How are Core Industry Solutions built?

Conventional applications are supported by IT systems built by waterfall methods supporting routine information processing based on formal methods, rather than creative design work. Analysts cry out that "IT doesn't matter", because IT does not extend the capabilities of the core of business or practice.

The *Active Knowledge Modelling (AKM*) methodology is the foundation for a more agile approach to solutions based on adaptive design methods and evolutionary IT infrastructures. Using AKM methods project workers and service providers can adapt, extend, and reconfigure solutions and services for innovative cyclic design and sustainable life-cycle support and knowledge reuse and management.

Solutions designed and delivered range from computing platforms and workplaces to configurable systems and visual design solutions and architectures, built by hardware, software and AKM models. AKM models are new solution components extending traditional ICT systems to involve practitioners and pragmatic enterprise knowledge models and workspaces.

The table below summarizes the main aspects/methods of the AKM methodology applied in most sectors.

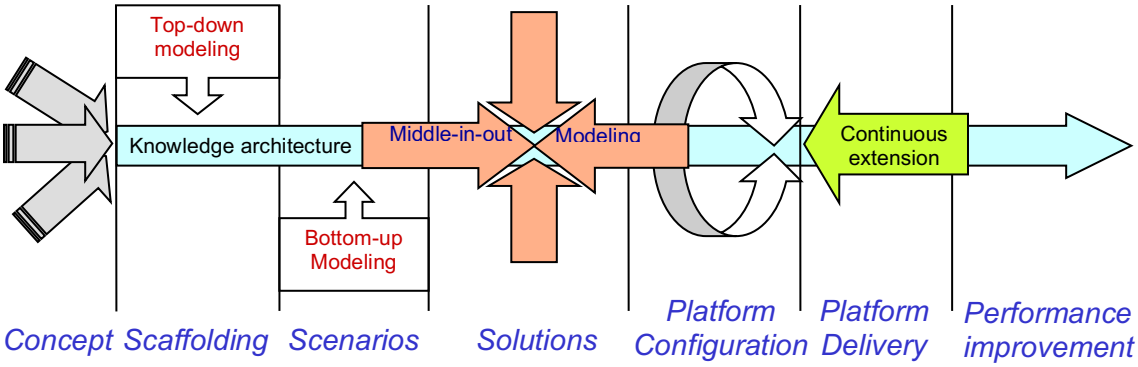
Active Knowledge Modelling – AKM Methodology	
Target roles – groups, teams and persons.	Consultants and internal IS people are solution builders. Super users support, customize and manage solutions. Modelling users who want to model and adapt their own solutions. Methodology developers who define methods for reusable solutions. Application integrators who set up data exchange and notification protocols with legacy tools.
Knowledge models of role workspaces and their knowledge aspects	Combining *generic meta-models* and sector specific meta-models Information, Roles, Tasks and Views – *IRTV* Products, Organizations, Processes and Systems - *POPS*
Cyclic project activities and task patterns	Concept development, Scaffolding, Scenario modeling, Solutions Platform configuration, Platform delivery and Performance improvement and Operations support
Work specific Views and Methods	Project Objectives, Configurable Visual Workspaces (CVW), Cyclic Design Patterns

Through *AKM models* the core platform is customized by up-to-date business knowledge represented in model views. Existing applications, services, and business processes are plugged into agile solutions in a model-driven way. The unique modeling and meta-modeling techniques of AKM ensure that business and project users, not IT support, control the solutions. This approach will help you:

- Cut IT costs dramatically, particularly for networked partnering.
- Increase stakeholder involvement, user interactions and interoperability.
- Sustain a manageable IT infrastructure adapted to each project.
- Support the core of your competitive advantage – your innovative design processes and methods.
- Let business and product needs directly control your IT infrastructure.
- Move your business logic from inaccessible and rigid software into visual, dynamic active knowledge models, workplaces and visual arenas supporting:
  - pragmatic user interaction and collaboration
  - continuous evolution of pragmatics
  - immediate interoperability
- Extend your business through collaboration spaces, and workspace views on the web.
- Provide valuable services through software factories and supply chains.

Main Activities, Approaches and Methods

The figure below identifies the 7 main activities/tasks of visual solutions development. These tasks are carried out partially in parallel, and often in a more cyclic manner. As illustrated, the knowledge models and knowledge architecture is continuously elaborated and improved as the project work progresses.



1. *Concept development* identifies existing approaches, solutions and methods, then analyzes and tests them on the problem at hand, in order to reuse best practices and gain ideas for innovation.
2. *Scaffolding* creates an overview model of the domain in question, identifying the roles of users and stakeholders, the tasks they perform, and the information they apply or produce.
3. *Scenario modeling* provides richer and more detailed representations of important task patterns. In addition to role responsibilities and detailed information flows, the model at this stage should also represent the views, with information content and services (tasks) needed for performing the work.
4. *Solutions modeling* specifies solution targets and functions, integrating scenarios in a purposefully designed knowledge architecture, guided by the core concepts created in the scaffolding model.
5. *Platform configuration* maps the information, role, task and view models to the execution platform, defining how information is stored, access control is enforced, tasks are executed, and views composed and managed.
6. *Platform delivery* involves training users, but as well as integrating existing application systems and databases, applying e.g. model-configured data exchange, parameterizing APIs and web services.
7. *Operation and improvement* deal with the continued customization, adaptation and extension of the solution to cover a wider scope, handle environmental changes, or reflect improved work practices.

For most projects, we advocate starting with an initial concept analysis and scaffolding modeling in parallel. As soon as the target user roles of the solution have been identified, pilot versions of workspaces for these roles should be created to support their communication.

Knowledge Models

Above we introduced the main activities of visual solutions development from a conventional process perspective. However, to really understand the many dependencies of the AKM methodology, we need to take a more product design-oriented approach supporting cyclic design. From this perspective, the knowledge architecture for AKM solutions development has four main dimensions:

1. User scenarios, and adaptive solution models
2. Targets and solutions development and configuration models
3. Product concepts, functions and design of configurable components
4. Platform integration and extension models, and work planning and implementation.

These parts are developed by different roles and teams:

1. End users model the content of their own work, assisted by modeling facilitators.
2. Change agents, consultants and super-users configure solutions.
3. Technical consultants integrate applications and extend the modeling and execution platform.

In a solutions development project, product modeling may start immediately, using the standard AKM templates. Later new solutions are phased in as the templates are enhanced. Though the solution at first will be generic and slightly cumbersome, an early start with product and work modeling is recommended.

- Expectation as to what kind of solution may be developed and provided, helping e.g. selection of the best scenarios and methods, and areas to improve by a certain solution.
- Ensures that concrete tasks and work content is at the center of attention throughout the analysis, as a safeguard against analysis-paralysis (emphasizing conceptual structures, correctness, and completeness rather over what is useful for solution development).
- Allows developers to quickly test existing concepts, methods, targets, and solutions, as well as suggestions and innovations that emerge during the development work processes.

The core aspect we are trying to capture in solution modeling is the context of the targets and users' work. The modeling facilitators must provide training, support, and consulting services during this kind of pilot solution modeling. A modeling expert should sit in with the users in joint modeling sessions.

Pilot Use-cases

The figure also shows that in parallel to the pilot solution modeling with the users, the solution configuration team works on the configuration models for the solution to be delivered (scaffolding, scenario, and solutions modeling). These models define and enhance the four dimensions of the knowledge spaces:

1. Information (I), the information needed to perform the work, which information is produced, etc.
2. Roles (R), who are involved in the activity, what is their responsibilities, which tasks do they perform, which information do they use, which views should their workspace consist of etc.
3. Tasks (T), which task are performed, which services are used to achieve the results.
4. Views (V), which views should be available in different workspaces, which information and services they give access to, what views contain and look like, which tasks should views support etc.

The IRTV meta-model is used for capturing, discussing, and managing the lessons learned from the user solution. It relates to the user solution model in a reflective way. The objects, relationships, properties, and other types created by the modelling teams are managed in evolving IRTV models.

The third part of the use-case architecture deals with *platform configuration, delivery, and improvement*. Technical consultants build this model to design and implement the software components and services needed to support new solutions. It should support all the information content, roles, tasks, and views.

Once the platform is ready, the IRTV model contains all the information needed for making the solution operational and deploying it with the project team. By adapting the IRTV structures of the configuration model, refined or variant platform solutions, e.g. for different product families, can be implemented with a minimum of effort. The three activities come together to produce evolving solution variants.

Concept Development

The project team explores, discusses, and analyzes existing visual solution concepts, in order to,

- Better understand the business challenges faced by the organization,
- Get new ideas about how visual solutions may be applied to solve identified business challenges,
- Improve solution quality and reduce costs by reusing existing solutions or services.

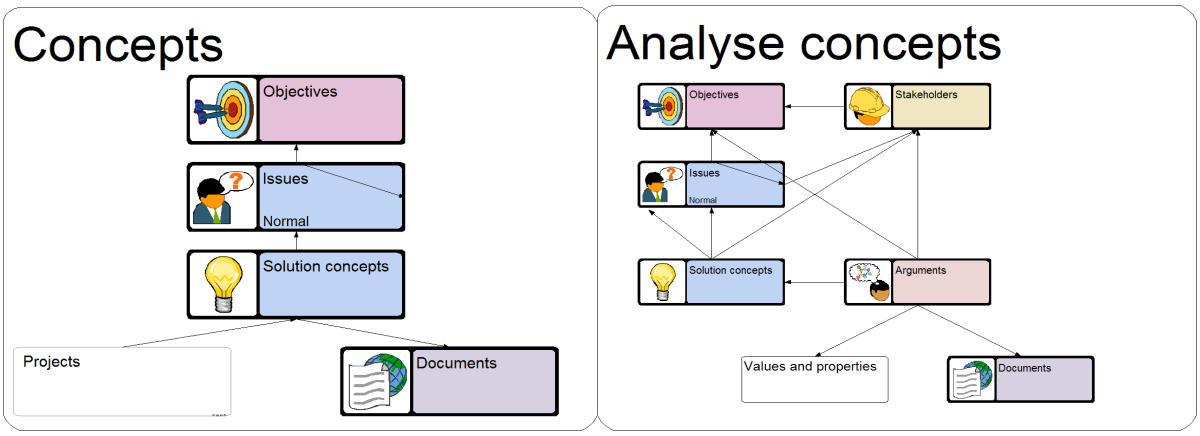
The solution concepts may come from any of the participants:

- Standard and basic solutions developed by AKM and their partners.
- Applications and methodology support solutions developed by independent solutions development consultants.
- Applications and solutions already being used by the customer company, e.g. based on spreadsheets and other documents, or simple database applications (e.g. Microsoft Access).

