Migration to DITA/CCMS – Implementation Checklist

Migrating legacy content into a structured format (like **DITA** XML managed in a **Component Content Management System** or CCMS) requires careful planning and execution. The following checklist breaks down key phases and tasks to ensure a successful migration. It reflects industry best practices and the roadmap **before, during, and after** migration, aligned with an AI-ready content strategy. *(Remember: migration is not a one-off project but a sequence of managed decisions.)* With extensive experience delivering solutions for global enterprises including HP, Panasonic and Horiba Medical, our expert teams are on-hand to guide you.

Pre-Migration Planning

- **Define Objectives and Business Case:** Clearly identify why you are migrating (e.g. multi-channel delivery, faster updates, reduced duplication, compliance needs, AI-readiness). Establish the value and **success criteria** upfront (e.g. reduce translation cost by X%, improve content reuse) for stakeholder buy-in.
- **Secure Executive Sponsorship and Stakeholder Alignment:** Get leadership support and involve all relevant stakeholders early – documentation teams, IT, product owners, QA, localization, etc. Ensure everyone understands the plan and their role. Early consensus on the decision to migrate and the chosen strategy is crucial to avoid roadblocks later.
- **Build the Core Migration Team:** Assign a project lead and define team roles (e.g. content strategist, information architect, migration engineer, lead writer, tool specialist). Clarify responsibilities for content cleanup, conversion, validation, and overall project management. (See **Roles & Responsibilities** section in the template for examples.)
- **Develop a High-Level Timeline & Budget:** Plan a phased schedule for the migration (audit, design, pilot, full rollout). Include time for tool selection, testing, and training. Secure a budget covering CCMS licensing (or cloud subscription), possible conversion tools or services, and training programs. Keep the plan realistic by anticipating learning curves and potential rework.
- **Communicate Early and Often:** Set up regular check-ins or a steering committee. Keep stakeholders updated on progress, findings from audits, and decisions (e.g. what will **not** be migrated). Early communication builds confidence and surfaces concerns in time to address them.

Content Audit & Complexity Scoring

- **Inventory All Content:** Compile a comprehensive content inventory. List what exists (documents, topics, pages), where it resides, format (Word, FrameMaker, HTML, PDF, etc.), and who "owns" or maintains it. This can be done in a spreadsheet for traceability.
- **Assess Complexity and Quality:** For each content set, capture complexity factors – e.g. use of tables, embedded graphics, lengthy procedures, conditional text or filtering, duplicate content, and any reuse or cross-references. High complexity (lots of conditional text, deep nesting, etc.) may indicate more effort to migrate. Also note content quality or pain points: areas with known inconsistencies, frequent updates, or user complaints.
- **Score or Categorize Content:** Using the factors above, assign a **complexity score** or category (e.g. Low/Medium/High complexity) to each item or content type. This

helps size the migration effort. For example, a simple FAQ in HTML might be "Low" complexity, while a 200-page guide with conditional steps is "High."

- **Identify Redundant or Outdated Content:** Flag content that is obsolete, trivial, or duplicated. Decide what can be **archived or deleted** instead of migrated to avoid carrying over content debt. It's often better to retire unused legacy docs than to convert them unnecessarily.
- **Determine Migration Candidates:** Based on the audit, determine which content is a priority for structured conversion and which might be left as-is (or handled later). **High-value, frequently-used content** should be slated for migration first. The audit deliverables typically include a **content inventory** and possibly a **maturity scorecard** that together inform the migration scope. *(Reminder: "You are not migrating documents; you are migrating a content system" – consider how all pieces interrelate rather than treating content in isolation.)*

