RO-Crate Use Case Template

Thanks Claude!

This structure would let the RO-Crate site filter/sort use cases by profile or entity type (useful for someone asking "who's using Workflow Run Crate for provenance?") rather than only by domain, task-category, or role as it does now.

Name: [Application/project name]

Task *(what the use case is trying to accomplish)*
One or two sentences describing the concrete goal. For example - instead of "manages workflows" say "packaging workflow configuration, logs, and outputs so an experiment can be re-run or audited later"

RO-Crate features used
A short list naming the specific mechanisms, e.g.:

- Profile(s): Workflow RO-Crate, Workflow Run Crate, Process/Provenance Run Crate, ELN, custom profile
- Core entities: `ComputationalWorkflow`, `CreateAction`/`ControlAction`, `SoftwareApplication`, `FormalParameter`
- Context/vocabulary: Bioschemas, schema.org extensions, custom JSON-LD context terms
- Packaging: zipped crate, crate-in-repository, crate-as-API-response

How the features are applied
A short description of the mechanism — where the crate is generated (at authoring time, at run time, at export), what triggers its creation, what consumes it (a registry, a viewer, another tool), and what the crate enables that wouldn't otherwise be possible (e.g. re-execution, provenance comparison, cross-repository indexing).

Outcome / benefit
One sentence on what using RO-Crate in this way achieves for the user or system, distinct from the task description above (task = intent, outcome = result).

Links: [tool/repo] · [example crate, if public] · [profile spec, if applicable]

Worked example (Galaxy, rewritten in this template):

- **Task:** Allow a researcher to export a completed workflow run — the workflow definition, inputs, outputs, and execution metadata — as a self-contained, shareable object.

- **RO-Crate features used:** Workflow RO-Crate profile for the workflow definition; Workflow Run Crate profile for execution provenance; `ComputationalWorkflow` and `CreateAction` entities.
- **How applied:** On workflow invocation completion, Galaxy serialises the workflow's CWL/Galaxy-native definition plus a provenance graph of the run (tool versions, parameters, dataset lineage) into a crate, exportable directly from the Galaxy history panel or via API.
- **Outcome:** The exported crate can be imported into WorkflowHub or re-run in another Galaxy instance with its provenance intact, rather than losing run context on export.

"The practical motivation for this work is that many scientists can describe the analysis they want but may struggle to construct the corresponding workflow in a particular workflow system. By retrieving and composing trusted workflows and components, and generating new steps only where no trusted equivalent exists, the project aims to make reliable workflows more accessible. It also investigates whether grounding in fixed, trusted workflows improves consistency, so that the same or similar requests yield equivalent workflows more often. The impact of this consistency is that workflows become easier, reproduce, compare, and reuse. Finally, method-preserving translation may reduce system lock-in by allowing trusted workflows to be re-expressed across systems while preserving the method they encode." reduce by 30%

Thinking about condensing scientific workflow management concepts by 30%

Here's a version reduced by about 30%:

The practical motivation is that many scientists can describe the analysis they want but struggle to construct the corresponding workflow in a particular system. By retrieving and composing trusted workflows and components, and generating new steps only where no trusted equivalent exists, the project aims to make reliable workflows more accessible. It also investigates whether grounding in fixed, trusted workflows improves consistency, so that similar requests yield equivalent workflows more often — making workflows easier to reproduce, compare, and reuse. Finally, method-preserving translation may reduce system lock-in by allowing trusted workflows to be re-expressed across systems while preserving the method they encode.

https://www.researchobject.org/ro-crate/use_cases can you propose a template for use cases that better reports what the use case task is trying to do, what features of RO-crate are being used and how those features are being applied

Identified structural gaps and architected comprehensive template framework

The current entries are short one- or two-sentence descriptions of what each tool or project is — they name the application but rarely say what the underlying task was, which specific RO-Crate features (profiles, entities, context terms) are used, or how those features are applied in practice. That makes it hard to compare use cases functionally.

