Ontology-based editor for metadata documents

Silvia Duca, Fabio Vitali

Department of Computer Science, University of Bologna, Via Mura Anteo Zamboni 7, 40127 Bologna, Italy
{ducas, fabio}@cs.unibo.it

Abstract. Editing metadata for documents in a large collection is a long and menial task. Many metadata schemas exist, among which, of course, Dublin Core, and they are usually only partially compatible to each other. Adopting any of these schemas often requires adopting specific tools for the insertion and editing of metadata that are rigid, do not allow for customization and specialization for the organization needs. In this paper we present g.a.f.f.e. (Generator of Automatic Forms - Final Edition), a metadata editor that allows for any each metadata schema considered, to be specified and customized. The metadata input and for the actual interface can to be customized and modified parametrically through the aid of domain and interface ontologies expressed in OWL. g.a.f.f.e. behaves as an independent desktop application as well as an add-in to MS Office applications, allowing for metadata to be stored either as independent OWL files or as properties within the documents themselves.

Keywords: "metadata; ontology; editor; intelligent agents; semantic web; dynamic editor; Dublin Core; DC; documents ontology; g.a.f.f.e."

1 Introduction

Specifying metadata about individual items of a large collection of documents is a long and tedious process most often done manually. Automatic extraction tools can only help so much, and most of the external qualities of a document can be determined through the active intelligence of a human reader.

Metadata are not usually confined to content-descriptive information (such as the topics discussed within the resource, i.e., what Dublin Core [1] would call the dc:subject field), but may span across multiple context, including bibliographic information (such as author, date, location, data format, etc.) as well as lifecycle (such as relationships to other documents, etc.). For a document collection to be fruitfully used, it is appropriate that not just any metadata value is associated to documents, but metadata according to a precise schema, in order to homogenize the descriptions of the documents.

Unfortunately, plenty of metadata schemas exist to describe documents and document collections, with many overlapping concepts and many differences. For instance, we may list data-format agnostic schemas such as Marc21 [2], PREMIS [3], Dublin Core [1] or FOAF [6], TEI [4] or FRBR [5], and more others as well as data-format specific schemas such as OpenDocument (ODF [7]), MS Office (Open XML, [8]), PDF [9] and more others, as well as the myriads of non-official and non-standardized metadata sets contained in the HTML META tags of all the Web documents in the world.

Document collections themselves can sometimes be a streamlined list of well-differentiated documents with clear source and status, but more often they will be a complex mess of extremely undifferentiated documents, sometimes even including many identical copies as well as undated versions and variants of the same document, of which it is important and useful to identify the history and lifecycle.

Metadata are either specified by a specialized human editor that, through access to actual content and to externally available information (and sometimes even through guesswork), manually identifies the relevant metadata values and adds them either to the documents themselves (if the data format allows it) or to a metadata database. More rarely, they are added by the content authors themselves if the collecting organization has the power to impose such task on them. In the first case, personal dedication to the need and purpose of metadata can be expected, but errors in interpretation or missing information can be had frequently, while in the second case the interpretation is surely authoritative, but the devotion to precision and completeness of metadata may vary considerably.

In both contexts of use, therefore, adequate and user-friendly tools to support the specification of metadata associated to documents belonging to a large collection are crucial to guarantee a satisfying quality in metadata themselves.

Alas, metadata editors do not abound, and even the few existing ones are specific to an individual schema with the exclusion of all others. Furthermore, even when an organization adopts a standard metadata schema, the temptation is strong to customize and extend it with organization-specific additional fields, which prevents the use of schema-specific tools. Furthermore, contextual schema-specific tools may not be able to exploit a baseline of default values, restricted value lists, or even invariable values that are relevant or appropriate to the specific organization or collection the documents belong to, thereby simply providing just an endless list of empty form fields without defaults, constraints, or automatically determined suggestions.

