Product descriptions as Programs: a Case Study.[1]

Kai A. Olsen
Molde University College and University of Bergen, Norway
kai.olsen@himolde.no

Abstract

The company in this case study manufactures furniture for ships. This business sector is characterized by a high degree of customization. One way of handling the various products is to have a separate product description for each variant. Experience shows that this results in a large number of different product descriptions, in one company as many as 40,000. As the number of different descriptions increases, it becomes easier to create a new description than it is to search for an existing description, thus exacerbating the problem. Therefore, the company in the case study placed great importance on keeping the product descriptions to a minimum, but without limiting the variability.

We were able to offer a solution. Based on previous research, we have implemented a system for the company whereby product structures are described by means of computer programs instead of using the traditional bill of material structure. This method of defining generic product descriptions makes it possible to handle variability without increasing the number of descriptions. In theory, this highly structured method should offer a better solution if the users manage to program the product descriptions. The idea of this research was to see if the more complex solution also worked in practice.

Keywords: bill of material (BOM), generic bill of material (GBOM), programming language, usability.

1. Introduction

To keep track of all components that go into making a product, a bill of material (BOM) structure is normally used (Mukhopadhyay, 2015; Stephens and Meyers, 2013). These are usually implemented in the form of a hierarchical table, where each line lists the subcomponent that “goes into” a product. By using a tree structure, one may also build layers of components; for example, a drawer may be a part of a cabinet, while the bottom and sides are parts of the drawer. To add a degree of flexibility, and to create a more generic structure (GBOM), one may add conditions to each ‘goes into’ relationship, often in the form of if-statements built up with Boolean operators, that is, the relationship only holds if the logical expression is true.

Today, these traditional methods will often act as a barrier. In a modern production line, we find very flexible, programmable machines that can easily change from producing one variant to another, often automatically. The marketing department and customers both want to take advantage of this flexibility. To be competitive, marketing must offer many variants of a product, and customers will exploit these possibilities by demanding their own personal variation. This customization demands more flexible tools than the old BOM structure (Shu, Chen, Wang, and Lai, 2014.

---

[1] *This paper was presented at the NIK-2015 conference. For more information, see //www.nik.no/.*

In previous papers (Olsen and Sætre, 1998; Olsen and Sætre, 2007b; Olsen, Sætre, and Thorstenson, 1997), we have presented a novel and alternative method for describing the component structure of products than the more traditional BOM approach. Instead of a simple tabular structure, with or without conditions, we describe each product by way of a program using a special purpose programming language. This 'products descriptions as programs' method seems to be original in the sense that we have not found any references to similar research. The most similar approach is that of product configurators, but these are generic systems made for one type of product, for example, a particular car model. The difference is that we leave the task of programming the product descriptions to the user.

We would have liked to provide further references to the programming method, but we have not found any in the literature. With our computer science (CS) background, perhaps we see this from a different perspective than most researchers within the field of production management.

Until now, the programming method has been theoretical, but now we have had the opportunity to implement the system in a real life situation as the main information technology (IT) system for ShipNor AS, a company making furniture for ships. In many ways, ShipNor represents the ideal case. Space onboard a ship is at a premium. While ship designers try to construct cabins of standard sizes, many of these are constrained by the structure of the ship. For the furniture company, this means that their collection of standard furniture must be adjusted to fit in each cabin. For example, if a desk is put next to a bed (bunk) one may be able to connect the desk to the bed, saving table legs at one end. If the bed is placed in a corner, one may avoid a headboard for the bed. If a wall (bulkhead) of a cabin is formed by the side of the ship, a table standing next to the wall must be adjusted so that it fits. This saves space, and reduces cost and weight. Limiting weight, especially of combustible items, is also important on a ship.

Our task, therefore, is to implement a system that puts as few restrictions as possible on the number of variants without extending the number of product descriptions that are needed beyond a minimum. Indeed, ShipNor was founded some years ago by 10 employees who had left another furniture company. This former company, which had been in the business for many years, had as many as 40,000 product descriptions, each described within a traditional BOM structure. In the current company, we aim to limit the number of basic furniture descriptions to just 40. Of course, 40 are much easier to maintain, whereas 40,000 present an impossible task.

