A consistent and unambiguous implementation of code generation (model to text transformation) from UML must rely on a well defined UML profile, customizing UML for a particular application domain. Such a profile must have a solid foundation in a formally correct ontology, formalizing the concepts and their relations in the specific domain, in order to avoid a maze or set of wildly created stereotypes.

The paper describes a generic profile for the code generation of component based distributed systems for control applications, the process to distil the ontology and define the profile, and the strategy followed to implement the code generator.

The main steps that take place iteratively include: defining the terms and relations with an ontology, mapping the ontology to the appropriate UML metaclasses, testing the profile by creating modelling examples, and generating the code.

### The Comodo metamodel

*Figures showing the Comodo metamodel with various components and their relationships.*

### An example: E-ELT prototype instrument

As an example, we show here the model corresponding to a small instrument described in the poster WEPK025, which was developed as a prototype for the evaluation of software and electronics for E-ELT instrumentation. In this model we define generic interfaces for Subsystems and Devices which are specialized for a particular application. For example, from the Device we specialize Motors and from Motors Filter Wheels.

All high level coordination components derive from a Master Component and share the definition for a common State Machine. As well, all Devices share a state machine specified in the Device Component.

### Architecture and deployment of the instrument

*Diagram showing the architecture and deployment of the instrument.*

### For more information:

- gchiozzi@eso.org, landolfa@eso.org, rkarban@eso.org