In this paper we propose g.a.f.f.e. (Generator of Automatic Forms - Final Edition), a fully configurable metadata editing tool, that can be used either as a stand-alone application or as an add-on to MS Office. The g.a.f.f.e. editor is based on Semantic Web technologies in order to improve the quality of the metadata inserted as well as the ease of specification of the metadata values. Through g.a.f.f.e. functionalities metadata authors (both as specialized professionals and as temporary role of the document authors) can specify metadata for documents according to any standard or customized schema using a form that can be fully customized to provide defaults, constraints, controlled vocabularies and value suggestions as appropriate. The application connects the metadata schema with the library of form elements (both specified as OWL[20] ontologies) through an instance ontology that describes abstractly the connection between the domain metadata schema and the form elements that display or edit the corresponding values. The instance ontology can be fully specified to allow for organization-wide defaults, controlled vocabularies, suggestions and constraints to be used. This is meant to ease considerably the effort of specifying metadata on the documents of the collection.

2 Related work

Metadata editors have existed since metadata in electronic form started being considered necessary for cataloguing and describing documents.

Some of these editors are specific to metadata schemas, such as DC-dot [10], developed by the University of Bath, which retrieves a Web page and automatically proposes metadata according to the Dublin Core standard. The generated metadata can then be edited using the form provided, with optional, context-sensitive help available while editing. Form fields for the metadata are only text areas and no domain-specific constraints, defaults or controlled vocabulary can be specified.
Similarly, Metamaker by FAO is a tool designed to create Dublin Core-compliant metadata. The tool allows the user to create metadata from scratch using a simple web form and save it in different formats (HTML, XHTML, XML, RDF or AGRIS AP), but employs controlled vocabularies of frequently used terms (taken from the AGROVOC and AGRIS thesauri) to allow for standardized description of the subject.

Relevant editors oriented towards other metadata schemas can be mentioned as well, including, RELOAD[11] (Reusable eLearning Object Authoring & Delivery), which is an editor for LOM metadata, or Visual Marc Editor [12] for the SLIM++ package, which provides a graphical interface to edit Marc21 metadata,

A few schema-agnostic editors exist, that accept some or all metadata schemas as input for determining the fields present in the form. For instance, TKME [13] is an application that allows creating and modifying metadata without a schema but with a hierarchical tree structure. The TKME editor does not give the possibility to constrain the metadata schema and consequently it is not possible to customize the interface of the editor.

Among ontology-based editors, we can include OntoEdit [19], that enables inspecting, browsing, codifying and modifying ontologies, and OXML [21], that allows the editing of a hierarchy of concepts both abstract and concrete. Ontologies are specified in the ontological engineering phase guiding the whole engineering process of ontology development. A remarkable editor, and perhaps the closest we could find to g.a.f.f.e., is Metasaur [14].
Metasaur provides a general visualisation tool for the ontology of a domain. Besides building lightweight ontologies for existing metadata schemas, the user can enhance the ontology with additional constraints and controlled vocabularies. In fact, it is possible to add metadata choosing among the classes and the properties in the loaded ontology. On the other hand, Metasaur generates the user interface in a completely automatic way, based only on the provided ontology and does not allow the user to model or customize the form itself nor the type, order and position of the widgets used to enter the metadata.

3 Users roles and tasks

Adding metadata to the items of a collection of documents is a complex task that requires the actions of people performing different roles. As such, the design of the g.a.f.f.e. editor did not refer to an abstract rules covering all these different roles, but assumed the existence of four well-described and different classes of users, each covering a very specific role in relation with the metadata of the collection. Although the actual editor is designed for only one of these four users:

- The *domain expert* (Alice in Fig. [1]) selects the metadata schema and provides the organizational customization of the fields and values. Alice carries out her specific roles by providing the document ontology described in section 4.1 and by describing the type of output desired (be it in the form of properties within the documents, individual metadata files or a single metadata database for the whole document collection). Alice is thought to be less interested in how the documents of the collection get their metadata and rather much more interested in the fact that they actually and correctly are getting metadata.
- The *form programmer* (Bob in Fig. [1]) generates the interface exposed to the users of the g.a.f.f.e., and can organize the controls and form panes as desired. Bob does so by generating the instance ontology (as described in section 4.3) and is interested in making the task of metadata specification and editing as easy and as straightforward as possible for the metadata provider. The interface can be fine-tuned with scripts for providing suggestions for values, automatic values and values validation.
- The *metadata provider* is the actual user of the g.a.f.f.e. (Catherine in Fig. [1]), activating it within MS Office or as a stand-alone application. Catherine has no direct involvement with the ontologies behind the scene, but is the one responsible for actually providing metadata values for manual fields, of checking the suggested values for semiautomatic fields, and of eventually making sure the metadata record ends up in its expected destination. Catherine can also provide useful suggestions to Bob for fine tuning the actual interface of the application. This is done by modifying the instance ontology and never by changing actual code of the g.a.f.f.e..
- The *final user* (Dylan in Fig. [1]), can easily find documents according to his needs by performing searches on the metadata stored with the documents. Although Dylan is never involved in the design and use of the g.a.f.f.e. tool, he is an important stakeholder in the design of the application, as he is the end user whose needs need to be satisfied eventually.