Information Model Development

- **Define the Information Model:** Outline **how your content will be structured** in the new system. Identify the content types you need – for example, DITA topic types such as Concept, Task, Reference, as well as any specialized types your domain requires (product specs, policies, etc.). Remember, *DITA itself is a standard toolkit, but your information model is the tailored set of content types and rules your organization will use*.
- **Establish Topic Structure and Granularity:** Decide on the modular breakdown of content. Each topic should cover a single idea or procedure (one task = one how-to topic, etc.). Avoid mixing multiple subjects in one topic. This **one-topic-one-purpose** approach ensures consistency and reusability. For example, a concept topic answers, "What is this?", a task answers, "How do I do this?", and a reference provides factual details.
- **Define Metadata and Taxonomy:** Design a metadata schema for your content. This includes attributes like product name, version, audience, skill level, region, etc. Controlled vocabularies (taxonomies) should be established for consistent tagging. Metadata is critical for future **filtering, dynamic delivery, and AI retrieval** – e.g. ensuring an AI or search can fetch all "installation" instructions for product X in French.
- **Prototype the Model with Samples:** Draft a few sample topics in the proposed structure to validate the information model. For instance, take a representative procedure and model it as a DITA Task topic (with steps, prerequisites, expected results), ensuring it fits the structure. This helps catch any model gaps or needed adjustments (such as adding a custom element or metadata field) **before migrating en masse**.
- **Align the Model with Business Needs:** Ensure the information model reflects how your business and users consume information. If your products are frequently updated or highly regulated, the model should facilitate easy updates and include relevant compliance metadata. If customers need quick answers, the model should favor small, searchable topics. **Design for the future:** think about how the structured content might feed not just manuals, but support portals, chatbots, or other channels.

Governance and Content Creation Guidelines

- **Establish Governance Early:** Formulate governance rules and a governance team **before content conversion begins**. Governance covers *who* will own and approve content moving forward, and *how* you will maintain quality and consistency. Identify **content owners** for each area (each topic or set of topics should have an owner

responsible for its accuracy over time). Define **approval workflows** (e.g. a subject matter expert and a documentation lead must sign off on any update to a procedure).

- **Develop Structured Writing Guidelines:** Update or create your style guide to include rules for structured content. Authors need clear guidelines on writing **topic-based** content (e.g. task topics should begin with an imperative verb and include prerequisites; concept topics should not contain procedural steps; limit the length of paragraphs for readability). Include rules for using metadata and consistent terminology. These guidelines enforce the information model in day-to-day writing.
- **Implement Quality Gates:** Plan how you will **validate content** during and after migration. Utilize automated checks where possible: **linting tools** for DITA or Schematron rules to enforce your model, link checkers for integrity, and content reuse checks to prevent duplication. Set standards for acceptable build errors or style violations (ideally zero). Establish a content review process that happens *before* publishing, not after. This might involve using the CCMS's workflow to route content for review and recording approvals.
- **Standardize Metadata and Terminology:** Create a controlled vocabulary for product names, features, units, etc., and ensure all writers use these terms. Consistent metadata values (for example, always tag the "Electrical Safety" topics with exactly that phrase) will improve findability and filtering. If your CCMS (like **Tridion Docs**) supports **taxonomy management**, configure it with your categories and required metadata fields. This not only helps during migration (when tagging legacy content) but sustains content quality long-term.
- **Plan Change Management and Training:** Recognize that moving to structured content is a big change for authors and other teams. Set up training sessions on DITA and the new CCMS. Designate a few team members as **"content champions"** who can coach others and reinforce the new practices. Communicate the **benefits of the change** (faster reviews, less copy-paste, easier publishing) to motivate adoption.
- **Prevent New Content Debt:** *Content debt* refers to the accumulated chaos of outdated, inconsistent content that slows teams down. Governance is how you prevent migrating your old content problems into the new system. **If you don't change how content is created and maintained, you will recreate the same problems in the new system**. Set up mechanisms (like quarterly audits or content quality metrics) to continuously monitor and address issues. Governance is an ongoing effort, not a one-time task – it's your strategy to **keep content healthy** after the migration. *(In short: migration is a one-time effort; governance is the long game.)*

Migration Execution Strategy: Rewrite, Refactor, or Convert?

