CAESAR - AN ARCHITECTURE FOR ENTERPRISE MODELLING IN THE AEC INDUSTRY

Tore Christiansen, Principal Research Engineer,
Jan Thomsen, Senior Research Engineer,
Det Norske Veritas Research **AS.**
N-1322 Høvik, Norway.
Electronic mail: tchr@vr.dnv.no, jthom@vr.dnv.no.

Abstract

This paper reviews work in progress concerning information modelling in support of enterprise engineering, and discuses how important modelling challenges are being addressed to support more cost effective development of offshore installations for oil production in the Norwegian part of the North Sea. The paper describes a framework and a methodology for information modelling of real world project enterprises and presents initial application examples from the offshore oil and gas industry.

***CAESAR** Offshore is a research program undertaken jointly by Norwegian oil companies, engineering firms, research institutions and the Norwegian Research Council, with the aim of utilising information-technology-based methods and tools which lead to more cost effective field development and operation. As a part of **CAESAR** Offshore we are developing an object oriented system architecture consisting of a framework and methodology for information modelling, based on our belief that complete and correct enterprise models of development projects must include both the project requirements, deliverables, activities and organisation. Thus information models of projects must represent both the objective, product, process and organisation dimensions.*

Based upon a model of engineering design, we explicate and relate the enterprise dimensions, and outline a way of describing the difference between planned action and actual behaviour. We implement our model architecture according to an information meta-model, based on a set of common reference entities, and a general offshore reference model. In our work we are using the offshore reference model, as the basis for modelling offshore platforms, design of hydraulic system for offshore production units, and project control systems for engineering design projects.

CONTENTS

1. INTRODUCTION

This chapter describes the underlying motivation for the CAESAR Offshore research program, as well as our objectives for developing the CAESAR enterprise modelling architecture.

1.1 THE MOTIVATION FOR CAESAR OFFSHORE

The remaining oil reserves in the North Sea are mostly located in smaller fields where exploration and development must be performed in a more cost efficient manner in order to justify the probable life-cycle economy. This is expected to lead to a series of changes in future development projects, including new organisation of project teams, new contractual arrangements, different responsibilities and roles between project partners, extended reuse of standard solutions, reduction in the amount of technical and contractual documentation, and introduction of digital documentation according to current and future international standards. The challenge facing project managers is to reengineer their project practises while understanding how these changes are likely to affect the duration, cost and quality of field development projects. CAESAR Offshore is a research program undertaken jointly by Norwegian oil companies, engineering firms, research institutions and the Norwegian Research Council, with the aim of utilising information-technology-based methods and tools which lead to more cost effective field development and operation. To accomplish its aim CAESAR Offshore consists of a series of research projects addressing product modelling, use of standard parts libraries, technical information transfer from development to operation, and reengineering of work processes and organisations.

Gradual change may be viewed as a set of "linear problems", where it is, for example, possible to create slightly different products using the same process, or meet gradual technological advances by training organisational members. This world of "incremental improvement" has typically been the focus of many TQM efforts. Business reengineering, however, involves fundamental changes and may be thought of as a "non-linear problem", requiring simultaneous restructuring of all dimensions of the business. It is not possible to completely change business objectives or products while still using the same business processes, nor is it possible to change the business process without retraining and motivating the organisation. To handle such "radical change" requires an understanding of all relevant aspects of the project. Such understanding is enhanced by information models which describe the ***complete*** set of aspects and phases of the project.

Enterprise modelling aims to address issues in connection with the overall business process of human organisations. Real world enterprises consist of human actors, characterised by their ability to handle change by adaptation, and their ability to learn from experience. Learning thus causes organisational members to modify their objectives, which changes products and processes, and give opportunities for new learning. Consequently, enterprises undergo a continual process of change, and planned action is very seldom replicated exactly in actual behaviour. This makes it extremely challenging to develop enterprise models

which exhibit "validity over time." Since development of information models is costly and time consuming, we must be able to use the models for a prolonged time to make model development economically viable. Extended use facilitated by models exhibiting a mix of flexibility and stability. This ensures that model's validity is maintained, while adapting the model to account for change and thus continue to give a ***correct*** representation of reality.

1.2 THE OBJECTIVES OF OUR RESEARCH

As a part of CAESAR Offshore we are developing an object oriented system architecture consisting of a framework and methodology for modelling of field development projects in the offshore oil and gas industry. The objectives of our work is to develop a model architecture which enhances a complete and correct description of development projects, and may be used to enhance understanding of probable effects of proposed or anticipated changes in project requirements and execution. These objectives require that our model architecture supports building of project model representations which are (1) ***complete*** in their description of all aspects of projects, including the project *objectives, product, process* and *organisation*, and (2) ***correct*** in their description of both planned action and actual behaviour. These models may then be formalised and used to predict probable effect of suggested changes. That is, complete, correct and operational models which can be used by project managers as tools for decision support

The focus of CAESAR Offshore is field development. That is, AEC projects consisting of conceptual design, engineering, procurement, construction, installation, commissioning and hook-up. The information models should also be useful for transfer of information to the operation and maintenance phases of the life-cycle, and ideally serve as a basis for life-cycle support in operation.

1.3 AN OVERVIEW OF THE PAPER

In Chapter 2, we discuss some important requirements to enterprise models of development projects, and describe how these challenges have been addressed in the CAESAR enterprise modelling framework.

In Chapter 3, we describe how the enterprise modelling framework has been implemented as a methodology and meta-model for building information models.

In Chapter 4, we show how the model implementation has been used to develop a reference model for offshore field development.

In Chapter 5, we describe a set of information model applications built on top of the offshore reference model.

In Chapter 6, we summarise our work and list a series of topics that must be addressed in order to extend the framework, validate the methodology, and enhance further modelling.

2. ENTERPRISE MODELLING FRAMEWORK

In order to be useful for describing projects over any sustained period enterprise models must satisfactorily address a set of problems regarding representation of and reasoning about the complexity and uncertainty of the real world. These problems include *instability of requirements, inconsistency of interpretation, inaccuracy of behaviour,* and *incongruence of action.* Current information models of projects generally do not address these problems, with the consequence that models for planning and execution of project management can not readily meet the real requirements of the working project team Thus time and money spent in model development is often not very effective since the resulting project model can not be used to inform and support project execution.

2.1 ADDRESSING THE REQUIREMENT FOR COMPLETENESS

A majority of current information models are pure product models, process models or organisational models, and do not combine these dimensions of the enterprise. Consequently, these models have limited applicability for real world problems involving multidimensional issues. Traditionally, product models do not reference to the processes which create and modify the product. Process models on the other hand rarely describe products (i.e., the input to and output from process activities) in any meaningful manner for configuration management of the product over its life-cycle. Furthermore, product and process models normally do not contain any information about the responsibility, competence of the organisation. We must therefore strive for more general models, which have a more complete descriptions of the enterprise dimensions.

Models are representations of systems, which in themselves are models of reality. Therefore, model completeness is a circular and difficult concept, which requires a careful definition of the model contents, relative to the purpose of the model. In our work we define, a project as an enterprise describable in terms of an organisation (the project team), carrying out a process (activities described the project plan) to create one or more products (the project deliverables), satisfying some (set of) predefined objectives (the project goals and requirements). In order to represent these dimensions in a meaningful manner we must define they ways in which they relate. This is done in the ***CAESAR model of engineering design***, shown figure 1.

The figure illustrates how the design objectives are represented by requirements, and related to the design product in an extension of the well-known FUTS notation (Willems 88) which also explicitly includes design requirements. We are using a base-lining view design process, represented by activities, which can contain sub-activities, and which are related in a precedence network. Activities take (some version of) the design product as input and produces (a later version of) the design product as output. Thus, the complete life-cycle may be broken down into a set of life-cycle phases, related to the product life-cycle by base-lining. Various organisational actors in the design team are responsible for

carrying out the various activities, and thus also for the product deliverables from their various activities. The actors are related to one another by a set of formal and informal coordination relations for control and communication.

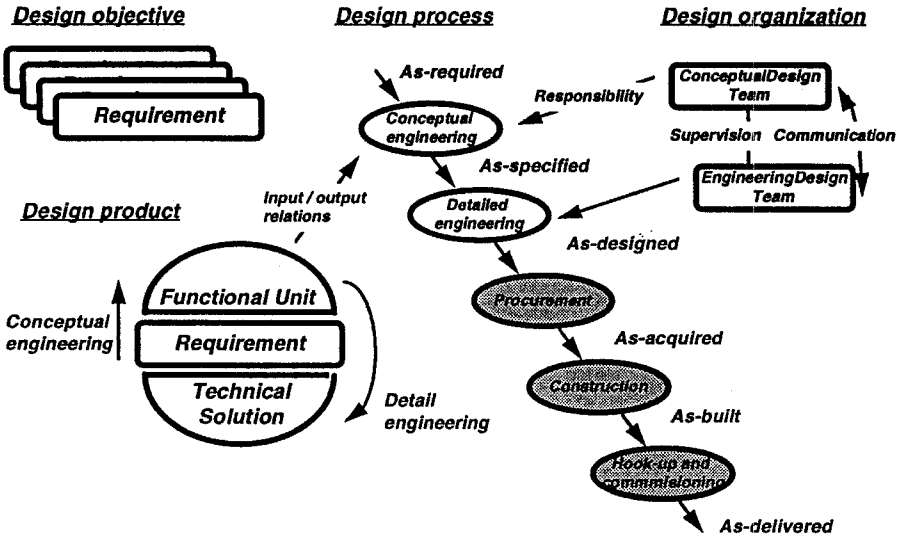

Figure 1: The CAESAR model of engineering design

*The **design objective** is described by the set of requirements to the project.*

*The **design product** is described in terms of these requirements, as well as the corresponding functional and technical solutions. The representation used is a revised version of FUTS "hamburger diagrams" (Willems 88); ReFUTS, where explicit representation of design requirements captures the essence of the design ("the meat of the hamburger"), thus capturing accountability and design rationale.*

*The **design process** is described in terms of design activities, resulting in relevant base-line versions of the design product. In each high-level activity (phase) there are various lower level activties related by precedence relations.*

*The **design organization** is described by the project team actors, their responsibility for design activities, and the various formal and informal coordination relations between actors.*

We may thus think of enterprise modelling in terms of a so-called OPPO notation (objective, product, process, organisation). We symbolise this multi-dimensional approach to building enterprise models in the general ***CAESAR enterprise modelling framework*** shown in figure 2.

We may also view this in terms of an enterprise modelling methodology, in which the project objective is systematically described in terms by functional decomposition into a set of requirements and associated solutions. That is, each requirement (at some level of detail) there is an appropriate solution (at the same level of detail). The set of requirements, functional units and technical solutions thus constitute models of the as-desired, as-specified and as-designed product,

respectively. The associated process description must include all relevant activities and the relevant sets of functional, logical and temporal relations between these activities. The description of the organisation must include all resources for carrying out the process. Thus the organisation model should represent the various actors, their responsibilities as well as the various formal and informal coordination structures (e.g., supervision, communication networks).