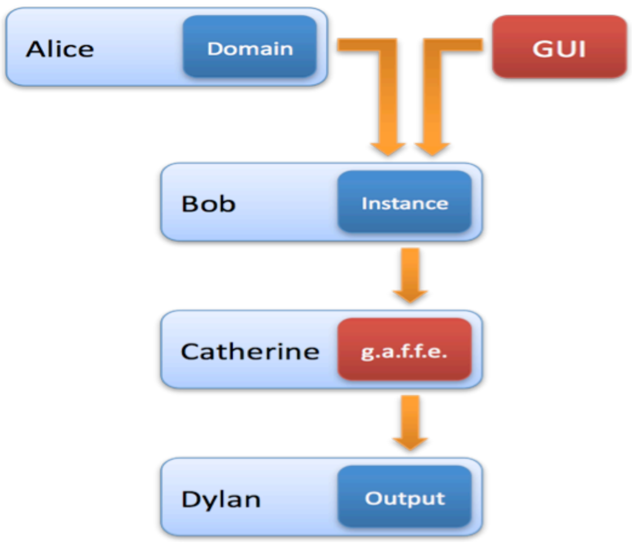


Fig. 1. Users' role in g.a.f.f.e.

By clearly separating the roles of the metadata activity and clearly assigning them to different personas, we can more clearly focus on what are the requirements for our main users (Alice, Bob and Catherine) in providing the best results for Dylan.

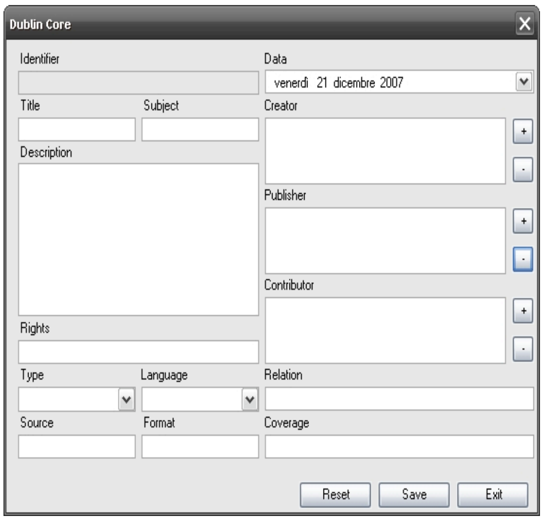
4 Ontologies used for g.a.f.f.e.

The g.a.f.f.e. editor generates a metadata editing application through the contribution of three different resources that are used for the full customization of the application. These resources are expressed as OWL ontologies for better understanding and control of the relations and properties of the involved concepts.

4.1 The Document Ontology

The Document Ontology summarizes abstractly the metadata schema chosen for the description of the documents of the document class. It is an OWL rendering of a standard schema (such as Dublin Core) and allows the specification of further extensions (for organization-wide new fields) and restrictions (for the specification of controlled vocabularies and range constraints).

Concretely, the Document Ontology is in fact composed of two different resources, the *standard, i.e.,* the representation of the schema as it is publicly known, and the *local modifications*, i.e., the extensions and restrictions proposed and managed within a specific organization or document collection, which are expected to evolve and change at a very different speed and through the action of different agents than the standard ontology.



venerdi 21 dicembre 2007

Fig. 2. The Dublin Core Ontology as shown in the g.a.f.f.e. application

Currently we are using two different standard Document Ontologies (Fig. [2]): on the one hand, a plain and non-customized OWL representation of the Dublin Core model, and on the other a fully customized rendering of the United Nations OCHA-FIS model [18], which has a structure that is intermediate in complexity between Dublin Core and FRBR [8], for instance separating metadata about the physical files and about the abstract idea of the document.