Not all content should be migrated the same way. Choose a conversion approach per content set, balancing speed vs. quality:

- **Rewrite (Clean):** Re-author content from scratch in the new structure, using the source material only as reference. This is best for **outdated, highly inconsistent, or critical/high-risk content** that warrants a fresh start. Rewriting yields the cleanest, most uniform topics (maximum long-term quality), but it's the highest effort. Plan to rewrite for content that has a lot of legacy quirks or known issues that automated conversion can't fix (or where content needs a major overhaul for clarity).
- **Refactor (Smart):** Convert content with an automated or semi-automated process, then *manually restructure and refine it* into proper topics. This approach works

well for **high-value content that has reusable parts** but wasn't written in a topic-oriented way. For example, you might convert a large Word manual to DITA, then split merged topics, reorganize sections into concept/task topics, and apply consistent tagging. Refactoring requires strong governance and training (writers must know how to reshape the converted content), but saves effort compared to rewriting everything.

- **Convert (Fast):** Use an automated conversion tool or service to directly **bulk-convert** content into DITA or structured format with minimal changes. This is suitable for **stable, well-structured legacy content with low complexity** and infrequent updates. For example, hundreds of simple reference docs or datasheets might be converted via script and largely left as-is (aside from minor cleanup). Conversion is the fastest path but **requires thorough QA** – you'll need to spot-check mapping accuracy, fix any broken links or formatting, and ensure the output is valid.
- **Archive or Defer:** In some cases, the best migration decision is **not to migrate** at all. Legacy content that is no longer needed by users can be archived in its current format (e.g. kept as PDF for record-keeping) instead of consuming migration effort. Make these decisions during planning to avoid wasted work.

Most teams use a hybrid approach – for example, *rewrite* the top 10% of critical docs, *refactor* another chunk that has high reuse potential, and *auto-convert* the remainder for speed. Document the chosen approach for each content category in your migration plan. A good rule of thumb from the field: **low-risk, simple content can be converted; high-value content deserves refactoring; high-risk or very outdated content might need full rewrite**. By tagging each content item with an approach (convert/refactor/rewrite/archive), you can prioritize resources and set expectations for effort. (The template includes a **Migration Approach Summary** section to list these decisions.)

Translation Strategy and Localization Considerations

If your documentation is translated into multiple languages, the migration must account for localization from the start. Structured content will impact translation processes and costs, so plan a strategy early:

- **Choose a Migration Path for Translations:** Decide how to handle existing translated content:
- *Source-Only Migration:* Migrate the source (e.g. English) content to DITA first, **do not migrate legacy translations**. After the English content is structured and finalized, treat the first release as a new translation cycle for all languages. This often means the initial post-migration release incurs higher translation cost (nearly a retranslation) because previous translation memory matches won't align due to the new topic segmentation. The upside is you ensure all translations align perfectly with the new content structure moving forward.
- *Source + Legacy Translations:* Attempt to migrate existing translated files into the CCMS/DITA as well. This is complex – you'll need to convert each language and map it to the new structure. It can preserve some translation investment (TM leverage), but expect **mixed results**: some parts will align, others may need retranslation or heavy editing. This path requires careful content mapping and maybe vendor support to ingest translations.
- *Staged (Phased) Approach:* Focus on migrating the source first and doing a **pilot translation** in one or two languages to validate the process, then migrate additional languages in stages. For instance, migrate English and do a pilot French translation in the new system. Once you prove it works (and fix any model issues

affecting translation), you then roll out translations to other languages. This spreads cost over time and reduces risk.

