On: Experience with Modelling.

By Anton H. Landmark

Contents

Introduction

On modelling

I believe it is of some importance to keep a broad view on modelling in general, in order to have a good understanding of the various aspects of modelling:

Thus after a general introduction various examples is presented below.

- We will take a look at modelling in general in historic and in present day perspectives.
- Then we will look at some aspects of the more modern modelling process.
- Towards the end we will concentrate on modelling for computer based systems, representing ***"Model driven software development".***
- At the end are some examples of models from a CAD-CAM system: Autokon (AUTOmatisk KONstruksjon)

The notion and the art of modelling is being used extensively today, consciously or not. Our world is very complex, too complex perhaps to fully understand and comprehend all aspects of a problem or a task at a glance. Consequently, we automatically make the necessary simplifications, concentrating on what we at the moment feel are the important issues for the task at hand.

Normally this works rather well in our daily life: we believe we have a clear opinion on which goal we are going for, what the purpose is for each step on the way, and we think we have a clear picture of the necessary requirements, both for each single step, for the whole process, and for the final result.

The reason why this works rather well in our daily life tends to be that we normally are working with (what we think / believe) are well understood matters in a rather familiar context, (and most often this is true). We intuitively recognize obstacles and problems, normally rather unconsciously, because we are moving in a familiar and well-known territory. This is why training and experience from a very young age is so important: familiarity.

However, training and experience has two sides to them: some positive and some negative effects.

Positive: They will more easily give an intuitive understanding of problems, dangers and obstacles, as well as an oversight over many of the requirements.

Negative: At the same time, it may tend to keep persons inside traditional (taught and not experienced) thinking, restricting the view of new aspects and possibilities.

These two points are included for two purpose: Consciously be aware of and learn from the positive aspects, try to avoid the negative ones.

Getting into a position (or state) where we more consciously are making the models, we are normally moving into less familiar environments. The normal situation is that no two modelling jobs / tasks are alike: each modelling job and the resulting model tends to be unique. Thus: using modelling in our daily work situation, we need to change our formal methods from task to task, both for analysis and for documentation form and contents.

We must also remember that the formal documentation is in reality one form of communication: partly with other people and partly with ourselves, partly over time, partly as some form of a discussion and elaboration (with oneself or with others).

In work situations, persons lacking training and experience within a field, and thus being "outsiders" to the problem domain, will normally meet what easily tends to become some severe problems:

- understanding the purpose of many of the aspects,
- unfamiliarity with the matter and the context,
- getting hold of the "unmentioned set" of requirements

Both the unfamiliar set of requirements and the "unmentioned set" tends often to be the more difficult ones. Trying to collect a complete set of them often fails, as many of the requirements are never mentioned: "everyone knows what those problems are, so we do not think of it as a requirement" is

often the answer when an "outsider" happens to stumble over one of them and remarks: "not included in any list".

May be most "insiders" do know it, but the "outsiders" do normally not, and the modellers tend often to be some "outsiders".

General on the modelling processes.

"Things to be modelled" varies widely. They are quite often processes, sometimes they are ideas, sometimes they are very concrete "things". Examples: repairing a car, how a board of directors are governing a company / a factory (processes); clothing fashion for next spring (ideas), a concrete based production oil platform to be placed on the sea bottom at 120 – 200m depth with a complete production platform on top (free from the sea), or sewage / drainage system for a city (things).

The models for the examples mentioned above will differ widely (repairing a car: body work, engine overhaul, a bus interior with broken seats etc.). Yet they are all models, resulting from a modelling process. The models will wary widely both as to form and content, as to information and to element sets. The various modelling processes will be widely different, but they will contain several common elements and also several special elements in well adapted combinations: "*elements*" here being both *things*, *objects* and *process descriptions* in various forms/formats.

One of the first things a modeller has to do when starting on a new task, is to study the modelling subject and job at hand, and compose a suitable modelling process for the job. In spite of what many people like to think, no standard modelling process fits all tasks. On the contrary, modellers / designers / developers with some experience do start the job by defining the model contents, based on a survey of the model subject. This survey must disclose and document which important aspects of the subject must be specified/defined/contained in the resulting model. From this documentation the modeller will select the modelling tools and elements needed, together with a specification / documentation of the required modelling process.

Experience with / from modelling

Experience means a person has recognised and faced the consequences of the decisions made in the modelling phase of some development (on experience: see below). It is (unfortunately) quite often not realised and understood that the modelling phase is the first (and the most critical) part of the design phase. Designers/developers base their work on the contents of the model, thus making it the foundation for the product.

If the model is wrong or incomplete as to the important aspects of the subject or to expected loads, the foundation is failing, and unwanted or unexpected "stress" somewhere in the structure will make it all collapse.

> It is tempting to compare this to robust buildings: St. Pauls cathedral in London, Pantheon in Rome and Hagia Sofia in Istanbul: close studies have shown that the basic foundation and subsequent loadbearing structure was perfectly matched to the loads. The Flavian Amphitheatre (Colosseum) and Maxentius Basilica, both in Rome, nearly made it: they missed the thermal load of the sun heating/cooling effect on the south side during centuries: an earthquake tore down their south side. Some other basilicas have collapsed due to improper foundation and/or unwanted/unexpected loads. We can also include "the bent pyramid" (Egypt), built by Snefru, father of Khufu (Cheops): unstable foundation/ground under one corner. Even some of the great builders could make errors.

Flaws in the model tends sometimes to pop up during implementation phase, but are mainly seen (and heavily affecting) the maintenance phase, weather the product made is a process, a computer based system, a building or some other product (e.g. a sour).

Experience with modelling thus implies that a person has been responsible for decisions made in the modelling phase, then subsequently been responsible in the implementation phase, and then been lucky enough to carry responsibilities in the maintenance phases. At the end, to really gain the experience, the person must reflect on what was good and what was not quite so good in the former phases, and what could have been improved upon.(This will mainly pertain to software systems.)

The use of "modelling" is extremely widespread: from fashion to making toys (for young and old people alike), from art to engineering, from military use to tourism, in rather most areas. It is hardly an area of modern life where modelling do not appear. So: What is Modelling? Giving a complete answer to this question is very hard and tedious, but we will try to come up with a rather limited but useful hints and points in the following sections. To some extent they may tend to lean towards modelling for computer based systems, as this for the time being is a rather relevant field. At the same time such models are normally abstractions of varies types of processes; these being rather illusive "items", hard to see and touch.

What is Modelling?

One definition (of several possible ones)

By making a model of *"something"* (later being called the "modelling subject" or simply "subject") we will in the following mean:
to make a representation of the important aspects of said *subject* for a given purpose in a given context.

Above all: one must document thoroughly: WHAT SORT OF PERSONS are going to USE this model for WHAT PURPOSE, and in WHICH WAY. This will quite often point to some important aspects.

"Important aspects" of the *subject* (and thus being part of the model) must include the goal it is intended to meet, the purpose it is intended to fill, all the requirements it is intended to fulfil and the loads it will be exposed to.

Without having all these three precisely defined, it is rather impossible to see if a given model will satisfy, and to see if something being built using the model as 'the blueprint' will be satisfactory.

Models are made for a wide variety of purposes:

- As a product in itself (e.g. in fashion, in art, in toys (model car, -airplane, -ship)),
- Made for information transfer (e.g. various types of maps,),
- Made as instructive specifications (e.g. construction drawings for a house, a ship, a car, an engine, a bridge, the administration functions of a company, the services offered by a company, etc.),
- Made for some further study or as a tool for the mind (e.g. a three dimensional model of some design, an atom model, a DNA model, a mental model for some abstract problem).
- Making the "blueprints" and the specifications for the development of a new software system/program.
- One frequently occurring model making is: Making plans for an activity, a project, a process etc.

The modelling job should be done in some specific cases:

- Initial study of the subject for a thorough *understanding* and *specification* of it *for the given purpose*. When the modelling is complete and correct, no one else has to look at the subject again; "the model says it all".
- Whenever the subject is *changed/changing*: a re-modelling must be made.
- Whenever the purpose is *changed/changing*: a re-modelling must be made.

In this paper we will first look at some examples of modelling in some historic perspective as well as some present day examples of modelling.

Modelling is not something new, it has been going on for at least 20 000 years. However, the importance of modelling activities has grown immensely over the last 100 years or so, and the "importance gradient" points still upwards (as does "the second derivative" of it).

I believe it is of some importance to keep a broad view on modelling in general, in order to have a good understanding of the various aspects of modelling. Thus the various examples presented below.

First we will take a look at modelling in general in a historic perspective.

Then we will look at some aspects of the modelling process

Towards the end we will concentrate on modelling for computer based systems, representing ***"Model driven software development".***

Modelling examples: in historic perspective.

Modelling is not something new. It has been going on in one form or another for quite some time, at least for the last 25.000 years..

In this text we will exemplify this by using some models of femininity over some years.

The first model example is the ***Venus of Willendorf***, also known as the ***Woman of Willendorf***, (fig. 1) is an 11.1 cm (4 3/8 inches) high statuette of a female figure. It was found in 1908 by a workman named Johann Veran[2] or Josef Veram[3] during excavations conducted by archaeologists Josef Szombathy, Hugo Obermaier and Josef Bayer at a paleolithic site near Willendorf, a village in Lower Austria near the town of Krems.[4][5] (Willendorf, a village in Niederösterreich. near the city of Krems). It is carved from an oolitic limestone that is not local to the area, and tinted with red ochre. The figurine is now in the Naturhistorisches Museum in Vienna, Austria.[6]

Figure: fig. 1

Age is uncertain. Dating of the site where it was found indicates it is 20.000 to 25.000 years old. (Some have estimated it to have been made between about 28,000 and 25,000 BCE.[1)]])

The “no face” – representation may indicate an idealized womanhood at the time: fertility, can survive famines, living at a place which normally has plenty of food. (important characteristics).

What was the purpose of the model? No one knows.

The next model example is considerably younger: Aphrodite of Melos (Venus de Milo) sculpture from Alexandros of Antioch, some time 130 - 100 BC, marble 2.04 m. high, today in Louvre, Paris. (fig. 2)

Aphrodite of Melos (or Milos) is probably the most known Hellenistic sculpture although no work survived from antiquity mentions this work. In a hand (lost) she holds proud the apple of Paris given to the kallistiti, the most beautiful woman.

Figures: fig. 2 & 3

The Hellenistic sculpture was unearthed on the island of Melos (*Melos* a Greek name for apple), one of the Cyclades islands, in 8.4.1820. It was inside a buried niche in a gymnasium in the ruins of the ancient city Melos (today the village Tripiti), .in a place dedicated to Hermes and Heracles. The statue was found in two pieces: the upper torso and the lower, draped legs.

The important characteristics of femininity are now changed: natural beauty, elegance.

What was the purpose? We do not know in detail. .

The last model example is considerably younger still, the mannequin showing fashion: Twiggy.

Figure: fig. 4

The important characteristics of femininity have changed once

again: now it is the slim line. (As in all three examples and as always: what is hard to obtain is always the desirable feature.)

The purpose? Commercial interests..

Modelling examples: more of modern day modelling.

The fourth model example is a map over some are of the ancient Rome, from some time after . 310 AD (Basilica Maxentius is finished), centred around Forum Romanum, stretching from the Flavian Amphitheatre in the east to the Capitolium and Forum Boarium in the west, from Trajan's Basilica Ulpia in the north past the Forum Boarium and a larger part of Palatin in the south. Also Cloacae Maximus is indicated in blue, from Transitorium / Forum Nervae down into the river Tevere, right under Forum Boarium.

