Hierarchical Case Based Reasoning to Support Knitwear Design

P. Richards, A. Ekárt
School of Engineering and Applied Science, Aston University, Birmingham, B4 7ET, U.K.
{richardp, ekarta}@aston.ac.uk

Abstract
Knitwear design is a creative activity that is hard to automate using the computer. The production of the associated knitting pattern, however, is repetitive, time-consuming and error-prone, calling for automation. Our objectives are two-fold: to facilitate the design and to ease the burden of calculations and checks in pattern production. We conduct a feasibility study for applying case-based reasoning in knitwear design: we describe appropriate methods and show their application.

Keywords:
computer-aided design (CAD), case based reasoning (CBR), knitwear design, similarity, adaptation

1 INTRODUCTION

The design and production of patterns for hand knitting is a very tedious process, involving a highly qualified team of a designer, a pattern writer, several checkers and knitters, a typesetter and proofreaders. There is very little documentation and training material available, most of the skills are acquired through learning by doing. The designers and other members of the team find it difficult to formulate the rules which decide how they perform various steps, an outsider would have to watch them, try and understand what they are doing and then learn by asking questions.

The designer first designs the pattern and produces a hand-drawn sketch on graph paper, which includes measurements. The pattern writer then manually calculates the pattern for hand knitting in different sizes. The checkers perform an initial check of the pattern and make any necessary corrections. The knitters then hand knit the pattern in one size and make alterations to the pattern if needed (for example to obtain the desired shape of the garment). The alterations are made on the pattern for six different sizes. The checkers manually check that all the calculations are correct. For a proper check, they sometimes hand knit a sample of the complicated parts of the pattern (a finishing, for example) to make sure that the result corresponds to the description. In the next step, the typesetter typesets the pattern using a standard word processor. Finally, at least three proofreaders check the typeset pattern for any mistakes (they perform a read-through check). If there are alterations needed, the typesetter will make them on the typeset pattern.

The process is very time consuming and requires a lot of human attention and effort. Everything in the process, except the typesetting, is done manually, without the use of computing technology. We demonstrate here how specialised tools can be developed to help the designer in producing the sketch and also to automate the checks, calculation and production of the written patterns.

1.1 Knitwear Design

Although knitting and knitwear design has a long history and also includes many calculations and steps that are almost mechanical, there have been surprisingly few attempts to use computers for improving the underlying processes. The design of machine-produced knitwear was investigated by Eckert et al. [1]. Communication difficulties arise in large design teams since the specifications of the garments are often inaccurate, incomplete and inconsistent. To alleviate this, the authors proposed that designers use an intelligent CAD system. Active critiquing, i.e. pointing out differences between a designer's goal and what is actually happening, can be used to catch errors as early as possible. In the user interface which was presented, choices were only offered if they were relevant, based on the answers to previous questions. Shapes were described using shape grammars and Bézier curves. Interestingly, they say that "case based reasoning could be employed to provide starting measurements for cutting pattern construction".

A system for automatically designing knitting stitch patterns for hand knitwear was presented by Ekárt [2]. It is possible to produce many designs that are not knittable, because they violate the rules of knitting. For example, if a width of a garment does not change then the number of stitches in a row must be equal to that of the previous row. A method of representing knitwear based on trees is able to avoid many of these invalid designs. Some heuristic measures were described, which can

identify aesthetically pleasing patterns. This avoids the fatigue which is involved in the alternative techniques of asking humans to evaluate patterns which have been generated.

1.2 Relation to Other Design Areas

Although there is no particular focus on knitting in the scientific design literature, several lessons learned in other areas of design can be applied to knitwear design. Tomiyama et al [3] discuss how design failure occurs due to the coupling of design parameters in complex designs. Also, they point out that the complexity of multi-disciplinarity means that often no one person can understand a design. Knitting patterns are inherently complex and the objectives of a compact textual pattern, an aesthetically pleasing and fashionable garment, and technical feasibility can be conflicting. 'Destructive coupling' (as described in Tomiyama) can occur, for example a seemingly small change to a garment can result in a large increase in length of the pattern.

Urbanic et al [4] describe a design recovery framework which they structure into an axiomatic design matrix format as a method of reverse engineering. Comparing knitwear design to the engineering example shows key differences. For example, the design parameters and constraints in knitwear design are typically much "softer" than in engineering.

Many studies on creative design conclude that human designers create new designs by studying past designs from various resources and combining them together, adding new elements. To assist the design and production of knitting patterns, we propose a case based reasoning system that allows for structured step-by-step production of new patterns from scratch or reuse and adaptation of similar patterns from the past.

We introduced the general idea of using case-based reasoning (CBR) in [5, 6]. In this paper we detail the CBR approach to knitwear design including the building of the case base from scratch. We first describe the main concepts of CBR and its application to knitwear design including examples. We then provide the software architecture and implementation details.

2 CASE-BASED REASONING

Case-based reasoning is a generic problem solving methodology based on the idea that a solution to a new problem can be obtained from solutions to similar problems encountered in the past [7]. In addition to using knowledge from previously experienced problems, CBR has an element of incremental learning: every new experience is recorded for future use.

CBR is based on typical human problem solving: an engineer designs a new product or part by possibly reusing features of a past design, whose specification presents a certain degree of similarity with the expected new product; a doctor examining a patient builds up the diagnosis based on a comparison of the symptoms with those described in a book or previously shown by another patient; a decision in court is taken by drawing the similarity to a case in the past. The key in all these situations is to remember the similar problem case from the past and adapt its solution to the new situation.

The main cycle of CBR consists of four steps [7]:

- **retrieve** the most similar case(s) in the case base;
- **reuse** these case(s) to attempt to solve the new problem;
- **revise** the proposed solution;
- **retain** the new problem and its solution in a new case in the case base for later use.

There are various methods to represent cases, in most situations the problem description and the solution are represented as attribute-value pairs. A suitable similarity function has to be devised to compare the new problem case with the ones stored in the case base. The best matching cases are retrieved from the case-base and adapted for reuse. Adaptation is often very simple, such as slightly changing values of selected attributes, or left for the human to do. In domains where there is a huge number of past cases available, it is very likely to find a very similar past case whose solution needs little adjustment to the new case. However, where there are few past cases available only, the adaptation needs to be more substantial and prepared to deal with larger differences between past cases and the new case. It is customary to allow the user to rate how well the new solution proposed by the CBR system performed and possibly repair it in the revise phase. The new case is then retained for future use in the case base if it is judged to be sufficiently different from the existing cases.

Lopez De Mantaras et al. [8] comprehensively discuss retrieval in CBR, e.g. whether to use surface features or data which is derived from a more in-depth analysis of the case. Some of the methods they discuss are variants on nearest neighbour

algorithms, others rely on a complex representation of cases, e.g. in a hierarchy or as a graph. They mention systems where a subset of cases is retrieved and presented to the user; in some of these systems they sacrifice some of the similarity in order to introduce diversity. They also discuss “adaptation-guided retrieval”, which uses domain specific knowledge to reduce adaptation failures in situations where the most similar case is not necessarily the easiest to adapt.