Before describing the tools that we have developed, we will describe our research methodology, the company and its challenges. While maintaining the principle of describing products by way of programs, our tool design was modified for the real world situation. Thus, the programming language and tool that we describe here is somewhat different from the prototype described in Olsen and Sætre (2007b).

2. Research Methodology

Traditional research methods seldom allow the researchers to go deeply into a case. Most often one has the role of a bystander, looking on the activities and trying to analyze and interpret them. In our case, the researcher had the task of developing the system on behalf of the company. The effort was partly funded as a research project, but mainly paid for by the company. The advantage for the researchers was that they were able to test their theories; the advantage for the company was that it provided them with the possibility of implementing a product line that allowed for extensive customization.

2.1 Action Research

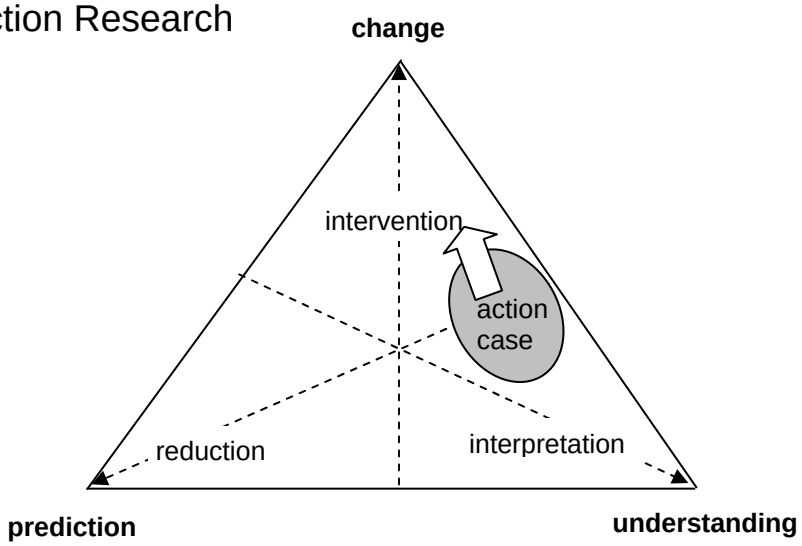

Figure 1. Research framework

The idea of action research is to let "researchers try out their theories with practitioners in real situations and real organizations." (Avison, 1999) Our method of investigation can be described as *action case* (Baskerville and Pries-Heje, 2015; Braa and Vidgen, 1999). Figure 1 shows that *action case* is placed in a landscape described by the terms change, prediction and understanding. As Braa and Vidgen describe, prediction is the aim of the systematic reduction of a positivist approach, understanding is the aim of an interpretive approach, while change is associated with an interventionary approach. Its position in the diagram shows that *action case* has strong change and understanding perspectives.

In our case, the change component may be even stronger, that is, in Figure 1 the *action case* element should be moved in the direction of the arrow. The goal of the company is to develop a system that will allow them to handle all customer demands for variation at the same time as maintaining an efficient production line, and limiting the number of product descriptions. The researcher's goal is to test ideas and tools in a real life situation.

The research performed here has much in common with what has been labeled 'participative action research' (Whyte, 1991). This is based on the idea that theory can be built on practice (Argyris and Schön, 1991; Checkland, 1991; Hult and Lennung, 1980). Of course, much of our scientific base is built up in this way, whereby creative practitioners find new and more efficient ways of performing daily tasks.

While there are huge benefits of getting involved in a real case, there may be a question of whether one is successful in reporting objectively (Hirchheim, Klein, and Lyytinen, 1996; Susman and Evered, 1978). The development of the system described here is based on our own research ideas, and there is always the danger that this may conflict with the need for an objective evaluation. However, this is a problem that all researchers with an agenda have to face up to. We are always involved, as we should be, in the research that we perform.

In our case, information was collected by participant observation, by individual interviews, and by discussions with personnel at all levels – with operators, foremen and managers. Many of the ideas that were incorporated into the tool came from company employees. While the researcher had control over the basic ideas and was responsible for the programming, the detailed specifications often came from employees. These were more in the form of "we need a report such as this", "these special components are first given edges, later cut into parts", and so on; in other words, requirements for functionality that would make their work easier.

