The use-cases for a "DITA to Resolved DITA" transformation

George Bina

@georgebina
george@oxygenxml.com

syncro soft

<oXygen/> XML Editor

Resolved DITA

Resolved DITA is a subset of DITA that **does not** contain reuse structures

- keys
- conditional content
- content references
- copy-to
- conref push
- conditional branches

It is basically the equivalent of an EPUB document – a TOC + HTML – only that it comes with DITA semantics

Earlier this year at DITA NA…

DITA translation is broken because the DITA reuse mechanisms breaks it!

Understanding the reuse problem

Rodolfo Raya on dita-users:

reuse/product → "iPhone" or "MacBook”

```
<p>The <ph conkeyref=’reuse/product'/> is great.</p>
```

in Spanish:

```
<p>El <ph>iPhone</ph> es grandioso.</p>
<p>La <ph>MacBook</ph> es grandiosa.</p>
```

We need a different DITA structure for Spanish!

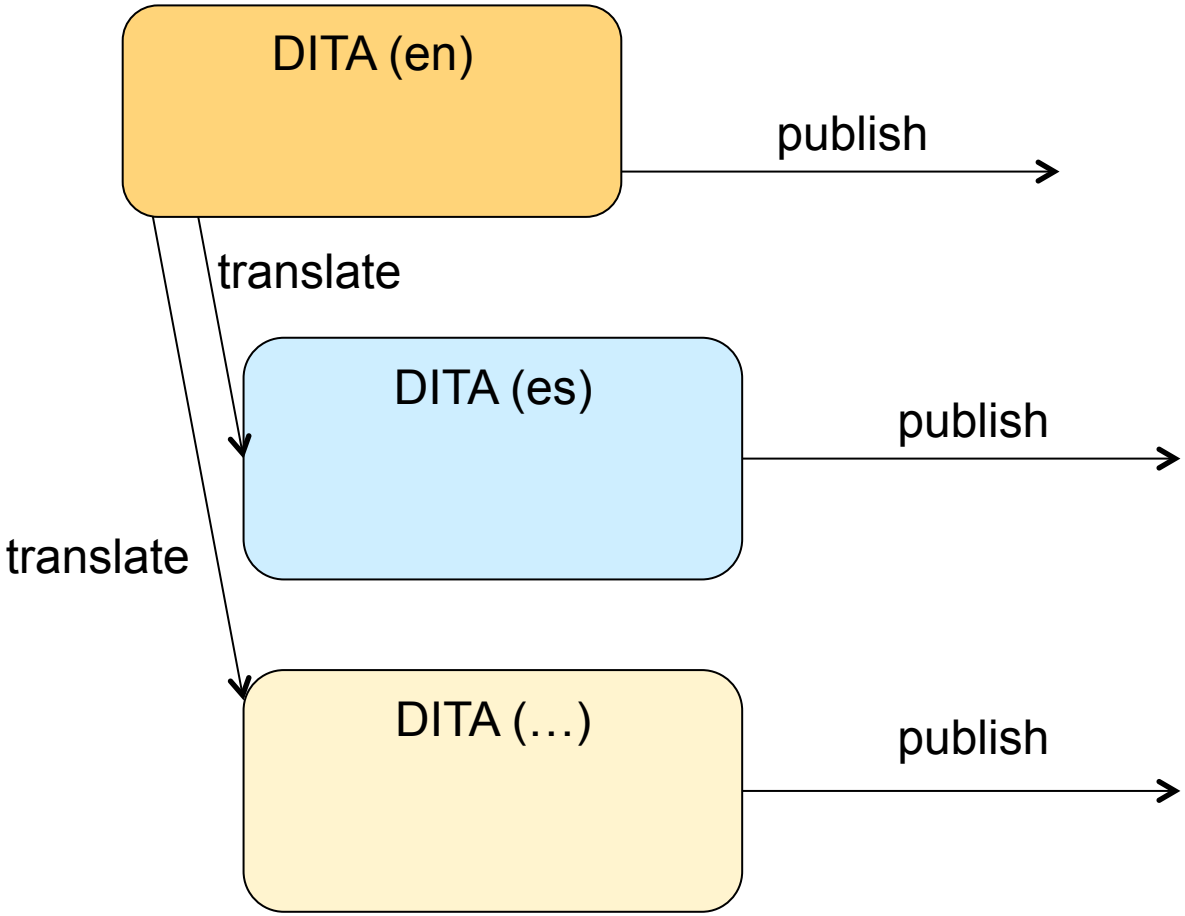
Possible solutions

Restrict the reuse in DITA source so that we do not get any translation issues

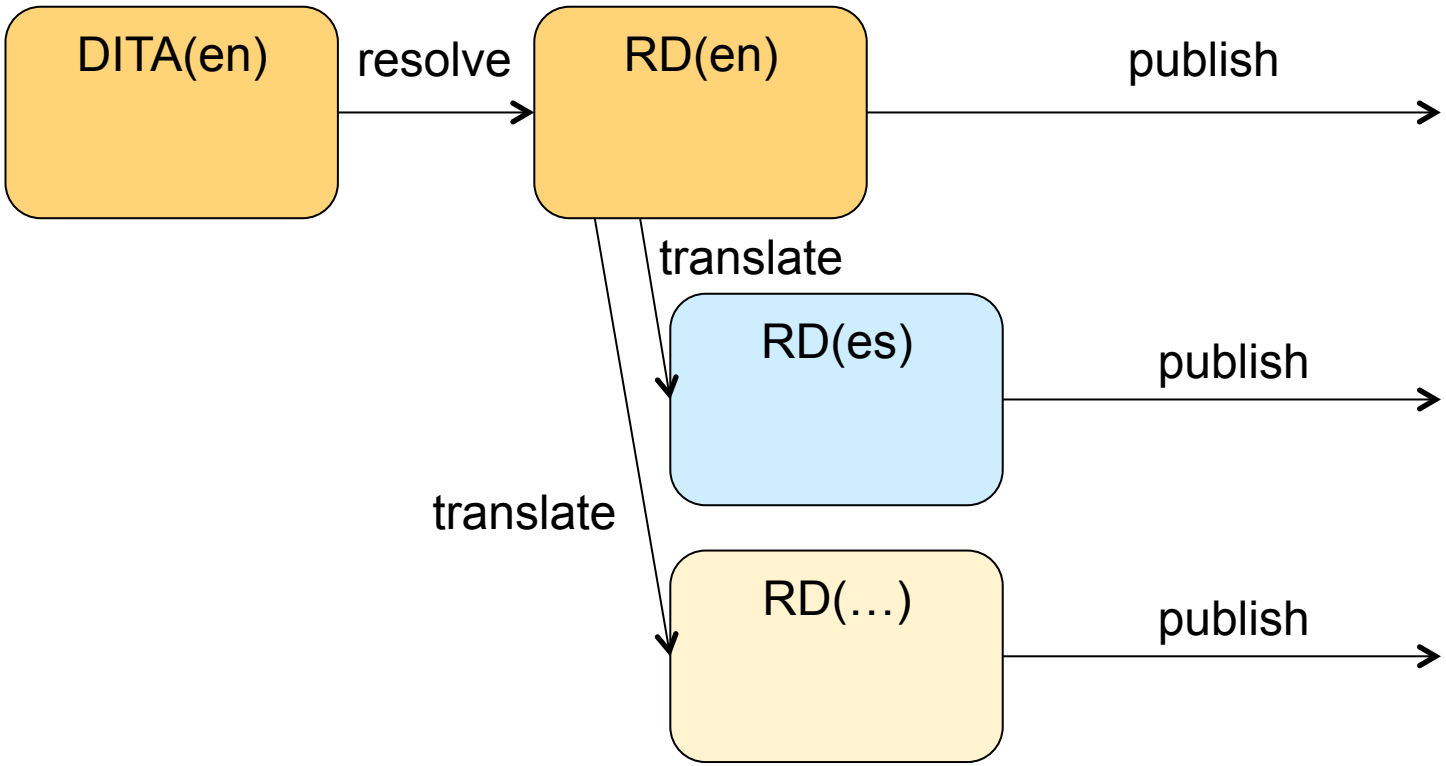
or

Resolve all reuse before translation, avoiding completely the reuse-generated problems

