

Variant management with XML

Divide and conquer!

The consequence of the need for improved and more selective customer relations is a louder call for a more efficient variant management system in technical documentation. There is more to this than just compiling an instruction that corresponds to a product's configuration.

The following technical documentation variants can be identified: product, customer, country and target group-specific variants. In addition, any combinations of the above are feasible.

Using XML

What are the variant management options for the technical editor using XML? First, let's imagine the sales and marketing department of a washing machine manufacturer. This manufacturer wants to offer one of its leading models in a higher price segment, with little technical input. It does this by defining a new machine that differs from the existing one only by the maximum spin speed. It increases the speed from 1500 rpm to 1800 rpm, gives the new model a snappy name and a higher price and "job done".

However, the problems associated with technical documentation may only just be beginning: These can be solved by one or more sophisticated solutions, depending on the tool used. In the worst case, the original manual is simply copied and the value 1500 replaced by the value 1800. A comparatively small variance in products creates a high redundancy of information.

If product changes now occur, the technical editor is required to know which documents are affected and which are not.

Strategic marketing considerations or customer requirements, however, may result in a large number of copies of the original document. If no suitable editing system is used, the "mastermind" talent becomes a key criterion in the technical editor's job specification.

The use of a content management system can create a way out of this dilemma. The original document is split into smaller modules and re-assembled in various ways depending on its intended purpose. However, the consequence of a multitude of variant-forming criteria would be a large number of small and tiny modules. This should present no problem for a content management system. The editor, however, must find out whether a usable module exists before writing. Assuming only a slight benefit, most would create a new module and explain their decision with "before I found this, re-writing was a faster option".

It is also safe to assume that large amounts of text composed from small modules are not always reader-friendly. A large group of technical editors find stylistically standard text fragments that guarantee follow-on text in any context difficult to compile.

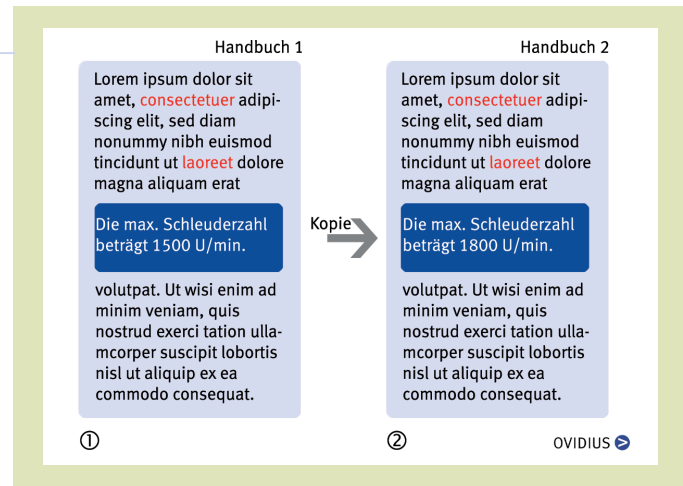


Fig. 1: Variant generation at its simplest - copy manual, change data.

About Ovidius

Ovidius is based in Berlin, Germany, and specialises in XML and SGML software solutions for creating, managing, and publishing of technical and scientific information. We help companies with complex documentation requirements, e.g. automotive, mechanical engineering, software "manufacturers", aviation and defence, medical engineering and IT companies.

This article was published in the 2/2007 edition of the "technische kommunikation" magazine under the title of "Divide and conquer!"

The author, Torsten Machert, is Managing Director of Ovidius GmbH.



Constructive versus destructive

A healthy compromise between the number of modules and zero redundancy often has to be found. In practice, two principles have been established for effectively maintaining a high variant variety with a largely absent redundancy - the constructive method and the destructive method.

Fig. 2 shows product variants in the form of two manual structures that use both identical and different modules. Chapter C, which has different structures in the two manuals, provides an interesting insight. While in "3", the chapter comprises the three sub-nodes C1-C3, "4" has no sub-node "C2". In [1], Dr. Wolfgang Ziegler talks about "fragmented reuse". However, there are still only very few editorial systems that support this interesting functionality.

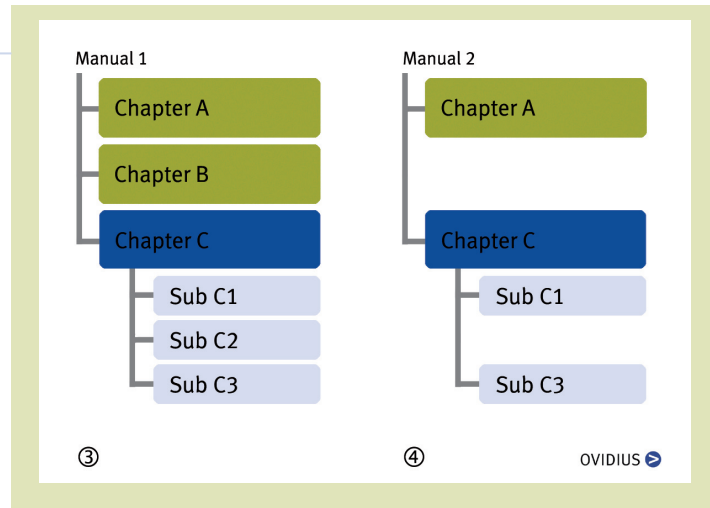


Fig. 2: The absence of a sub-changer, in this case "C2" is described in principle as "fragmented reuse".

