SAGENT: A Novel Technique for Document Modeling for Secure Access and Distribution

S. Hoque, H. Selim, G. Howells, M.C. Fairhurst, and F. Deravi
Department of Electronics, University of Kent, Canterbury, United Kingdom

Abstract

A novel strategy for the representation and manipulation of distributed documents, potentially complex and heterogeneous, is presented in this paper. The document under the proposed model is represented in a hierarchical structure. Associated 'metadata' describes the flexible hierarchy with the scope of dynamically restructuring the tree at runtime. All useful functionals can also be included within the hierarchy to minimize reliance on external programs in manipulating sensitive data. This gives the proposed model two key properties: generality (capable of representing any document format including future innovations) and autonomy (non-reliance on external programs). The model also allows incorporation of additional features for security and access control. Biometric person authentication measures are introduced. A brief example illustrates the key ideas.

1. Introduction

In modern distributed data processing environment, security issues regarding the integrity of documents in the distributed domain are of particular concern. This paper introduces the Securable Autonomous GENeralised documenT model (SAGENT), a document authoring and management system capable of addressing a number of key issues affecting the integrity and security of distributed, particularly multimedia, documents.

The need to assure secure access to modern heterogeneous documents is a challenge in modern commercial and military environments. Existing techniques for achieving this aim suffer from a reliance on external systems for the generation and manipulation of such documents. A truly reliable system for accessing documents should ensure that confidentiality, integrity and authenticity are respected. This requires some means for enforcing access-rights and copyrights. Many of these needs are currently met by using third party solutions providing cryptography, secure infrastructure, digital signatures, etc [1,2,3]. Any compromise of these external systems can seriously jeopardize the integrity of the whole system. A preferable alternative is to encapsulate the document data together with all the necessary associated operations for encryption, security, etc. as a single secure entity. In addition, to enable the document to be accessed over a non-secure network, the incorporation of measures especially tuned to handle high volume and multimedia data is also necessary[4]. These include data compression, authentication, and data viewing and manipulation functions. We introduce the SAGENT model to achieve this aim. The fundamental ideas behind SAGENT were previously reported as Autonomous Document Object (ADO) model[5] and this paper introduces recommendations to significantly enhance the capabilities of ADO.

The internal representation of the data in SAGENT is structured in a virtual hierarchy such that complex data elements are constructed by combining simpler components. This gives the model capability of representing virtually any document format, including those yet to be invented, as well as gives a tool to incorporate additional features to existing document formats.

Although the model described in this paper do not cover all of the design objectives[5], but it shows how many of these key features can be incorporated in the SAGENT model to provide a better tool for storing and distributing sensitive data in the modern arena.

2. The Document Model

The diverse nature of document types as well as varying user requirements necessitates different document formats. As the nature of these requirements changes over time, modification to these formats become essential and at some stage, new document formats are introduced. This evolutionary nature makes it difficult to develop a universal solution to document security. The problem becomes even more severe when heterogeneous data are involved and/or the document content is dynamic in nature. In addition, reliance on third party programs often can be seen as a threat to the security of sensitive document. In light of these, a new document model is proposed here.

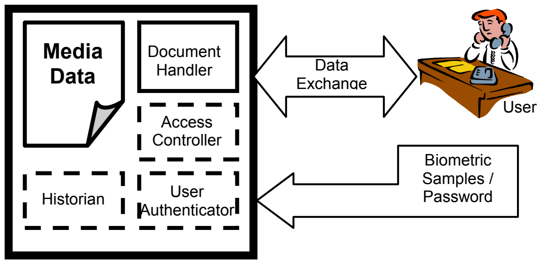


Figure 1: **The SAGENT Framework**

The proposition is modeled with a particular focus towards heterogeneous (e.g., multimedia) documents. The idea is based on object-oriented principle where complex document elements are created by inheriting attributes from relatively simpler classes and thus, creating a hierarchical tree describing the entire document. Multiple inheritance facilitates creation of multimedia components from mono-media classes. The hierarchy created for a particular document is described within the document in the form of its 'metadata'. By manipulating this metadata, the structure of the document can be altered giving extreme flexibility to the document. To incorporate new media types, a user only needs to create new classes by describing them in the metadata already present in the hierarchy. For existing document formats, this model can enhance their potential by incorporating extra features.

