BAX, M. P.; AUGSTEN, Eduardo. A Two-level Modeling and Implementation Approach for CDA Using Plone/Archetypes. In: 2nd International Conference on the CDA, 2004, Acapulco, 2004.

A Two-level Modeling and Implementation Approach for CDA Using Plone/Archetypes

1. Introduction

Health informatics domain on electronic health records (EHRs) is characterized by complexity, large numbers of concepts, and/or a high rate of definitional change. To approach the development of these systems with the traditional Information Tecnology, based on a "single model" approach [3], can be expensive to maintain and usually the system must be replaced after a few years.

With the help of Archetypes schemas [2], we illustrate how to approach the development of information systems in the health field using HL7 CDA document information structures in a two-level methodology. It is suggested that following this methodology, health information systems can be built more efficiently and last longer. The basic idea is to use Archetypes to separate the information level from the knowledge level, thus the knowledge level can be authored directly by domain specialists with little help of IT personnel. Archetypes is able to do all the heavy lifting needed to bootstrap a content type, allowing the developer to focus on other things such as business rules, planning, scaling and designing [2b]. If executed properly, this methodological approach has the potential for creating future-proof systems that could also be easily integrated with others systems in the health field. In fact, with the advent of shared care, there is a need to integrate patient records which spread in disparate health information systems [8].

Today, It can be said that, developing isolated standalone information systems does not make sense. Due to integration needs in the health field, information systems in this area benefit from open web-based standards of communication. Upgrading from HTML to XML, these open standards enable each community to setup its individual tags to describe content, no longer focusing on how content should be presented (as the case with HTML). Tags are now organized in multiple vocabularies, generally representing concepts of a specific domain. These ideas lead to the need of ontology development for organizing concepts

into classes, proprieties, and relationships. Ontology's can allow semantic integration and foster interoperability of systems in a specific domain.

Many health field standard developers around the world have already initiated efforts to organize concepts in the various knowledge health sub-domains. As an example, in the last years HL7 evolved from a proprietary syntax, clinical and administrative content representation and messaging to an open syntax for messaging, documents, context management, and decision support for the EHR.

The paper is organized as follows: in Section 1, CDA document architecture requirements are described revealing its interdependence with a Content Management System (CMS). Section 2, follows with the progression of the CMS platform's use of Archetypes and Extensible Markup Language (XML) in separating content, structure, and presentation; facilitating the implementation of CDA document templates. Finally, Section 3 illustrates the implementation of CDA specification requirements in the open source CMS, Plone using Archetypes - facilitating CDA design goals and principles as described in [1].

2. CDA Documents architecture

The HL7 Clinical Document Architecture (CDA) standard is a extensible and hierarchical document markup standard that specifies the structure and semantics of clinical documents for the purpose of exchange (Dolin R, Alschuler L. et al. 2001).

A document can be defined as a piece of text or information that would usually be authenticated by a signature eg. a progress note, a pathology request, a radiology report, or an account. A CDA document may contain text, images and multimedia, coded data. The CDA document can be stored either permanently or temporarily as a document in a computer system; and transmitted as the content of a message using email, HL7 or any other messaging system.

The standard allow machine processing of clinical documents in a way which makes the documents both human readable and machine process-able, and guarantees preservation of the content by using the eXtensible Markup Language (XML) standard. It is a useful and intuitive approach to management of documents which make up a large part of the clinical information processing arena.

The CDA does not model clinical content, but does standardize the markup of the structure and semantics needed for exchange of clinical documents. CDA documents are encoded in (XML) and derive their meaning from the HL7 Reference Information Model (RIM), using the HL7 Version 3 Data Types. The

complete CDA will include a hierarchical set of document specifications referred to as”architecture".

CDA is being used in electronic health records projects to provide a standard format for entry, retrieval and storage of health information. Many examples of CDA implementation exist, to list some:

- PICNIC (Ireland, Denmark, Crete)
- Regional Health Information System, Satakunta Macropilot (Finland)
- SCIPHOX (pronounced Sky Fox), Germany
- Electronic Health Record at the University of Munster
- MERIT-9, Japan
- and many others.