Price and Pegler [9] describe the Wayland advisor for the setting up of die-casting machines. Wayland consists of a series of fields, each of which has a weight chosen to set the significance; these are summed to give an overall match value. Wayland is an uncommon example of the successful use of CBR in a difficult design problem. There is a large value space for the inputs, and the relationship between those inputs and the desired outputs is not clearly understood. Nevertheless the system was able to find a ‘close enough’ solution, and as a result was successfully deployed in foundries.

Kolodner [10] discusses indexing and flexible methods of organising cases into a network, e.g. a memory organisation packet. She gives four methods of computing similarity: using an abstraction hierarchy, a qualitative scale, a quantitative scale, or comparison of roles. The choice of representation and similarity measure depends on the nature of the data, e.g. whether it is hierarchical, discrete choices or continuous range.

Bergmann et. al [11] discuss case representation, pointing out that the choice of *case representation and similarity measure are related*. Straightforward representations are feature vectors, textual and object-oriented. Generalised cases are achieved by introducing variables into cases to represent solutions to a wide range of problems. Complex cases may involve a choice of variables and wide ranges for their values. The importance of generalised cases lies in the fact that they can enable a CBR system *to perform well with a small case base*. Bergmann et al. also discuss CBR systems with a hierarchical representation. Larry Leifer of Stanford University famously said “all design is redesign”. This applies to knitwear since at the lowest level garments are composed of the same types of stitch. Therefore, a partonomic hierarchy is conceivably a good way of representing knitwear. However, comparing the detail of complex cases in order to judge their similarity is much more difficult than simply matching on surface features.

Craw [12] describes k-NN (nearest neighbour) algorithms and makes the point that recording how many times a particular case has been reused can, in future, help to identify problems with the coverage of the case base.

Mejasson et al. [13] use a weighted sum algorithm in their intelligent design assistant system, which assists designers of submarine cables. The algorithm involves the sum of the squares of individual distances; the weights applied to these distances can be set by the user. In order to compare values, their range was often mapped onto a qualitative scale, with the points on the scale treated as being of equal distance. They use an abstraction hierarchy to measure the distance between components.

Watson [14] underlines the fact that CBR is a methodology and implementers are free to use whichever technology is appropriate to implement, mentioning nearest neighbour, induction, fuzzy logic and SQL as examples.

Our SEACOP (Search, Entry, Adaptation and Output of Patterns) system for knitwear design described here could be regarded as similar to a hybrid rule-based and case-based system as discussed in Prentzas [15]. Complex problems are easier to solve using a hybrid approach, as the advantages of the methods can be combined. The main advantages within SEACOP are the reduction in the knowledge acquisition bottleneck and the fact that CBR tends to work well with missing inputs, which is a feature of our system in that the cases are not "homogeneous". As the knowledge in a case base grows through time SEACOP will allow new and established fashion to be dealt with.

Arcos et al [16] present a domain-independent means of improving CBR system performance by learning from failure. Contrasting performance with model predictions can identify failures in retrieval and adaptation, but this does not help in "weak theory" domains (such as knitwear) where there is no readily available model. Their model is dependent on a CBR system being able to provide a measure of confidence for its own solutions; from this they can do a "blame assessment" to find the source of failures. This does not seem feasible in the domain of knitwear design, due its inherently subjective nature, as the measure of confidence cannot be accurately defined. However, methods of evaluation are discussed in section 3.4.

The existing work shows the prevalence of k-NN algorithms using weighted sums and techniques to deal with symbolic, rather than numeric data. This approach is suitable for our domain, since adaptation of our hierarchical representation (see section 3.1) will mean we do not need to index our cases or organise them in a particularly complex way, leaving us to focus on the challenges of adaptation.

CBR is particularly suitable for the knitwear design problem. Designers themselves study many past patterns before they create a new design. A new design will always contain some new element or an interesting combination of elements used in previous patterns, but will be similar to previous designs. A new sweatshirt will most commonly contain the four ordinary *pieces*: front, back, right and left sleeves. However, their shape and structure will be designed according to new trends and wool type. There could be many neck and armhole types, various different lengths and sleeve lengths, possible accessories (such as a belt) or interesting borders, and the stitch pattern can vary across the whole sweatshirt.

Sirdar Spinning Ltd. is producing about 300 new patterns every year, so the case base will contain a limited number of very specialised cases. A new case cannot always be solved by adapting a single case, but rather by combining and adapting several sufficiently similar cases (over different attributes). Once the high level representation is created for the new case, the details at the lowest level must be rigorously produced in order to obtain a full solution. Our system cannot solely rely on the case base as the new trends may involve a new element or shape that has not been encountered before. The human user will be offered the possibility to create a new design themselves with the help of the system and then store it in the case base.

More generally, Tecuci et al [17] discuss systems which interact with humans to complete a task. SEACOP is such a system, utilising the fashion knowledge and intuition of the designer, and the rapid processing power and fatigue tolerance of the computer (described as the task issue by Tecuci). We agree that the operations performed by the human should be natural and easy; for example our interactive sketch wizard reflects the designer's visual way of thinking. Also, control should be divided between human and computer in a way that maximises performance.

3 CBR APPLIED TO KNITWEAR DESIGN

Here the specific areas of CBR, representation, retrieval, reuse and retention, as applied to knitwear design are described.

3.1 Case Representation

We represent our cases using several levels of detail:

1. *Specification*: A high-level feature-vector representation that is similar to a sheet currently used by knitwear designers.
2. *Sketch*: A visual illustration of the shape, consisting of points, lines or curves between them, and rules that govern them.
3. *Linked Regions*: Contains 'regions' of the sketch which delineate changes in pattern or tension, together with a small portion of the chart which describes the stitch pattern in each region.
4. *Chart*: A detailed and complete description of the knitting stitch pattern that makes up the garment.
5. *Pattern*: A series of textual instructions to a knitter.

The multi-level representation is consistent with the existing process, where detail develops as the design progresses. It also facilitates case-based reasoning since we can use the specification in the retrieval stage, so indexes are not required.

We store much of the data about our cases as XML files, in a human-readable format. This affords portability, and facilitates testing, since no interpretation of the files is required.

3.2 Similarity Algorithm for Retrieval

The proposed retrieval algorithm works by filtering the case-base up to three times, depending on needs:

- Firstly, it filters on the specification, using the algorithm described in section 3.2.1.
- Then, it may filter on the sketch, as described in section 3.2.2.
- Then, it may also filter on the chart, as described in section 3.2.3, if need be.

The algorithm compares full garments, and also pairs of *pieces* of the same type (i.e. front, back, sleeve for sweaters).

3.2.1 Specification Similarity

We use a weighted sum algorithm to assess the similarity of two specifications. The first step is the calculation of the raw similarity S_R; the higher the value of S_R, the more similar the garments are judged to be. Finding the most similar garment to the new one is a maximisation problem. We argue that maximisation of similarity is more intuitive to the non-technical user than minimisation of distance. Each feature in the specification for the new garment that we are creating is compared to the equivalent feature in the previously created garment stored in the case base. If the features match, a weight is applied to S_R. The values of the weights are calculated based on user preference. As designers find it hard to allocate weights directly they are only asked to provide order of preference and they can also change the shape of the weight conversion function (see section 4.7).

