Breaking Down Large Tickets

- Introduction
- Hints / Steps for Breaking Things Down
    - Warning Signs / Red Flags
- Case Study: Improve Extractor Catalog
    - Epic: Improve Extractor Catalog
    - Original Story Text
    - Original work discussed/requested for ticket
    - Red Flags encountered
    - Suggested Split
- Case Study: View Extractor runtime synopsis and granular metrics
    - Epic: Extractor Job Management
    - Original Story Text
    - Original work discussed/requested for ticket
    - Red Flags encountered
    - Suggested Split

Introduction

I have previously written a lightning talk introducing some helpful patterns for writing JIRA tickets. It introduces some terminology, as well as what should be expected from a fully-detailed ticket.

The purpose of this document is to supplement those definitions with examples of breaking down larger work items to increase transparency into the process and maintain forward momentum.

Hints / Steps for Breaking Things Down

1. Read the Epic carefully - try to visualize the full picture of what it would need to provide. Are there steps needed that are not covered by the existing tickets?
2. Read an issue that you suspect can be broken down - watch out for any Red Flags (listed below)
3. If you identify a Red Flag, try to think of how to make that part of the issue standalone - NOTE: it is ok for issues to form a hierarchy or to depend on one another
4. Write a new Story based on the subset of work you've identified, and then link it back to the original ticket
5. Check the original ticket again to remove any text / feature that is now covered by the new ticket(s)

NOTE: If you do manage to tease out one or more new issues, link it back to the original by clicking Link Issues and choosing "split from" or "split to"

Warning Signs / Red Flags

The following may not ***always*** be a bad thing, but they can indicate an issue that can be further broken down

- **Singleton Epic** - an epic with only a single Story is either too small to be an epic or too large to be a story
- **Not enough details** - very short ticket text with a very high estimate makes us wonder where those extra points are coming from
- **Commas / Conjunctions** (e.g. and / or / but) - can indicate a pause or change in focus
- **"also" / "as well" / "includes" / "including"** - extraneous things that can usually be split into separate tickets sometimes use these types of modifiers
- **a single Story requires changes to multiple repositories** - typically Stories can be broken down such that they only need to change 1 repo at a time (if they can't, this may indicate coupling or issues maintaining the "separation of concerns")
- **A clear disconnect or confusion between what was asked for in the ticket vs what was suggested by leadership** - this indicates that one or more people did not understand the full scope of the work requested
- ... and many more

Case Study: Improve Extractor Catalog

See https://digitial-product-engineering.atlassian.net/browse/SIMPL-115

Epic: Improve Extractor Catalog

```
Description
Catalog of extractors that provide extractor name, description, version, authors, rating, comments.


Acceptance Criteria - SIMPL
None
```

Original Story Text

```
Description
Catalog of extractors that provide extractor name, description, version, authors, rating, comments.


Acceptance Criteria - SIMPL
None
```

Original work discussed/requested for ticket

- Expand admin-only Extractor Catalog to make it public
- Add additional information to this view (e.g. version, rating, comments, etc)
- NOTE: While there were lots of unanswered questions, this discussion did match up fairly well with the original ticket text

Red Flags encountered

- Singleton Epic (Epic had the exact same text as this issue, verbatim)
- Not enough details (it's not even a complete sentence)
- Commas (first four items with commas are closely related, but the other two are not)

Suggested Split

- SIMPL-190: Expose Extractor Catalog to authenticated users
- more to come...

Case Study: View Extractor runtime synopsis and granular metrics

See https://digitial-product-engineering.atlassian.net/browse/SIMPL-41

Epic: Extractor Job Management

```
Description
Dashboard of job operations where a space admin can view number of jobs running, waiting, and failed. A user
can also view the history of a job and see its log files, status events, messages, resource allocation (memory
usage, cpu usage).


Acceptance Criteria - SIMPL
None
```

Original Story Text

```
Description
As a Developer, I want to be able to gain greater compute insights of my extractors in real time to better
optimize resource allocation and scheduling so that I can better profile the behavior of my extractor

Syngenta Comments:

NCSA Comments:
1) UI/UX and backend dev: 2 weeks
2) API dev: 1 week

Min Estimate (Weeks): 3
Max Estimate (Weeks): 3


Acceptance Criteria - SIMPL
None
```

NOTE: this is not a full Story. We only know the who/what/why but nothing about the "how" or "where". This can cause the estimate of "when" (e.g. the story estimate) to be significantly skewed.

Original work discussed/requested for ticket

- Expand Clowder UI to group Extractions not only by extractor, but also by an ID that is unique per-execution
- NOTE: This did not seem to clearly match up with any of the text in the original ticket

Red Flags encountered

- Short text with a high estimate - ticket contents should justify the estimate
- Disconnect between ticket text and what was asked for
- Required changes to multiple repos

Suggested Split

- SIMPL-191: Clowder should generate a unique jobId for related Extraction messages
- SIMPL-192: pyClowder should pass job_id back during status updates
- SIMPL-193: Clowder UI Extractions list should be further sorted by job_id
- SIMPL-194: pyClowder should include job_id for all logs to stdout/stderr
- NOTE: It is unclear if additional work will be needed, as ticket text grossly mismatched the work requested. I expect that future tickets will be needed to connect the last few dots there.