Patterns for Multi-Layered System Architectures

P. Brooks[1], C. Koch[2], Z. Kovacs[1], J-M Le Goff[2], R. McClatchey[1]

[1]Centre for Complex Cooperative Systems, Univ. West of England, Bristol BS16 1QY, UK
[2]EP Division, CERN, 1211 Geneva 23, Switzerland
Email: Richard.McClatchey@cern.ch

Abstract

The CRISTAL project has studied the use of a meta-model representation to manage the evolving needs of a large physics community. The meta-model is based on a multi-layered architecture and provides a description of the system to be built. This description is accessed by multiple applications and as the range of application domains grows, new requirements have to be satisfied which require extension of the meta-model. This paper reports on the building of a multi-layered system architecture by abstraction from this meta-model. It identifies design patterns that emerge from building the ontology which underpin all such 'description-driven' systems. The approach is evaluated in the domain of large-scale physics detector construction at the European Particle Physics laboratory, CERN.

Description-Driven Systems and Multi-Layer Architectures

'Description-driven systems' can be defined as systems in which the description of a domain-specific configuration is captured in a computer-readable form. This description can be interpreted by applications to achieve domain-specific goals. In a description-driven system descriptions are separated from instances and managed independently, to allow the descriptions to be specified and to evolve asynchronously from particular instantiations (and executions) of those descriptions. As a consequence a description-driven system requires computer-readable models both for descriptions and for instances. These models are loosely coupled and coupling only takes place when instances are created or when a description, corresponding to existing instantiations, is modified. The coupling is loose since the lifecycle of each instantiation is independent from the lifecycle of its corresponding description.

One practical example of the use of a description-driven systems is a workflow management system (WfM) where the business process model defines the instantiated workflows and the definitions are managed and enacted separately from the instantiations. WfM systems can be built on a multi-layer architecture and recently the OMG have standardised the architecture of WfMs using a multi-layer model [1]. In WfM systems the workflow instances (such as activities or tasks) correspond to the lowest level of system abstraction - the *instance layer* (see Figure 1). In order to instantiate the workflow objects a workflow scheme is required. This scheme describes the workflow instances and corresponds to the next layer of abstraction - the *model layer.* In order for the workflow scheme itself to be built, a further model is required to store the semantics for the generation of the workflow scheme. This model (i.e. a model describing another model) is the next layer of system abstraction, the *meta-model layer* (Figure 1).

The semantics required to adequately model application-specific information will, in most cases, be different. For example, the semantics for describing Product Data Management (PDM) systems (product types, product composition types etc.) will be very different from those describing WfM systems (activity types, activity composition types, actor types, etc.). To facilitate integration between meta-models a universal type language capable of describing all meta-information is required. The

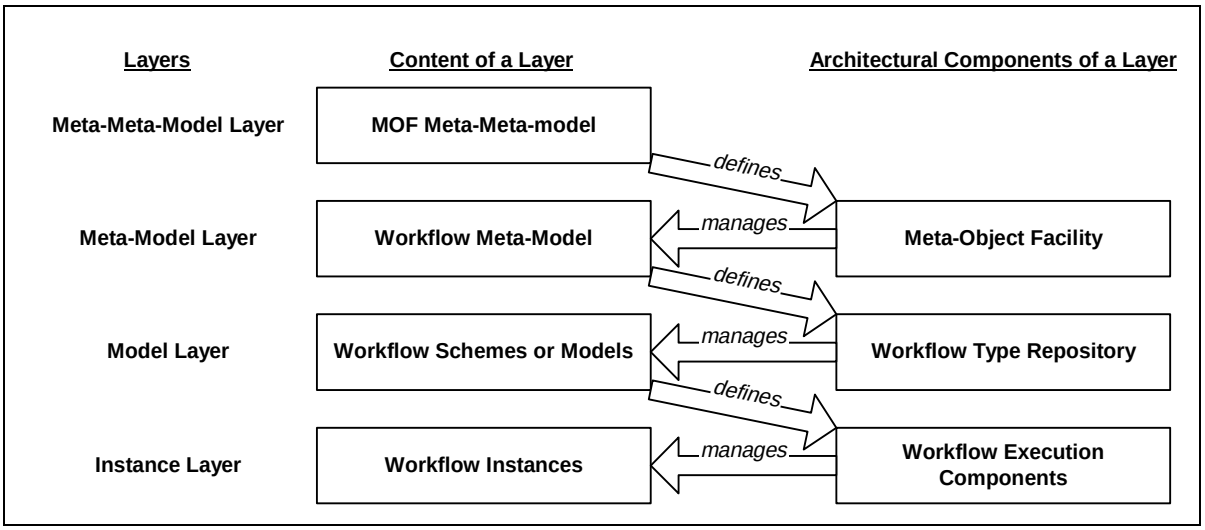

Figure 1: Workflow management systems as description-driven systems.

common approach is to define an abstract language which is capable of defining another language for specifying a particular meta-model, in other words *meta-meta-information*. The accepted conceptual framework for meta-modeling is based on an architecture with four layers: Figure 1 illustrates the four layer meta-modeling architecture adopted by the OMG, based on the ISO 11179 standard.