These steps are involved:

1. Capture the key business objectives and challenges that the solution should address.
2. Identify, describe and model the solution concepts deemed relevant. In addition to concepts based on existing solutions, completely new ideas should also be represented.
3. Analyze the concepts, capture arguments pro and contra their usefulness in this case, with reference to the business objectives and challenges.
4. Select one or a combination of concepts. If a template is available for the selected solution concept(s), include its tasks, views, roles, and information content in the model for your project.

The model below shows an example of a high level analysis of solution concepts. On the top, we find five business objectives, which leads to four challenges (issues) of concern to key roles in the organization. A set of solution concepts address these challenges, sometimes causing new challenges.



The steps involved in creating a concept development model such as the one above, manipulate different parts of the structure. First, we model key business objectives, and the issues and challenges that should be solved to meet these objectives, using these concepts.

Here we may also include documents that describe the solution concept, or previous projects where the solution was modeled, analyzed, or applied. Next, we start the analysis, arguing the strong and weak points of the alternative solutions. Finally, we select the concepts that we want to apply, and implement in the customer solutions.

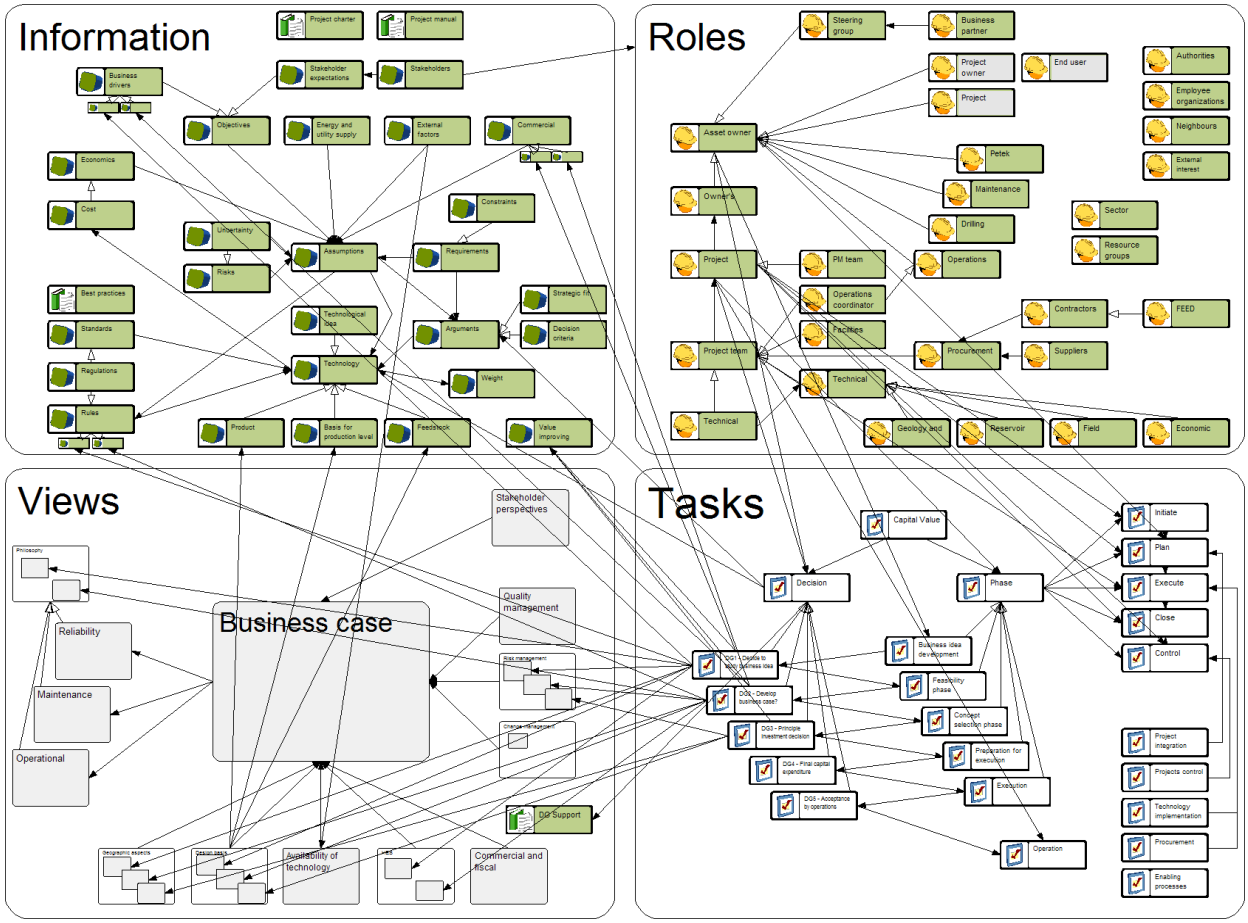
Scaffolding

The aim of scaffolding is to get an overview of the situation, the users and stakeholders of the companies involved, their main work activities, and the core content of their work. The scaffolding model is a top-level IRTV model that provides the conceptual framework for the construction of the main solutions. The scaffolding model should be initiated as soon as the project starts.

In this phase, we advocate starting with the modeling of roles, capturing who the main stakeholders are. If you have already started a *concept development*, you may already have defined a lot of roles. Once you have captured key roles, it makes sense to perform *task modeling*, to represent the responsibilities of each role. Task modeling often leads to the inclusion of more roles, when you for each task assess who is

involved and responsible. Instead, start *information modeling*, represent the content being worked on by each task. Finally, *view modeling* can be used to represent key viewpoints held by the roles, or collections of information (e.g. documents) that represent a recognized solution.

A scaffolding model identifying the main IRTV elements and their interdependencies, may look like this:



The view models may also be used to challenge the stakeholders on common assumptions such as the statement “basically everything is included in the business case”. This is visualized in the example above.

Role Modelling

To build the scaffolding model, the IRTV dimensions should be pursued in parallel. For communication with end users, however, we have found it useful to start with the Role models. Coming top down, the high level roles and positions in the organization is generally well known, and the roles constitute a suitable starting point for further analysis into which tasks each role is responsible for, and the most important information they provide or use. By simply identifying and listing the roles, we liberate these concepts from the organizational hierarchies that they are most often seen as part of. By focusing on the responsibilities of the roles in terms of tasks and information content, we direct the analysis towards work practices, rather than administrative reporting or human resource management structures. Often, though, we find it useful to capture simple classification among role concepts, and the main communication relationships between roles. The details of how these relationships are used later become part of the information, task, and view models. Implicit modeling techniques such as just grouping related roles or placing them in layers according to their decision making tasks, are often useful triggers of discussions. The benefit of this approach is that the users are first exposed to modeling in a domain they are familiar with. Although

simplistic, the analysis and discussions triggered by role modeling often leads to identifying critical issues, such as poor communication barriers and differences in focus and perspectives among the actors.

The example above shows a typical role model at the scaffolding stage. No explicit relationships have been captured yet, but the roles are placed near other roles that they work with. Horizontally, the roles are roughly placed according to where in the product and project lifecycle they are mostly involved, and vertically, we find officers and boards at the top, managers in between, and the people performing the work further down. This simple analysis led to the identification of an imbalance between the product organization to the right and the project organization to the left. The project organization has no counterpart to the "produktutviklingssjef" (product development manager) role. There is a need for the missing role, which for instance could be called "market manager", in that nobody currently is in charge of a systematized collection and prioritization of market knowledge such as requirements and trends.

In another case we had more difficulties agreeing on a common terminology for the roles. Therefore, some degree of classification has been included, identifying generic roles such as "End user" and "Project team member". In both these examples, we have included roles assigned to individuals, to teams, and for organizational units (such as "asset owner"). In this example from the O&G industry, the communication relationships helped us identify two major groups of roles, one supporting the asset owner in the early analysis, and another supporting the project manager in the later engineering phases. In some cases, the limited communication between these groups was seen as a problem. Difference in perspectives, for instance that the project group did not emphasize cost factors in their design to the same extent as the asset owner group, were also seen as problematic.

Task Modeling

At the scaffolding stage, the task models should identify the main activities performed in the domain of study, for instance the phases of product lifecycles. Though some break-down of tasks into subtasks may be warranted, this is not the place to perform a full hierarchical process design. Typically, you should capture the two or three main tasks of the key actor roles identified above. The responsibility for targets and participation relationships between roles and tasks should also be captured.

A high level example is modelled in the scaffolding stage. In this case, the scope of study was defined as the "Capital value process". According to the quality control project handbook of the customer company, this process consists of five phases. Each phase ends in a decision gate, which determines whether the process should be continued into the next phase, as well as the major direction to follow. Each phase follows the same pattern, with five major sub-phases (initiation, planning etc.). In addition, the concrete tasks were classified by type (project integration, control, technology implementation etc.). For each type of task, a different set of roles are involved.