Here's a template that separates those three dimensions explicitly:

Use Case Template

Name: [Application/project name]

Task *(what the use case is trying to accomplish)*
One or two sentences describing the concrete goal — e.g. "packaging workflow configuration, logs, and outputs so an experiment can be re-run or audited later," not just "manages workflows."

RO-Crate features used
A short list naming the specific mechanisms, e.g.:

- Profile(s): Workflow RO-Crate, Workflow Run Crate, Process/Provenance Run Crate, ELN, custom profile
- Core entities: `ComputationalWorkflow`, `CreateAction`/`ControlAction`, `SoftwareApplication`, `FormalParameter`
- Context/vocabulary: Bioschemas, schema.org extensions, custom JSON-LD context terms
- Packaging: zipped crate, crate-in-repository, crate-as-API-response

How the features are applied
A short description of the mechanism — where the crate is generated (at authoring time, at run time, at export), what triggers its creation, what consumes it (a registry, a viewer, another tool), and what the crate enables that wouldn't otherwise be possible (e.g. re-execution, provenance comparison, cross-repository indexing).

Outcome / benefit
One sentence on what using RO-Crate in this way achieves for the user or system, distinct from the task description above (task = intent, outcome = result).

Links: [tool/repo] · [example crate, if public] · [profile spec, if applicable]

Worked example (Galaxy, rewritten in this template):

- **Task:** Allow a researcher to export a completed workflow run — the workflow definition, inputs, outputs, and execution metadata — as a self-contained, shareable object.
- **RO-Crate features used:** Workflow RO-Crate profile for the workflow definition; Workflow Run Crate profile for execution provenance; `ComputationalWorkflow` and `CreateAction` entities.

- **How applied:** On workflow invocation completion, Galaxy serialises the workflow's CWL/Galaxy-native definition plus a provenance graph of the run (tool versions, parameters, dataset lineage) into a crate, exportable directly from the Galaxy history panel or via API.
- **Outcome:** The exported crate can be imported into WorkflowHub or re-run in another Galaxy instance with its provenance intact, rather than losing run context on export.

Synthesized repository structure and drafted proposal documentation

Proposal note formatted for a GitHub issue on `ResearchObject/ro-crate` to paste directly as an issue, or adapt it into a PR description if you draft the template file yourself first. GitHub

---

Title: Extend use case entries to report task, RO-Crate features used, and application

Body:

Currently, use case entries on the Use Cases page are short prose descriptions of the tool or project (e.g. what it is, what domain it serves), and are browsable by domain, task, and role. What's not consistently captured is a structured breakdown of: (a) the specific task the use case addresses, (b) which RO-Crate features are used (profile, entity types, context terms), and (c) how those features are applied in the implementation. This makes it hard to compare use cases functionally — e.g. to find "who uses Workflow Run Crate for provenance capture" rather than only by domain or tool name.

Proposal: Add three optional structured fields to the use case front matter / page template, alongside the existing prose description:

yaml

```yaml
task: >
  Short statement of the concrete goal (e.g. "package workflow configuration,
  logs, and outputs for re-run or audit").
ro_crate_features:
  profile: [Workflow RO-Crate, Workflow Run Crate]
  entities: [ComputationalWorkflow, CreateAction, FormalParameter]
```

```
  context: [Bioschemas]
application: >
  Where/when the crate is generated, what consumes it, and what it enables
  that wouldn't otherwise be possible.
```

These would render as an optional supplementary block under the existing description, so it's backward-compatible with entries that don't fill them in. Over time this would let the `browse by` mechanism be extended to filter by profile or entity type as well as domain/task/role.

Devised markdown documentation for WorkflowHub and Galaxy integration

Two worked examples, ready to drop into the GitHub issue as concrete before/after illustrations.

WorkflowHub

Current entry:

> WorkflowHub imports and exports Workflow RO-Crates, using it as an exchange format. They are a specialization of RO-Crate for packaging an executable workflow with all necessary documentation. It is aligned with, and intends to strictly extend, the more general Bioschemas ComputationalWorkflow profile.

