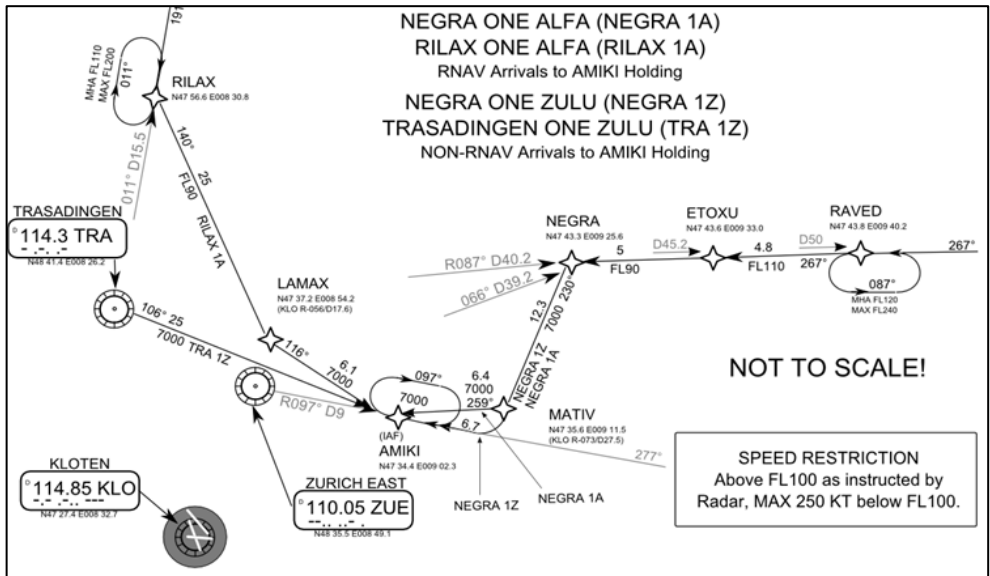


Joining The Dots



In this, the 3rd episode of a series of articles exploring the principles of ontological modelling, the Ontology Working Group (OWG) builds on previous episodes, introducing some new terms from the Unified Foundation Ontology (UFO) and expanding further on the worked example relating to flights and flight planning.

With these articles, the OWG aims to make the topic more accessible through a series of connected newsletter articles. Each episode expands on ontological aspects and by the end, will hopefully provide the reader with enough knowledge to read and contribute to models in the future.

What Is an Ontology?

As we mentioned in earlier articles, an ontology is not some esoteric formalisation, which is nice to have, but is essential to people's day-to-day work on real systems.

When we build a system, we are modelling our world and this involves developing concepts, terms and their relationships – this is called an ontology, or more formally: **“an explicit specification of a conceptualization”** [1].

If not explicitly specified, establishing a shared understanding of interrelated concepts will be much harder and miscommunication may be more common. For the system to operate correctly, any model used to support a part of the lifecycle of a system needs to be correct and consistent. Hopefully the concepts in the model are defined in writing, but at worst, meaning can just be held in the minds of the designers. An ontology also serves as a contract of meaning between stakeholders, who make what are known as 'ontological commitments'. That is, they commit to use vocabularies in a way that are consistent with the ontology's theory. [1]

Where Did We Get To? – A Recap

We are now on our 3rd episode (1st in May 2025, and 2nd in Oct 2025). This article may still be of interest if you have not read the previous two, but you will get a lot more value out of it if you read them first.

To recap on the earlier articles, the OWG uses the **Unified Foundation Ontology (UFO)** [2]. This is tailored for ontological modelling and implemented in **OntoUML** [3] (a customisation of the Unified Modelling Language, UML [4], with associated software support via application plugins). The following notation was previously introduced.