In many cases, the values in the specifications are “yes/no” choices. However, in some scenarios there is a choice from a list, and options may differ in their similarity to each other. For example, a wrist-length sleeve is more similar to a 3/4 length sleeve than it is to a very short one. We define a 2 x 2 matrix for the attribute values and the user can assign a similarity score to each pair of values. In these situations the weighted score is added to the raw similarity (S_R).

If the feature is not present in the garment then the weight is not calculated and other features which are dependent on this feature are ignored in the existing garment. For example, if we are creating a sleeveless garment then a weight ($W_{hassleeves}$) would be used for existing sleeveless garments; but not for sleeved ones. In the latter case, details such as the cuffs are irrelevant.

Equation (1) gives the raw similarity of a the specification of garment (a) to another garment (b), given the applicability (A_i), the weights (W_i) and the similarities (S_i) of all n attributes: fastener, button position, waist shape, neck shape, armhole style and sleeve length.

$$S_R(a,b) = \sum_{i=1}^{n} A_i W_i S_i(a,b) \tag{1}$$

The applicability A_i has a value 1 if the feature is present and 0 if it is not. Then S_R is normalised to give a value S_N which is a value in the [0,1] range where 0 corresponds to completely different and 1 to identical:

$$S_N(a,b) = \frac{S_R(a,b) - S_M(a)}{S_R(a,a) - S_M(a)} \tag{2}$$

where S_R of case (a) is its similarity to itself and S_M is the minimum rough similarity achievable on a garment (a):

$$S_M(a) = \sum_{i=1}^{n} A_i W_i M_i(a) \tag{3}$$

The minimum similarity scores M for the special case of a cardigan with buttons and sleeves are detailed in table 1; the general case will involve a subset of the terms in the product in equation 3.

Term	*Minimum score for..*	*Applicability*
$M_{fastener}$	the type of fastener	cardigans only
M_{btnpos}	the choice of button position	cardigans with buttons only
M_{sllen}	the option for sleeve length	garments with sleeves only.
M_{waist}	the option for shape of waist	(universal)
M_{neck}	the choice of neck shape	
$M_{armstyle}$	the armhole style	

Table 1: Explanation and applicability of terms when calculating S_M (equation 3)

The minimum similarity scores M take on values that depend on the particular value of the associated attribute in case (a). For example, a long sleeve may be judged to have no similarity to a very short one, whereas a mid-length one will probably have some similarity to both.

3.2.2 Sketch Similarity

On the next level, when comparing sketches, several features of the sketch are compared. The similarity of the outermost polygons on two pieces of the same type is computed by comparing the positions of equivalent points (such as bottom of the armhole) relative to the origin of the piece.

The similarity of panels (i.e. parts that have a special feature, such as a different stitch pattern from the background) on the sketch of the new garment and the stored case is obtained by comparing size, shape, stitch pattern, piece similarity of contained panels, as well as number of panels on the garment or pieces being compared. The stitch pattern similarity uses several domain specific heuristics, such as the proportion of holes for lace patterns.

In order for a sketch from the case base to be considered sufficiently similar to the query case for adaptation, all feature similarities have to be above a feature-specific preset threshold.

3.2.3 Chart Similarity

This is the lowest level of detail for comparison, done at the knitting stitch level. The pairs of garments, pieces or panels compared on this level have pairs of individual knitting stitches matched against each other and their similarity accumulated to form the similarity of the garments, pieces or panels, respectively.

Chart similarity is only computed in the rare occasion when the sketch similarity is so high that only adaptation at the lowest level is needed (for example, changing number of buttons).

3.2.4 Piece complexity

The retrieval algorithm (described in section 3.2.5) must be able to assess the complexity of a piece, and decide on the next step depending on it.

We devise a complexity assessing function based on specific domain knowledge. The principal feature defining the complexity of a piece is its background stitch and its shape; some stitch patterns are hard to fit into a particular shape. The presence of more than one pattern, e.g. because there is a border or yoke or cuff, increases the complexity. Other factors, specific to the type of piece, including the presence of buttons, the presence of pockets, the neck shape, the armhole and sleeve head shape, extra shaping e.g. fitted waist or the presence of a collar contribute to the piece's complexity.

The complexity (C) of a piece will have an associated value in the range [0,1] with higher values corresponding to higher complexity computed as the weighted sum of the relevant complexity components.

3.2.5 Retrieval

The retrieval algorithm works as follows:
1. The algorithm first produces a *pool* of garments that are sufficiently similar to the query case. The levels of filtering (i.e. specification, sketch and chart) applied will depend on whether the user added any features to the garment being designed on the sketch and/or chart level.
2. The most similar garment in the pool is selected. If its similarity to the query case exceeds a preset threshold value, and the similarity of each piece to the corresponding piece of the query case also exceeds a predetermined value, then the retrieval terminates, this garment is retrieved as a whole, and we proceed to adaptation.

Pieces for case merging are selected as follows:

3. If there are any unmatched pieces of the query case, then for the most complex unmatched piece (Q) a sufficiently similar piece from the garments in the pool is selected. If such a piece (R) is retrieved, then the algorithm moves to step 5; otherwise it goes to step 4.
4. The pool is expanded (by relaxing the threshold value) until it contains a similar piece or the whole case base is included in the pool. At this point, if no match is found, then the CBR will fail. Otherwise, step 5 follows.
5. The *matched pool* is defined as the set of garments from the pool whose pieces have been selected for the merging, and is initially empty. If the piece R selected as the best match for piece Q is part of a garment in the matched pool, then it is retrieved and the algorithm continues with step 3.
6. The piece M from the matched pool that is most similar to Q is found.
7. If M is deemed to be a sufficiently good replacement for R then M is selected for merging and the algorithm continues with step 3.
8. If there is no M to replace R, the garment containing R is added to the matched pool and the algorithm is repeated from step 3.
9. If all pieces have been matched the algorithm terminates.

If a single garment is retrieved, it is adapted in the traditional sense. If pieces from several garments have to be merged, a *divide and conquer* type method is needed.

3.3 Adaptation for Reuse

One of the assumptions of CBR is that if two objects are similar then one can be adapted into the other. However, the well-known '15-puzzle' explained by Archer [18] illustrates that this assumption is questionable in some circumstances. In the 15-puzzle, numbered tiles are moved around on a board with the goal to finish with the tiles in a particular order. Some configurations of tiles have no tile in the correct position but are solvable. However, other configurations involve just two of the 15 tiles being in the wrong position but are completely unsolvable. It needs further consideration to establish whether this holds in our domain. As a fallback remedy, our users always have the option to manually create a knitting pattern.

