Demand Compliant Design

Joaquin Sitte
Department of Computer Science
Faculty of Science and Technology
Queensland University of Technology
Email: j.sitte@qut.edu.au

Petra Winzer
Department of Product Safety and Quality
Bergische Universitaet Wuppertal,Wuppertal, Germany
Email: winzer@uni-wuppertal.de

***Abstract*—In this article we describe in detail a design method that assures that the designed product satisfies a set of prescribed demands, while at the same time providing a concise representation of the design that facilitates communication in multi-disciplinary design teams. This Demand Compliant Design (DeCoDe) method was in itself designed to comply with a set of demands. The demands on the method were determined by an analysis of some of the most widely used design methods and from the needs arising in the practice of Design for Quality. We show several modes of use of the DeCoDe method and illustrate with examples.**

I. Introduction

Modern communication and transport technologies have made the world a single market. This global market attracts many competitors and the stakes for each competitor are high. To compete, producers have to offer their customers more features and greater variety, which keeps the complexity of products escalating. In the face of this, speed and effectiveness in conceiving and bringing products and services to market are paramount for business success. New or improved products originate in design. Therefore those producers who can design effectively and efficiently have a competitive advantage. The provision of effective and efficient design methods is the challenge for design science.

Design Science postulates two main activities: *build* and *evaluate*, which parallel *discovery* and *justification* in the natural sciences [1]. Like discovery in the natural sciences, the build activity is a cognitive activity that is not well understood, and this makes the development of effective design methods difficult. Design method are also difficult to evaluate. A design method must not only be *useful* in itself, it must be justified by being superior in some way to pre-existing methods. Such comparison, strictly speaking, should be conducted in the same way as the clinical trial of a therapy. This is practically impossible because of the huge effort, with little financial incentive, that would be required to complete the design of the same product following, in total separation, two different methods. Even if such a comparison was done the result would only be applicable within the same application context [1]. Despite the many design methods proposed over the years (Pahl et al. [2] list 130 of them) and because of the difficulties mentioned before, a strong demand persists for evaluating and extending the effectiveness of design methods and their scope.

Evaluation of a design method has two parts: One is the direct evaluation of the process of applying the method, according to a set of performance criteria such as e.g. resource requirements. The second is indirect evaluation through the quality of the designed product. Both the product and the design method can be evaluated in the same way. Let us consider first the evaluation of the product or artifact.

The *Usability* (or Utility) of a product is an evaluation criterion often quoted in the design literature [3]. Where usability is defined as ”the extend to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specific context of use” (ISO 9241-11). This definition is similar to the notion of quality.

Most industrial standards define quality as *the totality of features and characteristics of a product or service that bear on its ability to satisfy stated and implied needs* (ANSI/ASQC Standard A 3-1987, DIN 55 350, part 11). We shall prefer this formulation because it suggests at once a way of evaluating the quality of a product by assessing the degree to which it satisfies each of the demands placed on it.

The purpose of design is the production of useful artefacts [4]. To be useful an artifact must satisfy some specific demands or requirements. These demand must be stated at the start to give direction to the design process. Every design method should lead to a product that satisfies the demands placed on it. The Demand Compliant Design (DeCoDe) method to be described in this article has its root in Design for Quality [5] [6] [7] and has the goal of facilitating the design of artifacts that meet all the specified demands.

In Quality Engineering the primary focus is on customer demands. A product design needs to consider all the stages in the product life cycle, from the initial design to manufacturing, through its useful life, and to its final disposal. Over the product’s life cycle there are other stakeholders besides the customer (user) that have their own demands on the product, such as the various members of the organisation that manufactures the product (owners, shareholders, employees) and the persons and organisations in the community affected by the product. Therefore the DeCoDe methods requires that the demands from all the stakeholders be regarded [8]. These demands have to be made explicit and met in an effective way. When we refer to demands throughout this article we always assume that they include the demands from all stakeholders.

The discipline of Requirement Engineering (RE) provides techniques for eliciting, prioritisation and administration of requirements [9]. DeCoDe takes the requirements obtained by RE techniques as inputs to guide the design process to achieve a demand compliant product. It achieves this by making the relations between the demands, functions, components and processes explicit and maintaining consistency and completeness at all stages of the design.

Once a working set of demands has been determined there are usually a multitude of solutions that satisfy the individual demands to different degrees, making design a multi-objective optimization problem. The experienced designer carries out the optimization intuitively based on his/her knowledge and experience. Occasionally, a designer may resort to the help of quantitative optimization techniques but final quality of the product depends directly on the designer's grasp of the main demands on the product system and his/her understanding how to best utilise the available components. Designers that have reached such a high level of familiarity with the product system may not feel the need for a method that makes explicit things that he/she already knows. It is not until a new designer has to take over or when a new designer joins a team or when design information has to be communicated to persons outside the team that the need of a disciplined approach to design realisation is felt. Anyway, many of the contemporary design task can no longer be carried out by a single designer. Products are increasingly of cross-disciplinary nature combining hardware, software and service elements. Design methods need to foster a common understanding of the product across the discipline specific idiosyncrasies and procedures [10].

A design method is in itself an artifact that can be evaluated by the degree to which the method meets the stated demands on the method. Therefore, we begin in Section II with a review of the capabilities of some widely applied design methods. From assessing the explicit and implicit demands fulfilled by these method we formulate in Section III a set of demands that design methods appropriate for the complexities of contemporary designs should meet. Section IV describes the DeCoDe method and its elements. The three main DeCoDe application modes are explained, with examples, in Section V. In Section VI we conclude by reviewing how the DeCoDe Method stacks up against the demands formulated in Section III .

II. Capabilities of Current Design Methods

From the many design methods that have been proposed we selected nine for reviewing their capabilities. These methods were found to be the most used according to a recent survey [11].

We observe in general that these methods:

1) Originate from the specific design domains of mechanics, electronics or computer software, and are not directly transferable to other domains. Examples are the VDI Directive 2221 based on the Pahl/Beitz construction method [12], Y-method from circuit design [13] and software engineering methods such as the waterfall method, evolutionary development or extreme programming. [14]
2) None of the methods specifies a system description, leading to a variety of incomplete descriptions that impede the comparison of the effectiveness of the methods. One of the main shortcomings of existing design methods is that they do not require, or assist in obtaining, a description of the whole product system. We need one design method for a product that contains mechanical, electronic and informatics components, and that interdisciplinary teams can use to develop a complex product.

Among the discipline specific design methods, design automation is most advanced in the electronic circuit design domain, where it automates the detailed work of converting circuit schematics to VLSI (Very Large Scale Integration) circuit masks.

The Y-method is primarily an elegant conceptual model to guide the design process, rather than a prescriptive design method. The Y symbolises the representation of the electronic system along three design dimensions: functional, structural and geometrical. For each of the three representations there are series of levels that progress from the higher abstraction level to the detailed realisation. The selection of the three axes and their levels are specific to electronic system design. Current trends in electronic systems design strive towards the automatic generation of the final electronic circuit from a high level specification. These methods assume that a set of specifications have already been produced and that no human intervention will be necessary beyond the high level system specification. The designer will not need to have any specific knowledge of circuits and embedded software. However the success of these methods relies on the special characteristics of software-hardware design and is thus not transferable to other domains. Electronic system design automation only addresses the electronic hardware-software components of a product and as such lacks a complete product description.

As the complexity of technical systems grows, especially due to the progressive intertwining of information processing, electronic, electrical and mechanical components, the number of requirements and the number of stakeholders also increases. It is no longer sufficient to attend only to the customer requirements, as mainly practised by modern quality management. The requirements of all stakeholders, including employees, suppliers and company owners, and those that the enterprise poses on itself for strategic reasons, have to be stated and met. At the same time the interactions among the components of a product and with the environment need to be mapped systematically.

Although there are methods that consider all stakeholder demands they are domain specific, fragmented and only partially address the design problem.

The following modern quality management techniques that deal with customer demands are applicable to all kinds of artifacts:

QFD (Quality Function Deployment) focuses on the cus-

tomer [15].

FMEA (Failure Mode and Effect Analysis) focuses on faults [16].

FTA Fault Tree Analysis focuses on cause and consequence of chaining of faults [17].

However they only address partial aspect and consider only the demands from the customer [18] [19] [7] [20]. Although FMEA and FTA deal with faults, faults can be directly linked to demands because a fault causes that one or more demands are not met.