- We use boxes and connecting lines (more formally **Classes** and **Relationships** using the modelling terminology, which are highlighted in this section in **bold**) to represent concepts in our worldview. For readability, classes mentioned in the text of this article are shown in **bold purple**.
- Ontological aspects are expressed as **Stereotypes** (the part of the class definition with Guillemets like this: «role»). **Stereotypes** mentioned in the text in this article are in **bold blue**.
- Arrows with a closed white triangle are **Generalizations**, or in other words, one 'thing' subtypes the more general 'thing' that is being pointed to (for example, a **Cessna** is a subtype of **Aircraft** and **Aircraft** is a **Generalization** of **Cessna**)
- An open arrow ➤ shows an **Association** between things. For example, in the phrase "Pilot 'flies' Aircraft", the concept 'flies' would be modelled as an **Association**. Note that **Associations** also have **Stereotypes**.
- **Association ends** are the numerical expressions at the end of each association, such as **0..*** and **1..*** (read as *zero or more* and *one or more* respectively). These show **Multiplicity** – the range of the number of each type of thing that can be related. This is used to help express constraints, such as a **Pilot** can be trained to fly one or more **Aircraft**, but a given **Aircraft** can usually be flown by, at most, two **Pilots** at any one time.

In the last episode, we covered the important concept of identity principles: these form sets of individuating characteristics that, in combination, give us the ability to separate one individual from another (individuate, more formally).

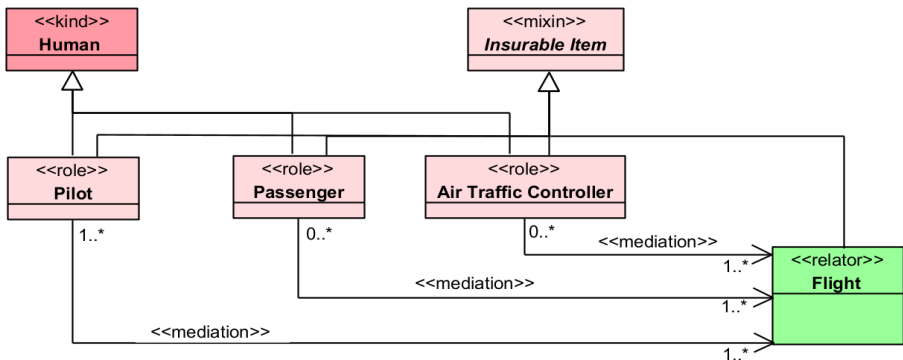
We have also covered some core UFO concepts, expressed as **Stereotypes**:

- **Kinds** reflect the objects we find around us like people, aircraft and dogs that share a common identity principle.
- **Roles** and **Phases** reflect things that **Kinds** might be during their lifetime.
- **Mixins** are used to describe shared properties of objects that do not all belong to the same **Kind** and so can span multiple **Kinds** that do not share the same identity principles.

We used an example model of a **Flight** to help explain the concepts and we'll further expand this model in this episode.

Our Story So Far

We've developed our story over the last two episodes to arrive at the following figure:



There are a lot of lines at the top of the diagram, but they are simply saying that:

Pilot, **Passenger** and **Air Traffic Controller** are roles of **Human**

Pilot, **Passenger**, **Air Traffic Controller** and **Flight** are also all specialisations of **Insurable Item**, even though they do not share a common identity principle. This is represented using the **Mixin** stereotype to indicate that **Insurable Item** can apply to a 'mix' of different subtypes. Note also that **Insurable Item** is written in italics, which is the UML convention to indicate a type that cannot be directly instantiated, known as an abstract **Class**. Abstract classes can only be instantiated through one of their subtypes.

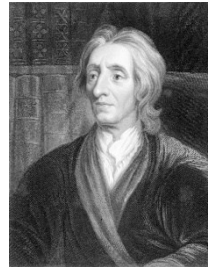
The **Multiplicity** tells us some interesting things too. A **Flight** must have a least one **Pilot** (1..*), it does not always need to have an **Air Traffic Controller**, and it does not necessarily need to have a **Passenger** (0..*), although it might have many. It is also the case that a **Passenger** can only exist if it is associated with at least one flight (no flight means no **Passengers**). Note also that this model will be over constrained for some scenarios because, for example, the minimum of 1 **Pilot** means that unpowered flights by drones cannot be modelled. It could also be under constrained for a model of commercial flights, which usually must have a copilot.