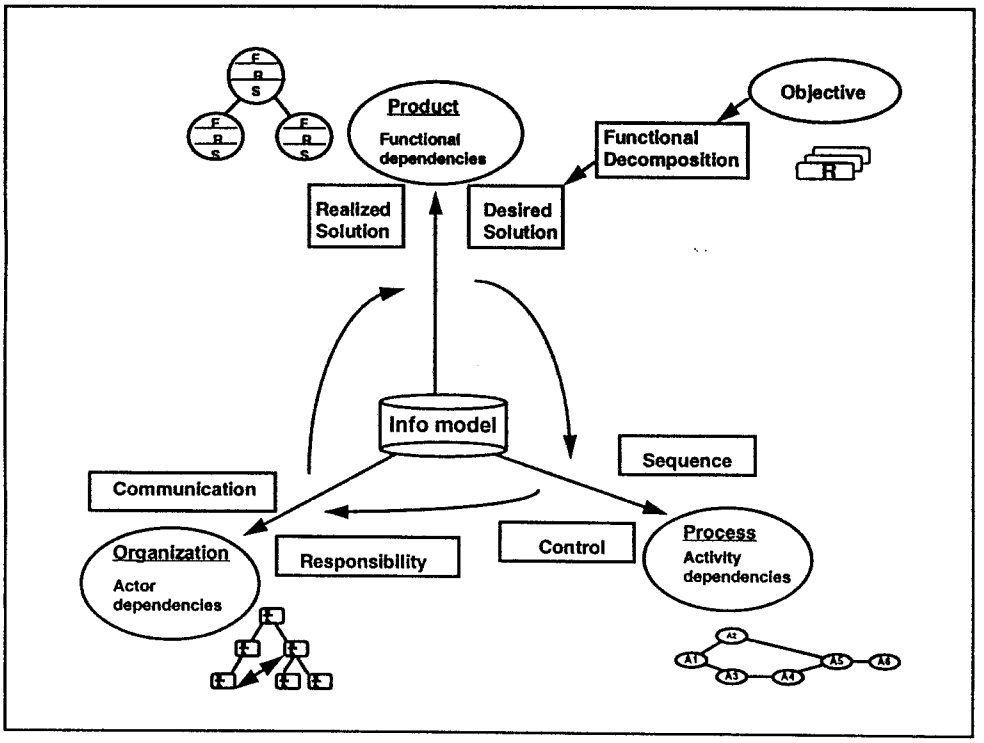

Figure 2: The CAESAR enterprise modeling framework

This figure illustrates how enterprise modeling includes both objective, product, process and organization dimensions. That is, from some stated (set of) objective(s), we may use functional decomposition to derive a requirement break-down structure representing our desired solution. We then translate this into a sequence of activities. Between these activities there exist various dependences, necessitating control of execution (a form of coordination). We assign responibility for the various activities to actors in the project organization. Because of the different formal and informal relations between actors, different needs for communication exist and must be addressed. Project execution results in some realized solution, which may or may not satisfy the project requirements (the desired model). A complete enterprise model of engineering projects should thus include all of these dimensions.

2.2 ADDRESSING THE REQUIREMENT FOR CORRECTNESS

To formally verify the correctness of a model we can give the model some set of inputs, and observe how this input affects the internal state of the model as well as the influence from the model to the external environment. Correctness means that both the internal and the external states should be equal to the observed state of the real world (according to chosen criteria). This requires recording of change,

and comparing the model representation and behaviour to see if changes in the model reflect changes in reality.

Maintaining correctness of the model over time involves both the explanations we use for the actual changes to the enterprise, and the flexibility of the model to allow representation of those changes. To describe the behaviour of real world enterprises, we may use different explanations to account for the relation between cause and effect. Figure 3 illustrates how project history may be understood in terms of either a rational logic of intention (March 88), an irrational logic of implication (March 88), or a boundedly rational logic of interpretation. It is this latter description which gives rise to causal omnidirectionality, obscuring the relation between cause and effect such that causality becomes a matter of interpretation according to chosen definitions.

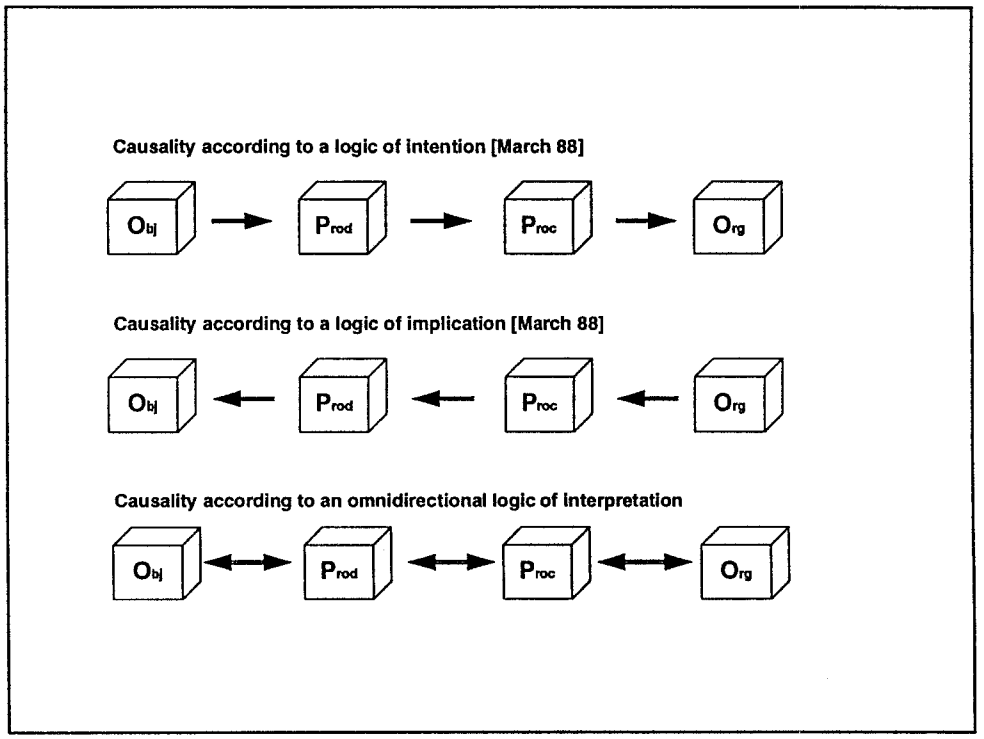

Figure 3: Different accounts of causality in project execution

In a rational world, a logic of intention prevails which explains projects as a set of objectives, for which we define one or more product deliverables, for which we define the required process, for which we employ suitable actors. In an irrational world goverend by a logic of implication we take the opposite view. Thus the set of available actors, with their preferences and historical relations, uniquely define which processes can (and cannot) be executed. This defines the resulting products, for which we then define some suitable set of objectives to explain the project results. Reality normally behaves as a combination of these two views, in which ratioal and irrational actions are interconnected. We denote this as an omnidirectional logic of interpretation, symbolized by two-way causal arrows. Note that the word OPPO is a palindrome. This symmetry symbolizes the way in which cause and effect is indistinguishable from a distance, in space or in time. That is, in reality the direction of causality is seldom clearly identifiable unless you know all relevant information about both the cause and the system and environment.

The implication for enterprise modelling is a need to capture the differences between a-priori specifications of intended information and a-posteriori descriptions of resulting information. This includes the differences between specifications (requirements) and resulting products (deliverables), the differences between project plans (prediction) and project execution (history), and the differences between planned action (policies) and actual behaviour (preferences). To describe such differences we propose a set of process measures of the difference between planned action and resulting behaviour. Figure 4 illustrates these so-called ***NODE measurements,*** which involve comparing normative plans and descriptive behaviour. An example of such a measure is the concept of verification quality used in the VDT simulation model (Levitt 94), which measures the degree to which policies for handling non-conformances by proper corrective action corrective action are followed up by project team members during simulation of the engineering design projects.

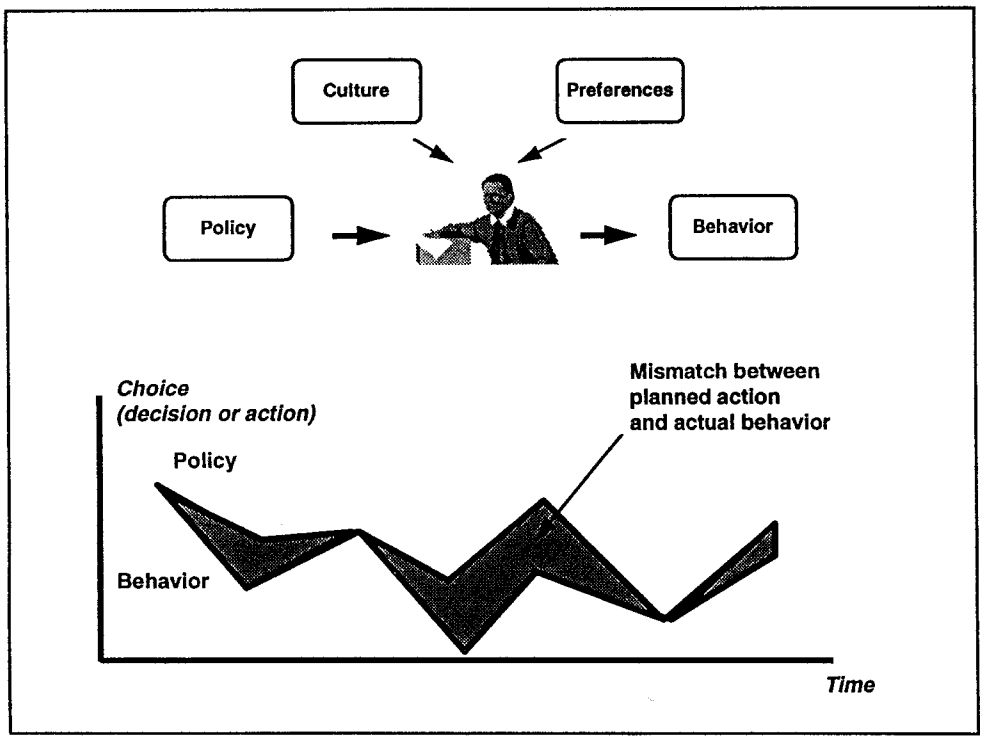

Figure 4: The NODE measures of coordination performance

In our attempt to develop models which exhibit validity over time in a world where complex processes are carried out by imperfct actors in uncertain environments, we have developed a set of measures for characterizing the differences between planned action and actual behavior. The former, so-called normative action, symbolizes decisions and action mandated by project policies and other underlying institutional boundaries. The latter, descriptive behavior, describes the actual decisions made and actions carried out during project execution. Our NODE process-measures are nominal measures of the difference between normative and desriptive action, and may be seen as measures of process performance. Current research (e.g., [Fergusson 94] has demonstrated the correlation betwen similar measures of process performance and end-user satisfaction. We may thus view the integral of the area between the curves as an aggregate indication of project quality.