Susman and Evered (1978) described action research as a five-phase cyclical and iterative process: (1) diagnosis, (2) action planning, (3) action taking, (4) evaluating, and (5) learning. We adhered to these phases in this project.

2.2 Research Questions

DECK	ROOM	Order no:	Item no	FURNITURE DESCRIPTION	COMMENTS	PCS	PRICE	TOT. PRICE	Finish
Bridge Deck	Emergency Bridge	1483	1	Bookshelf, Floorstanding, 2 shelves in height	W:600 H:892 D:350	4	1658	6632	F5150
Bridge Deck	Emergency Bridge	1547	1	Table Top, Edge Front/Sides	L:2440 W:370	1	589	589	F5150
Bridge Deck	Emergency Bridge	1541	2	Table Top, Edging All sides	L:1800 W:1200	1	1708	1708	F5150
Bridge Deck	Emergency Bridge	1560	2	Table Leg	OPAL72C	2	2229	4458	SATIN
Bridge Deck	Operation Plan.Are	1563	3	Binoccular Box, Plexi		2	562	1124	
Bridge Deck	Operation Plan.Are	1483	4	Bookshelf, Floorstanding, 5 shelves in height	W:600 H:2034 D:350	4	2857	11428	F5150
Bridge Deck	Operation Plan.Are	1525	5	Radiostation 1, special	L:3147/4722/3147 W:1120 H:1200	1	57376	57376	F7912/F5150
Bridge Deck	Operation Plan.Are	1559	5	Table Leg	H:730	3	100	300	GREY
Bridge Deck	Pantry	1557	6	Table Top, Corner, Edging Front/Right, incl.Vent Grid	L:5195/2630 W:600/600	1	3445	3445	F5150
Bridge Deck	Pantry	1557	6	Cupboard Section, doors: 5x 500 / 7x 600 + infill, vent g	H:890 D:590, See DWG	1	23006	23006	F5150

Figure 2. Traditional approach - products as spreadsheets

The traditional method of solving the large number of variants that are needed in this business is to have one description for each variant. Each description can be as simple as describing the components that go into the product by means of a spreadsheet (Figure 2). Some manufacturers also use simple systems that handle the BOMs, but in essence these also operate with component lists. The advantage of this approach is simplicity and flexibility – the product is just a list of components. Any type of furniture can be specified. The disadvantage is that there is a large number of different product descriptions, so high that the users lose the ability to see the overview. It was often easier to create a new product description than to search for an existing description that could be used. This was also detrimental to best practices since experiences that were incorporated into earlier product descriptions may have been lost. Another drawback is that there was a greater probability of formulating erroneous descriptions. In addition, since the basic building block may be a low-level component, data entry took time.

```
L:=£length-(16.4*2)
if  L  > $maxBoardLength then
  !del opp
  M:= L \ $maxBoardLength
  rest := L  - ($maxBoardLength * M)
  if rest > 0 then
    If rest < 300 Then
      include base(300;90;colorOver;fireRetardent) as cupboardBasePart
      include base($maxBoardLength-300+rest;90;colorOver;fireRetardent) as cupboardBasePart
      M:=M-1
    else
      include base(rest;90;colorOver;fireRetardent) as cupboardBasePart
    end If
  end if
  !legg inn hele lengder
  For i := 1 To M
    include base($maxBoardLength;90;colorOver;fireRetardent) as cupboardBasePart
  Next i
else
  include base(L;90;colorOver;fireRetardent) as cupboardbase
end if
```

Figure 3. Product descriptions as programs

While our approach of describing products by way of programs can reduce the number of product descriptions to a minimum, it is clearly more complex (Figure 3). While this method offers the flexibility of a programming language, it is clearly more formalized than the spreadsheet method. On the other hand, the programming method offers the possibility of

fine-tuning product descriptions, and of incorporating best practices. It is also possible to simplify the data entry part, that is, the part where products are inserted into a customer's order.

Table 1. Comparing the traditional (spreadsheet) method with the programming approach

	Traditional	Product descriptions by way of programs
Flexibility	Very high	High
Simple	Yes	No
No. of product descriptions	Very high	Very low
Supports best practices	No	Yes
Data entry	Extensive	Efficient
Probability of errors in the descriptions	High	Low

Table 1 evaluates the differences between the traditional spreadsheet method and the programming approach. The users have high expectations of getting a system that can reduce the number of product descriptions. However, the basic question is: Will they be able to program the furniture descriptions, or will this prove to be too complex?

