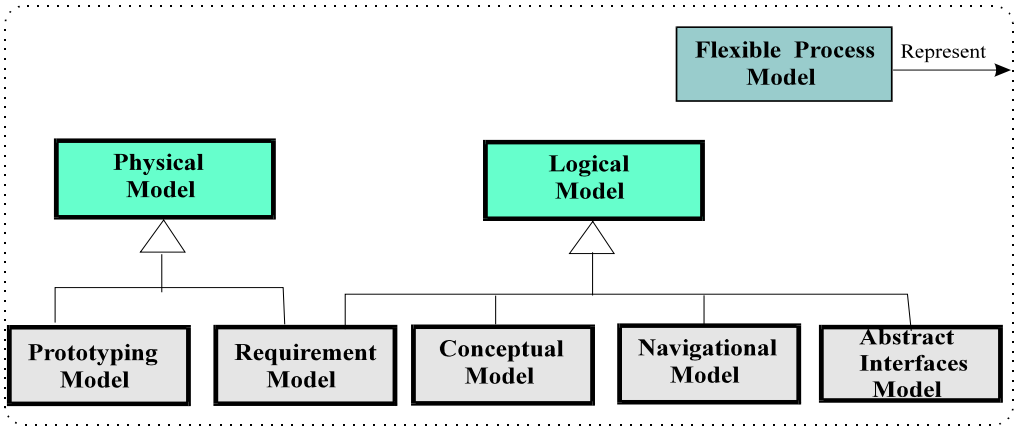

Fig. 4. Some logical and physical models yielded by FPM

Figure 4 shows a diagram of a logical models that are created and evolved by the FPM in the development phase, namely: the requirement model as well as conceptual, navigational and abstract interfaces models (we will not discuss some other models to support testing and quality assurance). It also shows physical models.

As previously explained the exploration phase produces a preliminary requirement specification that serves as input to the requirement modeling (**H**-task). From these natural language descriptions we can refine them and feed the use-case model as well as the interface and glossary models (fig. 5) that will be the output for the **H** process. We will briefly describe the requirement model because we use some adaptation to hypermedia applications of Jacobson's well-known requirement model [Jacobson 92]. We share the author's point of view that among the first software models to be created is the requirement model since it describes the system, the environment and their relationships, reflecting an external view of the system.

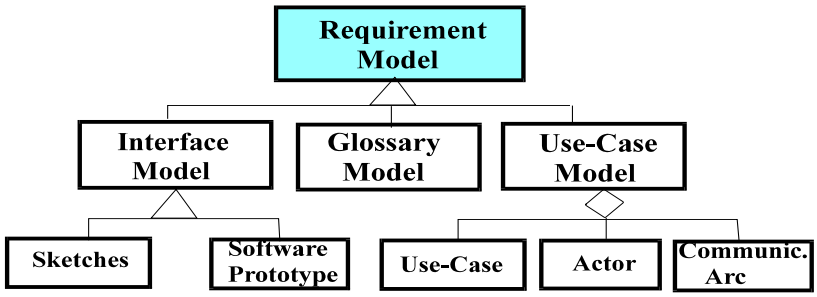

Fig. 5. The main components of the Requirement Model

A *use-case model* [Jacobson 94] is a graph with two kinds of nodes: use-case nodes and actor nodes. Besides, it has communication arcs. A communication arc links use-case and actor nodes so that an actor node is linked to at least one use-case node and the latter is linked to at least one actor node. This model treats each node as a class to be instanciated. An actor instance can create use-case instances. A communication arc between an actor node and an use-case node implies that it is able to send stimuli between instances of actor class and instances of use-case class.

The difference between use-cases of the requirement model and classes of the conceptual model (that will be the **I**-task's output) is that, classes communicate between each other inside the system, while use-cases are not able to communicate with other use-cases inside the same system for simplicity reasons. Finally, an actor is an entity that is modeled outside of the system, i.e., in the context, whereas a use-case is modeled inside the intended system. When an actor uses the system then it executes a use-case, i.e., a sequence of one or more atomic actions. The use-case collection represents the whole system functionality.