QFD is the most widely applied demand based design method and has evolved to make it easier and less costlier to apply [21]. QFD links the demands of the customer to the technical characteristics of the product with a matrix. This quickly shows which demands of the customer have already been realised and which demands of the customer have yet to be realised. If there is no link between a demand and the attributes of a component then the product needs to be redesigned. We used this idea of the QFD for the DeCoDe-method, but we include the demands of all stakeholders. Linking the demands with the components is one way of using the QFD. The QFD is also useful for comparing how various products meet customer demands. QFD allows a comparison of different products with regards to their degree of satisfaction of the demands. Thus the method tells designer what he has to develop during the product design. The main deficiency of this method is that it does not collect demands from all stakeholders. Normally the functioning of a product shows very quickly if demands are met or not. QFD does not describe functions. Therefore functions are not linked with the demands and the components and the design space lacks some essential variables. The demands must have a connection to the functions, the components and the processes characteristic of the product. Because, if a demand is not met with a at least one function, component and a process, then there must be a mistake.

A positive aspect of the QFD is that it contains a pair-wise comparison between the demands (The roof of the house of quality) This relative weighing process allows determining the order of importance of the demands for the customers. The designer can then evaluate how his product design realised the most important demands of the customers. The definitions of the demands, their detailed description and the statement of their interrelation, is necessary for the design process.

FMEA searches for faults during the design process but does the fault analysis without reference to the requirements. FTA shows the logical possibilities of mistakes. When there is a fault there is an unfulfilled demand.

Therefore requirement fulfilment could be checked by the FMEA. During the design process the possibilities of faults and their consequences have to be checked. However there is no connection between the QFD and the FMEA and the same is true for the connection between FMEA and FTA.

The FTA and the FMEA never show directly the demands and their connection to all aspects of a product. When mechanical, electrical or software engineers describe faults of the same product, they do it in different ways and the characteristics of the same fault may appear completely different. Therefore the communication between members in an interdisciplinary team of designers can become very difficult and we need to describe a product in way that is helpful for the interdisciplinary design team. This means that all professional experts must describe the product with the components, the function the processes and the demands, in the same way. This product description has to be checked during the design process at special key points.

Basically we must ascertain that the results of the QFD, the FMEA or the FTA can be wrong, if there is no description of the product system that links the demands, the components, the functions and the processes of the product. This description is absolutely necessary for an interdisciplinary design team.

There are still other methods that, like those mentioned above, only a focus on the demands and their satisfaction during the design process. For example Axiomatic design uses axioms to generate design solutions. [22], Kansei Engieering, incorporates emotional reactions of the of user into the development strategy [17] or Requirement Engineering (RE) [23] that focuses on strictly establishing and meeting the demands.

Kansei Engineering and Axiomatic Design also only focus on the customers and not on all stakeholders of a product. These two methods link the customer demands with the function and the components of the product but not with the processes. Requirement Engineering looks after how the demands from all stakeholders are met through the technical characteristics of the product. The functions and the processes are not included. Requirement Engineering has no connection to FMEA, to FTA or QFD etc. At present there exist no practical methods that systematically reveal the interrelations in complex systems and consequently support a full exploration of the design space [15], [16] [24].

Originally, Requirement Engineering was defined in software engineering as a systematic, collection, evaluation and documentation of demands trough a general iterative procedure [25] [24]. Requirement Engineering relates customers' and other stakeholders' demands to products and processes. Yet the relations of product components and processes, as well as their functions are hardly ever considered.

In a recent review of the requirement engineering literature Berkovich et al. found that the deficiencies are greatest for hybrid products that combined hardware, software and service aspects [9]. Such hybrid products, which include mechatronic products, are becoming an ever larger fraction of the product palette. In recent times Software Engineering has contributed strongly with concepts, methods and tools for requirement elucidation, analysis, negotiation, documentation and validation. The need of a clear requirement specification is acute in software engineering because of the complexities of designing a software product that is essentially intangible and that reveals itself only partially at a time through its usage. Because of this nature, Software Requirement

Engineering predominantly focuses on *user* requirements. However, irrespective of the nature the product, demands arise not only from the user but also form the other stakeholders of the product. We need a method that is universally applicable across all industries, and not like Requirement Engineering, that is been primarily intended for Software Design [26]. The method must foster the generation of new product ideas, the detection of faults or uncover unrecognised relations between the individual views. In addition to handling the demands of all the stakeholders a systematic design method has to expose the interrelationships between the components, processes and functions, which constitute the dimensions of a product system. Furthermore, each of these dimensions (requirements, functions, components and processes) should be structured hierarchically to allow the narrowing down of the design space. From the above review one notices that the current design methods comprehensive and only address particular aspects.

An early step in the method has to be the systematic comparison of the multitude of demands so that equal or similar requirements can be identified and unified. This unification has to be done in such a way that each demand can still be traced back to the originating stakeholder. Contradictory demands need to be assessed and weighted.

It is likely that new requirements arise during the design process, therefore it must be possible to insert new demands into the existing catalogue at the right place, anytime. Design knowledge should be stored systematically while guaranteeing knowledge base transparency. Product liability laws also make it imperative that traceability and effective documentation are an integral part of the method.

All the methods specify a process in that they prescribe a series of steps that will lead to the final design. However these steps are not specified clearly enough to allow a unique realisation by different designers and designs.

III. Demands on the Design Method

A design method consists of collection of activities (processes), sequential and/or concurrent, whose final result is a prescription of how to build an artifact. Following the design method should assure that the built artifact meets the demand set out at the start of the design, irrespective of whether the artifact is a material product, an organisational structure or a service. There is no great difference between a product and a service from the design point of view. A service consists of a collection of processes that realise the functions that fulfil a set of demands. The functions are performed by a service system consisting of artefacts (equipment) and people. Likewise, a material product, once it is in use, also executes, driven by an operator, a collection of processes to provide the functions it was designed for.

Furtehrmore, a method is the result of a purposeful design, and as such it is an artifact expected to meet a set of demands. Therefore the evaluation of a method consists of assessing how well it meets those demands.

From the general analysis and the review of the design methods in the previous section we formulate the following set of demands that a design method suitable for the complexity of contemporary designs must satisfy:

- Provide a systematic procedure for the discovery and precise formulation of demands.
- Include the demands from all the stakeholders.
- Encompass the demands from all phases of the product life cycle.
- Produce a product description in terms of the functions, components and processes and their relations to the demands and the relations among themselves.
- Provide a common product description amenable to be developed and used by all the disciplines participating in the design.
- Represent the design knowledge in a concise way that is easy to communicate.
- At all times maintain the consistency between the demands and the function, components and processes intervening in the design.
- Continuously detect omissions in the demands, functions, component and processes in the progress of the design.
- Identify potential product faults at design time.
- Document all changes in the demands, functions, components and processes over the design period.
- Allow back and forward traceability of all design changes.
- Be universally applicable to all kinds of artifacts.
- Prescribe a structured design process that can be automated.
- Be able to interface with design methods from different disciplines.
- Foster the generation of new product ideas.

Table I summarises to what extent the design methods discussed in the previous section meet these demands.

None of the methods examined in the table meet the demands 4, 7 and 8, which relate to the objective of assuring a demand compliant design, which indicates the need for a method that also meets these demands.

IV. The DeCoDe Method

Figure 1 represents Demand Compliant Design (DeCoDe) as a *design method* that comprises a model, a set of tools and a procedure. The model consist of the three views of a product: the functional view that describes product by the function it performs, the component (structural) view that describes the product by its parts and their relations, and the process view that describes the product by the processes executed by and on the product during its life-cycle. The tools consist of the various dependency matrices and the demand, function, component and process catalogues that specify the design. Finally DeCoDe specifies an iterative series of steps (procedure) that based on the model use the tools to evolve the design.