Mitra and Basak [19] survey the adaptation methods used in many CBR systems, and classify them in various ways, e.g. knowledge lean and knowledge intensive. Our domain perhaps lends itself more to *knowledge intensive adaptation* since the cases are highly structured. Stochastic methods are likely to require extensive repair mechanisms. The danger with derivational replay is that the way that designs are constructed by humans is likely to be idiosyncratic and inconsistent.

Substitution (swapping parts of the existing and newly created cases) and transformation (making structural changes to the newly created case) seem particularly applicable. However, the drawback of these techniques is that they require domain knowledge, so they reduce one of the claimed benefits of CBR, i.e. the elimination of the knowledge elicitation bottleneck.

In SEACOP, the adaptation is applied on the sketch, chart or pattern level. The linked regions representation is too 'high level' to be amenable to adaptation. Examples of adaptation rules and corresponding results are shown in table 2.

It is worth noting that our search space is large: the specification (see section 3.1) contains non-integer values so the number of different garments it is capable of specifying is theoretically infinite. As in our domain ease of adaptation correlates with structural similarity, then the danger is that the gap between the new solution and the nearest existing case will be too large for any effective adaptation strategy.

The obvious reuse strategy is to attempt to adapt the existing garment(s) with the highest similarity. Normally, if this is unsuccessful then the next most similar garment can be attempted, and so on. However, we may consider introducing diversity, as discussed in Aamodt et al. [7], since if the most similar case cannot be adapted then attempting adaptation on cases that are very similar to this may also be futile.

One of the ways to avoid the large search space problem is proposed by Watson and Perera [20] in their ‘divide and conquer approach’. Knitwear is composed of separate pieces, e.g. cardigans typically consist of a back, front, and two arms. The CBR could be applied separately on these pieces. The new cycle becomes:

- **repartition** the query case into constituent parts
- **retrieve** cases which have parts that are similar to the equivalent parts in the new case
- **reuse** the parts
- **revise** using one or more of the adaptation methods discussed

- **recompose** the parts back into a whole
- **repair** any inconsistencies between the parts
- **retain** the solution for the future.

Level	*Adaptation Situation*	*Example result*
sketch	Adding or changing the number of buttons	A garment with 5 buttons becomes one with 9 (the new buttons are inserted in-between the old ones)
	Changing the fastening type	A sweater becomes a cardigan with a zip
	Changing from a man's to a woman's cardigan	The buttons are moved to the other side
	Adding or removing a border, band or yoke	The stitch pattern in the top part of a sweater is made different to the background stitch
	Adding or removing a collar or hood	A hood is removed
	Adding or removing pockets	The pockets are removed from the front of a cardigan
	Changing the position of pockets	Pockets are moved towards the side of the garment slightly to make them fit in with a pattern better
	Resizing	A garment is made longer; this is not necessarily proportional (e.g. pockets may stay the same size)
	Altering a shape	A vertical panel on the front is changed into a rectangle
	Merging/cutting shapes	Two horizontal bands on a garment are merged to make one larger horizontal band
	Adding or removing a panel	A horizontal panel on a back is removed
chart	Altering a pattern	A reversed stocking stitch pattern on the front is changed to garter stitch
	Changing to a yarn with a different tension	An '8 ply' yarn which has 22 stitches and 30 rows to 10cm is replaced with a '4 ply' yarn with 28 stitches and 36 rows to 10cm
	Changing the 'background stitch'	A plain sweater with a stocking stitch background is changed to one with a lace background
	Changing the stitch of the collar, hood, or a band, border or yoke	The shape of the neck band is changed to make it correspond to a changed neck shape
	Making a sleeve 'fully fashioned'	The 'decrease' stitches which shape a sleeve are placed on the edges of the sleeve in a line, rather than being evenly dispersed throughout the sleeve
pattern	Altering the text of the pattern	The instructions for shaping a sleeve are grouped differently to reflect the fact that the shape has changed
	Changing the description of the garment	The title at the top of the pattern is changed, perhaps to make it consistent with other changes, e.g. "Women's short sleeve cardigan" becomes "Women's raglan sleeve cardigan"
	Adding in or removing the use of markers or stitch holders	The text is used to reflect the fact that markers or stitch holders are recommended
	Changing the colour of yarn	A grey sweater becomes a purple one

Table 2: Example adaptation situations

The major disadvantage of the divide and conquer approach is that the parts need to be independent. The potentially introduced inconsistencies are usually dealt with in the repair phase. In the case of a cardigan design, the inconsistency could be the production of pieces with slightly mismatched armhole, which can be repaired using well-defined rules. It is not appropriate to try to "merge" shapes in knitwear, since it would lead to garments where the shapes do not correspond to any established style. Instead, SEACOP's rules will establish a precedence order so that when the pieces of knitwear are recomposed, the shape of the dominant piece is effectively "cut out" of the other piece.

One could imagine an inconsistency of a front and a back with different length as such, but this can be very simply avoided by using strong constraints on the size of the pieces in the first place in the repartitioning phase. Alternatively, one of the pieces could be resized.

Purvis & Pu's COMPOSER system [21] used multi-case adaptation in two case-based design problems. The adaptation phase was viewed as a constraint satisfaction problem; they use a greedy algorithm to find a good initial solution then employ a minimum conflicts algorithm. The algorithm uses heuristics to change the values of variables until all constraints

have been satisfied. They claim that their approach requires less adaptation knowledge. The fact that the constraints have to be identified appears to be a disadvantage of this approach, but in our case most constraints are simple to specify (for example, matching length and matching armhole for pieces, symmetries) and the creation of garments 'from scratch' will have to include them anyway for the purpose of consistency checking.

3.4 Retention

To ensure compliance with potentially unpredicted detailed user requirements, the user will always have the final say in accepting the adaptation proposed by the system and potentially propose changes before the solution is produced.

We could develop a means of automatically measuring the amount of editing the user has to do on the output of the CBR. However, there may not be a strong correlation between this measure and success. Therefore, we propose to evaluate the output of the CBR by simply asking the user to score it on a scale of 1 to 5 inclusive, with higher scores corresponding to higher preference. All cases must be retained for the benefit of the designer, but only preferred patterns (e.g. score ≥ 3) will be retrieved by the CBR process in future.

4 ARCHITECTURE AND IMPLEMENTATION

The architecture of SEACOP is shown in figure 1.