The *glossary model* is a subset of a domain object model (from Jacobson). Nevertheless, it is powerful enough to capture essential problem domain keywords. These keywords and their related concepts are specified in a classified list, written in a natural language, that later feed the conceptual modeling (**I**-task) or the navigational modeling (**J**-task). It is an important means to communicate the key concepts in early stages between practitioners.

The *interface model* uses physical models such as paper sketches and software prototypes. (Despite the fact that we are discussing logical model, and an abstract interface model is a logic one, here we place physical model for the sake of clarity. However, prototypes and prototyping will be widely discussed in this section). These working models help to elicit functional and aesthetic requirements involving the users in the earlier activities. In hypermedia application building the right interfaces and the coherent navigational units are of crucial importance.

As was previously said, our hypermedia development process use OOHDMs' conceptual, navigational and abstract interfaces models because its independent-implementation models, are useful for specification and documentation activities and favor the reuse strategy.

During the *conceptual modeling* process (**I**-task) the developers use primitives such as classes, attributes (multiple-typed attributes), relationships and sub-systems (see fig. 6). The main objective of this task is to analyze and to specify the semantics of the problem domain. Software Engineering's well-known mechanisms such as generalization/specialization, aggregation and classification are applied. An important

technique that help documenting and maintenance tasks, is the specification cards in which they provide, in a structured textual style, details of the salient parts of the sub-systems, classes, objects and relationships allowing, on the other hand, to introduce information for traceability issues.

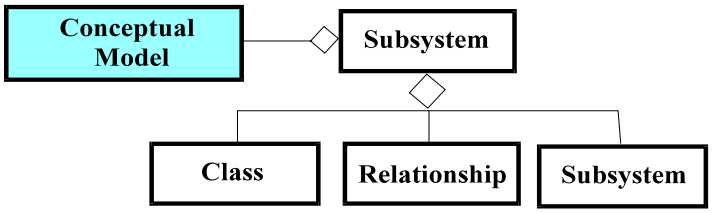

Fig. 6. Aggregate diagram of the Conceptual Model

During the *navigational modeling* process (**J**-task), the architects and developers work in the follow primitives: nodes, links, navigational classes and navigational contexts (see fig. 7). Navigational transformation models specify the dynamic. Nodes can be atomic or composed. Links attributes and behavior, their cardinality and their source and target navigational objects can be specified. We can also define access structures and guided tours that are useful for an intended user to find and to navigate information spaces. Nodes, links and anchors are navigational classes and we can build their related specification cards taking into account traceability issues (backward and forward).

The navigational context is a way to organize the navigation. It is a design primitive (or a design pattern) that is composed of nodes, links and other navigational contexts (maybe nested). This constructor allows to represent a coherent unit of related concepts and to establish appropriate semantic relationships fostering user orientation [Schwabe et al 96, Thüring et al 95]. Both navigational classes and navigational contexts abstract the static structure of a hypermedia application. To specify the dynamics, OOHDM offers a statechart-based constructor that allows building up Transformation Diagrams.

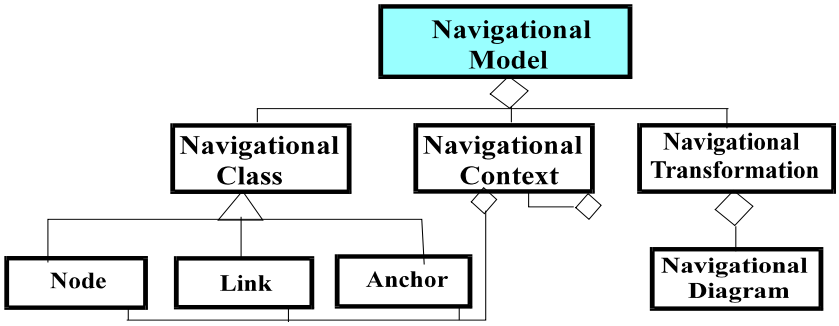

Fig. 7. Main components of the Navigational Model

In the *abstract interface modeling* process (**K**-task in fig. 3), we take into account what events will intervene in the action language and what interfaces objects the user will perceive, what transformations will be carried out and how the interfaces objects will be synchronized. In Rossi et al 95, we can look at the models developed to represent the static and dynamic behavior. The products of the K-task output are ADV (Abstract Data View) to represent the perceivable objects and the configuration diagrams that model the static relationships between ADV, external events initiated by user and the interface objects that cause navigation. To show the dynamics, we can specify, for

each ADV, its correspondent ADV-chart (fig. 8).

