# Approaches for handling Product Options

## Table of Contents



## Document Info

| Attribute | Value |
|---|---|
| Status | DRAFT v0.2 (2021-02) |
| Target Version | 1.3x |

# Overview

The document outlines an MVP approach to product option support within Waltz. There are three high-level candidates for approaching add option support to Waltz. They are:

- Use **measurables**, this approach has already been used with some success. However, numerous deficiencies would need to be addressed to fully support the requirements.
- Use **applications**, enrich the application entity, adding features needed to support product option requirements.
- Add a **new core entity** to Waltz which directly represents product options.

We will outline the high-level requirements and assess them against the three candidate approaches.

# MVP Recommendations

- Product Options remain as a measurable taxonomy
- Hosting Venue is introduced as a measurable taxonomy
- Product to Venue is mapped via generic entity relationships
  - Introduce relationship kinds for: `BUY | SELL | HOLD | PENDING 1-3yrs | DECOMM 1-3 yrs | etc`
  - Entity Relationships get boosted to 1st class entities (allowing bookmarks/assessments/change initiative relations/etc)
- 'Master' rating for a product is an assessment rating on the measurable
- Application alignment remains unchanged (ratings against product taxonomy)

## Benefits

- ✅ Quicker 'time to market'
- ✅ Adds improvements to entity relationships which will benefit other use cases (i.e. GDPR, Process/Programme)
  - ✅ Good reason to revisit the spindle functionality/ux
- ✅ Builds on existing Waltz functionality
  - ✅ Existing bulk loaders can be quickly adapted for ad-hoc ingestion of data
  - ✅ Entity Relationships have been in active use for several years without major issues

## Challenges/Drawbacks

- ❌ Product roadmap visualisation, better handled outside of Waltz initially.
- ❌ Spindle not seen as best presentation mechanism.
  - ✅ This can be mitigated by revisiting the UX

- Database level integration harder due to generic structures
  - This can be mitigated by providing API access

| | |
|---|---|
| **NOTE** | The remainder of this document discusses differing approaches to meeting the requirements and is included for reference purposes. |

# Requirements

## Product Hierarchy

*Technical products* are expressed as a **hierarchy** of the form


The first two levels should be considered *abstract* and should not be used for explicit mapping purposes. We should only allow applications to rely on *Product Options* directly. When viewing higher level elements (*Family* & *Product*) we would roll-up the lower level mappings.

**WARNING**

The current list product options seems to cover a lot of variations, some are listed below:

- Some are analogous to application instances (e.g. *AlfrescoECM*)
- … others are more like generic applications (e.g. *Tomcat*)
- … some are similar to offerings such as helpdesks (e.g. *Business Application Support*)
- … and some break down to level 4 elements which seem more like capabilities (e.g. *Data Lake / Consume Data*)

### Product Hierarchy Assessment

*Table 1. Approaches to modelling product hierarchies*

| Measurable Approach | Application Approach | New Entity Approach |
|---|---|---|
| Simple, hierarchies and 'abstract' nodes are built into measurables | Complex, applications are not hierarchical.<br><br>Hierarchy would have to be imposed as another attribute (similar to owning Organisational Unit).<br><br>The wide scope of what constitutes an Option also makes things problematic as Applications would lose their cohesiveness in Waltz.<br><br> For this reason we believe the MVP should discount this approach. | New entity, therefore free to adopt hierarchies. However, it is a **significant** effort to extend core Waltz entities with full Waltz capability support.<br><br>Similar to the application approach, the lack of clarity around the entity would likely complicate the Waltz data model for little gain.<br><br> For this reason we believe the MVP should discount this approach. |

# Product Lifecycle Statuses and Roadmap

*Product options* should have a **current state** lifecycle rating indicating whether the option is currently viewed as a strategic, maintained or non-strategic option.

| | |
|---|---|
| **NOTE** | Can *Product Families* and *Products* also have lifecycle ratings, or only at the option level ? |

| | |
|---|---|
| **WARNING** | It is not clear what we *do* with this information once it is held within Waltz. Do we intend to use it to cross-check application level mappings? Is it only for managing the taxonomy itself ? |

## Option 1: Assessment Ratings

 Current state can simply be expressed via *Assessment Ratings*. Need to ensure assessment only applicable for specific measurable categories.

| | |
|---|---|
| **NOTE** | Measurables do not currently support assessments. However, it would not be a large task to extends support to them.<br><br> We would need to consider how to ensure measurables see only assessment definitions applicable to their category. |

## Option 2: Dedicated Field on Measurable

 Alternatively we could introduce a specific field on the *Measurable* entity, however what would it mean in the context of a *Business Function* or *GDPR Item* ?

# Product Categorization (Measurable to Measurables)

A limitation of the product taxonomy is when we wish to classify products against multiple categories (i.e. Postgres can be both a document database and a relational database). It has been proposed that we allow the product taxonomy items to use other measurables to describe themselves against other taxonomies, e.g. via a technical function taxonomy.

## Option 1: Entity Relationships

 Use the spindle functionality to declare relationships between measurables.

 However, the entity relationship table is very generic and can be difficult to use as a basis for any bespoke extensions. It's also more difficult to extract information from it via queries. Therefore extraction may need to be preformed via API mechanisms.

## Option 2: Generalise Measurable Ratings

By using measurable ratings we can align product options to other taxonomies with a rating.

 Currently, only applications support measurable ratings.