- **Update Translation Memory Expectations:** Be prepared that moving from unstructured docs (e.g. long paragraphs) to DITA topics will **change segmentation**, lowering TM leverage initially. Translation Memories operate on segments (often sentence-based). When content is reorganized into smaller topics and chunks, many segments won't match exactly. It's common to see a **drop in pre-translation matches for the first DITA release** (sometimes the first structured release is almost like translating from scratch). After this, TM leverage will climb again in subsequent releases as the new content stabilizes. Plan and budget for this first-cycle translation effort so it's not a surprise.
- **Test the Localization Workflow Early:** As part of your pilot, include a **translation round-trip test**. For example, take a handful of topics, send them through your translation process (using your Translation Management System or vendor), then import the translations back into the CCMS and publish. Verify that:
- The content and **metadata** export and re-import correctly (e.g. no tags dropped, metadata like product names remain intact in the target).
- Conditional text and reuse elements function for translated content (are conditional sections appropriately translated or left out based on language rules?).
- Your publishing output can handle multi-language content (fonts, indexing, etc.).
- **Translation memory leverage** is measured to see the impact of the new structure (you might find, for instance, only 30% match vs 80% previously, due to new segmentation).
- Any integrated **machine translation** or AI-assisted translation tools in your workflow are compatible with the DITA content.
- **Coordinate with Localization Teams:** Involve your localization manager or vendor in migration planning. They can help decide whether to carry over legacy translations or retranslate, and set up appropriate workflows. Ensure that your **CCMS integrates with translation tools** (many CCMS, including Tridion Docs, offer localization connectors or at least XML export/import for translation). If using **AI translation** (MT), determine a process for post-editing and quality assurance – *AI can speed up translation, but human review remains essential for accuracy and terminology*.
- **Treat Each Language as First-Class Content:** In an AI-driven world, every language's content might feed user-facing answers (for example, a Spanish chatbot or knowledge base). So, apply the same structure and quality rigor to translations as to source content. Plan for **terminology management** and possibly separate approval for translated docs if required by regulators. Don't consider translation an afterthought – it's part of your information architecture and should be accounted for in governance and timelines. *(If you ship globally, your translation workflow is part of your content strategy, not just a downstream task.)*
- **Document the Translation Strategy:** In your migration plan (see template), record which approach you've chosen and any specific processes/tools to be used. Having this in writing helps set expectations with management (e.g. "First multilingual release will incur higher translation spend due to restructuring, but subsequent releases will reclaim savings"). Make it a **conscious decision point** rather than an implicit assumption.

Ensuring AI-Readiness of Content

One motivator for structured content is to enable advanced use cases like cognitive search and AI-driven assistance (for example, chatbots using your docs as a knowledge

base). To be **"AI-ready,"** your content must be consistently structured, rich in metadata, and governed for accuracy. Key requirements to address:

- **Fine-Grained, Modular Topics:** Break content into **small, self-contained topics** that each answer one question or complete one procedure. This granularity is what enables snippet-level retrieval for AI. For instance, an AI system using retrieval-augmented generation can pull a single topic as a source to answer a user question. Ensure each topic has a clear heading and that it *encapsulates* a single idea or task. Avoid overly long topics that cover multiple subjects.
- **Structured, Consistent Formatting:** Follow a predictable structure within topics (e.g. a task always has a **Steps** section, a concept might always start with a definition). Use semantic tags (like `<step>`, `<note>`, `<code>`, etc. in DITA) appropriately rather than embedding meaning in formatting. Consistency makes it easier for AI to parse and interpret content. Also, maintain **link integrity and reuse hygiene** – broken links or contradictory reused content can confuse both readers and AI.
- **Robust Metadata on Every Topic:** Tag topics with all relevant metadata attributes (product, topic type, skill level, etc.). This context is invaluable for AI search filters. For example, if a user asks "How do I replace the battery?", the system can filter answers to those tagged with product=XYZ and topic-type=task. Enforce mandatory metadata in the CCMS if possible. Also consider **chunk-level descriptors** – e.g. a short summary or keywords for each topic – to aid AI in quickly determining relevancy.
- **Stable IDs and Versioning:** Each topic (and significant element) should have stable identifiers. This allows precise reference by AI systems or APIs. Version control is equally important: maintain a **single source of truth** for each piece of information. AI or users should only be accessing the latest approved version of content. Having audit trails and version metadata means you can trace what the AI saw and when.
- **Content Quality and Unambiguity:** AI will **amplify** any content issues – if instructions are unclear or inconsistent, AI might produce incorrect or "hallucinated" answers. So, double down on clarity: ensure procedures are **sequential and unambiguous**, avoid insider jargon or assumptions, and keep content up-to-date. If content is properly structured and reviewed for accuracy, AI integrations will yield far better results. *(As the saying goes, "If content is inconsistent, AI scales the inconsistency. Structure is the antidote.")*
- **Platform Capabilities for AI Integration:** Check that your chosen CCMS or content platform supports the technical needs of AI integration. For instance, can it **expose content via APIs** or search indexes for chatbot consumption? Can it enforce the metadata and access controls needed so AI only pulls from the appropriate content sets? If using a solution like *RWS Tridion Docs* combined with **Semantic AI**, ensure you leverage its taxonomy/ontology features to make content relationships explicit (useful for knowledge graphs and semantic search).
- **Security and Access Control for AI:** Implement controls to **segregate sensitive or internal-only content** so that it's not inadvertently used in external-facing AI answers. For example, label and restrict internal draft content or regulatory filings from general search. Monitor AI usage with logs – know which topics are being retrieved frequently and verify they remain accurate. Define policies for what AI is *not* allowed to answer (certain topics might be marked as "refer to a human"). These governance steps ensure AI usage remains safe and on-brand.
- **Iterate Based on AI Feedback:** Once AI tools (like a documentation chatbot or cognitive search engine) are in use, treat their performance as feedback into content management. If the AI struggles to find relevant info, maybe the metadata needs improvement. If it gives inaccurate answers, it might indicate content that

is out of date or not specific enough. Make AI-readiness a continuous consideration in your content lifecycle. *(Ultimately, AI-ready documentation means optimizing for retrieval, traceability, and trust – exactly what structured content is designed to do.)*

CCMS Selection and Proof of Concept (PoC)

Choosing the right **Component Content Management System** is a pivotal decision. The CCMS will be the backbone for authoring, review, version control, and publishing in your structured content environment. Rather than focusing on flashy features, **base your selection on your workflow requirements and integration needs**. Key steps:

- **Define CCMS Requirements:** Document what you need from a CCMS to support your use cases. Consider:
- *Authoring & Review Workflow:* Does it provide a user-friendly DITA authoring interface (or work with editors like oXygen)? How does it handle contributor reviews and approvals (web-based review, commenting, change tracking)?
- *Reuse and Variant Management:* Can it manage content reuse at topic or element level (conrefs, keys) effectively? How about variants for products or regions (profiling/filters)?
- *Integration:* Ensure it can integrate with your **translation management system**, publishing pipelines (DITA-OT, PDF, HTML output), and any single-sign-on or user management needed. Integration with **legacy systems** (like an ALM or support database) or an API for exposing content to a help portal might be important.
- *Scalability & Security:* Enterprise-grade permission controls, version history, and audit trails are a must for many teams. The system should handle your content volume and user count without performance issues.
- *Metadata & Taxonomy:* Look for robust support for metadata: can you define custom metadata fields and enforce them? Does the CCMS offer taxonomy or ontology management (to drive consistent tagging and possibly feed into semantic search)?
- *User Experience:* A system that is too complex might face user adoption issues. The **best CCMS is the one your team will actually use** effectively. So, usability (for both writers and occasional contributors) is key.
- **Consider RWS Content Technologies:** For example, *Tridion Docs* (an RWS CCMS) is a DITA-based CCMS known for enterprise scalability. It supports full DITA 1.3, offers both a web editor (useful for subject matter experts) and integration with desktop XML editors like oXygen, and provides extensive publishing options (PDF, HTML5, etc.). It also has add-ons for taxonomy (Semantic AI) and connections to translation tools.
- **RFP and Vendor Demos:** Conduct a structured RFP (Request for Proposal) process. Provide vendors with specific **use case scenarios** to demonstrate – for instance, "show how an SME would review and comment on a topic" or "show variant filtering for US/EU regulatory differences." Avoid letting a generic demo drive the decision; instead, *you* set the agenda with real-world scenarios. Score each solution on how well it meets each requirement.
- **Proof of Concept (PoC):** After narrowing down to a top choice or two, run a **proof-of-concept** project in a sandbox environment. This is a **no-regrets step** to validate the CCMS in practice:
- **Scope:** Use a representative set of content in the PoC – include a mix of simple and complex documents, perhaps one complete section of your docs. Involve various **roles**: have one or two writers author/edit content, a reviewer or SME perform an