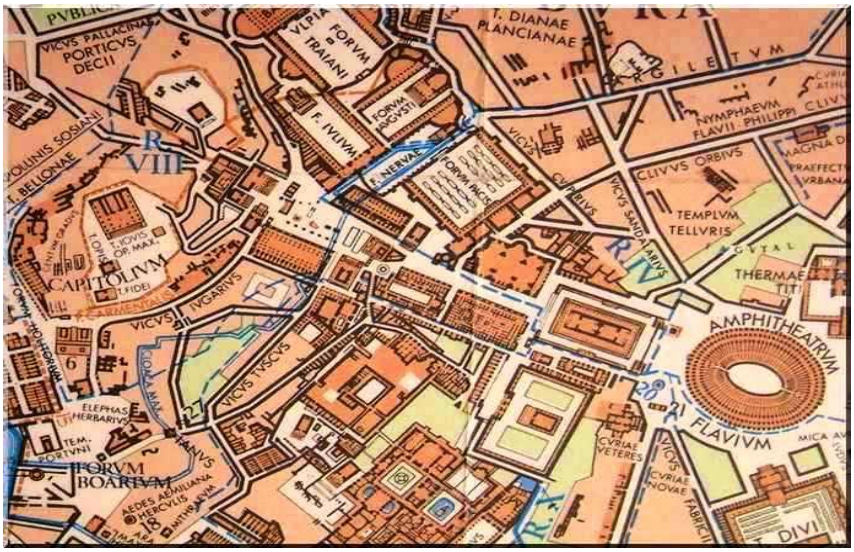


Figure: fig. 5

The fifth model example is a model of some of the same area as the fourth example, of the ancient Rome, from some time after . 310 AD (Basilica Maxentius is finished). This is also centred around Forum Romanum, which we see in the foreground.

Trajan's Basilica Ulpia is close to the top of the picture, below that is Cesars forum and Curia, towards Forum Romanum. We can also see the long and narrow Transitorium / Forum Nervae.

In the upper right part of the picture is Subura, (which is the origin of the word 'suburb').

Figure: fig. 6

Models in fig. 5 and fig 6 do to a large extent show approximately the same area at the same time period (approximately 320 - 350 AD.)

Do they give the same information?

Hardly. And they are for different purposes.

The model in fig 5 is a map, portable, and for orientation when you walk around in Forum Romanum.

The model in fig. 6 is a 3D model, an explanatory / visualisation model, built in 'hard' materials, (and is a rather stationary one at that).

This trigger off some important key words: *Purpose, Explanation, Requirements, Expectations.*

The sixth model example is taken from the development of a CAD-CAM system named Autokon (a sample from an example presented at the very end).

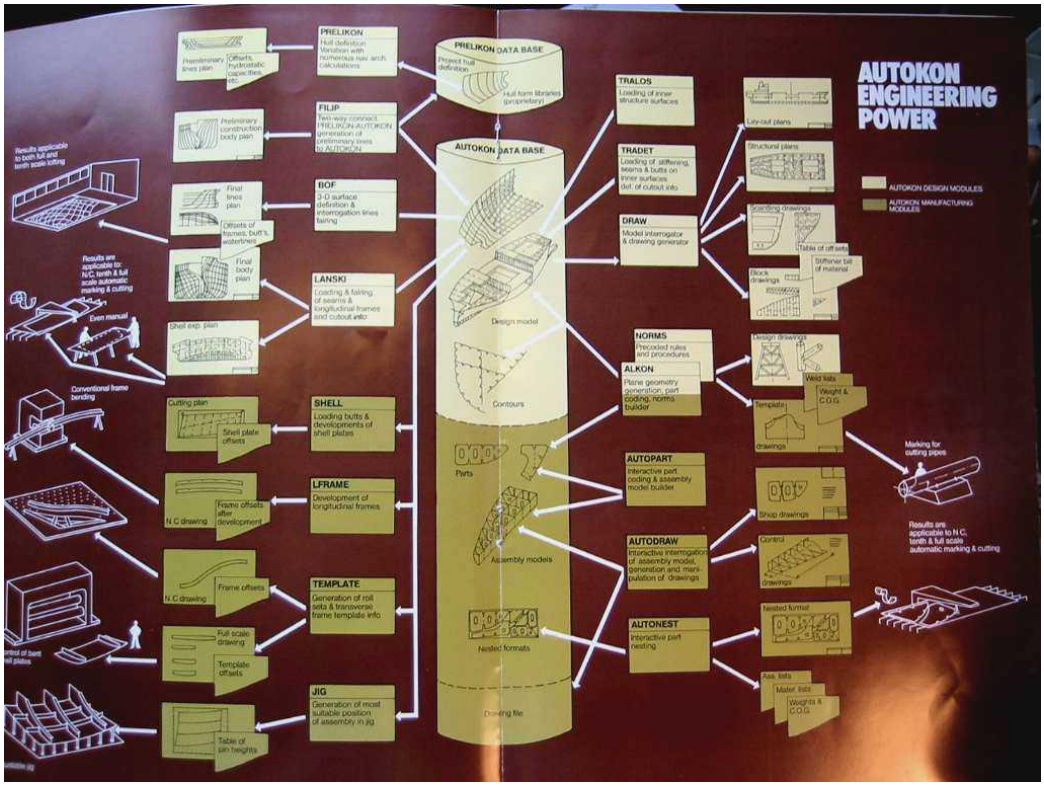


Figure: fig. 7

It presents the main parts of the system and their responsibility

In the middle id\s the Autobase, the Autokon database. The light coloured parts represent modules doing design jobs, the darker coloured parts represents modules doing harvesting and generation of control data for the various NC-machines and production data/specifications.

The seventh model example is another model presentation of the same system, used in a different context.

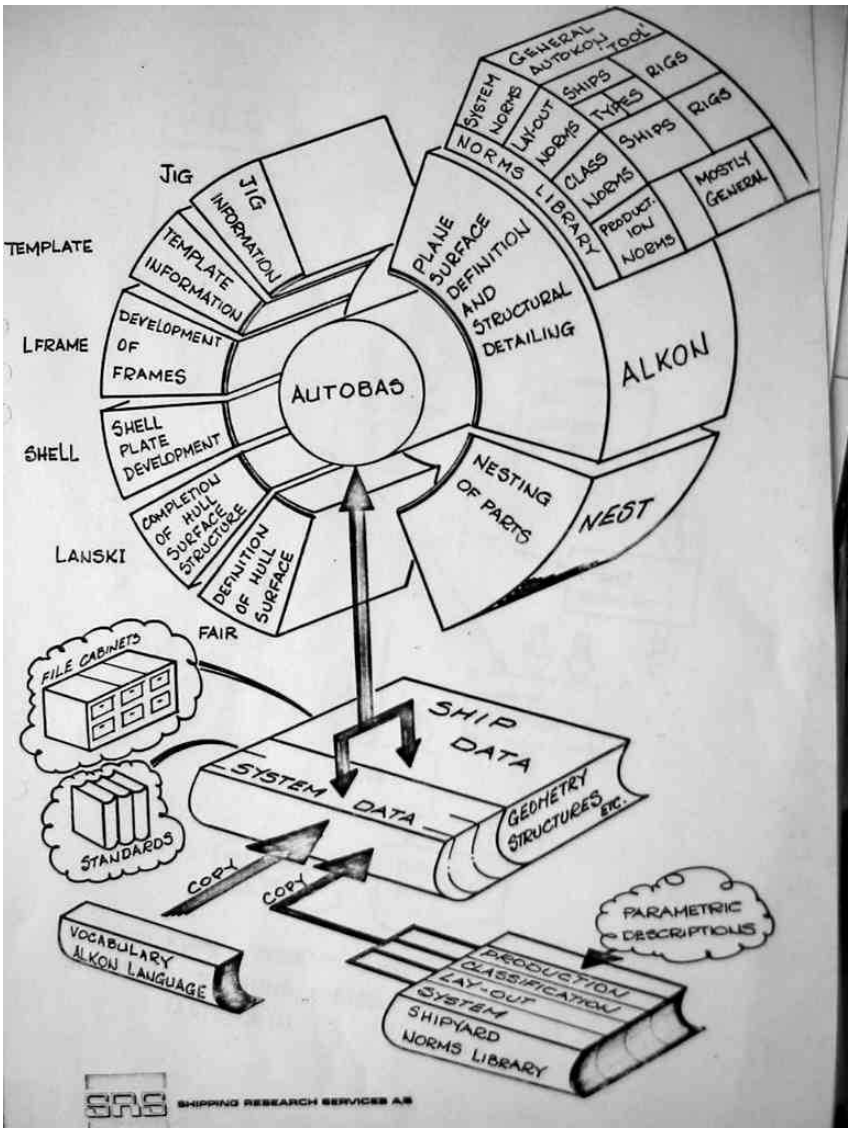


Figure: fig. 8

It should ne noticed that all the "norms" for the ALKON module are design rules and procedures (and some rules and procedures for making NC data for production machines and tools), all expressed in the shipbuilders language and terminology. ALKON thus contains compilers and specialized processors in addition to be able to cope with the various languages. (neither Germans, nor Frenchmen or Englishmen or Americans or Japanese wants to be forced to speak Norwegian: That is why you se (in front, left) that the vocabulary is part of the norms. Each yard may have their own language, their own set of tools of different makes, and their own set of rules for design and production. (They may of course also share these with

others to the extent they want.) If you enter the French, the German and the English vocabulary, ALKON can handle any mix of languages, in norms provided each word has a unique meaning. (If one and the same word appears in two different languages, it must have the same meaning in both.) Each "meaning"/notion may be written in several ways: RETT LINJE, RL, rl, Gerade, GR, straight line,SL all means the same thing and may thus all occur in the same vocabulary.

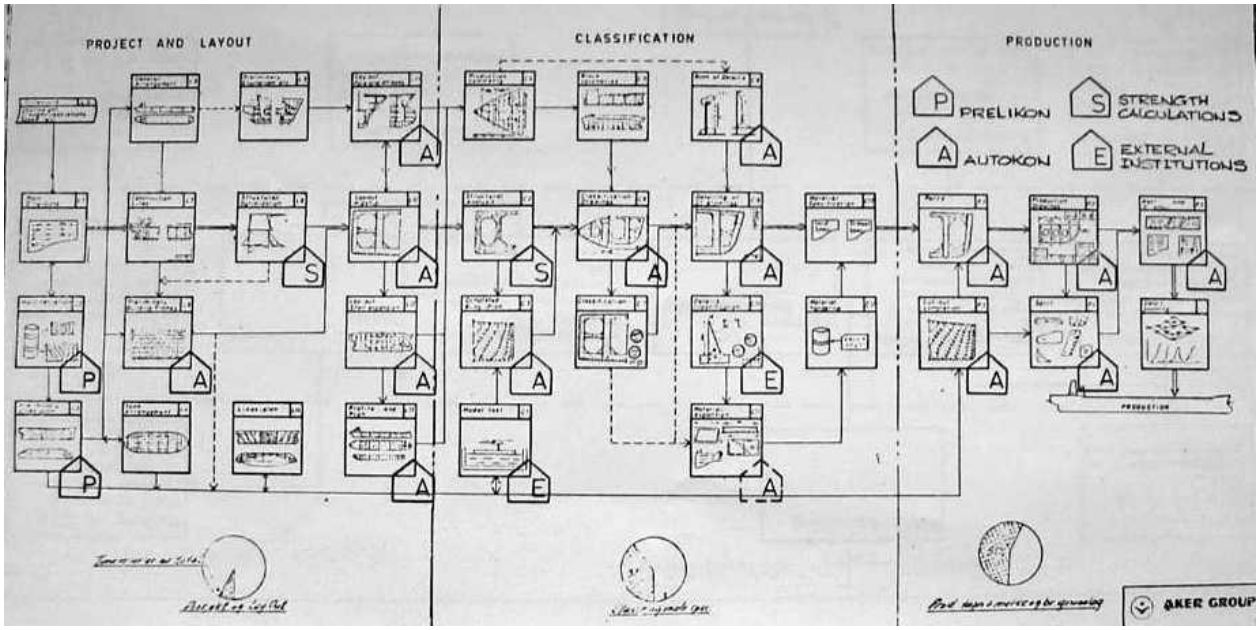


Figure: fig. 9

The eighth model example, (fig. 9)

is a model showing the process of making a new ship, and which software tools the yard was using in the process.