No.	Demand	QFD	AD	DSM	VDI 2221	V-Model	RE	Kansei	FTA	FEMA
1	Provide a systematic procedure for the discovery and precise formulation of demands	✓	✓	✓	✓	✓	✓	✓		
2	Include the demands from all the stakeholders		✓	✓			✓			
3	Encompass the demands from all phases of the product life cycle	✓		✓		✓	✓	✓		
4	Produce a product description in terms of the functions, components and processes and their relations to the demands and the relations among themselves									
5	Provide a common product description amenable to be developed and used by all the disciplines participating in the design.	✓	✓	✓	✓	✓	✓	✓		✓
6	Represent the design knowledge in a concise way that is easy to communicate		✓	✓	✓	✓				
7	At all times maintain the consistency between the demands and the function, components and processes intervening in the design									
8	Continuously detect omissions in the demands, functions, component and processes in the progress of the design									
9	Identify potential product faults at design time								✓	✓
10	Allow back and forward traceability of all design changes			✓	✓	✓				
11	Be universally applicable to all kinds of artifacts.	✓	✓	✓	✓	✓	✓	✓	✓	✓
12	Prescribe a structured design process that can be automated.		✓	✓						
13	be able to interface with design methods from different disciplines		✓	✓		✓	✓	✓		
14	Foster the generation of new product ideas				✓	✓	✓	✓	✓	✓

TABLE I
COMPARATIVE SATISFACTION OF DEMANDS BY THE MAIN DESIGN METHODS.

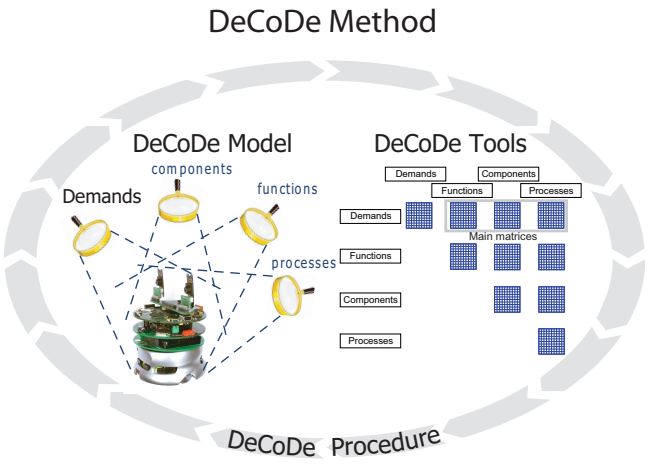

Fig. 1. DeCoDe is a design method that comprises a model, a set of tools and a method.

The DeCoDe method is based on simple ideas for organising the design information and capturing essential relations between the main design problem elements. It provides a framework for design knowledge documentation and for tracking the evolution of the design. It describes the whole of a complex product, finds out the gaps or the failures of a design and may create new solutions that are assessed against the demands for the selection of the best design. The DeCoDe method is a first step towards a formal description of the design task that will allow the application of quantitative techniques for solving the multi-objective design problem. The hallmark of the DeCoDe is the linking of the demands on the product with three complementary *views* of the product and the maintenance of consistency at all times as the design evolves.

A. Demands and Product Views

Demands on a product express either a function or a constraint on one or more functions, components or processes. The product or artifact needs to perform these functions under the specified constraints. The performance of the required *functions* will be the result of the running of a collection of *processes*. In turn, the processes result from the dynamic interaction of the physical components of the artefact. Accordingly the DeCoDe method takes three complementary views of a product. These views see the product as a collection of functions, a collection of processes or a collection of components, all linked to the demands. Each view is represented by the corresponding list, or catalogue, of functions, processes or components. The DeCoDe method uses matrices to capture and manipulate the relations between the demands and the elements of the three product views. There are relations among the elements in each of the catalogues and there are relations between the elements of different catalogues. These relations are captured in *Adjacency* matrices [27]. The rows of an *adjacency* matrix correspond to the entries in one catalogue and the columns to the entries in the same or another catalogue. There are 10 such adjacency matrices as shown in as shown in Figure 2.

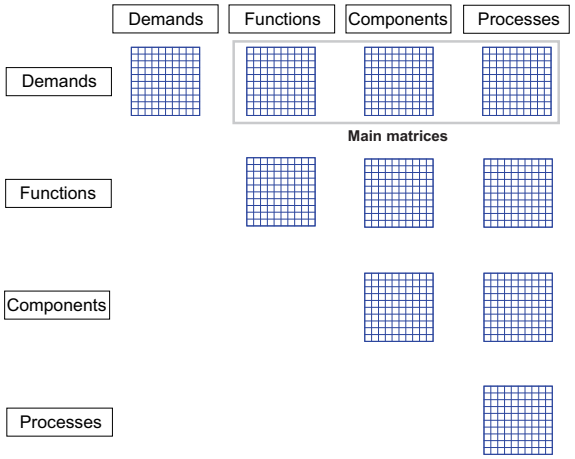

Fig. 2. The *adjacency* matrices of the DeCoDe method capture the relations between the elements of the catalogues. There are 10 possible matrices.

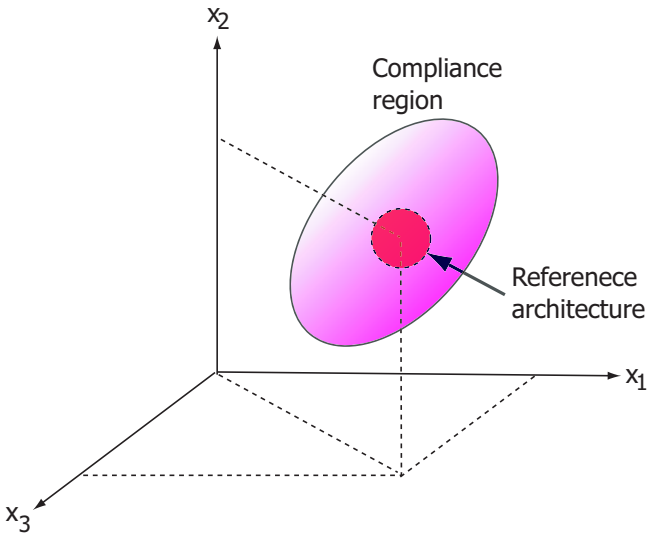

Fig. 3. Designs correspond to points in the design space spanned by the design variables that influence the satisfaction of the demands.

B. Navigating Design Space with DeCoDe

The notion of design space is useful for exploring design alternatives. Each design solution is a point in the space of design variables. Product design variables are those that influence the satisfaction of the demands on the product. The design space is determined by the nature of the product, and the demands placed on it.

Each design solution (design point) will satisfy the demands to a certain degree. Design optimisation searches for the design point that has the highest degree of satisfaction of demands. The solution space consists of those regions in design space that have a high degree of satisfaction of the demands. [1]

For example a digital camera can be characterised by the focal length of the lens assembly f, the lens diameter ϕ, width d_x and height d_y of a pixel and the number of pixels n_r in a row and a column n_c of the sensor array. Although this is a highly simplified characterisation of digital camera, it may be sufficient for evaluating the image quality produced by the camera. Any digital camera, real or imagined will have specific values of these five variables and the designer is free to choose from among a range of possible values. Thus a particular camera can be represented by a point in this five-dimensional design space

$$\vec{C} = (f, \phi, d_x, d_y, n_r, n_c) \tag{1}$$

In general a design can be described by a vector of values of the design variables u_i

$$\vec{u} = (u_1, u_2, ...u_N)^T \qquad i = 1..N \tag{2}$$

Because a camera is inherently defined by the function of projecting a 3D scene on a 2D image recording medium, there are clearly regions in design space where the corresponding construct no longer meaningfully provides the function expected from a camera. For example a camera with focal length of 1 m will still provide the desired function but will be useless in taking a group photo of some friends. Thus, although in principle a camera could be anywhere in the design space defined by the range of the variables, useful designs will only occur in particular regions of design space. These useful regions will be defined by the demands on the product. High magnification tele-cameras will occupy a different region in design space than pocket cameras.

C. Evaluation of Designs

Demands have to be formulated in a way that they can be evaluated quantitatively, even if only on a coarse scale. Demands for which no operational procedure of measurement of their degree of satisfaction can be specified, are largely useless for the design. With properly specified demands the *goodness* of the design can be evaluated. For that purpose we introduced the Comprehensive Quality Function (CQF) [8] which, for simplicity, is taken as a weighted sum of the degree of satisfaction of the individual demands

$$\mathcal{Q} = \sum_i v_i Q_i(\vec{u}) \tag{3}$$

Where $Q_i(\vec{u})$ is a function that measures the degree of satisfaction of demand i at the design point $\vec{u}$. The factor v_i weighs the importance of demand i. These weighting factors may be different for different usage contexts of the product. How the weighting factors can be obtained is explained in section IV-D below. With the CQF the design problem can be formulated as maximisation of the CQF over design space. The CQF describes the artefact by a scalar value, and in this

[1] A different notion of solutions space is used by Gries [28] where the axes are the measures of satisfaction of the individual demands. To find the optimal solution multi-objective optimization is used (Pareto optimal surface).

way approximates what is essentially a multi-objective design problem by a single objective optimisation. Of course nothing precludes to use of multi-objective optimisation techniques.

