# Design of the SGML-based electronic patient record system with the use of object-oriented analysis methods

E. Kuikka, A. Eerola, J. Porrasmaa,
A. Miettinen, J. Komulainen

**Report A/1999/1**

ISBN 951-781-609-X
ISSN 0787-6416

UNIVERSITY OF KUOPIO

Department of Computer Science
and Applied Mathematics

P.O.Box 1627, FIN-70211 Kuopio, FINLAND

# Design of the SGML-based electronic patient record system with the use of object-oriented analysis methods

E. Kuikka, A. Eerola, J. Porrasmaa, A. Miettinen
Department of Computer Science and Applied Mathematics,
University of Kuopio

J. Komulainen
Department of Pediatrics
Kuopio University Hospital

A patient record is typically a document updated by many users, required to be represented in many different layouts, and transfered from place to place. It is also an object for various types of queries. Opposite to the fixed-length data typical for administrative data the clinical data is usually represented as a free text. Thus, the patient record is a good candidate to be represented structured and coded using the SGML document standard.

The use of the SGML requires that the structure of the document is defined in advance by a Document Type Definition (DTD) and the document follows it. If the structure is defined based on the paper form, the structure does not necessary reflect what the doctors and nurses actually make in their everyday practise. This paper represents a method which derives an SGML DTD by starting from the description of the usage of the patient record in medical care and nursing.

## 1 Introduction

The electronic patient record (EPR) [Car94, BPD96] is an answer for the well-known difficulties in the use of the paper patient record [Bra94, PSRL94]. Numerous electronic patient record systems have been implemented in particular environments with the use of some predefined techniques (for example, certain databases, certain programming languages, etc.). This approach has influenced new problems when additional features have been added into the system, when data has been transfered between various systems and when old data needs to be processed with new program versions.

An opposite approach to these specific implementations is taken in a set of systems or prototypes which represent the patient record as a structured document containing the standardized markup [Ben96, PGR+97, YOO+98]. SGML (Standard Generalized Markup Language)[ISO86] is an international standard to define the markup notation used in structured documents. An SGML document follows the structure which is defined in advance with the use of a Document Type Definition (DTD). The DTD can be designed by considering the paper patient record and by defining the potential semantic elements and their relationships [MEA96, TW96]. The derived structure may, however, conflict with what doctors and nurses actually require in their everyday practices. The problem here is that the necessity of information is not considered and new information requirements may not be found. Further, the possibilities evaluate the medical care and nursing routines may not be discovered. Also, the queries from the EPR are not considered.

The aim of the paper is to develop a method for the derivation of an SGML DTD by starting from the description of the utilization of the patient record in medical care and nursing. Here we emphasize the improvement of the medical treatment, the rationalization of the information flow and the quality of the system. The method has the following steps:

1. The flow of the data in the clinic is considered from the patient records point of view. The traditional dataflow analysis [You89] is used.
2. The conceptual modeling is done using the Unified Modeling Language (UML) [Boo98]. Here it is important that the model describes the medical treatment in a natural way.
3. The tree diagram [MEA96] of the SGML DTD is created from the UML class diagram using a set of rules presented in Section 4.

# 2 UML class diagrams

Figure 1 presents an example of a part of the class diagram of the simplified patient record with the most important concepts and notations of the object-oriented approach explained below.

