Published in: Proceedings of the 30th HICSS, vol. VI, ed. by R. H. Sprague,
Los Alamitos, Ca.: IEEE Computer Society Press 1997, pp. 127-136

Enhancing Object-Oriented Modeling with Concepts to Integrate Electronic Documents

Ulrich Frank

Universität Koblenz-Landau
Rheinau 1, D-56075 Koblenz, Germany
E-mail: ulrich.frank@uni-koblenz.de

Abstract

This paper will present an approach that fosters a seamless integration of documents with corporate information systems. It is based on a conceptually enhanced notion of documents that promises more meaningful ways of processing documents and increased system integrity as well. The approach resulted from enhancing an existing methodology for designing object-oriented enterprise models as well as a design environment that is based on it. It suggests to widely abstract from current appearances of documents. Instead it recommends to begin with focusing on business processes. Analyzing the information required within a business process results in a preliminary object model. Various templates and checklists will then guide the analyst with identifying document candidates within the object model and specifying their semantics - also taking into account requirements that may result from providing transformations into standardized document representations such as SGML, HTML, EDIFACT, etc.

1. Introduction

For a long time documents have been of essential importance to business firms. They capture a large amount of information that is needed for decision making and for internal as well as for external communication. Currently we observe that documents are gaining even more attention. This is for different reasons. At first sight there is a technology push: There are new document-centered IT-systems that promise to effectively support mission critical tasks. An increasing number of vendors offer so called "document management systems" which usually focus on scanning, digitizing, and archiving paper documents. Often those systems are regarded as key technologies for enabling companies to accomplish effective business (process) reengineering. Another tremendous push is caused by the growth of the internet, namely the World Wide Web, which seems to become an outstanding medium for publishing certain kinds of documents. Other technologies that heavily rely on documents are workflow management systems, groupware software, or Computer Supported Cooperative Work (CSCW) systems in general. Both for document based software as well as for document exchange standards or wide spread proprietary concepts are of crucial relevance. Currently a few standards are already in use (like SGML [8], HTML [15], EDIFACT, MIME), while the practical relevance of others (like ODA/ODIF, [2]) is questionable. In addition to that there are new concepts as well as corresponding technologies that promote an essential new way of looking at documents (like OpenDoc [17] or OLE [5]).

Beside reacting on pushing technologies the shortcomings of the ways documents are currently dealt with are another reason to focus on documents. In contrast to widely accepted concepts and technologies applied for transaction-oriented systems documents are often handled in a cumbersome and hazardous way - although this aspect may sometimes be hidden by the convenience offered by state of the art text processors and spreadsheet calculators at first sight. However, a closer look at those products reveals that they are hardly sufficient to establish corporate document management systems that efficiently support cooperative work and that are efficiently to maintain at the same time.

There is a remarkable amount of recent publications (for instance: [9], [13], [18]) that deal with documents from various perspectives. While all of these research areas focus on important aspects of document handling in general, none of them however is precisely dedicated to concepts that aim at a tight integration of document storing and processing both with a company's business processes and its overall information system.

2. The Current Situation

Within about a decade electronic documents have become a natural part of today's office work. More and more business professionals are skilled in using document processing software in a productive way. Acting on competitive markets vendors make a great effort to design products such as text processors, graphical editors, spread sheet calculators, and additional utilities like text retrieval software according to their customers' needs. State of the art systems provide a wide functionality on an impressive level of convenience. At first sight it seems that there is only little left for further improvements. However, appearances are deceptive: Dealing with electronic documents often suffers from severe misconceptions that may have a considerable impact on a company's overall performance. With new communication technologies like electronic mail, EDI, or the World Wide Web they will cause even worse problems.

2.1 A Scenario

Consider the following scenario. A product manager with a vendor of computer hardware is using state of the art technology for preparing and handling various documents he is exchanging with customers, sales people, and other executives. Information on products and promotions has to be regularly sent to customers and sales people. Large parts of the documents for both groups of addressees are identical (which is, however, not true for the layout). Furthermore some of the content can also be used for internal reports. For this reason the product manager's assistant is glad that there is a feature like copy&paste, which is also needed to import data (like product features and prices, digital images of products, etc.) that are stored in a central database which can be accessed almost transparently. Sometimes documents include references to files that are used to store spread sheets or digital images - the latter gaining more and more importance with the company's recent effort to digitize almost any incoming letter. Establishing such references can be done in a rather convenient way (at least in case the file's location is known).

Since not everybody in our little scenario is using the same technology (or has the same preferences respectively) some transformation has to be done. There are a few progressive customers who prefer electronic mail. While most of them wish to get electronic documents in a representation they can edit with the text processor of their choice, not everybody has electronic mail software available that can handle MIME attachments. That makes it necessary not only to generate a number of different file formats but also to encode them into a stream of transmittable characters which then has to be included into an electronic mail message. And of course there is the Web: Since the company has decided to use the internet for enriching its communication mix the latest information on products has to be placed on the Web as well. The new version of the assistant's text processor features a converter that generates HTML sources on the fly. However, experience gathered so far shows that those sources do not result in satisfactory Web pages. Manually editing them is usually rather time consuming - with the effect that those pages are not always up to date.