D. The DeCoDe Adjacency Matrices

Matrices have been widely used in design methods [10] because they can succinctly represent relationships between a large number of elements. Adjacency matrices represent graphs. In DeCoDe each element of a catalogue is a node in a graph and each element of a matrix quantifies an edge.

The edges (lines connecting the nodes) of the graph represent the relations between the elements. Depending on the chosen relation these graphs may be undirected or directed. The matrices between different lists represent bipartite graphs. In a bipartite graph there are two sets of nodes and links exist only between nodes belonging to different sets.

In general DeCoDe uses weighted edges to not only express the existence or not of a relation but also to quantify that relation.

Assigning a binary value to a matrix element simply expresses the existence or not of some relation. For example, in the component-component matrix, a binary value can indicate the direct physical connectedness of two components. The relation is symmetrical and the adjacency matrix represents the block diagram (graph) of the product. Apart from telling to which other component a particular component is connected, other useful information can be directly extracted from this matrix. For example by summing the rows, or columns, the total number of adjoining components can be determined for any given component.

Sometimes demands are classified as essential or as desirable demands. DeCoDe does not make this distinction, instead it ranks the demands according to perceived importance. One can find many different relation between the elements of catalogues beyond the simple adjacency. For example, not all demands are of equal importance and the relation: *more important than* can be applied to the list of demands. The degree of satisfaction of the relationship can be expressed on a scale of from $-M$ to M. M indicates that the relation is fully satisfied and $-M$ is that the antagonistic relation is fully satisfied. To evaluate the relation *B is more important than A* between two demands A and B, we ask ourselves would we sacrifice some conformance to A in favour of B. For example, if demand A requires the robot to be smaller than 100 mm diameter and demand B requires the robot to be cheap, then would I rather have a larger robot if it were cheaper? How strong is that preference? Note that a smaller robot can be more expensive than a larger robot because it is more difficult to fit the same functions in a smaller space. If it were strong then we would accept a size increase even if the cost were only cheaper by a modest amount. Conversely if we evaluate the importance of A in relation to B then if A is less important (opposite) we would evaluate the relation to a negative number. If it is much less important then one would accept a large size increase for a small increase in cost.

This relation allows the ranking of the demands as follows: Let us assume that there is a function $v(q)$ that weights the importance of a demand. The value of this function is generally not known. However, we are only interested in the relative values between demands and these can be estimated by the design team. The relative importance can be measured as a real value or as an integer in a specified range. If demand j is more important than demand i then the difference in importance $r_{i,j}$ of two demands is

$$r_{i,j} = v(j) - v(i) \tag{4}$$

and

$$v(j) = v(i) + r_{i,j} \tag{5}$$

where $r_{i,j}$ is positive.

We now fill the values in the demand-demand adjacency matrix with estimates of the relative importance $r_{i,j}$, and take the sum over row i

$$\sum_{j=1}^{n} r_{i,j} = \sum_{j=1}^{n} (v(j) - v(i)) = \sum_{j=1}^{n} v(j) - nv(i) \tag{6}$$

Solving for $v(i)$

$$v(i) = \frac{1}{n}(\sum_{j=1}^{n} v(j) - \sum_{j=1}^{n} r_{i,j}) \tag{7}$$

$$v(i) = c - \frac{1}{n}\sum_{j=1}^{n} r_{i,j} \tag{8}$$

We find that the importance of item (demand) i is the average importance of all items, which is a constant, minus the average of the elements in row i of the matrix. If that average is negative then the importance of item i is greater than the average importance and if it is positive then item i is less important than the average. By rearranging the rows of the matrix so that $\sum_{j=1}^{n} r(i,j)$ increases from the top to the bottom the rows will be ordered in *decreasing* importance from the top to bottom. Applied to the demands this means that we have a prioritised list of demands. The values of $\sum_{j=1}^{n} r(i,j)$ can be used, after normalisation, as weighing factors v_i in the CQF.

When making the estimates we must not forget that the relative importance matrix is antisymmetric

$$r(i,j) = v(j) - v(i) = -(v(i) - v(j)) = -r(j,i) \tag{9}$$

The values of the relative importance cannot be assigned freely to the upper triangular part of the matrix because they are constrained by the following consistency requirement:

$$r(i,k) = r(i,j) + a \tag{10}$$

$$v(k) - v(i) = v(j) - v(i) + a \tag{11}$$

$$v(k) - v(j) = a \tag{12}$$

$$r(j,k) = a \tag{13}$$

In words this means that if item k has an amount a more relative importance to item i than item j has to item i, then item k also has a more relative importance to item j. This

can be seen as a kind of transitivity condition that has to be satisfied by the relative importance estimates.

To see how it works consider that the first row of the adjacency matrix has been filled in until column m. Then the values in column m from row 2 to m are already determined by the set of $m-1$ conditions:

$$\begin{aligned} r(1,m) &= r(1,2) + (r(1,m) - r(1,2)) && (14)\\ r(1,m) &= r(1,3) + (r(1,m) - r(1,3) && (15)\\ &\cdots && (16)\\ r(1,m) &= r(1,m-1) + (r(1,m) - r(1,m-1) && (17)\\ & && (18) \end{aligned}$$

that require

$$\begin{aligned} r(2,m) &= (r(1,m) - r(1,2)) && (19)\\ r(3,m) &= (r(1,m) - r(1,3)) && (20)\\ &\cdots && (21)\\ R(m-1,m) &= (r(1,m) - r(1,m-1)) && (22)\\ & && (23) \end{aligned}$$

By the symmetry condition the values of row m are also determined up to element $m-1$. By definition $r(m,m)$ is zero. Thus we see that as we fill in the first row from left to right all the remaining values of the square matrix underneath the filled first row are determined. The task is thus to fill in the first row in such a way that al the relative importances in the matrix *make sense*.

Specifying the direction and strength of the interaction between nodes gives rise to another useful adjacency matrix called the *influence* matrix. This is again a directed (asymmetric) relation because the influence of item i on item j can be stronger than the influence of item j on item i. For example the designers choice of the size of any image buffers will be strongly influenced by the number of pixels in an image sensor. But the designer is unlikely to let the size of the image buffer determine the choice of the resolution of the image sensor. The mutual influence of components may be obtained from a technical analysis of the design. For products already deployed the influence matrix may be inferred from statistical fault data. Let values of the elements in row i represent the influence of component i on all the other components. The sum of the values in row i gives a measure of the total influence of component i on the design. Likewise the sum of the values in a column j of the influence matrix gives a measure on how much component j is influenced by all the other components. We call this the sensitivity of component j. Plotting the sensitivity against the influence for each components provides a graphical representation of the criticality of the components in the design (see example in Figure 8). In the lower right of the graph we find the components with low influence and low sensitivity. These are not critical components. In the upper left of the graph we find components with high influence and high sensitivity. These are critical components. They will be strongly affected by changes elsewhere in the design while at the same time a change in the component will strongly affect the rest of the design.

The same method can also be applied to finding the influence of a function, process or component on the satisfaction of a demand, and vice-versa the effect of a demand on a function, process or component. This is very helpful when the demands on a product change. The row of the demand reveals which function, process or component will be affected.

The matrix that relates the demands to the component can capture some essential information that is not easy to express otherwise. Each row corresponds to a demand and each column to a component. An element of this matrix can represent an estimate of how much a component contributes to the satisfaction of a demand. Thus if a change is made to a component, by checking the values in the corresponding column one can immediately see which demands are affected and even how much; and vice-versa,

Often variations of an artifact are needed where the requirements vary somewhat around a set of core requirements. In this situations it is useful to develop a reference architecture for the artifact (Example family home, mobile phone camera or robot vision) The reference architecture will list a series of essential components, functions and processes of the artifact and a description of their interrelations so that the resulting artifact meets the core requirements.

V. DeCoDe Application Modes

The DeCoDe method can be applied to a design task in different ways according to the maturity level of the current state of the design.

A. The Direct Mode

This mode is appropriate when most of the demands have already been specified. The steps in the method are illustrated in Fig. 4.

B. The Constructor's Mode

The constructors believe they have an overview of the requirements and thus immediately begin thinking about the essential components of which the new product should consist. Because constructors are not inclined to change their ways, we show how the DeCoDe method can be used to fit their approach. In the following we describe a variant of the DeCoDe method for constructors and illustrate it with the example of the design of a logistics installation consisting of a conveyor line.