This analysis shows some of the limitations of conventional business process modeling, and it is not a good example of how multi-dimensional task analysis should be performed. The process hierarchy emphasizes administrative issues such as decision gates, common global structures, and a strict sequence of activities rather than concurrent and collaborative engineering. The core parts of the work, the creation of designs that solve problems, the analysis of costs, safety, quality, risks and other factors that aid decision making, are buried several levels down in the hierarchy.

The multi-dimensional IRTV approach dictates that you should look at tasks in connection with the information content being used and produced. This guides the analysis towards the concrete tasks where the core design work is performed, where the key information is produced. By using roles as analysis lenses, we ask which tasks are the most important one for each participant.

Another way of approaching the task dimension is to start with the critical design issues, the key questions that need to be settled in the process, e.g. "What kind of subsea equipment is needed for this oil field?", "What price should we propose in this contract bid?" An example of a set of such design issues, and their main interdependencies, is presented below. In addition to the structured well-tree of sub-issues, we find four generic concerns that must be addressed (capacity, competence, PR, legal issues). Starting from such elements, we need to ask "which information is needed for settling these issues (populating the information model), and which roles are in charge of settling the issues (correcting the role model).

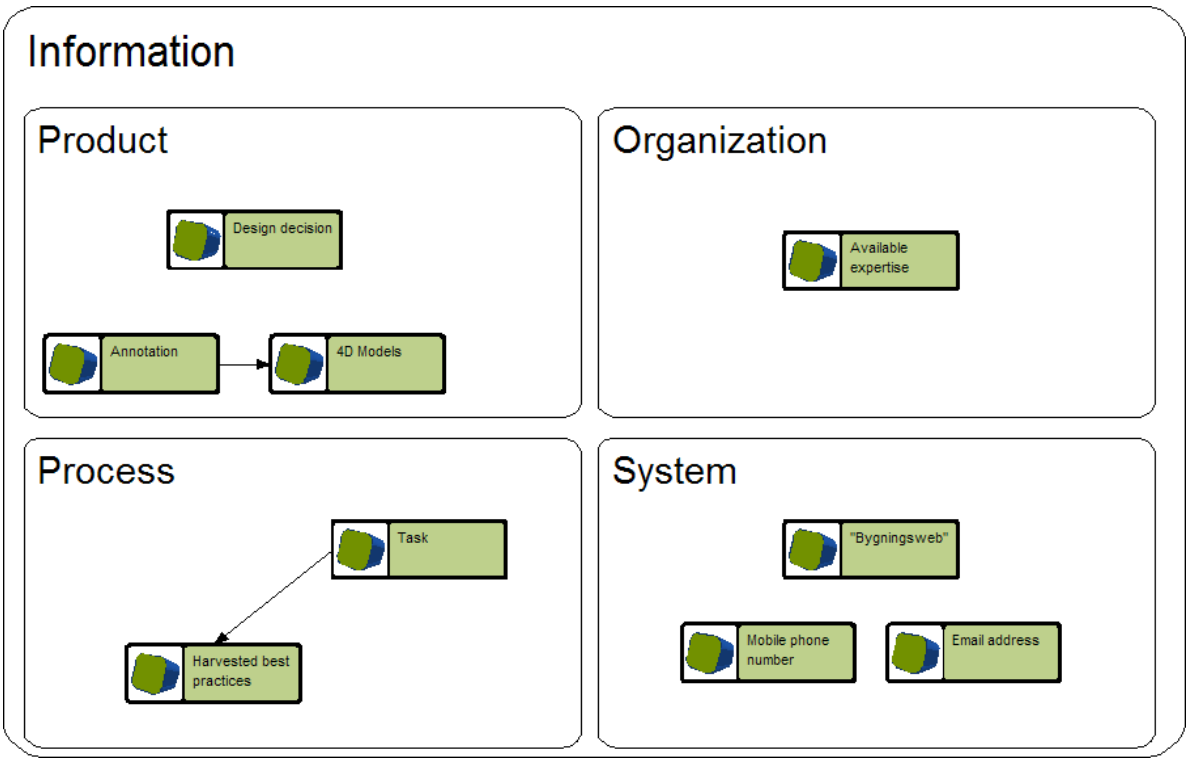
Information Modeling

The aim of information scaffolding is to identify the main information content of the domains, not to design language metamodels, glossaries, or structured ontologies. Therefore, we typically use a simple language for representing the information content, capturing the main elements, and their most important relationships and properties.

We suggest focusing on the product information, and related structures such as customer requirements and engineering knowledge, because this most often constitutes the core knowledge of the company. Operational processes and organizational aspects should already be covered by task and role models to the presently needed degree. In general, however, depending on the domain, all the dimensions of the other knowledge spaces may constitute relevant objects for information modeling.

The example below shows a simple information model, identifying some core concepts and their interdependencies. Not only does this model cover different entities of information, different aspects and forms of information are also represented, such as “assumptions” and “arguments”. In other projects, we have focused on uncovering the core parameters and properties used for describing products. The model below identifies five different product families (lower left corner), different kinds of product parameters (constraint, design dimensions, performance, and variant-defining), as well as some critical elements such as economic issues, which fall outside of these technical structures.

This can be further extended, using the POPS dimensions of the design space as lenses for information analysis, as illustrated below:



View Modeling

When we get closer to the level of detail needed for configuring an operational solution, view models will take a center stage in the development effort. Views capture what information and services a user filling a role needs in order to perform some tasks. Compared to the other dimensions, views capture the to-be-solution to a greater extent than the as-is problem situation

In the scaffolding phase, views offer the developers an opening to be a bit more inventive. Scaffolding view models can be applied for testing ideas with respect to what kind of solution the user will find valuable. Views can also be used for capturing structures that cut across the dominant information areas, such as cost, risk, safety, etc. In the example below, several such cross-cutting aspects were grouped around the central collection of information, the business case, reflecting the users' statements that "basically, it all ends up in economics". Such view models can be used for challenging users' perspectives, and for ensuring that critical aspects have not been overlooked, e.g. in the information and task models.

Another example is shown below. Here the views capture different levels of detail and abstraction in the company's product data. This model was used for discussing how one term, such as 'product' may mean similar, but slightly different things on different layers (and to different roles).

In this case from the construction industry, the level of house ("hus") was seen as the central one for the conceptualization of a product strategy. During the analysis, it was identified that the "standard staircase" ignored certain critical layers of knowledge, related to the construction project ("prosjekt") , the lot where the houses are to be built ("tomt"), and the units or apartments that te buildings will consist of ("boenhet"). This reflected the bias of the product development people who worked with the standards staircase towards technical aspects (the elements of buildings) rather than customer requirements (associated with apartment), market trends, and project management issues.

Scenario Modeling

Scenario modeling creates detailed descriptions of how a visual, model-configured solution must work to solve a given problem.

Whereas the scaffolding models are built top down to get an overview of the situation, scenario models focus on the concrete details of particular situations. We typically abstract and generalize from these particulars in order to build a useful solution based on scenarios. Scenario modeling is a *bottom up* analysis methodology.

In this phase, you ask more detailed questions about the IRTV dimensions of the problem. We may need a richer and more precise modeling language for each of the dimensions and their interdependencies. Starting from a scaffolding model, the first challenge is to select the right areas and scenarios to study in depth dependencies. Suitable scenarios typically have some of these characteristics:

- Great importance to the company, among its core competencies and processes
- Where performance problems have been recognized
- Simple enough to enable early demonstration of value, but complex enough to warrant a solution that utilizes structured visual modelling.
- In an environment that changes, where each project has some degree of uniqueness.
- Knowledge intensive and highly dependent on personal experience
- Areas where the information has not yet been structured, and documents are communicated.
- Areas where investments in CAD, ERP, PLM/PDM, etc. have not given the expected benefits.

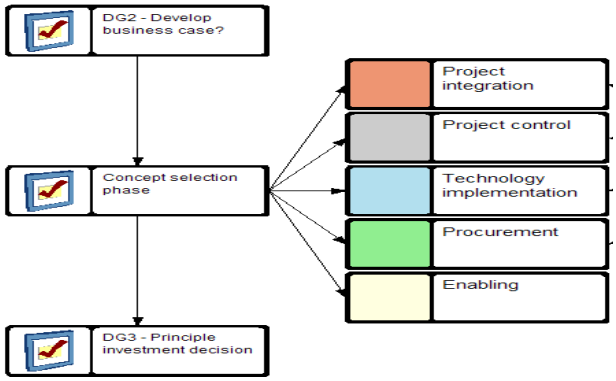
The most important criteria, however, is the availability and dedication of people who are knowledge in the field to participate in the scenario modeling process. The availability of a concrete, representative case is also paramount. An ongoing project is an ideal case as long as project time pressure does not come in the way of active participation in the modeling process. A previous project is still far preferable to not having a concrete example at all.

The scaffolding model offers some hints about what kind of scenarios me might focus on:

- The activity and work of a particular role, team, or group of roles
- A certain task, specified task-patterns or related group of tasks (sharing context)

- The lifecycle of some information entity, such as a product component design, or a view containing related information, e.g. a business case.

