CIDM
DEC 1, 2021

Refactoring XML Content for Iterative Model Development

with Rob Hanna,
President & Co-Founder

precision content

Rob Hanna

Co-Founder and Chief Information Architect at Precision Content

2014 STC Fellow

Voted into the Top 25 Global Content Experience Influencers for 2017

Helping our clients empower their people through better content, processes, and technology

Twitter: @singlesourceror

Session summary

Developing robust content models can never be a once-and-done endeavor where IAs are required to incorporate all aspects into the design prior to authoring. Instead, an iterative approach is needed to allow for incremental improvements not just during setup but long into production.

This session explores methods Precision Content uses to refine models and update the content corpus through scheduled refactoring activities.

Polling slide #1

How long have you been working in your CCMS?

Key Concepts

Iterative & incremental development

Technical debt

Refactoring

Iterative and incremental development

Is a form of Agile Project Management …

- used to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental)", and
- allows teams to learn from development of earlier parts or versions of the system, and user feedback.

Start small, divide into functional modules, discover requirements per module, build test and review, and iterate.

Helpful Example

Iterative and Incremental Product Development (Pic Credit: Ricardo Sazima)

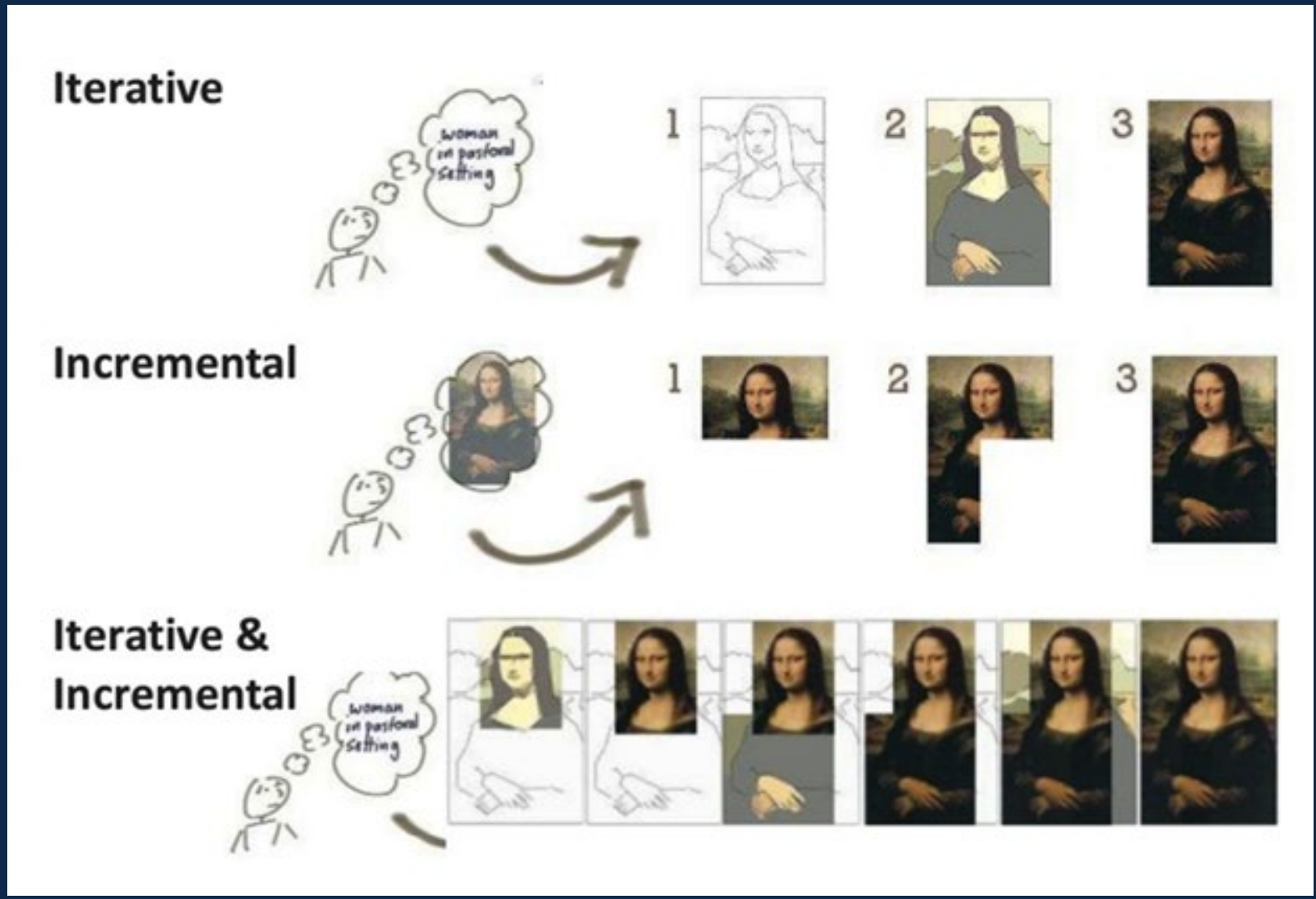
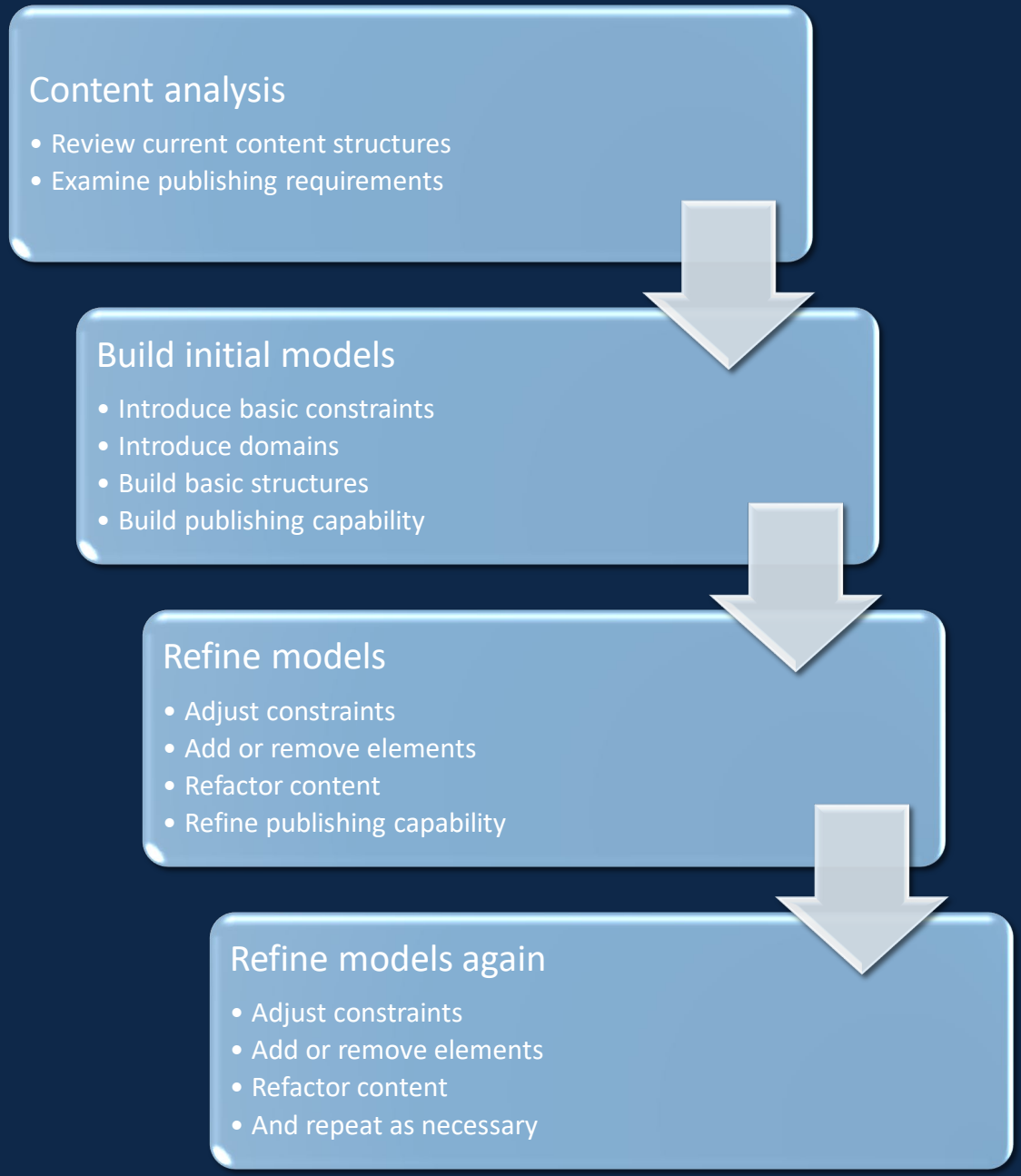


