**INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07**
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

# COMMUNICATING KNOWLEDGE: MAKING EMBEDDED CONFIGURATION WORK

**Gudmundur Oddsson[1] and Lars Hvam[1]**
[1]Department of Manufacturing Engineering and Management, Technical University of Denmark

## ABSTRACT
A lot of systems are assembled from near-independent mechatronic subsystems that have to be configured to match each other. An example of such systems are e.g. home entertainment systems, where TV, DVD and Receiver are matched to form an overall system, and compilation of pumps and controllers to form fresh water supply systems. Sometimes an external knowledge system keeps track of how each subsystem has to be configured, but the actual configuration is often done manually. Installing and maintaining those kinds of systems can be a tedious task and often requires repetitive labour. The idea is to "split-up" the product knowledge and encapsulate it into each subsystem. Then, when the subsystems are assembled, the configuration of each subsystem can either be done automatically or with minimum input. The concept could be called: *embedded configuration*.

This article will try to connect three aspects of making distributed knowledge system, the encapsulation of product knowledge, its subsequent encoding into product models, and finally, the communication of knowledge between the subsystems. There are two main reasons for focusing on communication, namely the encapsulation of knowledge and the communication between machines. One has to identify the information needed from outside for each subsystem to work. That information should also aid in finding the "services" that the subsystem can offer the overall system. Communication between subsystems has to be made explicit. A protocol has to be in place to tell the subsystem how to share its inner workings and how to be able to participate in the overall system.

*Keywords: Product Models, Knowledge Engineering, Configuration*

## 1 INTRODUCTION
"*Systems everywhere*" as so eloquently stated by Bertalanffy [1] is an excellent way to describe our modern society. It surrounds us with systems to help us in and to make better our daily lives. Many of those systems are "hidden" from us, and we only notice them when they fail. Think about the facilities that one uses daily, like electricity, water, sewers and heating. Those are complex product systems that are made by combining several subsystems (or independent products) to form a whole. Many of the subsystems are equipped with computers. By setting parameters those subsystems can be allowed a wide range of different setups, just like they were different products. To make the subsystem work in a context, these parameters need to be set. An example of an installation complexity could be the water supply in a given building of reasonable size which would require about four to six pumps, each with 500 parameters being controlled by a controller (in the sense of a controlling unit, not a person) that has 3000 parameters to be set. One can easily see the tediousness of installing such a system.

In order to make complex product system setups easier an idea is brewing. If one were to encapsulate as much product knowledge in each subsystem as needed, have internal configuration engines keep track of internal consistency and focus communication between subsystems so that only core information or knowledge is transferred, setups would be simplified. The idea is to make the systems kind of aware and give them knowledge to "self-configure", once the hardware connections are made. This idea has a lot in common with the field of artificial intelligence, specially the branch of distributed artificial intelligence and has inspiration been drawn from that field on how to go about solving it. This article will try to connect three aspects of making distributed knowledge systems,

namely the encapsulation of product knowledge, its subsequent encoding into product models, and finally, the communication of knowledge between subsystems. To achieve this, one has to look at the building blocks needed for such construction namely: knowledge, communication, how to communicate knowledge and finally, how to model in such a way that it will support the final implementation. The solution suggested for making distributed knowledge systems work, is called embedded configuration, and there will be a discussion on this principle later in the article. This article will end with a rationale for why the communication issue should be the focus of such an approach. Let us now go through the building blocks one by one, connect them into a whole and then tie them to the concept of embedded configuration.

## 2 KNOWLEDGE AND ITS PART IN COMPLEX SYSTEMS SETUP

Before moving into embedded configuration and communication, let us talk a little about knowledge. How is it viewed and defined, or what is maybe more relevant, how it fits together with the task of creating complex systems? Every undertaking humans do is built on our knowledge of things and surroundings. We are often not very aware of our knowledge and how it is structured, we just use it. If one is to construct an intelligent system, one has to know what knowledge is, and especially how to transfer or communicate it to the machines. Let us start with knowledge and its relationship with data and information.

**Attempting a knowledge definition**

Knowledge has been a research topic for some time now. What is interesting is that authors have different aspects to their understanding of knowledge and its dependency on data and information. Note that these views are not contradictory, they just represent different viewpoints and seem dependant on the domain from which the author focus his / her research. A quick, and by no means a complete, look at the dependency between data, information and knowledge is shown in Table 1.

*Table 1 – Different authors on data, information and knowledge*

| Author | Data | Information | Knowledge |
|---|---|---|---|
| Moore | Digital object<br>Objects are streams of bits | Any tagged data, which is treated as an attribute<br>Attributes may be tagged data within the digital object, or tagged data that is associated with the digital object | Relationships between attributes<br>Relationships can be procedural / temporal, structural/spatial, logical/semantic, functional |
| Wiig | | Facts organised to describe a situation or condition | Truths and beliefs, perspectives and concepts, judgements and expectations, methodologies and know-how |
| Nonaka and Takeuchi | | A flow of meaningful messages | Commitments and beliefs created from these messages |
| Spek and Spijkervet | Not yet interpreted symbols | Data with meaning | The ability to assign meaning |
| Davenport | Simple observations | Data with relevance and purpose | Valuable information from the human mind |
| Davenport and Prusak | A set of discrete facts | A message meant to change the receiver's perception | Experiences, values, insights, and contextual information |
| Quigley and Debons | Text that does not answer questions to a particular problem | Text that answers the questions who, when, what, or where | Text that answers the questions why and how |
| Choo et al. | Facts and messages | Data vested with meaning | Justified, true beliefs |
| Jensen Group | Representation of facts | Data plus Meaning<br>Understanding of patterns, relationships | Information plus Beliefs, Commitments, Assumptions, Design for application |