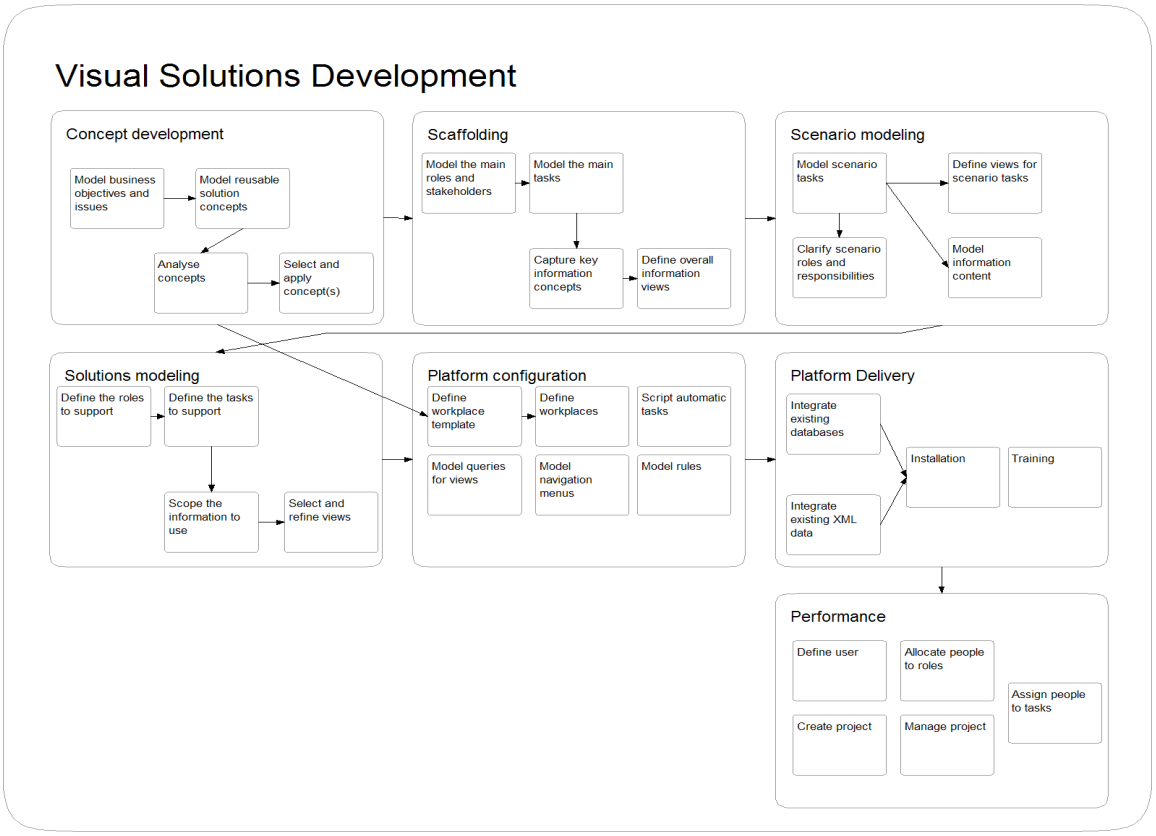
It often makes sense to start the scenario modeling of a new model view, bringing in just the elements that are part of the scenario. In one project, we decided to focus on these project activities:



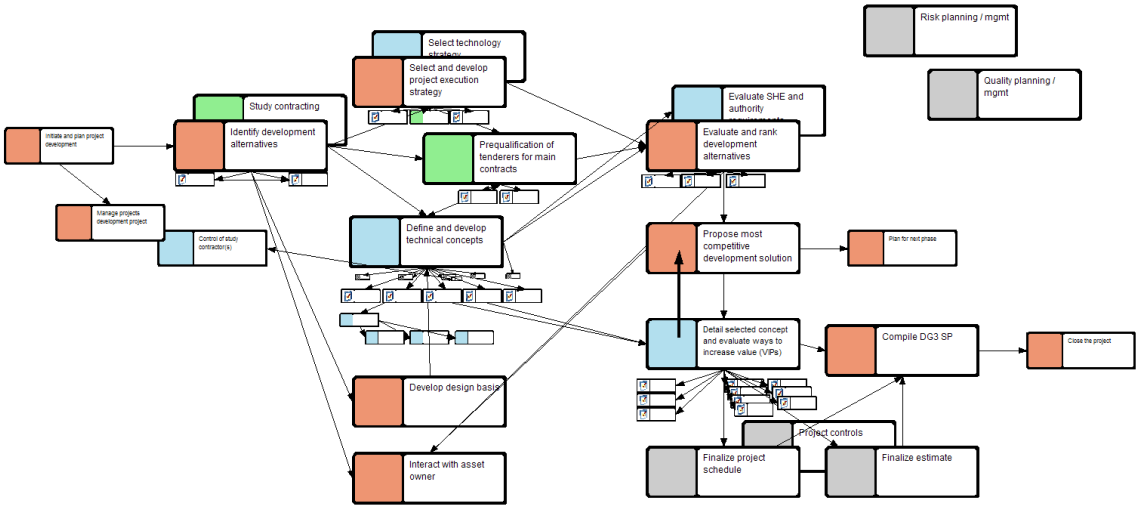
The scope is also normally restricted to a set of main tasks that participating roles are involved in.

Visual Solutions Development

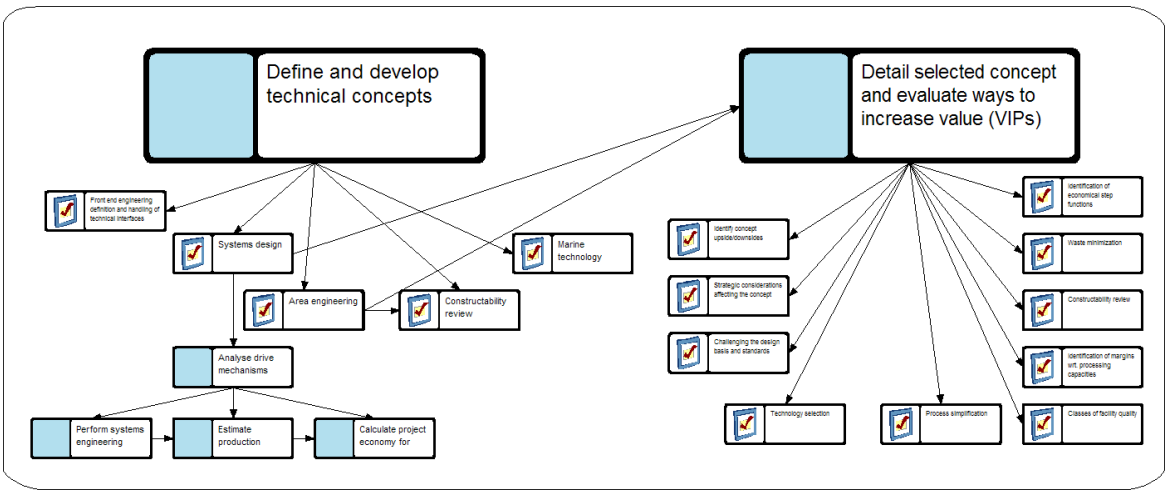
The next step is to refine the models for these tasks, representing subtasks and their interdependencies. In this case, a conventional process modeling notation, such as BPM or IDEF, may be applied, as in the modeling of the Visual Solutions Development process:



Note however that with AKM's task management approach, there is no need to completely structure the process, connecting every task into a sequence. If there is no absolute dependency between the tasks, it is often better to give the users some freedom to choose in which order they want to perform the tasks, and to which degree they want to work on multiple tasks in parallel. Sometimes a tree structure such as a work breakdown hierarchy is therefore just as good a depiction of the processes. Below, we have combined a tree structure with some sequencing.



Sometimes, the scope of the scenario, or at least the scope of what should be modeled in the first round, need to be further refined. In the case below, two subtasks were selected for detailed refinement.

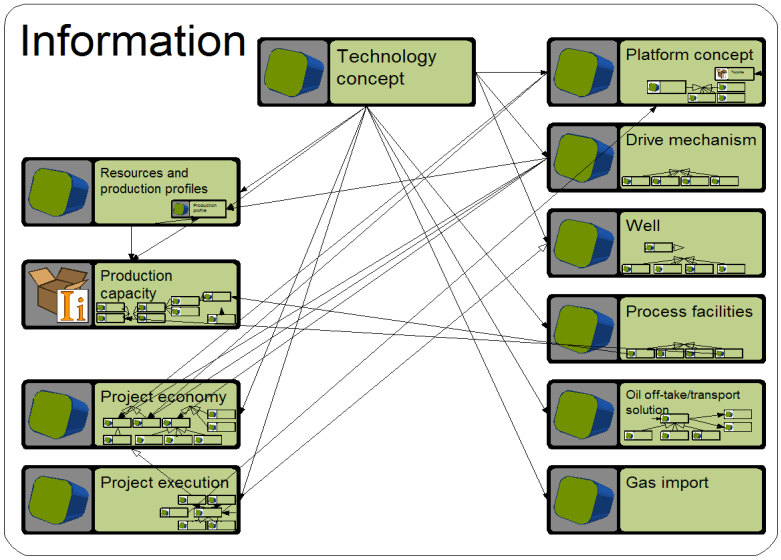


Information modeling

Above, we saw that *task modeling* play an important role in defining scenarios. Existing methodologies such as UML, with their use case diagrams that capture roles and tasks, also follow this approach. Like in UML, we advocate the use of textual scenario descriptions to complement the visual models at this stage.