Accordingly, the example we used at the outset would be split into the modules shown in Fig. 3.

If, however, further variant-forming criteria were to be contained in or added to a module, this module would have to be split into two new modules. The reasons behind such a procedure have already been discussed in the previous section.

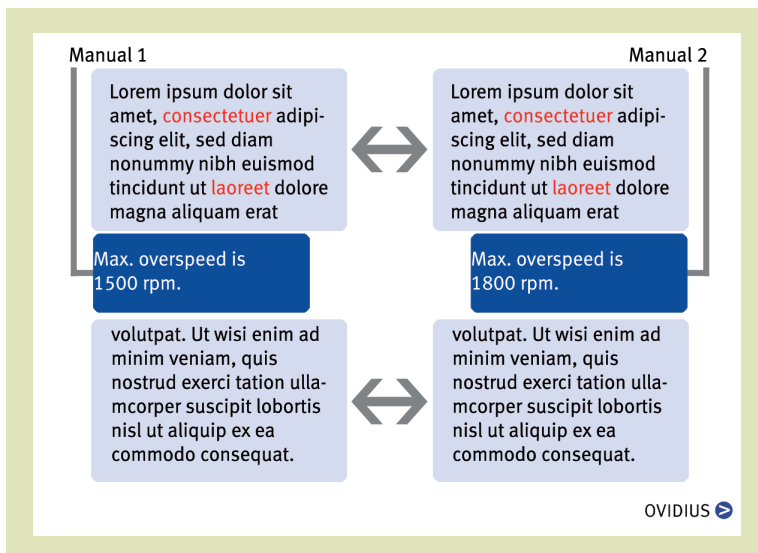


Fig. 3: Spinning speed data following fragmented reuse.

product characteristic the information applies. Fig. 4 shows a simple example. Since it contains no specific variant distinction - the first clause applies for all products, the second clause applies for product variant A. The third clause applies accordingly for variant B. Adobe FrameMaker experts are familiar with this kind of mechanism as "conditional text". XML, however, allows the formulation of conditions that far exceed the potentials of conditional text. Let's take a look at the diagram below, which is taken from an aircraft manual.

The information on the use or non-use of a node is coded in the "effrg" attribute of the "effect" element, which is the first child node of the element for which the usage is defined. A value of "0000-9999" means that the information applies for all configurations. The figures denote the serial number of an aircraft. The value is synonymous with "All". Since these attribute values are process-relevant, they must not be translated under any circumstances. Hence, the value "tous" will never be found, even in a French-

The destructive approach

The destructive approach is used as a principle in technical documentation on civil aircraft. A large aircraft manufacturer has placed 5,000 of its popular twin jet on the market. The manufacturer wants to be able to supply a configuration-specific maintenance manual for each aircraft. The question is, how many maintenance manuals does this manufacturer need to hold? The answer is simple: if it does everything right, only one. This one manual contains everything the manufacturer knows about maintaining the aircraft type. During publication, the information that has no relevance to a particular configuration is filtered out.

First, let's take a look at how the problem shown at the outset could be solved using XML (Fig. 4).

In the document, the XML nodes are told to which variant the information applies. Fig. 4 shows a simple example. Since it contains no specific variant distinction - the first clause applies for all products, the second clause applies for product variant A. The third clause applies accordingly for variant B. Adobe FrameMaker experts are familiar with this kind of mechanism as "conditional text". XML, however, allows the formulation of conditions that far exceed the potentials of conditional text. Let's take a look at the diagram below, which is taken from an aircraft manual.

```
<Section>
<Paragraph>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam eat</Paragraph>
<Paragraph variant="A">Max. overspeed is 1500 rpm.
</Paragraph>
<Paragraph variant="B">Max. overspeed is 1800 rpm.
</Paragraph>
</Section>
```

Fig. 4: The XML document states precisely to which variant which spinning speed data belongs.

language manual. The final clause applies for serial numbers "0000–0003, 0006, 0008–9999". The last example in particular demonstrates that requirements rarely exceed the functions Adobe FrameMaker can offer with conditional text: not only individual values but also value ranges, which have to be interpreted and evaluated with the used tools, are stated.

```
<Task>
<effect effrg="0000-9999">
<Para>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
eat
</Para>
<Para>
<effect effrg="All">
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
eat</Para>
<Para>
<effect effrg="Tous">
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
eat</Para>
<Para>
<effect effrg="0000-0003, 0006, 0008-9999"> Max. overspeed
is 1500 rpm.</Para>
```