Constructed from Stenmark [2], R. Moore lectures at Rice University, Houston, Texas and Jensen Group [3] (who compiled from Fahey, Nonaka, Wurman and Gange)

Some interesting aspects are notable in Table 1. Although most agree on data as facts or symbols and information as data with meaning, the authors reach different abstraction levels in their knowledge definitions. The IT view offered by Moore has knowledge with "lower" abstraction than e.g. Choo's "justified true beliefs". The views that would make most sense in this article are the ones presented by

the authors Quigley and Debons fused with Nonaka and Takeuchi. The views stated in Table 1 are quite well summarized in Mueller & Schappert [4] as a set of abstraction levels and general description to each level. It points to linkage as the most important aspect, not the definition of each level.

*Table 2 - General view on data, information and knowledge [4]*

| In the classical interpretation | That is: |
|---|---|
| *Data* is associated with syntax | *Data* per se has no meaning and may be seen as raw material for information |
| *Information* corresponds to semantic | *Information* is context sensitive and meaningful in the sense that it is interpreted data |
| *Knowledge* takes the pragmatic part | Since context is user (application) dependant, information then may be enhanced by its use, i.e. the pragmatic *knowledge* |

This is to say that how one moves between the levels is more relevant than giving a precise definition of those same levels. A definition of each level is shown in Table 2. Taking this view on the linking and picturing would result in Figure 1.

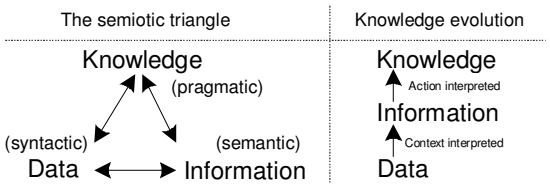



*Figure 1 - Relating data, information and knowledge [4]*

So, where is this leading? No matter what views on data, information and knowledge is taken of those presented in Table 1, by focusing on how to move between levels and identifying what is essential, it should be possible to use that to make more intelligent systems. The syntactic and semantic is linked with context mapping while the semantic and pragmatic has action interpretation link. That can be even more useful if we call it context and interpretation as pictured in Figure 2.

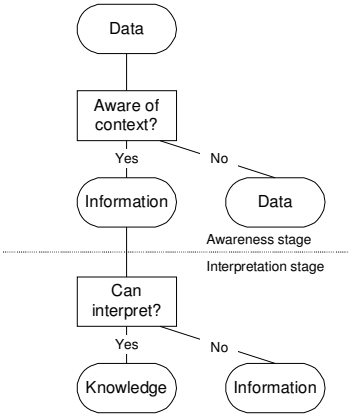



*Figure 2 - Moving between data, information and knowledge [5]*

The process of moving from data to knowledge[5], as indicated in Figure 2, should hold true for all viewpoints stated in Table 1. But, for the purpose of structuring data, information and knowledge in regard to embedded configuration, we will focus on context and interpretation and use the definition in Table 2 as our guide. This actually coincides with the point of view offered by Schreiber et al [6],who state that maybe a precise definition of knowledge is not needed to be able to manage knowledge and its communication.

This focus should also imply how communication can be handled. Let us move on to communication and its role in knowledge sharing.

## 3 COMMUNICATION

Communicating knowledge can draw inspiration from many different fields. Human beings do it all the time, and much research on this topic has been carried out. Computer science has tried to emulate humans in artificial intelligence, especially distributed AI and the design of complex control systems has focused on communication for quite some time. When we communicate in our daily lives, we usually do not differentiate between data and information on the producer side or information and knowledge on the consumer side. Matthias Rauterberg [7] shows this well in his producer to consumer view of communications as seen in Figure 3. So, what gets communicated is information or

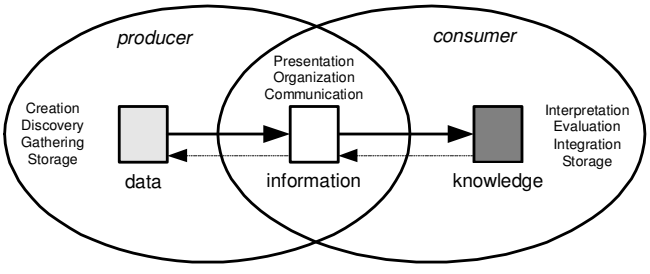



*Figure 3 – Producer / consumer communication [7]*

context-rich data, and it is then subsequently the consumer who interprets the communiqué to knowledge.

This producer / consumer terminology could be more easily understood by using communication theory [8], which states producer as sender and consumer as receivers (Figure 5). This is one of two other ways to look at communication, the other being communication layers (Figure 4). Even though these look different they are of similar rationale.

The lowest layer in Figure 4, the physical carrier, is the same as the medium part of Figure 5. The protocol will decide how coding and decoding is done and subsequently how the signal will be like.