One of the demands for the conveyor line was that it must be able to restart under full load after an emergency stop. This is clearly the most stressful process. The demand component matrix allowed the identification of the components involved in this process with the asynchronous motor being the key component. To discover the required motor characteristics it proved necessary to simulate the process. Following the DeCoDe method allowed to target the simulation to the essential parts.

Th constructors mode is appropriate when an existing product has to be changed or improved. In this section we use

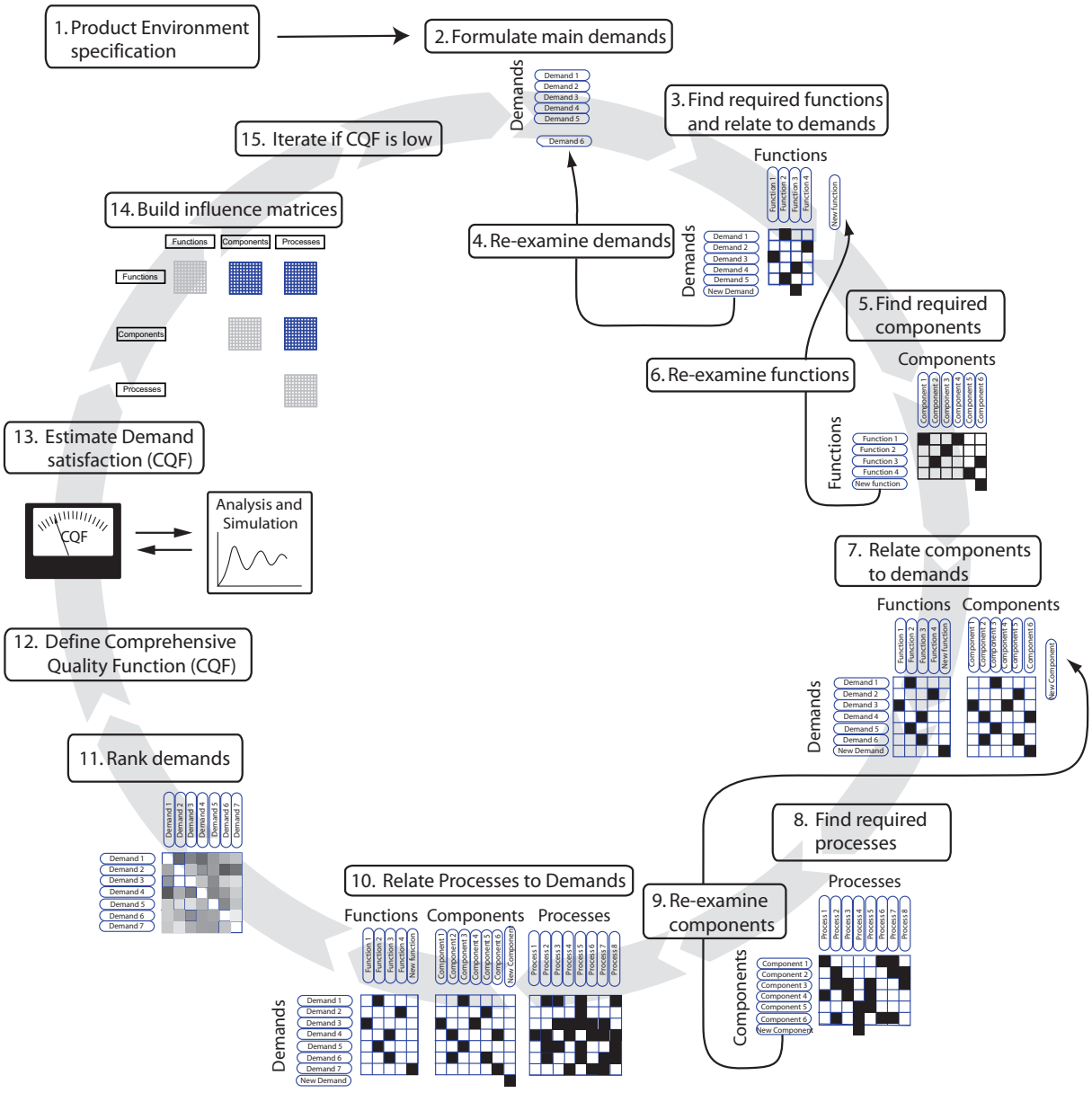

Fig. 4. Steps in the DeCoDe method.

the word *constructor* instead of designer, to emphasise that the constructor is a designer working close to the engineering side of the fabrication of the product. Constructors visualise the product. When they design a new product they:

- already have a vision of the product in their mind.
- are aware of some wishes form the customer or
- have some understanding of the market and competing products.

The DeCoDe method can accommodate the constructors approach by making the selection of components and their interrelations via the components-components matrix the first step in the DeCoDe method as shown in figure 5.

The relation of the functions to the components is captured in the function- components-matrix. This matrix will reveal which functions are missing or which components are missing. The interdependence between the functions is expressed in the function-function matrix. Only in the 3-rd step the designer turns to clarifying the demands by relating them to the functions in the demand-function matrix. From this matrix the constructor can see which functions already fulfil which requirements, which new requirements originate from the functions and which requirements are not fulfilled yet. Constructors primarily think in terms of components and functions. however, while the components are detailed in drawings and technical specifications, the functions are often not even listed.

By following the DeCoDe method, the functions of the product are made clear to outsiders for the first time in

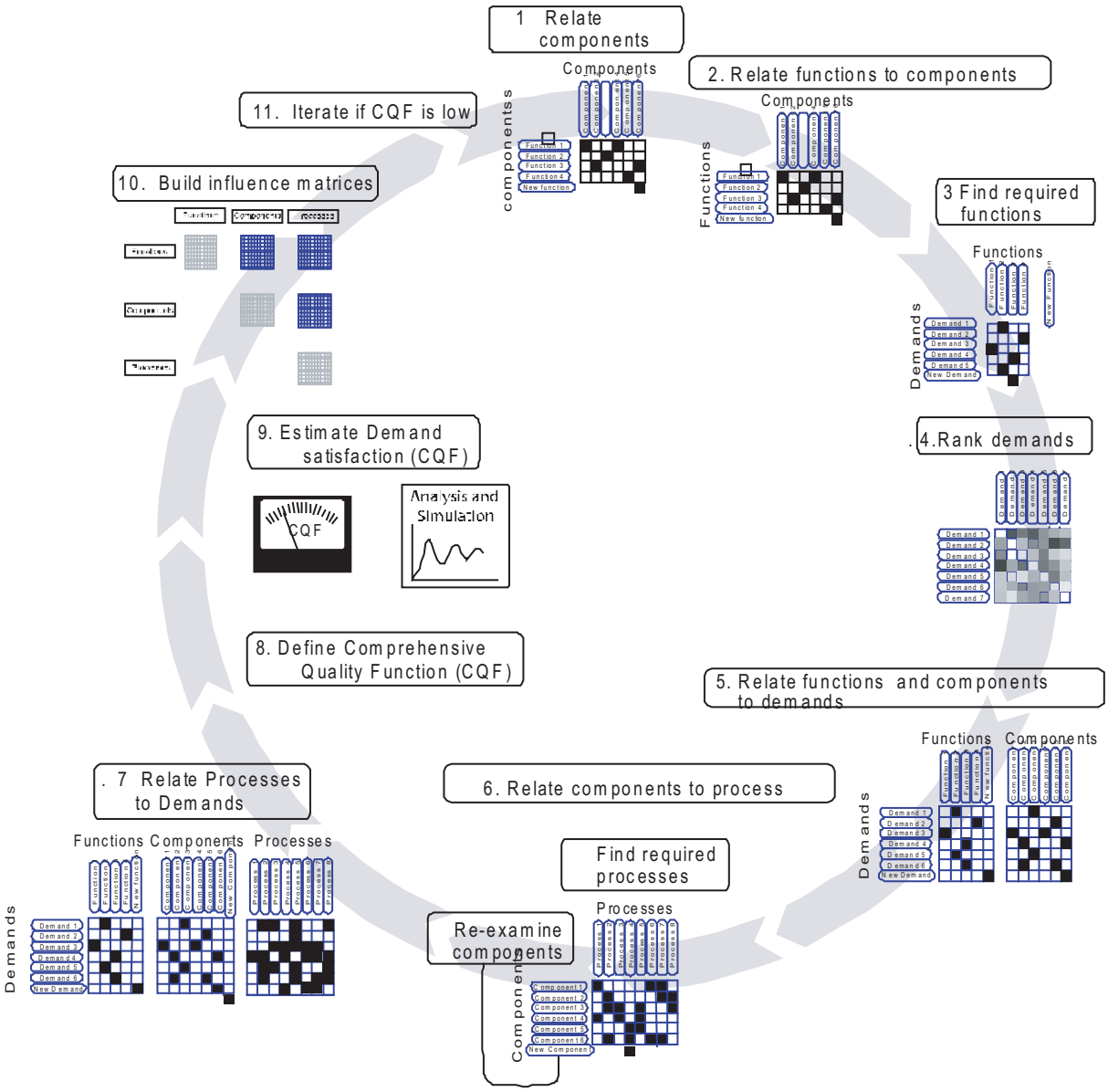

Fig. 5. Steps in the DeCoDe method adapted to the Constructor's approach.

the early stages of the birth process of the product. Up to now the ideas of the designer were manifest only in the drawings. Drawing give details of components but do not make explicit the functions of the components. However, it is only through the functions that their degree of fulfilment of the stakeholders' wishes can measured. That is why the functions must be made absolutely clear.

