

Deliverable D1.1

Guidelines and tools for producing standards, test-suites and API(s)

Project reference number	e-Content-22236-LIRICS
Project acronym	LIRICS
Project full title	Linguistic Infrastructure for Interoperable Resource and Systems
Project contact point	Laurent Romary, INRIA-Loria 615, rue du jardin botanique BP101. 54602 Villers lès Nancy (France) romary@loria.fr
Project web site	http://lirics.loria.fr
EC project officer	Erwin Valentini
Document title	Guidelines and tools for producing standards, test-suites and API(s)
Deliverable ID	D1.1
Document type	Report
Dissemination level	Public
Contractual date of delivery	M3
Actual date of delivery	31st March 2005
Status & version	Draft
Work package, task & deliverable responsible	Loria
Author(s) & affiliation(s)	Gil Francopoulo, Loria
Additional contributor(s)	With the help of Julien Nioche & Marc Kemps
Keywords	ISO, Guidelines, Process, test-suites, API

Document evolution

version	date	version	date
1.0	15th March 2005	1.6	30th August 2005
1.1	28th July 2005		
1.2	31st July 2005		
1.3	15th August 2005		
1.4	18th August 2005		
1.5	29th August 2005		

Introduction

The purpose of this document is first to help Lirics partners in the ISO process and secondly to provide a coherent methodology among the various Lirics work packages.

The document structure is as follows:

- Guidelines for ISO documents in order to help the Lirics partner in producing the best document as possible.
- Guidelines for Lirics documents.
- Guidelines for Lirics test-suites in order to have common representation schemes and as a consequence, to offer a good reuse among the various partner tools.
- Guidelines for Lirics API(s) in order to present a common syntactic appearance with a common semantic behavior.

The ISO process is outlined in an annex. This enables each partner to know enough about ISO without spending too much time in the huge ISO technical documentation.

A short presentation is outlined in another annex.

1 Guidelines for ISO documents

Rule 1.1: target

Don't forget that your work is dedicated to users. Users can be private or public.

Communicate with foreseen users. Listen to theirs requirements.

Let's recall the first rule of Google that is rather smart: "Focus on the user and all else will follow" (www.google.com/intl/en/corporate/tenthings.html).

Rule 1.2: quality

Write a good document. Be clear, concise and complete. Do not over-use the copy/paste functionality of your text editor. Avoid vaporware content.

For a good document, schedule enough time for documentation and writing. Do not start two days before the deadline.

Rule 1.3: overall process

The recommended approach for defining a standard is as follows:

Step-1 define your terminology

Step-2 model your subject by the means of an English text with possibly the help of UML.

Step-3 submit to experts for comments

Step-4 if needed, return to Step-1

Step-5 derive additional specifications like XML format from your model

Step-6 submit to experts for comments

Step-7 if needed, return to step-1 or Step-5

Rule 1.4: structure of the standard document

The complete set of rules is described in the ISO file "RulesForTheStructureAndDraftingOfAnInternationalStandard.pdf" available on the Lirics web site but it's not easy to read. The easiest way is to start from an existing document.

First, let's note that in the ISO jargon, a chapter is called a "section".

A 'ready to use' plan is as follows:

- Cover page
- Warning (merely copy and paste from an existing standard)
- Copyright notice (merely copy and paste from an existing standard)
- Table of content (to be automatically generated)
- Foreword
- Section-1 Scope
- Section-2 Normative references
- Section-3 Definitions
- Section-4 Specifications
- Annexes

The name of section-4 can be different depending on the subject of your standard.

Rule 1.5: don't be confused by "normative" versus "informative"

"normative" means that it's a part of the standard. "informative" means that it explains the standards.

Concerning the body of the document (i.e. part of the document that is not an annex), the text by default is normative. There are two exceptions. The first exception is a note that is automatically considered as informative. The second exception is an example that is automatically considered as informative.