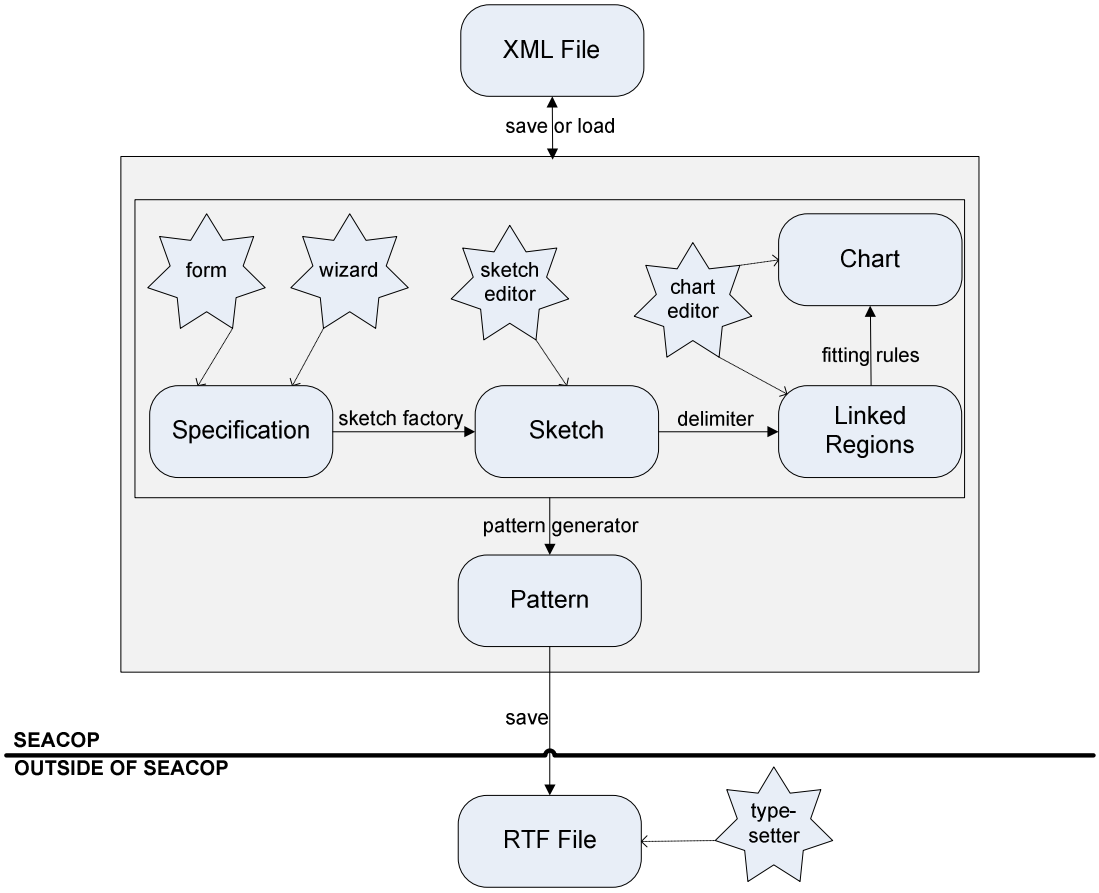

Fig 1: Architecture of SEACOP

We have explicitly coded our features and similarity measure into our Java software, as opposed to ‘adaptive programming’, described by Long et al. [22], using metadata instead. Our users require a tool to create new cases manually, so the features must be specified in our program code anyway. Our software tool offers both ‘creation from scratch’ and creation using CBR.

The specification of a garment is generated either from an interactive wizard, or a non-interactive form. Data can be added to the specification in three ways: the form, wizard or pattern generator. The form and wizard both add answers to questions

about a new garment, and this is used for generating a sketch as well as for the similarity algorithm. The pattern generator adds data which is used to generate the pattern (e.g. sizes of needles), but this is not necessarily used for similarity.

The specification is transformed using a series of rules into a sketch, which the user can edit using the sketch editor. The chart factory uses further rules to produce the linked regions representation, which consists of patterns of a series of shapes called regions which control the consistency of the pattern between sizes.

In most cases, the pattern generator will use the linked regions representation to create the textual pattern, avoiding having to explicitly generate a chart. However, the fitting rules are useful when the user wants to be able to view the chart in order to check the design that was automatically generated.

The user can edit the chart that they view, in which case the chart will be used to generate a pattern; the chart is a lower-level representation of the garment than the linked regions and the resulting pattern will be more verbose.

4.2 Process Flow

A useful case-based design system effectively automates the full CBR cycle: retrieve, reuse, retain and revise [7]. However, we start with an empty case base, and also new trends emerge, so manual methods must be supported. Many studies (e.g. [23]) have confirmed that iteration must be supported when producing a semi-automatic design system. Accordingly, the users will be allowed to move back and forward between the specification, sketch, and linked regions stages. Automated creation may jump straight from the specification to the linked regions.

4.3 Transformation Rules

To support the manual creation of garments, we require rules which will enable us to convert the specification into the sketch, the sketch into the linked regions, and the linked regions into the chart. Equally, to ensure that the representations do not become inconsistent, we need inverses to these rules. This is analogous to the discussion given in Börner [24]. The rules are coded in the system using domain knowledge elicited from experts at Sirdar. For example, the feature-vector representation of an armhole shape is just a single integer. This is transformed into a series of points which either form the outline of the shape, or in some cases control points for a Bézier curve. This is then discretised in the chart into a matrix of individual stitches.

The main purpose of the rules in the *delimiter* is to determine the boundary of regions, and how the dimensions of regions change between different sizes of the garment. A region boundary will occur when:

- The tension or yarn or colour changes.
- A specific feature occurs such as a pocket.
- The stitch pattern changes (for example, a cable is included).

The fitting rules are concerned with how to fit a stitch pattern into a particular shape. The basic fitting rule can be summarised as:

1. Find the smallest bounding box of the shape.
2. The stitch pattern's tension is provided as a rectangle. Repeat this rectangle both horizontally and vertically until it completely fills the bounding box; if necessary use partial repeats of the pattern to ensure each stitch within the area covered by the bounding box is covered.
3. Discretise the shape and flag those stitches that are not covered by the shape with a 'no stitch' marker.
4. Remove unknittable stitches, e.g. holes that are the last stitch on a row.
5. Insert increases or decreases so that each row is valid, i.e. the number of increases and decreases is balanced out by any change in the number of stitches for a row.

The pattern generator will use similar algorithms to the fitting rules, but it will generate pattern text, rather than a chart.

4.4 Initial Stages of Pattern Creation

Figure 2 shows the wizard and sketch of the knitwear design system. Designers want a continuous display of the garment, so the outline of each of the pieces to be knitted is displayed in the background, as they are providing the data for the garment.

The Sleeves stage of the wizard is displayed in the foreground. The armhole style option will change the shape of the armhole and sleeve; in this example a set-in sleeve has been chosen. The user is then offered options as to whether the garment is sleeveless, and whether the left and right sleeves are automatically the same ('left & right perfect symmetry'). There are also options that control whether the shaping of the sleeve features a straight portion at the top or bottom, and whether it is "fully fashioned", which is a detail about how the shaping is achieved. There are options to control the length of the sleeve, the tightness of the cuff, and the pattern that is used for the cuff. The box at the bottom of the window shows the progress starting from Size, then Fastener, General, Neck, Pockets, and finally Sleeves.