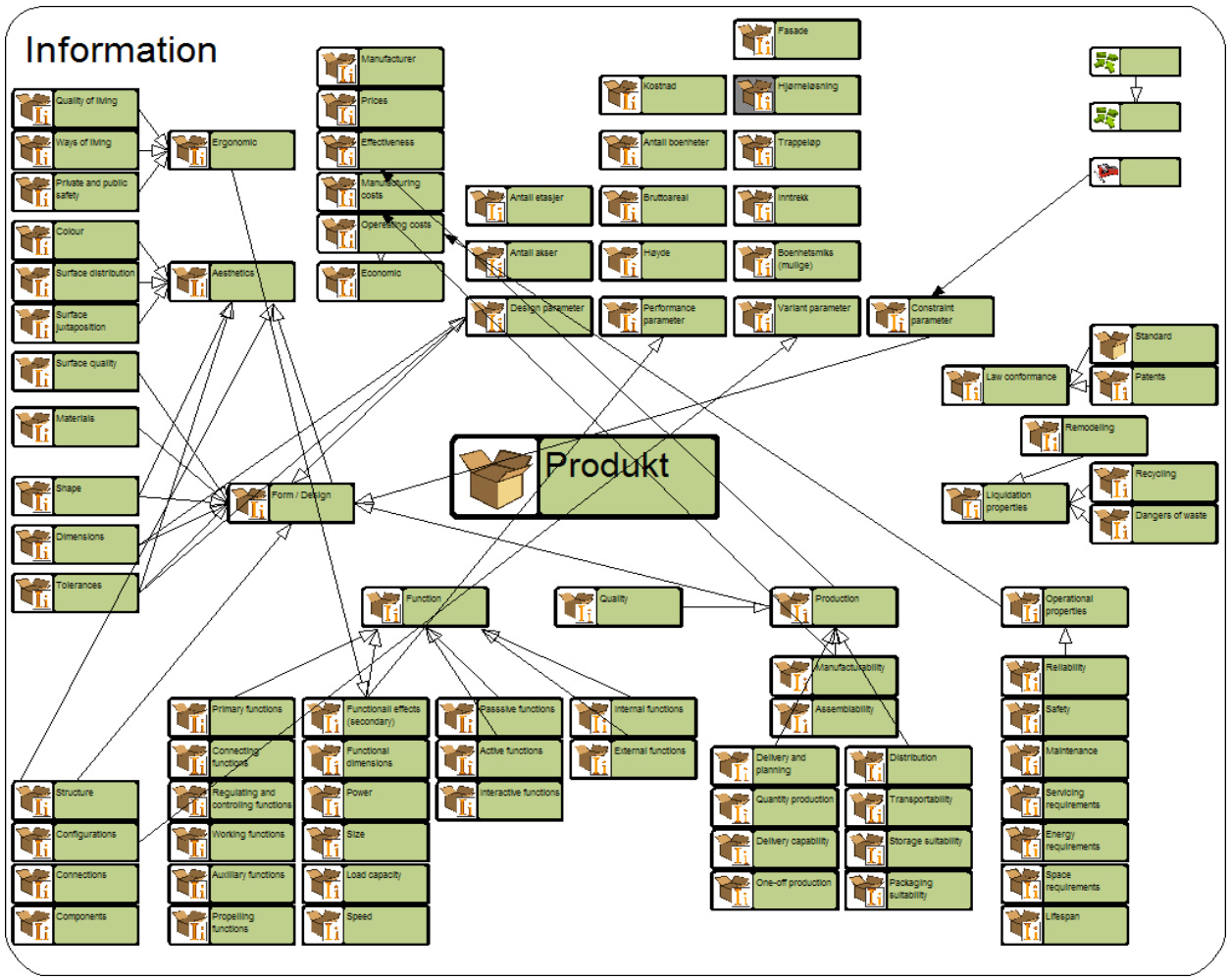
We must, however, not forget the other dimensions of the knowledge space. Alongside the task models an *information model* should be developed that manage the content of the information needed and created by the tasks. Typically, existing documents are a useful source of input to the information model. By representing the information and task dimensions in parallel, we also ensure that no task which produces or uses critical information is forgotten.

An example information model is depicted below. It shows the main elements, and some of the more detailed content associated with each element. There are also relationships between the tasks above and the information elements, indicating production and usage of information.



While the example to the left emphasizes the information object classes, it is sometimes more useful to focus on the properties and values that will be worked on, as illustrated below:

A combination would highlight both the key concepts, and the properties that the users assign values to during their tasks. The example below also includes key documents that describe some of the concepts. The snapshot shows object classes in the left column, and their properties to te right.



Defining Views for Scenario Tasks

To complete the scenario models, we take the joint information/task modeling one step further, by introducing the view concept. While information modeling aims to capture the overall information structures of the problem area, view modeling tries to identify exactly what information the roles need for performing a given task. In scenario modeling, information and views should be worked out in parallel.

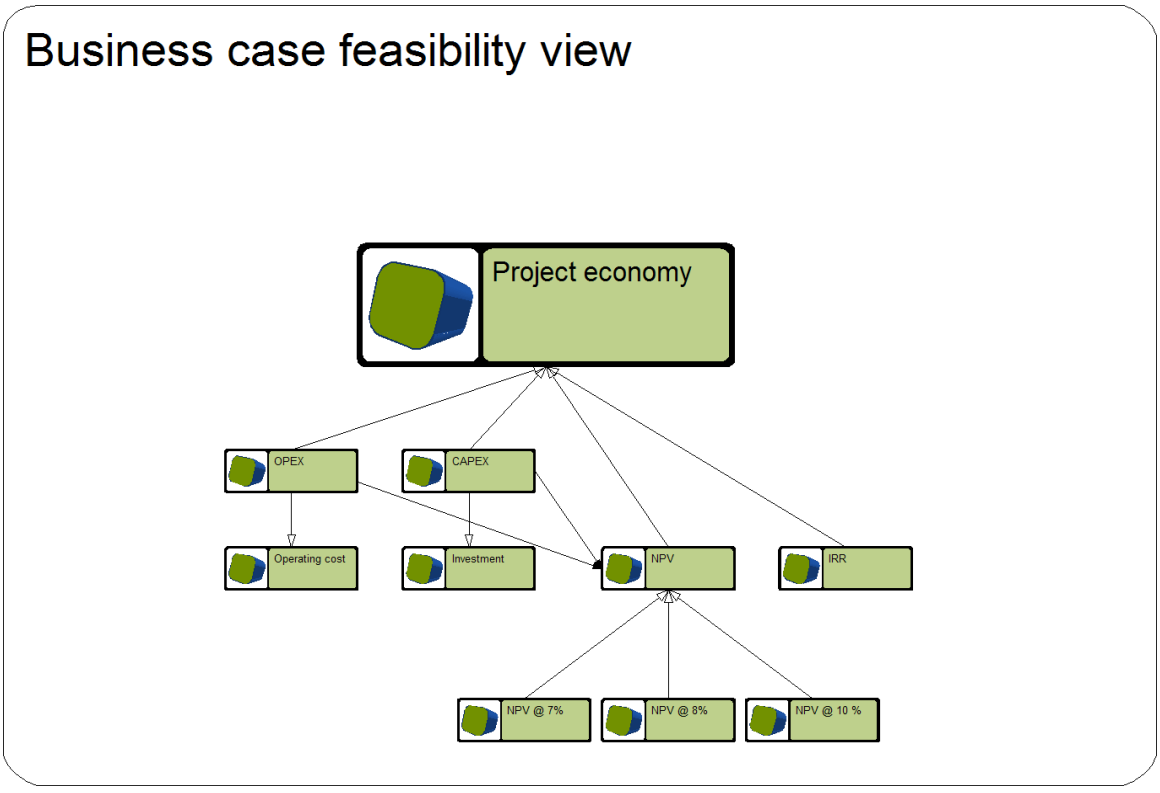
Scenario views define what information should be available in the workplace of a given role when (s)he is performing a certain task. Depending on the scenario case and desired solution, we may decide to make view models for detailed tasks, or for more high level situations, such as the involvement of a certain role in a given phase or process. These issues and questions may help in capturing relevant views for practical work:

- As-is practical work

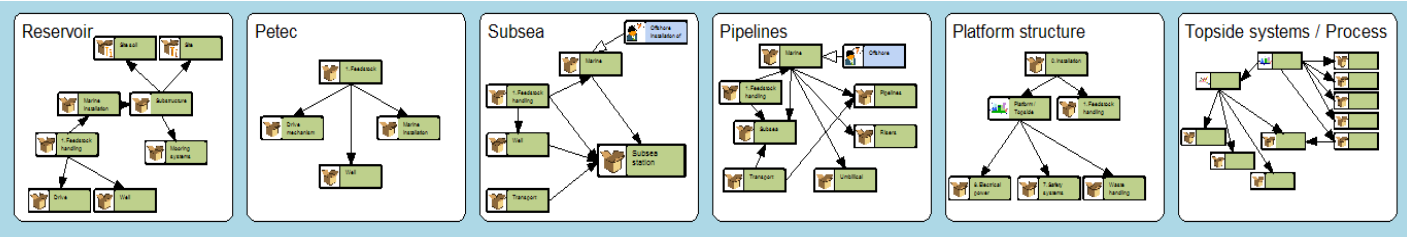
1. Which information do they process?
2. Which information do they send to others?
3. Which information do they receive from others?
4. Which information do they own?
5. How much do they see of the work of others and the discussion in engineering disciplines?
6. In which areas do they propose new solutions?

- To-be work practices
  - Identify needs and opportunities for improved knowledge sharing and interdisciplinary collaboration

The example below shows one role view, defining which elements are worked on in the task “Assess business case”, performed by the “Project controls” role.