The *meta-meta-model layer* is the layer responsible for defining a general modeling language for specifying meta-models. This top layer is the most abstract and must have the capability of modeling any meta-model. It comprises the design artifacts in common to any meta-model. At the next layer down a (domain specific) meta-model is an instance of a meta-meta-model. It is the responsibility of this layer to define a language for specifying models, which is itself defined in terms of the meta-meta types of the meta-meta modeling layer above. Examples of objects at this level from manufacturing include workflow process description, nested subprocess description and product descriptions. A model at layer two is an instance of a meta-model. The primary responsibility of the model layer is to define a language that describes a particular information domain. Example objects for the manufacturing domain would be product, measurement, production schedule, composite product. At the lowest level user objects are an instance of a model and describe a specific information and application domain.

A recent thesis [2] studied the integration of product data and workflow management through a common description-driven data model for the CRISTAL project. In building the data model a set of design patterns [3] were identified including the item description pattern, an enriched directed acyclic graph pattern, a publish/subscribe pattern, a version pattern, an enriched homomorphism pattern and use of a mediator pattern for retrieval of information from the integrated product and process data model. It was speculated that these design patterns are part of the essential elements of any description-driven system. Figure 2 shows how the item description pattern allows the dependency between a description and its instantiation to be handled. In the figure the item description pattern has been employed to provide the semantics that relate the Item class (of the model layer) with the ItemDescription class of the meta-model layer. The pattern can also be applied to relate an Item instance (at the instance layer) with its corresponding ItemDescription instance (at the model layer). The multi-layer architecture which forms the basis of a description-driven system is a direct consequence of the use of the ItemDescription pattern. Later in this paper these patterns are used to build an ontology which facilitates not only product and process integration, but the development of an enterprise model which spans multiple domains.

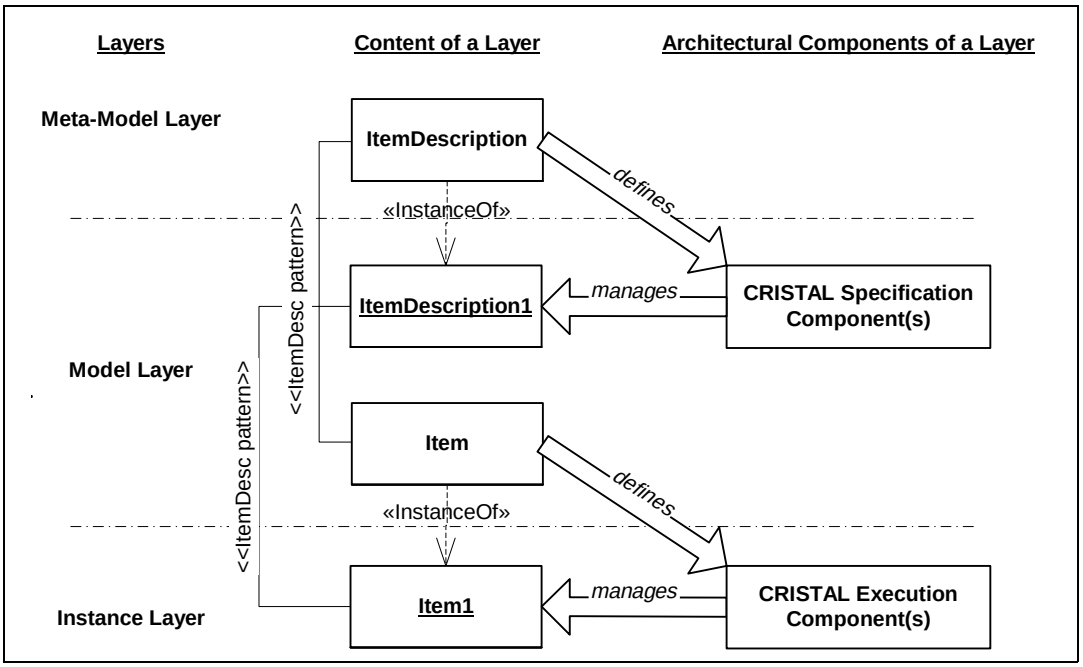

Figure 2: The CRISTAL three-layer architecture.

Description-driven systems features can be realised through the adoption of a multi-layered architecture. Description-driven systems are flexible and provide many powerful features including reusability, complexity handling, versioning, system evolution and interoperability The study reported in this paper investigates how product and process information can be handled through the use of a common meta-model in a so-called description-driven system. The description-driven approach is outlined and its role in integrating product and process models for a data warehouse example is identified. The resulting meta-model is general in form and can be used to produce materialised views (so-called 'viewpoints') onto the data warehouse.

Essentially the meta-model can be used as the basis of an ontology describing how products and processes are inter-related, as it is predicated on a set of basic concepts and their relationships. This paper shows how a domain-specific ontology can be abstracted from a description-driven system designed for that domain, how a domain non-specific ontology is a natural extension of this process of abstraction and how, using this ontology, specific knowledge bases describing specific domains can be implemented. A prototype has been developed which facilitates this study of a common ontology for product and process modeling. The objective of this prototype is to integrate a Product Data Management model with a Workflow Management model in the context of the CRISTAL (Cooperating Repositories and Information System for Tracking Assembly Lifecycles) project currently being undertaken at CERN, the European Laboratory for Particle Physics in Geneva, Switzerland. This position paper begins by describing the CRISTAL environment in the following section. In the third section, the need to access data through different views is discussed, and in the following section, the need for further abstraction is elaborated on. Finally, a conclusion and further remarks is presented.

