Development of Complex Systems using xtUML

Nils Paulsson, PhD, Senior Architect, Saab Aeronautics
Cortland Starrett, President and CEO, One Fact Inc.

Abstract

Executable translatable UML (xtUML) is a precisely defined semantic dialect of UML that descends from the Shlaer-Mellor Method of Object-Oriented Analysis and Recursive Design. Data, control and processing aspects of system behavior are encoded according to a set of execution rules. The complexity of large systems is managed by working at a higher level of abstraction. The modeling language expresses functionality more efficiently than platform specific alternatives. The requirements of the deployment platform are expressed as a separate concern in the model compiler freeing the application model of these details. This approach factors application modeling out from deployment platform design resulting in two separate forms of reusable IP: application models and target model compilers.

Complex System Partitioning

xtUML is a top-down MBSE[1]-methodology that addresses complexity by identifying **natural partitions** of a given system. The highest partition level is the *Domain*. A domain is a subject matter, having justification of existence on its own merits. As such, domains are not created. Domains, and their interdependencies, are **found** through analysis & elaboration by subject matter experts.

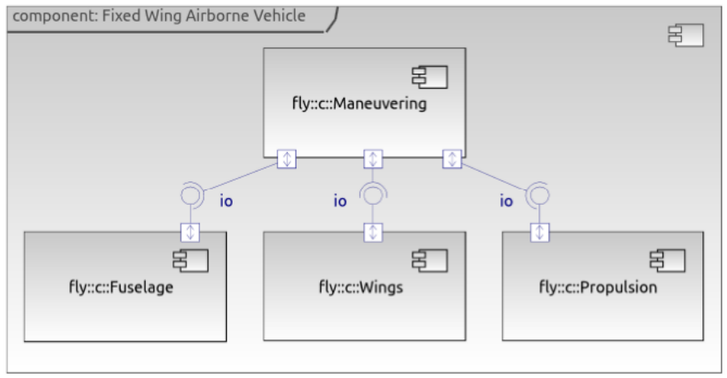


Figure 1. Domains for a fixed wing airborne vehicle

A domain is modeled by a distinct set of classes – i.e. a **common vocabulary**. Classes behave according to rules and policies that are characteristic of the specific subject matter.

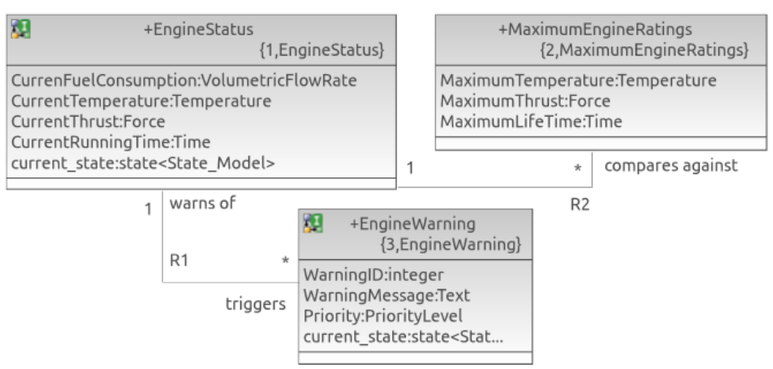


Figure 2. Class examples In the Propulsion domain

[1]MBSE – Model Based Systems Engineering
[2]PIM – Plattform Independent Model
[3]PSM – Plattform Specific Model

Parameterized Designs

Parameterized designs rests on the fact that subject matter-centric information models often are more stable over time than functional oriented models. By populating the *MaximumEngineRatings* class (figure 2) with different data sets, different engine characteristics can be realized without having to regenerate or, even worse, remodell the system. A **generic** engine monitoring function can the then be defined as in figure 3.

```
if ( self.CurrentTemperature > max.MaximumTemperature )
  generate EngineWarning1:flag( message:"Critical temperature!" ) to warn;
end if;
if ( self.CurrentThrust > max.MaximumThrust )
  generate EngineWarning1:flag( message:"Critical thrust!" ) to warn;
end if;
```

Figure 3. Engine monitoring function

The Model is the Solution – PIM & PSM

Functional application requirements and implementation technology requirements evolve and change at different paces. Some applications are stable across multiple generations of technology. Some technologies get reused in multiple applications.

Using aviation as an example, applications have useful lifespans that span generations of technology. Planes through all of time have fuselages, wings and propulsion. The concepts of altitude, bearing, thrust, yaw and pitch are largely independent of the technologies used to realize them. Materials, fuels and electronics change and emerge at rates different from the applications upon which they are deployed.

The functional behavior of the aircraft can be specified and encoded in one set of models. The technological implementation requirements for MPU, RTOS and network protocol can be designed as a separate concern. Separating the concerns of application from implementation reduces the complexity of each. Separating application from implementation also results in independently reusable artifacts. In both cases, application and implementation, the model is the solution.

The models capturing the application (PIM[2]) can be reused on new technology platforms. The model compilers, implementation patterns and technology choices (PSM[3]) can be reused to deploy other applications. In aviation, the deployment technologies may be applied to new models of aircraft or to other forms of airborne vehicle.

System of Systems Analysis

The high level of abstraction, a subject matter focus, parameterized designs and pluggable model compilers enable early simulation & analysis of a complex system in a *System of Systems* fashion.

The models in figures 1 - 3 are equally applicable to a fighter, a cruise missile and an UAV. Various intended realizations can be simulated & tested by populating the models with different data sets. The system can yet be evaluated at a level of abstraction **comprehensible** by subject matter experts. E.g. how does a propulsion using a jet engine compare to an electric engine with respect to UAV performance taking life time cost, time in air and IR-signature into account?

Case Study

Modular Evolutionary System Core (MESC) is a reusable model-based architecture for tactical planning.

- Focus on planning functions for autonomous missiles.
- Strict subject matter separation.
- xtUML-models identified during development of planning demonstrators for various missile systems.

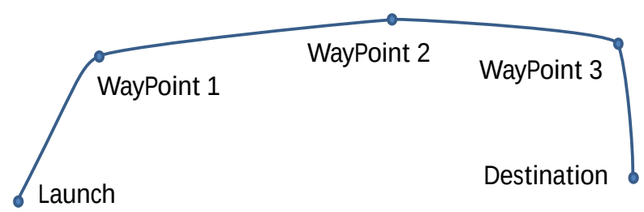


Figure 4. A planning example

Case summary

- Integration of systems and software engineering disciplines.
- Captures corporate knowledge of planning without muddling models with implementation details.
- Supports future demonstrator development efforts, e.g. planning autonomous missions for UAV:s.
- Enabled rapid development (< 3 months) of an air-defense demonstrator.
- > 250 000 lines of generated code.
- > 80 % reuse between missile specific realizations of planning functions.