Important aspects of the modelling process

Before going on, with important aspects of the modelling process we will repeat when to do such a job:

The modelling job should be done in some specific cases:

- Initial study of the subject for a thorough *understanding* and *specification* of it *for the given purpose*. When the modelling is complete and correct, no one else has to look at the subject again; "the model says it all".
- Whenever the subject is *changed/changing*: a re-modelling must be made.
- Whenever the purpose is *changed/changing*: a re-modelling must be made.

The examples given so far trigger off the important key words which we meet time and again: *Purpose, Explanation, Requirements, Expectations.*

But first we will meet tree important, general points coming into play for most modelling efforts.

Even if the three first examples are showing some of the probably oldest and longest ongoing modelling activity, the notion of modelling and the benefits from using models has increasingly been recognized, and accordingly has spread to new areas of activity. The need for modelling methods and tools has increased accordingly, as has the various ways to display the functionality, the characteristics, the capabilities and the resulting models.

One area where modelling is being heavily used is: making plans, whether in job situations or in private life, for projects of various types, activities, or may be for travelling. Most points discussed below are fully relevant to such modelling.

After some general discussion on modelling at large, we will focus on one area that is rather relevant today, and may thus be of special interest to some groups of people: *modelling in connection with development of computer based systems*.

The expression: "*modelling in connection with development of computer based systems"* used above, is selected with intent and care. There is a widely spread misunderstanding that modelling for computer-based systems mainly is supposed to focus on a few apparently critical definitions and/or documentations, aiming at:

- The functionality of the system.
- An initial indication as to what sort of items (model elements) might be involved, or for some initial attempt at construction.
- While the more general focus appears often to be on: *get the system up and running as soon as possible and as cheaply as possible*.

This misunderstanding is often caused by a few factors:

"Physicality / concreteness". Making a model of or for something that is or will be a physical item is probably easier and more manageable than making a model of something tangible / abstract: Construction drawings for a building, a car, a ship, or a map of a city, or an artistic drawing of how a house will blend in with its future surroundings. These are all models of a physical nature. Such a model feels to be of a rather concrete nature, as the subject for the model tends to be of a concrete nature. In such cases, a feeling of correctness and completeness comes rather easily.

An example of such an error: An architect should design a new house for a friend of mine. It would have a rather nice view of the countryside, but unfortunately, from what had to be the end wall. The architect was not at a loss what to do: On the upper floor he placed a nice living room with the entire end wall being of glass. A brilliant view! Alas: when the construction company came, the question came up: What is going to support the roof? Having any angles around, available for that job?

A plan for or a model of an organisation and of how it should/will work and function is in the border area between modelling a subject being concrete and being abstract.

A political idea is rather abstract, and the implementation of its model tends often to be rather a failure. The reason for this may be that the implementation of the idea is made rather intuitively, without any proper model having been made, studied and analysed.

One big question is then: *how concrete or abstract is a computer program*? Many system developers have been working so close to systems over a considerable time that they feel the systems are rather concrete. They have real problems realising that such a system is abstract, (e.g. how much do the system weigh?) Other developers are so inexperienced and new to the game they do not really realize the abstract nature of a system. After all, the documentation is quite concrete, and it may need some experience to clearly feel the distinction between the system itself and its documentation.

b) **A lack of experience** on the part of the persons doing the modelling, the makers of modelling methods, methodology and/or modelling tools.

First of all: we must realize that modelling is an early, important and *highly critical part of a **system design***. Many people believe the design is something that follows later on in the development process. They tend to forget that the models being made initially is the basis for the next and then the following steps: The model is not thrown away and the design (whatever that is) is being restarted from scratch. On the contrary: the next steps are elaboration and refinement of what exists from the previous steps. Each step in the development process is based on the results of all the previous steps.

This means that if the initial model (the early steps in the development process) is wrong, the whole system will in all probability be way off the target! (Unless the whole process is stopped and all previous results, including the models are being discarded). This would be a severe waste of money, time and efforts, and must be avoided.

The responsibility of modellers and modelling. The modelling method, methodology and/or modelling tool has a severe and heavy responsibility in ensuring that the early models really are absolutely correct, in as much as meeting and fulfilling the needs and qualities presented by the

problem at hand. The model constitutes the real foundation for the whole system. The main difficulty in making a modelling method, methodology and/or modelling tool is NOT to figure out a nice way of describing/drawing/presenting various parts of the model (this is completely secondary) but to ensure completeness and correctness (whatever that is) of the models from the very start. This appears normally to be the hard part of any development.

One must avoid a situation described in a Norwegian fairy tale: Two men are going to meet, and they both plan what to say when they meet. One man is going to collect some items, the other on is sitting outside his shed, making a new shaft for his axe. They both are really within their own context and state of mind. Alas, their anticipation on how the meeting and chat will evolve is completely different. Neither one is trying the obtain an overall picture of the situation, both focusing on a small and limited part of it.

So, when they meet, the following exchange takes place:
"Good morning my good man." (He is trying to be polite, being on a rather touchy mission)
"Axe shaft". (He will probably ask me what am making)

Both men miss the over-all picture, being very self-centred, focusing on only their limited part..

In the modelling domain: *the answer must meet the needs of the entire problem.* How to ensure this to be the outcome?

The answer appears to be a methodical and systematic approach to three tasks:

a) Composing a methodology: Select and combine methods and tools in such a way that a methodical and systematic system development process takes place in present task.

b) To make sure that whatever methods and tools are selected, the system development is progressing in a strictly methodical and systematic way.

c) To make sure that whatever methods and tools are being used, the different parts of the system constitutes a consistent, coherent unit.

The discussions presented above may appear to apply mainly to software systems. It does. It also stresses tree important points:

a) No standard tool fits all cases.

b) Ensure a logical progression.

c) Make sure the elements cooperate correctly, no inherent misunderstandings please

.

General important points in modelling

Targets.

For all modelling there are a few important points of general validity.

I) What is the goal for the effort?

II) What is the purpose of making the model?

III) What are the requirements for the model? The full and complete set please.

IV) Fail-safe handling and pathological situations and actions. (Far too often neglected).

Complexity.

The subjects to be modelled may wary widely as to complexity. This may require insight, understanding, knowledge and/or mastering in a few or in many fields and topics from the modeller or from the modelling team. Some examples of varying complexity may be:

- A simple computer program to calculate the amount of paint needed to cover some item being limited by e set of rectangles and triangles. (e.g. outside walls of a house, a swimming pool, walls and/or ceiling of a room)

- A model for studying the working/efficiency of a gravity driven drainage/sewage system.
- A system for designing a fuel injection system for a petrol or a diesel driven car. (will contain sensors, actuators and control software. Typically: sensors for air pressure, air flow, air temperature, RPM, torque/effect required, timing (within engine cycle). Actuators for metering of fuel, fuel delivery timing and for fuel pressure (into intake manifold or into cylinder). Then software to collect and handle measurements, control the actuators and (possibly) to coordinate the various software parts/packages and to ensure stable, non-oscillating operation.
- Even more complex: design of an autopilot for an airliner, containing a navigational unit, aircraft attitude controller, an anti-collision system, engine controller and a communication handler including report handling of position etc. to AAC ASC, ATC.

An example on modelling points.

To try to illustrate these points, the making of a model aeroplane is being used.

I) I will participate in a competition for model aeroplanes at XXX, taking place on DD.YY.
I will participate with a model of a P-51 Mustang.

II) The competition includes both static appearance and flying capabilities. Therefore:

a) The plane must have a realistic outside appearance and a detailed cockpit inside.

b) The plane must have sufficiently good flying capabilities to fulfil the required predefined display patterns in a controlled flight.

III) The plane will participate in class CC.

a) Weight, size and engine must be within class CC limits.

b) The radio must be able to handle all 3 control surfaces, throttle, flaps and brakes.

c) The tail wheel must be connected to the rudder action, for ground handling.

d) The plane must be ready 6 days before DD.YY for practice and adjustments

e) Fuel supply and spare parts must be available with the plane.

IV) Fail-safe and pathological situations.

a) Ensure sufficient power supplies for RC equipment.

b) Bring proper size bag(s) to carry wreckage away in, in case of crash

Goal for the modelling effort.

Modelling goals is a rather complex part of the modelling effort, as well as an important one, often forgotten and/or neglected.

One way of looking upon this may be the <u>purpose</u> questions:

a) Why is the model being made?

b) Where is the model to be used? (in what way, and in which context)

c) What set of users will utilize the model or models?

d) What are the capabilities of the users? (Education, job training, their mental pictures and opinions of the jobs in the environments, their set of notions, the set of various "languages" they are using, their understanding of the notions within each language).

Another, way of looking upon this may be the usage aspects:

g) Modelling for informing and for orientation.

h) Modelling for understanding.

i) Modelling for some control- and/or production system

j) Modelling for users: Set of users and requirements to/from the users.

These two sections may have some common elements: both need some informative map-like representation, showing where things are, and possibly some sort of motion.

Point g) is mainly concerned with various types of mapping, e.g. maps of a geographic nature, or maps showing where something is located (physically, organisationally, etc.). Examples of the widespread use of such mapping may include: geographical maps of: areas, resource locations, meteorological data, city street maps, location of pipes and cabling (e.g. in a city, in a factory, on an oil handling platform, in a ship, an airplane, a car, or a building,

These maps have a wide variety of requirements connected to them, such as accuracy, dynamics, dependencies, readability. Etc.

For point h) we do primarily think of various types of processes, such as chemical, mechanical, the inside of living entities (physiological, mental), within organisations, in inter-human relations, various types of interactions. But models of "things" and "phenomena" are also included here, such as the Rutherford model of the atom, models of DNA, models of psychological phenomena.

Many of these processes are having rather strict requirements attached, such as to accuracy, dynamic behaviour and real-time constraints. These must be reflected / incorporated in the models.

(The notion ‘real-time’ is often misunderstood to mean: to operate in our concept of time or to be operation 'on-line'. The meaning for operational systems is: the reaction / response to an event must be present within a predefined time frame, given in the relevant time-concept, i.e. not necessarily in our perception of ‘time’.)

Point i) is mainly concerned with various types of process control system, ranging from controlling a hydro-electric turbine-generator set to controlling a nuclear reactor, or from the mating of the deck- and substructure parts of an gravity oil platform at sea, and then the proper placement of such a platform at its site. We can also think of smaller and/or less critical systems, such as heat and ventilation system in a house, or the injection control system in a piston engine, for an automobile or for a ship. Or what about the production line for bottles of soft drinks? Or how about controlling the production line of some medicine?

Point j) is the set of requirements from the model to the users, as to concepts and capabilities on the part of the user. It must also include the users of whatever products are derived from the models made, e.g. users of a software system/program.

This point contains some sort of duality:

What sort of users is this model intended for, and who is going to use it? Are intended user qualification or expected knowledge defined anywhere?.

Users of some product derived from this model: will they "understand" the product? Will they be able to use the system without feeling they have to fight the system?

Purpose of the modelling effort.

The main purpose is to study, understand and document all aspects of the subject, relevant to the given purpose and context.

Here we define the general functionality of the system and the various interactions:

a) The functionality of the model: what is to be accomplished?
   - how shall the interaction be between the model and its environments?
   - Which components do we see?
   - Interaction between components,
   - within the various model parts.

b) The set of user scenarios. A detailed definition of how the interactions (given in 'point a) are going to be carried out.