The CRISTAL Project

The Compact Muon Solenoid (CMS) experiment [4], currently being constructed at the European Laboratory for Particle Physics at CERN, will comprise several complex detectors for fundamental particle physics research. The CRISTAL system is a distributed product data and workflow system being developed to monitor and control the production and assembly process of the CMS detector at CERN.

In essence, CRISTAL employs WfM and PDM techniques to provide an infrastructure in which the engineering data describing the detector and amounting to a few Terabytes can be warehoused [5]. This detector description will later be used in the capture of several Petabytes of physics events once the CMS experiment gets underway from 2006 onwards.

The design of the CRISTAL prototype was dictated by the requirements for adaptability over extended timescales, for system evolution, for interoperability and for complexity handling and reusability. In adopting a description-driven design approach to address these requirements, a separation of object instances from object descriptions instances was needed. This abstraction resulted in the delivery of a meta-model as well as a model for CRISTAL. The assembly of CMS is being carried out by groups, distributed geographically over several continents, with responsibilities for individual sub-detectors. Each group needs to be only loosely coupled to others for final detector integration and must preserve their autonomy during the assembly process. Each CRISTAL system is set up to manage the accumulation of potentially Terabytes of physical characteristic data into a detector data warehouse during the construction of a particular CMS sub-detector. The construction follows a specific production plan and each detector is assembled and tested in a step-wise fashion. A distributed object-oriented database is used to hold both the engineering data and the definitions of the detector components and of the tasks which are performed on the components.

An approach has been taken in the CRISTAL design, which promotes self-description and data independence. A multi-layer architecture has been developed to cater for the CRISTAL (meta-)model which facilitates data integration. This allows separation of definitions from instantiations through the use of so-called ‘meta-objects’, promotes object re-use, reduces complexity and facilitates self-description.

Accessing Data from Multiple Views

As any construction process evolves, so more data, and the relationships between different aspects of the data, must be permanently recorded in the construction repository. To cope with this the construction system used must, ideally, be able to support dynamic self-reconfiguration. The CRISTAL repository has been designed to be self-describing i.e. to retain knowledge about its dynamic structure and for this knowledge to be available to the rest of the distributed infrastructure through the way that the system is plugged together.

The CRISTAL meta-model is comprised of so-called ‘meta-objects’ each of which is defined for a class of significance in the data model: e.g part definitions for parts, activity definitions for activities, and executor definitions for executors (e.g instruments, automatically-launched code etc.). Figure 3 shows the meta-object concept. In the model information is stored at specification time for types of parts or part definitions and at assembly time for individual instantiations of part definitions. At the design stage of the project information is stored against the definition object and only when the project progresses is information stored on an individual part basis. This meta-object approach reduces system complexity by promoting object reuse and translating complex hierarchies of object instances into (directed acyclic) graphs of object definitions. It is believed that the use of meta-objects provides the flexibility needed to cope with their evolution over the extended timescales of CRISTAL production and the flexibility required to cope with ad-hoc activity specification.

In designing the CRISTAL meta-model, UML has been followed; the result being a detailed model, presented elsewhere [6]. This model describes relationships, types, inheritance, containment and other associations between the meta objects in the system. The meta-objects in the model are definitions, for example, part definitions or activity definitions and the definitions are either elementary or composite in nature. CompositeMember objects capture the membership of objects in other objects. The data description world of, in this example, parts and the process description world of, in this case activities, displays an elegant symmetry with respect to compositeness.

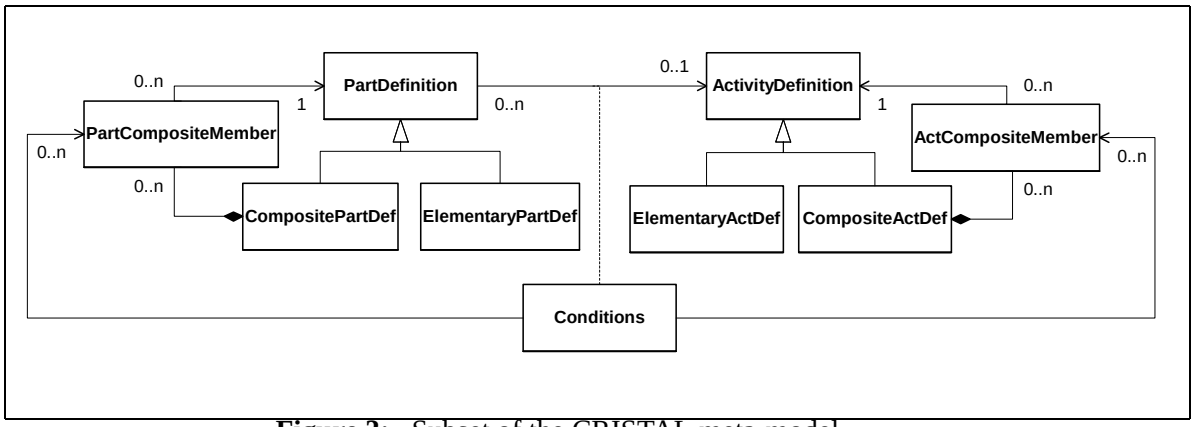

Figure 3: Subset of the CRISTAL meta-model.