The best way of answering this question is to offer them a system where this is possible. Experience with this system for real world problems will determine whether or not the programming approach offers the flexibility needed to include the description of existing products, new products, and new production processes.

3. The Company and its Products

Figure 4. A cabin.

ShipNor AS was founded in 2013. The objective of the business is to produce furniture for ships. This includes bunks, tables, cupboards, shelves, and so on (Figure 4). Modern vessels such as those that support oil exploration and production at sea may have several hundred cabins. In addition, there may be a large number of offices. Furniture for ships requires a better quality than standard furniture. Due to the fact that ships move, drawers need a locking system so that they remain closed in heavy seas, tables need to be bolted together, bunks need guards, and tables may need sea rails. However, the main difference is in the customization.

Figure 5. "Simple" tables.

For example, a ‘simple table’ (see Figure 5) can be round, rounded, oval or rectangular. It can be a stand-alone table, connected to other tables, or for example a corner table. Thickness, length, width, top color, edge color, and the type of corners are all specified by the customer. Edges are dependent on position. For example, a table in a corner will not have edging on the parts that connect to the walls. If the table is stand-alone, then it may have four legs in a type and color specified by the customer. However, legs may not be necessary where there is a drawer section, or where the table is connected to a wall or to another piece of furniture.

Of course, there are more complex furniture items than a ‘simple table’. Bunks may come as single or double, may have end parts or guards, and may have drawers underneath – of different sizes, and positioned according to a customer’s specifications. Cupboards come in different sizes; some have doors, others have open shelves or drawers. If the cupboard is used to house a fridge, air vents will be needed in the door and in the table top above. Several cupboards may be put together in a row with a common top or bottom.

The furniture is made out of plywood, with laminate on one or two sides. Both plywood and laminate come in sheets of standard sizes, approximately 1.2 meters × 2.4 meters. If the customer specifies furniture, for example tables that are larger than a single sheet, two or more sheets must be connected.

Usually the production line starts with gluing laminate onto the plywood, then cutting the pieces. The cutting is performed by a Computer Numerical Control (CNC) machine that can optimize the use of plywood. After cutting, the pieces are moved to a machine that grinds the edges and corners, and which can also add PVC strips for the required color edging. All drill holes, for example those for hinges, are made by this machine. The final parts are then moved to the assembly area where the furniture is put together. Large furniture, such as wardrobes, are not assembled, but sides, tops and doors are packed flat, one on top of the other.

An order may consist of many hundreds of pieces of furniture and thousands of components, so production will usually start with one type of furniture, often beds. The batch of furniture selected is given a production number that follows the furniture and components through each stage of the production. Since beds and bookshelves use the same type of sheets of plywood and laminate (the same thickness, often a similar color, and the bookshelf will usually fit into the small piece of sheet left after producing a bed), the production manager may choose to produce these under the same production number, knowing that this will offer a less wasteful use of the sheets.

For large orders, the production manager may want to produce some parts first. For example, most beds have drawers. These can be made before the beds. The drawers are placed in storage, and retrieved from there when the beds are produced.

Until recently, the traditional way of making furniture has been to assemble the parts of drawers, cupboards, shelves, and so on, using screws, dowel pins and glue. The company is now implementing a new patented method of 'clicking' pieces together. This is the same method that is used for flooring, but now extended to furniture. The advantage is great savings in assembly costs. Furthermore, the click-assembly is so simple that it can be performed by anyone. Using this method, it now becomes possible to postpone the assembly process until the flat-packed components are moved to each cabin. This saves freight and storage space, and reduces the risk of damage to the furniture in transit.

Click furniture will need new descriptions since different components are needed, and some measurements will have to be changed (to offer space for the profiling that the click system requires).

4. The System (shipIT)

The task for the researcher is to develop a system that:

1) Offers the possibility of describing any type of furniture in any variant.
2) Limits the number of product descriptions.
3) Can ensure an efficient production line.
4) Provides all data needed for administrative reports and documentation.

We will fulfill requirements 1) and 2) by describing products by way of programs. This will offer the necessary flexibility to describe any variant of any type of furniture. With clear specifications of every piece of furniture, requirements 3) and 4) will be easy to fulfill.