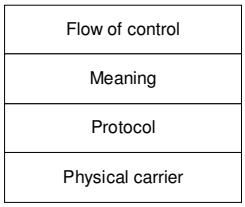



*Figure 4 - The communication layers [9]*

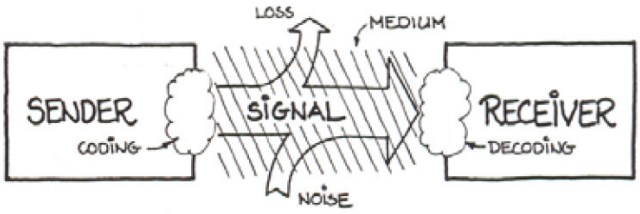



*Figure 5 - Communication theory [10] but based on Shannon & Weaver [8]*

The two higher layers, Meaning and Flow of control, are not explicitly drawn in Figure 5, but the first mentioned, Meaning, would implicitly be what the sender wanted to say to the receiver. Flow of control is not present in the communication theory, but one could think of ways to add this to the picture.

In anthropology yet another way to look at communication exists, the high and low context communication put forth by Hofstede [11]. This focuses on the awareness stage (in Figure 2) and on how much knowledge / context one has to add to the actual "signal" to get the "right" meaning from the communication. Think about the following: ask anyone in most western cultures for directions to a place that does not exist, and everybody (well, most likely everybody, apart from some jokers) will tell you that the place is nowhere to be found. Do the same in some middle-east countries where it is considered rude not to help, and some people might try to guide you to somewhere.

To visualize this difference, one could draw a "context" axis and place the three levels there. Figure 6 and Figure 7 show high and low context communication by placing the three levels on an imaginary "context" axis where longer distances depict more context knowledge. In those figures the x-axis has no special meaning.

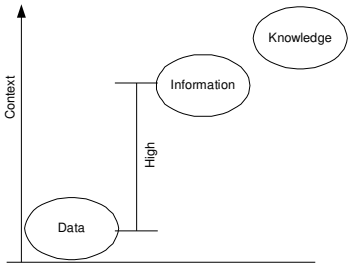



*Figure 6 - High context communication*

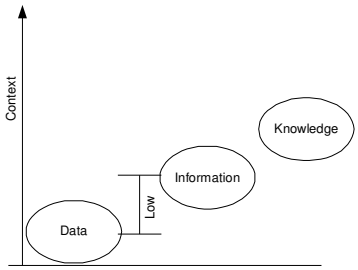



*Figure 7 - Low context communication*

The context of communication has another impact; it places a role on the sense-making process. Sense-making in communication has been researched in, among others, the sense-making theory [12]. Its core is "asking the right questions" when communicating. Think about the following: you go to the library and ask for Shannons & Weavers article on communication. The librarian gives you what you ask for, but you later realize that it was not exactly what you where looking for. If, in the beginning, you could explain to the librarian why you were there, like: I am working on how to communicate knowledge and need literature related to communication, the librarian would have had a chance to "interpret" or put into "context" what you wanted and find something relevant. In relation to what has been said earlier, the first is communicating data and the latter is communicating information. When communicating information the receiver gets the possibility to "interpret" or "contextualize" it to multiple sets of data. This could be drawn onto the data, information and

knowledge mapping figures with an arrow running down from knowledge to data. The lower the context barrier (like the bar in Figure 6 and Figure 7), the "easier" the mapping between data and information / knowledge ought to be. Like this paragraph on communication hints, one would most likely not have knowledge in the communiqué, only information, and it would then be up to the receiver to make the interpretation.

# 4 COMMUNICATING KNOWLEDGE

We constantly try to communicate knowledge in our surroundings. This might be best illustrated with an example. The other day I helped my mother move photographs from a digital camera to a PC. This should be a straightforward task. The problem (well, one of them) was that we live in different countries, so I guided her through the telephone. This took a lot longer than expected and required many iterations and explanations. After we had finished the task, I came to think of how this actually related to my work and the communicating knowledge "dilemma". It made me ponder on the following: Let us assume that you are a computer literate person with some IT knowledge and you were to communicate your knowledge to an elder semi-computer literate person who is to perform a simple (from your point of view) task. Would you prefer to do this through the telephone in your own language, meaning that you could only use words and no observations nor manual guidance, or on site in an unknown language and having to rely on mime, observation and manual guidance?

An interpretation of this scenario could be that in the first case, one would try to decode one's knowledge into information and then ask the other person to do carry out a task / an action that he / she does not understand as he / she does not have the knowledge to do neither the interpretation nor the putting into context. It would be a lot like playing blind-chess and having a semi-independent "machine" to move your pieces, because people do not always do precisely what one says or your prescriptions are not precise enough. One would then constantly have to check if the person had done what you asked him / her to do. In the other case, one would not have to decode one's knowledge or to explain both interpretations and context. A simple "follow my lead" would suffice, even though one could not speak a word in the other person's language.

This talk on knowledge and communication is all good and well. Let us now put it into context with the problem at hand, namely the use of embedded configuration to increase usability.

# 5 EMBEDDED CONFIGURATION AS AN AID TO REDUCE COMPLEXITY

A lot of systems are assembled from near-independent mechatronic subsystems that have to be configured to match each other. An external knowledge system sometimes keeps track of how each subsystem has to be configured, but the actual configuration is often done manually. Installing and maintaining those kinds of systems can be a tedious task and often requires repetitive labour. The general idea is to "split-up" the product knowledge and then encapsulate it into each subsystem. Then, when the subsystems are assembled, the configuration of each subsystem can either be done automatically or with minimum input. The concept could be called: *embedded configuration*. To better explain embedded configuration let us use system theory and very simple graphics to go through the underlying rationale.

