Documenting a Linux Distribution

We will analyze in this presentation an interesting documentation project which, because of its specific constraints, needed innovative solutions to meet traditional business production requirements.

The presentation is made up of three parts:

1. the many constraints that constitute the problem,
2. the technological choices that are the base of the content management system,
3. implementation details on how to solve specific problems.

1. The documents: a Few Elephants

A full Linux distribution represents:

- about 5 CDs full of installable applications;
- those applications range from text editors to Internet servers;
- and are available in a few dozen different languages from Afrikaans to Vietnamese.

The documentation that accompanies the various Mandrake Linux distributions:

- An Installation Guide
- A User Guide
- A Reference Manual
- System and network services documentation

All that in six languages totalizing more than 4000 pages of documentation.

2. Documents Related Constraints

The numbers below are used to refer to those constraints in Section 14.

- Many publishing formats (1): paper, PDF, HTML, online help;
- Many languages (2): more than 6, must be easy to add more;
- Quality (3): resulting documents are meant to be sold, documentation represents a good part of a distribution's added value;
- Version (4):There must be different books for slightly different products.

3. Context Related Constraints

- Time (5): a product changes up to the day its documentation must be printed;
- Release cycle (5b): two new versions of main product a year;
- Versatility (6): some supposedly frozen product features can suddenly be modified;
- Reduced core documentation team (3 people) on different continents (7);
- A lot of contributors for different tasks (proofreading, translation, etc.) scattered all over the globe (7b);
- Use of Open-Source tools (8): as far as possible;
- Content reusability (9): Other people, either in the company or outside of it, must be able to reuse some of the material in other documents.

4. A few Advantages

The context also brings a few advantages to the problem:

- Contributors are quite technically skilled (10);
- page layout design requirements are not complex (11).

5. Using XML DocBook (A)

- The standard format for the documentation of Open-Source applications;
- Separating content from presentation is crucial for constantly evolving material;
- Allows easy use of single-sourcing techniques for different documents;
- provides many output formats out of the box, cosmetic changes are easy to perform.

6. CVS: Concurrent Versionning System (B)

- Ideal for remote individuals to share source files;
- Manages situations where two people are modifying the same file at the same time;
- makes sure everybody always has the latest version of a document at hand.

The use of these technologies which form the base of the system, (XML DocBook and CVS) has quickly been made possible thanks to the technical skills of the team. Otherwise some further developments to hide the XML and CVS complexity would have been required.

7. Internet Communication Tools (C)

In order to create an effective and friendly working environment, as people in the same physical office enjoy, a wide range of communication tools must be available.

- Collaborative Intranet
- Mailing list
- IRC

Needless to say that a certain amount of practice and discipline is needed to reach an optimal level of efficiency, using the right tool for the right purpose.

8. Open Source License (D)

- More in phase with the open source community standards;
- required by people, notably translators and proofreaders, wishing to work for free;
- allows to freely share content with third party authors.

The various open source licenses available provide mechanisms to ensure people don't "steal" material from publicly available documents. Up to now it seems that such licenses do not cause any harm to their users.

9. Dividing Content Into Topic Modules (E)

This technique consists of isolating elementary topics into separate source files (modules).

- Try to make the modules as independent as possible, to allow for reuse in a different context;
- allows you to rearrange the sections of a book at a glance;

- the use of little chunks of information makes content management easier and more accurate;
- allows you to begin translating a document as soon as the first module has been written.

10. Use of Conditional Content (F)

This simply relies on the "condition" attribute available in DocBook. It allows you to tag any piece of data to be included or excluded, depending on specific output needs.

- Allows you to maintain versatile modules, containing common content for various applications, but proposing different specialised versions at will for different targets;
- Provides a means to quickly derive a slightly different version from existing material, while sharing common content, thus making maintenance easier.