Various executives have to be provided with printed monthly reports. Those reports all have a similar structure. There are some differences in the content (for instance: one manager always wants to have certain information on a particular customer to be included in his report). Furthermore the executives have different preferences for the graphical

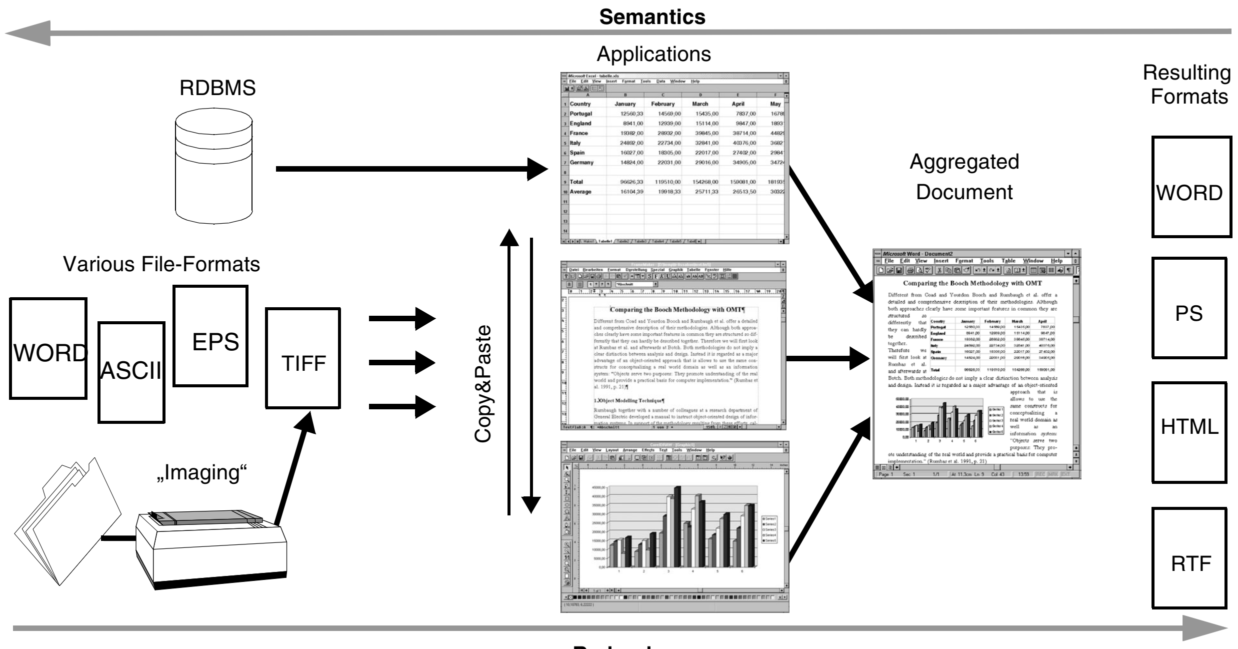


Fig. 1: Creating Documents in a State of the Art Environment

presentation of data. Those variations keep the assistant busy - sometimes too busy: Occasionally it happens that a report does not correspond to particular requirements. Searching for documents is supported by a text retrieval system. While it is amazingly fast, it is not always able to find the documents the user has in mind. This is the case whenever the user does not know a characteristic text pattern, but only remembers properties such as "a bar chart on the revenues for laser printers from 1990 to 1994".

2.2 Reasons for present problems

This little scenario, which is hardly exaggerating the complexity that arises from the way documents are produced and handled today, illustrates a somewhat bizarre situation: Even with modern equipment and skilled users state of the art document handling is afflicted with severe problems. Despite the convenient use of copy and paste *integration with existing data* happens on a rather low semantic level - resulting in a great amount of redundancy. This is not only a threat to system integrity but also a waste of resources required for updating redundant information - not to speak of the cost that is caused by inconsistent documents. Consider for example a document that contains data representing a customer. In this case it will certainly make a difference whether this data is simply characterized as an array of bytes which might be interpreted in various ways or as an instance of a concept "Customer". This example also illustrates the low level of integration provided by today's platforms. Imagine you had copied a number of strings representing a customer from a database to a report you prepare with a text processor. As soon as the data is pasted into the document it will loose most of its semantics. Within the text processor there is no customer any more. So there is no way to get access to related information (like revenues) from within the text processor. Furthermore you may change the strings representing the customer producing an inconsistent state (for instance by overwriting the average monthly revenue).

Furthermore the storage technology is not appropriate: While it has been the general consensus for many years, that corporate data should be stored and managed by data base management systems this is different with documents: They are usually still stored as files - with all the shortcomings that file systems impose on integrity and security. There is still a remarkable amount of time consuming manual work which in principle could be automated. For instance: Transforming a document to a number of acceptable web-pages. Restricted retrieval capabilities are another example. Another severe problem is the *lack of standardized Representations*: Exchanging documents is often difficult since not everybody is using the same representation. A situation that becomes worse with an increasing number of alternative media which can be used to spread documents.

2.3 The need for conceptual modeling

To accomplish both a high level of integration and of it is necessary to enrich electronic documents with an appropriate amount of semantics. Furthermore it has to be taken into account that documents are often in a way reifications of business processes (the German word for process - "Vorgang" - can also be used to denote the document that is subject of a process). For this reason existing documents may symbolize business practices that ought to be overcome. Taking them for granted will hinder the discovery of alternative ways to organize a business process. In order to exploit the potential offered by electronic documents it is necessary to widely abstract from traditional notions of documents and familiar ways to use them. For this purpose we need an appropriate conceptualization of electronic documents, and - based on that - an approach to design conceptual models that are suited to provide a semantic foundation to integrate data with documents.