## Explaining Embedded configuration

The concept of embedded configuration and its benefits can be explained by the system theory, where elements, relations and boundaries are presented in a graphical way like for example in Skyttner [13], but built on von Bertalanffy's ideas [1]. Let us think of a system (Figure 8), it has elements that have internal relations. Some of the elements require input from the environment (e.g. users) and therefore transcend the system boundary. Here one assumes that the elements are some kind of parameters that have to be set. Some are set with internal relations while other require input from outside the system. Note that the abstract figures of the system that follow do not state anything about what the system does; only that it needs inputs to be set to a working state.

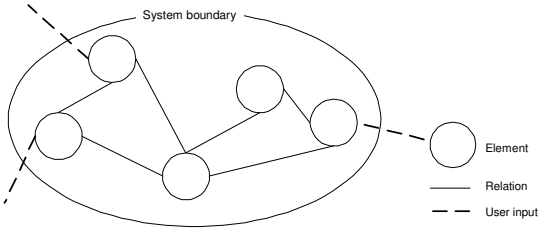



*Figure 8 - A simple system with elements and relations*

For explanatory purposes, let us assume that the system is a multipurpose system and the user inputs needed are parameters to select some of the intended behaviours. If many of such systems are combined to a larger system (Figure 9), an "installation" problem arises, where subsystems have to be "matched" or configured to the overall system-intended application. The following rationale only deals with the problem of installing such systems, but not the subsequent use or purpose (could also be called system assembly). It is about making a system that works, but not about what it is to do. Back to Figure 9 and how the various user inputs are related: Someone, like the user or an external information system, has to set all the user input parameters to make the system work. But the inputs are not independent; some of them can be related in one of two ways, they are to match the hardware together, i.e. one subsystem is connected to another and the parameters make each subsystem "aware" of each other. The other way is related to the use of the final system, the application. That is, when the overall application is decided, some of the inputs will connect together. Both of these relations can easily be expressed with rules, constraints or mappings in an integrated system. Things get trickier when the final system is constructed of unknown subsystems, i.e. when it is not predefined what subsystems are present in the final system. The enclosing of the solution space or the matching of hardware is the first thing that relates some user inputs to others. Graphically, this can be shown by connecting interrelated inputs, so when one input is set, some others will automatically also be set, as shown in Figure 10. The interrelating of these inputs is a parallel to constraining the solution space. As these are completely hardware-related we will name them *Hardware-induced configuration.*

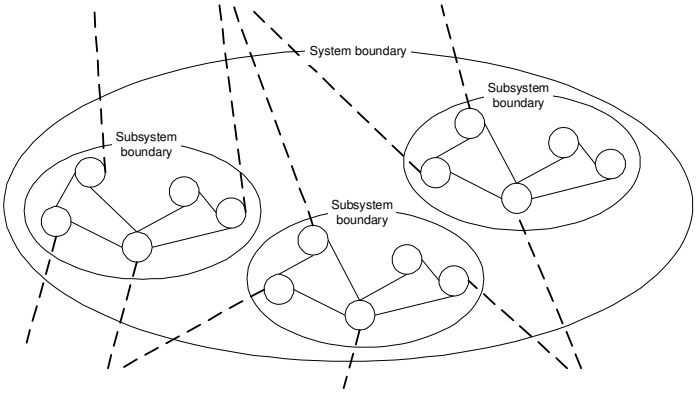



*Figure 9 - Subsystems combined to form a system*

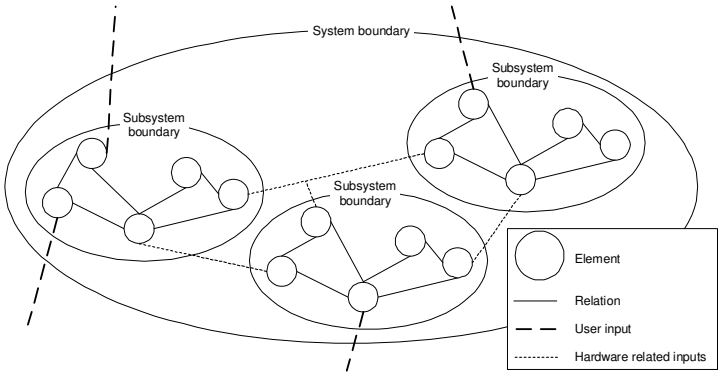



*Figure 10 - Some relation are hardware setup related*

The nature of these relations and that all needed information for connecting the subsystems lies within the overall system, makes one argue that this linking should be accomplished completely automatically. The other kind of interrelation between inputs is the application related inputs. These can be set when the overall usage of the system is determined. Once the application is selected, some inputs will relate to each other and by answering or setting one input, several others may be determined. Graphically, this can be shown just like the hardware setup, and this is done in Figure 11.

Application relations rely on user inputs to be set so they are most likely not automated, but the relationships should be identified before the parameters are set. This relates to the selection of a specific

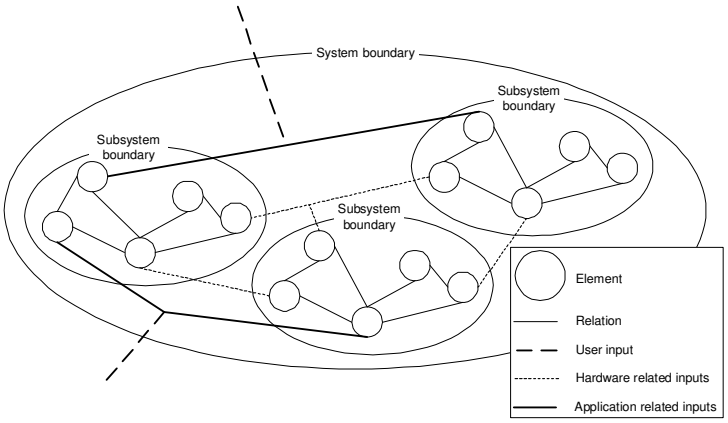



*Figure 11 - Some relations are application related*

solution and could be named *Application-induced configuration.* What is maybe not stated explicitly here is that inside each subsystem part of configuration can take place and that it is completely "contained", i.e. each subsystem will make sure that its internal workings are in order and no illegal settings are present.