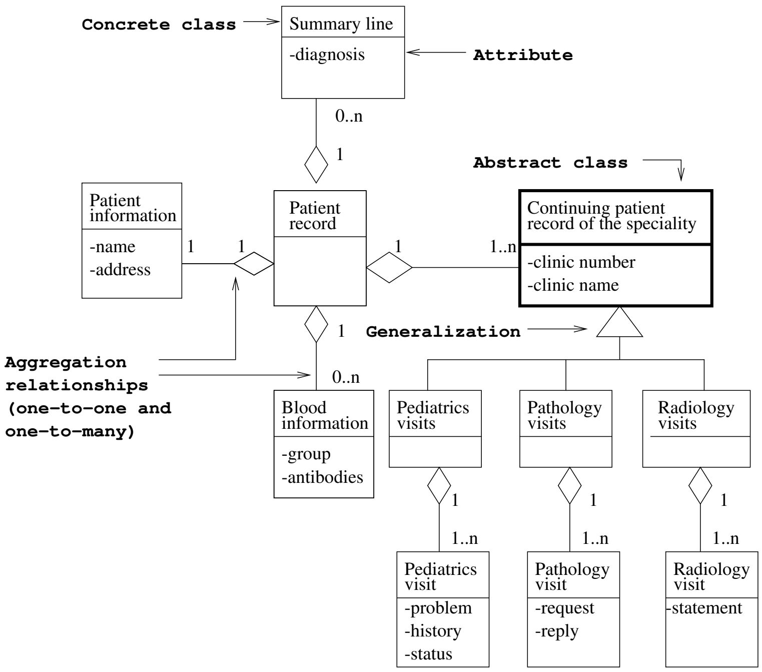


Figure 1: An UML class diagram

The concepts of the real world are *objects*, for example patient record, clinic, blood test, etc. Each object possesses an *identity*, a *state* (consisting of the values of its *attributes*), and a *behavior* (methods). An attribute of an object can be an object. This leads to the *aggregation relationship* between objects. The relationship of the aggregation may be one-to-one, one-to-many and many-to-many (the last one is changed into two one-to-many relationships). The objects are defined in *classes* which form a class hierarchy. The objects sharing a common definition belong to the same class, while objects resembling each other belong to the same class hierarchy. The class hierarchy allows subclasses to inherit properties, i.e. attributes and methods, from superclasses. The superclass is a *generalization* of it's subclasses. *Concrete classes* define objects. Other classes are *abstract classes* and they are used for the definition of the properties that subclasses inherit. The class hierarchy enables the analysis of the similarity of the objects.

The object-oriented approach offers concepts and notations especially for conceptual modeling of the domain. The reusability and similarity of the concepts is carefully considered. Salminen et al. [SKL97] have developed a method for document analysis applying the methodology of Shlaer and Mellor [SM92]. The experiences were encouraging.

In the definition of the structure of the EPR we utilized Unified Modeling Language (UML) [Boo98]. In the UML the analyst defines *use cases*, *class diagrams*, *state diagrams* and *interaction diagrams*. Figure 1 represent a class diagram.

Our target was to analyze carefully where in the organization the information of the EPR is utilized and in which order, what is the importance of the information and when, where and why the information about the patient is gathered to the EPR. Therefore we made a careful analysis with doctors, nurses and secretaries of the Department of Pediatrics of the Kuopio University Hospital. We felt the UML insufficient for the description of the flow of the information in the organization. Hence we utilized the traditional data flows [You89] of the structured analysis in describing how the patient record is used in a hospital clinic and how it is transported from one hospital clinic to another.

## 3 Structured document and SGML

SGML (Standard Generalized Markup Language) [ISO86] is an international standard to define the notation which is used to represent structured documents containing free text. A structured document consists named elements of the content whose identifiers and hierarchical relationships are defined in the DTD. The DTD is common for a set of documents (for example, books, articles, patient records) and all the documents of the type have to follow the DTD. The DTD defines a hierarchical structure, which, for example, can be represented as in Figure 2.