The data model developed for CRISTAL has been designed to be generally applicable to production management environments. In fact the model, shown in Figure 3, is sufficiently generic in nature to describe many applications in which both data descriptions and activity descriptions (and their inter-relationships) are captured in a database and in which traceability is required of each execution of an activity on a part or product. The structure of the CRISTAL data model is based on directed acyclic graphs describing physical and catalogue aggregation as defined by [7]. However, the object model proposed in [7], although also based on directed acyclic graphs, does not provide semantics for the relationship between physical and catalogue aggregation nor does it explicitly capture the membership of one object in its aggregate, as in the CRISTAL model. The CRISTAL model is rich in semantics and, consequently, could be applied to general aggregation-based data management systems.

Figure 3 shows that there is an association between a given activity meta-object definition and a named part meta-object definition and that this association carries semantics. The left side of Figure 3 captures the product aspects of the meta-model and the right side the process aspects, as needed to integrate PDM and WfM. The CRISTAL data model has been designed so that each assignment of a Part Definition to an Activity Definition is declared for a specific purpose. In detector construction, the assignment is made to indicate the activity to be instantiated for the assembly of a particular instance of a part of a given part definition. Each assignment has associated with it some conditions: in detector construction, the data model captures the definition of the conditions required for each assignment of an activity definition to a part definition.

This technique can be generalised for other applications. For example, the association of a maintenance activity to a part will require quite different conditions to be captured than when the detector was constructed. Also, the association of a calibration activity to a part would require calibration-specific conditions to be captured. In other words, the identified association between the process and part description worlds carries rich semantics. It allows many other links to be made between aspects of the overall CRISTAL data model: the same mechanism can be used to assign agents to activity definitions for the purposes of enactment or the assignment of agents to part definitions for the purposes of resource management. In the following section of this paper this mechanism is shown to be useful in building an ontology which relates product and process worlds for knowledge retrieval. Users will require flexible ways to find, access and share data captured in the construction data warehouse. The actual information required will depend on the "viewpoint" and the role of the user in the organisation. A meta-model will assist the retrieval of information for selected detector components provided a ‘meta-query’ mechanism is developed which can navigate the warehouse meta-model, can interpret the structures in the warehouse and can present the data in a form meaningful to the end-user. The meta-query facility comprises a set of software processes which can be invoked either by a viewpoint-specific application or by a viewpoint non-specific application.

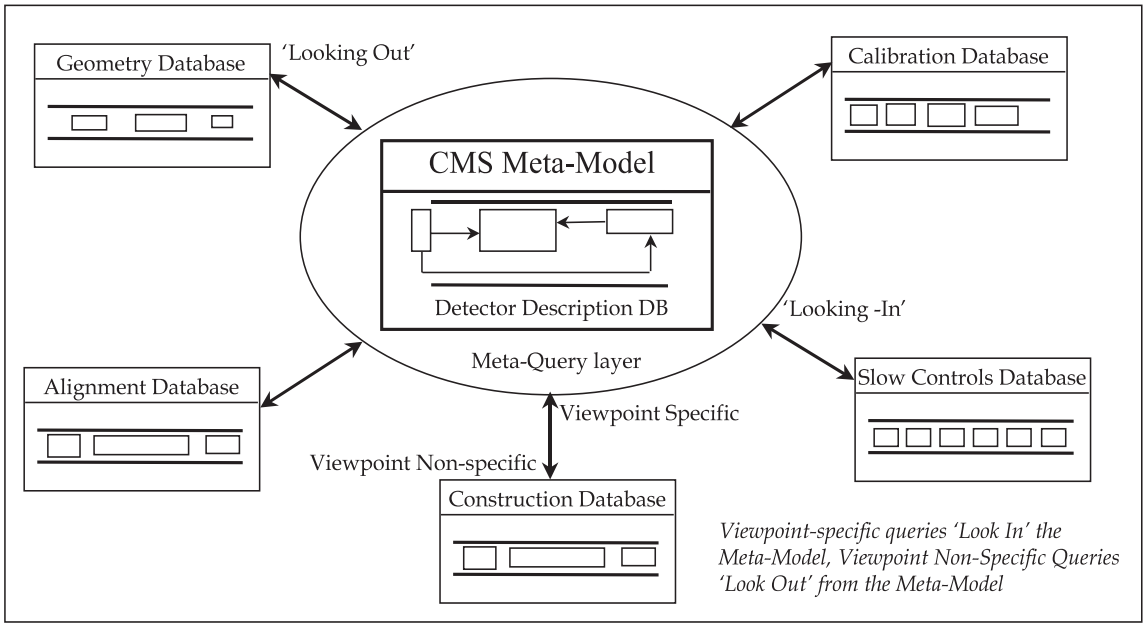

Figure 4: A generalised meta-model and query facility for CMS.