Image taken from "Agile Methodology — Incremental & Iterative Development: A Case Study" by Rashmi Chatterjee https://medium.com/@rashmichatterjee88/agile-methodology-incremental-iterative-development-a-zomato-case-study-e23f42935a71

Iterative and incremental content model development



Technical debt

Technical debt is the difference between **optimal** and **minimally viable** product delivery

We are often required to make some allowances to sacrifice “the ideal” for “the expedient”. With content this regularly means that we can accept less consistent structures and metadata so long as it “looks good” and does not contain informational errors.

These inconsistencies can dramatically confound publishing and effective content reuse down the road.

How much technical debt are you carrying in your source content?

Polling slide #2

How much technical debt are you carrying in your DITA source content?

What is refactoring?

For computer programming

- In computer programming and software design, code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior. Refactoring is intended to improve the design, structure, and implementation of the software (its non-functional attributes), while preserving its functionality.
  - Wikipedia

For information architecture

- In structured authoring environments, XML content refactoring is the process of restructuring existing XML code—changing the factoring—without changing the rendered content. Refactoring is intended to improve the design, structure, and implementation of the content, while strictly preserving its meaning. Refactoring can be applied to topics, maps, and metadata.
  - Precision Content

Polling slide #3

How often do you refactor your content corpus?

The problems we're trying to solve

Common challenges with CCMS deployments

Typical issues may include

- Content Model Changes
- Bringing in a New Corpus of Content
- Improving Reuse Capabilities
- Content Cleanup

Content model changes

Content model changes requiring updates to production DTDs

- Content model is too restrictive
- Content model is too loose
- Too many unused elements
- Some elements are consistently misused
- New elements needed for new publishing capabilities
- Upgrading to new DTDs – DITA 2.0
- Moving to a new authoring or review tool

Bringing in a new corpus of content

Your company has acquired a new set of products with their own DITA content

- Adaptations are required to accommodate new content or publishing capabilities
- Style guide changes requiring broad content updates
- Harmonization of metadata across all products
- Map updates to standardize content structures
- Clean up of legacy content

Improve content reuse opportunities

Finding and fixing content reuse across a corpus can be challenging

- Cleaning up profiling attributes and conditional reuse
- Introducing keys or resolving keys in the content
- Fixing or adding new conrefs across the corpus
- Adding or repairing relationship tables

Content cleanup

Making programmatic changes to text content is challenging. Replace All won't do!