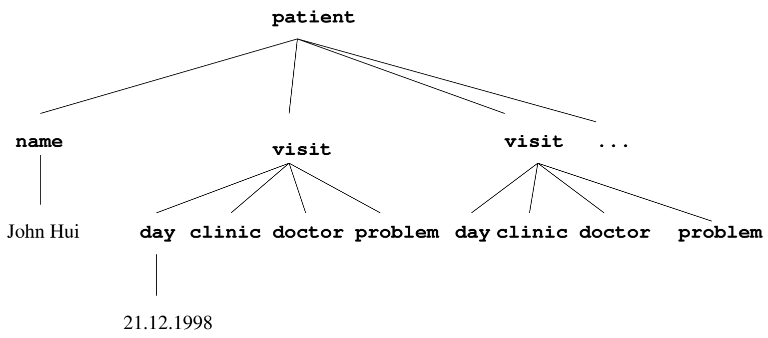


Figure 2: A hierarchical structure of a patient record

The DTD defines for each element *a content model*. It specifies the subelements, their order and existence conditions. The DTD consists of a set of rules. Each rule starts by the characters `<!ELEMENT` and ends with the character >. The rule has three parts: the name of the element, indicators for the existence of the markup, and the definition of the content model. The rules for the patient record in Figure 2 start as follows

```
<!ELEMENT patient        - - (name, visit*)>
<!ELEMENT name           - - (#PCDATA)>
<!ELEMENT visit          - - (day,clinic,doctor?,problem)>
<!ELEMENT date           - - (#PCDATA)>
...
```

This DTD defines that the `patient` element consists of the name of the patient (`name`) and zero or more visits (`visit*`) in the clinic. The patient name is a sequence of characters (`#PCDATA`) defining a *content element*. The `day`, `clinic`, `doctor?` and `problem` elements are contained in the `visit` element, in this order and by the condition, that the `doctor` element is optional. The characters – – after the element identifier state that the element markup has to exist in the document instance.

A document instance according to this DTD contains the markup which uses the start and end tags. For example, the start tag for the element `patient` is `<patient>` and the end tag `< /patient>`. The instance starts as follows:

```
<patient><name>John Hui</name><visit><day>21.12.1998</day><clinic>...
</clinic><doctor>...</doctor><problem>...</problem></visit><visit>...</visit>...
```

If the content model of the element exists in many places of the DTD, it is possible to define a *parameter entity* to shorten the DTD as follows:

```
<!ENTITY % timeandplace "day,clinic">
```

The parameter entity is used like an element in the DTD, but there are no tags for the parameter entity in the document instance.

# 4 From the UML class diagram to the SGML DTD

The potential semantic components are designed in the UML class diagram. In this section the question is, which decisions can be derived from this diagram. We present a set of rules which the analyst can use to generate the DTD tree representation defined in [MEA96]. The DTD can be easily written from the tree representation. The generated rules can be grouped into transformation rules and placement rules. Figure 3 represents a sample class hierarchy and the resulting DTD tree in order to clarify the explanations of the rules.

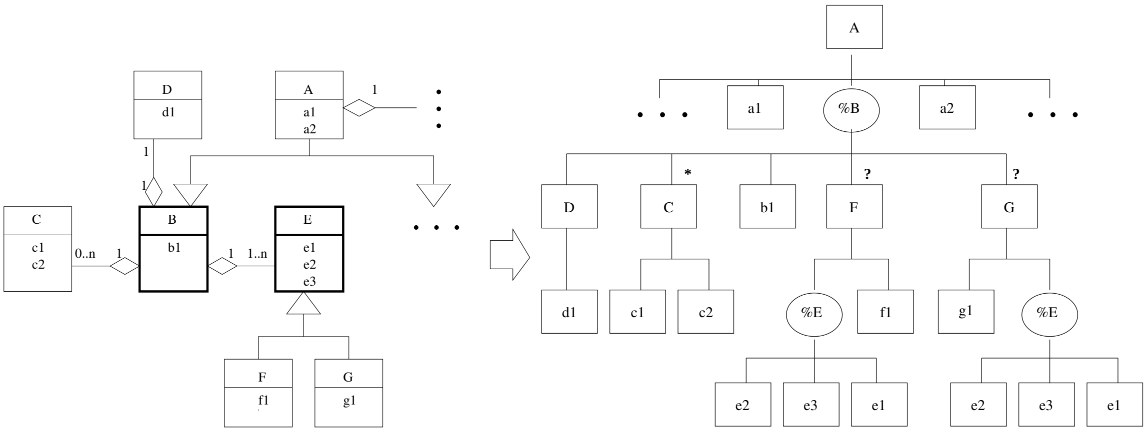


Figure 3: An UML class diagram and the corresponding DTD tree representation

The *transformation rules* are as follows.