Note that this system is developed à propos our idea of in-house programming (Olsen, 2009; Olsen and Sætre, 2007a), that is, that software for niche companies should be developed and owned by the company. This offers both competiveness and control.

4.1 Describing Products by Way of Programs

Furniture is described by a set of attributes and a program. We will introduce these concepts with an example.

Egenskap	Vis til kunde	Tillatte verdier (f.eks.; 40;60;100)	Standardverdi
length	☑		300
colorOver	☑	1040;1040AR+;1614;1834;2001;2253;5147;5150;5373;	$colorOver
colorPVC	☑	1040;1040AR+;1614;1834;2001;2253;5147;5150;5373;	$colorPVC
seaRail	☑	with; without	without
fireRetardent	☑	true;false	false

Figure 6. Attributes for a triangular table

The snapshot in Figure 6 presents the attributes for a triangular table: its name, whether this attribute should be included in the scope of a delivery report to the customer (a checkbox), the range of acceptable values, and a default value. The default value may be an explicit value (for example, 300) or a project variable (for example, $colorOver). Project variables may give a value for the whole ship, for a single deck, or for a specific cabin. For example, $colorOver and $colorPVC for the ship's hospital will usually be set to white. The idea is to use default values as often as possible, to avoid unnecessary specifications.

```
L:=sqr(2*length*length)
!merk at hjørner og kanter kan bli endret når trekantbord slås sammen for saging og kanting
include PoplarBasis(28;L+20;L/2+10;colorOver;colorOver;colorOver;0;0;0;0;'with';'without';'without';'without';fireRetardent) as me
include mountingProfile(length-50;0) as mounting
include mountingProfile(length-70;0) as mounting
if searail then
  include searail(L) as searail
end if
```

Figure 7. Program for a triangular table

The program describing the triangular table is presented in Figure 7. As can be seen, the language uses arithmetic expressions, if-statements, and an include statement. The latter may refer to another article described in the system (for example, PoplarBasis – a type of plywood). or to an end-component (for example, searail). End-components are ordered from suppliers, and require no further breakdown.

£length	☑	computed	
£width	☑	computed	
depth	☑	589.5;390	589.5
height	☑	570; 730; 890	730
colorOver	☑	1040;1040AR+;1614;1834;2001;2253;5147;5150;5373;	$colorOver
colorUnder	☑	1040;1040AR+;1614;1834;2001;2253;5147;5150;5373;	$colorunder
colorPVC	☑	1040;1040AR+;1614;1834;2001;2253;5147;5150;5373;	$colorPVC
backwall	☑	0; 6; 15	0
position	☑	left;right;middle;between;	middle
N	☑		1

Figure 8. Attributes for a generic cabinet (extract).

Cabinets (cupboards) can be ordered in rows. Each part will have its own specification, but there are also shared parts. These are described through the attribute set in the example shown in Figure 8. Note the £-attributes. These will be computed by the system. The N-attribute will specify the number of cabinets in a row. A part of the program for describing generic cabinets was shown earlier in Figure 3. This part will compute the length of the common base for the N cabinets. If the length is greater than the maximum length of a sheet, then pieces will have to be joined together (computed in the for-loop).

4.2 Inserting Furniture for a Customer's Order

A customer's order will originate as a drawing of each deck of the vessel, with symbols representing tables, bunks, chairs, and so on. ShipNor will then enter the furniture specifications for each cabin into shipIT.

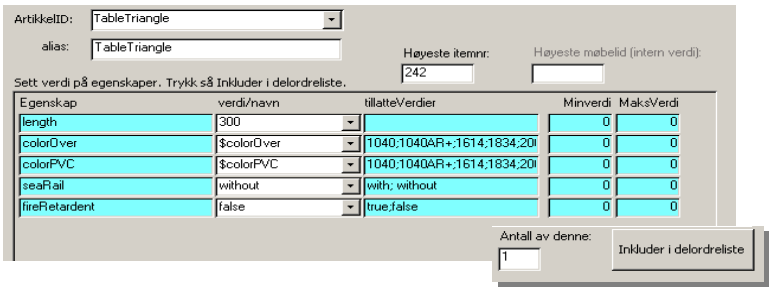

Figure 9. Specifications of a triangular table.