Making Things Count

Kinds, **Roles** and **Phases** are all examples of concepts that allows us to individuate and count their instances. These concepts have the higher-order categorisation of **Sortal** (higher-order categorisation will be coloured **bold green** here). We can thank John Locke for coining the term in 1690! There are a couple of other important higher-level categorisations of UFO concepts that we will introduce later.

Examples of **Sortals** are: **People**, **Runways**, **Computers**, **Locks**, **Aircraft** and **Companies**. **Roles** and **Phases** are **Sortals** too – we can certainly individuate and count, for example, the number of **Broken Seats** (a phase of **Seat**) and **Students** (a role of a **Human**).

If you can't individuate and count instances of a class, it is a **Non-Sortal**. Examples of **NonSortals** are generally more abstract, such as adjectives like **Loud**, substances like **Metal** or **Fuel**, and characterising qualities like **Redness**.



Are we saying we can't count **Fuel** and **Metal**? That would be a problem if we want to ensure an aircraft made of **Metal** has sufficient **Fuel** to reach its destination! We can count **Metal Fuselages** and the **Tanks** that hold the **Fuel**, but if we consume some **Fuel** then what remains is still referred to in the singular as the **Fuel**. In such cases, instead of counting we have to measure amounts of the substance. To solve this problem, OntoUML provides a **Quantity** stereotype that we can use. This provides identity and so is also a **Sortal**.

Close Encounter of the Subkind

We've said that a **Kind** represents the things in our world concepts that provide an identity principle for their instances and do not require a relational dependency. Because **Kinds** are **Sortals**, we can individuate the instances of that particular **Kind**. There are, however, often many things that we consider to be different, but are all readily identifiable as belonging to a common **Kind**. For example, a **Boeing 737 Transport Aircraft**, **Airbus 320 Transport Aircraft** and a **Bombardier Dash 8 Transport Aircraft** are all **Aircraft** and share the same identity principle.

In these circumstances, it is useful, not least for convenience (avoiding repetition for example), to indicate that a class has an overarching identity provider. We therefore introduce **Subkind** as a new stereotype that allows us to build classic taxonomies of things. **Boeing 737** can then be a **Subkind** of **Boeing Transport Aircraft**, which in turn can be a **Subkind** of the **Kind Aircraft**.

Rigidity

We have touched on the idea of **Roles** and **Phases** being temporal properties of a particular **Kind**, an example might be an **Airborne Aircraft** being a **Phase** of the **Kind Aircraft**. The **Aircraft**, being a **Kind**, will remain an **Aircraft** throughout its existence, so like all other **Kinds** and **Subkinds**, it is a **Rigid Sortal** – whereby, 'rigid' should be understood to refer to properties or classifications that are considered essential and unchanging for an entity across all possible situations and times.

By contrast, if a concept includes temporal properties, that is, properties that exist only for part of the thing's lifetime, then that is called **AntiRigid**. So, **Airborne Aircraft**, being a **Phase**, is **AntiRigid**. In fact, because individual **Airborne Aircraft** can be distinguished and share identity principles, they are also a **Sortal**. As such, a **Phase** is referred to as an **AntiRigid Sortal**. Other examples of **AntiRigid Sortals** include: **Student Pilot (Role)**, **Airline Staff (Role)** and **Healthy Human (Phase)**.

Thinking about identity and concepts such as **Sortals** and **Rigidity** may seem to add additional complexity but, as before, this provides structure to how we organise and reason with things in our world, so we don't end up modelling things that don't make sense. It can also help us to improve the amenability of our models to automated semantic checking, which is becoming essential for complex system safety. There are also additional constraints that apply, such as:

- A **Rigid** type can't be a subtype of an **AntiRigid** type. For example, a **Computer** cannot be a subtype of '**Operational Computer**' as it is still a **Computer** even when it is switched off.

There are many other rules that apply, and these can thankfully be encoded in modelling tools, so we are guided towards building consistent models. Those with a deeper interest may wish to explore the full OntoUML specification [3].

Maintaining Good Relations!

Any model, such as an architectural design, needs more than just a collection of objects to give it meaning. It is only through the conceptual relationships between those objects that we really harness the expressive power of our models.