**Rule 1:** The concrete class generates an element. (For example, classes `A`, `C`, `D`, `F` and `G`)
**Rule 2:** The abstract class generates a parameter entity. (For example, classes `B` and `E`)
**Rule 3:** The attribute generates a content element (For example, the attributes `a1` and `a2` of `A`)
**Rule 4:** The parameter entity may be divided into many parameter entities.

The *placement rules* are the following.

**Rule 5:** The parameter entity corresponding to the super class in the generalization relationship is placed into the content model of the element or parameter entity corresponding to the subclass. (For example, the super class `E` of the subclasses `F` and `G`)
**Rule 6:** The part in the aggregate relationship is placed as an element or a parameter entity of the content model of the element corresponding to the aggregate. (For example, the aggregation between the class `B` and classes `C`, `D` and `E`)
**Rule 7:** The element corresponding to the attribute is placed into the content model of the element or the parameter entity corresponding to its class. (For example, the attributes `e1`, `e2` and `e3` of the class `E`)

When we applied the rules to the class hierarchy on the left side of Figure 3 the tree diagram of the DTD on the right side of Figure 3 was produced. The class diagram does not, however, define all the information needed to generate the DTD. First, the order of the attributes or parts is not fixed. Orders were defined by discussing with the users and examing the paper patient record. Second, elements corresponding attributes of the abstract class may be grouped in various ways when they are placed into the content models of the subclasses. For example, the attributes of `%E` may be required so that the `e2` forms a group and `e3` and `e1` another group. In this case, the abstract class will generate several parameter entities, one for each group (Rule 4); the names for these parameter entities are formed by indexing (for example, `%E1` and `%E2`). In practise, the grouping can be made in the following way: first, every attribute is placed into the content model of its own parameter entity; second, if later it is found that certain attributes always belong together, a common parameter entity is generated for them.

The relationship between the aggregate and the part defines how many elements are generated into the content model. If the relationship is one-to-one, then only one element for each part is generated. If a part may exist many times (relationship one-to-many) there are two possibilities. If the relationship is `0..n`, then the repeating element is marked by a star character $*$ (for example, the class `C`). If the relationship is `1..n`, then the repeating elements is marked by a plus character $+$. If the repeating part is abstract it generates a parameter entity into the content model of the element of its subclasses. In this case it is replaced by elements or parameter entities corresponding to the subclasses. This situation exists in our example for the class `E` which actually should be marked by a $+$ character. But since the class `E` has two subclasses `F` and `G`, the parameter entity `%E` is replaced by elements `F` and `G` and located into the content model of both `F` and `G`. The character `?` defines that `F` and `G` may exist or may be missing.

# 5 Results and conclusions

We applied the method to define the SGML DTD for the patient record of the Department of Pediatrics in the Kuopio University Hospital.

First, we made a careful analysis with the staff of the outpatient ward of the Department of Pediatrics. We utilized the traditional data flows in describing how the EPR would be used and transfered in the department. We noticed that the doctors and nurses felt the data flow analysis as a natural way to describe their work. They felt it more easy and natural to describe what information they need in taking care of the patients than to design the class diagrams and to classify the objects. They also felt that the diagrams gave them a lot of information about their actions.

Second, we modeled the content of the EPR with UML class diagrams. Here we tried to increase uniformity, reusability and comprehensibility of the system. At the same time we took into the consideration the queries that the doctors and nurses desired to ask from the EPR data.

Third, we applied the rules represented in Section 4 to the UML class diagrams to generate the SGML DTD of the EPR. The rules produced the tree diagram almost automatically without requiring “too much intuitive design”. Instead of including the whole DTD below we represent the DTD generated from the class diagram of Figure 1.

```
<!ELEMENT patient-record      - - (patient-information,summary-line*,blood-information*,
                                   pediatrics-visits?,pathology-visits?,
                                   radiology-visits?)>
<!ELEMENT patient-information - - (name,address)>
<!ELEMENT summary-line        - - (diagnosis)>
<!ELEMENT blood-information   - - (group,antibodies)>
<!ELEMENT pediatrics-visits   - - (problem,history,status,
                                  %continuing-patient-record-of-the-speciality)>
<!ELEMENT pathology-visits    - - (request, reply,
                                  %continuing-patient-record-of-the-speciality)>
<!ELEMENT radiology-visits    - - (statement,
                                  %continuing-patient-record-of-the-speciality)>
<!ELEMENT (name,address,diagnosis, group,antibodies,problem,history,status,request,
```

```
          reply,statement,clinic-number,clinic-name) - - (#PCDATA)>
<!ENTITY %continuing-patient-record-of-the-speciality "clinic-number,clinic-name">
```