Concerning annexes, there are two distinct sorts of annexes:

- Normative annexes that are considered as being integral parts of the standard
- Informative annexes that are considered as illustrations of the standard. The purpose is solely to ease the understanding of the standard thus not to restrict or add anything.

In order to avoid any ambiguity, you must insert in the title of your annex one of the two strings: "(normative)" or "(informative)". Normative annexes must be presented before informative annexes.

Rule 1.6: don't mix specifications and examples

The definition and specification sections may contain examples but in separate sub-sections in order to avoid confusing the reader about the scope of the text. It's better to insert examples in a sub-section whose title contains the word "example".

The purpose of an example is to explain, to link to the reality, to give an example of use. The purpose of an example is not to add specifications or to define restrictions to the specification part of the document.

Rule 1.7: cover page

The easiest way is to adapt the content from an existing cover page.

The number of the document (located in the top) is given by the TC37/SC4 secretary.

Never use a date that is automatically computed by your computer: this is misleading, the reader thinks that the document is dated and it's not the case. It's like having an undated document.

In the bottom part of the page, a small item indicates the stage of the document by the means of four digits (for details, see annex-A).

Rule 1.8: definition section

This section is important. You are supposed to define all terms that are either non usual or ambiguous.

For non usual terms, it's recommended to ease your reading in providing a definition. This avoids the reader to misunderstand your specification in case he finds a conflicting definition.

For ambiguous terms, it's better to take position in the "definition section" and to avoid using them as foundations for your model. But sometimes, it's not possible to avoid them in examples. For instance, the term "collocation" is rather used in linguistics in general and within WordNet users, but the meaning in both contexts is not the same (respectively, meaning "co-occurrence" and "MWE"). So, if you want to talk about WordNet, you need to state about your choice.

As a first phase, you can take examples from the various standards within TC37 or from some famous web sites. But it's likely not to be enough: probably more work will be needed.

Provide a good isolated definition is one point. But the overall coherence of your definition section is a more important point. A good set of advices can be found in ISO 704. Briefly, the smart trick in ISO 704 is to sum up your set of definitions by a representation change: you start from a set of isolated texts then you draw the connections between your definitions. You can use 704 graphical notations or UML if you want. The result is impressive. You can find an example in the file "Study on the LMF definition section" located in the Lirics web site.

Rule 1.9: modeling

If you need to model something, the best intellectual tool is definitely UML. UML is defined by OMG. In the Industry, for modeling tasks, UML now replaces SADT, Entity Relation, Merise and OMT. After a long period of method diversity and incompatibilities, UML is recognized and appreciated as the 'de facto' and 'de jure' standard. Let's note that UML is not currently very popular among the Academic field.

The reference is "Rumbaugh, Jacobson, Booch: The unified Modeling language reference manual, second edition Addison-Wesley 2005".

Diagrams ease understanding and communication.

They also permit to produce comprehensive power point presentation in a couple of seconds.

The following types of models can be useful for us:

- class model in order to specify abstract static concepts
- object model (sometimes called instance model) in order to exhibit an example that reifies a class model
- activity diagram in order to present actions and states

A short presentation can be found in an annex of the current document.

If higher level recurring sub-models are needed, the technology known as "design pattern" is recommended. The reference is "Gamma, Helm, Johnson, Vlissides: Design patterns, Addison-Wesley 1995".

Instead of drawing diagrams by hand, it's more efficient to use a software tool that is specialized in UML. If you need to choose a tool, focus your attention on object models because a certain number of software does not implement them and just implement a sub-set of UML.

LMF has been modelled with MagicDraw (www.magicdraw.com) whose "community edition" is free (but restricted).

Rule 1.10: data categories

Don't forget that the Data Category Registry is here to provide the constants you need for your own project and for communication with other tools and data.

Rule 1.11: XML

Don't enter too deeply into XML considerations before having a stable UML model. It's like beginning to code before knowing what to code.