The queries either navigate a generalised warehouse meta-model to project out viewpoint-specific data (i.e."looking out" from the meta-model) or they navigate the meta-model to correlate effects between viewpoints (i.e. "looking in" the data model). In the viewpoint-specific case the agents perform the traversal of the detector description, following selected detector components in the construction tree and extract the relevant physics data for the application. In the viewpoint non-specific case, the queries are used to determine the effect of a system-wide change on individual viewpoints or sets of viewpoints i.e across viewpoints.

Figure 4, described in detail in [5], shows the architecture of a meta-model based system for CMS which encompasses multiple viewpoint databases (e.g Geometry, Calibration, Construction). In effect these databases are static 'materialised views' of the data in the warehouse, extracted for specific applications. In each case data has been extracted from a general CMS detector description database (having a description-driven architecture) via the meta-query facility. This generalised extraction facility can navigate the detector description, from a physicist-defined viewpoint, looking for specific data associated with a set of user-defined detector components. The result is a totally integrated set of collaborating databases which can be navigated for data extraction from a selection of viewpoints.

The Need for Further Abstraction

In Figure 4 the detector description database lies at the heart of the system. Being multi-layered in nature it can cope with evolving product and process definitions and it can also capture the description of the viewpoints (or materialised views of the data warehouse). The outlying viewpoints contain instances of data items which have been extracted from the data warehouse according to their definitions in the detector description database. The viewpoint solution to data sharing is sufficient for end-users provided that the viewpoints are read only databases. The viewpoints are static in nature - any evolution of the viewpoint content must be catered for centrally in the detector description database.

For some product and process domains this restriction is too severe. In these domains the viewpoints themselves must be writable and must be allowed to evolve with time and in these cases schema evolution in the viewpoints cannot be prevented. In the CRISTAL example, petabytes of data will

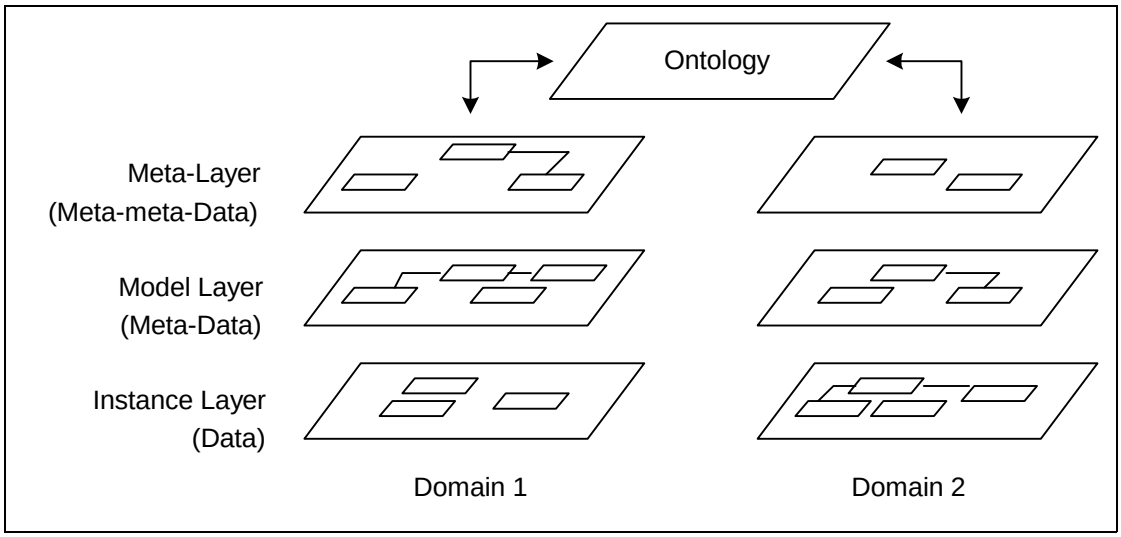

Figure 5: The OMG four-layer architecture across multiple domains.

ultimately need to be written in a viewpoint database which captures experimental physics data. The natural solution is to extend the description-driven approach to span not only the central detector description databases, but for it also to include the viewpoint databases. In this way, evolution is handled in each of the materialised views. There are two ways of extending the description-driven approach to cater for multiple domains. Either the detector description model and its meta-model have to be enriched to cater for a collection of domains (which requires all domain-specific software to be adapted) or the system is seen as a collection of separate domains that all have their own model and meta-model layers. The latter approach has been adopted to minimise management difficulty in CRISTAL.

As an example of extending the description-driven approach to cater for multiple domains consider the following example. Information gathered for a particular detector element during the construction of the detector will evolve over time and the data will be captured, and its evolution handled in the detector description database. Simultaneously, the same detector elements may be accessed for a variety of purposes e.g. alignment and controls and each purpose may have its own granularity of components defined. In addition, these granularities may change with time and new domain-specific data may be made persistent for each granularity. Viewpoints describing how detector elements are accessed (in this case for alignment and controls) may evolve in a manner independent from how these detector elements are being assembled (in the detector description database). Consequently, a new description - a functional view - of the detector is required, which maps more appropriately to the required data representations of the domains. Figure 5 shows how the OMG four-layer architecture for multiple domains can be used as the basis of a cross-domain ontology.