Only when the constructor has compared the requirements to the functions, by which new requirements may have originated, is he ready, to prioritise the requirements. The DeCoDe method requires this in the 4-th step by filling in the demand-demand matrix through the pair-wise comparison and relative weighting of demands.

In the 5-th step the weighted requirements are once more compared to the components and functions. This comparison checks to what extent the requirements are already fulfilled by the components and functions of the new product, which in turn points to the directions the new product must be developed further. Exactly those components and functions that fulfil the most important requirements either not at all or only partially are filtered out by this matrix. At the same time, new requirements that have not been recognised up to now, can be also determined by the interaction of components and functions. This allows the constructor attain a better design process that leads to a product of better quality. All these steps can be iterated as many times as needed in the judgement of the constructor.

Only when the constructor has a more precise image about

the new product, he is ready to think about how this can be produced. In this phase of the development process the constructors consider how the components can be produced by which processes. To realise this systematically and comprehensively the DeCoDe method offers the components to process matrix in step 6.

By assigning possible process to every component is becomes clear whether the components can really be manufactured. If it cannot be produced, a new component must be chosen or a new processes must be conceived. New requirements can arise again from this consideration. These are again compared with the functions, components and processes, as shown in step 7. With the demand-function, demand-components and demand-process matrices it becomes clear which demands are already fulfilled and which demands must still be fulfilled by changing components, functions or processes in the future.

With this basis it can be decided in step 8, which requirements are the most important to be fulfilled. The relative importance of the demands determined in step 4 provides the weight coefficients of the demands in the Comprehensive Quality Function (CQF). In step 9 the degree of fulfilment of the requirement is determined by the components, functions and processes which allows the evaluation of the CQF. The 10-th step encompasses the representation of all cross dependencies between the requirements, components, functions and processes with the remaining matrices as in figure 2.

If new ideas originate or the customers have new demands or the processes change, then all 10 steps can be executed again.

The result of executing the steps just described was tested on the design of a new logistics installation comprising a roller conveyor line. Conveyor lines are key logistic systems in today's industrial infrastructure, for example baggage conveyors in airports. Their failure immediately causes substantial financial loss and disruptions. The goal of the design was to optimise performance on several indicators, among them energy consumption. Conveyor lines are mostly driven by asynchronous electric motors. Because of low efficiency of these motors on starting and stopping these motors are typically over-dimensioned in conveyor systems. Figure 6 shows the experimental conveyor systems with its main components. Because of the complex interactions between the motors, rollers, belts, support structure and load to be transported, an interdisciplinary team of designers was appointed. The various specialists initially had their own ways of describing the components and functions of the conveyor system. After a first agreement on the components of the system it was necessary to conduct several brainstorming sessions to reach a common understanding and description of the functions of the conveyor line.

The relation of the components to the functions were analysed and represented in the Function-Component matrix shown in figure 7. In this form the dependencies are easy to see and to communicate to team members and third parties. The rollers, the belt and the goods that must be transported

Fig. 6. Components of the conveyor line mechatronic system. The components are the straight segments, curved segments and drive mechanism.

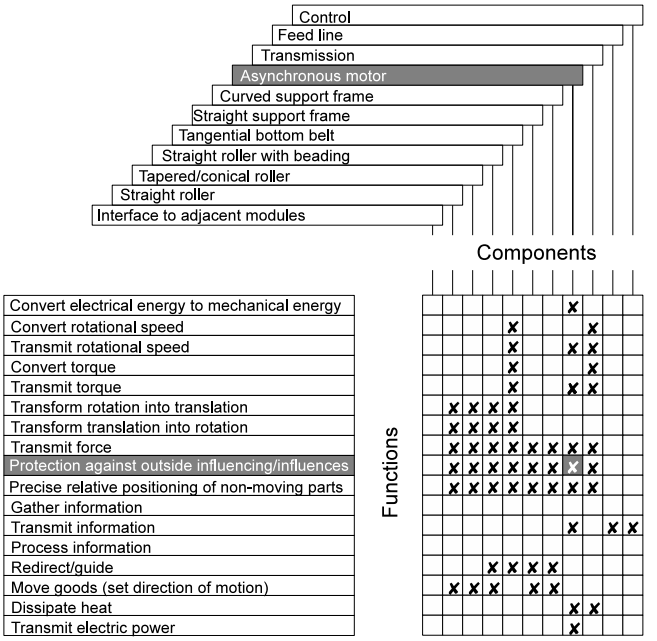

Fig. 7. The Function-Component matrix for a section of the conveyor system. The components related to the asynchronous motor and the safeguard function are highlighted and can be identified easily.

influence substantially the impulse to be provided by the drives. In summary, the application of the DeCoDe method gave the design process a clear structure, facilitated the common understanding of the design issues by the team members and mistakes could be minimised. Also the manufacturability of the logistic installation and the resulting demands were considered early.

C. The Progressive Mode

This mode is appropriate when starting a new design and only very general and vague demands are given. The method builds a demand list iteratively and concurrently with the functions, components and processes lists. Here we will illustrate the method with the design of an educational and entertainment soccer playing robot. A autonomous soccer playing robot is an advanced mechatronic product involving leading embedded systems technology. We chose to use it as an example artifact to illustrate the main steps of the DeCoDe

procedure on a new design.

1) Demands: The first thing to do is to phrase a brief overall description of the product. In this case we choose:

> An inexpensive robot that can participate as a player in a team of one or more robots in a soccer style game as specified by the AMiRE 2008 rules.

We will call this the *zero-th order* demand or system specification.

Designing a robot for the AMiRE 2008 game rules allows us to use these rules to derive a set of specific demands for the robot. The rules prescribe the maximum diameter of robot, size and markings of the playing field and the duration of the match. We will add other demands that derive from marketing considerations.

Therewith we arrive at an initial demand set. The robot must:

1) fit into a cylinder of 100 mm inner diameter.
2) move and manoeuvre across the playing field with a speed of up to 1 m/s
3) detect and locate the stationary objects in the playing field such as markings, goals and boundaries.
4) detect, identify and locate mobile objects in the environments such as the ball (a white 40 mm diameter squash ball) and other robots.
5) distinguish between the various robots on the playing field.
6) perceive the various referee whistling signals for starting and stopping the game.
7) allow the user to change the robot's behaviour (programmable).
8) have sufficient on board computing capacity for implementation of game winning behaviours.
   and
9) have a cost that is affordable to tertiary students, schools and university departments.

This list of demands should suffice at this early stage. The demands could be ordered by doing pair-wise comparison. However, this is better done after having determined the functions, components and processes in a first round as there may arise additional demands or modifications.

A mutual influence matrix of the demands could also be build. For demands the mutual influences should be very small as we want the demands to be orthogonal [22].

Next we need to identify the functions the robot has to perform to meet the demands.

2) Functions: The first demand is a *constraining* demand and as such does not translate into a function. The next demand: Move and manoeuvre across the playing field requires several functions. Just moving needs two functions:

1) Vary the speed of motion (accelerate/decelerate).
2) Change direction of motion.