An example will illustrate this. Think about your home and the television and DVD player that most of us have. When installing the DVD player, one has to go through a menu to tell the player what kind of TV is present, i.e. normal or widescreen. This is in essence, hardware-induced configuration. Once a DVD disk is placed in the player, one has to select an "application", what one wants to do, for example, select subtitles or a language. This is the selection of a specific solution or application-induced configuration. This example could of course be made much more complex, but it should highlight the two important aspects of embedded configuration, that is the hardware- and application-induced configurations.

Combining subsystems to form a system, where the subsystems have to be configured to work in the overall system, is a tedious task. As seen in Figure 9 to Figure 11, it is possible to reduce the user inputs by relating inputs to each other. This encapsulation of both hardware awareness and application hopefully require fewer inputs by the user. By connecting complexity to the number of activities one could argue that complexity is reduced in such a concept and hence the usability for the user is increased. This, of course, remains to be seen (and proved), but the working hypothesis is that this holds true.

It is easy to put forth such concepts, but to show how to make them viable and construct them is another matter. When constructing such a system, several issues have to be clear. One would want the setup to be tolerant towards new subsystem versions and be able to demonstrate redundancy by reconfiguring if some subsystems fail or partially stop functioning. This requires that the product knowledge of each subsystem has to be stored internally in each subsystem and they then have to be able to communicate meaningfully to cope with the overall system. The reasoning in this chapter also points towards why communication between modules should be the focal point of the concept. To be able to inter-relate hardware and application settings between subsystems, one has to know what each subsystem requires in order to function, and how one could ensure that such knowledge sharing deals with the issues mentioned earlier.

### Industry example of where Embedded configuration is needed

Appling the aforementioned rationale to the example mentioned in the introduction of the article should aid to clarify the suggested method. When connecting a pump to a controller, some parameters have to be set to "tell" each device what it is to be connected to. The pump has to know that it is being controlled externally, and it has to "relinquish" its own control. The controller has to know the attributes of the pump, so that it can control the overall system accordingly. These things are all hardware-related in the sense that they help define the solution space, i.e. what applications are possible for the whole system. If the subsystems could communicate meaningfully, these tasks would "disappear" from the installation. When selecting a specified application, some of the devices have to be told what is being done. As a part of the application, redundancy has to be set. What is the pump to do if the connection to the controller is lost and vice versa, how shall the controller handle lost connection to the pumps? This research is based on a case study at a major pump producing company, and while empirical work is still underway, preliminary results suggest that roughly one third of all the parameters (in all devices) have to do with hardware-induced configuration, half with application-induced configuration or selection of a solution and subsequent operations, and finally the rest should probably not be parameters at all as they are never changed.

The concept of embedded configuration has now been introduced, and we have identified that both knowledge and communication have to be modelled to make this work. Let us next look at ways to achieve this.

## 6 MODELLING KNOWLEDGE TO SUPPORT COMMUNICATION

Modelling is done to support all kinds of activities. Since models are abstraction of the world their use is helpful in decomposing problems and structuring knowledge. Designers of physical artefacts have for some time used models to aid in the design process. Artefacts are made by humans to serve some purpose. Herbert Simon put forth the boundaries for science of the artificial in his like-named book [15], where he states on page 5 that: "Artificial things can be characterized in terms of functions, goals and adaptation". These artefacts are often discussed when they are being designed, and the designer has to know the intended purpose and then synthesize a solution. Many have tried to aid the designer by making the process more explicit and develop tools to that end. When designing complex systems, like an industrial plant [14], another axis besides the one identified by Simon becomes apparent, the whole-part look on the system, where for each level one has to look at the means-ends axis. This is shown in Figure 12.

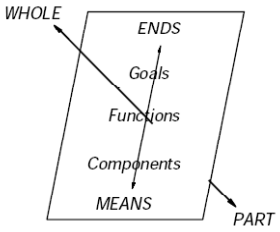



*Figure 12 – Designing industrial plants [14]*

The purpose or goal of each level is realized with structure or components that behave in a functional way towards a goal. This requires functional thinking or reasoning in design and many researchers agree on that. Like functional and physical dimensions [16], the mapping between functions, design and processes [17], functional models [18], technical system theory [19], capturing functional knowledge [20], the Function-Behaviour-State modelling [21] and last but by no account least, the functional structuring of Pahl and Beitz [22]. All these researchers have their way of trying to capture the purpose of the artefact being designed and their view should be helpful when we encode the purpose into the product model. Once the purpose is known, mapping between the means-ends levels is required. There are many ways to do this. The product family master plan or PFMP (sometimes called product variant master or PVM) technique has been developed to illustrate variance in product families. It has evolved from the Andreasen's Chromosome model [23], and the PFMP latest variation [24] has three views, the customer view, engineering view and the part view. The first relates to the goal from Figure 12, the second to functions and the third to components. The PVM is also used in the procedure for making configuration systems [25]. Another method is the GTST (Goal-Tree-Success-Tree) method suggested by Modarres [26], where the authors map structure and functions together. Within the field of product configuration Mittal & Freyman [27] have suggested that knowledge structuring should be done on physical structure, functions and the mapping there between. Forza & Salvador [28] add the layer of performance on top of functions and components, which could also be called goals or applications. In short, most agree that different abstraction levels are needed to fully describe a product and its purpose. This is like "encoding" the knowledge and rationale that the designers went through when designing the product [29] into the product model.