Rule 1.12: underlying technology

Carefully evaluate underlying technologies specifications. Consider as an important criterion the set of constraints you will possibly add to your foreseen users. Prefer stable standards to non-stable Academic projects or niche proprietary technology.

Here is a list of stable technologies:

- XML, aka ISO 8879 as extended by TC2 to allow for XML

- UML-2.0, OMG defined and mainstream in the Industry
- Relax NG, aka ISO 19757-2. Let's recall that only the extended syntax has been normalized. The abbreviated syntax of Relax NG is not part of ISO 19757-2.
- TMF, aka ISO 16642

Rule 1.13: physical format

Contrary to what could be thought, the internal physical for ISO is Word native format. You can use whatever format you want before the DIS stage but at this stage, you must provide a Word file to the central secretary. So, it's more efficient to start with Word format. If a partner does not want to use Microsoft Word an alternative strategy is to use Open-Office that is able to read and write Word native format.

The standard ISO style sheet is available on the Lirics web site as "ISO template for Word".

For dispatching the document to experts and make it available on a web site, it's better to provide a PDF file. Printing accuracy and web interoperability are higher. So, you have to take into account that you will need a PDF converter one day or another.

In LMF, we use Word-2003 together with Adobe-Standard-6 and we did not have any problem.

Rule 1.14: English style

English versions of ISO standards are not written in literary English. Whatever you call it "International" or "Globish" it's simpler than traditional English.

So:

- Avoid long sentences
- Use a spell-checker and a grammar checker
- In the final stages, ask an English native speaker to review your text
- If a non usual word is needed, you must avoid it or define it in the definition section
- Never address the reader. You cannot use something like "Let's note ..."

Rule 1.15: Translation

If a version in French or Russian is foreseen, write the English version first and then translate.

When you write an English document, you must give a French name to the TC37/SC4 secretary. This French title will appear together with the English title within the ISO central database. Ask a French speaker for a translation.

Rule 1.16: language coverage

Avoid thinking in European language centric manner. Never forget Semitic and Asian languages. Just recall that China has 1.6 billion inhabitants and European Community solely 450 millions.

Use Unicode instead of something else for strings.

Consider that your work could be applied to minority languages. ISO 639 language coding on two letters is not powerful enough for instance for Portuguese creoles. ISO 639 language coding on three letters is better.

Reckon with the fact that your work can be applied to languages where the direction of reading and writing must be tackled. Not all languages start from left to right and top to bottom: various other options are used. So, if needed, consider ISO 15924 for code scripts.

Often, a word form is not unique and native. A word can have diverse auxiliary forms. Each form may be a transliteration or a transcription (e.g. "traditional Chinese characters" vs. "pinyin and tone").

Rule 1.17: multi-word expressions coverage

Don't forget that a word is not necessarily restricted to a simple word. Consider multi-word expressions that are of great importance in a lot of languages, being English, French or Chinese.

Don't confuse the notion of word with the notion of token.

Never say that a word is separated by two spaces. Chinese do not use spaces and there is the notion of compound word.

Rule 1.18: Generic versus specific

The correct level of scope is not very easy to find.

On one side, you must not target something too generic and broad because you will not produce anything useful. The purpose of a TC37 standard is not to define a general way of representing information. The danger is to obtain something that looks generic but that is in reality too general to be of any practical use for your foreseen users. Don't produce something that can be called "xxx for dummies".

On the other side, you must not target something useful but specific to a particular use or sub-community. In this case your scope will be too narrow. You must not favor a particular school.

You must base your analysis on best practices of the field.

Rule 1.19: ISO aware

Lirics members should know, at least in the outlines, what is the ISO process. The full description can be found in the Lirics web site in the file:

"ProceduresForTheTechnicalWorkOfISO.pdf".

This document is rather complex to read. You can find a digest for Lirics in annex-A of the current document.