By using the word "node" we wish to highlight the similarity between physical and organisational structures. In an abstract sense they both consist of elements, connected by nodes and exposed to external loading. Both structures have a capacity for withstanding load and a resulting response due to the load. In real structures it is the properties of the connections between elements, i.e., the nodes, which more than anything determine the response and performance of the structure. In an organisation structure this behaviour at the nodes takes the form of various means of coordination between actor elements.

2.3 ADDRESSING THE REQUIREMENT FOR LIFE-CYCLE SUPPORT

The products of modern engineering and construction are typically large and complex artifacts, representing major investments, and thus requiring a long period of useful operation in order to ensure proper amortisation and satisfactory return on the initial investment. Because of job market mobility the average period of employment is often longer than typical product life-cycles. Moreover, for large projects, there is often significant replacement of personnel during the project period. These are serious problems for traceability and capturing of design rationale, and therefore represents real treats to experiential organisational learning An approach to meet this challenge is development of models with the aim that these models should remain valid during the complete life-cycle

The CAESAR framework and methodology is based on our belief that complete and correct enterprise models of development projects must include both the project requirements, the project deliverables, the project activities and the project team organisation. That, in itself, should be a considerable help to enhance the validity of models over time, since the models may now contain information about reasons for change, results of change, contents of change, and the actual change action itself.

In addition, building the model is in itself a costly and time-consuming activity, and it is therefore it is not practical to constantly redevelop new models for all kinds of new purposes. We would ideally like to have an all-purpose model, which can be tailored to all kinds of modelling needs. This too is enhanced by a complete and correct model of the complete project enterprise.

3. INFORMATION MODEL IMPLEMENTATION

Having outlined our framework for enterprise models, we turn now to a discussion of how we are implementing the framework in a digital information model. This chapter outlines our modelling principles, reference definitions, modelling methodology and information meta-model. It should be pointed out however, that the implementation effort is an ongoing process, and thus not all parts of the implementation are complete or consistent.

3.1 THE MODELLING PRINCIPLES AND COMMON REFERENCE ENTITIES

In our view any information modelling effort should rest on a set of modelling principles, reflecting a consistent philosophy and approach to systematically describing reality.

Our first principle is to always have a clear distinction between model description and model instantiation. This principle, which may be summed up by the statement that ***a description of a thing is not the same as the thing itself.*** We implement the principle by the usual distinction between object classes, representing the model description, and object instances, representing the model instantiation. In particular, we explicate this distinction in our reference model by defining a separate class to represent the collection of all item instances in enterprise models.

Our second principle is to always keep in mind that the same information may be interpreted in many different ways by different individuals in different situations. This principle is summarised by the statement that ***any thing may represent anything.*** Since meaning is contextual, and since the context varies with participants and situation, the purpose of any model determines the model structure and content. We implement this principle by use of multiple inheritance to define different "aspects" to associate different sets of information with a given entity. That is, different sets of attributes, giving different types of information about an entity, may be inherited from different parent classes.

Our third principle is that enterprise models should be built from a set of basic entities, replicated and combined in order to represent more complex entities. This principle, which states that ***large and complex things can be described as a collection of simple things***, is of course a fundamentally reductionist view of reality. We implement this by using part-of relations to systematically compose compound object types by combining simpler (basic or compound) types. We can extend the reductionist description by adding new aspects, represented by object types which describe properties in the complex entity that are not represented in any of the simpler entities.

Based on these principles we describe a set of common reference entities (CORE). These reference entities include ***models***, which are the purpose of ***structures***, which are the function of ***types***, which are the description of ***items***, which are the

contents of models. These reference entities, which are implemented as object classes with inverse is-attribute-of/has-attribute relations, are shown in figure 5.

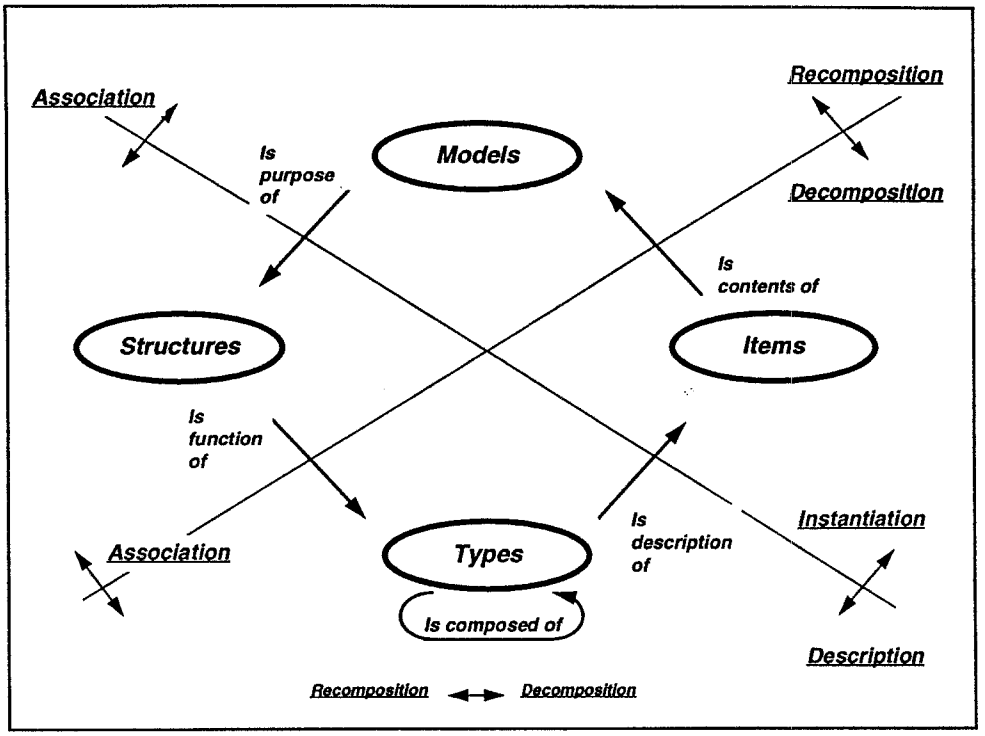

Figure 5: The Common Reference Entities (CORE) and their relations

***Models** describe the purpose of modelling, and have an associated **structure** at the class level, which is the function of a top-level **compound type**. The object types may be composed of any number of compound or **basic types**. Types are descriptions of **items**, which are the actual instance components of the model. Thus, models are composed of items, which represent real world entities. Items are described by types in a type hierarchy, using multiple inheritance to include attribute sets for any number of information aspects. At some level a type is assembled representing the complete model. The model structure points to the top-level type and the functional description of the model as represented by the various relations between types.*

The common reference entities are specialised as appropriate for the various enterprise model dimensions (objectives, products, processes and organisations). On top of these sets of dimensional definitions we describe the modelling resources, as sets of ***basic types*** to describe the set of possible basic constituents in a given domain. We combine the basic types into ***compound types***, and instantiate relevant types as ***items*** to represent specific real world objects (whether physical, informational or fictional). Sets of items collectively make up model intantiations, described by the set of object types and the structure of relations between them.

The creation of compound types takes place both through combination of basic types by the mechanisms of aggregation and/or multiple inheritance. That is, we may define a new compound types as a collection of simpler basic types. An example of this is the way in which a process plant is defined by the collection of process equipment and piping. Alternatively, we may specialise using multiple

inheritance to add new information aspects to basic types. Thus for example we may create a compound activity type for information processing by defining it as a subclass of flow objects with input output control and resource attributes, and successors objects with attributes describing precedence relations.

In particular, we view the function of the model as something held by the structure of relations between object classes. In our framework we have explicated this in the structure object class, which carries an attribute called function and one or more pointers to compound type (which in turn points to many other types). Reuse of descriptions is a useful mechanism both for describing reality and for creating information modelling. Using the common reference entities as basis for model building, a given type can be used in many structures (that is, have many functions) and a given structure can be reused for different purposes (i.e., be used to describe different models).

3.2 THE INFORMATION MODELLING METHODOLOGY

In order to define information models in a consistent and systematic manner we need to describe a methodology for modelling, using the common reference entities and relations described above. Thus, as indicated in figure 5, building models can be viewed as ***associating*** the purpose, function and top-level type to relate purpose, function and description, ***decomposing*** the type structure to describe the model contents, ***instantiating*** the various types to create items and ***recomposing*** the items to assemble the final model. An initial elaboration of this modelling methodology is illustrated in figure 6.

The model purpose is associated with the appropriate function and contents to define the associated structure and top-level compound type of the desired model. The structure can be reused from previous models, in which case the modelling is largely finished, or can be built up by decomposition until the type structure is complete. This model description can now be analysed by instantiation of the various object types (classes) as a corresponding set of items (instances). In this analysis select the appropriate attribute values for each instance, and analyse the items to ensure that their properties and behaviour is locally consistent. Then the items can be recomposed into the realised model, which should can be checked against the desired model.

On first sight figures 2 and 6 are similar in appearance. Note however that while figure 2 shows a meta-model of reality, and thus the objective is typically that of the project itself (e.g., "reducing field development cost" for the CAESAR Offshore program), figure 6 shows a meta-model of our modelling approach, and thus the objective here is to build an enterprise model which support (in some specified way) the project objectives.

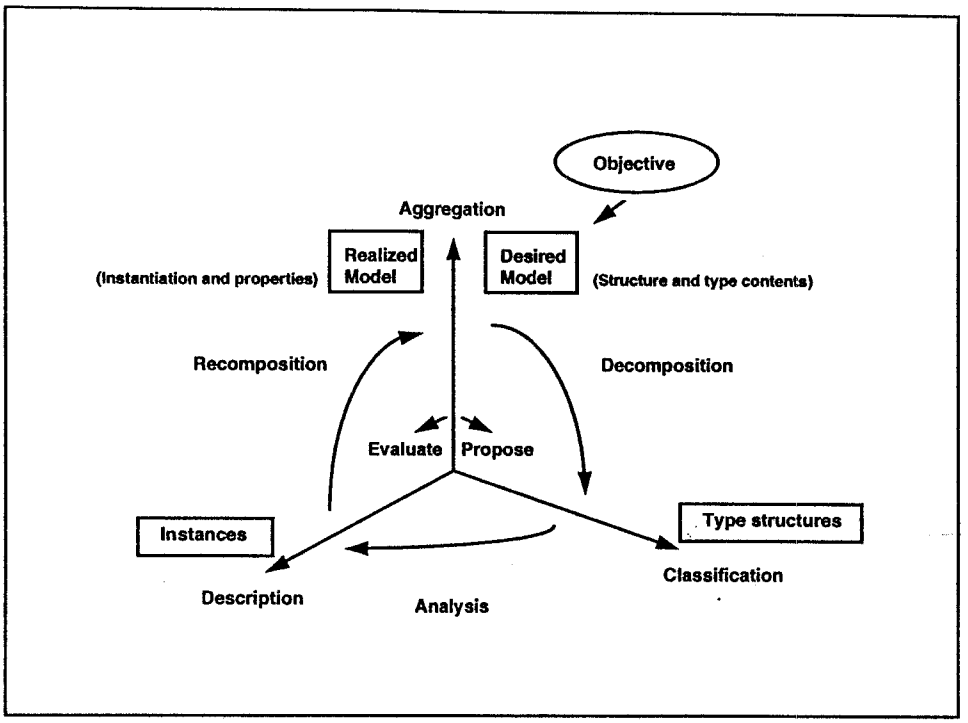

Figure 6: The CAESAR information modeling methodology

This figure illustrates our methodology for creating information models. Given some desired overall model purpose identify the appropriate function and decompose the model iteratively into its component types. These compoound and basic types describe the model in increasing detail and thus the decomposition is in effect a form of classification. We next analyse the various object types by instantiating the object types into a corresponding set of object instances representing real world entities. That is we describe (or analyze) the model constituents by assigning attribute values to all relevant attributes and verifying consistent properties and behavior of the items. Finally, we recompose the model components into an aggregate model in order to sum relevant attribute values for the component objects into global model values. Our information modeling methodology may be seen in terms of a decomposition-analysis-recomposition cycle, or a an evaluate-propose cycle.