From a general point of view traditional data as well as documents can be conceptualized as objects or classes respectively - an idea that has been discussed for long, especially in the area of office automation (see for instance [12], [19]). A document as an object would have a structure its data is stored in and a behavior that is defined by the operations available for accessing, manipulating, and maybe presenting the encapsulated data. Both the structure and the behavior would be specified/implemented in the class a particular document is an instance of. Furthermore one could take advantage of generalization/specialization - for example with specializing "MonthlyReport" from "PeriodicalReport". Like any object a document may be composed of other objects. A document implemented as an object would allow for operations like "search all documents that contain revenues" or "substitute a bar chart representation with a pie chart". Conceptualizing documents as objects would also allow for creating so called hypermedia- or multimedia-documents: Objects may have various references to other objects and they may be visualized in any way that seems to be appropriate for the information they provide. From a software-engineering point of view it seems to be sufficient to regard electronic documents as objects. However, for designing conceptual models of application domains it is not advisable to treat objects and documents as synonyms. In order to take into account the role documents play for organizing and coordinating office work it seems to be more appropriate to characterize documents as objects with special features. For objects that serve as documents it is not sufficient to manage the information they encapsulate. The way the information is presented is of essential relevance, too. This includes basic presentation modes (such as text, graphics, video, audio) as well as the overall layout - which may be varying depending on user preferences. Sometimes documents have to be created for legal or institutional reasons. Furthermore the notion of a document is essentially associated with the proof of evidence. That implies measures to prove the identity of the author(s) or the person(s) whose interests and intentions are involved. Documents are usually media for exchanging information. This requires means to represent their content in a way that gives the receiver an acceptable chance to interpret it according to the sender's intention. This suggests to provide transformations into appropriate standardized representations for electronic documents.

While an object-oriented approach in general seems to be appropriate, general object-oriented modeling methods ([4], [10], [15]) are not sufficient for our purpose: They lack enhancements that support the specification of special document. Furthermore they do not include specific concepts to analyze and - if required - to redesign business processes. Dedicated document modeling methods (like [9], [16]) on the other hand primarily focus on describing documents without regarding associations to other objects within an information system. They do not provide any support to model business processes as well. Standardized representations of documents such as SGML or ODA/ODIF are sometimes recommended as the key to effective corporate document management (for instance: [18]). They are of crucial importance for efficient information exchange (see above) and for the use of mass produced software as well. However, they are certainly not suited for designing conceptual models.

3. A methodology for the conceptual design of integrated document management systems

In order to allow for illustrative object-oriented modeling of information systems that support a tide integration of documents and other objects we used an existing methodology for object-oriented enterprise modeling that we had been working on for a few years already. It includes concepts for analyzing business processes and information flow within office domains. For this reason it is in principle well suited for describing the information residing in electronic documents. Applying the methodology however revealed that it would be very helpful to take into account the special concepts and requirements related to documents. For this reason we enhanced the methodology as well as the development tool that is based on it with document-oriented features.

3.1 Multi Purpose Enterprise Modeling (MEMO): An overview

Unlike other object-oriented methodologies (like [4], [10], [15]) MEMO (Multi Purpose Enterprise Modeling) is not only focusing on software or information systems in general. Instead it provides a framework for coordinated modeling of business information systems, a company's organization, and corporate strategy (see [6], [7]). MEMO is accompanied by a development environment (MEMO Center) that allows the user to navigate through the views of an enterprise model on various levels of detail. MEMO Center has been implemented in Smalltalk, using the VisualWorks® environment. It controls a model's integrity and allows for simulation and fast prototyping. MEMO is based on the assumption that an enterprise model provides both a foundation for integrated information system and a medium to systematically monitor and improve a company's performance. It suggests to start with defining the corporate strategy. For this purpose it provides modeling concepts that are based on Porter's value chain approach. A systematic decomposition of the value chain will then help to identify core business processes. Modeling business processes is of crucial importance for applying MEMO. Not only that this is the prerequisite for organizational redesign, it also fosters the identification of objects. A business process is modeled as an ordered graph of tasks, where a task in itself can be decomposed into other tasks. A task requires, operates on, and produces information. Information is grouped into three categories: information which resides on a computer based information system, forms, and files. Forms have a formal structure, that is they contain fields, have well defined states (like 'complete', 'incomplete', 'consistent'), and may be a set of constraints which define the permissible operations. Their content may be changed within a task. The term file is used to summarize documented information that is read-only - like office files, letters or journals. The information used within a business processes serves to identify object candidates for designing a preliminary object model.

Each process is assigned an organizational context by referring to the organizational units which are responsible for its execution. Furthermore each process can be assigned business rules like: "All tasks should be managed by the same person.î The tasks can be described in a very detailed way - depending on the effort that is to be spent for analysis. Among the more important features are: minimum and maximum execution time, decision rules, required resources (for instance: copy machine, printer), exceptions, people (roles) needed to communicate with, communication media etc. In order to support the re-design of business processes MEMO provides the analyst with means to analyze existing processes and design heuristics. The model of a business process allows the analyst to detect media clashes (for instance: information that originally resides in the information system and is then being transferred to a form), to identify bottlenecks, or to draw communication nets. By adding statistical data on workload, capacity and probabilities of the tasks' possible outcome the model can be utilized to perform simulations.

System design aims at refining and formalizing the object model resulting from analysis. Among others this includes the specification of attributes and services, the refinement of generalization/specialization hierarchies, and the specification of transactions. Furthermore it may be necessary to conceptually integrate existing data and applications by encapsulating them within appropriate objects. In order to minimize semantic gaps between analysis and implementation the concepts used in the different phases (like value chains, business processes, organizational context) are all defined in a common object-oriented meta model.