In the example above we saw that a **Flight** was a relationship between **Pilot**, the **Passenger** and the **Air Traffic Controller**. A **Relator** is the thing that must exist for two or more individuals to be connected in a material way. Other examples of **Relators** are: **Marriage**, **Investigation**, **Enrolment**, **Employment** and **Subscription**. As the **Relator** exists for the entire time the relationship exists, then it is also **Rigid** as we saw with our **Flight Relator**. In fact, all **Relators** are actually **Rigid Sortals** as you can individuate and count this type of relationship.

We can now expand our model to include a further **Relator**: **Flight Authorisation**. The **Flight Authorisation** is the authorisation given by the **Air Traffic Controller** for the **Pilot** to follow the associated **Flight Plan**, which includes the specific **Flight Route** to be taken for the **Flight**.

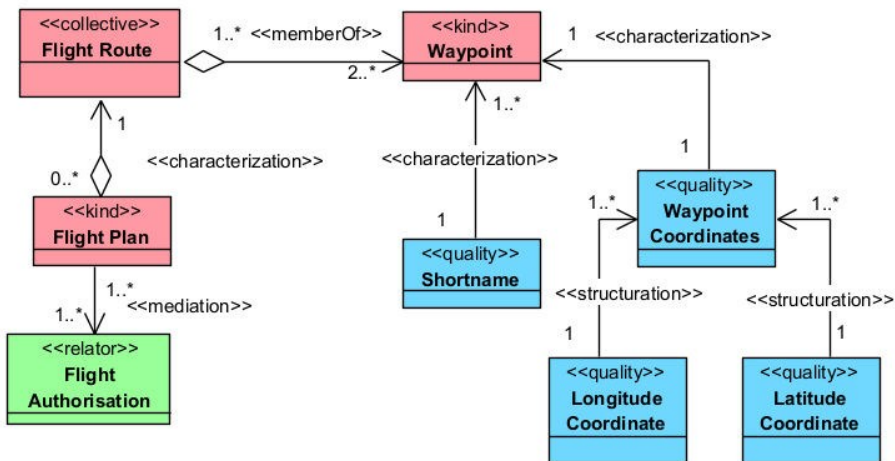
Note that **Flight Plan** will comprise many things but as we are interested specifically in exploring **Flight Route** we've modelled that explicitly. This is shown as a **Shared Aggregation** relationship with a hollow  diamond at the containing element end, to indicate the **Flight Plan** contains a part known as **Flight Route**. This form of **Aggregation** relationship allows the two things it connects to have different lifecycles and multiple membership – the **Flight Route** for this particular **Flight Plan** can exist in isolation from this **Flight Plan** and may well be adopted by other **Flight Plans**, possibly at the same time.

While multiple **Flight Plans** may contain the same route, additional constraints may be required to ensure that two **Aircraft** do not fly the same **Flight Route** at the same altitude at the same time.

But what is a **Flight Route** in reality? In many cases it is a collection of **Waypoints** which are representations of physical locations on the Earth the Pilot intends to navigate between.

We now introduce another useful stereotype to help us model a collection, using the stereotype **Collective**. This is used to bundle similar objects that are all perceived as contributing equal parts to the whole. Our **Flight Route** is therefore a type of **Collective** that has a number of **Waypoints** as its members².

Waypoints have exactly one set of **Waypoint Coordinates**, which is modelled as a combination of a **Latitude Coordinate** and **Longitude Coordinate**, being alphanumeric values, representing physical locations. For example, in the chart at the beginning of the article, for the **Waypoint** with **Shortname** **ETOXU**, the **Latitude Coordinate** is **N47 43.6** and the **Longitude Coordinate** is **E009 33.0**. Note there is no altitude component to our **Waypoint Coordinates**, which exist at all altitudes at once (so more than one aircraft can pass **Waypoint Coordinates** at the same time, assuming they honour the rules of separation).