3.3 THE INFORMATION META-MODEL

In order to use the framework and methodology to build real world applications we need to structure the information model for consistent definition, storage, maintenance and reuse. For this we have defined a layered information meta-model to describe enterprise models. This meta-model consists of a set of layers representing the complete model architecture.

The modelling principles and common reference entities discussed in chapter 2 are defined by layers 1 and 2 respectively. Layer 3 defines the enterprise dimensions (OPPO), while layer 4 holds the resources in the General Offshore Reference Model (GORM) a set of basic types for each dimension. These layers collectively make up the ***system level***, that is, the parts of a model which should normally not be altered during model building. The upper three layers are called the ***user level***, where the user may combine and extend the basic types into compound types at

level 5, in order to build a top-level type for which the function is defined by the various type relations in a structure object at level 6. At the application level (level 7) the type structures are instantiated as a set of items, given properties locally, and assembled to a model instance with aggregate behaviour. This information meta-model is illustrated in figure 7 below

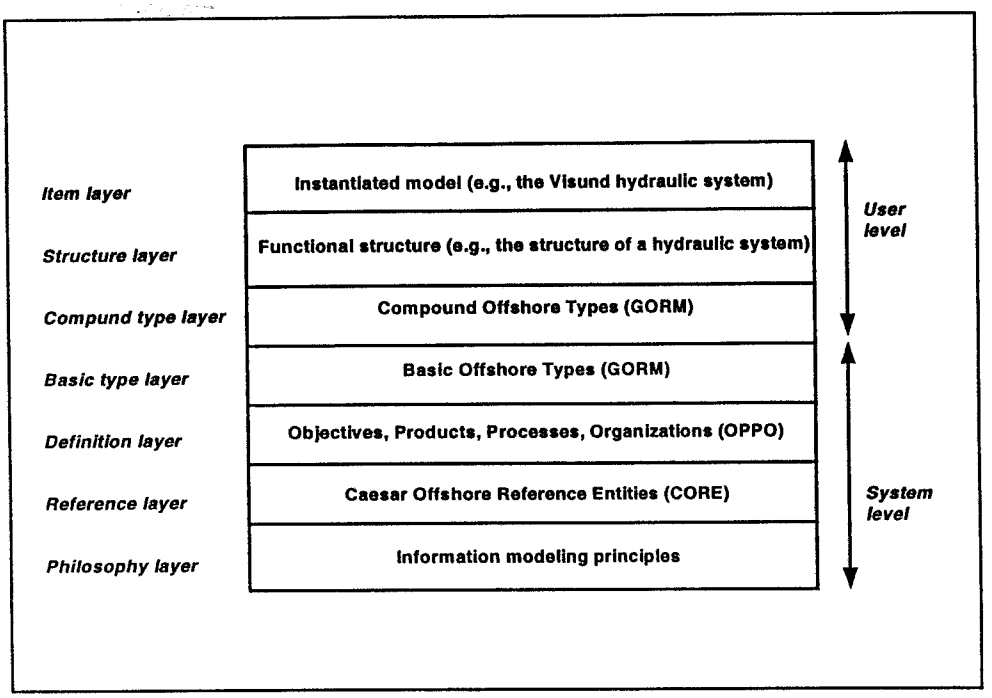

Figure 7 The CAESAR information meta-model

The meta-model is organized in a series of layers in a manner similar to the OSI-reference model for open systems interconnection. The ***system level*** *contains the foundation and basic architecture for defining various CAESAR information models.*

- *the* ***philosphy layer*** *defines the principles on which information modeling in CAESAR is based. This layer is not implementable, but is used to guide the implementation at other layers.*
- *the* ***reference layer*** *contains the Common Reference Entities (CORE).*
- *the* ***definition layer*** *contains the basic model architectures for modeling objectives, products, processes and organizations, using the basic classes in the CORE.*
- *the* ***basic type layer*** *contains the building blocks resources for describing the contents of a given domain. An example description at this layer is the General Offshore Reference Model (GORM), which is a reference model of object types used to build information models of offshore field development and operation.*

The ***user level*** *contains layers for describing, structuring and instantiating models*

- *the* ***compound type layer*** *contains combinations of basic types. These composite types can, in turn be combined to define other compound types.*
- *the* ***structure layer*** *is built by linking composite types with suitable relations to describe the various structures in a given domain.*
- *the* ***item layer*** *contains the instances of object types, and thus that represent the actual real world entities being modeled.*

Together, the contents in these layers define a complete CAESAR nformation model.

4. GENERAL OFFSHORE REFERENCE MODEL

Having presented the general modelling framework and methodology in the previous chapter we now apply it to offshore engineering projects. This chapter describes an initial version of a general offshore reference model (GORM), containing resources for modelling offshore development projects.

4.1 THE REFERENCE ENTITIES FOR OFFSHORE PROJECTS

In the GORM we define the building block components for enterprise modelling of offshore development projects. These building blocks are part of the definition layer of our meta-model, ***abilities*** (objective components), ***artifacts*** (product components), ***activities*** (process components) and ***agents*** (organisational components). The primary relations between the various building blocks are that activities consume and produce artifacts, while abilities control activities and agents support activities. This description is consistent with the general CAESAR model of engineering design described in chapter 2. The GORM definition is shown in figure 8, represented in the OMW object-oriented CASE-tool (TM Intellicorp, Mountain View, California).

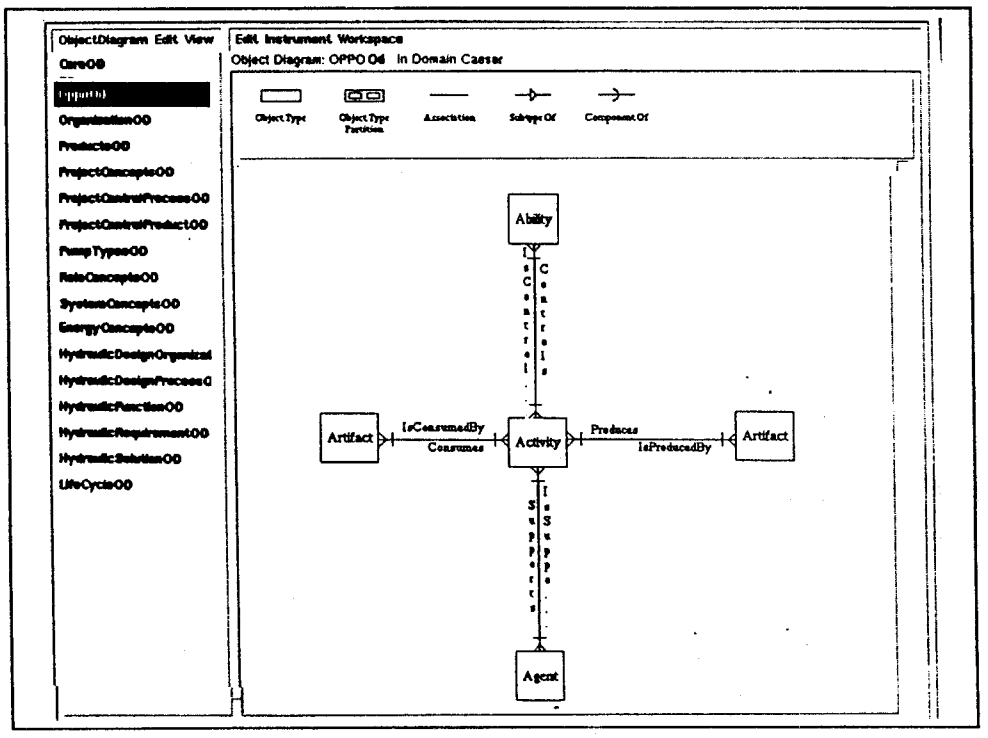

Figure 8: Basic entities for the general offshore reference model
The figure shows how the information modeling framework has been used to define the required elements of the reference model in tems of the various items describing objectives - abilities, products - artifacts, processes - activities, and organizations - agents. ***Abililties*** *represents the desired outcomes of the project.* ***Artifacts*** *are created to meet abilities.* ***Activities*** *have precedence relations (predecessor-successor) to other activities.* ***Agents,*** *support activities end up with responsibility for the artifacts resulting from the activity.*

4.2 THE BASIC TYPE RESOURCES

We use the general model of engineering design presented in figure 1 above to guide our definition of products as a collection of artifacts with appropriate versioning to account for life-cycle history. That is, a product in offshore development is anything which requires configuration management, i.e., typically anything which is tagged. Moreover, products (tagged items) may be composed of other products (tagged items) and/or collections of lower level (untagged) components.

Figure 9 shows a set of GORM basic types for describing artifacts. These basic types, which may be thought of a set of classifiers, include product types (partly shown on top), version types (as-required, as-designed and as-built), artifact content types (physical and informational), artifact types (facilities, systems, etc.) and function types (according to discipline). These basic types are not meant to be a complete set, but are included as indications of the type of descriptors needed. The figure is a picture from the Object Browser of Kappa (TM Intellicorp, Mountain View, California), an object oriented development tool which has been used for implementing all of the model architecture.