For a computer-based system, this means a complete definition of all the interaction between the various users and system. This includes:

a) A definition of all commands to the system: the functionality and the full set of effects / side-effects each command will have. (It also includes a precise definition of all the parameters to go with each command, as to information carried, value range and accuracy).

b) for each request to the system: a precise definition of all reactions and displays of results, from the system to the user.

Some readers may feel that these design rules and lists are too detailed. Well: These decisions must be made at one time or another. Experience, however, have shown that they should be made by persons having the full overview of the nature of the task at hand (the purpose of making it all), and of all implications in it. Leaving the decisions to some implementation face of a development will give one of two effects (or normally both at the same time):

1) Placing a far too heavy burden upon the implementers (of being fully experts on the subject/task as well as being expert implementers),

2) Resulting in an inconsistent and erroneous system (causing wrong reactions in some unexpected and possibly erroneous situations).

No-one can be an expert on everything: let the presumably better qualified persons make the decisions within the various fields of knowledge, expertise and.experience.

Requirements for the model.

This tends to be a set of non-functional points, covering various aspects of the model capabilities, restrictions, life span activities, and possible legal points. This set is strongly case dependant, although some points tend to pop up in most cases. A non-exhaustive sample set is given below in an arbitrary order, most of which should be compulsory.

a) Expected life span, (including a discussion on the premises for it.)

b) Accuracy, (including a discussion on the premises for it.)

c) Maintainability, (including a discussion on the premises for it.)

d) Reliability, of two kinds: *Correctness* and "*Up-time*" (including cost estimates for failures and a discussion on the premises for it.)

e) Dynamics, i.e. possible changes in purpose over time, (including a discussion on the premises.)

f) Readability, i.e. how easy will users understand the model, how easy will users obtain the information contained in the model and being intended for transfer to them.

g) Dependencies: on other tools or on other components. This is a complex point, consisting of several parts:
   - External tools, e.g. compilers or drafting tools supplied and maintained by other parties.
   - Component independencies, e.g. for complex models consisting of several parts: are the various parts independent of each other?
   - With respect to modifications: will changes made in one part show up in other affected parts as a need for changes?

h) Economy constraints, being a very important and complex point:
   - The Cost/Gain ratio over the system life-time of products based on the model.-
   - The design phase is normally controlling the major economic elements over the lifetime.
   - The economy of the various modelling phases, the initial one and later re-modelling ones.
   - Are proper checkpoints in place?

i) Time constraints. Also being a complex point:
   - Are all the time constraints from the subject included in the model?
   - Are the time constraints on the model development process present?
   - Are the proper checkpoints in place, both in the model and in the modelling process ?

Relevant points. Not all points may be relevant for all development situations. Before the start-up of a modelling task, a skeleton list of attribute groups and items must be set up and filled with all attributes/elements that may possibly be relevant. (This may be regarded as an initial TOC or a checklist). Striking one element out by writing: "irrelevant, because - - - -" is much simpler and safer than adding some element later. Such a "late addition" element normally means the development has to re-start from the point in the process where the "forgotten element" should have been entered. .

Computer based systems will normally need service and maintenance, although not all systems, e.g. systems made for a short-lived, special and well defined jobs A characteristic feature for this type of systems is: They have a well-defined start-up time and a well-defined shut-down time or situation..

One example: An oil-producing platform for the North Sea, built in two parts: the concrete-based substructure (150-200 m high) and the steel deck structure (some 50-60 m wide). These two parts must be assembled: the substructure being lowered at deep waters until the top of the legs are only a few meters above sea level. Two old tankers are then carrying the deck in over it until in the right position. The substructure is raised by evacuating ballast. The deck is welded on to the legs, and the whole unit is raised sufficiently high up to clear shallow waters on its way to the oil field.(A delicate operation, since the centre of buoyancy tends to pass through the centre of gravity, causing an unstable state.) It is installed at oil field by being forced down 5 -10m into the sea bed. During this operation, it is essential to have full control of the water level (ballast) in each of the 19 cells, each being 50m high and 20m in diameter.

The model must handle several real-time problems and measuring/instrumentation problems, e.g. how to measure deviations from "vertical", when it is very hard to know what "vertical" is? "Direction of gravity" may not be used; during normal operations there are 5 to 8 heavy tugs moving unpredictably in close vicinity, shifting direction of gravity far beyond acceptable tolerance limits.

Such jobs are very specialised and well defined, and the systems will operate for only a short time, typically a few weeks maximum, including some towing by tugboats and positioning (e.g. the Shell Brent B, some Statfjord platforms etc.).

Maintenance. Most Computer based systems are subject to wear and tear, although normally not of an abrasive nature. Makers of modelling methods, methodology and/or modelling tools must take this into account, and not only focus very short-sightedly on ‘how to quickly get it working’. For any reasonably successful computer based system, the costs of maintenance and trouble-handling normally tends to be far greater than the initial development costs. This is especially true if the system is not properly designed and prepared/made for maintenance.

And remember: the modelling is the first and the most critical part of the design phase.

As indicated above, experience with modelling is an important element for any maker of modelling methods, methodology and/or modelling tools.

Experience. So what does experience mean, and how to obtain it?

In short experience means to have personally faced some of the mistakes so easily made at the various stages in the system production process, and has had to take corrective actions ASAP, since costs are accumulating fast while the system is down, (e.g. 1200 persons standing idle, twinning thumbs, waiting for you to get thing going again. It is unpleasant.)

Experience thus means that the person in question has made (or been member of the team making) a full design of a system, (including the various models needed), has implemented the system fully (has been sweating over the incompleteness of the set of models), installed it on a few different platforms (swearing at the stupid and unnecessary differences between operating systems), trained a set of users (and improvised for some of the obvious-to-all-features that no one ever mentioned), corrected program errors that suddenly appeared (and has been told: after H hours n-hundred people will twinning thumbs / have nothing to do until the system is operational again)(any pressure here??), and implemented some of the additions / modifications that suddenly became necessary (due to the fact that the world around us tends to be changing over time), (surprised?).

In short: experience means: to have gone through a full life cycle for at least one system and has personally faced some of the mistakes and troubles easily made in the process, and has had to take corrective actions (ASAP).

A common misunderstanding in connection with modelling experience is how and when to obtain it: Making a complete set of models for one or more systems using a given tool, means the person has gained experience with that specific modelling tool, but not gained experience in modelling some system for proper, correct, complete, reliable, and economical operation. That experience is only gained after the models has proven their quality in real life use, i.e. during the implementation phase and in practical operation over some time.

Is this statement a bit hard, wrong and unjust?

Just as experience with a tool only is gained after the tool has been truly used and tested in real life, experience in modelling is only obtained after the model has been truly tested in real life.

Model types. Normally modelling for a computer based system will include (in addition to due consideration for service and maintenance) elements of:

- Modelling for orientation.
- Modelling for understanding.
- Modelling for informing.
- Modelling for production.

Any system being used as a tool must fit ‘the hand using it’. It must fit as to terminology, to functionality, to “ways of considering the problem” (ways of “thinking”/”mentality”) and as to adaptability.

This implies that the system contains the same “mental model” of the task at hand as the operator (the tool user) has.

As some rather simple (and obvious) examples of tasks, let us consider extreme cases of lifting devises and tasks: as one extreme: ‘lifting a cup of tea’, as another extreme: ‘lifting a sunken ship’. There are many ‘lifting tasks’ in between these two extremes, such as loading/unloading a ship, an aeroplane or a truck.

And to make it all a bit worse: concrete – abstract lifting: lifting the constraints of passage through a given border passing gate (e.g. checkpoint Charlie in Berlin in not too distant history), lifting a blockade (or for that matter: 'lifting some art to a new level').

One may say: lifting tasks are not performed by computer based systems. Quite correct. But what about *controlling* the lifting operation? What sort of ‘mental picture’ would such a system need?

One must keep in mind: There is a wide variety of tasks. No single modelling type or method fits them all (at least at present, and may probably never do so).

At one time there were some people being rather eager at standardising everything. As an example: One bright fellow came up with a brilliant idea: how about a wall-mounted standardised shaving machine for all men. Early in the morning they could just stick their face into the machine and receive a quick and efficient shave, even before they were fully awake. “But not all faces are alike”, someone objected. The answer came promptly: “not initially, but after the first shave they will be”.

Would tasks and computer based systems suffer the same fate as the faces? There is a saying: “there is no substitute for real brain ware”. But again: many rather substantial statements have proven to be not quite correct. Let us keep an open mind; attacking the problem from a different and unexpected angle may give a different results.

As of now: let us stay with the assumption that tasks of different natures are best handled using different types of approaches and of modelling, although it is both permitted and rather useful to learn from each other. (and to "set the brain into some gear".)

More on requirements to computer based systems.

We are here going to discuss in more detail some topics primarily pertaining to computer based systems.

Some requirements that tend to appear in many models are: expected *life-span* (may influence the need for maintainability etc. to some extent), *accuracy* of e.g. numeric values, the *dynamic nature* of models, inter *dependencies* between various models and between models and systems made from those models. Users are normally concerned about *reliability (correctness of results, resistance against failing/crashing,* as well as *maintainability* over the expected life-span. (i.e. having the proper functionality at all times). For all models proper *readability* is very important (*easy* and *fast* to get an *unambiguous* understanding of). The *economic constraints* involved (cost - gain, win – win situations, limits) are one of the all-important guidelines, as are the *time constraints*.

Expected Life Span

How long is the problem/need going to be present, what sort of changes are expected in the problem/need and some estimate for when. This may be regarded as a problem/need profile over time.

It is equally important to have a corresponding gain profile over time: amounts and changes.

One must bear in mind that these profiles are guesses and estimates.

These profiles are mainly of interest for modelling for computer based systems, where they are of special interest for subdividing models into model elements: which modelling element to use, and how to subdivide these elements further.

These profiles may also give an indication of maintenance costs and of the gain balance.

Simple sum-up: Expected (duration of problem/need) – (stability of problem/need)

Economy

This is a double point:

1) The cost of obtaining the model (and the subsequent maintenance by someone, e.g. regularly to by "the new version" or "the updates" for the system.

2) The cost of producing and maintaining the system defined by the model. This normally means a proper and sober cost / benefit evaluation.

For this question to be relevant there must exist a need to be met, some gain in sight, something that can be done better, faster or cheaper. The various aspects of gain must be clarified and evaluated (must be assigned a price tag). Such needs appear in three major types:

- A solitary need (only this company need the system) (e.g. a control system for a specific nuclear reactor)
- A specific, few parties need the system. (e.g. in the production line of specific parts in some few factories)
- A general utility need by many parties. (e.g. a spread sheet program)

A potential developer must balance the total value of gain against total costs (development costs, possible marketing costs and marketing gains, and sober maintenance costs).

As to a computer based system there are two ways of obtaining it (in addition to stealing it): buy one or develop one. We will here focus on the latter alternative: to develop one.

However, one should always remember: regardless of how it is obtained, the cost always is a set of cost elements (as is the case also for other types of "things":

a) Acquiring it. (whether buy or develop)

b) Introducing it. i.e. starting to use it, including training and possibly changes in production lines.

c) In case of new development: possible marketing of the new tool.

d) The various aspects of maintenance.

For a production company (as for several other users), maintenance is a rather complex element:

i. Changes in set of production tools, e.g. new tools.
ii. Changes in set of products
iii. Changes in product demands.
iv. Changes in set of raw materials.

v. Replacements in the set of computers: Source/types, OS., other elements in the platforms.
vi. Change in management
vii. New technology somewhere.
viii. How about bugs/errors.?
ix. And how about losses caused by system breakdowns?