Translating DITA content



Should change to



RD (**Resolved DITA**) is DITA without reuse ⇔ HTML but with DITA semantics

What do we need?

- DITA to Resolved DITA conversion
- Resolved DITA to XLIFF and back
  - we can reuse existing DITA to XLIFF and back transformations
- de-duplicate XLIFF content blocks if exact matches are charged by the translation provider
  - we can use the translation memory for this

If exact matches are not charged then all we need is the DITA to Resolved DITA conversion!

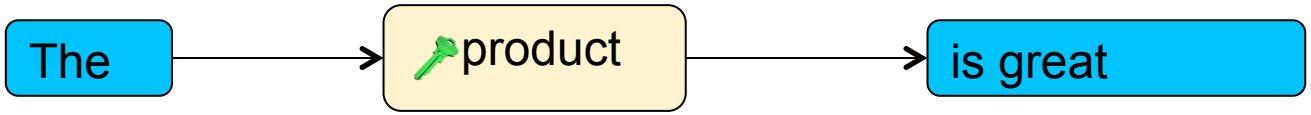
Why not translate the output?

That will work, except for some issues:

- styling/formatting the translated content which will have a different length than the original
- we will need to use many different tools (depending on the number of different output formats)

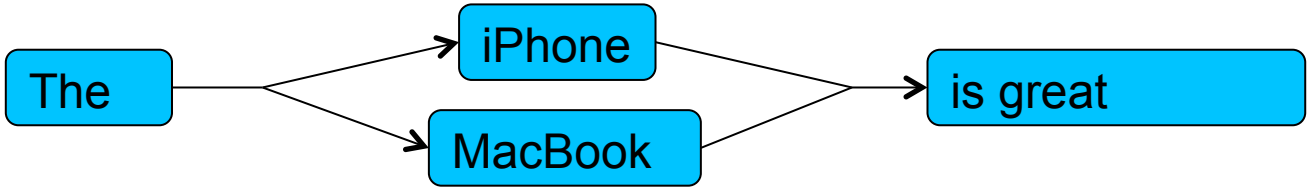
We cannot avoid work

If we are provided with a powerful mechanism to express N different variants into a single form



In order to translate/review we need to analyze all N variants...

- in the single form, identifying variants and expanding the content in our mind
- in the expanded form - may be more work but it is a lot easier



Conclusions

Resolved DITA simplifies the requirements on translation (no DITA knowledge required)

Tools requirements are not very complex and they do not introduce new processing in addition to the DITA to Resolved DITA transformation

Resolved DITA makes it easier if we want to create a publication to include multiple deliverables from the same content

Does it make sense to have a community ready-to-use implementation for this idea?

What triggered this talk?