An example is presented in Figure 9, which shows the specifications of a triangular table. Usually default values are applied. If the number of items required is one, the item may be inserted with a single click on the include-button (‘Inkluder’).

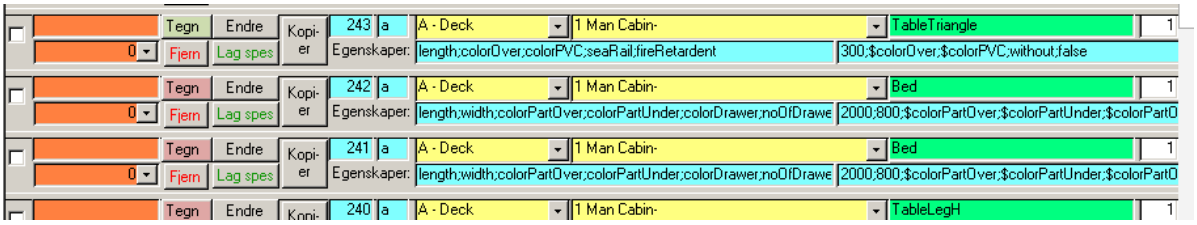

Figure 10. An example of a furniture list for a specific order.

The furniture included in an order is presented as a list. An example is shown in Figure 10. If several cabins have similar furniture, it is quite simple to copy all specifications from one cabin to the others, or from one deck to another. These specifications may later be changed if necessary. The idea is to specify an order with the minimum of input.

4.3 Data for Production

The production manager can at any time select the furniture that is to be produced, and ask the system to present the necessary data for the different workstations. The system will then execute the programs for each selected piece of furniture, using the attribute values as input.

Standard kappliste for Testprosjekt **Pordrenr: 0**

artikkelID	Komponentnavn:	ant.	T	ply	L	W/H/D	Col.O	Under	PVC	Verdier
TableTriangle		**1**	0		300	0	5373	any	5373	
PoplarBasis	TableTriangle	**1**	28		444	222	5373	5373	5373	0;0;0;0;+;-;-;-
mountingProfile	mounting	**1**	0		250	0				
mountingProfile	mounting	**1**	0		230	0				

Figure 11. A material list (example)

As an example, the material list for a triangular table is shown in Figure 11. This standard list is used for the grinding and edging operations; other lists are produced for the cutting, and for assembly.

4.4 Other Functions

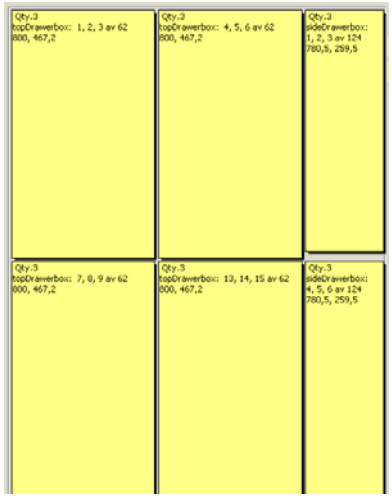

Figure 12. Utilizing sheets.

Once specifications of all furniture are completed, it is then possible to offer a large set of new functions. An example is drawings. Formerly these were produced manually using Autocad. Now these can be produced automatically.

The system can also compute the number of laminate and plywood plates needed for a complete project by using an optimization algorithm. This is especially important if the laminate specified by the customer is of a non-regular type. Special laminate takes time to order, and there can be additional costs if too few or too many are ordered. An example of the output from this system is shown in Figure 12. This example suggests a layout for cutting a set of components from a sheet, ensuring that 98 percent of the sheet is used.

5. The Research Project

We followed the five-phase method described by Susman and Evered (1978). Each phase is discussed below.

5.1 Diagnosis

The employees of ShipNor, who are also the owners, contacted us to see if we had any solution to the variant problem. We suggested using our previous research results. This offered a win-win solution. ShipNor was given the opportunity to tackle this problem in a novel way, hopefully one much better than that used by their competitors, and we had an ideal case to try out our research ideas in practice.

5.2 Action Planning

We started by arranging a set of workshops with users at all levels, from the manager to the people on the factory floor. One problem was enabling them to envisage what the system could do. Therefore, from the outset we decided to develop a prototype system, and to use this for actual data. In the beginning the system would be used in parallel with the traditional spreadsheets method. Thus we were able to reduce risk, and at the same time obtained comparative results.