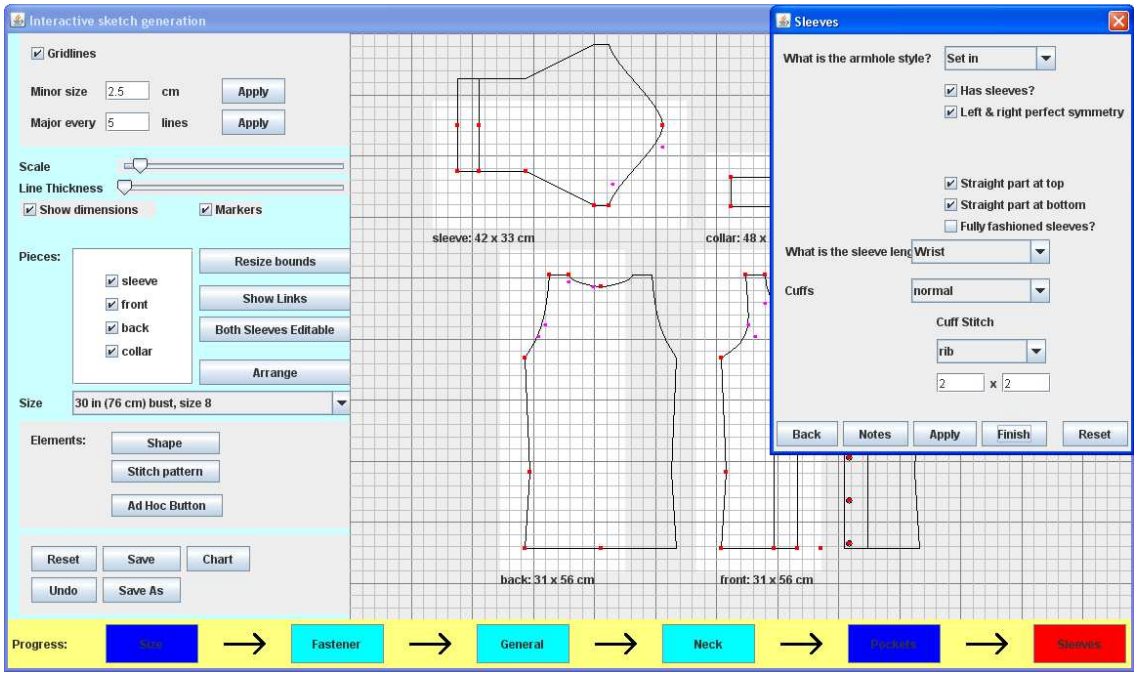

Fig 2: Interactive Sketch Generation

Figure 2 shows the last stage in the wizard; the progress is indicated by the colour of the button:

- *grey* refers to a stage that has not yet been visited (not shown in this example as it represents the last stage)
- *red* is the current stage
- *light blue* indicates that the stage has been visited and something was changed from the default settings
- *dark blue* indicates that the stage has been visited but the default settings were simply accepted; this can sometimes indicate carelessness or errors.

The top left area of the window contains various controls. The user can turn off or control the spacing between the gridlines, using the Gridlines panel. The 'scale' slider allows zooming in and out and the line thickness can also be altered. If the 'show dimensions' box is ticked then a label appears under each piece of knitwear denoting the size of its bounding box. The markers are the red squares and purple circles which denote the ends of line segments and the control points for Bézier curves; the use can opt to hide these by unticking the 'markers' box.

The sketch initially shows the whole garment, as it would be constructed by a knitter in separate pieces, i.e. sleeves, back, and front. The user manipulates the sketch of the garment by dragging the markers around with the mouse. There are, however, restrictions on how the sketch may be manipulated, for example the editable half of the sketch must lie within the *edit area*, which is shown as white rectangles in figure 2.

The left middle area of the window contains mostly presentation related options: the user can tick boxes to indicate which pieces of knitwear they want to see. They can also resize the edit area, show or hide the 'linked points', and choose whether the right sleeve is editable or not (in which case it is a copy of the left). The arrange button resets the position of the pieces of knitwear, so they are centred on the sketch panel.

The user can also control the size of the garment that is displayed; in the example in figure 2 the 'base size', i.e. the smallest one, is shown. Larger sizes are shown but cannot be edited (as they are computed as functions of the smallest size).

The elements panel allows the user to add a button or insert a shape or stitch pattern into the sketch. Example shapes are: vertical band, horizontal band, square, rectangle, diamond. The software can be amended to include other shapes at a later stage, if this is required.

The bottom left part of the window contains buttons which the user can use to reset the sketch back to the default, save the sketch, undo the last operation, and proceed to the chart stage.

4.5 Control of Constraints in Sketches

A multi-levelled system of constraints enables the integrity of the sketch to be preserved. When the user tries to manually alter shapes then at a fundamental level, since the line segments represent the outline of the shape and structures such as borders, it is important that they do not cross over. Also, they always intersect with another line segment or curve. These fundamental rules always apply, and if a move of a point attempts to violate them then it will be disallowed.

Other rules are more discretionary. For example, extreme deviations in the Bézier curve would produce a sleeve which may be knittable, but is unlikely to be practical or aesthetically pleasing. Rules which restrict the movement of points based on domain knowledge are included, but the user is able to 'turn off' these rules, for example, if they are trying to design new shapes.

By default, certain features of the sketch are preserved when the user drags a point:

- Lines are horizontal or vertical.
- The character of Bézier curves is preserved.
- The alignment of buttons is fixed.
- The shape of pockets is prescribed.
- Connections between pieces are kept through so-called *linked points* operating on two separate pieces in the sketch. For example, they can be used to keep an armhole consistent with the sleeve.

This is also discretionary and can be turned off by the user

Figure 2 shows that many of the pieces are composed of two symmetric halves. The editable half is contained within a the *edit area*, indicated by a white rectangle. The user will only be allowed to move points inside this box. However, the user can resize the edit area and also turn off the symmetry so the entire piece becomes editable (in which case the bounding box is automatically enlarged as appropriate).

The features are preserved by automatically moving points in response to a change, so that movement cascades throughout a garment. There is potential for conflict since one point on the sketch may be related to several features. Therefore we implement a precedence algorithm. If it is possible to repair a potential failure through a small change (for example, by shortening a line to stop it from crossing another line), this is automatically done.

4.6 Final Stages of Pattern Creation

Figure 3 shows an example of a chart editor, depicting a small portion of a garment. The knitting chart is a matrix of elements indicating what the knitter should actually do step-by-step to achieve the shape and structure of the garment. Each element corresponds to one stitch and is represented by a symbol carrying significance. For example, a circle typically means to create a hole and a triangle to combine two stitches into one. Each row in the chart corresponds to a row of knitting in the finished garment. The user can edit these symbols by numerous means, e.g. drag and drop with a mouse.

There are two yarns: a light purple coloured one for the bottom and a darker purple-coloured one for the top, and there is a border knitted with different size needles. Note that the colours on the screen do not have to correspond to the actual colour of the yarn.

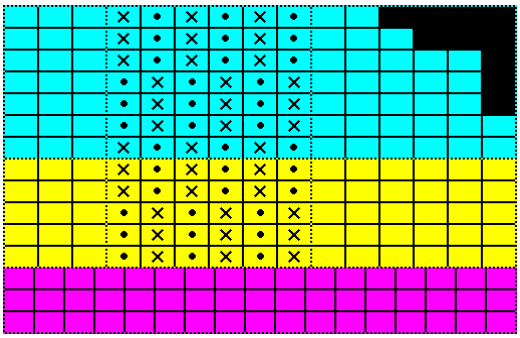

Fig 3: Proposed Chart Editor

There are seven regions in figure 3, each bordered by a dashed line.