Reliance on external interventions to manipulate document contents is reduced in the proposed model by making provision of incorporating the necessary functionals as components of the document hierarchy. This gives the second desirable property *autonomy* to our model. The proposed model is named Securable Autonomous Generalized Document (SAGENT) Model.

A small set of data is initially required from which a new complete document can be created. This is called the Minimal Model and this basic model is used as the starting point for any new document to be created. The following section describes the minimal model components and how they can be exploited to create a real document.

2.1 The Minimal Model

All elements in SAGENT are represented as objects and the class describing these objects is the meta class for the original element. Thus different meta objects describe different classes in the SAGENT hierarchy. The class defining the meta objects is called the meta-class and is also represented as an object in the meta-class itself. This self describing component thus permits run-time modification of the SAGENT structure.

In order to provide a basic framework and rudimentary functionals, some primitive elements must be present in the hierarchy. A user then can append his own elements and functions to SAGENT. This minimal set of components is called the Minimal Model. The Minimal Model consists of three classes: ***Meta***, ***Root*** and ***Method***.

The meta class contains the attributes and methods to describe a class (e.g., defining the super classes and constituent methods). These methods are for modifying the class structure (for example, adding/removing new classes, modifying the attribute list, etc). The class hierarchy can, in principle, be modified at any stage by the end-user by simply redefining the super class attributes in the meta-objects, although in practice some restrictions should be imposed to maintain the document integrity.

Further to representing each class as an object, their constituent methods (or functions) are also represented as objects. These objects are instances of a class named ***Method***. Functions defined for this ***Method*** class return information about the various instances of the ***Method*** class (such as method name, argument types, etc.) and are themselves also represented as instances of this class.

The ***Root*** is the root of the hierarchy tree. The ***Root*** doesn't have a super-class (i.e., its super-class attribute is set to null) and its principal purpose is to bind the elements in the document in a single tree.

The constituents (attributes and methods) of the minimal model classes can be found in [5]. It should be emphasized that although in order to allow maximum generality, the minimal model is the theoretic starting point for any new document, in practice, it is envisaged that templates will be made available for popular standard document forms such as XML, MPEG-7, etc. These templates shall contain additional classes and instances to address the particular requirements of the standard formats being modeled. The model will gradually grow as new instances are added to represent the actual document data.

3. Access Control

When sensitive data are contained in a document, it is essential that measures be adopted to restrict access only to genuine users. The problem becomes more acute when a document needs to be accessed by multiple users each having different access rights. Often is the case where the same user may need to play different roles at different times with different access rights. The SAGENT model can be made capable of handling all these scenarios.

In order to identify a valid user, a new ***User*** class needs to be created whose objects shall represent valid users. Typical attributes to this class are user_name, authenticator_type, user_category, etc. The creation /deletion of objects to this class must be controlled so that fictitious user accounts cannot be created (or valid users given inappropriate access rights) to perform malicious activities. Every user is assigned a category that determines his global access rights. This global access rights can be restricted for certain objects when the objects have their own access control modes. Thus local access-rights always overrides the global right of a user.

When a new document is created, the person who creates this document is called the *Owner*, who is by default authorized to modify the ***User*** class. The document enforces this by verifying the identity (either biometric[6] or otherwise) of the *Owner*. Once the document is created, the *Owner* may include additional users of the document using a method addUser and assign them appropriate access categories to allow them manipulate the document. Subsequently, users will be able to define the structure of the document and include their contents as required. Three access levels have been envisaged. Those users granted an edit category have full access and are able to create objects. They therefore can contribute to the document development. Users with a private category also have full access but only limited to their own objects. Finally, users with the browse category are only able to read/view the document.