This is an example of a paragraph with some alternative content in it:

```
<para>Your machine boots up using a little program called
<phrase condition="IA64">EFI</phrase>
<phrase condition="IA32">BIOS</phrase> that ensures peripheral
initialisation.</para>
```

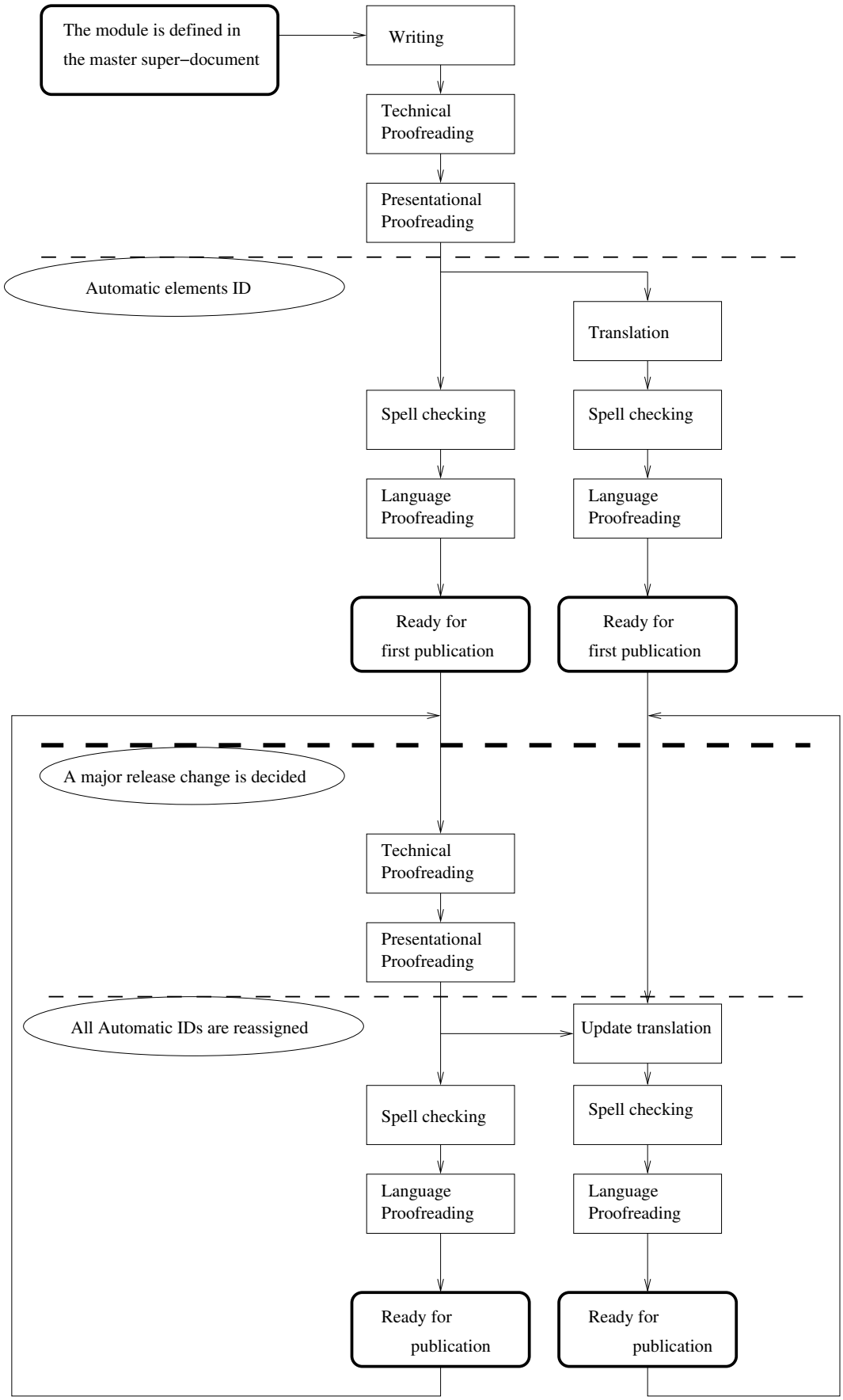
When compiling my manual for a Pentium© processor, I'll direct the system to exclude all content whith `condition="IA64"`. I'll then exclude `IA32` parts for Itanium© processors.