approval cycle, maybe a translator do a sample translation, and an admin or developer test an integration like publishing or single sign-on.

- **Scenarios:** Execute key workflows in the CCMS. For example, **reuse** a topic in two publications and try updating it once; manage a **variant** (such as conditional text for two product models) and publish two outputs; run a **version compare** between two edits; simulate a typical **publish** to your required formats and check output quality; perform a round-trip through the translation connector if available. This hands-on approach will reveal any workflow hiccups or missing features.
- **Timebox and Measure:** Limit the PoC duration (e.g. 2-4 weeks) and avoid scope creep. Before starting, define **success criteria** to evaluate at the end. For instance: *"Authors can perform basic edits and save in under 5 minutes per topic; review comments round-trip without data loss; publishing a full manual takes <10 minutes and output passes quality checks; translation of a sample set can be completed via the integrated workflow,"* etc. Also measure subjective feedback (did users find the interface intuitive?).
- **Evaluate Results:** If the PoC meets (or fails) the criteria, use that evidence to make the final CCMS selection. Document any discovered limitations or needed workarounds – for example, you might find you need an additional tool for PDF customization or a plugin for something. Better to know *before* you buy. The goal is to avoid expensive surprises later.
- **Obtain Stakeholder Buy-in:** Present the PoC findings and your recommended CCMS to stakeholders (management, IT security, etc.) along with the evidence. Emphasize how the chosen solution supports your workflow and governance needs (e.g. "Tridion Docs had the best review workflow and metadata control, which are critical for us"). Getting formal approval at this stage ensures everyone is on board with the tool investment.
- **Plan the Implementation:** Work with the vendor on deploying the CCMS (cloud provisioning or on-prem install). This includes **configuring the system**: setting up user accounts/permissions, defining the information types and metadata fields in the CCMS, installing the DITA Open Toolkit for publishing, integrating the style sheets or templates for output, connecting the translation module, etc. Also plan data migration in detail (how to get your existing content into the CCMS – addressed in the next section). Finally, schedule training sessions for the team on using the new CCMS interface effectively. The smoother the rollout, the quicker your team will start realizing benefits.

Pilot Migration and Workflow Validation

Instead of jumping straight into migrating everything, it's highly recommended to **start with a pilot project**. A pilot serves as a proof-point and learning opportunity on a smaller, safer scale. Here's how to approach it:

- **Select Pilot Content:** Choose a **representative set of content** as the pilot scope. This could be one manual or a subset of each content type you produce. The pilot set should be complex enough to test your processes (include a few challenging pieces like a procedure with lots of steps or a document with many cross-references), but manageable in size (maybe 5-10% of the total content). Often teams pick a **customer-facing guide or a frequently-used knowledge base section** as pilot content – something that will show clear value if improved. Document the selection criteria and which documents/topics are in scope. *(The content inventory and audit from earlier will help identify good pilot candidates.)*
- **Define Pilot Goals and Metrics:** Establish what a "successful pilot" looks like. For example: *"Pilot content migrated to DITA, published without loss of fidelity; 20% reduction in review cycles time for pilot docs; translation of pilot content shows*

X% TM leverage; authors report positive feedback on new tools." Set **baseline measurements** (e.g. how long it took to update/publish those docs before) and target improvements. Typical pilot success metrics include content reuse rate in the pilot (if applicable), turnaround time for an update, initial translation cost vs. previous, and quality measures like fewer errors post-publish. By measuring and hitting these targets, you build a case to proceed.