Fig. 5: A more precise variant coding using series numbers

```
<Section>
<Paragraph age="99" accompanied="yes" escort="parents">Lorem
dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam eat</Paragraph>
</Section>
```

Fig. 6: Besides the value structure, several criteria can be shown to determine one variant.

```
<Section>
<Paragraph>
<eff>
  <if>
    <condition name="age" value="99" />
    <and/>
    <condition name="accompanied " value="yes" />
    <and/>
    <condition name="escort " value="parents" />
    <andnot/>
    <condition name="residence " value="berlin" />
  </if>
</eff>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam eat
</Paragraph>
</Section>
```

Fig. 7: This sequence permits any logical link.

As well as a complex value structure, there can also be several criteria for the use of node: The Karl-Valentin museum in Munich offers free entry to visitors who are 99 years old and appear to be accompanied by their parents. If you wish to code these conditions in XML, a structure as shown in Fig. 7 can be obtained:

The clause is used only if the attributes of "age", "accompanied" and "escort" contain the required values. This example is limited in that all conditions are linked by a logical AND. If you want to permit any logical operands, different constructs must be used. Fig. 7 shows one option.

References

[1] Ziegler. W (2006): Variant management in CMS – Five methods for fine work. In: technische kommunikation, S. 3, P. 40–44.

Side effects

The destructive approach involves several side effects that must be respected and borne in mind.

If filter criterion "B" is selected, both the "A" blocks are removed. The result, if filter criterion "C" is used, is an empty manual. If filter criterion "A" is used, block "B", including its sub-node "A" is removed from the manual. This renders the reference from the first block "A" on sub-node "A" in block "B" invalid. The link is, of course, also valid if "A" is the filter criterion.

The fact that filtering creates inextricable cross-references may, in particular, be considered a deficiency. Even the constructive approach cannot prevent the presence of "dead" links. The industries and companies that have gone for this kind of procedure, however, consciously accept this deficiency since it is clearly outweighed by the benefits:

- An XML-based coding of the variants imposes no technical limitations.
- Variant management is independent of the used tools. It should, however, be supported by it. The significance of this aspect becomes clear if we consider that information has a disproportionately longer lifespan than the tools used in its creation and publication.
- Variant management can be easily expanded to cover further filter requirements.

The best of both worlds

Now you understand the two methods, its time to put the record straight: S1000D was mentioned above as an example of the constructive approach. In reality, however, the two approaches can be combined and practically applied here. Autonomous and "manageable" modules are combined to create manuals for various target groups and purposes. However, to be able to show minor differences between products, certain nodes within a data module can be told which configurations they apply to. The more flexible concept offers the combined constructive and destructive method: a high variant variety can be formed while maintaining low redundancy.

The technical editor maintains the overview throughout the editorial process. He can see at all times, which information applies for which variants. Appropriate stylesheets offer further support in the used Editor for handling several variants. Finally, the combination of destructive and constructive approaches enables the handling of highly flexible module sizes.

Conclusion

The methods illustrated here are the fundamental principles of XML-based variant management. It is precisely the combination of constructive and destructive methods that creates a powerful tool. The Darwin Information Typing Architecture (DITA) has implemented this approach. The key task of editorial systems is to provide the most generic approach possible for support of such concepts. The technical editor who tailors his variant management system to the potentials of a system has certainly been poorly advised. Hence, changing the system brings about new variant management. If you have gone for XML, you should stay on the path of virtue for as long as possible, perpetuate the openness of XML and implement XML-based concepts.

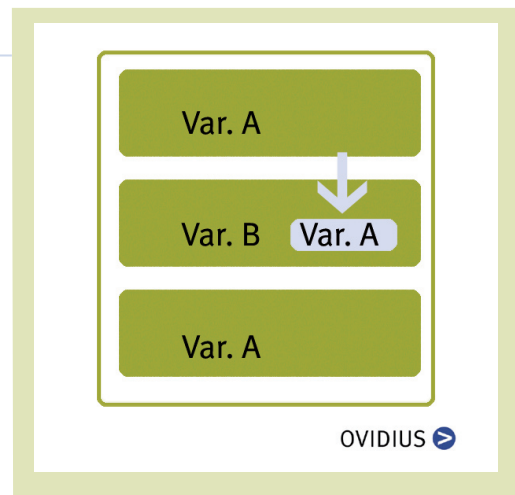


Fig. 8: Invalid links occur after a certain filtering.

Ovidius Berlin
Alte Jakobstr. 79-80
10179 Berlin
Germany

Contact

Phone: +49 (30) 4081895-0
Fax: +49 (30) 4081895-99
Web: www.ovidius.com