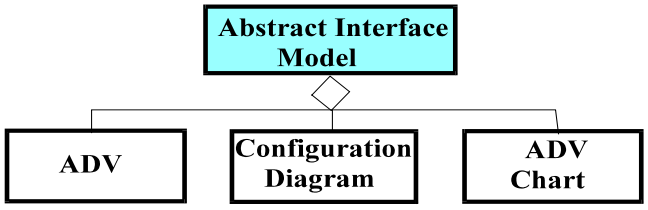

Fig. 8. Aggregate diagram of the Abstract Interface Model

Our *physical modeling* process (**E**-task) is achieved by a systematic use of prototyping that we consider as a software development strategy. This strategy essentially promotes the learning, construction, demonstration and validation cycle (**E**-**F**-**G** tasks) based on the experimental feedback loop between developers and users. It helps to discover and specify functional and non-functional requirements [IEEE 93], to experiment analysis and design alternatives and to make an evolutionary base of the future system grow. The prototyping strategy benefits the apprenticeship process by minimizing the subsystems building risks. It is characterized by a high degree of iterations and parallelisms with other activities, by a user's high level participation, and by a great use of working models and advanced production tools. The **E-F-G** process outputs are paper sketches and mainly software prototypes.

By software prototype we mean a physical and dynamic model built on computers that is a partial implementation of the entire system or from components of it, and it is used to learn, discover, evaluate and evolve functional and non-functional requirements.

- Our proposal hierarchically classifies the strategy in three concrete classes: rapid-functional prototyping (RFP), evolutionary prototyping (EP), and O-O flexible prototyping (OOFP) that inherit the behavior from both RFP and EP. In figure 9 we see the main responsibilities modules for the strategy.

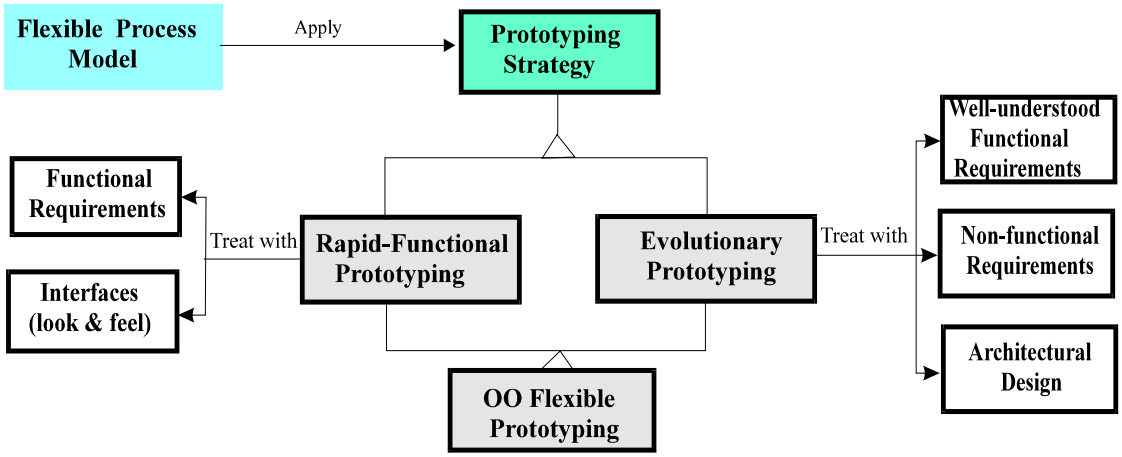

Fig. 9. Diagram of the main responsibilities of the O-O flexible prototyping strategy.

Some attributes and objectives of the rapid-functional prototype and the strategy are, namely:

- it is built as quickly as possible, sacrifying the completeness though not the correctness and the aesthetics of the interfaces.
- it must be planned if the prototype will be discarded or will be able to reuse it and

extend it.