Proposed structured entry:

yaml

```
name: WorkflowHub
task: >
  Allow researchers to register, discover, and cite computational workflows
  in a way that preserves enough documentation and metadata for others to
  understand, reuse, or re-run them.
ro_crate_features:
  profile: [Workflow RO-Crate]
  entities: [ComputationalWorkflow, SoftwareApplication, FormalParameter]
```

```
  context: [Bioschemas ComputationalWorkflow]
application: >
  On upload, WorkflowHub packages a workflow's definition file(s), README,
  license, and author/institution metadata into a Workflow RO-Crate; on
  registration from an external source (e.g. a GitHub repo), it extracts and
  validates the crate rather than generating one from scratch. The crate is
  the canonical exchange format for both import and export, so a workflow
  registered elsewhere can be pulled into WorkflowHub, and one registered on
  WorkflowHub can be exported for use in another compliant system.
outcome: >
  Workflows gain a portable, machine-readable description that survives
  transfer between registries and tools, rather than being tied to
  WorkflowHub's own database schema.
```

Galaxy

Current entry (paraphrased from the site, since Galaxy's current listing is brief):

> Galaxy is a workflow platform for accessible, reproducible, and transparent computational research.

Proposed structured entry:

yaml

```
name: Galaxy
task: >
  Let a researcher export a completed workflow run — the workflow
  definition, inputs, outputs, and execution metadata — as a
  self-contained, shareable object.
```

```
ro_crate_features:
  profile: [Workflow RO-Crate, Workflow Run Crate]
  entities: [ComputationalWorkflow, CreateAction, FormalParameter]
  context: [Bioschemas ComputationalWorkflow]
application: >
  On invocation completion, Galaxy serialises the workflow's native
  definition together with a provenance graph of the run (tool versions,
  parameters, dataset lineage) into a crate. This is exportable directly
  from the Galaxy history panel or via the API, generated at export time
  rather than authored manually.
outcome: >
  The exported crate can be imported into WorkflowHub or re-run in another
  Galaxy instance with its provenance intact, rather than losing run
  context on export.
```

Both are written so they'd drop straight into the front matter of a use case page under the existing `page_id` pattern, with `task`, `ro_crate_features`, `application`, and `outcome` as the new optional fields alongside the current prose description.

Filtering UI

How the "browse by" filtering UI might incorporate `ro_crate_features.profile` as a new browsing facet, since that's the part most likely to prompt discussion from maintainers.

Here's a plain-text sketch of how the "browse by" facet could be extended, following the same pattern as the existing domain/task/role browsing.

Current browsing structure (per the site): entries are filterable by *domain*, *task*, and *role*. This is likely implemented as tags/collections in the front matter of each use case page, aggregated into filter lists on the main use_cases page.

Proposed addition — a fourth facet: RO-Crate feature

Browse by: [Domain ▾] [Task ▾] [Role ▾] [RO-Crate feature ▾]

RO-Crate feature ▾

Profile

- ☐ Workflow RO-Crate
- ☐ Workflow Run Crate
- ☐ Process Run Crate / Provenance Run Crate
- ☐ ELN (Electronic Lab Notebook)
- ☐ (custom / unspecified)

Entity type

- ☐ ComputationalWorkflow
- ☐ CreateAction
- ☐ FormalParameter
- ☐ SoftwareApplication

Context / vocabulary

- ☐ Bioschemas
- ☐ (other extensions)

Mechanics:

- Populated the same way domain/task/role are now — pulled from the `ro_crate_features` front-matter block proposed earlier, so no new backend beyond what already aggregates the existing tags.
- Selecting a profile (e.g. "Workflow Run Crate") filters the use case list to entries that declare it, same interaction pattern as selecting a domain today.
- Entries without the field simply don't appear under that facet, so it degrades gracefully for older, unmigrated entries rather than breaking the page.

Why it's useful beyond browsing: this facet is the one most likely to matter to someone building or extending a tool (e.g. "who else has implemented Workflow Run Crate consumption")

as opposed to domain/task/role, which serve people looking for use cases in their own research area. It's a different audience — implementers rather than researchers scoping out RO-Crate for their field — so it's worth pitching to maintainers as filling a gap rather than duplicating existing facets.

Automated population of template

How might we automatically populate the template