5.3 Action Taking

The system was developed over a period of six months. It was then tested on actual data. We learned that most of the products could be described through the programming language. However, we continually had to improve the functionality.

From the beginning we knew that some furniture could not be described in a simple manner. One example is special purpose furniture for the bridge. After testing various solutions, we implemented a ‘special’ furniture type. This could, for example, be as simple as adding all components, thus replicating the spreadsheet. However, often a special variant would also be a variant of another piece of furniture. In this case one could create a ‘special’ based on this furniture description, and then edit the component list.

The very first product descriptions were programmed by the system developer. Later on we found that users themselves could do this. In many cases they also managed to correct errors in the descriptions made by the developer, which was a good sign.

5.4 Evaluation

After approximately six months of testing we had a fairly complete system, where all the furniture was described by programs. The completeness of the programming language was tested when the company received an order from a large shipyard that had their own design. In a very short time, the users were able to do the modifications that were needed. No additional changes had to be performed to the programming language.

The company is now introducing a new method for joining components together, the ‘click’ method described above. This requires some changes in the production phase; for example, some components can be worked on before they are cut into individual pieces. We managed to describe this by introducing a new function, a ‘collect-verb’ that told the system to join components for preprocessing.

We have not performed any formal evaluation of the system. However, it has replaced the traditional methods, and is in full use today. In other words, the system has survived in the real world. The users are very clear that they feel that their company has an advantage in the marketplace because of this system. One advantage is that the company is able to automatically produce extensive documentation for the customer at an early stage, including complete component lists, drawings, estimates of volume and weight, and so on.

5.5 Learning

We found that the users mastered the system, that is, they learned how to program the furniture descriptions. We use a plural here, but in fact everything relied on one user. She has only basic administrative education, but has a long experience in the furniture business. She also grasped the concept of programming early on, and is now teaching her colleagues.

There is a danger of action research, perhaps not so much that one may report too positive answers, but that many things can go wrong in the real world. The basic ideas may be sound, but the system may still fail. In our case we see that the positive result was really dependent on this one user.

6. Discussion

The system described here, shipIT, has been in use for nearly a year, the first few months as a pilot. Since then, there have been additions and changes to the furniture collection, each of which was easy to implement through the ‘programming’ method. All in all, this seems to be practical proof that the methodology offers the necessary flexibility. As stated above, furniture descriptions and modifications are wholly maintained by company personnel.

While the fundamental ideas were well understood before we started developing the system, other requirements became apparent after completing the first version of the system. This was in many ways due to the startup nature of the company, and that production procedures had not been finalized when system development began. However, most changes were easy to implement, and did not require any large modifications.

The original plan had been to describe everything through the ‘programming’ method. We found that this was not practical. In order to ensure that the specifications for complex furniture, such as cabinets and office landscapes (large offices with many connected desks), could be described in an efficient manner, we designed a special functionality in shipIT to handle these. The drawback is that fundamental changes in these types of furniture may require a change in the shipIT code.

The main advantage with the system, according to its users, is that the number of product descriptions is so limited that it is easy to get a complete overview. Also, if one adds a modification to a description, for example to facilitate production of a particular item, the system will remember it. That is, with few furniture descriptions it is advantageous to use more time on each of these.

With the limited number of product descriptions in shipIT, it also becomes feasible to add more details for each piece of furniture. That is, with fewer descriptions one can spend more time in formulating good descriptions. For example, the number of packages required to transport the parts is now included, as well as the volume of each part. Using these data, the system can compute the total weight of all items, the number of packages that are needed, and the container sizes required for shipping.

The system automatically computes the production time necessary for each piece of furniture, also taking variants into account. The next step of calculating the cost of producing the

furniture and giving a quote, is a simple one. Since we also record the actual time needed to produce each piece of furniture, we have the possibility to make an adaption to the system to adjust production times according to experience. This provides shipIT with all the data needed to make detailed production plans. In addition, shipIT can implement all standard functions required to keep an inventory of stock, and for its procurement.