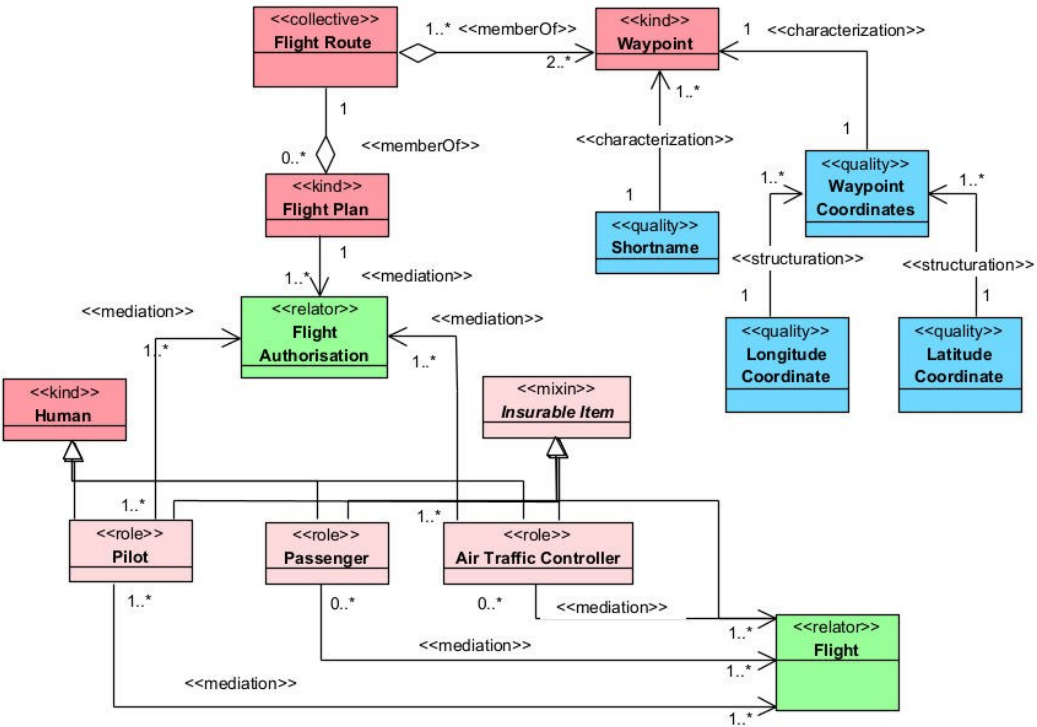
To model the **Latitude Coordinate** and **Longitude Coordinate** component of our **Waypoint Coordinates** we use another concept called **Quality**, which has three subtypes. A **Perceivable Quality** of an object is one that you can measure like **Weight**, **Height**, **Colour**, **Position** or **Speed**. A **NonPerceivable Quality** is one that cannot be directly measured by an instrument, like **Passenger Satisfaction**. Finally, a **Nominal Quality** is used to make reference to an individual or quality, like a **Human Name**, a **Book Title** or a **Boarding Card Number**.

² In the diagram, classes below Flight Authorisation have been omitted for clarity but still form part of our model.

Note that, importantly, we are, at the moment, modelling only the representation that **Pilots** use in their **Flight Plans**. In our example, **Shortname**, **Waypoint Coordinates**, **Latitude Coordinate** and **Longitude Coordinate** are all modelled as **Nominal Quality** because they are character strings on a chart. If we were to extend the model to include the physical points on the Earth that the strings represent, these points would be modelled as measurable **Perceivable Qualities**.

Up until now, we’ve only focused on class stereotypes, but there are also many stereotypes for relationships, of which we’ve already seen **Mediation** (how a **Relator** mediates between things) and now we introduce **Characterization** (how a **Quality** relates to an object). For example, **Shortname** has an association relationship with the **Waypoint** shown using the **Characterization** relationship stereotype. We’ll introduce just one more useful relationship stereotype: **Structuration**. This allows us to provide structure to a **Quality** so **Waypoint Coordinates** can be split into the textual representations of **Latitude Coordinate** and **Longitude Coordinate**.

In our model, **Waypoint Coordinates** has one and only one **Latitude Coordinate** and one and only one **Longitude Coordinate** value and can only be associated with a single **Waypoint**. As mentioned previously, altitude, is not relevant to our **Waypoint** as so is not a component of our **Waypoint Coordinates** representation. Also, a particular **Waypoint** has exactly one short abbreviated name (**Shortname**).