In Figure 3, the ***Root*** class has been expanded by incorporating two additional attributes to facilitate access control. The author attribute holds a list of users allowed access to a given object, and the object category holds the nature of access permitted. Setting/resettting values to these attributes is generally restricted to the *owner* only. According to the mandatory security policy assumed in the model, the object category cannot be decided by the *author* but only by the *owner* in order to preserve information integrity. For a discretionary access model intended to preserve information confidentiality, *author*s may be permitted to restrict access to the objects they create. Although this approach may seem too restrictive, the severity of the expected risks in a distributed environment requires mandatory policies, which translate the security responsibilities from the *user* to a special manager (*owner*). *User*s whose identifiers are not included in the user list (author attribute) of the object are not allowed even to browse the object. This user list can be modified by adding or deleting users through a setAuthor method. Finally the author of the object is assigned the most permissive category, namely edit.

4. User Verification

A novel feature of SAGENT model is the integration of biometric identity verification in order to control access to virtual documents in circulation within a distributed user community. The following is a description of how biometric measures is incorporated within a SAGENT framework. The generality of the SAGENT system, however, means that other schemes are also possible to suit particular user requirements. Recall that only the minimal model components are essential to a SAGENT document.

It has already been pointed out that a ***User*** class stores valid users' details within the model. Each user has a name (or id) and other information including that verifies his identity. Biometric templates hold samples of the user's biometric measurements recorded during his enrolment and these templates are compared with live samples to verify the user's genuineness. Templates from multiple biometric sources may be stored for added flexibility[7] and the user may be given option to choose modality depending on his current circumstances (e.g., availability of devices etc.). For high security environments, a multi-modal verification mechanism may be incorporated where a user must match all the diverse where a user must match all the diverse templates stored.

Thus a biometrically secure model should have a ***User*** class with attributes like template, enrolment date and time, verification mode, etc. This list is not exhaustive and the owner of the document needs to decide this based on his particular expectation from the document and its users. The ***User*** class also needs additional methods to support enrolment and verification.

5. History Log Keeping

Recording human-computer interaction has been in practice since the earliest command shells and text editors. With the drastic fall of costs for digital storage, processing and telecommunication, this field is now getting more and more focus and as such, digital records of activity are common at work, marketplace and home[8,9]. By incorporating an event log mechanism, the SAGENT can be made more trusted. This history tool can enforce accountability from the users' side as well as support an error recovery system[10], which is a de facto standard for almost all modern document manipulation products.

Even though history log is not an essential feature of the SAGENT framework, most of the SAGENT documents are likely to be provided with some sort of history databases. Since, this is extremely application dependent, the design and implementation issues are discussed in a general manner in this section. We assume that the objective of the history log is to support UNDO and to establish accountability of the users. At the design stage, the paramount issues are types of history data, selection, capture and aggregation of history information, storage (internal) and weeding/compaction, external display, retrieval and management, etc.

The purpose of the history Log is to record all the events for a document's lifetime (or for such a period as desired by the *owner*). Therefore, it involves the user, the media data, and the method. The user is the person initiating the event, the method defines the nature of the event, and the media data is the object on which the event takes place. History log is therefore a series of records consisting of identifiers for user, media data and methods and may be some additional information depending on the particular application's needs. The success of the history log depends on how judiciously information is chosen to be stored. It has been established that sparse histories are easier to manipulate but are often less useful than extensive histories. Besides data structures should be generic across applications

With a view to support UNDO and establish accountability, we initially choose to record the following attributes in the present history module for the SAGENT. These are User id, Event name, Object id, Time stamp, etc.

History is a non-essential component of SAGENT.

These features can be treated as extensions to SAGENT and should be governed by user-defined methods. But the nature of historian is such that if introduced, its work should remain transparent to the users. So the job of the historian is a system responsibility although its invocation and nature is subject to the owner's discretion. This was achieved by introducing a method named HISTORIAN.