- The three dark purple regions on the top are one colour of yarn. In the centre there is a 6x3 repeating pattern. On the right there is some shaping indicated by the black colour ('no stitch').
- The three yellow regions in the centre correspond to a different colour of yarn (i.e. light purple) with the same tension.
- The magenta region at the bottom is the same yarn as the yellow region. However, it is a different size needle and therefore a different tension.

The product of the design process is a written knitting pattern, ready to be printed. An extract of a written pattern is shown in figure 4. It can be seen that the written pattern is a series of textual instructions to the knitter, partly codified, using industry standard terminology. The knitting pattern can be produced in a fairly straight-forward automatic way from the knitting chart, using predefined rules.

1st Row.K0 [2:2:0:0:1], p2 [2:2:1:2:2], * k3, p2, rep from * to last 0 [2:2:4:0:1] sts, k0 [2:2:3:0:1], p0 [0:0:1:0:0].2nd Row.K0 [2:2:0:0:1], p2 [2:2:1:2:2], * yon, s1, k2tog, psso, yfwd, yrn, p2, rep from * to last 0 [2:2:4:0:1] sts, k0 [2:2:0:0:1], (yon, s1, k2tog, psso, yfwd, yrn, p1) 0 [0:0:1:0:0] times.3rd Row.K0 [0:0:1:0:0], p0 [2:2:3:0:1], * k2, p3, rep from * to last 2 [4:4:1:2:3] sts, k2 [2:2:1:2:2], p0 [2:2:0:0:1].From 1st to 3rd row sets patt. Keeping continuity of patt as set (throughout) work until back measures 44 [46:50:54:58:60]cm, (17 1/4 [18:19 ¾:21 1/4:22 3/4:23 3/4]in), ending with a rs row.

Fig 4: Pattern

4.7 Similarity Preferences

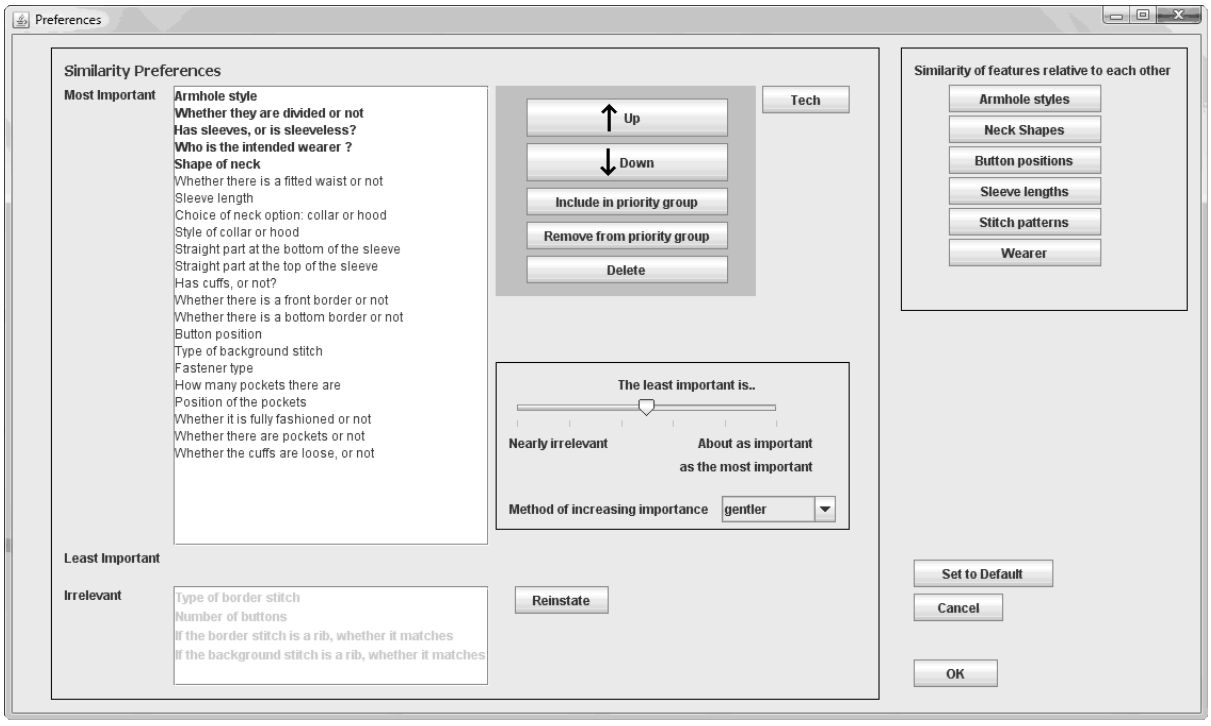

Figure 5. Preferences

In order to facilitate the retrieve stage of CBR, we allow the user to specify their preferences for similarity between the new pattern being created and the retrieved pattern from the case base. We allow the user to indicate the relative importance of features, for comparison; our approach involves ranking features in order of preference, since people are typically more comfortable doing this than they are assigning numeric weights.

Figure 5 shows the preferences window. On the left, the users can rank in order the importance of features; important items towards the top of the list are highly relevant. Items in bold are all of equal and maximum relevance, and items in the separate list at the bottom are irrelevant. Hence, we cater for situations in which the user feels that several factors are highly significant, but they cannot decide which one is more important. Also, by allowing factors to be marked as irrelevant, we allow the user to exclude things that have no bearing on similarity. The slider is used to set the scale. We offer a choice of seven functions for the progression of weights from one (most important) to zero (irrelevant). Figure 6 shows an example with 28 features, none of which are irrelevant. The shapes of the functions are shown; the middle is arithmetic progression.

In the preferences (Figure 5) we also allow the user to set the relative score for similarity between options for neck style, armhole style, etc. These options are available by clicking on the corresponding button in the top right box. Sleeve shapes are an example: set-in is similar to semi-set in, but different from raglan. Similarities are expressed using a Likert [25] scale, which is mapped linearly into the [0,1] range and the weighted scores are added up to produce the specification similarity as described in Section 3.2.1.

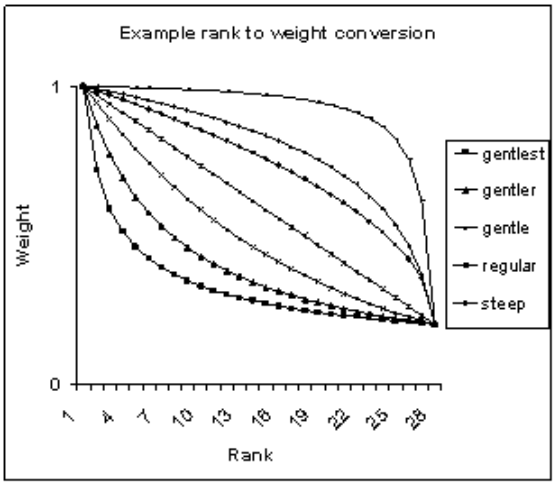

Figure 6: Functions for converting ranks to weights

5 EXAMPLE OF CBR APPLICATION

Table 3 illustrates an example by listing the specification of a query case (Q) and two cases from the case base (R and S). The similarity values $S_N(Q,R)$ and $S_N(Q,S)$ are calculated as per section 3.2.1 where $S_M(Q)=0.75$.The pattern referred to as trellis lace has the stitch pattern shown in figure 7; it is 4 rows and 12 stitches that are to be repeated.