Our aim in this project was to test out methods described in earlier research in a real world situation. We soon discovered that this was not the aim of the users. Sure, they wanted a system that could handle variants, but they also wanted much more. Today therefore, the system also includes the functionality required for a complete business system: reports, stock handling, ordering materials, collecting and presenting data on hours used, planning, price and time evaluation of finished orders, and so on. However, with all the data available from the prototype system, producing a full system was an achievable objective.

7. Conclusion

In earlier research, we have described a method for handling flexible product descriptions. The idea is to describe product structures by way of programs and not as table-based BOM or GBOM structures. In this way, one can limit the number of product descriptions to a minimum.

In the project described here, we have implemented a system based on these ideas. This has been tried out in a real world case for a company producing furniture for ships, an example of a business that requires a high degree of customization of its products. Through this effort, we were able to put into practice the findings of our research; this enabled the company employees involved to describe any type of product in the system by way of a program, and designs originated with the customers. The highly generic approach of using programs allows us to include every possible variant into one product description, thus limiting the number of product descriptions to a minimum.

The system is today in full use in the company concerned, and has replaced all traditional methods used for describing products.

Acknowledgements

I would like to thank Otto Hammerø for initiating the shipIT development, and for allowing researchers to use ShipNor as a test case. Irene Thorsrudhagen Danielsen has participated on behalf of the company, and has been a very active pilot user, offering helpful advice through many discussions. Kari Tolaas has been helpful in defining specifications for the procurement and time registration parts of the project.

The first version of this paper was returned with a set of comprehensive reviews. I would like to thank the reviewers for their effort in helping me to improve the manuscript.

References

Argyris, C. and Schön, D.A. (1991) Participatory action research and action science compared. In W.F. Whyte (Ed.), *Participatory action research*, Newbury Park, CA: Sage.

Avison, D., Lau, F., Myers, M. and Nielsen, P.A. (1999) Action research, *Communications of the ACM*, 42, 1, 94-97.

Baskerville, R. and Pries-Heje, J. (2015) Projecting the Future for Design Science Research: An Action-Case Based Analysis, Lecture Notes in Computer Science, Springer Link, Volume 9073, 2015, 280-291.

Braa, K. and Vidgen, R. (1999) Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information systems research, *Accounting, Management & Information Technology*, 9, 25-47.

Checkland, P. (1991) From framework through experience to learning: the essential nature of action research. In Nissen, Klein and Hirchheim *Information Systems Research: contemporary approaches and emergent traditions*. Amsterdam: Elsevier.

Hirchheim, R.A., Klein, H.K. and Lyytinen, K. (1996) Exploring the intellectual structures of information systems development: a social action theoretical analysis. *Accounting, Management & Information Technology*, 6(1-2), 1-64.

Hult, M. and Lennung, S. (1980) Towards a definition of action research: a note and bibliography, Journal of Management studies, 17, 241-250.

Mukhopadhyay, S. K. (2015) Production Planning and Control: Text and Cases, Third Edition, PHI Learning Pvt. Ltd.

Olsen, K.A. (2009). In-house programming is not passé – automating originality, IEEE Computer, April edition.

Olsen, K.A. and Sætre, P (1998) Describing products as programs, *International Journal of Production Economics*, vol. 56, no 1.

Olsen, K.A.and Sætre, P.L (2007a). IT for niche companies: is an ERP system the solution? Information Systems Journal, Vol. 17 Issue 1, 37-58

Olsen, K.A. and Sætre, P.L. (2007b). ERP for SMEs – is proprietary software an alternative? Business Process Management Journal, 3, 13, 379-389

Olsen, K.A., Sætre, P. and Thorstenson, A. (1997) A Procedure-Oriented Generic Bill of Materials, *Computers & Industrial Engineering*, 32 (1), 29-45.

Shu, T., Chen, S., Wang, S. and Lai, K.K (2014) GBOM-oriented management of production disruption risk and optimization of supply chain construction, *Expert Systems with Applications*, Vol 41, No 1.

Stephens, M.P. and Meyers, F.E. (2013) Manufacturing Facilities Design and Material Handling, Purdue University Press.

Susman, G. and Evered, R. (1978) An assessment of the scientific merits of action research, *Administrative Science Quarterly*, 23, 582-603.

Whyte, W.F. (ed.) (1991) Participatory Action Research. Sage, Newbury Park, CA, USA.