- Standardizing capitalization of titles in your content
- Adding ALT text to tables and images
- Fixing links to external resources
- Changing product names or components across the corpus
- Identifying and resolving style guide violations
- Fixing punctuation across the corpus

Tools for refactoring

Using what we already have

Choosing the right tool for the job

The tools you select will depend upon the size of the problem you are trying to solve

NO REFACTORING – *Impact 5 or fewer topics*

- Changes by author in XML editor

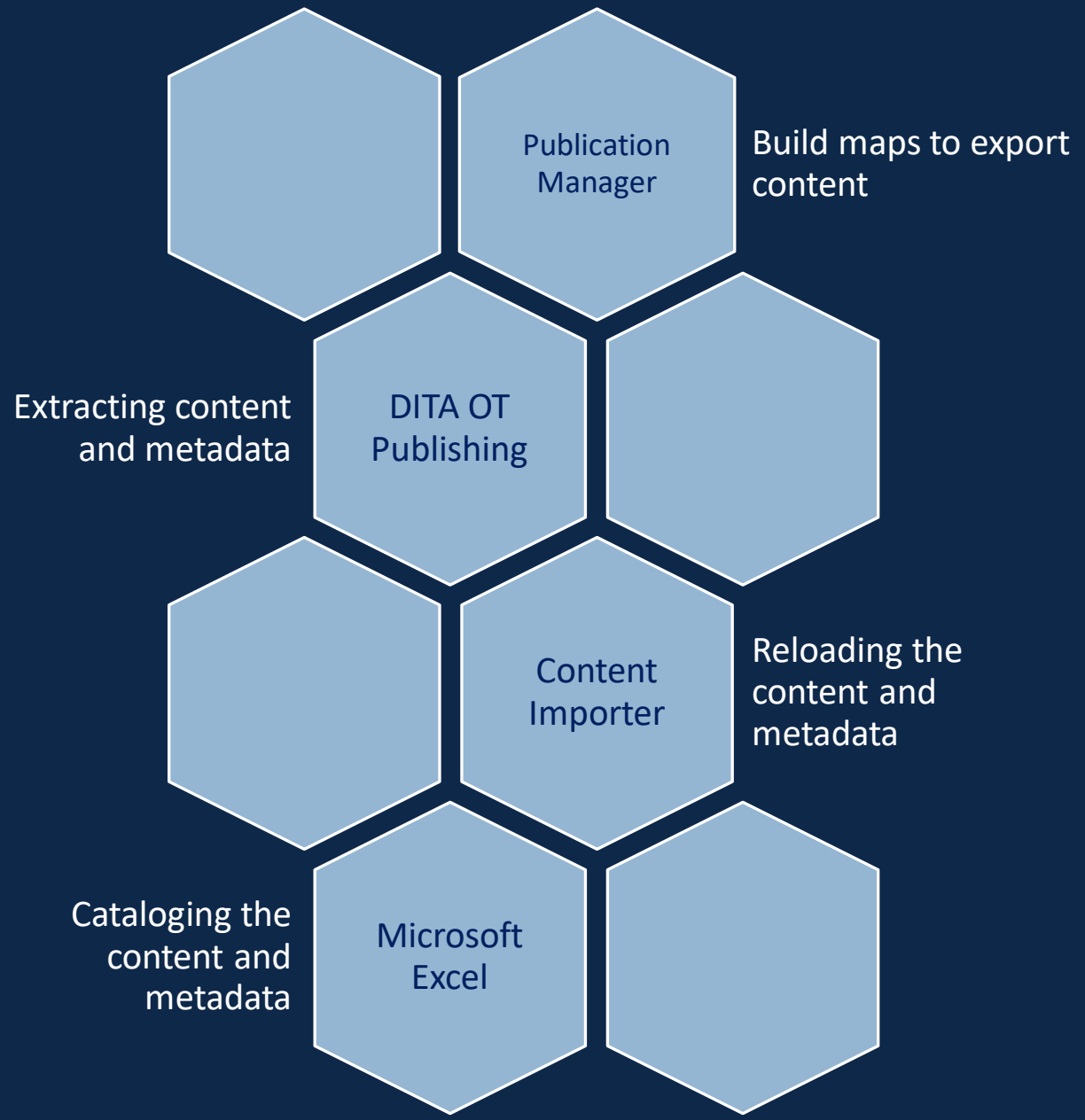
LIGHT REFACTORING – *Impact 6-20 topics*