2 Guidelines for Lirics documents

Rule 2.1: style sheet

In order to begin a Lirics document, the simplest thing to do is to start from the file "emptyLiricsDocument.doc" available on the Lirics web site. This file respect the ISO style sheet and has a specific cover page for Lirics.

3 Guidelines for Lirics test-suites

The purpose of these guidelines is to have common representation schemes and as a consequence, to offer a good reuse among the various partner tools.

Rule 3.1: common DTD

All test-suites must exhibit an XML format based on GMT DTD. Let's recall that GMT is defined in TMF (aka ISO 16642). In the ISO 16642 document, the GMT DTD is not defined as a one piece portion. Instead of that, the various fragments are explained in separate sections. In order to ease DTD use, the various fragments have been collected and presented in the annex-B of the current document.

4 Guidelines for Lirics API(s)

The purpose of these guidelines is to provide a good and coherent vision for the API.

Rule 4.1: unit of processing

Depending on the standard, the unit of processing can be different. In certain situations, the whole structure will be manipulated. In other situations, only fine grain operations will be needed.

In the API documentation this separation must be clear because the consequences in terms of time and space are usually important.

In order to avoid user misleading, the following terminology must be used:

- Service API for interactions with a processor (i.e. web services, local program)
- Linguistic Content API for manipulating the linguistic content returned by a processor

Rule 4.2: underlying technologies

The range of targeted usages is rather broad in terms of architecture (local, distributed), in terms of volume (one word, one text, one thousand texts) and mode (interactive mode or batch).

Two technologies are recommended:

- WSDL for a web services usage. WSDL is defined at www.w3.org/TR/wsdl
- Java interface specification for a fast usage without any input-output (i.e. in core access) nor XML parsing.

Rule 4.3: vocabulary

The name of the functions must be carefully chosen. The various Lirics members that work on APIs must use the same vocabulary and apply the same conventions.

The terms "get" for reading and "set" are clear and usual.

The term "load" is not recommended because the user can misinterpret its behavior. The term "load" can be understood as "upload" or the contrary "download". "upload" means a transfer from a client to a server and "download" a transfer from a server to a client. But the term does

not give you any information whether your application is modified or not. It depends whether you are a consumer or a producer. The choice of "get" and "set" seems better.

If a mechanism for browsing a potential large structure is needed, the notion of 'iterator' is recommended. The usual java terminology like "hasNext" and "next" for respectively controlling iteration and stepping forward is recommended.

Rule 4.5: documentation

APIs must be documented by code examples based on use cases. For instance, for an API on the lexicon, two typical use cases will be presented:

- Use case-1: get the whole lexicon, in other terms: a global read-only access
- Use case-2: create one word, in other terms: a local side effect on the lexicon

Annex A: ISO Digest

A.1 Introduction

The full documentation is in "ProceduresForTheTechnicalWorkOfISO.pdf" available in the Lirics web site. Briefly, the outlines can be sum up by the two notions:

- The people and the tasks
- The document

A.2 The people and the tasks

Until the level we have to deal with, there is the notion of technical committee (TC). Ours is TC37. The chairman is Havard Hjulstad and the secretary is Christian Galinski.

Under the TC level, there is the notion of sub-committee (SC). The TC37 four sub-committees are:

group	name	chairman	secretary(ies)
TC37/SC1	Principles and methods	Louis-Jean Rousseau	Claudia Dobrina
TC37/SC2	Terminography and lexicography	Gerhard Budin Madsen	Nicole Sévigny
TC37/SC3	Computer applications for terminology	Bodil Nistrup	Winfried Hennig + Gottfried Herzog
TC37/SC4	Language resource management	Laurent Romary	Kiyong Lee

Within Lirics, the two SC we have to deal with are TC37/SC2 and TC37/SC4.

Under the SC level, there is the notion of working group (WG). A WG has a convener and holds one or several projects.