3.2 Document-oriented enhancements

Tailoring MEMO for the design of corporate document management systems implied to take into account a number of requirements. First it was necessary to introduce a notion of electronic documents that includes essential document semantics without being restricted by the traditional notion of paper-oriented documents. Based on such a conceptualization a specialized modeling methodology should help with the identification of document candidates and the detailed specification of document semantics. Since the presentation of information is essential for documents it is desirable to support the design of appropriate presentations on a conceptual level already. At the same time the methodology should help with preparing for efficient and consistent mappings to the relevant standard representations.

For the purpose of conceptual modeling we define electronic documents as (composed) objects with the following characteristics:

- they have a specific meaning within a certain context of action - like a business process.
- they serve to coordinate cooperative work within this context. That includes the exchange of information between participating actors.
- they help to record the way this work is done - thereby allowing to inform about the present state of a process or relevant states of the past. This function may be implied by legal or institutional regulations.
- furthermore a document provides it's users with one or more modes to present and optionally to manipulate the encapsulated information.

Both the idea of editing objects within a document and the use of object references are essential properties of a radically new approach for designing and using information systems: It suggests an enhanced concept of a document to become the general metaphor for interacting with object-oriented information systems, thereby overcoming the traditional notion of an application. A document contains all the objects relevant within a particular context (a task, project, etc.). Depending on the user's access rights the objects can be viewed or edited within the document. OLE2 as a proprietary technology (see [5]) gives a first impression of these future systems. OpenDoc, a technology propagated by several different vendors goes one step further: it provides a layer on top of various operating systems that will allow - provided certain preconditions are fulfilled - to exchange documents together with the embedded objects within heterogeneous environments (see [13], [17]). The concept of electronic documents suggested above is compatible with these new technologies. Thereby the investments in conceptual models should be protected against future technological changes.

3.3 Identification of document candidates

The purpose of analysis is to get a deep understanding of the business functions that have to be fulfilled in the relevant domain. We found that often it is not helpful to ask what information or objects are needed at first place: Normally people will not be able to give a complete and sound picture, and - more important - by only looking at information you have hardly any measure how to evaluate the way it is used. Instead focusing on processes proved to be more promising, since most business professionals seem to prefer process- or task-oriented ways to conceptualize office domains. At the beginning of analysis we recommend to widely abstract from documents. Instead one should concentrate on information content and essential functionality. This will result in models of business processes that incorporate any information that is relevant - no matter whether it may reside within a computer system or on traditional media. Those models are then to be analyzed in order to overcome present redundancies and media clashes. Thereby it should be found out whether and how information that currently resides outside computer systems is to be digitized. In order to prepare for the following design phase it is now time to abstract from that information, that has to be kept on traditional media. The process model thereby becomes a workflow model.

The information that is required and produced within a workflow now has to be described within an object model. Typically a task within a workflow operates on objects of more than one class. For this purpose we introduce the concept of a "context specific object collection". It is not only specified by the semantics of the contained objects. Furthermore the objects' states have to be taken into account, too: A task is triggered by a certain state of a context specific object collection and may produce one or more other states of it. Other context specific object collections are used to specify the units of information that are exchanged with other actors.

Only now it is time to shift the focus to documents. The first candidates for becoming documents are the context