Manoeuvring, such as moving towards a desired position on the playing field while avoiding other players and remaining within the play field, requires the perception of objects and features of the environment, their relative location in relation to the robot, and the capability of moving appropriately in response to these perceptions;

3) Recognise and locate goals (angle and approximate distance)
4) Recognise and locate playfield boundaries
5) Recognise and locate other robots
6) Recognise and locate the ball

For the management of the game it is necessary that the robot be able to switch between behaviours, such as: dont move and wait for signal, play, or move to designated position on the field, in response to referee whistle signals. The corresponding function is:

7) Interpret whistle signals and select behaviour.

The next demand: *allow the user to change the robots behaviour (programmable)* can be realised by a function:

8) Replace current behaviour program on the robot by another one and/or modify settings for hardwired behaviour

Demands 8 and 9 are constraining demands that do not generate a function.

At this stage we notice that if we have a function that changes the behaviour of the robot we also need a function that:

9) enacts a soccer playing behaviour that is stored in the robot as a computer program and/or is hard wired.

Having gone through all the initial demands we have a first list of functions and we can proceed with filling in the demand-function adjacency matrix. Not every demand generates a function, but clearly every function needs to contribute to meeting at least one demand. This can be tested by filling in a demand-function adjacency matrix that values the contribution of each function to each demand. Constraining demands may be more difficult to meet if certain functions have to be present. This is the case for the cost and may also be the case for the size. In this case we need to assign some negative value to the contribution of these functions to the constraining demands.

By filling in the demand-function adjacency matrix we see that although we have a function that enacts a selected behaviour, the kind of behaviour is left open. The zero-th order demand of playing a soccer style game autonomously is not reflected in the current demand list. The need for specifying this ability is clear, as different behaviours will require different functional capabilities. Such kinds of mistakes and oversights are common occurrence in early designs. The inherent consistency requirement of the DeCoDe method facilitates the early detection of errors and omissions.

3) Components: The next step is to list potential components required to provide the functions identified so far. Going through each function leads to the need of the following components:

1) Motors
2) Gearbox/transmission
3) Motor controller

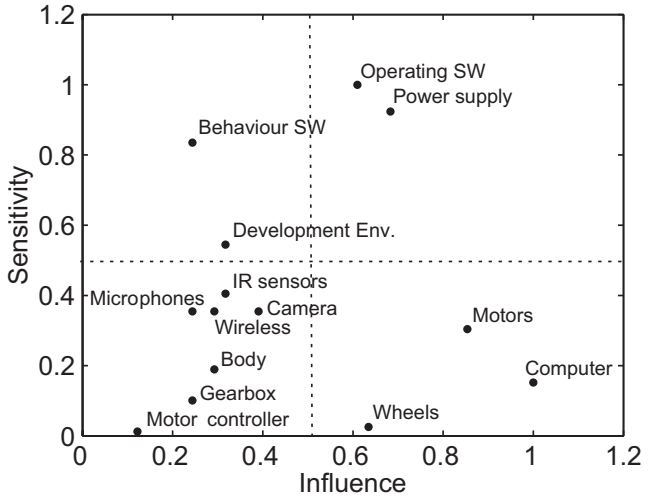

Fig. 8. Mutual influence of mobile robot components. The components in the upper left quadrant are critical in the sense that they have strong influence but also are highly susceptible to other components.

4) Wheels
5) Body
6) Power supply (battery)
7) Computer
8) Proximity sensors (infrared)
9) Camera
10) Microphones
11) Wireless transceiver

For realising the functions 3 to 6 a camera and infrared proximity sensors are proposed. For long range object detection a camera offers lower cost and higher flexibility than alternative sensors such as ultrasound or laser range finders. For short range obstacle detection infrared sensors are a low cost choice.

Now the function-component matrix and the component-component influence matrix can also be built. Figure 8 shows the influence vs. sensitivity plot for the soccer robot components.

The main matrices encapsulate some of the design knowledge at this state. This information can be recovered by reading the matrices. In particular the row and column sums provide an easy interpretation. In the demand-component matrix for example the sum of a row indicates the influence of the corresponding demand on the whole design. We see that the demands for controlling the motion are the ones that have the most influence on the whole robot design.

The sum of the columns indicate how much influence the demands, taken together, have on the corresponding component. Not surprisingly the component most susceptible to the demands is the computer. In the demand component matrix the row corresponding to the demand on cost can be filled with the relative cost of each component providing directly a cost estimate.

D. Processes

As mentioned earlier there are two classes of processes: those related to the life cycle of the product (its design, manufacture, deployment and disposal and those processes that realise the purpose of the product. Here we will only consider the latter type.

The components provide the physical substrate for the processes that perform the functions that satisfy the demands. Therefore a good starting point for listing the processes that are required is by going through the component catalogue and visualise in what processes they participate. Going through the list we can easily build the component-process matrix. On a first pass we came up with the following processes:

1) Execute behaviour
2) Activate new behaviour
3) start computer
4) power-on self test POST
5) move under remote control
6) emergency stop
7) recharge battery
8) Monitor sensors

The first process identified: *execute behaviours* immediately points to a serious omission in the component catalogue. The computer is a component required for executing behaviours, but to do so it also requires software. So far we had not listed software in the component catalogue. Clearly there are many software components required for the robot to play a soccer game. This deficiency becomes very clear when we build the demand-processes matrix and observe that this matrix is very sparse with few processes relating to the demands. This cannot be. Either important processes are missing and/or demands are still missing.

The processes of loading a new program, move under remote control and monitor sensors suggest another demand, namely that these processes, being directed by the robots master (owner) must be easy to do for the human master. The demand can be formulated as:

Intuitive and easy to learn user interface and behaviour generation

E. Demand Refinement

During the first *round* of DeCoDe several matters arose. One was the lack of a demand that specified that the robot be capable of soccer playing behaviour. This omission can be corrected by adding a demand that could be formulated, for example, as: *Execute a soccer playing behaviour autonomously for the duration of a game*. Formulated in this way the demand implies the existence of enough computing capacity to do so. This makes the demand for *have sufficient on board computing capacity for implementation of game winning behaviours* redundant. However it expresses the knowledge of the designer that some of the actions in the robot soccer game may require intense computations and that these must not take too long to complete. Therefore it seems that this latter demand is ill formulated as it puts a vague

constraint on the robot's capabilities of executing behaviours that has an effect on many, if not all, functions. Maybe the demand should emphasise that the robot has to react quickly to sensory input to win a game. This would be a complement instead of a redundancy to the demand for the ability of playing a soccer game.

Intuitively as designers we know of the key role of the main processor has in determining the capabilities of the robot. However so far we have not been able to *put the finger* on the issue. After further analysis we arrived at a different formulation of the demand that still reflects our intuitive concern about processor performance while allowing an operational evaluation of it. The new formulation is that:

the delay of the response to perceptions, the reaction time, must be less than a specified value.

What that value is can be estimated, for example, on the maximum allowed displacement of the ball during the reaction delay. This demand is still broad in the sense that it leaves open the technical means of generating the reaction. It could be generated by a single high performance conventional von Neumann sequential processor or by several less capable processors or by a distributed network of many low cost processors. In the case of a single central processor an estimate of the processing speed can be obtained by making use of the simple model for the program execution time that says that the total execution time is the number of instructions N times the average number of processor clock cycles per instruction i times the duration of a clock cycle c.

$$T = NIc \tag{24}$$

N is mainly determined by the task, i and c are processor characteristics. Thus in our first approximation the demand can be quantified

$$NIc < \delta_{max} \tag{25}$$

The maximum delay can be estimated from the demand that the robot can move with a speed of up to 1 m/s. If we allow a maximum displacement of the robot of 5 cm before responding to a sensory stimulus then maximum delay will be 0.05 seconds (50 ms). The robot's most complex response must be generated within these 50 milliseconds. To translate this into processor specification we first need to estimate the number N of instructions to be executed for the response in order to obtain

$$Ic < \frac{\delta_{max}}{N} \tag{26}$$

For a vision system this requires capturing a new image at least every 50 ms, which means there must be a sustained frame rate of not less than 20 frames/second.

At this point a first DeCoDe design cycle is the next refinement cycle could be started. Stepping through a full cycle illustrates the application of DeCoDe from the initial rough concept and how DeCoDe reveals weak points in the design and forces the designer to address these points at an early stage.