- it permits to evaluate functional requirements understood by the developer but that need to be experimentally validated with the user; it also allows to dynamically capture poorly understood requirements.
- it allows to discover aesthetic and interactive aspects of interfaces and navigational objects.

it permits to yields inputs to specifications of the requirement model as well as to the conceptual, navigational and abstract interfaces models (behavior and attributes, controls, navigational contexts and transformations).

Some attributes and objectives of the evolutionary prototype and the strategy are:

- the completeness, correctness and quality attributes of the developed prototype are very important. It is slower than the RFP although planned to evolve toward the final component or subsystem version.
- the input to the E process may be the evolution of components prototyped and validated by means of a RFP or from components yielded in the operational phase.
- it allows to build upon solid bases the understood critical requirements, and to discover architectural aspects of design and to verify non-functional requirements such as performance, security, etc.
- it permits to yields inputs to specifications of the conceptual, navigational, abstract interfaces and verification models.

O-O flexible prototyping inherits the previously stated features and it is flexible in two ways. First, because depending on the problem to attack, it already has either the RFP strategy or the EP strategy, or both at once. At best it applies EP to build and to evolve the better understood and consensuated requirements, while it applies an RF strategy for the poorly understood features or for those that require user validation. So the OOFP offers a software base that is the foundation upon which the system evolves, using the working development environment of the final versions as much as possible. Secondly, the strategy is flexible by the potential balance that takes place between the prototyping heuristics and tools and the logical models for specification and design based mainly on mechanisms that support the object-oriented principles and techniques.

Finally, we can observe input-output dependencies between logical and physical models with some detail in fig. 10, which intends to complement the processes and dependencies that were shown in fig. 3 and the products yielded as previously discussed.

We see how flexible prototyping implements a requirement, navigational and abstract interface model helping in specification and documentation tasks. Likewise, one logical model feeds another logical model or a physical model feeds a logical one or vices versa. For instance, in the navigational model, a node maps a logical window from the defined classes in the conceptual model and it is able to group attributes from one or many classes (relationship "*correspond to*"). On the other hand, links in the navigational model are mapped from relationships of the conceptual model. It is important to point out that we can design "n" navigational views from a conceptual model (link "*mapped from*"). This allows us to construct different hypermedia applications for different user profiles from the same conceptual schema. Such mechanism facilitates architectural pattern and model reuse. (If we will build only a user profile we could avoid, for cost and time restrictions, to build the conceptual model and work directly in a navigational model).