In summary a higher level of abstraction to an ontological layer is required to cater for multiple domains. In the construction domain, data are represented in a product tree, in which compositions are made not according to conceptual aggregation but strictly following the way components are assembled in constructing the detector. On the other hand, in the domains of part usage (e.g in alignment or control), the data model is structured by the way the detector is read out, that is the granularity in which measurements are taken during detector operation. A further need for abstraction to an ontology becomes apparent when the querying of domain information is investigated. To enable a study of viewpoint extraction, a prototype performing the translation of the stored construction data into appropriate representations has been implemented. An existing deductive database engine (called dlv, based on concepts developed in [8]) has been used to handle domain-specific queries. Details of dlv are presented elsewhere [9]. The query facility uses the CRISTAL meta-model to specify the queries that need to be executed for viewpoint extraction and to optimise access to the terabyte-sized construction database. To

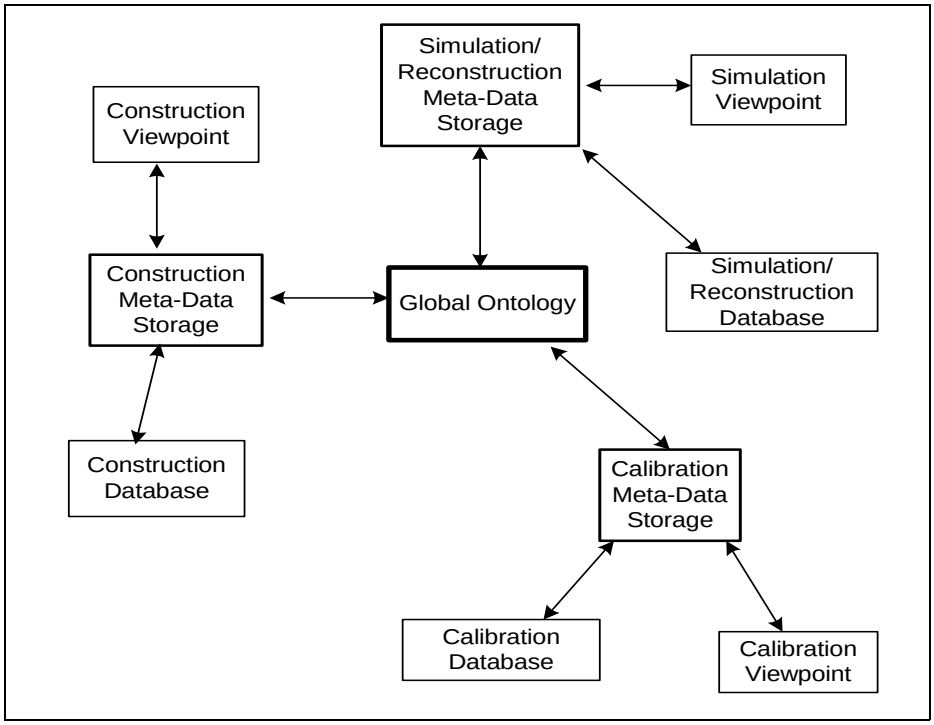

Figure 6: Towards a global ontology by adding a meta-meta model layer.

facilitate this, dlv needs to access both the meta-model and the model layers in order to translate queries specified by the user at the meta-model layer into the appropriate set of queries at the model layer.

In the prototype an agent-based system is being built up integrating many different databases from different domains with the different applications using the common ontology. Data extracted from the CRISTAL repository can be published to a set of agents that can perform different tasks. One category of agents performs analysis algorithms, where different versions of algorithms, performing the same analysis but using different methods, may co-exist. Other agents perform histogramming of data, and others collect and store computed data (such as calibration coefficients). Yet another type of agents provides access to the viewpoints, i.e. to the detector description and to characteristics of detector components. Besides these agents, there are those that have to generate viewpoints from data of different domains, as well as agents that perform cross-computations concerning several viewpoints. For this, agents have to work with the meta-models of several domains, which they need for their common virtual world in which they are able to communicate. Consequently, agents need to refer to a higher level of abstraction than the meta-model and therefore require ontological information.

Additionally, agents need an additional layer of abstraction to be able to communicate in terms that are appropriate to the meta-model layer. By adding another layer of description above the meta-layer, the process of abstraction can be extended into a meta-meta-model. Through the use of this fourth layer, schema evolution can be truly made unnecessary, as this highest level of abstraction incorporates a model of object-orientation. In Figure 6 three meta-models are shown : those of the construction, calibration and simulation/reconstruction domains. These domains are correlated - the evolution of data in one domain (e.g the construction) effects data in other domains (e.g. calibration). Each domain has its own meta-model and meta-data and viewpoints can be constructed on a domain-by-domain basis. To allow inter-domain meta-model navigation a global ontology is required, as shown in Figure 6.

Towards an Ontology of Description-Driven Systems

In implementing the prototype ontology, it was decided to utilise the UML meta-model [10]. Using it to build a model of object-orientation as the meta-meta-model of the system, it is possible to integrate different application domains and to construct a global meta-meta-data repository. This repository is a collection of all the domain meta-models and it should experience monotonic growth as the only way of change, since meta-models undergo careful analysis and design phases. This incorporation of a meta-meta-model into the system allows use of the meta-model as the abstract medium of communication for the agents, and as the execution of the aforementioned meta-query facility.