- **Migrate and Validate Pilot Content:** Perform the actual conversion for the pilot content using the chosen strategy (rewrite/refactor/convert). This is effectively a dry run of your migration process:
- If using conversion tools, run them on the pilot docs and fix any conversion issues.
- If rewriting or refactoring, have the writers create the new topics and maps for the pilot content in the CCMS.
- **Load the content into the CCMS**, ensuring all metadata is applied and links between topics are working.
- **Publish the pilot content** from the CCMS to your outputs (e.g. generate the PDF or WebHelp) and compare it against the legacy version for completeness and formatting. This checks your publishing toolchain and templates.
- Go through a **review cycle** with the pilot content using the new workflow (SMEs reviewing in the CCMS or via its web interface, for example). This tests the review/approval process and identifies if stakeholders are comfortable with it or need more training.
- If applicable, perform the **translation round-trip** on the pilot content (as noted in the translation section) to validate the localization process.
- **Train and Support the Pilot Team:** The people involved in the pilot (writers, reviewers, etc.) should be the first to get training on the new model and tools. Provide close support (maybe daily check-ins or a chat channel for quick help) as they work through the pilot. Capture their feedback—both to improve the process and to gather testimonials on the benefits observed. For example, a tech writer might report *"I can update a topic and publish it in minutes, which used to take a day"* – valuable anecdote for management.
- **Document Issues and Adjust:** Treat the pilot as an experiment. Inevitably, some assumptions will be challenged. Maybe an element of your information model needs tweaking, or the CCMS configuration needs adjustment (e.g. adding a metadata field you overlooked). Perhaps the conversion script needs fine-tuning to handle a certain style of tables. Log all findings and **refine your migration playbook** based on them. It's far better to discover these in a pilot than mid-way through full migration.
- **Demonstrate Pilot Success:** At pilot completion, compile the results and compare to goals. Ideally, you can show a before/after comparison: *"We reduced the review cycle for the pilot doc from 2 weeks to 3 days"*, *"We identified 30% redundant content now consolidated via reuse"*, or *"Quality improved (support tickets on the pilot content dropped in the next release)"*. Present these outcomes to sponsors and the wider team. This showcases the **business value** of the structured approach (faster time-to-market, improved quality, etc.) and builds momentum for the next phases. In essence, the pilot provides the evidence to confidently scale up the migration.

(Key principle: Design for repeatability. Start small, prove the model and workflow, then scale with governance. The pilot is where you prove it.)

Full Migration Execution and Post-Migration Governance

With pilot lessons in hand, you can plan the **rollout to full content migration** and the long-term governance to keep content optimized:

- **Phased Rollout Plan:** It's usually wise to migrate in phases rather than a "big bang." For example, phase 1 might be one department's docs or one product line, phase 2 another, etc. Prioritize content groups by business need (perhaps new product lines first, or areas with most customer impact). Create a migration schedule, e.g. migrate 100 topics per sprint, or one manual per week – whatever is feasible. Continue to apply the chosen **migration approach (rewrite/refactor/convert)** for each batch and use the refined process from the pilot.
- **Content Freeze and Cutover:** Determine if you will enforce a "content freeze" on source documents during migration phases. Often, teams freeze edits on legacy content while it's being converted to avoid double-maintenance. Alternatively, implement a policy that any changes in legacy docs must also be applied in the new CCMS once that content is migrated. Plan for a final **cutover** date when the old system is retired and the CCMS becomes the source of truth for all content. Communicate this to all stakeholders so they know where to find the latest content at any time.
- **Quality Assurance and Validation:** As each chunk of content is migrated, perform **QA checks** similar to the pilot: verify structure, links, media, and output formatting. Also, confirm that **legacy URLs or identifiers** (if important for external links or references) are maintained or properly redirected in the new outputs. It can be helpful to have a checklist for each doc migrated, e.g.: *Topics structured? Metadata applied? Links working? Content approved in CCMS? Published PDF matches original?*
- **User Acceptance Testing:** Particularly for large docs or complex sets, consider having the end-users (or internal SMEs) do a quick review of the first fully migrated deliverables. They may catch subtle issues (like missing context that was in a legacy cover page, or a navigation difference in the output) that you can address early. Gathering user feedback also helps ensure the new format meets their needs. If any critical issues are found, adjust the process and perhaps retrofit corrections to already migrated content.
- **Retirement of Legacy System:** Plan how and when to retire or decommission the old content repository or tools (e.g. old Wiki, file shares, FrameMaker, etc.). Ensure all necessary content has been migrated or archived. Provide access to archived content if needed (some companies keep an read-only archive of the old docs for history, accessible in case of emergency). Once confident, shutting down the old system will encourage full adoption of the new.
- **Post-Migration Governance:** After migration, **governance becomes the focus** to keep content on track. Establish a **governance board or content council** that meets regularly (e.g. monthly or quarterly) to review content health. This group should use the governance rules set earlier to monitor:
- **Content Quality Metrics:** Track things like reuse rate (how many topics are reused across publications), review cycle time (author to publish), support ticket volumes related to documentation, and translation costs. Many CCMSs can generate reports for some of these. If metrics stagnate or worsen, investigate and take action (e.g. additional training or cleanup projects).
- **Adherence to Model and Style:** Periodically audit new content being created. Are authors following the template? Are there any new "rogue" patterns emerging that weren't in the model? Use tools (or peer review) to catch deviations and correct them through coaching or updating the guidelines.

- **Change Control:** Manage how new content requirements are handled. For example, if a new product line is introduced, the council decides how to integrate that into the information model and metadata schema. Avoid ad-hoc additions that circumvent governance.
- **Avoiding Content Debt Recurrence:** Just as importantly, ensure that old habits (like copy-paste duplication or storing info outside the CCMS) don't creep back. Governance is ongoing discipline. Keep the motto: *"We don't want to accumulate content debt again."* If content debt (duplication, inconsistency, outdated info) is seen building up, treat it like a codebase with tech debt – schedule refactoring or cleanup sprints to address it.
- **Continuous Improvement:** Leverage the efficiencies gained to further optimize. For instance, if reuse is saving writers time, repurpose that time for enhancing content quality or creating new content. Review your success metrics after a year; if you overshot targets (say you achieved higher reuse or bigger translation savings than expected), you might set new goals or expand the structured approach to other teams. Keep an eye on **emerging tools or AI features** (maybe your CCMS vendor releases a new semantic search plugin, etc.) that could add value on top of your now-structured content.
- **Celebrate and Communicate Success:** Finally, recognize the effort the team put in and communicate the outcomes. For example, *"In the first year post-migration, we delivered 4 releases with 30% fewer resources, cut translation cost by 25%, and saw customer self-service use increase by 40% thanks to better documentation."* Sharing such wins with upper management and the team helps cement support for maintaining the structured content program. It also underscores that documentation is now a **strategic asset** in the company (ready to support products, customer success, and AI initiatives) – not just a cost center.

By following this checklist, your organization can systematically **plan, track, and execute** a content migration to DITA/CCMS with minimal surprises. The result will be a **scalable, governed, and AI-ready content repository** that brings long-term business value. Use the template below to organize your migration project details and keep everyone on the same page throughout the journey.