To map these connections visually may become quite a tedious task with many relations. Therefore, others researchers have suggested to use Design Structure Matrixes (DSM) [30] to help the designer to design with function-structure mappings [31]. Another problem constantly arises both in design and modelling; how does one accomplish functional decomposition? Some good guidelines on product decomposition can be found from different authors, the four relationship types [32], the DSM view [33] but even more important is the attempt to include the purpose of products in decomposition strategy [34], where the authors map requirements to functions. A sharp definition of the functional language is required (e.g. Lind's Multilevel flow models [9], Stone's Functional basis [35] or Kitamura & Mizoguchi's Functional ontology [36]) to structure the decomposition and allow reuse. Sørensen [37] nicely ties the purpose of artefact to the knowledge and communication needed in his control system design suggestion seen in Figure 13.

This could be connected to both knowledge definitions (Table 1) and to communication (think about the librarian and the search for articles). The main impact will be on communication. It is smitten by how, what and why of the design phase. This will influence how data, information or knowledge is structured, and it will then connect to the context and interpretation needed to make sense of a communiqué.

Models used in designing are not usually accessible in later phases of the artefacts life cycles. This is probably not a problem as the artefact is completely known and all its "functionality" and "goals" lie implicitly in its structure (as structure is the only "physical" thing and hence the only thing the designer can influence). The problem arises when the system is composed of several subsystems, which have to work together, but it is not known beforehand which subsystems will be present in the final system. To allow such a system to gain some intelligence, they have to be able to have a meaningful communication, i.e. not only exchange data, but be able to ask more "higher level questions" like, "what do you do", "how do you do it" and "why do you do it". This makes a prerequisite on the models and their content (see Table 3).

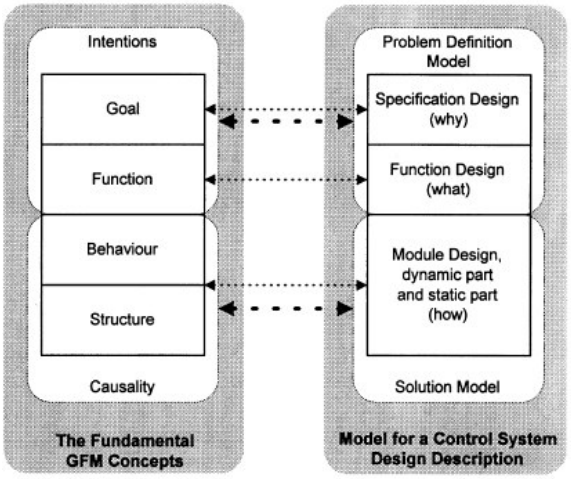
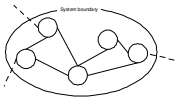
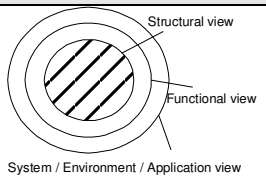
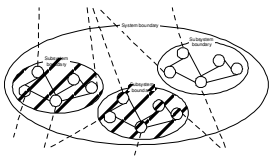
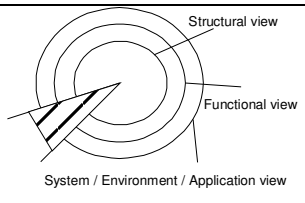
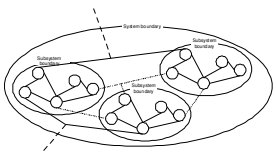
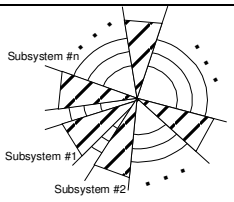



*Figure 13 - Designing a control system [37]*

A further explanation of the encoding needs when subsystems are combined requires consideration of the "participants" in the communication. If a person were to assemble the subsystem, this person could "decode" the physical structure into functions and hence obtain the goal of the assembled final system. The person would not need explicitly stated functions and goals to make sense of the final system. If one substitutes the person with a machine, the machine will be "dumb" and only do what one tells it to do. It will, therefore, not be able to deduct the final system function just by knowing its physical structure. It has no context to make functions from the structure, nor does it have the intelligence to interpret the functions to a goal. For the machine to make sense of the final system it would require explicitly stated functions and goals of each subsystem and how to deal with these to construct the final system functionality and purpose.

*Table 3 - What models include in different setups*

| | System | Include in model |
|---|---|---|
| When knowing the whole system as in integrated designs, modelling only needs to includes structure because that will implicitly include functions and goals |  |   |
| The problem arises when the whole system setup is not known. More "abstract" information are needed from the design, both functionality and purpose / goals, to allow later compilation into overall system functions and goals |  |   |
| Combining subsystems into system requires compilation of overall system functions and goals. This calls for communication on multiple levels, i.e. functions and applications (goals) to allow for both what's and why's |  |   |

To construct the models and their content, but to keep focus on making them as simple as possible, working backwards would probably be sensible. As the models are intended to be a structuring of relevant information / knowledge and to be integrated into each subsystem, it would be wise to look at what the subsystems need to know to function in the overall system setup. The communication should focus on what and why and not so much how. That should then help to decide what should be modelled and put into place. To achieve this, one would have to decrease the "context gap" between data and information and make interpretation towards knowledge easier. Of course, there are many

ways to achieve this. One is to encode information and make communication explicit, so maybe it could be based on ontology [38]. That would allow for structure, but ensure ways to expand and adapt the communication. So, it would be an agent thinking [39] in the modularization of knowledge needed, and inspiration should be drawn from the artificial intelligence field. Another view is offered from computer science, that is the knowledge level [40] thinking that wants to make the purpose of the system explicit and hence independent of the media in which it is realized.