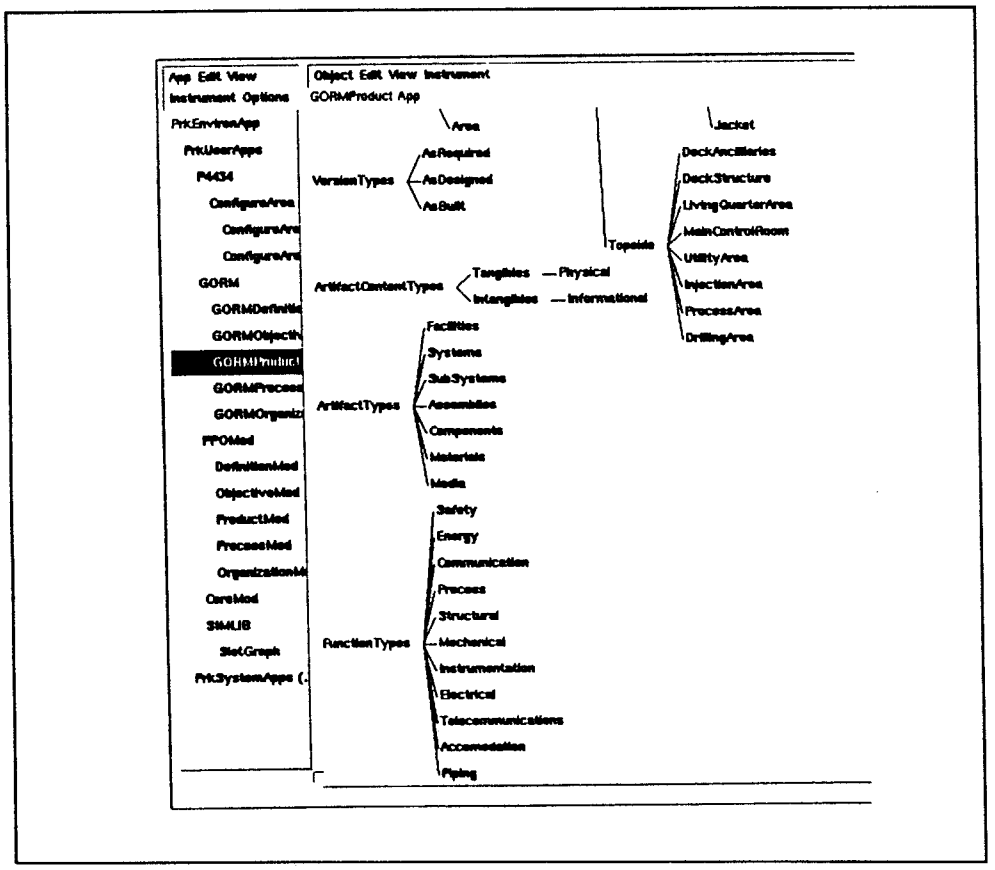

Figure 9 : A product model reference for offshore field development

The figure shows a selection of basic types in the GORM resource set for describing artifacts.

Similarly, we can think of the process as a collection of activities, each of which can contain sub activities, and between which there exist a set of precedence relations. For example, offshore engineering is divided into concept, basic and detail engineering phases, where concept engineering is a predecessor of basic engineering, which is a predecessor of detail engineering . Each of these phases are divided into activities for each functional or geographical module, which are in turn divided into sets of activities for the various disciplines. The various discipline activities are typically subdivided in a hierarchy of sub-discipline levels. Thus, for example, conceptual engineering is divided into activities for processing facilities, deck-structure, subsea systems etc. Each of these sets of activities are divided into disciplines like process, electro, automation, safety etc.

Figure 10 below shows a set of GORM basic types for describing activities. These include life cycle phase types (in which the activity takes place), standard activity types (from a typical standard activity break-down structure of oil companies), flow object types, content types, precedence types. The various basic types have attributes describing the function and behaviour of its members. For example, any activity item which is of type flow object is described by its input, output, control and resources. For activity instances these relational attributes are values pointers to artifact or agent object instances.

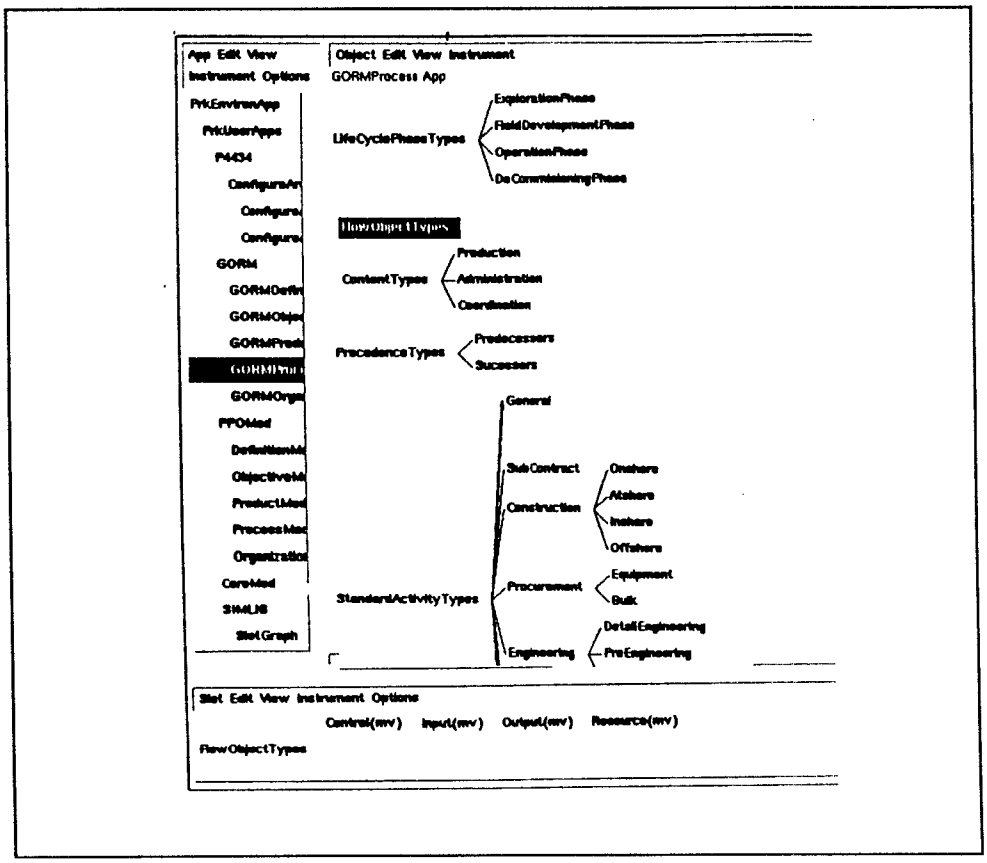

Figure 10 : A process model reference for offshore field development

The figure shows part of the GORM basic types for describing activities.

The organisational concept in the reference model includes all resources required for carrying out offshore projects. That is, agents may be either (human) actors capable of carrying responsibility (a special form of support), or one of the various (computer and other) tools used by actors to carry out activities according to their responsibilities. Between the various agents there may be supervisory and/or communication relations. An illustration of this organisational model is shown in figure 11. A similar set of GORM basic types exist to describe actors, including actor type (a list of possible roles in offshore development), resource type (a list of the different types of resources) and relation type (competition, control, communication, verification, competition).

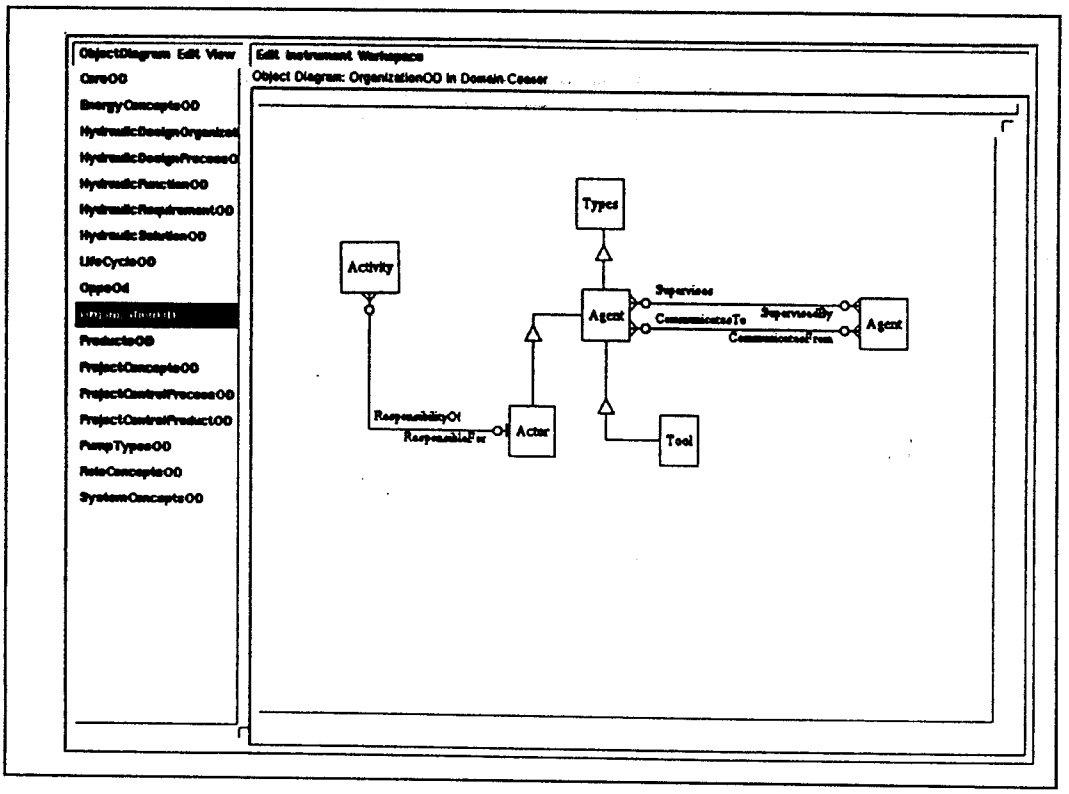

Figure 11 : A organization model reference for offshore field development
The figure shows the GORM resources for defining compound agent types.

4.3 A COMPOUND TYPE DESCRIPTION FOR CONCEPT DESIGN

The various basic types in GORM may be used to define compound types, by combination of basic types. In the current implementation this is done by creating subclasses as children of several basic classes. Through multiple inheritance the compound classes will inherit the various attribute sets (with any defined default values). Thus the various basic classes (parent classes) may be thought of as different aspects of the compound (child) class.

Figures 12 gives a simple example of this by showing how a set of compound activity types for configuration design have been described by combining the flow object type description (input, output, control and resource attributes) and the successor precedence type (precedence attributes). Thus these engineering activities have a flow (IDEF-0) aspect and a precedence (project plan) aspect. These compound activity types may themselves be used to define new compound types.

