Rules 3.0

- Severity, Confidence, Effort
  - Similarly to the ILW triage methdology we should try to estimate the full importance of a result using these three gauges.
  - Severity is much like before, but complemented with
    - Confidence: How certain is the rule that this particular result is correct?
    - Effort: How much work would be required to fix this result? The default here should likely be conservative, and only in cases where we can reasonably say that it's lower should we do so.
- Rules should have a constant set of string templates instead of building strings and setting them using e.g. the ResultBuilder. Allowing dynamic templates like in Rules 2.0 makes localization more difficult.
  - The Result can then have a set of pointers to templates.
  - This can also help Rules that use TypedCollectionResult attributes, as with this solution they could then point to a template as a basis for how to display each entry in the list.
    - We can also probably make returning collections better in more ways, any more ideas would be appreciated.
- Results should be descriptive of more than just the top problem in a flight recording. Currently some rules point to the top offender, and some point out all offenders. We should align the rules so that for any case where there is more than one offender the result should describe all of them. By not doing this we are forcing a user to resolve one issue and as soon as they've done so they'll just see the second worst offender instead of having a chance at fixing both directly.
- Properties of results should include
  - Preferred visualisation (e.g. for stack trace aggregates)
- Small change to remove the need for wrapping using a future, cancellation can instead be implemented using a Supplier<Boolean>.
- EventAvailability logic is too rigid. Currently it only does a boolean AND operation on all given EventAvailabilities (i.e. pair of Event Type and Availability).
  - We can either reduce the need for this rule by deciding that a rule should only rely on one specific type of event, not check against different GC implementations etc.
  - OR we can implement more complex operators in the logic, so that a rule is evaluated if either G1 events are present OR ZGC events, to give an example.
- We should add a ThreadActivity type as a default JMC type. This should indicate that a result is related to activity on a set of pairs of Thread & Time. This will make it significantly easier to complement the JMC UI with data from the Rules.
- There should be global context / settings for which the rules are evaluated. For example, it can be important to know what is important to prioritise for a recording, for example throughput vs pause time vs memory vs cpu.
- We should split up the description of what was found and the recommendations of next steps.