- Brute force methods
  - Text search and replace
  - Oxygen XML Refactoring

ITERATIVE REFACTORING – *Impact 20+ topics*

- XQuery with Base-X
- XSLT
- Other XPath-based scripting (MSXML)

Tools essential for refactoring with Tridion Docs



Microsoft Excel demonstration

We use Microsoft VBA and MS XML 6.0 to read and write XML to Excel.

- VBA parses map, topic, and metadata to worksheets in Excel for analysis and updates
- VBA is used to build topics and maps or insert new elements to the XML files on the local machine

Process for refactoring your corpus

1. Test
2. Test
3. Test again

Refactoring process

1. Plan
2. Develop and test scripts
3. Extract production content
4. Execute scripts and evaluate results
5. Reload content into production
6. Validate reload

Stage 1: Plan

Always start with a well-constructed plan for refactoring

- Schedule periodic refactoring events during implementation and onwards into production
- Work around publishing deadlines
- Maintain a prioritized list of items for refactoring
- Minimize downtime for authors by scheduling the exercise over a weekend
- Do not stray from your plan with last minute requests

Stage 2: Develop and test scripts

Start developing your refactoring scripts a minimum of two weeks before your refactoring event.

- Automate each step of the refactoring and test, test, test
- Start development on your local machine and move to a sandbox environment when ready
- Source control your scripts
- Document a full system test to validate the refactoring exercise

Stage 3: Extract production content

Extract your content to perform the refactoring

- Announce a content freeze to cease all further writer work
- Make sure all content is checked into the repository
- Take a snapshot of your CCMS database to create a restore point
- Export your content
  - Build your extract maps
  - Run DITA/XML publishing job to export the content and .met files to your local machine
- Publish your content to all transtypes so that you have something to test against after the refactoring

Stage 4: Execute scripts and evaluate results

Build in sufficient time to run your scripts and validate each step

- Don’t wait until the end of your refactoring to test your content, test after each iteration
- If part of the refactoring fails, remove it from the refactoring tasks for this event
- Load content to your sandbox for testing before moving to the next step
- You may need to refactor links, topic references, and conrefs to use GUIDs rather than the exported filename prior to loading it back to the system

Stage 5: Reload content to production

Use Content Importer to reload the content into the system

- Ensure your metadata files specify the correct version to reimport back into the system. In the absence of a specified version number, Importer will import the new content as the next revision to version 1.
- Ensure the system is configured to import directly to released status
- If the reload fails, back out and restore the database from the snapshot

Step 6: Validate reload

Test every output to validate that the content has not changed unexpectedly

- Use a robust PDF compare tool to easily highlight any differences in output for PDF, RTF, and other FO-formats. We use Workshare Compare for testing
- Use a code comparison tool to highlight any difference to output for HTML, JSON, or other XSL-formats
- Don’t forget to test your content in all pertinent authoring and review tools
- If testing fails, back out the changes by restoring the database snapshot and reschedule refactoring once you’ve worked out the bugs

How we can help you

Visit us at www.precisioncontent.com

Experts in DITA and intelligent content delivery

We're a full-service, end-to-end technical communications consultancy, technology innovator, and systems integrator offering professional services, training, and tools.

About Precision Content

precision content

Areas of Expertise

Precision Content is home to thought leaders and expertise in the areas of

- DITA/XML design and implementation
- microcontent best practices
- structured authoring methods
- content lifecycle management
- information architecture
- content strategy, and
- omnichannel content delivery.

precision content

Trust in Your Content

Thank you

Reach out to us at
more-info@precisioncontent.com