For the four elements a – d, d) maintenance is nearly always the by far largest cost, contrary to normal belief and what one feel initially.

Maintenance also tends to be the major system killer, long before the problem it should handle is solved/disappeared. The ratio Cost/benefit exceeds 1, and high maintenance costs tends to be the major cost element.

One fact often being overlooked is that the main key element for maintenance cost lies in the solutions selected for the design of the initial model. The modelling process and the model quality can easily cause a factor of 2 to 3, up or down in the maintenance cost. This is one reason why the modelling tends to be one of the most critical parts of a system development.

Simple sum-up: Gain vs. cost (purchase – development – marketing - maintenance)

Accuracy

One aspect of accuracy is the ratio between the largest and the smallest value reliably to be represented in a system. The max value and the min value must explicitly be specified.

It also includes some evaluation of whether the units of measurements selected for use in representations are wisely and correctly chosen.

Are all representations of values/magnitudes kept to the same accuracy?

NB. Not all models may need an accuracy evaluation. However, in all models a statement should be present confirming that the accuracy problem has been considered, together with the conclusions reached and the premises for them.

Accuracy and reliability have a common point: are the results presented to be relied upon as being correct?

As an example, let us think of some sorts of maps: for a tourist mat an accuracy of some 10 – 20m may satisfy, for a map of city piping (water, drainage, cables) dug down in the streets, an accuracy better than 0.4 to 0.5 m must be expected, while for piping in the design of cars and aircrafts an accuracy of 1 to 2 mm or better may be required. In the design of an IC quite a few decades better is required.

A hologram may be regarded as a filter, consisting of a set of lines. For these lines to work as an optical filter, according to Huygens principle, i.e. dimensions must be in the vicinity of the wavelength of light. This means an accuracy of ca, 15 Å

In accounting systems two valid decimals may normally be required.

These requirements must be designed into the system, not left to some arbitrary chance that some implementer happens to be sufficiently awake and up to this problem. For instance: where are the various operations specified? What types of parameters must be used in order to ensure the accuracy? Details? Yes, but extremely important details in most cases. As an example: where can we use floating point representations? Is everyone aware of how notoriously inaccurate REALs are? (Just think of taking the square root of 2 and square it again: you hardly get 2 as an answer in most programming languages. Or how about dividing 10 by 3.0, and then later multiply the answer by 6. Do you get 20 as an answer? Descent programming languages do give you 20)

Let us as an example take a look at a floating point representation (a normal REAL), used for representing currency: two valid decimals required. What is the value range permissible in order to maintain the accuracy?

A normal REAL is using 32 bits representation. 1 bit is used for the sign bit, 8 bits are used for the exponent, leaving 23 bits for the mantissa. In order to maintain two valid decimals, 13 bits are needed (at the very end of the mantissa, but inside it.). (Remember that each step in the ladder in the

two-decimal sequence is a sum of binary fractions.) This leaves 10 bits for the integer part of the number, i.e. a range from -1023 to +1023.

The dirty part of this fact is that if we enter a value of say 2345.67into such a variable, do we normally get some error or warning? In most programming languages: NO. We can normally not define explicitly which accuracy we want. Only in advanced languages can we declare something like: "I want to declare a variable ABC having 2 valid decimals".
The only result of entering an out-of-range value (like the 2345.67) into the variable ABC defined in C or in C++, is that the accuracy requirement is no longer valid, it is removed.

Whenever some special accuracy is relevant, proper actions must be made / specified in the modelling process. This is always true, whether the accuracy is 0.01 or 100.

Simple sum-up: What is the maximum error/deviation from "the true value" that the problem / task will / may tolerate?

Dynamics

Is the model of a static nature, or will it wary over time. If it is non-static: when was it issued, for how long is it expected to be valid, and who is responsible for keeping it properly up-to-date?.

Simple sum-up: When will the system be obsolete / out-dated, and for what reason?

Dependencies

Which other models, systems and sub-systems are dependent of the model or the models, and on which other models, systems and sub-systems does this system depend?

These dependency data may be kept with the set of models, with the dependent systems, or both places. Or, it may be given as a separately defined description, referred to by all relevant parties. (a dependency tree).

It should always be kept in mind that more than one other model, system / sub-system may depend on a model or set of models. Whenever a modification is required at one place, the entire dependency tree must be checked for consequences and possible side-effects.

If the model Is of a non-static nature, who is responsible for required updates / maintenance?.

Will updating of models and possible dependent systems be done automatically, and by whom?

Simple sum-up: Dependency trees are required for relevant systems parts.

Reliability

This requirement normally contains two parts combined into one: the system gives you the proper service you expect from it:

- The system does not suddenly quit / crash / disappear.
- You can rely on what it delivers (results / answers are being correct)

The last point includes '*correct computations*' in all variations, as well as the expected *security* against theft or malicious manipulations of data and / or procedures.

One or more parameters should be included here: Costs incurred if / when the system is failing in one way or another. (Various ways of crashing, theft of data or corrupted data.)
It must be kept in mind that data corruption may also result from failing accuracy or reliability.

Simple sum-up: to which extent can I trust the system (faults, secrecy?), and what may the cost be if it fails.

Maintainability

This requirement is closely linked to the life span profiles and to separation of concerns in the selection of modelling elements.

The maintainability is partly a requirement and partly a modelling / design element: how to obtain the proper maintainability by obtaining a proper separation of concerns and by selecting a proper modelling type and modelling strategy.

Proper handling of life span profiles and the maintainability may affect the maintenance cost by an significant factor.

Simple sum-up: Keeping control over the maintenance cost by proper separation of concerns and by selecting a proper modelling type and modelling strategy.

Readability

A model is made to create an understanding of some problem area. If seen / read / used by more than one person, it is of great importance that ambiguities are avoided, i.e. that all readers have the same understanding of the various aspects, even if the information they receive may differ considerably. (Remember here: "Information" is the knowledge / facts that the readers receive in their minds.)

The main aspect of readability is thus: the same common understanding must be created in everyone, even if various readers obtain unequal amount of understanding (understand unequal parts of the '*universe of discourse'*.

A model representation giving rise to ambiguous understanding is not only close to the point of being useless, it is on the brink of being directly dangerous.
Proper readability is thus of prime importance.

Simple sum-up: Fast and easy creation of an unambiguous understanding

Time constraints.

What are the time constraints on the model development, and are proper checkpoints in place?

This really means to make a model within the modelling work: a development time related plan.

A proper set of check points in the development plan is important in order to make the whole process manageable. Proper check points permit relevant actions to be made whenever some unforeseen or unexpected occurrences take place. These may represent possible shortcuts to be made, or they may represent unexpected problems, of a technical or an economical nature (or both).

Such occurrences are most often manageable when detected sufficiently early, detected to late they can at best create experience.

Purposes of models.

Modelling for informing and for orientation.

This section and the next one have some points of contact where they touch one another: both need some sort of a map, showing where things are, where to proceed, etc.

This section is mainly concerned with various types of maps, e.g. maps of a geographic nature, or maps showing where something is located (physically, organisationally, etc.). Examples of the widespread use of maps may include: geographical maps of: areas, resource locations, meteorological data, city street maps, location of pipes and cabling (e.g. in a city, in a factory, on an oil handling platform, in a ship, an airplane, a car, a building, books on subjects in a library, - -)

These maps have a wide variety of requirements connected to them, such as accuracy, dynamics, dependencies, readability etc.

Modelling for understanding.

We do here primarily think of various types of processes, such as chemical, mechanical, inside some living entity (Physical, physiological, mental), within organisations, in inter-human relations, various types of interactions, etc..

Many of these processes are having rather strict requirements attached, such as to accuracy, dynamic behaviour and real-time constraints. These must be reflected / incorporated in the models.

These models may in addition have some more common ones, such as inter-model dynamics, inter-model dependencies, readability etc.

(The notion ‘real-time’ is often misunderstood. It means: the reaction / response to an event must be present within a predefined time frame, given in the relevant time-concept, i.e. not necessarily in our perception of ‘time’. It is often mistakenly taken to mean some 'on-line' situation. 'On-line' will often have an element of ‘real-time’ operation attached, as too slow response may have a negative effect.).

Modelling for design in general

In some respects this may be regarded as specific cases of the two types mentioned above. If we are making some item so big or complex that we are unable to keep all relevant data, constraints and considerations in our head continuously, we may have to note them down in some orderly manner. When more than one person is involved, the descriptions must be in forms familiar and uniquely understandable to all involved parties e.g. a house, a machine, a mprocess.

Design for a house. A house is so complex that a larger set of drawings are normally needed: the architect drawing of lay-out and facades etc. Then comes the specialities: drainage and sours, water supply, foundations, wall structures, electricity and electronics, relevant HVAC. etc., specialised drawings, but interconnected through common elements. Although lots of efforts are put into trying to avoid it, interdependency is a serious matter: changing something somewhere tends to cause lots of modifications within several other disciplines (e.g. moving a wall of a service shaft, like for an elevator, for HVAC, or moving the place where gas, electricity, water, drainage etc. is needed in some quantity).

Design models tend to be of such a complexity that it will consist of several parts, some of map-type, some of process-type. The subject of the design is often of a rather complex nature. Consequently it is a rather difficult job to split the whole thing into a set of subparts or subtasks in such a way that each subpart appears to be a complete unit in itself. At the same time each subpart may itself be a complex entity to be further subdivided in a similar way.

It is equally important that at any level, everyone can at one glance see all data / information / possibilities relevant to what is being done. But in this type of complex decompositions several 'unseen' dependencies do exist, and may not be observed by the users, and thus tends to represent a severe problem. Essentially this is a synchronisation problem, handling the 'unseen' dependencies.

This turns out to be an interesting problem for the modelling tool builder: how to supply the model builder with the proper tools to handle such evasive connections / dependencies between the various part models in a complex decomposition job?

In some way it may remind of the various levels of activity when a person is writing some part of a document: the writer is focusing on selecting the proper elements (characters, words etc.) to convey the ideas, and do not care which procedures or program parts are activated by the activity. When the key for ‘A’ is pushed, an A is expected to appear where the cursor is, without caring how it got there. That is the responsibility of some other process, occurring at a different (lower) level.

Those making the document handling system, (the lower level processes) were working on several tasks: composing characters into storage units, handling the cursor, displaying characters in windows etc.

But in turn: they did not care about which elements in the ICs etc. are being activated or deactivated, (which were turned on and which was turned off by their actions), that was again a different (lower) level activity.

And yet, the author and the system builder (and the system) may have some aspects of common interest: various aspects of storage. Where should the collections of characters be stored for later retrieval? Apparently a common interest, but from widely different perspectives: The system must perform storage management, while the author wants a handle to it as an entity.

Modelling for design of computer based systems

Normally modelling for a computer based system will include (in addition to due consideration for service and maintenance) elements of

- Modelling for orientation.
- Modelling for understanding.
- Modelling for informing.
- Modelling for production.

Some discussions on those subjects are also placed ahead of this section.

Computer based systems are rather abstract things, sometimes handling items of an abstract nature, e.g. a system for activity network planning, or a document handling system. These two are both examples of abstracts: the first one being an abstract handler of abstract items, the next one being an abstract model of something concrete, a document, when printed.

This abstract nature may make modelling easier to handle, yet harder: It requires a deeper understanding of the matter at hand, several systems / opinions may be present among users, more 'solid' feelings and believes, and some variations as to the semantics of notions and protonotions.

Computer based systems are also deeply involved in various aspect of communication. Here we may get into more difficult terrain: pictures and movies (as examples) are they abstracts or not? What when they are printed or put on a CD / DVD? What when being played, and what about music? We may suddenly be in a market of selling and buying. What about various types of mail? Or searching for data? Here are statements, accusations, opinions?!