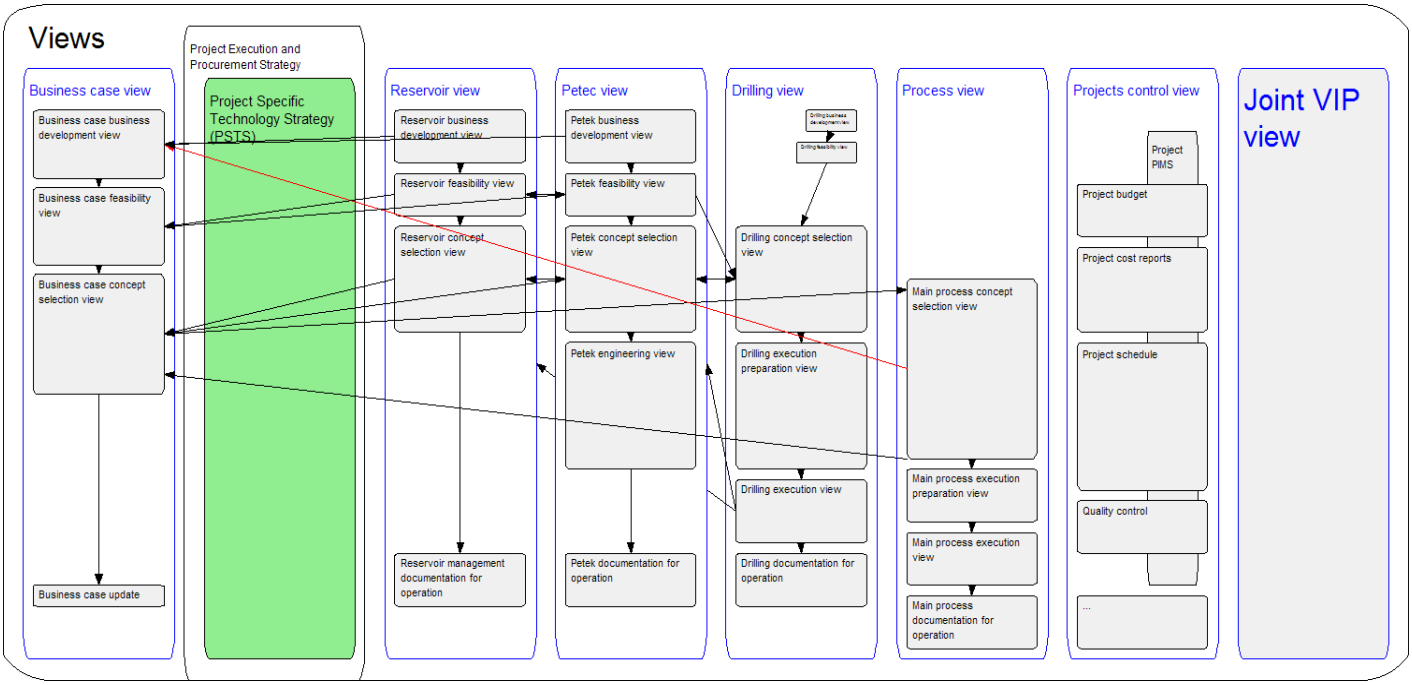
The more solution areas and scenarios a project model captures the more information elements are the different roles are responsible for during the “Basic engineering” phase:



In a complete and detailed scenario, we should combine information, role, task, and view modeling. Such a model can be organized in different ways, depending on what we want to achieve. The first example defines role views as vertical swim-lanes, and follows the succession of tasks in the process downwards.

The vertical axis then reflects the passing of time, indicating the relative start time and duration of each task, with concurrent tasks placed at the same horizontal line. When we open up each view, the information content is shown, like in the examples above. In addition to communication dependencies captures by the inclusion of the same information elements in multiple views, the dominant communication links have been represented as relationships between the views for each task within the role swim-lanes.

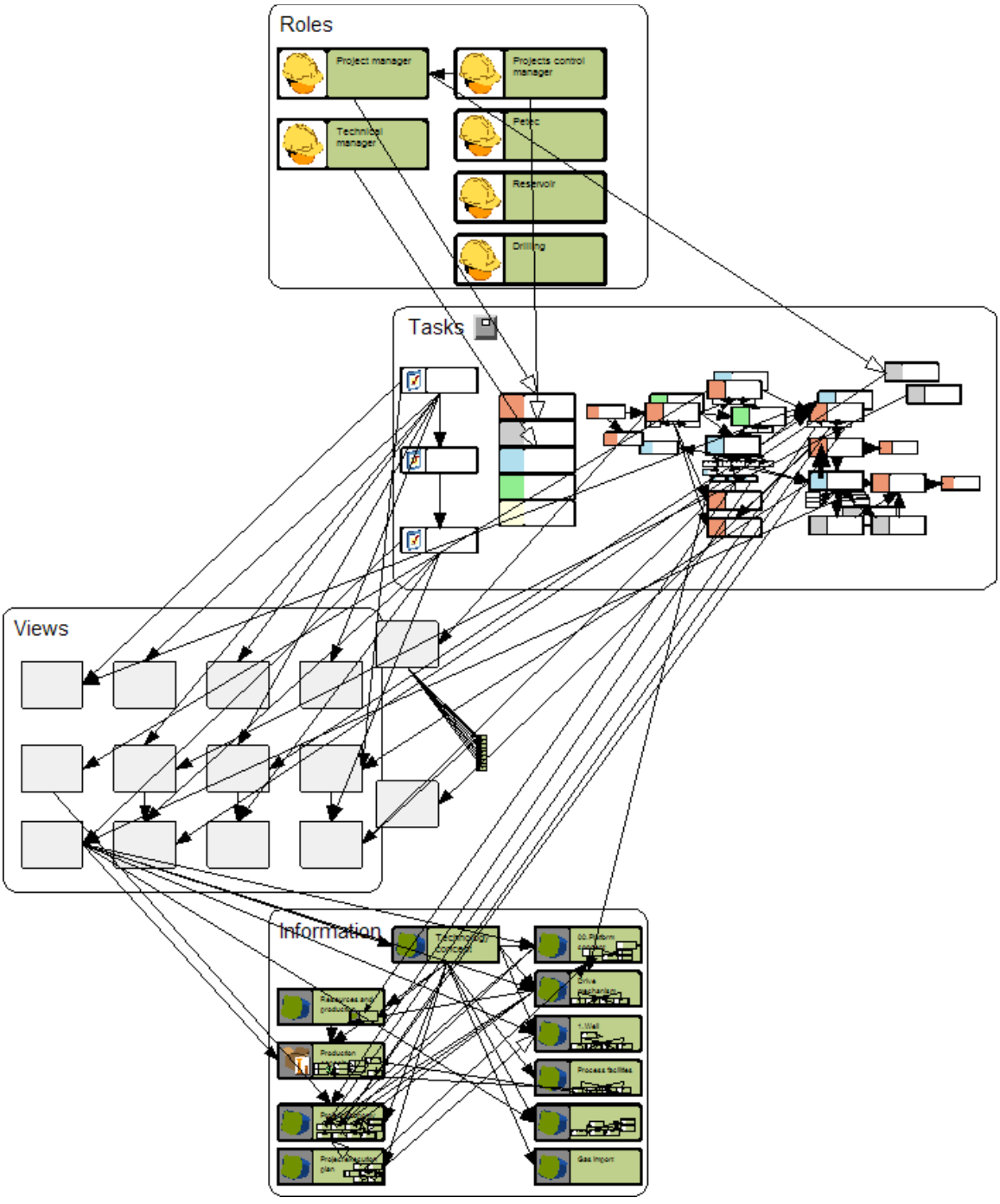
We commonly use a red color to highlight items of concern. In this case, the feedback from the later phase “process view” to the early phase “business development view” indicates a potential source of rework and delay.



During the first attempt at modeling this scenario, we did not have enough information about some of these dependencies. The starting point thus looked like this:

During a modeling session with involved users, more knowledge was represented, resulting in the model presented earlier. As shown, two cross-cutting views, aimed at improving inter-disciplinary communication and knowledge sharing, was also added, for the collaborative “project specific technology strategy” and “value improving practices (VIP)” tasks. The order of some roles was also altered, to fit better with the degree of collaboration between the roles.

Scenario modeling thus focus on roles and tasks at the start, diving into detailed information and view models, resulting in a scenario model which may look something like this:

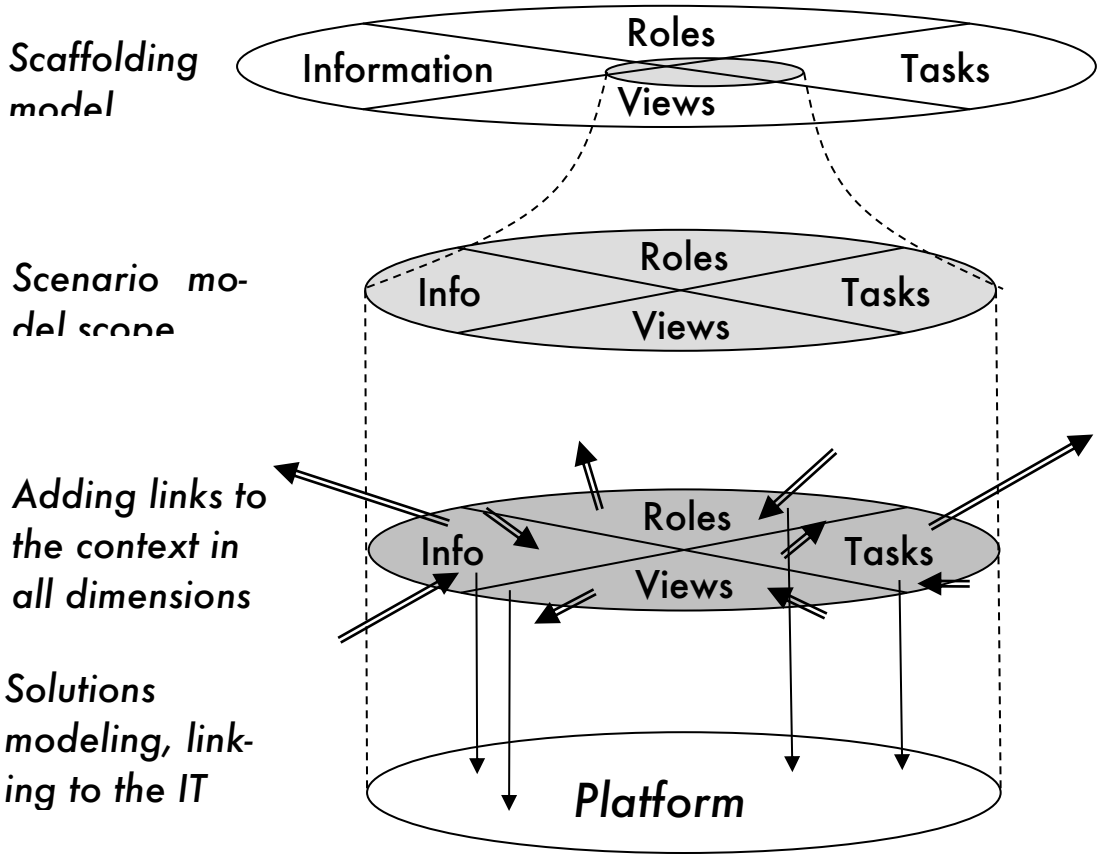


Scoping and Refinement

During scenario modeling, we sometimes see that the scope of the scenario is widened, by including elements from the context around the original scenario, e.g.