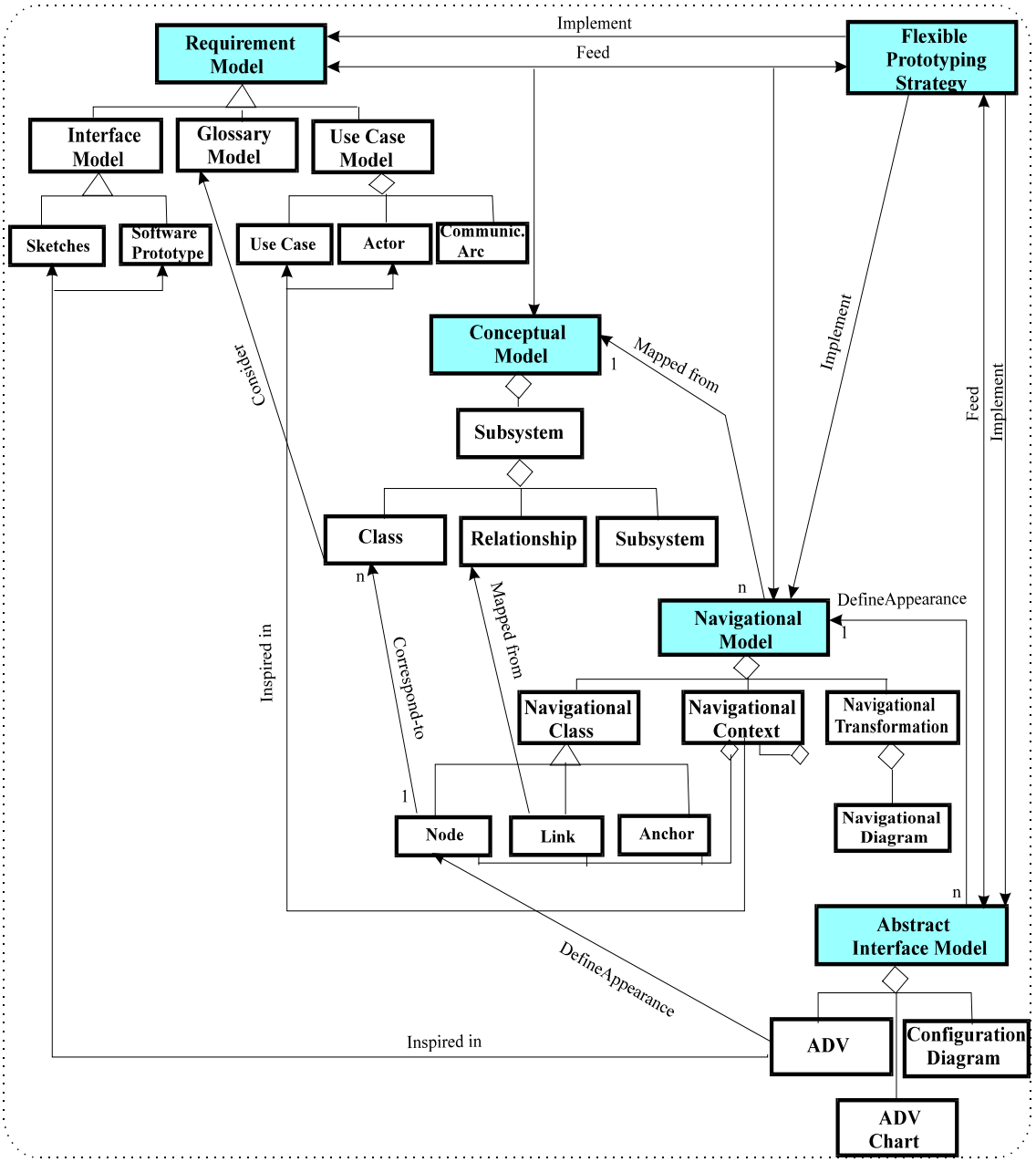

Fig. 10. Details of the Requirement, Conceptual, Navigational and Abstract Interfaces models and their relationships with Flexible Prototyping strategy.

4. Behavioral perspective of the flexible process model.

The hypermedia development process (and any software process in general) proceeds on several directions and stages simultaneously respecting some order. A process model strategy gives guidelines to help in making better choices in the development process at any time, but it should not prescribe rigid rules that will be valid for all the circumstances. However, building hypermedia artifacts by the systematic use of a process model is good enough.

As was said in the introduction section, the flexible process model, when executed, is carried out in a mix of iterative, parallel and opportunistic style allowing the partitioning of a problem into sub-problems so as to attack incrementally portions of less

complexity.

The *iterative* behavior is by definition the key of the flexible prototyping cycle: to iterate means that certain activities will be performed more than once to reach models improvement. By each **E**-**F**-**G** cycle (fig. 3) the physical model is demonstrated to the users in order to discover additional requirements, to define architectural aspects and to validate the correctness and usability of the prototype.

This strategy also allows *concurrent* activities between logical and physical models. In each cycle we can work in the prototype and update the specifications. Besides, there could exist a parallelism between activities of a navigational design stage and activities of an abstract interface design stage at any moment, and it is possible to work concurrently in a RFP while an evolutionary prototype is being implemented. So, the parallelism is a consequence of the no secuentiality between certain tasks, of the strategy of problem partitioning and of the evolutionary approach.

We will also say that the development process is *incremental*: this means that, by each iteration, there will be generic increments of the requirement, conceptual, navigational and abstract interface models as well as of the physical models. With each cycle the prototype will be refined and increased and after *n* iterations along the project, the requirements will be complete and the products will be in the operative phase.

The *opportunistic* behavior is such that, as long as the developers are engaged in prototyping and specification tasks, they follow an order dictated neither by formalisms nor by rules, but rather by following a creative mental process, as is rightly observed by Nanard et al . It is common that a hypermedia developer passes from prototyping an interface node to specifying a recently discovered navigational context (opportunistically). Then it starts to sketch it and next it goes to work in a RFP, and so on.