However, we do have other use-cases for allowing entities other than applications to utilise measurable ratings. For example, data flows could enriched by linking to taxonomies such as region and (financial) product.

| | |
|---|---|
| **WARNING** | We would likely need to rework measurable categories within Waltz. It is unlikely a single rating scheme would be able to cover usages of a taxonomy from both a product and application (and potentially other entity) perspective.<br><br>It is likely we would need to split the measurable categories into the raw taxonomies (the hierarachy of items) and instances of that taxonomy which bind entity classes, categories and rating schemes. This would allow a single measurable category (i.e. Region) to be used in multiple category instances with differing rating schemes across potentially different entities (apps, measurables, flows etc). |

```
HOSTING VENUE > PRODUCT OPTION (Rating)
-----------------------------------------
GCP > Databases / Postgres (Buy)
Fabric On Prem > Databases / Postgres (Hold)
Azure > Databases / Postgres (Sell)
AWS > Databases / Postgres (Pending 1-3yrs)
```

is equivalent to ?

```
PRODUCT OPTION > HOSTING VENUE (Rating)
-----------------------------------------
Databases / Postgres > GCP (Buy)
Databases / Postgres > Fabric On Prem (Hold)
Databases / Postgres > Azure (Sell)
Databases / Postgres > AWS (Pending 1-3yrs)
```

as:

```
Databases / Postgres > GCP (Assessment: Sell)
```

would be inconsistent with the first 'fact'

Using this we can extend the constraining assessment functionality to prevent a product with an overall rating of 'Hold' being marked as 'Buy' in any hosting venue.

Measurable Rating Planned decomm feature takes care of product removals.

| | |
|---|---|
| **NOTE** | Problems arise when mapping applications to product as they cannot express which venue they are using the product in. |

## Option 3: Embed hosting variants into product hierarachy

```
|- Databases
   |- Postgres
      |- Postgres GCP  (Assessment: Buy)
      |- Postgres Fabric On Prem (Assessment: Hold)
      |- Postgres Azure (Assessment: Sell)
      |- Postgres AWS (Pending 1-3yrs)
```

 Nice and easy, but pollutes model with hosting aspects. Good excuse to improve the taxonomy management features in Waltz. Difficulties arise because taxonomy is now a hybrid (part sourced from product reference store and partly hosting variations provided by Waltz)

 Application alignment becomes much easier.

# Application to Product Alignment

The current approach for app to product alignment in Waltz uses measurable ratings with a custom rating scheme which defines if the strategic rating and a *underpins* or *consumes* relationship to the product.

## Current state Product Alignment

 Current measurable ratings covers this requirement.

## Future state

 For indicating that an application will no longer be using a product Waltz can already support this via `measurable_rating_planned_decommission` entries.

| NOTE | Unlike the app/fn decomm feature, there is no need for replacement apps as changes are focussed on changes to individual applications. |
|---|---|

### Option 1: Simple rating scheme extension

 Add a rating scheme item similar to `CONSUME - PLANNED 1-3yr`. This requires no additional dev work.

### Option 2: Extends Measurable Ratings Model

 To express future changes to an apps relationship to the product taxonomy then a new entity would be required.

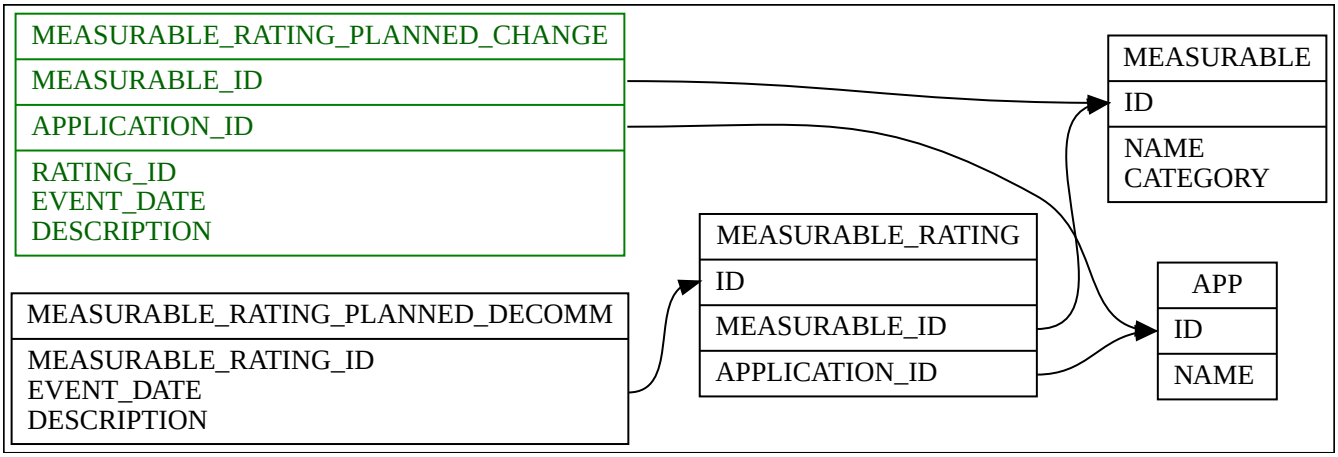


*Figure 1. New Entities to support future state measurable ratings*

### Option 3: Extend Measurable Rating attributes

Add an effective date and entity lifecycle field to measurable ratings.

 However, this would not allow rating changes and complicates usage of the rating tables as all queries must be aware that draft records are included.

# Product to Change Initiative Relationships

 This is already supported via the related viewpoints (spindle) functionality. This utilises the generic `entity_relationship` framework of Waltz.