It's a confusing mess!

But, after all: we can sort them into a few categories.

- Text (character representation),
- Pixel sets, items or streams,
- Octet streams to be passed on to (fed to) special HW. (sound)
- Some piecewise mixture of pixel sets and sound octets.

'Octets' may also possibly be replaced / substituted by 'Bytes'?!. Those two notions do not necessarily have the same meaning, even if both at present are of the same size. ('Byte' is the minimum addressable unit in the memory, while 'Octet' denotes a unit consisting of 8-parts, e.g. 8 musicians playing together.)

The categories given above are examples of DATA to be handled. After all, computer based systems are handlers of various types of data and data collections in various formats, some standardized, some may be specially designed for given systems / applications.

Whenever we are making a model, we may be working on a set of tasks at several levels, each one providing a specific service to other tasks. For complex items it is crucial that the whole is split into the proper set of tasks and levels, such that each task has a unique, well defined job to do, taking on a well specified and unique responsibility in the given context.

One level (or more levels) might with some advantage be regarded as a well composed set of specialised workers making up a smooth running team.

Modelling paradigms.

Modelling for computer based systems may be regarded as variations over three main types:

A) Use of Patterns

B) Data set storage and display

C) Modelling some sort of process or processing.

Use of Patterns

Paradigm A. For a computer-based system, experience has shown that "use of Patterns" tend to give good, reliable and maintainable systems. It consists of a set of elements that cooperate in a special wav.. They tend to promote a proper, methodical and good use of *components*, i.e. separating a system/problem into a set of cooperating units, each one having assigned an exclusive responsibility. If one unit/component needs to do some job/calculation within the responsibility area assigned to someone else, it will send a request to the responsible unit saying: "will you please do so-and-so job for me" (normally using something resembling a "procedure call"). This is so whether the units involved are people or components in a computer-based system (or some other type of units)

Exclusive responsibility assignments must be respected if safe and maintainable operation is to be achieved.

An example: Among other things in a system: it needs to handle points and curves. Component A (assigned the responsibility of some type of planning) needs to know the distance between point p1 and curve c3. It sends a request to component B (assigned geometry handling): "Please give me the distance from p1 to c3".

Only component B knows the representation of points and curves, everyone else only refer to such items by names. If for some reason some geometric representation needs to be changed, it will only affect one component (B), no one else. If curves and points were made available to everyone in the system in some common area, any change in geometry (or some other storage) it would cause the entire system to be modified (as it is then unknown which part in the system might access what).

The Model – View – Controller (MVC) Paradigm.

This is a paradigm used in computer-based systems, and originates in Object-Oriented (OO) system development. In OO everything consists of cooperating objects (components) An object consists of a set of messages it can understand and react correctly upon (often regarded as "procedure calls"), and (most often) a set of objects. We may regard OO as systematic, recursive modularisation. Even an Integer is an object, understanding arithmetic operation (+, - etc.) and operations such as: "as Integer", as Real", "as Text", "as Fraction" etc. (Smalltalk and derivatives from it)

An MVC consists of 3 components: a model, a view and a controller in a computer-based system: The *model* is the description/definitions only of the given item to be represented. The *view* and *controller* appears as a pair, each one knowing about the other and about the model.

The *view* is responsible for displaying the model in some given way/context.

The *controller* is responsible for handling all commands relevant to this display.

The *model* do not know about the view or the controller, but it will normally have a list of *"dependants"*. Whenever someone changes something in the model (using a proper message), the model will notify everyone in the list. The view has initially notified the model: "I am one of your dependants". So, whenever the controller (resulting from a command) has caused some change in the model (using a message to the model), the model will notify the view: "so-and-so is changed". The dependant (here the view) will then decide on whether this should cause any action from me.

Complicated? It may appear so, but is a very powerful way of doing things: it opens up for the use of several dependants, which all will work perfectly, e.g. several simultaneous views on the same object/model. How about several users accessing the same item? They will all see the same item,

and be notified of any change, and react on it provided it is relevant for their use of the item. If they are working on other, unaffected aspects of the item, thy will not react to the modifications.

Data modelling

Paradigm B) is normally known as 'Data modelling'. It may be regarded as tabular in nature: a set of tables, each with a fixed set of columns and a fixed or variable set of lines. Any change in the set of columns in any table (whether in number of columns or meaning of the entries in a column), or the set of tables will normally be rather costly and tends to be error prone.

It may be quite useful in many cases; we will here subdivide into four categories

None of these modelling types should make predictions, as this would include some functional or process element, which is not present in a mere data set. The presents of functional or process elements would place it within type C.

B1) Given a fixed set of data elements which we will display in some given ways. The set will normally contain several columns and several lines. The data elements may be presented in various patterns, and may thus be regarded as some sort of statistics.

The display patterns may be applied to several data sets of similar shape, possibly with various number of lines.

B2) A continuously augmented data set, corresponding to the static set in B1). The augmenting may be done by some automatic logging, or by some periodic updating. Some mechanism for reduction of the data set size must be made. The augmenting frequency may normally be of importance.

B3) Data sets of type B1) or B2) where the user may select the display patterns within given predefined sets.

B4) Data sets of type B1) or B2) where the user may define the display patterns dynamically.

This will require a specific display pattern definition language, including element selection mechanisms. Element selection could be based on element position in the tables, or on element value criteria.

Some of these display mechanisms may require the data to be stored within a relational database, as some of the selection mechanisms may be depending on data element value criteria, not only on their positions in the tables.

Process modelling.

The C) type modelling contains various types of processing elements, either of a functional nature or as models of some process. A function is here regarded as something having the syntactical position of a value element, while being semantically defined by an internal and / or external combination.

As an example: the trigonometric functions, being semantically externally defined, while:

ValA<-Fun(Fun234(56, 24), 678)

may be internally semantically defined, dependent upon context,

while Hyp<-PytagHyp (4, 3) should return 5 (and be externally semantically defined).

Process modelling contains four main types of modelling:

C1) Pure Functional modelling

C2) Component modelling (with randomly selected elements)

C3) Pure Responsibility modelling

C4) Combined functional and responsibility modelling.

Pure functional modelling as in C1) is rarely applicable, otherwise good.

Component modelling C2), using elements. Randomly picked elements should whenever possible be avoided, for reliability and maintenance reasons at least. This method is too 'muddy' and error prone.

C3) and C4) are cleaned-up versions of it, and should replace it. Normally the main reason for selecting it is lack of knowledge and experience (stumbling into the task, feet and back first).

The two types C3) and C4) are both the best choice in most cases. They are both a cleaned-up and useful version of C2). All components are selected with care, assigned an exclusive and unique responsibility. These types of modelling are also well suited for distributed systems.

Whenever some specific single-responsibility task is to be modified, that specific component is to be changed. Unless some operation on that task has been formally changed (e.g. requires 4 instead of 3 parameters) no other component is affected. In case an operation is formally changed, the dependency tree comes in rather handy: from that tree one may read directly which other modifications are required.

The analogy to most well operated companies should be noted: they are all divided into various departments, each one with special responsibilities, e.g. Sales dep., Accounting dep., Production dep., Maintenance dep. These may again be subdivided in a similar manner, such as: Production of X1, Production of X2, etc. or: Marketing of X1, General PR, etc.

Modelling processes.

The modelling process is far too often depending on the tool being used, which is rather unfortunate.

Tool selection is a rather critical aspect of the modelling process. Alas; it tends to be a rather random choice: what happens to be "in" in the environment. Students are in the habit of learning, and sometimes they learn unconsciously when they should not: they pick up whatever is flouting around, unfortunately often without critical evaluation. "ah, something that looks promising, something new, (at least to me), maybe I can use it." For better and worse. And then they bring it along into work places. (We must not blame the students; they do their job of learning. Whoever is in charge of the development has to bear that burden.)

Tools, especially free ware, should be used with some suspicion and great care. Remember some of the points above, those about readability and those about hidden connections between model parts. Make sure the various aspects are reasonably covered, or make sure they will be covered outside of the tools.

Summary point: Select carefully the tools and the process for the job. Do not make some arbitrary selected tool decide upon the process and the quality / usefulness of the product.

The modelling process should follow a concisely, predefined pattern, based on a set of specific, well defined requirements selected for the product in question.

Tools are nice to have and use, especially so if they fit the task to be handled. A good toothbrush is a very useful tool to remove unwanted residue (between teeth), but I would not try to remove unwanted residue (a big tree left by a former owner of a property) by cutting it down using a. toothbrush. So: a tool for removal of residue may not always be a good tool for removal of residue.

So, let us select a proper set of tools for this removal job: make an axe and a good chain saw replace the toothbrush, and get on with the job.

Next we may notice: the sequence of events tends to be important: It may not be a good idea to start the removal of the tree by first cutting off the very high top and the upper branches of the tree, even if we are being told to start at on end.

Many system modellers tend to do just such a thing: start at some wrong end. They start selecting model elements / components and interactions even before the goal, the real purpose and the set of requirements are properly defined. Even the full and proper functionality of the system is still not properly defined, nor is its operational surroundings. (What will be will be, regardless of what is needed, expected or required).

Summary point: Select carefully the proper sequence of actions in the modelling task.

No fixed sequence of events in the modelling job will always fit all instances. But, some general approaches may be given as an outline:

- Move from requirements towards solutions,
- Move from the general to the specific,

- Move from the surroundings towards the core of the system.

These points should not be followed sequentially; on the contrary: iterations must be made in sequences specific for just "this job".

Another more concrete guideline is:

- What is the goal?
- What is the purpose and functionality?
- What are the requirements and surroundings?
- What will my future life look like?

These points may contain several sub points, all of which are discussed previously, especially those on requirements, surroundings, marketing, maintenance and future.

At places with experienced persons, some variant of a sarcastic statement may quite often be heard: "I have got the solution, what was the problem".
This is in recognition of the fact that it is very easy to jump to a premature conclusion. And with some experience such occurrences are recognisable.

In modelling (as in the other parts of system development): the result must be built step by step, one on top of the other. Do not try to build a step "out in solid thin air", it does not work.

At the same time there is a saying circulating in "aviation environments":
"Learn from others, you do not live long enough to make all the mistakes yourself."

That is also valid here.

Frequently occurring mistakes.

One rather commonly occurring mistake in modelling is improper use of data typing, e.g. highly improper use of INTEGER and REAL types, normally caused by lack of experience and/or lack of insight into and understanding of this type of work.

Do use a representation valid for the nature of the value. e.g. do not use an integer to hold a telephone number. (example: part of the number may be a country code. Wen to use that code, how to recognise that it is present? Does an integer maintain the explicit + sign?)

One reason for this mistake may be ignoring the big difference between how people tend to think of floating point numbers and how the computer (the ALU) is handling them.

For floating point numbers (normal or long): a reasonable good accuracy for a number N is only obtained for the region: $-1 < N < +1$. Moving away from that region, accuracy is falling off rather rapidly, even when computers tend to use normalized numbers.

Representing a value requiring two digit accuracy in a REAL type, the value must be checked to be within the limits: $-1023 < N < 1023$

Also remember the difference between 'mathematical arithmetic' and 'computer arithmetic', e.g. in 'mathematical arithmetic': $(\sqrt{2})^2 = 2$, while in 'computer arithmetic': $(\sqrt{2})^2 < 2$.

In short: REALs are lousy for accuracy: they lose it without warning, and not only in decimals.

This point is mainly of concern for Accuracy and for Reliability. If those two points are of little or no interest, forget about this point.

As examples: one should scarcely be using *real* value type for specifying amounts (length, weights, money) whenever some accuracy is required, One should also consider the use of ratio (*numerator / denominator)* more often when fractions are involved (e.g. 10 / 3 as two numbers instead of one inaccurate one: 3.333333…).