CDA aims to give priority to documents generated by clinicians in order to:

- Standardise the format of the many thousands of types of clinical documents
- To support exchange of clinical information for human readability, and information processing;
- To promote longevity of information by separating the data from the systems that store it (to avoid obsolescence as occurs with technological processes and by being computer platform independent;
- Allow appropriate local adaptation of the standard to meet national or specific user requirements.

A CDA document consists of the following structure:

- Header -- this contains the key descriptive information about the document (metadata) such as who wrote it, who it is intended for, and type of document.
- Body -- this contains the text of the document which may be structured at least under key headings or sections. It is possible for the text to contain coded values. It is also possible to have not text information in the body such as an image of an x-ray (using the DICOM standard representation).

CDA has been developed in 3 stages: Level 1 through Level 3. Level one has a structured header and structured body of message with limited coding capacity for content. Levels 2 and 3 impose more structure to allow the representation of "context" or constrained fields and more coded data. The standard has been published for level 1 and level 2, level 3 is currently in draft stages.

CDA level 2 and 3 documents (along with the standard electronic health record architecture) require the use of templates and archetypes which define the key

information and context of complex health concepts such as family history or blood pressure.

In the next Section it is described how one could model CDA documents in Archetype, enabling one to approach the specification and implementation of Electronic Medical Record systems following a “two-level” development methodology.

3. CDA Documents and Content Management Systems Archetypes

There is great potential in implementing Document Standards like HL7's CDA within an open source Content Management System (CMS) like Plone [5], which include: reduction in paperwork, predictability of data content, flexibility of implementing a simple approach to imaged data and text to the more sophisticated approach using structured data.

In the simplest of scenarios, health providers should be reimbursed quicker and more efficiently by sending attachments electronically and payers should save money currently allocated to the paper process e.g. mail room, and routing paper documentation. Not to mention further savings with reduction/ elimination of paper and postage.

This section describes how an open source CMS platform using XML schemas makes it easier to implement CDA document templates by separating content, structure and presentation. It shows that CDA specification can be implemented in the open source software Plone and using Archetypes, in a way that efficiently fit most of the CDA requirements.

Archetypes is Plone's next-generation content type generation framework. Most content management projects involve introducing new types of content, which in the non-trivial case requires an informed understanding of how Zope and the CMF [6] work. Archetypes provides a simple, extensible framework that can ease both the development and maintenance costs of CMF content types while reducing the learning curve.

Archetypes simplifies the creation of new content types under Zope/Plone platform. It works by proposing a Schema driven automatic form generation for content editing and visualization. It allows a simplified way to integrate with rich content types, and provides a lower entry bar to the requirement specifications of highly complex and dynamic knowledge fields as is the case in health domain. Besides, it also lowers the entry bar to the complexity Zope places on the creation of new content objects.

A content type might be a document, event, image, or any other content object that is made available for users/content creators to add to a Zope content management site. Content management projects involve introducing new types of content, which can be both time consuming and complicated. Archetypes provide a simple, extensible framework that can ease both the development and maintenance costs of content types while reducing the learning curve for the simpler cases.

Using Archetypes to encode CDA documents has the following advantages:

- Automatically generates forms based on RIM [8] for editing and presentation of content types
- Provides a library of field types, form widgets, and field validators
- Integrates custom fields, widgets, and validators
- Automates transformations of rich content
- Easy installation of the generated content types in Plone platform, providing storage transparency
- References implementing relationship between content types
- Basic security bootstrap

The creation of a new type using Archetypes involves a main text file (written in python language [7]) that defines the fields and other objects within the type, their properties, and their behavior. This file is the schema type specification. Archetypes uses this information, the schema, to auto generate, on demand, all forms and pages needed to add, edit, and view content (types data). Once the schema is ready it needs to be installed in the platform as a new Plone product.

3.1 Specifying and Building a simple document: "Progress Note"

As seen above, Archetypes is able to do all the heavy lifting needed to bootstrap a content type in the Plone framework, allowing the developer to focus on other things such as business rules, planning, scaling and designing.