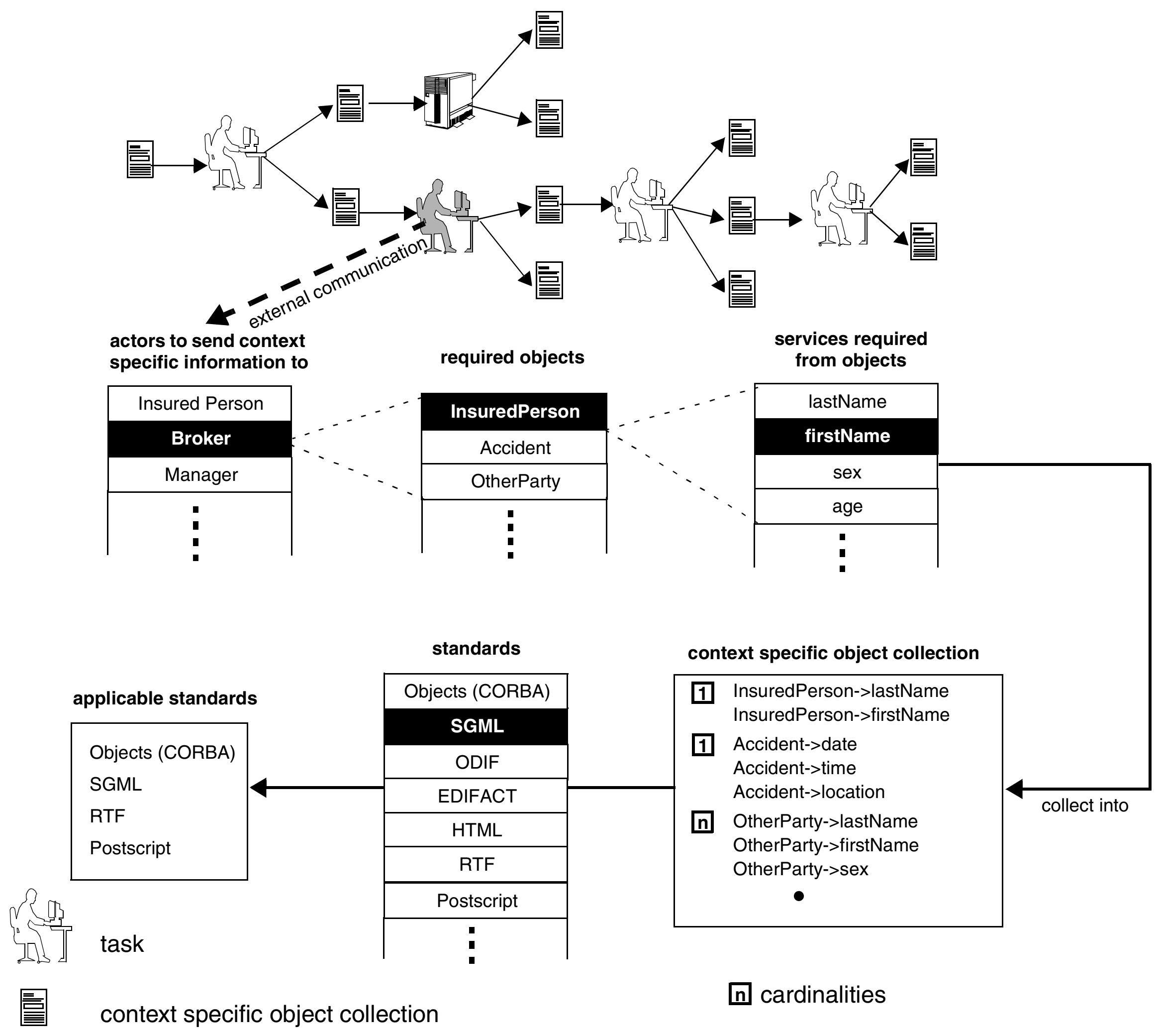


Fig. 2: Deriving document candidates from workflow models

specific object collections that are required and produced by tasks within a workflow. Their function is to provide the information that is worked on within a workflow, and - literally - to document the state of the workflow. Other candidates are the object collections that serve to exchange information between the participating actors. In order to decide whether detected document candidates should be in fact conceptualized as documents they have to be checked against the essential document features (see 3.2). Besides those logical document candidates one has to take into account that sometimes documents have to be created for legal or institutional reasons. The definition of their logical structure, their layout, or the media they have to be presented on can be more or less restrictive. Examples for documents of this kind are tax forms, contracts, dissertations - or papers submitted to a conference. In other words: Those documents are not primarily a vehicle to fulfill a task. Instead their creation is a task on its own. This implies that with those documents it is hardly possible to abstract from present occurrences - at least in the short run.

3.4 Specifying document semantics

Analysis will result in model of future workflows that contain references to a preliminary object model as well as descriptions of possible document candidates. Furthermore it specifies the object collections that are to be exchanged with other actors or machines. The design phase aims at formally refining these models in greater detail. In general this requires attributes to be assigned and specified by defining a type or class as well as other constraints proposed by MEMO´s object model. The services, too, will be specified in more detail by assigning pre- and postconditions, exceptions possible during execution, etc. (see [7]). Furthermore the generalization/specialization relationships between classes have to be defined - as well as the associations between objects (for instance: InsuredPerson *uses* Contract). The document candidates have to be conceptualized as classes. Those classes have to be enriched with document-specific semantics. This includes the *specification of additional information*, the *definition of layout and interac-*

tion-semantics as well as *a conceptual support for standard representations/protocols*. Furthermore it may be necessary to specify classes that allow to encapsulate existing document files and document editors.

In addition to the information specified for the objects within the context specific object collections it may be necessary to add document-specific information - like date and/or time, a document title, the name of the author or textual comments. For any of this additional information it has to be decided whether it can be taken from existing classes or from classes still to be added to the object model. For any information it has to be defined whether it's value or it's object-reference should be copied into the document.

In order to prepare for a convenient definition of the layout associated with a document class MEMO allows for enhancing the class description with presentation specific details that, however, are not restricted by the limitations of static read-only media. Each class an attribute is declared as can be assigned a default presentation (like a "textView") and a label that is to be presented with an instance of the attribute. Objects may also include services other than those that simply provide access to attributes - for instance: a service that calculates a person's age from his date of birth. To cover the presentation of those services as well, their input and output objects can also be assigned a default view. With this additional semantics in place MEMO Center allows to generate a preliminary user interface (in other words: a default presentation) for a document class. The generated layout may then be modified manually - the window size can be changed, as well as the size and position of widgets.