11. Use of a Strict Document Workflow (G)

We call workflow the different states a module must pass through to reach the "good for publishing state".

- Allows different people to perform different tasks on the same module in a predetermined sequential way;
- makes sure no task (notably proofreading) is forgotten;
- allows you to trigger translation of a newly written module before it passes all states (spell checking for example);

This is a Workflow example (`borges-modules-workflow.png`) with six possible states.


The module is defined in the master super−document
Writing
Technical Proofreading
Presentational Proofreading
Automatic elements ID
Translation
Spell checking
Spell checking
Language Proofreading
Language Proofreading
Ready for first publication
Ready for first publication
A major release change is decided
Technical Proofreading
Presentational Proofreading
All Automatic IDs are reassigned
Update translation
Spell checking
Spell checking
Language Proofreading
Language Proofreading
Ready for publication
Ready for publication

12. Synchronization System (H)

This system relies on the assignment of project wide unique IDs to every chunk of data, associated to a revision number. The revision number is meant to be increased by authors each time they change the meaning of the associated chunk. The main problem is therefore in the hands of the authors. They must be educated to change the revision number whenever they actually change the meaning of the data, not for a spelling change for example. No automatic system can detect a semantic change in that case.

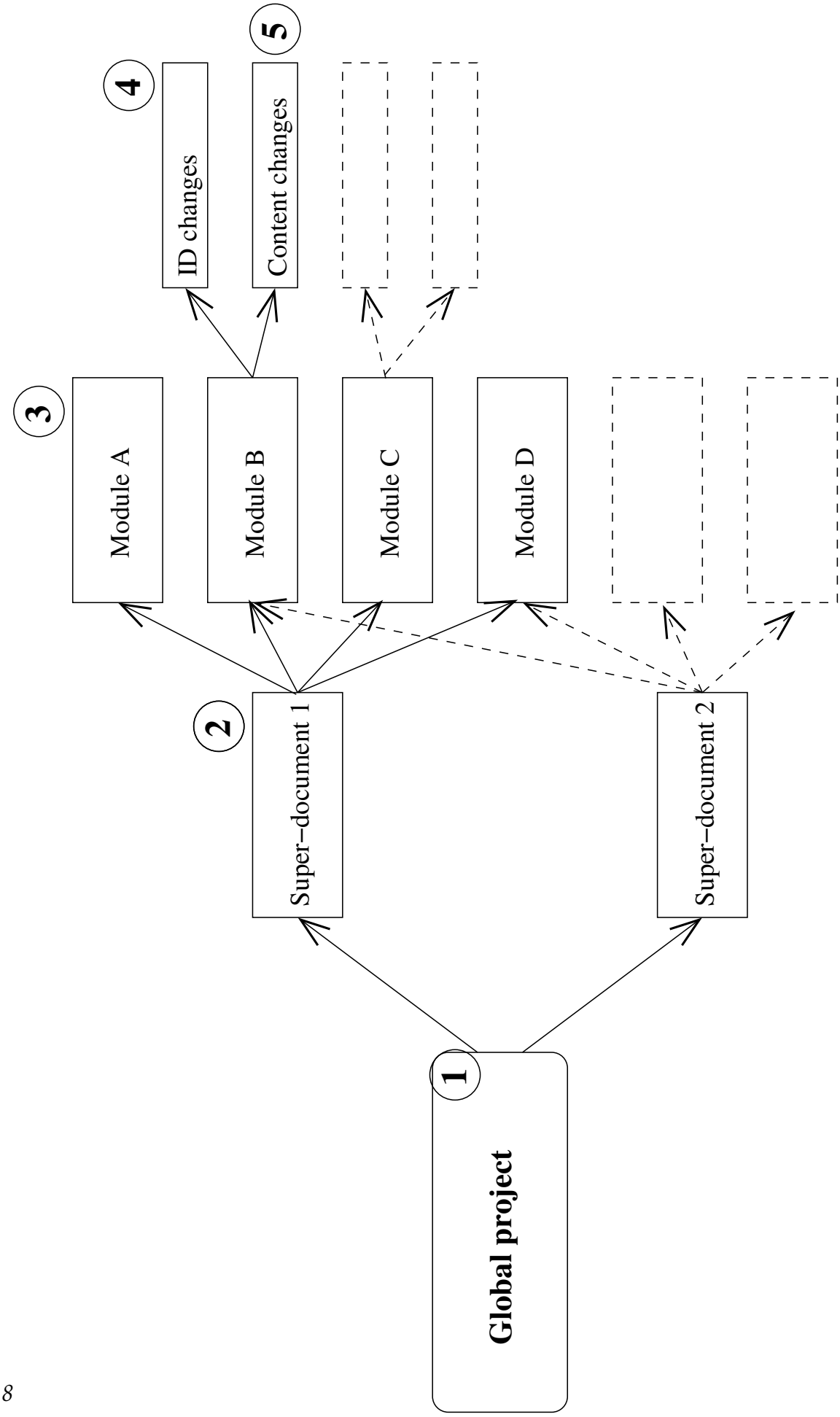
- Allows you to immediately warn translators if they need to update their translations;