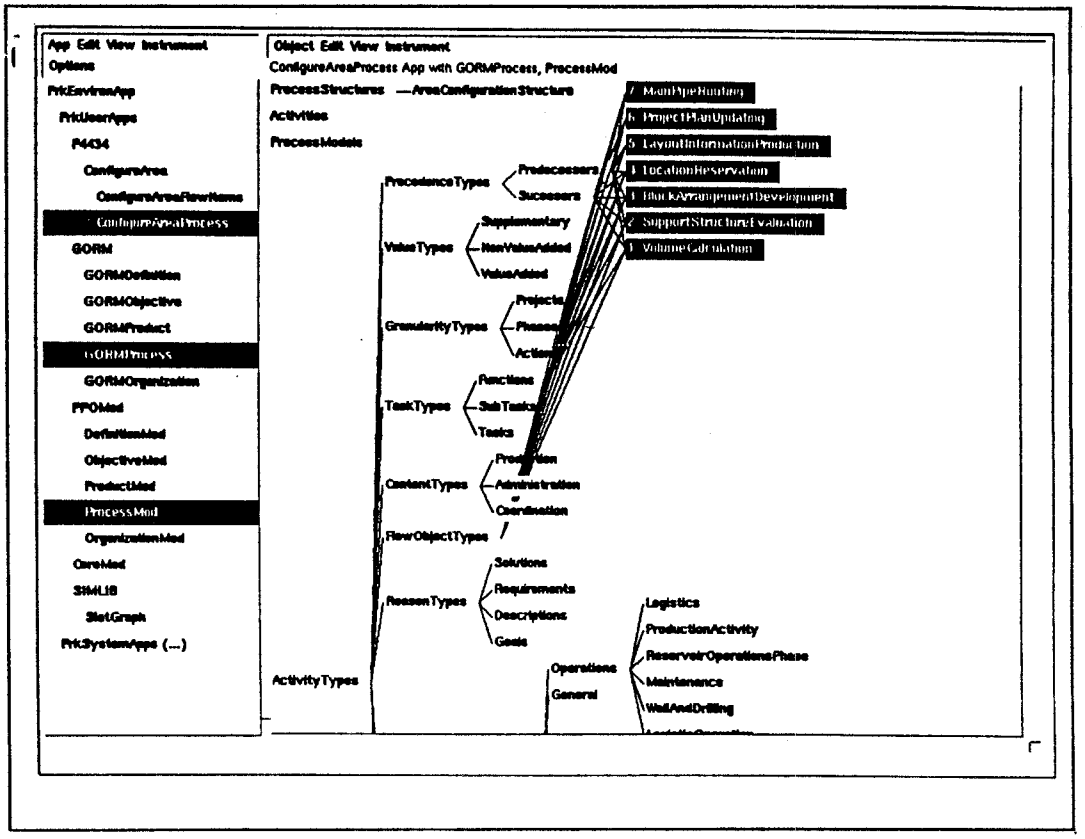

Figure 12: Compound activity types for describing configuration design

The figure shows a set of compound types, created by combining GORM basic activity types .

Figure 13, likewise, shows how a set of compound artifact types have been created to represent the product of the configuration design activities by combining the informational artifact content type with the relevant version type. These artifact types thus have both a content aspect and a versioning aspect both of which may be used to support configuration management.

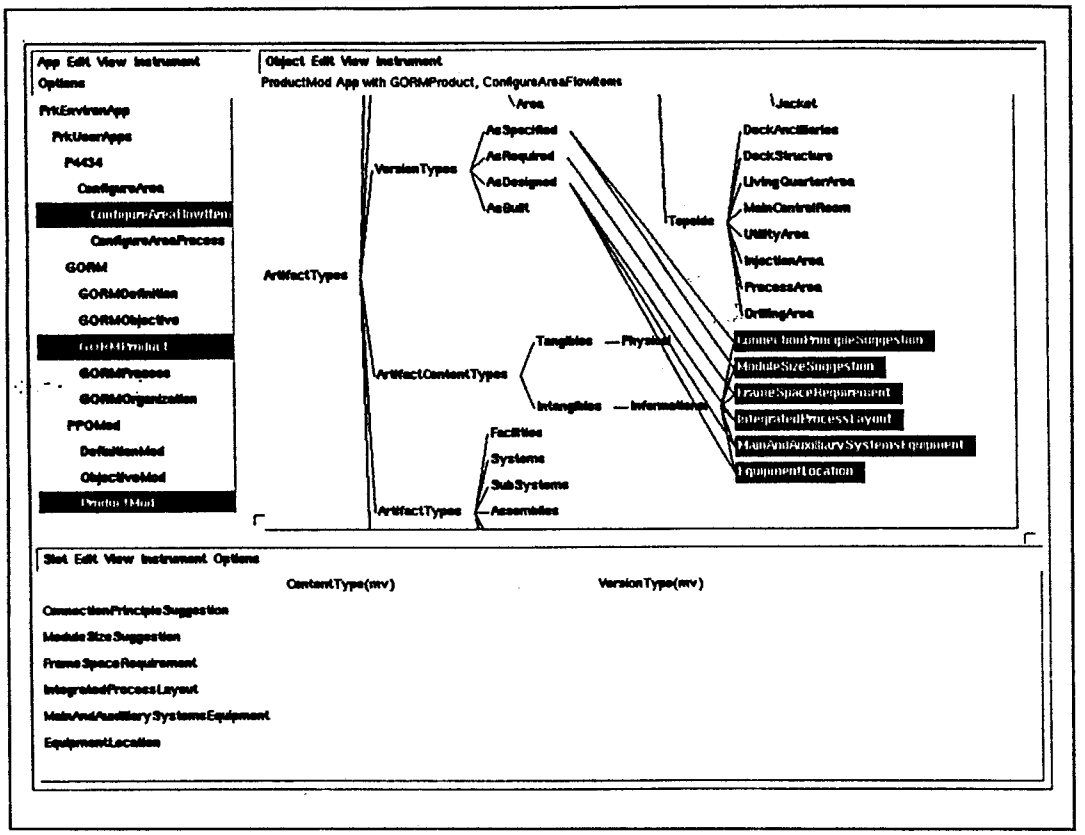

Figure 13 : Compound artifact types for describing configuration design

The figure shows a set of compound types, created by combining GORM basic artifact types.

5. APPLICATION TO OFFSHORE DEVELOPMENT PROJECTS

In this chapter we ***present*** a few preliminary examples of how we hope to use the framework and methodology for modelling offshore development projects

5.1 A MODEL OF JACKET PLATFORM DESIGN

This model is an attempt to represent an overview of the necessary information for planning of conceptual design and detail engineering of an offshore jacket platform. It is an application of the general offshore reference model (GORM) to develop a high level description of all main elements in field development. We foresee that such a model may be used to support life-cycle support, configuration management and experience feedback. So far we have developed simplified representations of, the product structure, including specialisation and decomposition, the process structure, including precedence between activities, and the organisation structure, including supervision and communication. We have not yet represented the requirements structure, including functional, economical, temporal, environmental and safety requirements.
Figure 14 shows how the artifact items of the platform are instantiated from compound artifact types, which themselves are specialisations of more general basic artifact types. For example, the actual Jacket Support Structure is an instance of Support Structures (compound type), which is a subclass of systems (basic type) which is a subclass of Artifact Types.

Figure 15 shows the decomposition hierarchy of the same Jacket Support Structure, defined by Part-of relations between the various instances. Thus for example, we see that Can112 is part of Joint1, which is part of Leg1, and so on. Normal over-riding is used to specialise attribute value descriptions of the various classes, sub-classes and instances have. That is, for example, the Material of Leg 1 is HighGrade8081q1 steel, which is a specialisation of the default value "Tensile Steel" value of the Material attribute of the Legs class. Note also, in this particular case, that the value at the instance level is actually a pointer to another instance. In our model representation we make full use of descriptive values, relational pointers to other objects or attributes, and behavioural pointers to methods.

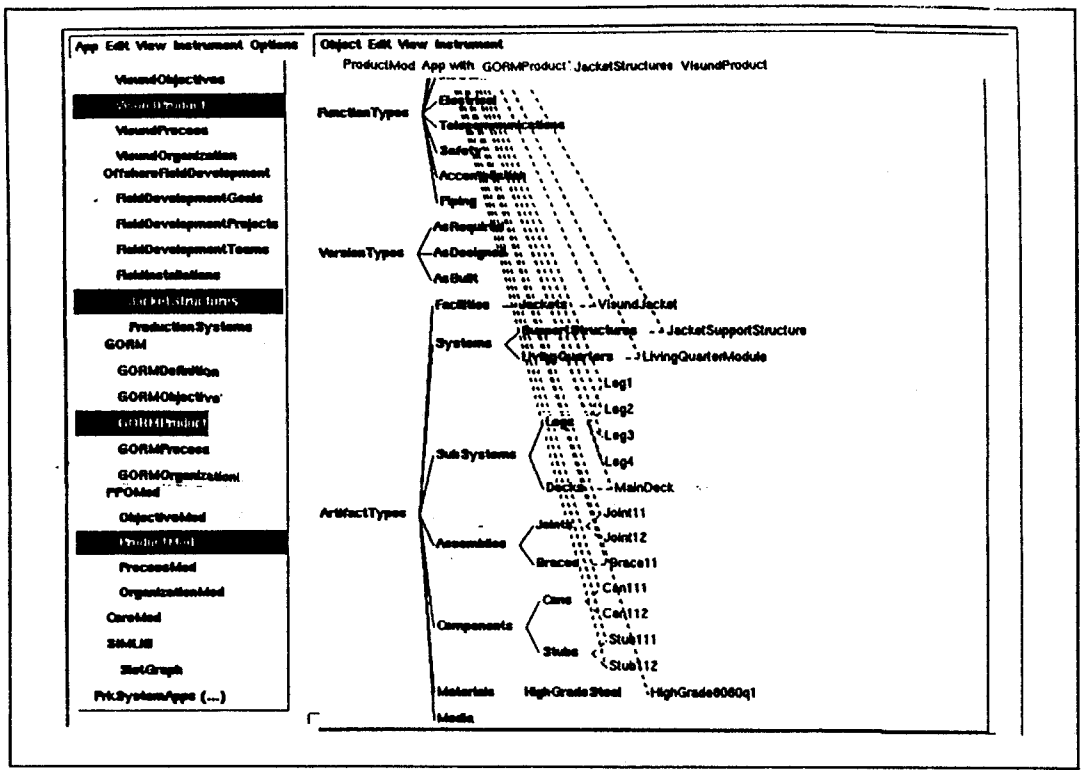

Figure 14 : A product model of offshore field development

The figure shows the specialization hierarchy for part of a product model of a jacket plattform. The product information is a technical solution version (i.e., this model represents information at some stage of engineering design).

The model is built using the CAESAR modeling architecture , and thus -

*- **basic types** lie to the left (e.g., "artifact types")*

*- **compound types** lie in the middle (e.g., joints and braces), and are defined by multiple inheritance from several basic types (specialization by combination), in which different attribute sets are inherited.*

*- **artifact items** (e.g., leg 1) are defined as instatiations of the relevant object types. These items are also instances of the "physical" artifact content type (outside of the figure).*

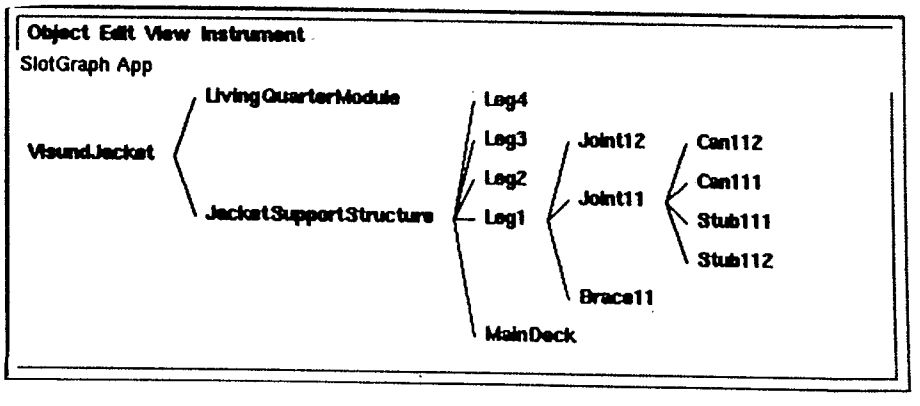

Figure 15 : A product model of offshore field development

The figure shows part the decomposition hierarchy of the a jacket plattform, model in figure 14, with a set of part-of relations connecting the various object instances.