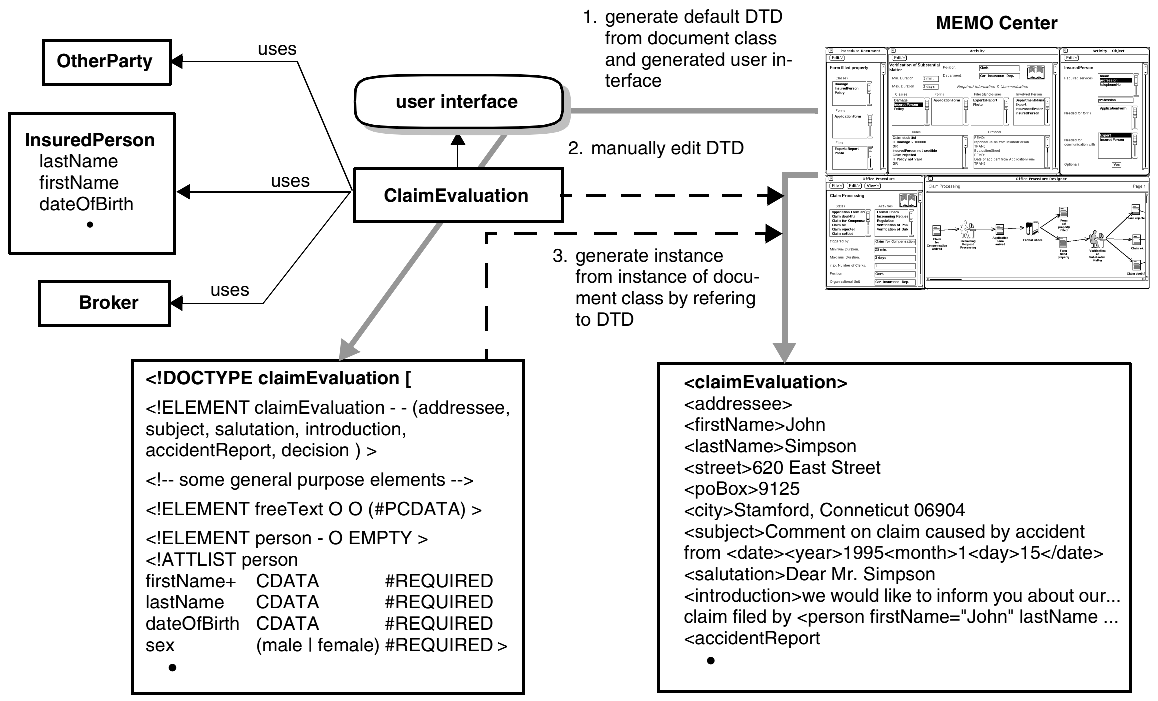
Notice that until this point there is no need to specify the medium (like telephone, fax, e-mail, etc.) that is used for transmitting a document. This abstraction is for a good reason: Something that is sent as fax today may be sent as a MIME-mail tomorrow or as an object the day after tomorrow. Apart from this abstraction it is a good idea to assign those protocols to a document class that could in principle be applied to represent it, because this is rather a conceptual than an implementation decision (see fig. 4 for an example). The complexity of this decision may increase when one standardized representation is embedded within another. For instance: an order as an EDIFACT message embedded in an ODIF document (for a detailed example see [1]).

Currently we mainly focus on three representations/protocols: The interface description language (IDL) proposed by the Object Management Group (OMG), SGML and HTML. In order to preserve as much document semantics as possible the OMG's IDL is the first choice: In principle it allows to wrap a standardized interface around an existing class implementation without changing it's semantics. The CORBA (Common Object Request Broker Architecture) technology proposed by the OMG ([3]) will then allow for transparent communication within distributed heterogeneous environments. The object model proposed by MEMO contains the information that is necessary to define such an interface (like class hierarchies, classes of attributes and signatures of operations). For this reason an IDL-interface does not require additional specifications. Instead MEMO-Center allows to directly generate IDL-code from the object model.

Mapping a document to SGML can hardly be accomplished without additional specifications. This is for various reasons. SGML requires the state of an electronic document to be represented as a sequence - according to a specific syntax. A document class per se does not contain clear information about how to create such a sequence from it's instances. Since SGML does not allow to represent various data types there has to be an appropriate mapping of the objects delivered by a document's services to the strings that are to be used within a standard representation. Finally it has to be taken into account that a specific SGML document usually does not require all of the information provided by an instance of a document class. For example: The document class "ClaimEvaluation" (see fig. 3) allows to access all the services offered by the embedded object "Broker" - like address, number of employees etc. A SGML-report prepared for a department manager will probably not contain these details. In order to specify the mapping to SGML MEMO suggests the following steps:

- MEMO Center parses the user-interface generated for the document class, selecting the read-only services. It then takes the names of the selected services and the class-names of the embedded objects they belong to as defaults for creating corresponding SGML tags. The tags are ordered in a way suggested by the position of the corresponding widgets within the user-interface. The tool assigns the tags with cardinalities according to the cardinalities specified in the object model (see "firstName" in fig. 3). In order to generate a preliminary DTD MEMO Center uses a generic DTD template that is then initialized with the tags and default element types (usually PCDATA).
- The default tags created by the tool as well as the element types may now be changed in order to correspond to existing conventions.
- Since a user-interface is not a sequential medium the default mapping outlined above may be partially arbitrary. If the elements of the generated DTD are ordered in a way that is not acceptable, it can be manually edited - resulting in a final DTD that is stored within the object model. The tool prevents to delete services from a class as long as they are referenced by a DTD.

The final DTD now defines how to map an instance of the document class to an instance of the DTD. MEMO Center stores the associations between tags and corresponding


OtherParty
uses
InsuredPerson
lastName
firstName
dateOfBirth
uses
Broker
uses
user interface
ClaimEvaluation
1. generate default DTD from document class and generated user interface
2. manually edit DTD
3. generate instance from instance of document class by refering to DTD
MEMO Center

```
<!DOCTYPE claimEvaluation [
<!ELEMENT claimEvaluation - - (addressee,
subject, salutation, introduction,
accidentReport, decision ) >
<!-- some general purpose elements -->
<!ELEMENT freeText O O (#PCDATA) >
<!ELEMENT person - O EMPTY >
<!ATTLIST person
firstName+   CDATA            #REQUIRED
lastName     CDATA            #REQUIRED
dateOfBirth  CDATA            #REQUIRED
sex          (male | female)  #REQUIRED >
  •
```

```
<claimEvaluation>
<addressee>
<firstName>John
<lastName>Simpson
<street>620 East Street
<poBox>9125
<city>Stamford, Conneticut 06904
<subject>Comment on claim caused by accident
from <date><year>1995<month>1<day>15</date>
<salutation>Dear Mr. Simpson
<introduction>we would like to inform you about our...
claim filed by <person firstName="John" lastName ...
<accidentReport
  •
```

Fig. 3: Mapping a document class and it's instances to SGML

services that deliver the information that is necessary to initialize the SGML document. Fig. 3 illustrates this mapping in a simplified way. Based on the specification MEMO Center generates the code that is necessary to perform the mapping (that includes, for instance, transformations of non-string objects delivered by the selected services into strings required for SGML documents).