5. Related Works

The proposed FPM has taken into account some ideas from traditional process models such as the waterfall model and the spiral model [Boehm 88] as well as some ideas from the newer process model, that fit better with O-O constructors, such as the recursive/parallel proposal [Berard 93] or the fountain proposal [Henderson et al 90]. However, these process models do not adequately fit hypermedia development artifacts because there are models, tasks, activities and resources that pertain to the new fields of hypermedia discipline.

On the other hand, during the last years, a good number of criteria, models and techniques that put emphasis in structuring and producing hypermedia application have been published [Conklin 87, Garzotto et al 91, Grønbaek et al 94, Nanard et al 91], and, most recently hypermedia methodologies and development processes that consider some Software Engineering principles have emerged, namely: HDM [Garzzoto et al 93], RMM [Isakowitz et al 95], EORM [Lange 94], OOHDM [Schwabe et al 96], among others. Most of them (HDM, EORM, OOHDM) focus chiefly in analysis and design phases and, a little degree, in the construction phase. For instance, OOHDM is a four-stepped method that, at each stage, produces or modifies a model. We use its O-O and logical models because they establish a clear separation of concern between conceptual, navigational and abstract interfaces, as was previously discussed. Although EORM also use O-O models, such a clear separation of concerns does not exist. RMM covers almost all the phases, tasks and activities but leaves aside

the definition of tasks such as project planning, quality assurance tasks and, on the other hand, its logical models are not O-O based.

Generally, no one treats the concept of physical modeling as a fundamental strategy to guide the development of hypermedia subsystems that feeds specification, design, coding, integration and validation/verification as we propose it. We apply a set of strategies to create and evolve at the same time both logical and physical models. We consider that the logical models improve the consistency of the specifications and provide well-defined guides throughout the development cycle but the use of appropriate prototyping strategy and experimental feedback loop are also of vital importance for the hypermedia process.

6. Conclusions

We have proposed a novel approach for building hypermedia artifacts called Flexible Process Model and we have discussed the foundations of logical and physical modeling. Besides, we have presented a view of functional, methodological and behavioral aspects of the process model, stressing, in a medium level of granularity, the main phases, tasks and activities.

This process model was motivated by the fact that a more rigorous and systematic use of established Software Engineering's principles, methods and models to develop hypermedia artifacts can contribute essentially to improve quality products and users' satisfaction. Some benefits of the FPM that would be desirable to point out are:

1) it covers all the essential phases and tasks of an hypermedia project.
2) this clear break down produces greater visibility to the hypermedia project that, ultimately, contributes to project planning and scheduling, and helps to establish milestones and metrics.
3) it fosters a positive balance by a systematic use of logical and physical modeling.
4) it facilitates human communication.
5) it propitiates process improvement and standardization.

We are now working among others things, in defining a canonical conceptual model to process modeling domain that can help to understand the inherent complexity and can contribute to build the different views. An initial area of research is the designing and construction of Process-Centered Software Engineering Environments that takes into account some or all of these perspectives, to support guidance and/or enactment of software processes.

References

[Berard 93] **Berard, E.** ,1993, *"Essays on Object-Oriented Engineering",* Vol 1. Prentince Hall.

[Booch 96] **Booch, G.** ,1996, *"Object Solution: Managing the Object-Oriented Project",* Benjamin/Cummings

[Boehm 88] **Boehm, B.**, 1988, "*A Spiral model of Software Development and Enhancement",* IEEE Comp. 21, 5 (May 88).

[Conklin 87] **Conklin, J.**, 1987, "*Hypertext: an introduction and survey",* IEEE Comp. 20, 9, pp.17-40

[Connell et al 95] **Connell, J.L.; Shafer, L.**, 1995, "*Object-Oriented Rapid Prototyping*", Prentice Hall.

[Curtis et. al 92] **Curtis, B.; Kellner, M.; Over, J.**, 1992, "*Process Modelling",* Comm. ACM 35, 9; pp. 75-90.

[Davis 92] **Davis A.**, 1992, "*Operational Prototyping: a new development approach",* IEEE Software 9 ,5 (Nov 92) pp.70-78

[Garzotto et al 91] **Garzotto, F.; Paolini, P.**, **Schwabe, D.;** 1991, "*HDM, a model for a design of Hypertext Application"*, Proceed. of Hypertext'91, ACM Press.