- Roles that are not formally involved, but nevertheless are important stakeholders, decision makers, or consumers of the information produced in the scenario,
- Information produced somewhere else, but applied in the scenario,
- Related tasks that e.g. apply the same information or resources.

This evolution patterns is illustrated below:



Solution Modeling

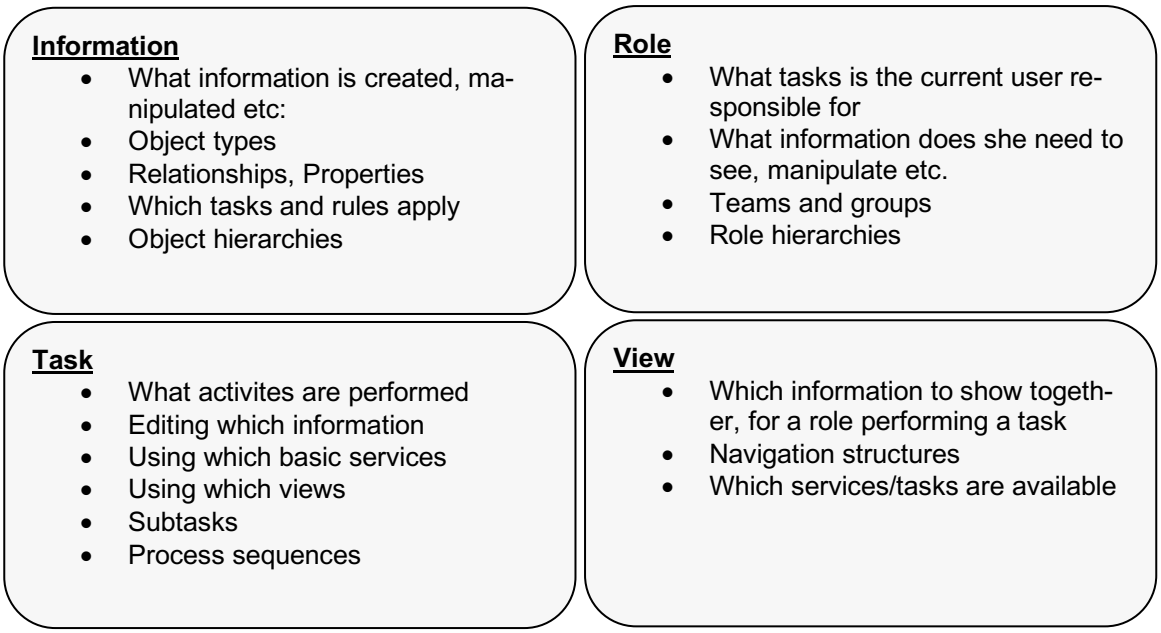
Solution modeling brings together the top down overview created in the scaffolding phase, and the bottom-up scenario models, performing a sort of *middle-in-out* analysis. In this process, particular scenarios are generalized and aligned to form a coherent and integrated solution. As shown in the figure above, the final important task in this phase, is the linking of IRTV model elements to executable platform services for information, role, task and view management.

The execution platform that directly executes IRTV models, is the Active Knowledge Modeling (AKM) platform. The analysis leading up to the final solution configuration, covering the concept, scaffolding, scenario and early solutions modeling activities, can however be applied in more conventional software engineering projects as well. The following phases of the methodology, however, assumes that you will configure a solution in the AKM platform, utilizing the Configurable Visual Workplaces (CVW) interface within Metis modeling clients, supported by the Metis Enterprise repository for collaboration and data sharing.

Solutions Modeling

Using the IRTV Knowledge Space

The overview below summarizes the four dimensions that we are modeling for supporting :



The AKM platform comes with an initial model that specifies the core, most generic elements within each dimension, and their relationships to elements in other dimensions. The standard visual workplace likewise contains generic views and services that activate these generic structures, e.g. generic views for editing the information about a product element. These workplace components can be extended and customized by providing more detailed specifications in any of the four dimensions:

- If the customer defines more specific types of *information* objects, relationships, properties etc., the modeling and visualization services automatically adapt to include the new features.
- No *view* model extension is needed to accommodate new information types, but if the customer wants to override the default specifications and customize how the new elements are to be displayed and manipulated, view models must be updated as well.
- If the customer adds new and more specific *roles*, new workplaces for the roles are also made available. The services and configurations defined on the generic level apply to the new roles according to the role specialization hierarchy.
- Defining new and specialized *tasks* is probably the most common form of detailed customization and instance level adaptation, e.g. to meet the specific requirements of a particular project. Some automatic tasks can be defined as a sequence or more complex process of subtasks, while others are defined as rules. The triggering and coordination of tasks can be associated with modeled sequences, or rules that react to the occurrence of different kinds of events, such as “Every second Thursday at 11:00”, or “Every time a new product variant is created”. The AKM platform as well defines a set of generic tasks that have scripted code to be executed when the task is invoked. Other tasks are implemented with a view, a window in the workplace with some information content and other services (tasks) available for manipulating the information.

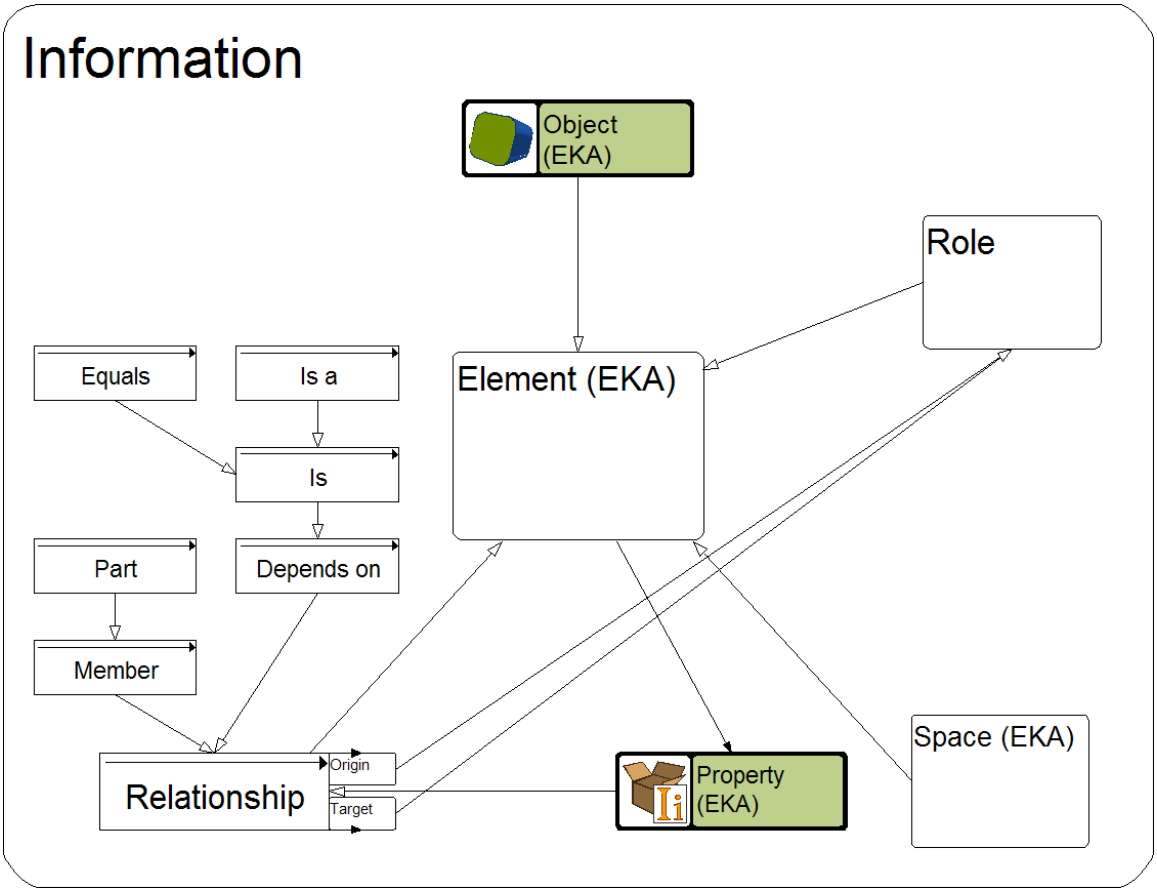
During this phase, the models should be specified as concretely and precisely as possible, to get the most out of the automated support. We should however keep in mind that many of the most important tasks will be performed by humans, and perhaps just represented as a vague item in the project plans. Nevertheless, to support precise modeling of IRTV structures, we below introduce the core languages used for each dimension.

Information Modeling

The information management component of the AKM platform defines how information models are stored and manipulated. Because all elements, not just POPS and surrounding spaces, but also the other di-

mensions of the workspace (roles, tasks, views) can be represented as information elements, the language for information modeling should contain all that is needed for representing any kind of active knowledge model.

This information model is called the Enterprise Knowledge Architecture (EKA). It is depicted below.



All constructs are regarded as ***Elements***. ***Spaces*** contain elements, but one element may be found in multiple spaces. Conventionally, most model elements will be ***Objects***. All kinds of elements may have ***Properties***, and ***Relationships*** link two elements through ***Origin*** and ***Target Roles***[1]. Relationships, roles and properties are also elements, so they may possess properties and have relationships to other element.