Earlier this month at the DITA-OT monthly call…

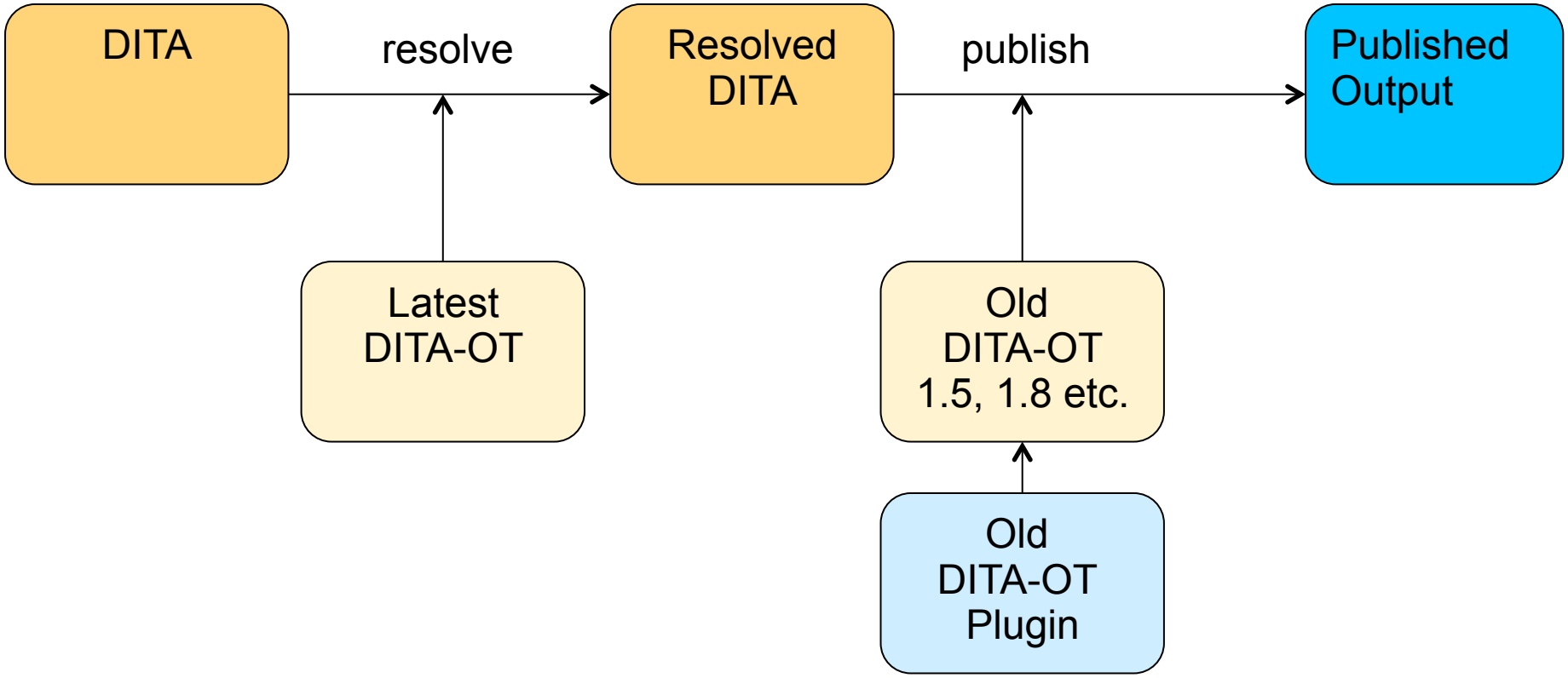
someone said:

“

DITA to Resolved DITA removes the need to update old plugins to work with the latest DITA-OT

“

Use plugins for old DITA-OT versions



Current status

DITA-OT already provides a plugin to generate resolved DITA

org.dita.normalize

https://github.com/dita-ot/org.dita.normalize

But the result is not 100% valid DITA

New DITA processing proposal

Split DITA processing in two, processors should be

- DITA Core processors (DITA to Resolved DITA)
- DITA Publishing processors (Resolved DITA to Output)

Publishing processors depend only on Resolved DITA

they work with any DITA version as Resolved DITA stays the same

Separate concerns

- separate content from styling
- separate DITA reuse from DITA semantics

Final ideas

We need a specification

- Resolved DITA should be a clearly specified format – it will be the input to DITA Publishing processors

Resolved DITA removes the cost of updating plugins to the latest DITA-OT

The split of processing in Core and Publishing processors reduces publishing time

- We transform DITA to Resolved DITA once and then we publish that through multiple publishing processors

Does it make sense to have a community ready-to-use implementation for this idea?

Thank you

Questions?

george@oxygenxml.com

@georgebina

http://www.oxygenxml.com