The building blocks are now all in place, so we can summarize the arguments and present our case for why communication should be the main focus when modularizing product knowledge and making embedded configuration work.

## 7 WHY COMMUNICATION SHOULD BE THE MAIN FOCUS

It is time to make our case. The rationale for communication focus has been hinted along the way in earlier chapters but this chapter will present a summary of it. There are two main reasons for focusing on communication, namely the encapsulation problem and the communicating between machines problem. Let us start with a look at the encapsulation of knowledge. To make each subsystem aware of its role in the final system, knowledge on its physical structure, functions and goals is needed. These three aspects have to be tied together, so that the subsystem can know what physical structure gives what functions in order to serve specific goals. The major issue here is to identify what information is needed from the outside to make each subsystem work, and what "services" the subsystem can offer the overall system. By making this communication explicit on a higher level than the physical structure level, one could simplify the communication and make it more robust (version and upgrade tolerant). This coincides with Suh's axiom design [41] for simpler and cleaner designs. It will also help modelling the internal knowledge needed and to decide on what should be included. It does not serve any purpose to have more than what is needed encapsulated in each subsystem, it is though a problem to identify what is actually needed.

The other aspect is communication between machines, here between subsystems. As the subsystems are "dumb" and cannot decode physical structures to functions or goals, those have to be explicitly stated within each. A communication protocol has to be in place to tell the subsystem how to share their inner workings and how to be able to participate in the overall system. If these were not known, all other modelling and structuring would be for nought and would not mount anything. Knowledge is not for much if it cannot be communicated.

## 8 DISCUSSION

This research is driven by the need to simplify installations of complex product systems like water supply systems and the like. It assumes that there are several parameterized subsystems that have to be connected to form an overall system. As parameters imply software, this is indeed kind of software engineering related and could be seen as such. It is a sub-goal to suggest a method that is tightly coupled with software implementation and that does not require double work in modelling and programming. The thoughts presented here are also much related to distributed artificial intelligence. They can be viewed as a tools for constructing such systems and even a start for a method as has been sought after[42]. Another thing worth mentioning is that benefits from the suggested solution will only be fully reached at a system level, meaning that initial costs of knowledge engineering might be high, and it is only when looking at the complete life-cycle of the product system that the rationale makes sense.

## 9 CONCLUSION

In this article we have presented a way to simplify setup of complex product systems with the help of embedded configuration. To achieve this one has to focus on what subsystems need to communicate to structure the required internal knowledge and to form a communication protocol. The simplification of internal workings is due both to hardware- and application-induced configuration that would take place both within the overall system and in each subsystem. By relating parameters in such a way, user inputs should decrease drastically, and the overall usability of the installation increases. In our case we have rationalized that this should be done with embedded configuration and

the expected result is enhanced usability. The next step can be said to be two-folded. Firstly, to construct a system based on this philosophy and to show that it actually leads to the expected results. And secondly, to further develop the modelling tools and methods for supporting the making of embedded configuration systems or in essence, a distributed artificial intelligence system.

## REFERENCES


[1] L. V. Bertalanffy, *General system theory. Foundations, development, applications*, Revised repr.of 1968 ed. ed. New York: Braziller, 1973.

[2] D. Stenmark, "Information vs. Knowledge: The Role of intranets in Knowledge Management," Proceedings of the 35th Hawaii International Conference on System Science: IEEE, 2002.

[3] The Jensen Group, "Changing how we work: The search for a simpler way," The Jensen Group, Northern Illinois University College of Business,1997.

[4] Mueller and Schappert, *The Knowledge Factory - A Generic Knowledge Management Architecture* 1999.

[5] S. Ahmed, L. Blessing, and K. Wallace, "The Relationships between data, information and knowledge based on a preliminary study of engineering designers,", DETC/DTM-8754 ed Las Vegas, Nevada, USA: ASME, 1999.

[6] G. Schreiber, H. Akkermans, A. Anjewierden, R. d. Hoog, N. Shadbolt, W. van de Velde, and B. Wielinga, *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambrigde, Massachusetts, USA: The MIT Press, 2000, pp. 1-455.

[7] G. W. M. Rauterberg, "The new economy: e-commerce, intellectual property rights, trust and other dangerous things," slides from ICT congress in Netherlands: 2001.

[8] C. E. Shannon and W. Weaver, *Mathematical theory of communication*. USA: University of Illinois Press, 1949.

[9] M. Lind, "Representing goals and functions of complex systems," Institute of Automatic Control Systems, Technical University of Denmark, Copenhagen, Denmark,Technical Report 90-D-381, 1990.

[10] J. Buur and M. M. Andreasen, "Design models in mechatronic product development," *Design Studies*, vol. 10, no. 3, pp. 155-162, 1989.

[11] G. Hofstede, *Culture's Consequence: Comparing Values, Behaviours, Institutions and Organizations Across Nations*, 2 ed. USA: SAGE Publications Inc, 2001.