13. Automatic reports (I)

1. Global project progression reports;
2. per manual/language progression reports;
3. modules status with respect to workflow;
4. outdated translations reports;
5. side by side original - translations reports.

Additionally whenever a task becomes available for a module, the contributor in charge of that task receives a notice by e-mail.

This is the modules hierarchy (`borges-reports-diagram.png`).


5
4
ID changes
Content changes
3
Module A
Module B
Module C
Module D
2
Super−document 1
Super−document 2
1
Global project

14. Problems & Solutions

This table shows which choices helped solve which problems.

	1	2	3	4	5	5b	6	7	7b	8	9	10	11
A	★			★			★			★	★	★	★
B							★		★			★	
C						★							
D									★	★			
E						★					★		
F				★									
G			★					★	★				
H		★			★		★						
I		★							★				

15. A Practical Example

In April, the documentation team was asked to deliver for May the documentation for the special versions of the operating system for Opteron© and Itanium2© processors. These two versions were based on the previous version (9.0) of the standard operating system which was then at 9.1. Required languages were English and French.

This is how each feature of the system was used:

- A new branch derived from 9.0 is created on CVS to hold Opteron and Itanium specific changes;
- all tasks are assigned to engineers in France and Ukraine, writers in France and Argentina, a proofreader in New-Zealand;
- all needed changes are brought to existing modules in English, using conditional content for one or the other processor or for both at once;
- changes are mirrored in French;
- after an ultimate proofreading, all documents are generated from source and shipped.

The process lasts one month, resulting in a 450 page document published in 8 versions: for Opteron and Itanium, in English and French in PDF and HTML.

16. Conclusion

- Available software coupled together allowed us to build up a customized and powerful Content Management System with no heavy programming;
- XML allowed us to easily add to plain content all the needed meta-information to efficiently manipulate and reuse the information;
- CVS and intelligent use of Internet communication tools allowed remote teams of people to efficiently work together;
- There may be some trunks and tails sticking out of the windows, but the van drives fast and safe.

17. Thank You!

- All the tricks described in this presentation are implemented in the open sourced Borges CMS (`http://www.mandrakelinux.com/en/doc/project/Borges/`) software;
- Get more information about XML DocBook consulting the DocBook Wiki (`http://docbook.org/wiki/moin.cgi/FrontPage`);
- Enjoy the end of your conference.

Special thanks to Anne and Frank Shipley, Fabian Mandelbaum and Roberto Rosselli del Turco for making this presentation presentable.