Collect and take care of your mistakes, do not be ashamed of them. They are your experience, part of your assets.

An Example: Size, complexity: from overall view to details.

As promised in **The sixth model example** (page 7) elements of a more detailed and (hopefully more coherent) example will be given here. First a short presentation of the history of why and how the need for the system evolved. Then something on how and when the various requirement came along. In between some examples of products and requirements posed by the products.

This story behind this example is one reason why maintainability in previously chapters is so strongly stressed. All the application areas did not appear at once; on the contrary: they grew little by little out of a very good cooperation between users and developers.

Autokon was developed as a CAD-CAM system, starting at the CAM end. (The abbreviations stands for: CAD: Computer Aided Design, CAM: Computer Aided Manufacturing.)

This gradual appearance of new application parts has not been unique for this system. Most "living" systems are in constant changing, e.g. Internet systems, computer Operating Systems (OS), the car toll road ring in Oslo (Bomringen).

It all started in the early days of Numeric Control (NC) for production machines/tools, e.g. flame cutters, lathes, larger drawing machines (say 2 by 3m size), used in the heavy industry, e.g. shipyards. The company SI (Senter for Industriell forskning) developed an at the time rather unique path-generating control unit in the late 1950s (operational at the Stord Yard in 1958) the ESSI control unit, controlling a German built flame-cutter. .

Flame cutters (later replaced by laser cutters) are cutting the thousands of plate parts from (e.g. 6 by 16m or 8 by 20m) steel sheets. (A 60 000 tons tanker has approx. 60 000 such parts). NC machines gave smoother cuts and much higher precision, thus cutting assembly work/time by a significant factor. But NC machines had to be fed with a precise description of what to do/where to go (path description).

Instead of having to calculating points and various types of elements very accurately and feed the, required element specifications to the controller, the shipbuilder were given a language, based on their own terminology in which to ask a computer to generate the path specifications. In some "long-hand" form, it could look like this: "Start a contour, start-point (xx, yy), follow a straight line in direction (dd) until intersection (,) with a circle, having - - - ". Normally they preferred to write it in a much shorter form: SCON:SPT(xx, yy)SL:DIR(dd) INT(,) CIR: - - - ".

They were given a vocabulary, by which to describe the path. Some of the other words are: "ending as tangent to:" (ETG), "begin as a tangent" (STG), "Common tangent" (between circles and/or to/from contours) (CTG), "round-off circle" (RND) (between two elements), the rule being: "If you can uniquely sketch it, the compute can generate it".

Similar descriptions may be stored under a name, constituting a "Norm" or a rule, e.g. a "cut-out", (which normally starts and ends with "INT" or "RND").

A norm describing how to generate one or a set of complete web frames or similar parts, gives rise to the name ALKON (ALgoritmisk Konstruksjon) for one of the modules in the system

In order for ALKON to work properly the hull shape had to be defined. The module FAIR was made to do the complete hull fairing job. Then the positions for all the longitudinal members were found, together with the type of profile and their angle to the hull side. (Module LANSKI)

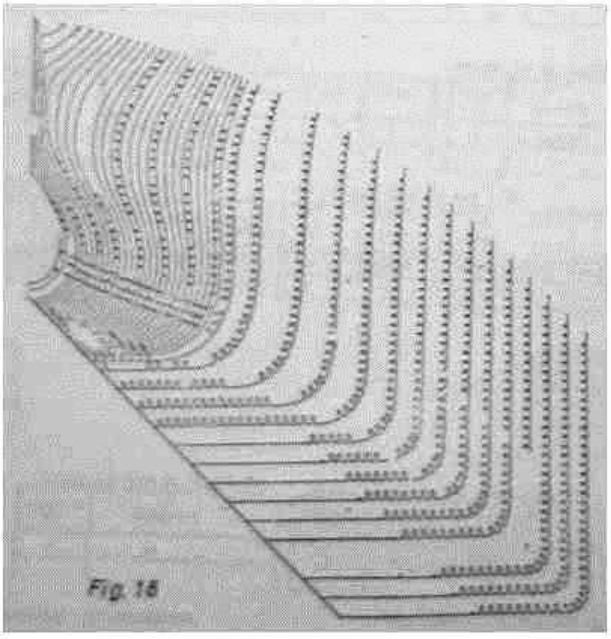


Figure: fig. 10: a faired hull and some initial positions of longitudinal members.

The result of all this is shown in *fig. 10*, being a check-out drawing in the middle of the process.

One of the presentations of the system is given to the right, fig 11. Quite often several representation of conceptual models may be used with advantage: the purpose of models is to sort out and to document "my own ideas and concepts of a problem area, as well as those of other people".

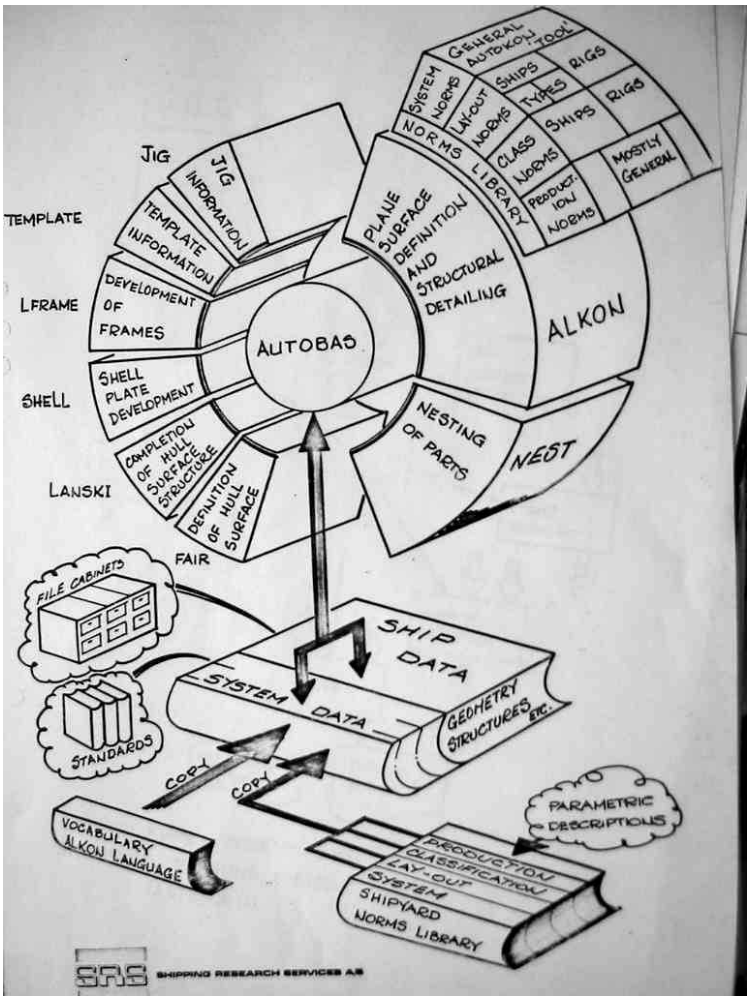


Figure: fig. 11

The goal for the system was to aid the yard in the design and construction of a ship: "to build the ship first in the computer, in the storage module AUTOBAS, then harvest the specifications for the various production processes/activities/jobs, the whole process being shown earlier in the *eighth model example.*

The discussions with the ship owner and his crew is not included here, but the module used there is seen in the picture below, as the module 'PRELIKON' (top left).

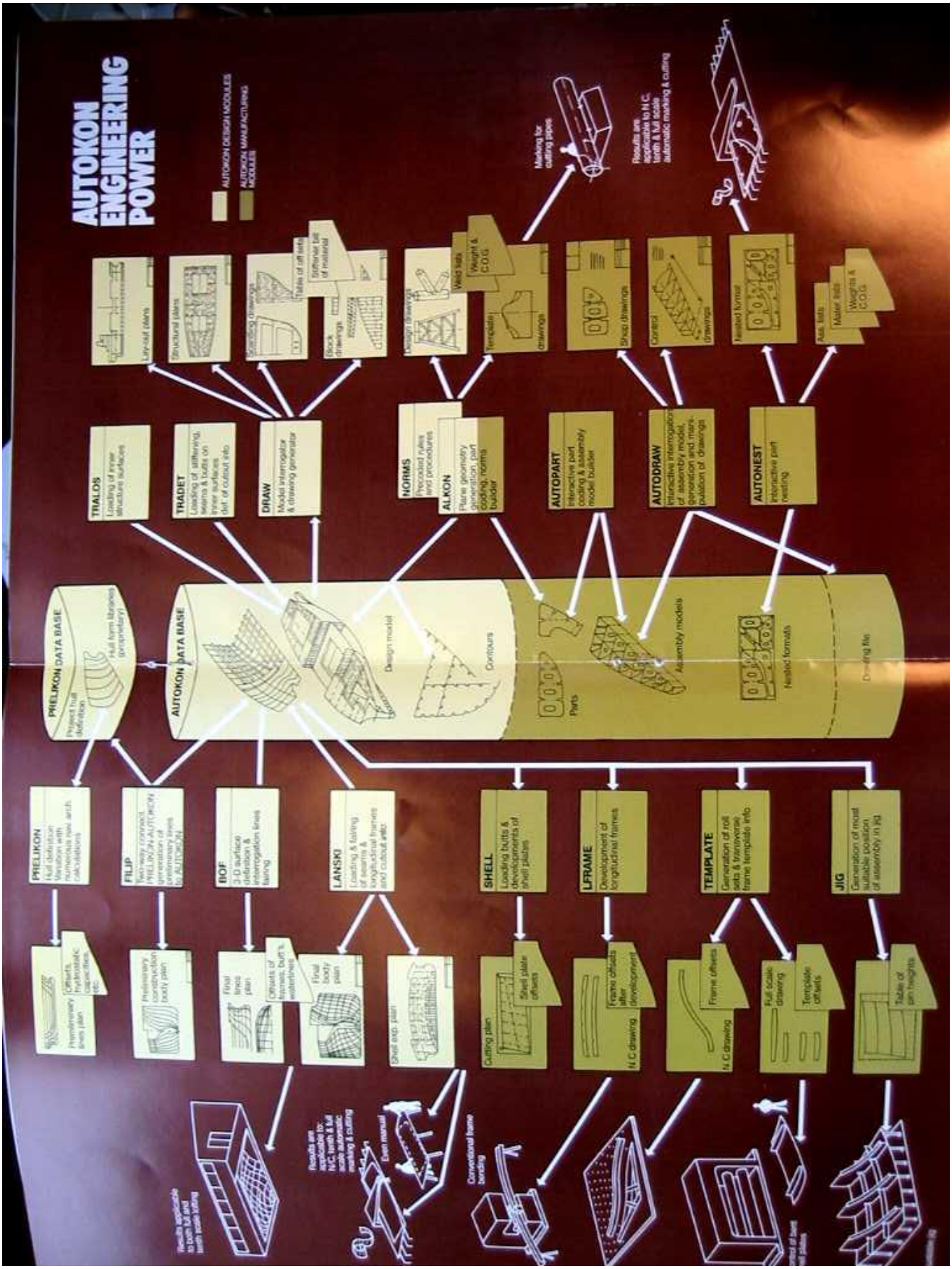
The "production data" from that process is mainly length, width, height, capacity/tonnage, together with the shape of stem, stern and the funnel. (Appearance may often be important).

All the data/specifications and production data are collected and stored in the AUTOBASE storage module. This was a rather critical module in several ways, and proved rather demanding in the modelling phase.

Using a common Data Base? Yes and no. The FAIR module is working in 3-dimentional space. So are LANSKI and SHELL, but they are using different coordinate systems. ALKON and NEST are both working in some 2-dimentional coordinate systems, but are both well aware of the 3-dimentional nature of the process and the product. They are all working, using their own data formats and structures. Yet it is essentially the same data they are all working on. AUTOBAS is the great format transformer, keeping track of all the dependencies.