[Garzotto et al 93] **Garzotto, F.; Schwabe, D.; Paolini, P.**, 1993, "*HDM, a model based approach to Hypermedia Application Design "*, ACM Transaction on Information System, Vol. 11, 1, Jan 93, pp. 1-26.

[Goldberg et al 95] **Goldberg, A.; Rubin, K.**, 1995, "*Succeeding with Objects: decision frameworks for project management"*, Addison-Wesley.

[Grønbaek et al 94] **Grønbaek, K.; Trigg, R.H.**, 1994, "*Design issues for a Dexter-based hypermedia system",* Comm. ACM 37, 2 (Feb94) pp. 40-49

[Henderson et al 90] **Henderson-Sellers, B; Edwards, J.**, 1990, "*The Object-Oriented systems lifecycle",* Comm. ACM 33, 9.

[IEEE 93] **IEEE Recommended Practice for Software Requirements Specifications**, 830-1993 Standard

[Isakowitz et al 95] **Isakowitz,T.; Stohr, E.; Balasubramanian, P.**, 1995, "*RMM: a methodology for structured hypermedia design",* Comm. ACM 38, 8 (Aug 95) pp. 34-48

[Jacobson 94] **Jacobson, I.**, 1994, "*Scenario-based Design",* J. Caroll Ed. ACM Press, Ch 12 : pp. 309-336.

[Lange 94] **Lange, D.**, 1994, "*An Object-Oriented design method for hypermedia information system",* Proceed. of the 27th Annual Hawaii International Conference on System Science.

[Nanard et al 91] **Nanard, J.; Nanard, M.**, 1991, "*Using Structured Types to Incorporate Knowledge in Hypertext",* Proceed. of Hypertext'91, ACM Press, pp. 329.

[Nanard et al 95] **Nanard, J.; Nanard, M.**, 1995, "*Hypertext Design Environment and the Hypertext Design Process",* Comm. ACM 38, 8 (Aug 95) pp. 49-56

[Olsina 96] **Olsina, L.** , "*View of a Process Model to Develop Hypermedia" (in Spanish),* Proceed. of the IV Congress of the SCCC (Computer Science Chilean Society), Valdivia, Chile, 1996.

[Olsina 97a] **Olsina, L.**, 1997, "*Systematic use of Flexible Process Model to build Hypermedia Artifacts".* Poster Session, Hypertext 97, Southampton, UK.

[Olsina 97b] **Olsina, L.**, 1997, "*Object-Oriented Flexible Prototyping to support Hypermedia Flexible Process Model".* III Workshop em Sistemas Multimídia e Hipermídia (WoMH 97), pp. 3-14, Sao Carlos, Brasil.

[Olsina 97c] **Olsina, L.**, 1997, *Applying the Flexible Process Model to build Hypermedia Products".* Hypertext and Hypermedia: Tools, Products, Methods (HHTPM 97), (paper accepted), Paris, France.

[Rossi et al 95] **Rossi, G. ; Schwabe, D.; Lucena C.J.P. ; Cowan, D.D.** , 1995, "*An Object-Oriented design Model for Designing the Human-Computer Interface of Hypermedia Application",* Proceed. of the International Workshop on Hypermedia Design (IWHD95), Springer Verlag .

[Rossi 96] **Rossi, G.**, 1996, "*Uma metodologia Orientada a Objetos para o projeto de aplicativos Hipermídia",* Doctoral Thesis, PUC-RIO, RJ, Br.

[Rumbaugh et al 91] **Rumbaugh, J; Blaha, M; Premerlani, W; Eddy, F; Lorensen, W., 1991**, "*Object-Oriented Modeling and Design*", Prentice Hall.

[Schwabe et al 96] **Schwabe, D.; Rossi, G. Barbosa, S** , 1996, "*Systematic Hypermedia Application Design with OOHDM",* Hypertext 96, US

[Thüring et al 95] **Thüring, M.; Hannemann, J.; Haake, J.,** 1995, "*Hypermedia and Cognition: Designing for Comprehension",* Comm. ACM 38, 8 pp. 57-66