In building an ontology, meta-models of different application domains have been used to enable communication between software agents. However, having access to meta-models alone at the highest level of abstraction is insufficient for applications accessing the ontology. In addition the model of the data (i.e. its schema) must be available for applications to query. Then the meta-model describes an abstraction of all the objects of the system whereas the model specifies how instances of the objects, specific to a domain, are specified and together they provide the knowledge required by agents about the system and how it can be accessed. Consequently, not only are both meta- model and model layers required by agents in an ontology, but these two layers must be tightly coupled.

Earlier research [2] has shown that there are certain distinguished design patterns that recur in many different application domains (e.g. those of PDM and WfM discussed earlier) and that should also be formally specified in the ontology. It is easy to model these patterns in an ontological formalism since they are architectural patterns, identified by their structure and relationships. In Figure 2 the ItemDescription pattern was identified as being central to the construction of a description-driven system. It is through use of the ItemDescription pattern that tight coupling is achieved between the meta-model and model layers of the ontology.

Another important element of description-driven system is the homomorphism pattern which describes how two ItemDescription patterns are related. As a consequence of using the ItemDescription pattern semantics in the homomorphism pattern, and the fact that semantics (conditions) have been added to the association between item descriptions, there will necessarily be semantics attached to the association of one item class to another. However, the constraints that result from the use of the homomorphism pattern cannot be modeled with UML but need to be part of the ontology. One way of representing these constraints is through the expressive power of propositional logic. As an example, consider an acyclic graph pattern appearing as part of the meta-model of a specific domain. That acyclic graph pattern will translate into a tree pattern at the model layer via the ItemDescription pattern. Using UML it is not possible to express the fact that a node in an instance of the graph pattern (at the meta-model layer) cannot recursively appear. As a consequence it is also not possible to preclude this in the instantiated tree at the model layer. In this case, on top of modeling these patterns in an ontological formalism, it is necessary to cater for the constraints between meta-model and model layers, and inside a layer, through a mechanism such as propositional logic.

One other important aspect that is required by description-driven systems and which must appear in an ontology, which has been abstracted from a description-driven system, is that of versioning. As stated earlier, versioning between multiple layers in a description-driven system must be asynchronous. However, versioning itself is not part of object-oriented languages and therefore propositional logic must be used to supplement UML in order to handle the constraints that emerge from the use of a versioning pattern [2]. Since the ontology has been abstracted from a description-driven system, it must cater for the set of design patterns that underpin multi-layer systems, including homomorphism, versioning, complex graph, complex tree patterns etc. Figure 7 shows a summary of the design patterns that emerge from a study of description-driven systems. The dependencies between patterns are shown as arrows - for example the versioned graph pattern uses the version, complex graph and publish/subscribe pattern

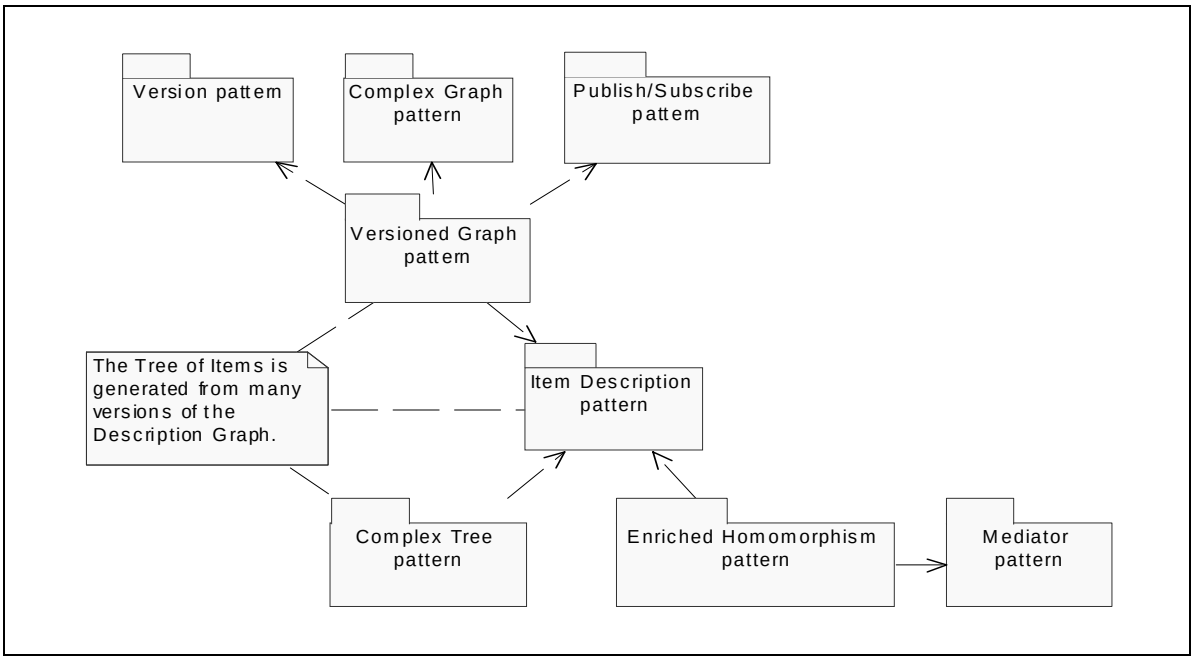

Figure 7: Description-driven system pattern summary.