Different from SGML HTML implicitly includes layout-semantics. Although HTML is originally based on SGML it is possible to map an instance of a document class to an HTML document without additional specifications. As outlined above MEMO Center allows to generate prototypical user interfaces for any class. The specification of a prototypical user interface can now be used to generate a corresponding HTML document. It has to be taken into account however that not all of the widgets can be represented in HTML in an equivalent way. This approach to generate HTML can be find in a similar way in commercial software like VisualWave®.

So far documents within the object model have been specified in a top down approach. However, usually it is also necessary to still use existing applications and the data structures they operate on. In order to take account of such existing components it is necessary to specify classes that provide a conceptual description of those components. For instance: class 'WordDocument' represents documents produced by a certain word processor. In order to integrate them with a more sophisticated model one can use an association labeled with the reserved denotation "represents". For example: "WordDocument represents Document". Document could then serve as a dummy class that would delegate to the services provided by WordDocument. Other classes would serve to wrap existing files and applications. Although a conceptual model is a prerequisite for integration, it does not guarantee a satisfactory solution: Often the interfaces provided by existing components are rather poor.

4. Remarks on the architecture of integrated document management systems

From our point of view the notion of an integrated document management system is misleading. Our vision is characterized by integrated object systems where some objects may have the special semantics that makes them documents (see 3.2). The objects would be stored in a central object management system that would allow the users to interact with the objects via context specific (we could also say: document-oriented) user interfaces in a distributed heterogeneous environment. The object management system would control the constraints implied by the object model it is based on. The exchange of information would be accomplished via the transmission of objects. This level of communication would be supported by reference object models on the application level - like the standardized business objects planned by the OMG. However, how does this vision relate to current situation? Dealing with documents today usually implies the use of proprietary editors - with proprietary file formats. They are hardly suited for an efficient integration into open object-oriented information systems. The only chance to overcome the burdens imposed by proprietary editors in the short run are standards like SGML - may be DSSSL (Document Style Semantics and Specification Language, [18], pp. 115) as well - and HTML. With the tremendous popularity of the web and an increasing awareness of the benefits imposed by SGML more and more vendors offer support for HTML and SGML. This may allow to abstract from the peculiarities of proprietary formats in future information systems. However, the user of such a system would usually not directly edit HTML or SGML code. In order take advantage of the highest semantic level that is appropriate within a given context he would rather interact with objects. In the ideal case those standard representations would be created and maintained transparently. They would only serve to exchange information and to use existing software. They would however not be used to store information - very much like the use of Postscript today: generate the representation for a temporary purpose and then forget about it.

The number of standard representations to be supported by a system may vary in time. For this reason it is a good idea to separate domain objects from those objects which are responsible for the mapping to the standard representation. The domain objects should not have to know which standard representation depends on them. This design challenge is very similar to the task of managing user interfaces. For this reason we have prototypically implemented an architecture, that is based on the Model View Controller (MVC) concept used in Smalltalk ([11]). It supports a transparent mapping to SGML. Currently we do not use a controller object because it is not intended to update the model by changing the SGML-representation. Every SGML document is managed by an instance of the class "SGMLDocManager". This class provides a functionality that is similar to a view within MVC. Every instance of the class subscribes to certain aspects within the model. Whenever the model sends a notification that one of these aspects has changed, the document manager asks the model for the new state by sending the message that corresponds with the particular aspect. It then maps the change to it's SGML document. There are two basis approaches to perform the mapping: generate the whole structure or update an existing structure by propagating the elements that have changed. Since the SGML-representations are either as files or as streams within an ODBMS an efficiently updating them is hardly possible. Therefore SGMLDocManager always generates a new structure, replacing the existing one. In case the mapping should only be performed at a explicit request by the user, the document manager would only subscribe to an aspect that relates to a specific user action (like pressing a "Print" button).

Treating documents as composed objects results in a notion of documents, that is somewhat different from the one proposed by technologies like OpenDoc. It suggests to challenge the holy WYSIWIG metaphor: While user-interaction with objects implies a (graphical) interface, that does not necessarily mean to always take into account the layout of the document the object is embedded in. This is for various reasons. An object may be embedded in many documents, each of them presenting it in different ways. In these cases it does not make much sense to put emphasis on WYSIWYG when editing an object. Different users of a document may have different layout preferences. On the other hand there may be unified corporate layout regulations that do not leave any space for individual variations. Finally the traditional argument against WYSIWYG: Why should a user, who is usually not a layout expert, waste his time with creating awkward document layouts?

