Ontology and Thesaurus Integration Best Practices with References

Here are the key integration best practices with references:

1. Separate the domain model from the vocabulary Keep ontology classes distinct from thesaurus concepts. The ontology defines the schema; the thesaurus populates it with terms. This is the foundational principle behind using SKOS alongside OWL.

- Allemang & Hendler, *Semantic Web for the Working Ontologist* (3rd ed.)
- Miles & Bechhofer, SKOS Simple Knowledge Organization System Reference, W3C 2009

2. Use dual-typing judiciously Typing a resource as both `skos:Concept` and an OWL class instance (as you do with `the:maskingFluid a skos:Concept, vad:WatercolorAdditive`) is a recognized bridging pattern but should be applied deliberately. It conflates the concept-as-term with the concept-as-class-instance.

- Isaac & Summers, SKOS Simple Knowledge Organization System Primer, W3C 2009
- van Assem et al., *Converting and Integrating Vocabularies for the Semantic Web*, ISWC 2006

3. Use SKOS mapping properties for cross-vocabulary alignment When bridging independently evolved vocabularies, prefer `skos:exactMatch`, `skos:closeMatch`, `skos:broadMatch`, `skos:narrowMatch`, and `skos:relatedMatch` over OWL axioms like `owl:equivalentClass`. The SKOS mapping properties are lighter weight and don't trigger unintended OWL inferences.

- Miles & Bechhofer, SKOS Reference (above)
- Baker et al., Library Linked Data Incubator Group Final Report, W3C 2011

4. Declare ConceptSchemes explicitly Every `skos:inScheme` reference should point to a declared `skos:ConceptScheme` resource. Undeclared schemes break SKOS validation and make navigation unreliable.

- Miles & Bechhofer, SKOS Reference (above)
- Bechhofer & Miles, SKOS Core Guide, W3C 2005

5. Maintain namespace discipline Separate namespaces for separate concerns — as you do with `vad:` for the ontology and `the:` for the thesaurus — is a best practice. It preserves modularity and makes provenance clear.

- Heath & Bizer, Linked Data: Evolving the Web into a Global Data Space, 2011
- Noy & McGuinness, Ontology Development 101, Stanford 2001

6. Use a separate alignment module for cross-ontology mappings Rather than burdening either ontology with knowledge of the other, declare mappings in a dedicated alignment file that imports both. This preserves the universality of each model.

- Euzenat & Shvaiko, *Ontology Matching* (2nd ed.), Springer 2013
- The Linked Open Vocabularies project demonstrates this pattern at scale

7. Align with established upper ontologies where possible Grounding your ontology in well-known upper models (SKOS, Dublin Core, FOAF, Schema.org — all of which you already use) improves interoperability and reduces the need for custom alignment work later.

- Niles & Pease, *Towards a Standard Upper Ontology*, FOIS 2001
- Your current use of `dc:`, `foaf:`, `schema:`, and `skos:` already reflects this practice well

8. Document design decisions in the ontology itself Use `rdfs:comment`, `skos:scopeNote`, and `skos:editorialNote` to record rationale inline. Your use of `# MERGE NOTE:` comments in CWVA is good practice — formalizing some of those as `rdfs:comment` annotations would make them machine-readable.

- Rector et al., *OWL Pizzas: Practical Experience of Teaching OWL-DL*, EKAW 2004
- Allemang & Hendler (above)

9. Validate regularly against SKOS and OWL profiles Tools like the SKOS Play validator, OOPS! ontology pitfall scanner, and Apache Jena's `riot` can catch issues like undeclared schemes, dangling references, and property typos before they propagate.

Your system already reflects most of these practices — the gaps noted in the review (undeclared pigment scheme, the `skos:altfLabel` typo, the `DrawingForm` naming inconsistency) are precisely the kind of thing routine validation catches early.