Whenever the user executes a method from the method class, the HISTORIAN method is automatically invoked. In the minimal model, the HISTORIAN consists of a NOP operation thus does nothing (except for wasting a few CPU cycles). When history log is desired, this HISTORIAN method is replaced by a method supplied by the owner/administrator, which updates the history database every time it is called.

6. A Case Study

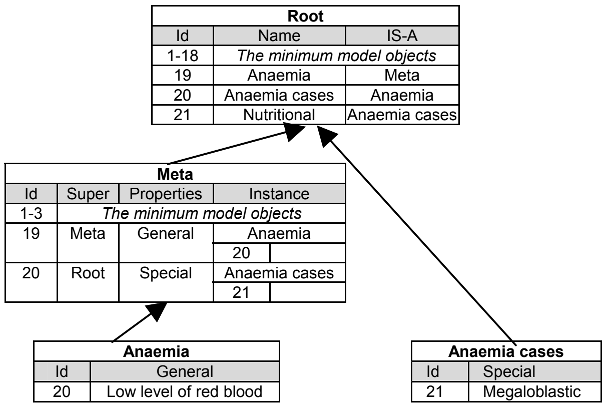
Some of the features of the SAGENT model have been described in the previous sections. As an example of its practical applicability, the experience from implementing HyperMed[11] is presented here.

HyperMed is an educational hypermedia application to train undergraduate medical students. The students treat a number of patients with varying symptoms. Every patient has an associated card, which shows his/her clinical history (e.g., symptoms, previous treatments, test results, other relevant data). For each test series, all data are collectively held in order to study the reaction of the patient to a particular treatment. Students are organized in groups representing medical specialties, and each group takes care of its own patients. Group members can discuss, diagnose, and take decisions about the treatment, as well as modify the patient card. In order to proffer a treatment, students may access the patient cards belonging to other groups, but they cannot participate in their treatment.

The concept of SAGENT, when used in the design of HyperMed, bring about significant improvements where a number of features which could not be represented efficiently in alternative models[12]. Some of these are presented below:

Definition of hyper document structure: The repositories of HyperMed, e.g., the illnesses, etc., can be conceptually described using hierarchical structures with inheritance of characteristics. For example, 'anaemia' may be organized as a tree generalizing various types of the anaemia (e.g., nutritional, haemolytic, aplastic) and each of them divided into different particular cases (e.g., iron deficiency, megaloblastic). The general characteristics, such as low levels of red blood cells, are defined at the top layer and are inherited by all cases of the illness.

In SAGENT implementation, this relationship has been directly modeled by creating a sub class from the ***Meta*** class (see Figure 2) that include the common features and generalize the particular cases. Generalizations make it possible to define elements that are inherited by different classes without duplicating information. Generalizations cannot be modeled in alternative schemes, since the only abstraction mechanism considered is aggregation, which does not allow the heritage of common features.



Root

Id	Name	IS-A
1-18	*The minimum model objects*	
19	Anaemia	Meta
20	Anaemia cases	Anaemia
21	Nutritional	Anaemia cases

Meta

Id	Super	Properties	Instance	
1-3	*The minimum model objects*			
19	Meta	General	Anaemia	
			20	
20	Root	Special	Anaemia cases	
			21	