In figure 16 we see a set of engineering activities to design a template support structure as part of the subsea production system. The network shown is that defined by a-priori project planning, and illustrates the high degree of concurrency required by the client's temporal requirement to shorten the total development time drastically. Thus a lot of analysis activities traditionally carried out in sequence must now be carried out simultaneously. This, of course, makes it necessary for the various design team members to coordinate closely during project execution, and leads to intense "coordination load" (Christiansen 94). Note that this is the same representation as used in the VDT (Levitt 94).

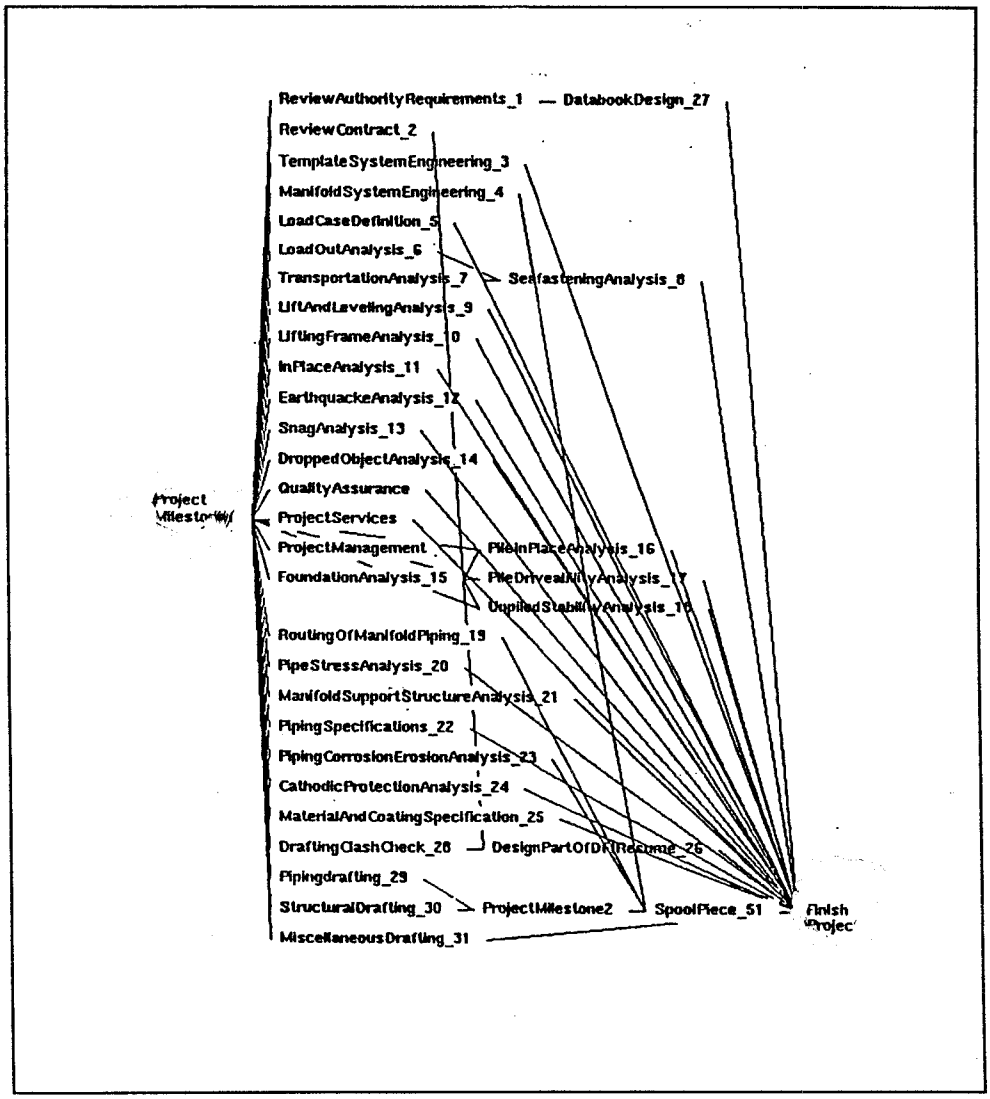

Figure 16 A process model from offshore field development

The figure shows an example of engineering activities for a subsea oil production module. The various activities are connected by predecessor-succesor relations. A can be seen, the present design process involves a great deal of concurrency, as typical for engineering projects in offshore field development.

In figure 17 below we see the formal supervision hierarchy for the part of the project team responsible for carrying out the subsea template engineering activities (in figure 16). Thus, the client, supervises the project manager supervises the various subteam leaders, and so on. This also is the same representation as used in the VDT (Levitt 94). The various team members are represented as instances of the actor class (basic type), with attributes to describe their capability and capacity, and their various policies and references for action. Some of the relevant attribute values are default for all members of the project organisation (e.g., the project organisation to which they belong), inherited by the instances from class level definitions, while other values differ from actor to actor and are defined at the instance level (e.g., craft specialisation and level of skill)

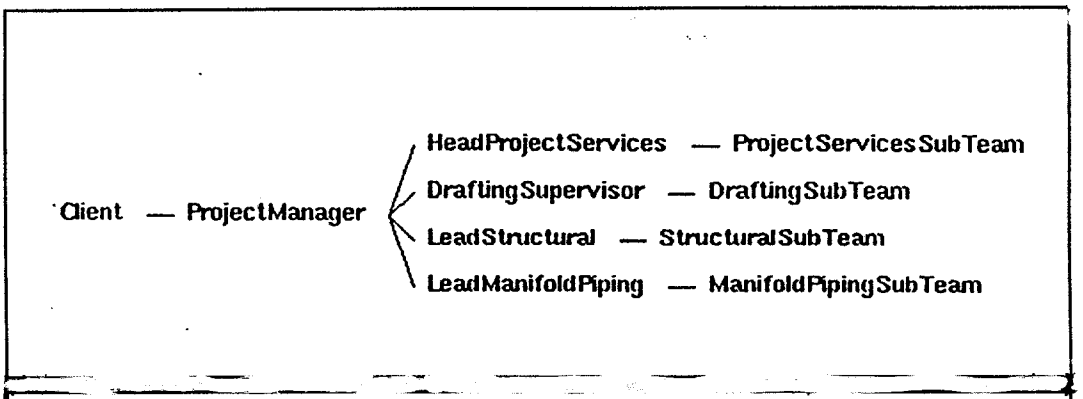

Figure 17: Organization model for offshore field development

This figure shows the formal hierachy for part of the engineering design team of the deisgn process of figure 16. The various actors represent either individuals or subteams, and are connected by formal supervision relations (shown) and informal communication relations (not shown).

5.2 A MODEL OF HYDRAULIC SYSTEM DESIGN

This model is a representation of the design process for hydraulic systems on offshore production platforms. The model includes representation of the various design requirements, which is a top level hydraulic energy requirement, decomposed into requirements to produce, store and distribute energy. Next is a representation of the functional units and technical solutions, The functional units describe the production of hydraulic energy by volume and pressure changes, while the technical solution is described as a topological breakdown structure. This completes our ReFUTS description (see chapter 2) of the hydraulic system.

As illustrations of this model, figure 18 shows a library of standard pumps which may be used in pump dimensioning, while figure 19 shows the design process activities. Note that the various pump candidates are instances of a compound artifact type in GORM, while the design activities are instances of compound activity types.

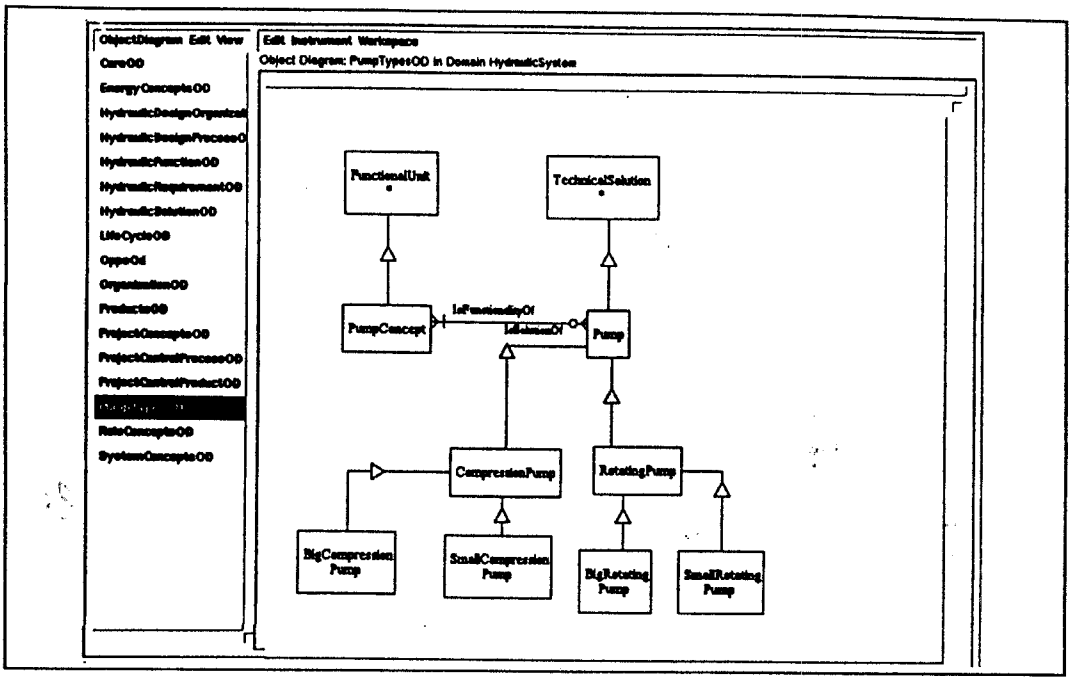

Figure 18: Product model versions for the hydraulic system

This figure shows the different life-cycle versions of part of the hydraulic system. The functional unit "pump conept" is in response to the requirement "produce energy". The pump concept is solved by the technical solution "pump". The figure also shows how the pump may be one of several types of pumps., The specialisation hierarchy may be viewed as a library of reusable stadard parts which may be tried out in the design process.