4.2 The GUI Ontology

The GUI ontology is a collection of abstract concepts describing the forms and its elements. Through the GUI Ontology the user (Bob in the schema described in section 3) can describe and personalize the form without accessing the code of g.a.f.f.e., but simply instantiating the GUI base classes in the Instance Ontology.

The ontology is organized around three main classes:

- *Form*: it represents the actual editing form as graphic windows in g.a.f.f.e.. It describes geometrical properties such as width and height, left and top, plus a text property for the title. Forms contain controls.
- *Control*: it describes abstractly all the components of a form. The control class has several sub-classes that describe each widget used as controls in the form. Among the widgets we find basic form elements such as text boxes, radio buttons, check boxes, as well as more sophisticated elements such as tree viewers, date and time pickers, and the *ListAddRemove*, which is a complex widget composed of a list and two buttons (to add and remove elements) specially designed for editing ontologies (in the Dublin Core form example

shown in Fig. [3], selectors control the editing of the dc:creator, dc:publisher and dc:contributor values).

- *Config*: it is the class that defines the main window and represents the basic configuration set of the g.af.f.e. application. It also contains a reference to the pathname of the instance ontologies used by the application, which allows the final user to be provided with a predefined list of ready-to-use ontologies in the configuration menu.

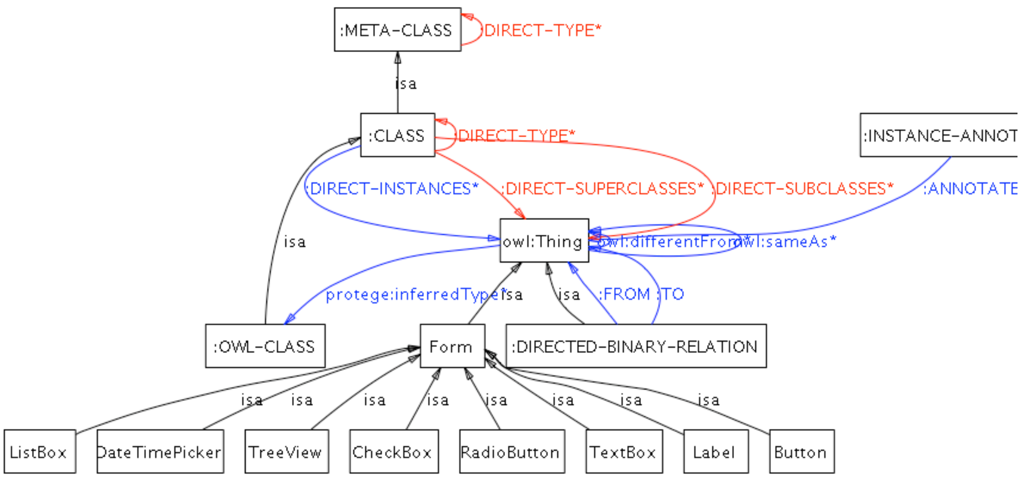


Fig. 3. The GUI ontology regarding controls used by g.a.f.f.e.

4.3 The Instance Ontology

An instance ontology is composed of instances of forms containing controls associated to properties of the classes of the document ontology. One of the forms must be designated as the main form, and it is the first one being shown. A simple approach would be to designate as main form the one associated to the main class in the ontology (most probably a class describing the document), but things could become quite more complex than this. Local needs may be more complex than that.

All properties corresponding to literal values (i.e., plain values such as Document Title or Document Creation Date) are best represented as plain controls such as TextBoxes or DateTimePickers when their values refer to uncontrolled vocabularies, or such as ListBoxes, CheckBoxes, or RadioButtons when controlled vocabularies are appropriate.

On the other hand, properties representing relations to other classes would require a specialized control such as ListAddRemove that has a complex behaviour: it is composed of a TreeView for the display of labels of the associated instances, coupled with two "+" and "-" buttons for adding or removing instances.

By clicking on the (+) button a selector of existing instances of the related class is shown. The metadata provider can select an existing or click on a New button that opens a new input form for the creation of a new instance of the class according to the corresponding instance of the GUI ontology.