For building a simple schema declaring the document type it is necessary to know in advance which fields are needed and what data type they will contain. As an example to illustrate the general idea lets build a simple Progress Note class of documents. See the ProgressNote class schema presented in Fig 1.

```
1 from Products.Archetypes.public import *
2
3 class ProgressNote(BaseContent):
4   """ A Simple Progress Note """
5   schema = BaseSchema + Schema(
6            StringField('body',
7            widget=TextAreaWidget(label='Body Content')
8          ))
```

Fig. 1 – Progress Note Document Class.

This is a very simple example, but it already shows the power of Archetypes and how easy it is to build new content types. Let's look at it in detail and see what's going on.

In line 1, we import all the fields, widgets and classes made publicly available by Archetypes. In line 3 you see the actual class declaration. It says that ProgressNote class will inherit from BaseContent, which is a base class provided by Archetypes that handles validation and access of fields defined in a schema; it also handles lots of other things such as initialization of the object and indexing. BaseContent is for normal, non-folderish content type; for folderish content type base class, BaseFolder should be used. Line 4 includes the docstring, which should always be provided, as it is considered good style. In line 5 the schema definition is started using a variable called BaseSchema. BaseSchema is the schema for BaseContent, which has some common fields such as title, id and the Dublin Core fields.

Then, to these fields, one add its own custom schema, which in this case is composed of a string field named "body", for which we will use a TextAreaWidget with the label "Body Content".

In Archetype schema language there are plenty of other fields and widgets available, such as BooleanField/BooleanWidget, IntegerField/IntegerWidget, SelectionWidget, MultiSelection widget and so on. It is also possible to define other specific fields and widgets to handle special cases.

Suppose now, it becomes important to make the content of a "Progress Note" instance searchable, and make its content be required when editing, so there won't be empty Progress Note body in the system? It would require a simple change in the schema definition to the one presented in Fig 2.

```
schema = BaseSchema + Schema(
        StringField('body',
        searchable=1,
        required=1,
        widget=TextAreaWidget(label='Body Content'),
        ))
```

Fig 2. Adding constraints to the model.

As specified in CDA documents, a Progress Note would have many more properties and constraints and a complete specification for all the CDA

documents architecture would require much more space than we have here. However, what it important here is to emphasize the "two-level" modeling approach supported by Archetypes: business managers in the knowledge field are allowed, with little help from TI personal, to understand and even generate and maintain the needed document types in a declarative and simple format. All the mechanics necessary for persistence/storing, editing, visualizing and managing the instance data is done behind the scenes by Archetypes and Plone. In this manner, if the specifications change, all that is needed is to declare the new additions or constraints to the schema files and click a button to rebuild the entire low-level data model.

4. References

[1] Liora Alschuler et al. Version 3 Standard: Clinical Document Architecture Framework. Health Level Seven, 2000.

[2] Sidnei da Silva. Introduction and Installation of Archetypes. Available at http://plone.org/documentation/archetypes/ArchetypesDeveloperGuide/index_html

[2b] Sidnei da Silva. Archetypes: An Introduction (Part I). Zopemag. October 6, 2003. Available at: http://www.zopemag.com/Issue006/Section_Articles/article_IntroToArchteypes.html

[3] Thomas Beale. Archetypes: Constraint-based Domain Models for Future-proof Information Systems. OOPSLA 2002, workshop on behavioral semantics, 2002.

[4] Steve Spickemire et al. Zope: Web Application Development and Content Management. New Riders Publishing, January 2002.

[5] Reuven M. Lerner. At the forge: introducing Plone. Linux Journal, Volume 2003 Issue 109, May 2003.

[6] Reuven M. Lerner. At the forge: CMF types. Linux Journal, Volume 2003 Issue 112, August 2003.

[7] Amy Hendrix. Snake charmer. NetWorker, Volume 8 Issue 1. March 2004.

[8] Zhang Xiaoou, Pung Hung Keng. An XML-based virtual patient records system for healthcare enterprises. Enterprise information systems IV, January 2003.