We can now congratulate ourselves on expanding our model to include **Flight** planning aspects. The model is still fairly simple, but we can already use it to communicate and reason about our worldview.

The model forces us to think about the multiplicity of the relationships we've established, such as the number of **Waypoints** in a **Flight Plan**. We probably need a minimum of two; the place we are starting from and the place we will land (even if this is the same location), and that might actually be OK for a really short flight. Longer flights will of course have more and in fact, in real air traffic control domains, navigational organisations may add additional **Waypoints** to help safely control flight movement across their airspace.

We should also permit the same **Waypoint** to appear more than once in a given **Flight Route**. This might be required for "there and back" type flights or maybe a circular, or figure of eight type route.

What We've Learned

Here's a recap of what we've covered in this edition:

- We recalled why ontological modelling is important and applies to everyone's day-to-day endeavours and refreshed our memories on what we learned in the previous article covering **Classes**, **Stereotypes** and identity principles.
- We learned the term **Sortal** to refer to categorisations that allow us individuate and count their instances and explored the temporal phases of things through their **Rigidity**
- We learned about **SubKinds** and explored relationship stereotypes by providing an example of a **Flight Authorisation** covering the **Relator** class stereotype and the **mediation** relationship stereotypes
- We finally expanded on the example to introduce the idea of a **collective** of **Waypoints** forming the **Flight Route** and how **Quality** can be used with the stereotyped **Association characterization** to describe what individuates a **Waypoint**. We also introduced **structuration** to decomposed different aspects of a **Quality**

This episode hopefully shows the power and value of ontological models: they make us be very explicit and clear about the relationships we are trying to articulate and if we find difficulty in modelling our ideas through an ontological model, then it may be a sign that they are not fully developed yet.

As well as articulating what we care about, ontologies also help show deficiencies in our thinking, and this is perhaps even more important from a safety risk management perspective.

We will leave you with a couple of questions to consider, which explore how robust our model is at the moment:

- In reality, an individual **Human** cannot be both a **Pilot** and a **Passenger** on the same **Flight** – does the model prohibit this situation?
- Does the model prohibit **Waypoints** located at different **Waypoint Coordinates** from having the same **Shortname**? If not, do you see any problems that might arise?

In the next episode we will answer these questions and explore the idea of an *anti-pattern* – if we think about a design pattern as being a common arrangement that has worked well in the past, then an anti-pattern can be considered an arrangement that frequently seems at first like a good approach but has flaws and so should be avoided.



References

- [1] T. R. Gruber, A Translation Approach to Portable Ontology Specifications, in Knowledge Acquisition, 5(2):199-220, Academic Press, 1993, <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>
- [2] G. Guizzardi, A. Botti Benevides, C. M. Fonseca, D. Porello, J. P. A. Almeida, T. P. Sales, Ufo: Unified foundational ontology, Applied ontology 17 (2022) 1–44, <https://doi.org/10.3233/AO-210256>
- [3] OntoUML Specification, Marek Suchánek, <https://ontouml.readthedocs.io/>, accessed Jan 2026
- [4] Unified Modelling Language, <https://www.omg.org/spec/UML>
- [5] European Geostationary Navigation Overlay Service, https://en.wikipedia.org/wiki/European_Geostationary_Navigation_Overlay_Service, accessed Sept 2025

Image attributions:

navigation mao: El-mejor, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
John Locke: ID 19445045 © Georgios Kollidas | Dreamstime.com

The Ontology Working Group (OWG)



The OWG was set up in 2020 to help address issues with the lack of formalisation in terms and conceptual relationships used in standards and guidance documents. In particular, the similar but at times different, interpretations of terms and concepts used between safety and security domains is an ongoing cause of confusion and misunderstandings and presents barriers to more formal integration of these disciplines.

The OWG has its origins in the Data Safety Initiative Working Group (DSIWG), which, between 2016 and 2019, explored the development of an ontology model for the concepts described in the Data Safety Guidance (DSG). However, it was realised that the work couldn't be completed without a wider ontological model for risk, and so the OWG was established and is currently working on developing a clear, systematic terminology, and unambiguous model of risk and value concepts and their relationships.