## Use Case1

This note outlines a use case of an ontology fragment for an engineering example. The example is an abstraction of an actual engineering product development scenario that I have experience with. The representation of this example within OWL was first reported in in 2008 [Application of OWL 1.1 to systems engineering](http://www.academia.edu/download/30680405/owled2008dc_paper_9.pdf). This paper was in regard to how to use an upper ontology to represent SE artifacts. In this case we used DUL. The ontological approach was further developed in [Representing Product Designs Using a Description Graph Extension to OWL 2.](http://ceur-ws.org/Vol-432/owled2008eu_submission_45.pdf) The focus of this paper was how to represent two identical parts used in two different roles . The focus of those papers was to determine how suitable OWL was representing system engineering ontologies. The focus of this note is on how an ontological representation can be used to avoid problems in the engineering development process.

### Use Case Description

The use case involves multiple l hierarchical decompositions such as systems and subsystem, a logical architecture and its components, and physical decomposition. At the preliminary design phase the program has an organizational hierarchy, a system decomposition and a logical architecture. The hierarchies are closely related, but are distinct for good reasons. Each of the hierarchies reflects institutional knowledge of the domain, e.g., aircraft design, as well as customer requirements. One is not a refinement or specialization of the other. Where they do not coincide is where the design process can run into trouble. While different enterprises may use different terminology the subject matter is common to a broad range of engineering development situations.

This use case has been simplified to have 3 (composite structure) decompositions. Each decomposition has interconnections within the decomposition and there are interconnections between the nodes in the different decompositions. Multiple problems common to engineering efforts can be identified by analysis of the full graph consisting of the decompositions together with all of the cross links. These problems could be avoided by ensuring that the graph structure satisfies well-formedness rules appropriate for the enterprise. We use an organization decomposition which we call the IPT structure. The IPT structure is responsible for all work done and is satisfactory. While different kinds of organizations are used to organize the work any engineering organization will have some kind of organizational decomposition. The most common organizational structures reflect a mix of the system decomposition and area of cross cutting interactions between subsystems. For example, For example one might have an organizational entities for a the airframe, subsystems, and mission or payload system, together with cross cutting organizational node for flight handling qualities.

Many problems occur when no organization has responsibility for particular tasks or do not have the authority (resources) to perform the work. For example, no one is monitoring that the fuel system piping is too close to components which are sufficiently hot to be dangerous. The information needed to define the horizontal links between decompositions are dependent on the product domain, even though there is a lot of commonality between say aircraft and cell phones. In both cases problems with the battery interaction with subsystems may cause problems and there may be no organization responsible to watch for and fix these problems.

This use case uses both a logical architecture and a physical architecture decomposition. Again these are very distinct and cannot be forced to be the same. For example, in this example the fuel system logical architecture calls for two fuel tanks connected by a pump together with input and output ports. This architecture is needed to be sure that the fuel system satisfied its “mission” or “function” of storing fuel and providing it on demand to the engine. While the physical architecture may have parts corresponding to the components of the logical architecture. This is not necessarily the case. The logical architecture is established before the physical architecture. For this aircraft design the aircraft physical architecture has no components with part numbers corresponding to the fuel tanks. The fuel is stored directly in cavities within the wing box structure. There is, for example, no blatter corresponding to a fuel tank. This was done for weight saving. Of course the logical architecture has to be allocated to the physical architecture, but the correspondence may be somewhat indirect as in the diagram of Figure 1. The two fuel tanks are “allocated” to the wing box. There is a direct correspondence between the pump in the logical architecture and a pump part of the physical architecture which may be procured within a bill of materials.



Figure 1

The use case reflects an engineering process which would not be appropriate for some developments. In this case the logical architecture is reasonably well-defined before the requirements are given. This would be the case for example, for aircraft and cell-phones, but not necessarily for spacecraft for mars exploration. The detailed engineering is how to be sure that the ultimate physical architecture satisfies the requirements and constraints which are implicit, but not stated in the requirements.

During the preliminary design phase the program had a system-subsystem decomposition and a logical architecture. Both are represented as decomposition structures. The notes in the organizational hierarchy have responsibility and allocate budgets. They also perform roll-ups of cost, weight, thermal properties, etc during the development phase. THe rollups are used to validate that the program is on track to meet its requirements.

The program did not have a physical architecture in the sense of specific parts with part identifiers for several years. A physical architecture results from a detailed design. Its parts can be given with specifications to suppliers to be manufactured. For example, while a physical architecture may have parts for fuel tanks, this particular design did not have fuel tank parts. The fuel was simply stored in cavities in the wing box. The elimination of bladder tanks within the wing box was to save weight.

The use case will focus on the fuel system as one of the subsystems which is part of the ontology, and the logical architecture of the fuel system. The logical architecture of the fuel system preceded the physical architecture by several years. The logical architecture of the fuel system is described in the following diagram. At the preliminary design phase one can correspond the fuel system with an organizational structure. The organization component has responsibility for the fuel system. Responsibility does not mean that the IPT owns the parts of the fuel system only that it has responsibility to be sure that the functionality of the fuel system is satisfied.

The logical architecture reflects the institutional knowledge of the components of fuel systems for aircraft. The logical architecture in this abstracted example has two fuel tanks, the two fuel tanks are connected by a pump whose job is to equalize the fuel in the two tanks. The fuel system has input and output ports for filling the tanks and supplying fuel to the engine. Often the physical architecture has the same decomposition as the logical decomposition. However, this turned out not to be the case here. All of the parts in the physical decomposition have part numbers. In this aircraft design there were no part numbers for the fuel tanks. The “function” of storing fuel was satisfied by the wing box. The wing box has a part number and it has two cavities which are used to store fuel.

### Analysis

The graph of the use case fragment has 3 hierarchies. Each hierarchy can be described as a composite structure where the components within the structure may have associations. The full graph has associations between nodes in different hierarchies.

The most general ontology level consists of composite structure hierarchies with their conditions to be well formed. Large classes of engineering development programs and projects use multiple hierarchies such as illustrated here. In general they are a bit more complex than the 3 hierarchies used here. The syntactic rules for such a program graph ensure that all task which are identified for a system are covered in the organizational hierarchy and are resourced. These tasks often relate to the interaction between multiple subsystems.