The EKA does not separate between meta-classes, classes and instances because

- One person's roof is another one's floor, thus an instance in one view may be a class in another.
- AKM models represent mutually reflective views. The definition of an element is not found in a single class, but through a set of reflective views.

Instead, a special relationship called ***Is*** between two objects (or relationships or properties), denote that the origin is defined by the target, and can thus express both specialization (Student is Person) and instantiation (George is-a Person). The instantiation relationship ***Is-a*** has similar meaning as Is, but it is used to separate meta-levels (for the modeling contexts where this is required). Finally, ***Equals*** is a bi-directional Is-relationship, which implies that the two elements represent the same concept, phenomenon,

[1] The concept 'Role' here refers to a participant in a relationship, in general. Throughout this methodology description, the term 'Role' usually refers to 'Actor roles', roles of individual persons, groups, teams, organizational units etc.

or entity. Equals is typically used for representing mappings between models that represent different perspectives on the same domain.

Other relationship types include general links and associations, and decomposition with (***Part***) and without (***Member***) ownership. Note that this approach enables classification, decomposition and states of properties, relationships and views just like objects.

Aspect Modeling

The above classification of core modeling concepts is no taxonomy. The same element may be represented in different ways in different views, for instance as an object in one view, a relationship in another and a property in the third. How the element is presented typically depends on the level of detail needed by the particular role. Many language constructs will also be defined as member or multiple core concept classes. As an example, take the concept “state transition”, which can be a task, a relationship and possibly an event or a rule at the same time. The core concepts can thus be seen as aspects or facets that any element may include as part of their definition.

Multiple reflective views are captured as multiple “Is” or “Is-a” relationships from an element. This approach can also be applied to mix in new *aspects* locally. For instance, if a group wants to add a cost dimension to a process model, they simply add an “Is” relationship from “Object” to “Cost unit” in their model. All objects within the model will then inherit the properties and behavior of cost units.

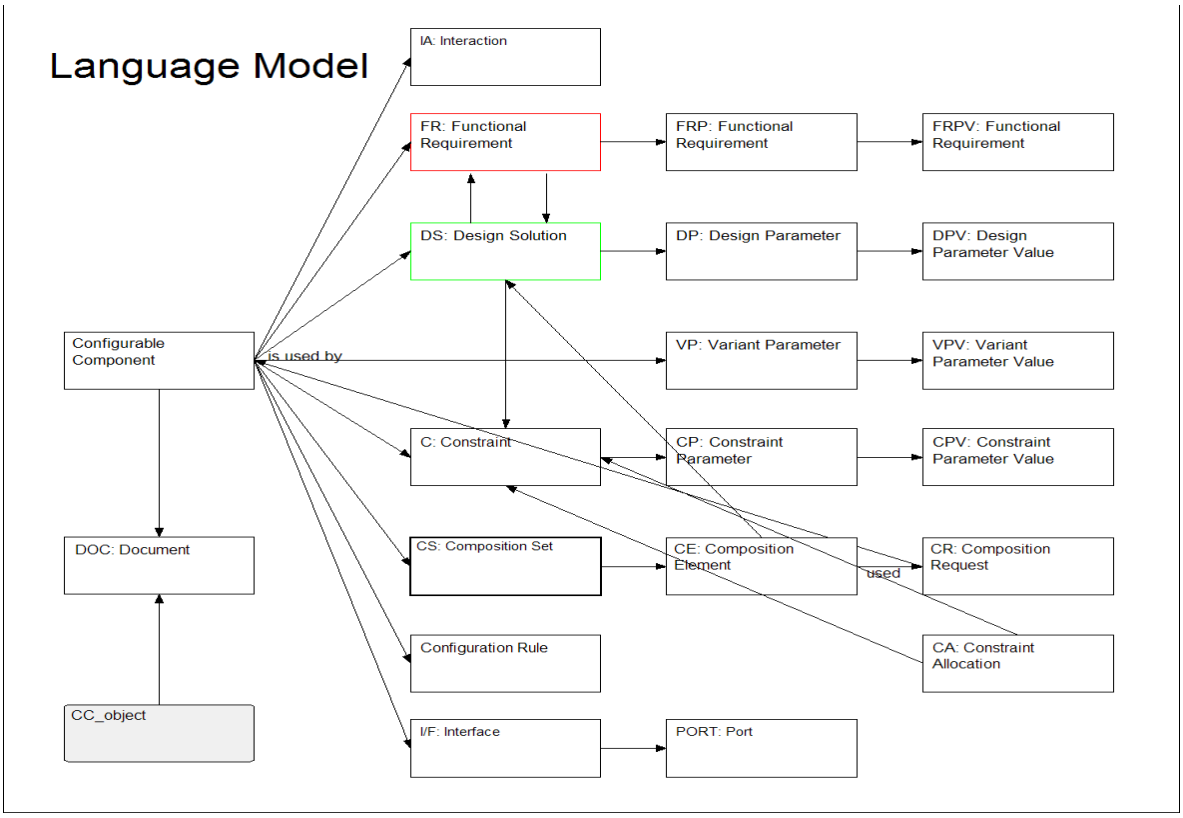
Reflection and Metamodeling

The EKA is inherently *reflective*; it makes no separation between meta-levels. Objects, relationships and property elements can be applied at any level. Inherent reflection also makes the EKA *coherent*, in that users apply the same modeling constructs and operations on any meta-level. They may perform “metamodeling” operations such as adding a property in the same way on instances and classes, or for that matter relationship and property instances and classes. This facilitates *instance level exceptions and evolution*. Similarly, users may perform modeling operations on objects representing classes, e.g. adding default parts and property values.

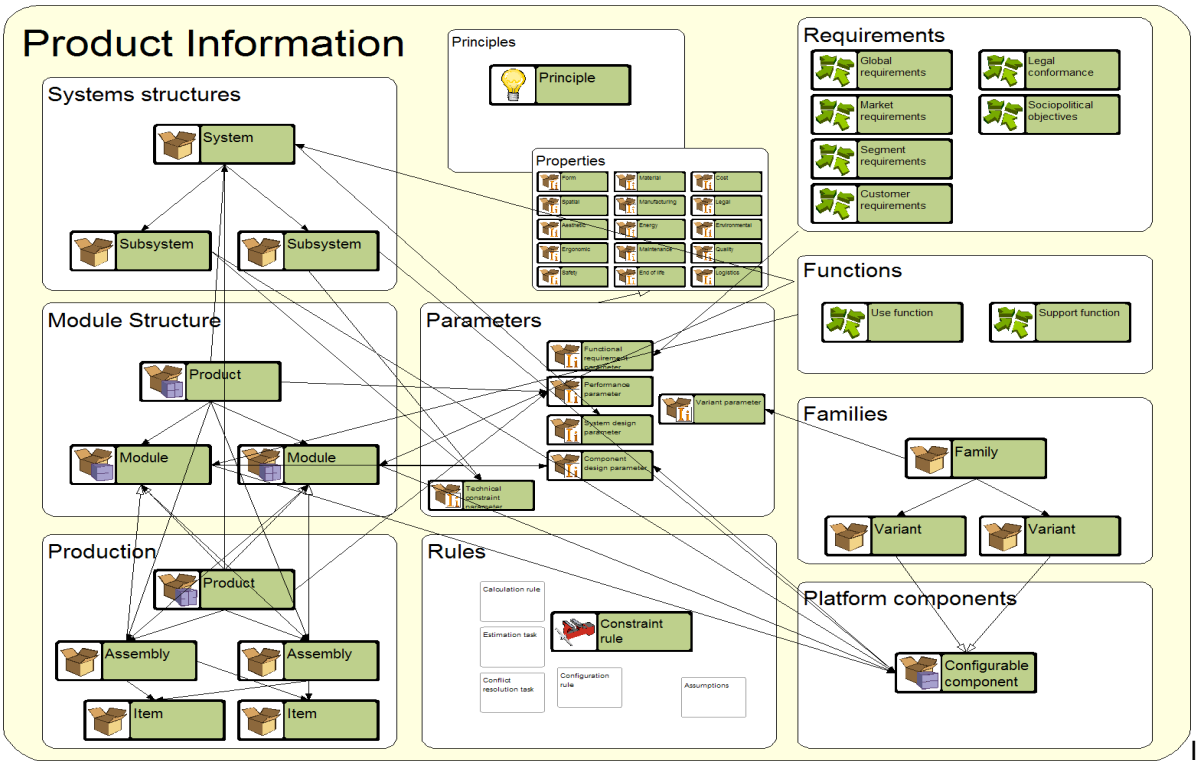
The Content of the Knowledge Architecture

During solutions modeling, the various information elements captured during top-down scaffolding and bottom-up scenario modeling, and as well any elements that you reuse from an applied, should be brought together into an overall architecture. This architecture should capture the content of your solution in a structured way.

For configurable components, the knowledge architecture contains elements of these types:



A more comprehensive design methodology being developed by AKM, provides an even richer picture:



In other cases, the solution reuses the standard IRTV concepts, and adds some additional types, like this solution for capturing design rationale, decisions and processes in the offshore oil&gas industry:

LIST of Documents describing AKM Model-driven Solutions

These are the documents that will be prepared and made available to all persons registered with the KAVCA company AKM members.

These are the documents that will be produced:

1. How are Sector Generic models and solutions built.
2. How are Customer specific models and solutions supported and built by meta-modelling.
3. How are Project Solutions for Cyclic Design of Oil and Gas sector Solutions built.
4. How are Project Solutions for Cyclic Design of other Industry Sectors built.
5. How are Cyclic Design and Delivery of Treatment to Dementia Patients built.
6. How can Project Solutions for Innovation and Learning at Universities be built.
7. How can Project Solutions for Cyclic Design of the Public Service Solutions be built.
8. How are Project Solutions for Cyclic Design of the Industrial Service Solutions built.