VI. Conclusion

Products and services that fail to meet stakeholder demands will quickly fall prey to competitors in the global market place. The fate of a product is largely decided during its design phase. Design is a cognitive activity that is not well understood. The consequence is that design activity so far has been loosely structured and some demands on the artifact subject of the design may be neglected or are addressed too late in the design cycle causing expensive redesigns or an unsatisfactory product. The main goal of the DeCoDe design method is to conduct the design activity in such a way that it leads to a demand compliant product. This is achieved mapping an initial set of demands to three different but complementary views of the product: the functional view, the structural (component) view and the process view. The main tool of the Decode method are matrices that capture in a clear and easy to manipulate way the multitude of relations between these elements. Discovering and recording these relations, and keeping them consistent as the design progresses assures that all functions, components and processes in the artifact are justified by the demands and that there are no demands left that are not served by some functions, components and processes. The DeCoDe method can be adapted to designs of various levels of maturity as we have illustrated by discussing three main application modes with examples.

The measure of success for a design is the degree to which the designed artifact meets the stated demands. This applies to design methods as well, as they are also artifacts product of a purposeful design. Based on are review of some of the most widely used design methods and observations form the practice of design for quality we formulate a set of demands that a design method should meet. A design method should not only assure that the designed artifact meets the demands but the method needs to meet other demands as well. There are demands on design such as clearly structured and complete documentation, backward traceability of the design evolution, ease of communication the design to design team members and third parties.

Assessment of the degree of satisfaction of the demands allows the evaluation and comparison of the DeCode method with other design methods. By design, DeCoDe meets all the demands stated in section III to a high degree, except the first demand for which it relies on Requirement Engineering techniques.

The DeCoDe method has been applied to industrial and advanced engineering design. DeCoDe was chosen by the design team in the KitVes project [29], after conducting a survey of design methods, because of the ease of communicating the design, creating a common language, and understanding of the key design issues by designers for multiple disciplines. KitVes is an European Union project aiming at harvesting high altitude wind power by maritime vessel for on board usage. In the Promesys project [30] aimed at determining the reliability of mechatronic systems DeCoDe was used for obtaining the reliability data for a particular

drive mechanisms. By describing drive mechanisms in use in different industrial application using the DeCoDe views, components that fulfilled similar demands as the target design were identified. Actual reliability data of these deployed components could be used for estimating the reliability of the target design, instead of having to rely on estimated probabilities.

When initially proposed the DeCoDe method seemed to require too much extra work from the designer. An early application of the method in an industrial problem [20] suggested that it may not be necessary to carry out the method in its full detail to obtain benefit of the time savings obtained by designing a better quality from the start. In such cases it can be applied with the help of commonplace software tools like spreadsheets and matrix manipulation packages.

REFERENCES

[1] S. T. March and G. F. Smith, "Design and natural science research on information technology," *Decision Support Systems*, vol. 15, no. 4, pp. 251 – 266, 1995.

[2] G. Pahl, W. Beitz, J. Feldhusen, and K. H. Grote, *Engineering Design, A Systematic Approach*, 3rd ed. Berlin, Germany: Springer, 2007.

[3] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75 – 105, Mar. 2004.

[4] H. A. Simon, *The Sciences of the Artificial*. MIT Press, 1996.

[5] J. Sitte and P. Winzer, "Mastering complexity in robot design," in *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2004*. Sendai, Japan: IEEE, Sep. 2004, p. 1815 1819.

[6] A. Lex, J. Sitte, and P. Winzer, "Generic managements design - a method of collecting knowledge systematically during the developing process," in *Proceedings of the 8th International Symposium on Measurement and Quality Control in Production IMEKO ISMQC 2004*, Erlangen, Germany, Oct. 2004, CD Proceedings.

[7] J. Sitte and P. Winzer, "Demandcompliant design of robotic systems," in *Proceedings of the 2005 International Conference on Mechatronics and Automation ICMA 2005)*. Niagara Falls, Canada: IEEE, Jul. 2005, pp. 1953 – 1958, cD proceedings.

[8] ——, "Quality driven enterprise control," in *Proceedings of the 8th IFAC Symposium on Information Control Problems in Manufacturing (INCOM"95)*, Beijing, Oct. 1995, pp. 345–350.

[9] M. Berkovich, S. Esch, J. M. Leimeister, and H. Ktcmar, "Requirements engineering for hybrid products as bundles of hardware, software and service elements - a literature review," in *Proceedings of the 9th International Conference on Business Informatics*, Vienna, Austria, 2009, pp. 727 – 736.

[10] U. Lindemnn, M. Maurer, and T. Braun, *Structural Complexity Management*. Berlin: Springer Verlag, 2009.

[11] S. Schlund, F. Riekhof, and P. Winzer, "Probleme bei der entwicklung mechatronischer systeme ergebnisse einer industriebefragung," *ZWF-Zeitschrift fr wirtschaftlichen Fabrikbetrie*, no. 01-02, pp. 54–59, 2009.

[12] G. Pahl, W. Beitz, J. Feldhusen, and K. H. Grote, *Konstruktionslehre. Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung*, 7th ed. Berlin, Germany: Springer, 2006.

[13] D. D. Gajski and K. R. H., "New VLSI tools," *IEEE Computer*, vol. 16, no. 12, p. 11 14, Dec. 1983.

[14] I. Somerville, *Software Engineering*, 8th ed. Pearson Education, 2007.

[15] "Kooperationsorientiertes Innovationsmanagement. ergebnisse des BMBF-Verbundprojektes GINA: Ganzheitliche Innovationsprozesse in modularen Unternehmensnetzwerken," H.-J. Franke, C. Herrmann, B. Huch, and S. Loeffler, Eds., 2005.

[16] J. Gausemeier, U. Lindemann, G. Reinhart, and H.-P. Wiendahl, in *Kooperatives Produktengineering. Ein neues Selbstverstaendnis des ingenieurmaessigen Wirkens*, ser. HNI-Verlagsschriftenreihe, J. Gausemeier, Ed. Paderborn, Germany: Heinz Nixdorf Institut, 2000, vol. 79.

[17] S. Schuette, J. Eklund, J. R. C. Axelsson, and M. Nagamachi, "Concepts, methods and tools in kansei engineering," *Theoretical Issues in Ergonomics Science*, vol. 5, no. 3, pp. 214–232, 2004.

[18] J. Sitte and P. Winzer, "Systematic design of complex artifacts: Robot vision," *International Journal of Information Acquisition*, vol. 5, no. 1, pp. 51 – 63, 2008.

[19] ——, "Methodic design of robot vision systems," in *Proceedings of the 2007 International Conference on Mechatronics and Automation ICMA 2007)*. Harbin, China: IEEE, Aug. 2007, pp. 1758 – 1763, cD proceedings.

[20] ——, "Evaluation of a new complex system design method on a mechatronic automotive product," in *Proceedings of the 2006 IEEE International Engineering Management Conference (IEMC 2006)*. Salvador, Brasil: IEEE, Sep. 2006, p. 279 282.

[21] Y. Akao and M. G. H., "The leading edge in QFD:past, present and future," *International Journal of Quality & Reliability Management*, vol. 20, no. 1, pp. 20–35, 2003.

[22] N. P. Suh, *Axiomatic Design: Advances an Applications*. New York: Oxford University Press, 2001.

[23] M. Kauppinen, S. Kujala, T. Aaltio, and L. Lehtola, "Introducing requirements engineering: How to make a cultural change happen in practice," in *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE02)*, 2002.

[24] I. Sommerville, "Integrated requirements engineering: A tutorial," *IEEE Software*, vol. 22, no. 1, pp. 16–23, 2005.

[25] I. Graham, *Requirements engineering and rapid development: an object-oriented approach*. Harlow, England: Addison-Wesley, 1998.

[26] D. Franzotti Togneri, R. Almeida Falbo, and C. S. Menezes, "Supporting cooperative requirements engineering with an automated tool," in *Proceedings of the Vth Workshop on Requirements Engineering*, Valencia, Spain, 2002.

[27] A. Lex, "Methodik zum Anforderungsgerechten Roboter Design," Master's thesis, Bergische Universitaet Wuppertal, Wuppertal, Germany, 2003.

[28] M. Gries, "Methods for evaluating and covering the design space during early design development," Electronics Research Lab, University of California at Berkeley, Tech. Rep. UCB/ERL M03/32, Aug. 2003. [Online]. Available: http://www.gigascale.org/pubs/385.html

[29] E. Commission, "Kitves," http://www.kitves.com, 2009.

[30] BMBF, "Prozesskettenorientiertes regelkreismodell fr ein nachhaltiges robustes design mechatronischer systeme," http://www.promesys.org, 2009.