In the example in Fig [2], the dc:creator property is associated to a ListAddRemove of instances of the Agent class (e.g., people, that could be described through a FOAF class). In the TreeView the selected Agents are shown (or, rather, the labels associated to each selected instance of the Agent class). By clicking on the (+) button, a selector is shown listing all existing instances of the Agent class. Catherine can choose here the most appropriate value, or decide that none exist. By clicking on the New button, the an empty form with the controls associated to the Agent class is shown on top of the previous forms, allowing Catherine to create a new instance of such class. When approving such dialogue window, the newly inserted value is shown in the selector and selected already.

5. The g.a.f.f.e. application

As mentioned, the g.a.f.f.e. editor can be used as either an MS Office add-in or an independent application.
In the main screen of all MS Office 2007 applications, the Home tab shows the g.a.f.f.e. add-in interface composed of three buttons:

- "Tag it!", that initializes the application and shows the form window as specified in the Instance ontology.
- "Settings", that opens a menu of recently used Instance Ontologies and gives the possibility to add a new one.
- "About", that shows the information about the g.a.f.f.e. editor.

After selecting the appropriate Instance ontology in the Settings menu, the metadata provider can edit the metadata associated to the document by clicking on the "Tag it!" button.

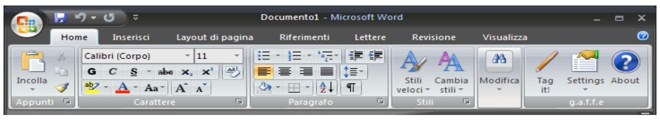


Fig. 4. The add-in for g.a.f.f.e. for MS Word 2007

Users needing to associate metadata to documents that are not MS Office files may use g.a.f.f.e. as an independent application available in the main bar of Windows.

The use is similar to the previously described Office add-in: by clicking on the g.a.f.f.e. icon with the right mouse button, the application shows a menu with the options "Tag it!", "Settings" and "About". Clicking with the left mouse button will immediately activate the "Tag it!" form window. The functionality and the use of the form are the same of the g.a.f.f.e. Office Add-In.

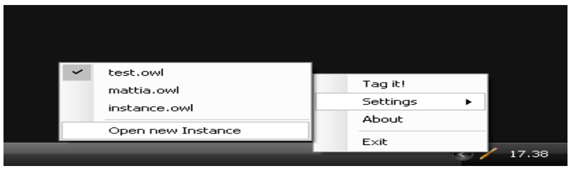


Fig. 5. The stand-alone g.a.f.f.e. editor

5.1 The g.a.f.f.e. form windows

Regardless of whether it is used as an MS Office add-in or as an independent application, the form window is the rendering of an instance of the form class as specified within the Instance Ontology set at startup. The GUI Ontology relies on at least one form pane (the main form) to be present in the Instance Ontology with standard dismissal buttons, although as many forms are probably needed as there are editable classes in the Document Ontology.

The g.a.f.f.e., both in its embedded as well as autonomous manifestation, provides many usability functionalities that are meant to improve the ease of use of the application for the metadata provider. For instance, the most important are:

- All text fields remember the last 10 specified values, as well as any number of default values as detailed in the Instance Ontology. This allows Catherine to easily retrieve and use recently used as well as frequently used values for metadata fields.
- Values from controlled vocabularies can be shown either as check boxes (for non-exclusive values), as radio buttons, or as pop-up menus. Default values are pre-selected.
- As mentioned, references to other classes can be displayed as lists with "+" and "-" buttons to add or delete instances, and the whole process of managing the addition and deletion of instances of related class is automatic managed by the ListAddRemove control. It is possible to associate default scripts (for values that are suggested as defaults when displaying the form, e.g., the current date) as well as validation scripts (for the verification of the inserted values as they are being inserted by the metadata provider), that return warnings and error messages as appropriate.

5.2 Technical details

The core of the g.a.f.f.e. application is a .dll library that is identical in both manifestations of the application. The dll makes use of SemWeb [15], a .NET library developed for C# that provides classes that read, write and manage RDF and OWL documents. VISTO 2005 SE Tools for Office Second Edition [16] is a Visual Studio component that allows C# and VB.NET programmers to develop applications for Office.

The Second Edition provides flexible modules that allow personalized interfaces for MS Office add-ins. Protégé [17] is used for the creation of the relevant ontologies. Protégé is a Java open-source software developed by Stanford Center for Biomedical Informatics Research.