The design process is represented, in our current OMW (TM) implementation, both as a set of activity instances and as a set of process operations. Thus the design process can be simulated to check various aspects of process logic. The simulated scenario includes capacity calculation, pump dimensioning and design approval, carried out by the client's conceptual design team, hydraulic supplier and certification body, respectively. Thus a scenario simulation may be set up where the hydraulic supplier specifies a pump which is too small for the specified power requirement, causing the certification body to refuse design approval, and forcing the hydraulic supplier to repeat the pump dimensioning for a larger pump.

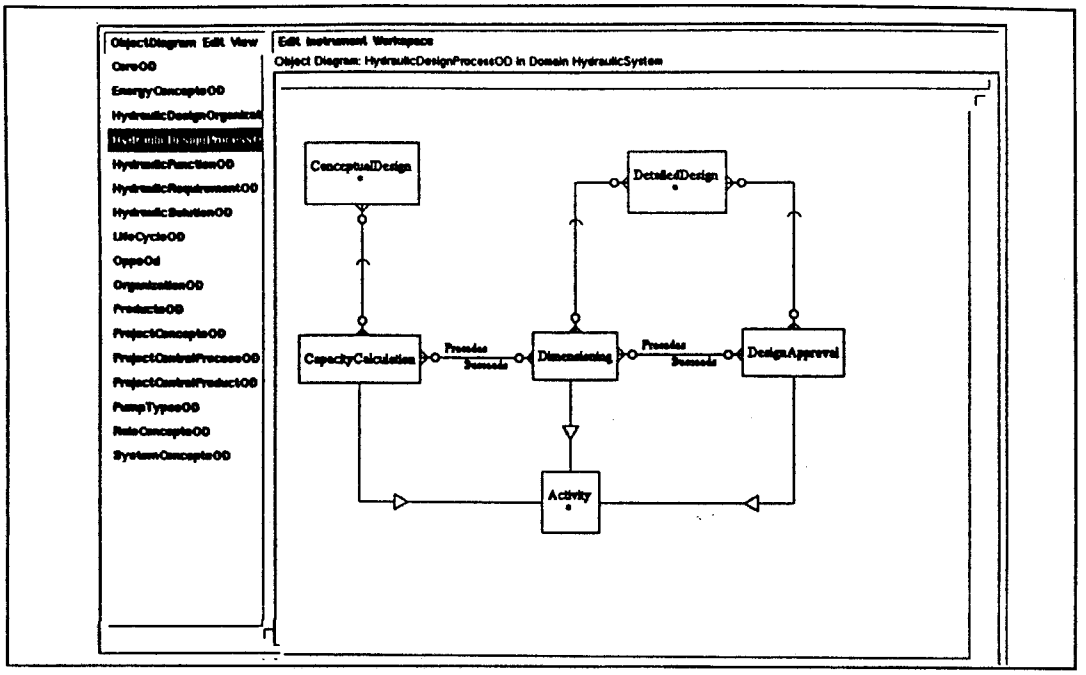

Figure 19: Hydraulic system design process for offshore structure

This figure shows a model of a set of activities in the conceptual and detailed engineering design of a hydraulic system for processing oil and gas onboard an offshore plattform. The picture shows three design activities (capacty calculation, dimensioning and design approval), which are part-of conceptual and detailed design, and instances of the activitiy item class.

5.3 A MODEL OF PROJECT CONTROL

This model is currently being developed in order to illustrate the process of project control in engineering design projects. The current model describes the flow of information between planning, monitoring, forecasting and management, and is intended to be used as a tool to study how different means for project control affects and is affected by the progress of various project activities.

Figure 20 shows the an outline of the process model to be simulated logically in OMW (TM), with relevant input and output to control and project activities. These inputs and outputs may in fact be thought of as instances of informational artifact types, constituting a "product model of project control" as illustrated in figure 21.

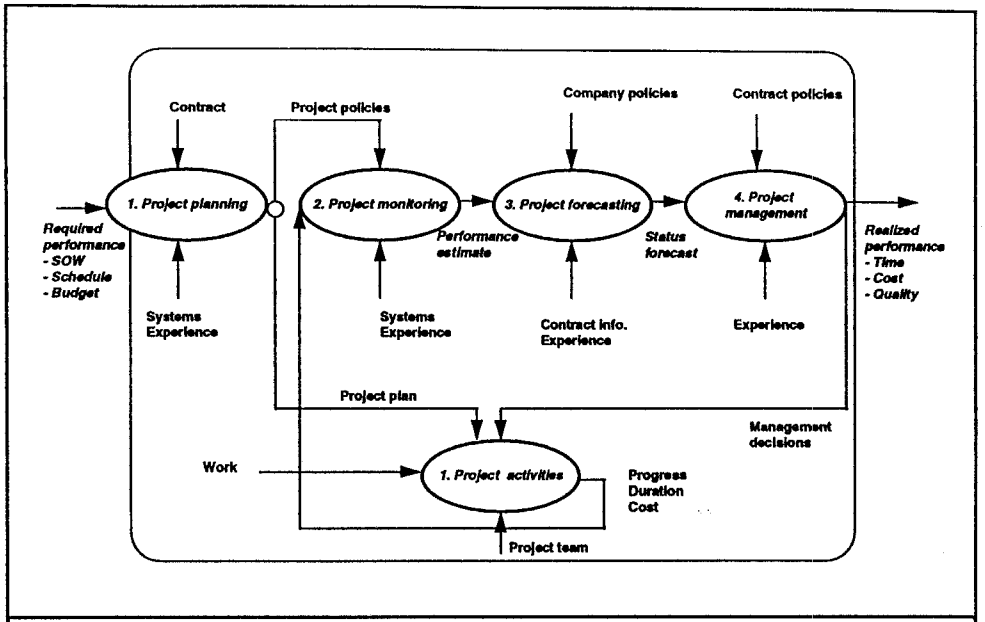

Figure 20: The project control proces

This figure shows a picture the process model we are developing to simulate project control, consisting of planning, monitoring, forecasting and management, with precedence and indications of input, output to activities as well as relevant control and resource elements. The model shows various causal dependencies and includes a representation of project activities. It is our intention to integrate the project control model with other models of project content, such as for example the hydraulic design model described above.

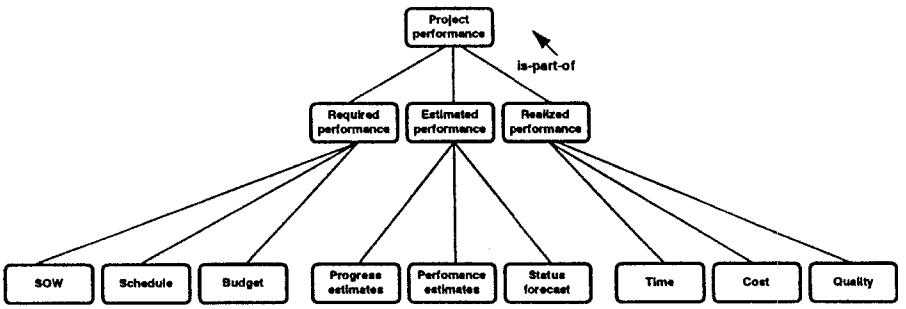

Figure 21: A product model representation of project control

This figure shows the "product model" for project control, in terms of the various versions of the top-level "product", project performance. The as-required, as-estimated and as-realized versions of performance has been decomposed into their various components.

6. CONCLUSIONS AND FURTHER WORK

In this paper we have ***defined*** the framework and methodology for our model. We explicated our definition and ***represented*** it in a suitable object oriented model building environment (OMW, TM and Kappa, TM). We then ***presented*** a few initial example models. In the continuation of our work we must now ***apply*** the framework and methodology to model real world project applications in offshore engineering. It is our hope that these models may then be used for a variety of different purposes including -

- as reference models (e.g., the GORM) for discussion, understanding and agreement between project participants in different roles and disciplines, and between different project partners.
- as descriptions of particular project instances (e.g., the engineering design of hydraulic a system for a given field installation) for planning, optimisation and documentation.
- as executable models for studying logic and rationality of complex systems (e.g., project execution and control system), and thus guide in project execution.
- as executable models for predicting probable effects of proposed changes (e.g., the VDT simulation system (Christiansen 93) (Levitt 94)) before committing resources to carry out the change.

Much work remains before the framework and methodology presented here can be applied to practical problems of reasonable size and complexity. First and foremost we must extend the model representation to include all relevant information for one (or a few) chosen application examples. Also, we must extensively test the suitability and usability of both the framework and methodology by subjecting using them to build models in different types of domains. This includes, importantly but painfully, letting other engineers take part in model building, with and without (!) guidance. In this connection there remains considerable work to be done in terms of documentation.

We plan to integrate the representation used in the general offshore reference model (GORM) with the representation in VDT (Levitt 94). It should be possible to simulate the various engineering design process models to see how given changes may affect duration, cost and quality of design projects.

In the CAESAR Offshore program we intend also to extend the CAESAR model of engineering design to encompass the complete product development, including procurement, construction, hook-up and commissioning. Future work may also be carried out to include operational phases (including maintenance, updating, de-commissioning and scrapping).

We feel, however, that even the major challenges are yet to be tackled, we have made some progress towards defining and implementing a usable framework and methodology for modelling reality in offshore field development.

ACKNOWLEDGEMENTS

We would like to thank several of our colleagues and contacts for contributing their insight, support and many helpful discussions. They include -

- Bjørn Egil Hansen and Janne Krogh in CAESAR Offshore Support Project 1.

- Tor Sverre Brynildsen, Tor Johan Kristiansen and Lars Christensen in CAESAR Offshore Core Project 4.

- Nils Sandsmark, the CAESAR Offshore Program Coordinator

- Ray Levitt, Yan Jin and John Kunz from the Center for Integrated facility Engineering (CIFE) at Stanford University.

REFERENCES

(Christiansen 93)
Christiansen, T.R.
"Modeling effieciency and effectiveness of coordination engineering design teams
PhD Thesis, Civil Engineering Department, Stanford University, Oct. 1993
Published as DNV Research report no. 93-2063

(Christiansen 94)
Christiansen, T.R.
"Modeling coordination load distribution in the VDT"
DNV Research report no. 94-2000

(Fergusson 94)
Fergusson, K. J. & Teicholz, P.
"Facility Quality Measurement as the Engine of Continuous Product and Process Improvement in the AEC/EPC Industry"
First Int. Architectural/Engineering & Construction Division Conference

(Levitt 94)
Levitt, R.E., Christiansen, T.R., Cohen, G.P., Jin, Y., Kunz, J. & Nass, C.I.
"The Virtual Design Team:
A computational simulation model of project organizations"
To appear in Management Science

(March 88)
March, J.G.
"Decisions and organizations"
Oxford, UK, Basil Blackwell, 1988

(Simon 58)
Simon, H.A.
"Administrative behavior"
New York, Macmillan

(Thompson 67)
Thompson, J.
"Organizations in Action: Social Science Bases in Administrative Theory"
New York, McGraw-Hill, 1967

(Willems 88)
Willems, P.
"A functional network for product modeling"
PLI-88-16, IBBC-TNO, July 1988