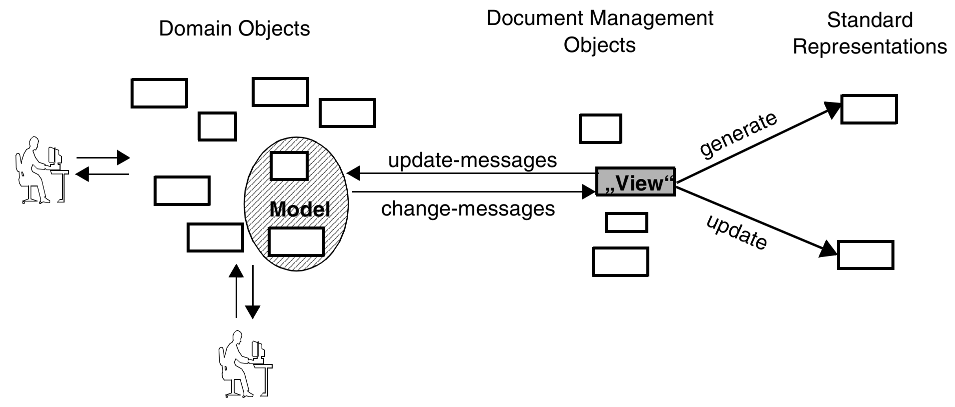


Fig. 4: Adapting MVC for transparent mapping to standard representations

5. Conclusions

The present notion of a document is still essentially influenced by the idea of printed paper. While many companies are currently in the process of scanning existing paper documents, the resulting digital images do not provide an acceptable orientation for efficient information systems: They are only a symptom of a temporary migration phase - nevertheless this migration often causes substantial investments. The current hype about the web and HTML respectively as well as today's fancy document editors can hardly hide the fact, that something like document management does hardly exist. Instead there is the management of files with unknown content. In order to prepare for information systems that integrate documents and data on a high semantic level it is necessary to design appropriate conceptual models. Those models should also help with redesigning the organization of cooperative work in order to fully exploit the potential offered by electronic documents. While in principle an object-oriented conceptualization of documents seems to be very promising, general object-oriented modeling methods (like [4], [10], [15]) lack concepts to model documents and the application domains they are used in. Enhancing MEMO with document-oriented concepts proved to be a helpful approach so far. We have applied the methodology to design and to prototypically implement an object-oriented information system for our institute. The implementation was mainly done with VisualWorks® and Gemstone®.

References

[1] Acebron, J., Appelt, W., *Including Application Specific Information into ODA Documents: A Case Study for EDIFACT Messages and STEP Product Model Data*. Resarch Paper No. 812, GMD, Sankt Augustin, 1994

[2] Appelt, W., *Document Architecture in Open Systems. The ODA Standard*. Springer, Berlin, Heidelberg et al., 1991

[3] Ben-Natan, R., *CORBA: A Guide to Common Object Request Broker Architecture*. McGraw-Hill, New York et al. 1995

[4] Booch, G., *Object-oriented Analysis and Design with applications*. 2nd ed., Benjamin Cummings, Redwood/CA., 1994

[5] Brockschmidt, K., *Inside OLE2*. Microsoft Press, Redmond/ Washington., 1993

[6] Frank, U., "An Object-Oriented Methodology for Analyzing, Designing, and Prototyping Office Procedures", *Proceedings of the 27th HICSS* (Nunamaker, J. F., Sprague, R. H., Eds.), Vol. IV, IEEE Computer Society Press, Los Alamitos/CA., 1994, pp. 663-672

[7] Frank, U., "MEMO: A Tool Supported Methodology for Analyzing and (Re-) Designing Business Information Systems", *Technology of Object-Oriented Languages and Systems* (Ege, R., Singh, M., Meyer, B., Eds.), pp. 367-380. Prentice Hall, Englewood Cliffs, 1994

[8] Goldfarb, C., Rubinsky, Y., *The SGML Handbook*. Clarendon Press, Oxford, 1990

[9] Isakowitz, T., Stohr, E.A., Balasubramanian, P., "RMM: A Methodology for Structured Hypermedia Design", *Communications of the ACM*, Vol. 38, No. 8, pp. 34-44, 1995

[10] Jacobson, I. et al., *Object-Oriented Software Engineering. A Use Case Driven Approach*. Addison Wesley, Reading/Mass. 1992

[11] Krasner, G.E., Pope, S.T., "A cookbook for using the model view controller user interface paradigm in Smalltalk-80", *Journal of Object-Oriented Programming*. Vol. 1, No. 3, 1988, pp. 26-49

[12] Lochovsky, F., Lee, A., "Document Management Systems", Tsichritzis, D. (Ed.): *Office Automation*. Springer, Berlin/Heidelberg 1985, pp. 21-40

[13] Reinhardt, A., "Managing the New Document", *BYTE*, August, pp. 91-104, 1994

[14] Raggett, D., *The Definitive Guide to HTML 3.0. Electronic Publishing on the World Wide Web*. Addison-Wesley, Reading, Mass. 1995

[15] Rumbaugh, J. et al., *Object-oriented modeling and design*. Prentice Hall, Englewood Cliffs/NJ et al. 1991

[16] Schwabe, D., Ross, G., "The Object-Oriented Hypermedia Design Model", *Communications of the ACM*, Vol. 38, No. 8, 1995, pp. 45-48

[17] The OpenDoc Design Team, *OpenDoc Technical Summary*. Component Integration Laboratories 1994

[18] Travis, B. and Waldt, D., *The SGML Implementation Guide. A Blueprint for SGML Migration*. Berlin, Heidelberg, etc., Springer, 1995

[19] Tsichritzis, D.C., Form Management, *Communications of the ACM*, Vol.25, No.7, July, pp. 453-478, 1982