Anaemia

Id	General
20	Low level of red blood

Anaemia cases

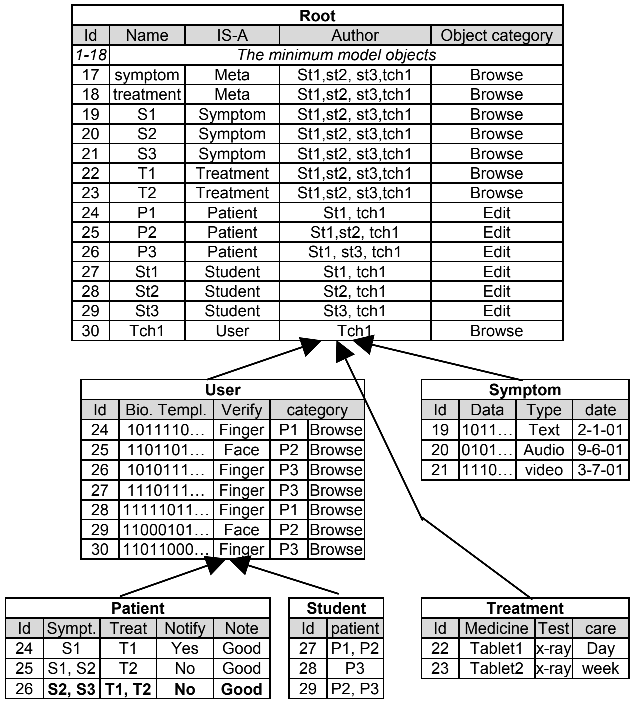
Id	Special
21	Megaloblastic

Figure 2: **Inheritance of common features in SAGENT**

Design of the relationship between the symptoms and the treatment: The ***Patient*** class has the attribute symptoms, which can be portrayed using text, image or video. A particular symptom can be counteracted by means of different treatments and each treatment consists of medicine, test and/or care. When a student decides to treat a patient, he creates a link between one or more symptoms in the patient card and the treatment or treatments that will be applied. Several patients can have the same symptoms but may need different treatments and, therefore this link must not appear for all the patients sharing the symptoms but only in the specific patient being treated.

In HyperMed implementation using SAGENT (Figure 3) each symptom is modeled as a class, to allow the representation of the symptom as text, image or video. Each treatment is modeled as a class which has three attributes: medicine, test and care. The students create links between the ***Symptom*** class and the ***Treatment*** class by sharing object identifiers between the classes. In this way the treatment could be different or the same among the entire patient population even if they have same symptom.

Notification of updates: Every time a student creates a link between symptoms and treatment, the teacher is immediately notified to evaluate the action. To represent this feature using the model a method is tied to the ***Patient*** class called addTreatment. This method updates the treatment attribute in the patient class and also sets the notify attribute. The teacher can check the notify attribute in order to determine if a change has been made by the



Root				
Id	Name	IS-A	Author	Object category
1-18	*The minimum model objects*			
17	symptom	Meta	St1,st2, st3,tch1	Browse
18	treatment	Meta	St1,st2, st3,tch1	Browse
19	S1	Symptom	St1,st2, st3,tch1	Browse
20	S2	Symptom	St1,st2, st3,tch1	Browse
21	S3	Symptom	St1,st2, st3,tch1	Browse
22	T1	Treatment	St1,st2, st3,tch1	Browse
23	T2	Treatment	St1,st2, st3,tch1	Browse
24	P1	Patient	St1, tch1	Edit
25	P2	Patient	St1,st2, tch1	Edit
26	P3	Patient	St1, st3, tch1	Edit
27	St1	Student	St1, tch1	Edit
28	St2	Student	St2, tch1	Edit
29	St3	Student	St3, tch1	Edit
30	Tch1	User	Tch1	Browse

User				
Id	Bio. Templ.	Verify	category	
24	1011110…	Finger	P1	Browse
25	1101101…	Face	P2	Browse
26	1010111…	Finger	P3	Browse
27	1110111…	Finger	P3	Browse
28	11111011…	Finger	P1	Browse
29	11000101…	Face	P2	Browse
30	11011000…	Finger	P3	Browse

Symptom			
Id	Data	Type	date
19	1011…	Text	2-1-01
20	0101…	Audio	9-6-01
21	1110…	video	3-7-01

Patient				
Id	Sympt.	Treat	Notify	Note
24	S1	T1	Yes	Good
25	S1, S2	T2	No	Good
26	**S2, S3**	**T1, T2**	**No**	**Good**

Student	
Id	patient
27	P1, P2
28	P3
29	P2, P3

Treatment			
Id	Medicine	Test	care
22	Tablet1	x-ray	Day
23	Tablet2	x-ray	week

Figure 3: **The HyperMed implementation (partial)**

student. The notify attribute can be reset only by teachers.