(see [2]) - and there are constraints between these patterns (e.g. between the versioned graph, complex tree and item description patterns), as shown by the note in Figure 7. These constraints must be represented in and satisfied by the ontological representation, which, as stated above requires additional semantics to that provided by object-orientation.

Conclusions

In the CRISTAL project meta-models are used to provide self-description for data and to provide the mechanisms necessary for developing a meta-query facility to navigate multiple data models. Using meta-queries, data can be extracted from multiple databases and presented in user-defined viewpoints. The object models are described using UML which itself can be described by the OMG Meta Object Facility [10] and is the candidate choice by OMG for describing all business models. The CMS meta-model of Figure 4 acts as a repository of knowledge against which meta-queries are issued to locate and extract data across multiple databases. Agent processes are used to ‘look into’ the meta-model and extract data from a user-specified viewpoint and to ‘look out’ from the model to correlate effects between viewpoints. The overall effect is to produce an integrated set of cooperating databases accessed through a meta-query facility.

Work in the area of design patterns [3] is directly relevant to the ideas expounded in this paper. Foote and Yoder [11] have applied the concepts of pattern representations to the domain of data description. They conclude that candidate patterns are required to describe meta-data structures and their inter-relationships. Design patterns are thus needed in object-oriented design to describe meta-schemae such as CRISTAL meta-objects. Similar conclusions are being drawn by Riehle & Gross [12] in the field of design frameworks, where the framework behaviour is driven by repository-based descriptions and where descriptions of an organisation’s business operation is separated from the business application.

In conclusion, This paper has proposed that an ontology, which is rich enough to support multi-domain access from multi-purpose agents, can be abstracted from a description-driven system design. While UML provides sufficient expressive power to model software, certain restrictions, such as its inability to express formally constraints or functions, are serious impediments for the modeling of sharable

knowledge. UML as a language is therefore insufficient to provide sufficient semantics for agents to be able to access knowledge in the ontology and mechanisms such as propositional logic must be used alongside UML. Together UML and propositional logic provide the expressive power required for agents to exploit the ontology. Such an approach could reasonably be applied to organisations developing technologies for 'virtual enterprises' (such as in [13] and [14]) where collections of autonomous databases could be related via a central enterprise meta-model.

References

[1] W. Schulze, C. Bussler & K. Meyer-Wegener., "Standardising on Workflow Management - The OMG Workflow Management Facility". *ACM SIGGROUP Bulletin* Vol 19 (3) April 1998.

[2] Kovacs, Z., "The Integration of Product Data with Workflow Management Systems through a Common Data Model". PhD thesis, University of the West of England, 1999.

[3] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., "Design Patterns - Elements of Reusable Object-Oriented Software". Addison-Wesley Longman Publishers, 1995.

[4] The CMS Collaboration, *CMS Technical Proposal*. January 1995. Available from ftp:// cmsdoc.cern.ch/TPref/TP.html

[5] Estrella, F. et al., "The Design of an Engineering Data Warehouse Based on Meta-Object Structures". *Proc. of the Data Warehouse & Data Mining workshop at ER'98*. Singapore, November 1998.

[6] Baker, N. et al., "An Object Model for Product and Workflow Data Management". *Proc. Workshop at the 9th Int. Conference on Database & Expert System Applications*. Vienna, Austria August 1998.

[7] Blaha, M., and Premerlani, W., "Object-Oriented Modeling and Design for Database Applications". Prentice Hall Publishers, 1998.

[8] Leone, N., Rullo, P., and Scarcello, F. 1997. "Disjunctive stable models: Unfounded sets, fixpoint semantics and computation". *Information and Computation*, **135**(2):69-112.

[9] Bihlmeyer, R., Faber, W., Koch, C., Leone, N., Mateis, C., and Pfeifer, G. "dlv - an overview". In *Proceedings of the 13th Workshop on Logic Programming (WLP '98)*, October 1998.

[10] Object Management Group Publications, Common Facilities RFP-5 Meta-Object Facility TC Doc cf/96-02-01 R2, Evaluation Report TC Doc cf/97-04-02 & TC Doc ad/97-08-14 .

[11] Foote, B., and Yoder, J., "Metadata and Active Object-Models". *Proc. of the Int. Conference on Pattern Languages Of Programs*, Monticello, Illinois, USA, August 1998.

[12] Riehle, D., and Gross, T.,"Role Model Based Framework Design and Integration". *Proc of the 1998 Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'98)*, pp.117-133. ACM Press, 1998.

[13] Gaines, B., Norrie, D., and Lapsley, A., "An Intelligent Information System Supporting the Virtual Manufacturing Enterprise". *Proc of the IEEE Int. Conferencs on Systems, Man & Cybernetics*, Vancouver, Canada. October 1995.

[14] Hardwick, M., Spooner, D., Rando, T., and Morris, K., "Sharing Manufacturing Information in Virtual Enterprises", *Communications of the ACM* **39**(2):46-54, 1996.