[12] B. Dervin, "Sense-making theory and practice: an overview of user interests in knowledge seeking and use," *Journal of Knowledge Management*, vol. 2, no. 2, pp. 36-46, 1998.

[13] L. Skyttner, *General systems theory*. Singapore: World Scientific Publishing Co. Pte. Ltd, 2001.

[14] M. Lind, "Modeling goals and functions of complex industrial plants," *Applied Artificial Intelligence*, vol. 8, no. 2, pp. 259-283, 1994.

[15] H. A. Simon, *The Science of the Artificial*, 3 ed. Cambridge, Massachusetts: The MIT press, 1996.

[16] K. Ulrich and S. D. Eppinger, *Product Design and Development, Third Edition*. Boston, Mass.: Irwin McGraw-Hill, 2004.

[17] Z. M. Bi and W. J. Zhang, "Modularity Technology in Manufacturing: Taxonomy and Issues," *International Journal of Advanced Manufacturing Technology*, vol. 18, no. 5, pp. 381-390, 2001.

[18] W. Y. Zhang, S. B. Tor, and G. A. Britton, "A functional modelling approach for modular product design," In *International Conference on Manufacturing Automation: Advanced Design and Manufacturing in Global Competition*, 2004, pp. 31-38.

[19] V. Hubka and W. E. Eder, "A scientific approach to engineering design," *Design Studies*, vol. 8, no. 3, pp. 123-137, 1987.

[20] Y. Iwasaki, R. Fikes, M. Vescovi, and B. Chandrasekaran, "How things are intended to work: capturing functional knowledge in device design," In *Int. Joint Conferences on Artificial Intelligence*, 1993, pp. 1516-1522.

[21] Y. Umeda and T. Tomiyama, "Functional reasoning in design," *IEEE Expert*, vol. 12, no. 2, pp. 42-48, 1997.

[22] G. Pahl and W. Beitz, *Engineering Design – a systematic approach* Springer-Verlag, 1996.

[23] M. M. Andreasen, "Syntesemetoder på systemgrundlag - Bidrag til en konstruktions teori [in Danish]." Department of Machine Design, Lund Institute of Technology, 1980.
[24] U. Harlou, "Developing product families based on architectures: Contribution to a theory of product families." PhD Thesis Technical University of Denmark (DTU), 2006.
[25] L. Hvam, "A procedure for building product models," *Robotics and Computer-Integrated Manufacturing*, vol. 15, no. 1, pp. 77-87, 1999.
[26] M. Modarres and S. W. Cheon, "Function-centered modeling of engineering systems using the goal tree-success tree technique and functional primitives," *Reliability Engineering and System Safety*, vol. 64, no. 2, pp. 181-200, 1999.
[27] S. Mittal and F. Frayman, "Towards a generic model of configuration tasks," Int. Joint Conferences on Artificial Intelligence, 1989, pp. 1395-1401.
[28] C. Forza and F. Salvador, *Product Information Management for Mass Customization: Connecting customer, front-office and back-office for fast and efficient customization* Palgrave, 2007.
[29] J. S. Gero, "Design prototypes. A knowledge representation schema for design," *AI Magazine*, vol. 11, no. 4, pp. 26-36, 1990.
[30] D. V. Steward, "The design structure system: a method for managing the design of complex systems," *IEEE Transactions on Engineering Management*, vol. EM-28, no. 3, pp. 71-74, 1981.
[31] M. Van Wie, C. R. Bryant, M. R. Bohm, D. A. Mcadams, and R. B. Stone, "A model of function-based representations," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 19, no. 2, pp. 89-111, 2005.
[32] T. U. Pimmler and S. D. Eppinger, "Integration analysis of product decompositions," *6th International Conference on Design Theory and Methodology*, vol. 68, pp. 343-351, 1994.
[33] T. R. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *IEEE Transactions on Engineering Management*, vol. 48, no. 3, pp. 292-306, 2001.
[34] A. Kusiak and N. Larson, "Decomposition and representation methods in mechanical design," *Journal of Mechanical Design, Transactions of the ASME*, vol. 117B, pp. 17-24, 1995.
[35] R. B. Stone and K. L. Wood, "Development of a functional basis for design," *Journal of Mechanical Design*, vol. 122, no. 4, pp. 359-370, 2000.
[36] Y. Kitamura and R. Mizoguchi, "Ontology-based systematization of functional knowledge," *Journal of Engineering Design*, vol. 15, no. 4, pp. 327-351, 2004.
[37] M. U. Soerensen, "Application of functional modelling in the design of industrial control systems," *Reliability Engineering and System Safety*, vol. 64, no. 2, pp. 301-315, 1999.
[38] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. London, UK: Springer-Verlag, 2004.
[39] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2 ed. Upper Saddle River, New Jersey, USA: Pearson Education, Prentice Hall, 2003.
[40] A. Newell, "The knowledge level," *Artificial Intelligence*, vol. 18, no. 1, pp. 87-127, 1982.
[41] N. P. Suh, "Axiomatic Design Theory for Systems," *Research in Engineering Design*, vol. 10, no. 4, pp. 189-209, 1998.
[42] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152, 1995.

Contact: Gudmundur Oddsson
Technical University of Denmark
Department of Manufacturing Engineering and Management
Produktionstorvet, Building 425
2800 Kgs. Lyngby
Denmark
Tel: +45 45 25 48 00
Fax: +45 45 25 60 05
go@ipl.dtu.dk
www.ipl.dtu.dk , www.productmodels.org and www.usec.dk