The method produces a DTD which is very suitable to be used by the users. Since the classes and attributes in the UML diagram have very content-based names the DTD also uses the specific identifier names. The SGML editor programs use these names to advice the user. However, we are convinced that using another set of rules to the same UML diagrams, also a DTD more flexible for queries can be generated. This will be the next step in our work. To get more material to test the method we are currently analyzing the work flows in the inpatient ward of the Department of Pediatrics. The interesting point will be whether the generated DTD differs from the DTD generated for the outpatient word.

## Acknowledgments

This work was financed by the University of Kuopio. The authors would like to thank the persons of the ELSA (electronic patient record) project of the Kuopio University Hospital and the Department of Pediatrics for their help.

## References

[Ben96] T. Benson. SGML, HTML and EPR, application of the standard generalized markup language (SGML) in electronic patient records. available http://www.mcis.duke.edu/standards/HL7/committees/sgml/references/sgmlepr.htm, copied July 29, 1998, 1996.

[Boo98] G. Booch. *ULM Users Guide.* Addison Wesley, Chicago, Il, 1998.

[BPD96] T. Bürkle, H.-U. Prokosch, and J. Dodeck. Steps towards integration of knowledge based functions into a hospital system. In J. Brender, J.P. Christensen, J.-R. Scherrer, and P. McNair, editors, *Medical Informatics Europe '96*, pages 286–290. IOS Press, 1996.

[Bra94] V. Bradley. Innovative informatics CPR: Computerized patient record. *Journal of Emergency Nursing*, 20:230–232, 1994.

[Car94] P.C. Carpenter. The electronic medical record: perspective from Mayo Clinic. *International Journal of Bio-Medical Computing*, 34:159–171, 1994.

[ISO86] ISO 8879. *Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML)*, 1986.

[MEA96] E. Maler and J. El Andaloussi. *Developing SGML DTDs from Text to Model to Markup.* Prentice Hall, Upper Saddle River, New Jersey, 1996.

[PGR+97] D. Pitty, C. Gordon, P. Reeves, A. Capey, P. Vieyra, and T. Rickards. The place of SGML and HTML in building electronic patient records. In C. Pappas, N. Maglasveras, and J.-R. Scherrer, editors, *Medical Informatics Europe '97*, pages 329–333. IOS Press, 1997.

[PSRL94] D. Paty, D. Studney, K. Redekop, and F. Lublin. MS COSTAR. a computerized patient record adapted for clinical research purposes. *Annals of Neurology*, 36:S134–S135, 1994.

[SKL97] A. Salminen, K. Kauppinen, and M. Lahtovaara. Towards a methodology for document analysis. *Journal of the American Society for Information Science*, 48(7):644–655, 1997.

[SM92] S. Shlaer and S.J. Mellor. *Object Lifestyles – Modeling the World in States.* Yourdon Press, Englewood Cliffs, NJ, 1992.

[TW96] B. Travis and D. Waldt. *The SGML Implementation Guide.* Springer-Verlag, Berlin, 1996.

[YOO+98] H. Yoshihara, K. Obe, K. Ohashi, R. Yamamoto, S. Yamazaki, Y. Hirose, K. Matsui, T. Hishiki, Y. Yamashita, M. Kimura, K. Minagawa, and H. Oyama. Standardization of exchange procedures of clinical information using Medical Markup Language (MML). In B. Cesnik, A.T. McCray, and J.-R. Scherrer, editors, *MEDINFO 98 Proceedings*, page 129, 1998.

[You89] E. Yourdon. *Modern Structured Analysis*. Yourdon Press, Englewood Cliffs, NJ, 1989.