Definition of the security policy: The HyperMed users, teachers and students, are organized in medical specialties. For each specialty, the teacher can browse the students' works, whereas students can modify the objects of their patients. Students can also include personal comments analyzing a particular clinical case. In addition, all students can browse (but not modify) the medical databases and the patients' cards belonging to other specialties. The privileges to manipulate the information items depend on the context in which they are included and on the user or user-groups involved. The document model compels the designer to face the requirements regarding information security by defining ***User*** class and access categories during the design stage. Classes and objects are categorized depending on the context in which they are included. For each medical specialty, groups of teachers and students are defined as a sub class from the ***User*** class. Users are assigned a security label, which defines their privileges to manipulate the hyper document elements. For example, in Figure 2, the patient class is assigned an "edit" category in order to guarantee that users with the appropriate privileges will be able to modify them, while the ***Symptom*** class and ***Treatment*** class which can only be read, have a "browse" category. Students have "edit" category for the elements of their own patients while they have "browse" category for the patients of other groups. Therefore, the combination of security categories of classes/objects and privileges of users fulfills the requirements of the application.

No other model allow such a detailed security policy to be specified, and this points to a key advantage of the model and structures proposed here.

7. Conclusion

A generalized autonomous document model is presented. The model is flexible enough to accommodate unanticipated data formats or user requirements. The model is also applicable to other areas where a dynamic system is pre-requisite. The optional features for access control and biometric authentication enhances reliability and trust on the proposed model. The history log mechanism ensures accountability as well as error recovery. The flexibility of the model is illustrated and the advantages are clearly demonstrated.

References

1. A.Menezes, *et al*: Handbook of Applied Cryptography, CRC Press, 1996.
2. J.Dittmann, *et al*: "Using Cryptographic and Watermarking Algorithms", Multimedia, **8**(4):54-65, 2001.
3. F.Andres: "Multimedia and Security", IEEE Multimedia, **8**(4):20-21, 2001.
4. K. Kumar, *et al*: "The HotMedia Architecture: Progressive and Interactive Rich Media for the Internet", IEEE Trans. on Multimedia, **3**(2):253-267, 2001.
5. W. Howells, *et al* : 'The Autonomous Document Object", Proc. of 6th Int. Conf. on Document Analysis and Recognition, pages 977-981, Seattle, USA. 2001.
6. W. Shen, R.Khanna (eds.): "Automated Biometrics", Proceedings of the IEEE, pp.1343-1346, 1997.
7. N. Mavity, *et al*: "Adaptive User Agents for Intelligent Biometric Interfaces", in Proceedings of RASC 2002, pp.72-77, Nottingham, UK. 2002.
8. C. Plaisant, *et al* : "The Design of History Mechanism and Their Use in Collaborative Educational Simulations", in Proc. of the Computer Support for Collaborative Learning, pages 348-359, Palo Alto, CA, 1999.
9. B. Shneiderman, "The Future of History", in 17th Annual Symp. and Open House, Human Computer Interaction Laboratory, Univ. of Maryland, MD. 2000.
10. R. Hazemi, *et al* : "User Requirements for Undo Support in CSCW", in Proc. of the HCI'95, pp.181-193, Huddersfield, UK., 1995.
11. P.Diaz, *et al* : "Design of an Educational Hypermedia Application using Labyrinth Formal Model", in Proc. of EDMEDIA/EDTELECOM'97, pp.269-274, 1997.
12. D. Schwabe *et al*: "Developing hypermedia applications using OOHDM", Proc. of 1st Workshop on Hypermedia Development, PA, USA. June 20-21,1998.