Each project is assigned a number. A project can have one or more project leaders. The project leader(s) write(s) the document with the help of the convener. For a project with different project leaders, if they do not agree, the convener will help them to reach an agreement.

Lirics scope comprises the following projects:

group	convener	project leader(s)	abbrev term of project	full term of project	ISO number
TC37/SC4/WG2	Kiyong Lee	Eric de la Clergerie	MAF	Morpho-syntactic Annotation Framework	24611
		Thierry Declerck	SynAF	Syntactic Annotation Framework	24615
TC37/SC4/WG4	Nicoletta Calzolari	Gil Francopoulo + Monte George	LMF	Lexical Markup Framework	24613
TC37/SC3/WG1	Sue Ellen Wright	Laurent Romary		Data category revision	12620

Additionally, there are some ISO projects and standards that are (at different levels) related to Lirics.

abbrev term	full term	ISO number
	language codes	639
	terminology work: principles and methods	704
	presentation/representation of entries in dictionaries	Revision of 1951
	country codes	3166
	XML 1.0	8879 revised for XML
	script codes	15924
TMF	terminological markup framework	16642
	relax NG	19757-2
	terminology of language management	21829
	terminology work: guidelines for applying conceptual modeling in terminology work	24156
FSR	feature structure representation	24610-1
LAF	linguistic annotation framework	24612
	word segmentation: principles and methods	24614-1
	Word segmentation: CJK	24614-2

A.3 The document

Each project is associated with a document. The terminology is rather complex because there is a double naming: one for the project stage (e.g. preparatory stage) and one for the document stage (e.g. working draft). What is misleading is that, sometimes the two names are related (e.g. committee stage vs. committee draft) and sometimes they are not (e.g. enquiry stage vs. draft international standard).

The project stage has a double naming: one is numeric (e.g. 30.00) and one is alphabetic (e.g. committee draft registered). There is the notion of stage (e.g. committee stage or 30) and the notion of sub-stage (e.g. committee draft registered or 30.00).

The document stage has a particular naming. It's not very easy to avoid mixing project stage naming and document naming. The two types are fully described by two arrays in ProcedureForTheTechnicalWorkOfISO.pdf, pages 96 and 97.

You need to remember the following usual abbreviations:

AWI	Approved Work Item
WD	Working Draft
CD	Committee Draft
DIS	Draft International Standard
FDIS	Final Draft International Standard
IS	International standard

The project life cycle can be sum up as follows:

project stage	milestone	action/deliverable	sub-stage	recommended elapsed time in months
preparatory stage	starts with	registration of AWI	20.00	0
	intermediate milestones	first WD followed by other WDs	20.20	6
	ends with	approval to register first CD	20.99	12
committee stage	starts with	registration of CD	30.00	-
	intermediate milestones	ballot on CD, including conversion to DIS	30.20	-
	ends with	proposed draft international standard	30.99	-
enquiry stage	starts with	registration of DIS	40.00	24
	Intermediate milestones	ballot on DISs	40.20	-
	ends with	proposed FDIS	40.99	-
approval stage	starts with	registration of FDIS	50.00	30
	intermediate	ballot on FDIS	50.20	33
	ends with	rejected/approval text of IS	50.98	-
publication stage	starts with	approved text of IS	60.00	-
	ends with	publication of document	60.60	36

The recommended elapsed is computed from the AWI date in a cumulative manner. During intermediate stages the project leaders can issue the number of intermediate documents they want. Once the document acquires DIS status, the technical content must not vary: only the form of the document must be improved.

If a project is late, a warning is issued by the central secretary in Geneva and the project is considered as being in warning state. If a project is very late the project is considered as being in critical state. At the end of such a state, if no progress has been made, the project can be automatically cancelled by the central secretary.

So, one advice: if you are late, produce intermediate versions in order to exhibit progress.