6 Conclusions

In this paper we have presented the g.a.f.f.e. editor, a completely modular metadata editor that can associate metadata values to documents according to any metadata schema.

Besides being independent from the metadata schemas themselves (as long as they are expressed in OWL) the editor also allows complete freedom in customizing them (by constraining and expanding the base schema) and in designing the actual interface, allowing for custom forms to be built and used according to the requirements of the organization collecting and describing the documents.

The editor has been tested with two rather different schemas in terms of complexity and variety, Dublin Core and OCHA-FIS, thereby demonstrating its flexibility and expressive power.

At the moment, though, all relevant document and instance ontologies need to be built directly in OWL using tools such as Protégé, which somewhat reduces the ease of installation and configuration of the application. In the near future we plan to provide a meta-schema approach that would allow to generate both the document ontology (and any customization thereof) and the instance ontology using the g.a.f.f.e. editor itself. Interested readers can find the gaffe application and some documentation at the URL: http://tesi.fabio.web.cs.unibo.it/Main/Gaffe"

References

[1] DMCI, Dublin Core Metadata Initiative, 05/11/2007, http://www.dublincore.org/

[2] Library of Congress, Network Development and MARC Standards Office, MARC 21 Format for Bibliographic Data, http://www.loc.gov/marc/bibliographic/nlr/nlr.html

[3] Premis Working Group, DATA DICTIONARY FOR PRESERVATION METADATA, http://www.oclc.org/research/projects/pmwg/premis-final.pdf

[4] The TEI Consortium, TEI P5 Guideliness for Electronic Text Encoding and Interchange, (2005), http://www.tei-c.org/release/doc/tei-p5-doc/html/

[5] International Federation of Library Associations and Institutions, FUNCTIONAL REQUIREMENTS FOR BIBLIOGRAPHIC RECORDS, Final Report - http://www.ifla.org/VII/s13/frbr/frbr.pdf

[6] FOAF Project, Friend Of A Friend, Dan Brickley and Libby Miller, http://www.foaf-project.org/

[7] OASIS, Open Document Format for Office Applications (OpenDocument), Committeee Specification http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

[8] Ecma International TC45 – OFFICE OPEN XML, (2006), http://www.ecma-international.org/news/TC45_current_work/TC45_available_docs.htm

[9] Adobe System, PDF reference, version 1.7 http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference.pdf

[10] Andy P.: Dc-dot, UKOLN University of Bath, http://www.ukoln.ac.uk/metadata/dcdot/

[11] RELOAD, RELOAD Editor, 20/06/2006, http://www.reload.ac.uk/

[12] Algorhythms, Visual Marc Editor for the System for Library Information Management (SLIM++), (2001), http://www.slimpp.com/SlimPPsite/vismarc/

[13] Peter, S.: Tk Metadata Editor, TKME Documentation, 19/10/2007, http://geology.usgs.gov/tools/metadata/tools/doc/tkme.html

[14] Judy, K., Andrew, L.: An Ontologically Enhanced Metadata Editor, School of Information Technologies, University of Sydney, NSW 2006, Australia (2006)

[15] Joshua, T.: Semantic Web/RDF Library for C#/.NET, 25/10/2007, http://razor.occams.info/code/semweb/

[16] Microsoft Corporation, Visual Studio 2005 Tools for Office Second Edition, (2007), http://msdn2.microsoft.com/en-us/office/aa905543.aspx

[17] Stanford Center for Biomedical Informatics Research, What is protégé?, (2007), http://protege.stanford.edu/overview/index.html

[18] United Nations' Office for Coordination of Humanitarian Affairs, OCHA Metadata Standard, (2006), http://www.humanitarianinfo.org/IMToolbox/08_Data_Standards/Metadata/Metadata_Standard_OCHA_January_2006_v2.doc

[19] Staab, S., Madche, A.: Ontology engineering beyond the modeling of concepts and relations. In ECAI'2000 Workshop on on Applications of Ontologies and Problem-Solving Methods, Berlin, 2000.

[20] Michael K. S., Welty, C., McGuinness, D. L.: OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-guide/

[21] Erdmann M. OXML 2.0. Reference manual for users and developers of OXML-the XML-based Ontology Representation, (2003)