So: which one of the figures 11, 12, 14, and 15 are system models? They all are, but made for different purposes. They represent the system as seen from different sides, different viewpoints, different purposes. They all clarify different aspects of the system, some at different levels, e.g.11 and 15.

Multi-level models are rather quite common, say in the range 2 to 5 or 6 levels. It is rather important to be able to see each separate module in detail, while one easily see the interaction between components, both those at the same level as well as those at different levels. Cross-level interaction tends to shout out: "Something is logically wrong here, this is messy and unhealthy". It is quite correct when a module is interacting with it own direct submodules. But it is rather questionable if one module is interacting directly with submodules of another module.



This picture: *fig 12*, shows an overall view of the AUTOKON system, on which modules does design work, (in light colours), and which ones are involved with production data generation and harvesting (in darker colours).

On the sides (top and bottom) are indicated various production equipment being controlled.

One job was to maintain the model for the entire system: the system was changing continuously throughout its life-time.

At the same time, models for all the other, separate modules kept on changing, and it all had to fit together.

The hardest three to model was the ALKON, the NEST and the AUTOBAS. This was mainly due to changes in production tools/equipment, changes in computer platforms, and to changes in ship requirements.

As for AUTOBAS, no standard database system vas ever applicable. The storage formats were not known outside AUTOBAS; sometimes some module needed to have new data elements added for storage, while these were irrelevant to other modules. Thus AUTOBAS must be able to deliver data from the same item stored in different selections/formats.

In addition AUTOBAS had to work as overflow-area for arrays in some programs and as stack storage: a mix of LIFO and FIFO stack behaviour: any stack must be able to handle LIFO and FIFO calls in any mix.

As to ARRAY-overflow one rule must apply: any array (or stack for that matter) must always have room for one more entry, never become full. Whenever the length is influenced by the external user, some overflow mechanism must be present. If an array in normal operation will contain up to 30 entries, a dimension of 60 or 80 should do, but the array still needs an 'overflow mechanism'. It must at least be capable to hold say 50 000 entries. (We have experienced that!)

In addition, AUTOKON was also used in other areas: for floating oil drilling platforms e.g. Aker H-3 and H-4 platforms, (se picture fig. 14), and they did have some deviating requirements. So did the aviation industry (See picture of fairing of BAC-111, fig. 13)

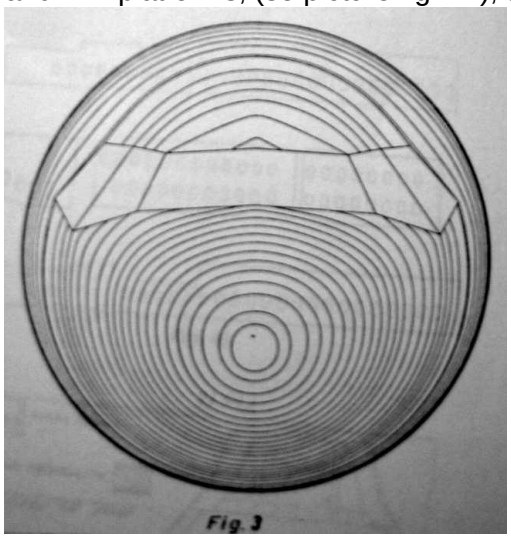
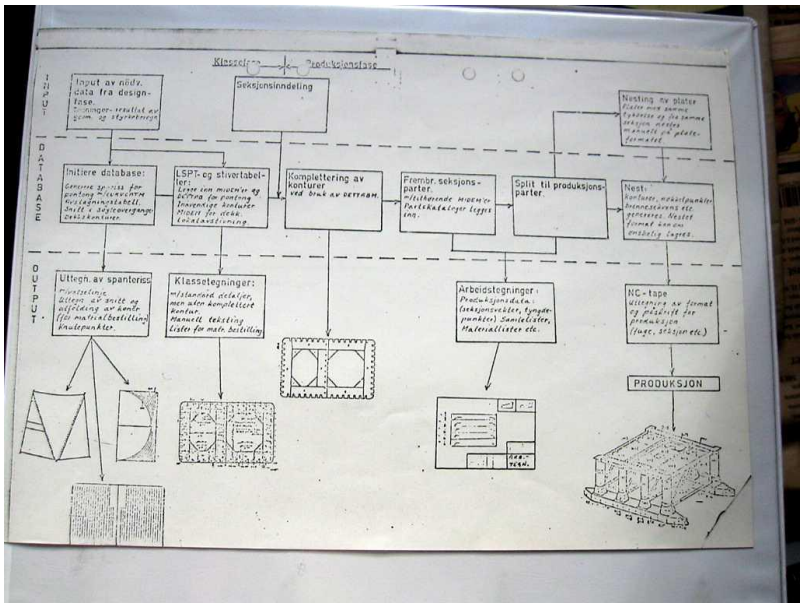


Figure: fig. 13

Figure: fig. 14



The set of requirements in the discussion above is very far from complete. One important (and in many modelling cases forgotten) requirement is *accuracy*.

Ships can be rather big, size normally given in some version of 'register tons', which is a volume unit. (One register ton being 100 British cubic feet, or approximately 2.8317 m^3.). If a tanker of some 500 000 tons was placed in Oslo, with the rudder stem at the main entrance to the Parliament (Stortinget), the stern would pass the National Theatre and the University, passing well into the subway station entrance. E.g. the *Batillus*-class supertankers, built by Chantiers de l'Atlantique shipyards at Saint Nazaire in the mid/late-1970s were more than 414m long, and with a beam of 63m.

(Chantiers de l'Atlantique had been using Autokon heavily in their production since early in the 1970s, as did Electric Boat division of General Dynamics in Groton, Connecticut, a major supplier of US nuclear subs.)

All measurements in the larger ships had to be correct within <u>one</u> mm all over, i.e. 1 mm in 500m, or an accuracy of 1 in 500 000.

So far we have been looking at some requirements and some top level representations of the system, using some 'module block representations'. But each of these modules (with assigned responsibility) constitutes again a (separate) system, consisting of modules, which are assigned their individual responsibilities. This type of subdividing was used recursively, down to the level where flow-charting took over. ("Divide et Empere").

It is extremely difficult to have a good, consistent view over the entire problem area and at the same time see all the minor details. A strict and consistent methodical approach is required, based on stringent "Divide et Empere" principle. (In the early phases, it tends to be a "Divide et Concere").

An example of a next step in this multi-level modelling, is given in *fig. 15*, the upper level of the ALKON module.

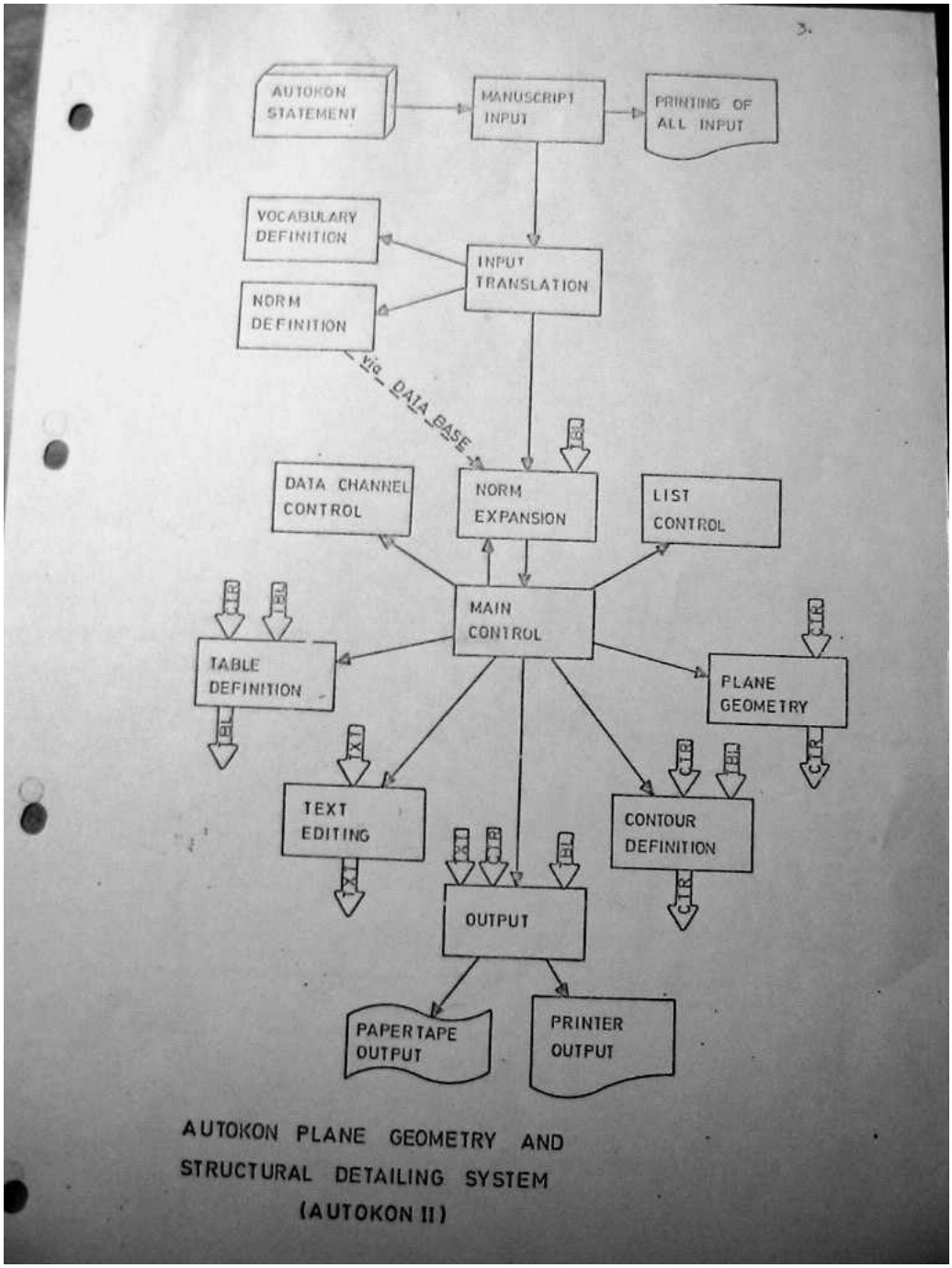
(a photo of a page in the 12 volume documentation.)

For modelling, from the top to the bottom, a *<u>methodical</u>*, *<u>thorough</u>* and *<u>easily readable documentation</u>* is a *<u>must</u>*

The thin arrows defines activation, while the fat arrows with text inside indicates data storage activity.

Figure: fig. 15

This version of the documentation uses the 'card input symbol' for 'file input'.



Interactive operation is not very useful in spite of what many people feel: Reliability and accuracy requires all actions to be well prepared, with full consciousness as to what to do. The various routine jobs are stored as norms in N levels. But of course you may interactively give orders like "start generation of the full set of frames and parts from frame n1 to n2".

Manuscripts/curve descriptions should be retraceable, meaning they must be saved in a file for possibly later corrections.

Many of the models in fig. 15 does in turn contain a full set of modules, each of which has to be modelled / described. This means that modelling must be done at many levels, each of the 'composite modules' being a complex system in itself.

As an example: the ALKON MAIN CONTROL contained two different virtual processers (virtual CPUs), each one running its own special program (residing in a table). Whenever a somewhat different behaviour was needed, the 'program' in the table was (temporarily) replaced.

Also the CONTOUR DEFINITION module contained a rather complex virtual processor.

Below is a brief and compact reminder of important points included.