Annex B: GMT DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT struct ((feat | brack)*, struct*)>
<!ATTLIST struct
    type    (<insert your comma separated types>) #REQUIRED
    id      ID          #IMPLIED
    target  CDATA       #IMPLIED>
<!ELEMENT feat (#PCDATA | annot)*>
<!ATTLIST feat
    type    CDATA       #REQUIRED
    target  CDATA       #IMPLIED
    source  CDATA       #IMPLIED>
<!ELEMENT brack (feat+, (feat | brack)*)>
<!ATTLIST brack
    source  CDATA       #IMPLIED>
<!ELEMENT annot (#PCDATA)>
<!ATTLIST annot
    type    CDATA       #REQUIRED
    target  CDATA       #REQUIRED>
```

Annex C: short presentation of UML

C.1 Introduction

This is in fact, the subset of UML that is used in LMF. LMF complies with the specifications and modeling principles of UML as defined by OMG. LMF uses a subset of UML that is relevant for linguistic description.

The following notions are used:

- The notion of class
- The notion of relationship
- The notion of instance
- The notion of action
- The notion of package

C.2 The notion of class

A class is a named descriptor for a set of objects that share the same attributes and relationships. Classes are described within a class model.

C.3 The notion of attribute

An attribute is the description of a named element of a specified type in a class; each object of a class separately holds a value of the type.

C.4 The notion of relationship

A relation is a connection between classes. This includes association and generalization. Relations are described within a class model.

C.5 The notion of association

An association is a relationship between two specified classes that describes connections among their objects. The extension of the association is a collection of such links. Associations are the “glue” that holds together the model: without associations, there is only a set of isolated classes. An association holds two ends. Each end has "a multiplicity" and an ordering qualifier.

The multiplicity is the specification of the range of allowable cardinality values that a collection may assume. The multiplicity range is an integer interval with its minimum and maximum values.

An ordering qualifier specifies whether the connection forms a set (an unordered collection) or a list (an ordered collection).

C.6 The notion of aggregation

An aggregation is a form of association that specifies a whole-part relationship between an aggregate (a whole) and a constituent part. It is not permissible for both ends to be aggregates.

C.7 The notion of generalization

A generalization relationship is a directed relationship between two classes. One class is called the parent or the super-class, and the other is called the child or the sub-class. The parent is the description of a set of objects with common properties over all children. The child is a description of a subset of those objects that have the properties of the parent but that also have additional properties peculiar to the child. A parent may have more than one child and a child may have more than one parent. Generalization is a transitive and anti-symmetrical relationship. No directed generalization cycles are allowed. A child inherits the attributes and associations of its parent.

C.8 The notion of instance

An instance is an object that conforms to a class. Instances are not described within a class model but within an instance model (sometimes called an object model). In the LMF document, instances are used in examples of word descriptions.

C.9 The notion of action

An action is a primitive activity whose execution results in a change in the state of the instances. Actions and instances may be linked by dependencies.

C.10 The notion of dependency

A dependency is a relationship between two elements of which a change to one element may affect or supply information needed by the other element.

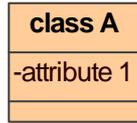
C.11 The notion of package

A package is a grouping of classes and relations. Usually there is a single root package that owns the entire model for a system. A package may contain nested packages. Packages may have dependencies to other packages.

C.12 Graphical notations

Each notion has a graphical notation that is precisely defined as follows:

notation for a class with an attribute



notation for an association



notation for an aggregation



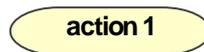
notation for a generalization



notation for an instance



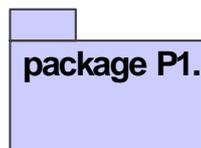
notation for an action



notation for a dependency



notation for a package



Annex D Web pointers

Additional information can be found at:

ISO Central web site in Geneva: <http://www.iso.org>

TC37/SC4 web site in Korea: <http://www.tc37sc4.org>

Lirics web site in France: <http://lirics.loria.fr>