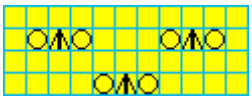

Figure 7: Trellis Lace

On simple visual inspection of Table 3 it is hard to decide whether Q is more similar to R or S. Knowing preferences, for example that the similarity in garment type is more important than the background stitch, one can infer that Q is probably more similar to R than to S. Our similarity function derives the specification similarity values 0.84 and 0.65 respectively. Therefore case R will be retrieved first, and with a similarity threshold for single case adaptation of. 0.8, it can be adapted for the solution. In this case, adaptation involves:

- Swapping the buttons over (since a woman's cardigan is buttoned on the opposite side to a man's).
- Making it longer and wider (since the length adjustment and width allowance are higher).
- Switching the background stitch from Stocking Stitch to Trellis Lace.
- Adapting the waist shape to make it fitted.

- Decreasing the length of the sleeves.

Stage	*Question or Option*	*W*	*Query Case (Q)*		*Case R*			*Case S*		
			Value	*S*	*Value*	*A*	*S*	*Value*	*A*	*S*
1	Who is the wearer?	0.58	Woman	0.58	Man	✓	0.00	Woman	✓	0.58
	Minimum size	0.49	30in (76cm)	0.49	30in (76cm)	✓	0.49	30in (76cm)	✓	0.49
	Length adjustment	0.61	2-3cm	0.61	0cm	✓	0.32	3-5cm	✓	0.43
	Width allowance	0.64	5-8cm	0.64	0cm	✓	0.14	6-9cm	✓	0.45
2	Garment Type	0.97	Cardigan	0.97	Cardigan	✓	0.97	Sweater	✓	0.00
	Fastener type	0.85	Buttons	0.85	Buttons	✓	0.85	(none)	×	0.00
	Number of buttons	0.07	6-8	0.07	6-8	✓	0.07	(none)	×	0.00
	Button position	0.52	Entire Front	0.52	Entire Front	✓	0.52	(none)	×	0.00
	Front Border	0.76	Yes	0.76	Yes	✓	0.00	(none)	×	0.00
	Front Border Stitch	0.31	2x2 rib	0.31	2x2 rib	✓	1.07	(none)	×	0.00
3	Background stitch	1.00	Trellis lace	1.00	Stocking Stitch	✓	0.50	Trellis lace	✓	1.00
	Waist	0.82	Fitted	0.82	Normal	✓	0.41	Fitted	✓	0.82
	Bottom border	0.70	Yes	0.70	Yes	✓	0.70	Yes	✓	0.70
	Bottom border stitch	0.28	Stocking stitch	0.28	Stocking stitch	✓	0.28	Stocking stitch	✓	0.28
	Yoke?	0.73	no	0.73	no	✓	0.73	no	✓	0.73
	Yoke Stitch	0.16	(none)	0.00	(none)	×	0.00	(none)	×	0.00
4	Neck Shape	0.88	V	0.88	V	✓	0.88	V	✓	0.88
	Neck Option	0.79	Band	0.79	Band	✓	0.79	Collar	✓	0.40
	Collar Style	0.37	(none)	0.00	(none)	×	0.00	Folds over	×	0.00
	Collar stitch	0.25	(none)	0.00	(none)	×	0.00	2x2 rib	×	0.00
	Band stitch	0.19	2x2 rib	0.19	2x2 rib	✓	0.19	(none)	✓	0.00
5	Pocket position	0.55	L front, R front	0.55	L front, R front	✓	0.55	(none)	✓	0.00
6	Armhole style	0.91	Set in	0.91	Set in	✓	0.91	Semi-set in	✓	0.68
	Has sleeves?	0.94	Yes	0.94	Yes	✓	0.94	Yes	✓	0.94
	Straight part at top	0.46	Yes	0.46	Yes	✓	0.46	Yes	✓	0.46
	Straight part at bottom	0.43	Yes	0.43	Yes	✓	0.43	Yes	✓	0.43
	Fully fashioned sleeves	0.10	No	0.10	No	✓	0.10	Yes	✓	0.00
	Sleeve length	0.40	¾ Length	0.40	Wrist Length	✓	0.30	¾ Length	✓	0.40
	Cuffs	0.67	Normal	0.67	Normal	✓	0.67	Normal	✓	0.67
	Cuff stitch	0.13	2x2 rib	0.13	2x2 rib	✓	0.13	2x2 rib	✓	0.13
Raw Similarity				15.78			13.40			10.47
Normalised Similarity				1.000			0.84			0.65

Key

L left
R right
✓ applicable (A=1)
× not applicable (A=0)

Table 3: Example of specification similarity

As an alternative, merging of the two patterns may be possible as S has the same background stitch and the same sleeve length as Q so the way this pattern is fitted into the shape can be reused. Thus, by altering the parameters which control the relative desirability of adapting one garment versus merging several, we can explore the trade-off between the accuracy of case merging and the undesirability of the repair operation that it incurs.

If the similarities of the specifications of cases R and S relative to Q had been closer to each other, then we would have compared the sketches of the cases in order to gain a more accurate measure of similarity. Alternatively, if the user had not

altered the sketch (from the ‘default’ one produced by the rules) but instead had changed the stitches in the chart, then the charts would need to be compared.

6 CONCLUSIONS

Knitwear design involves many creative processes and as such it is a human activity that is hard to reproduce or automate using the computer. At the same time, the production of the associated knitting pattern includes several stages that are repetitive and mechanical for the human, which suggests automation. In this paper we described how the process of knitwear design, including the production of the knitting pattern can be automated using case-based reasoning.

We proposed a software system that allows both design from scratch and design based on previously created patterns. The system works in stages, starting with a simple specification, then a sketch, then the linked regions, which is convertible to a detailed chart and subsequently to a written text pattern. These stages are interactive; the user can decide how much they want to change the most commonly used options offered by the system. The linked regions or chart is then used to produce the written pattern automatically. The proposed system has two main goals: to facilitate the design and to ease the burden of calculations and checks. The designers will be able to easily reuse past patterns or their parts. Errors that are often present in early stages of pattern production and propagate to later stages will be prevented from occurring.

We envisage that through the use of case-based design, the company will be able to respond rapidly to fashion trends. Case-based reasoning systems reuse good practice, learning with each design that is produced, and thus becoming an automated repository of knowledge. Our feasibility study shows that CBR is particularly suitable for this problem domain. We investigated the important aspects of CBR: representation, similarity measures, adaptation, reuse, and retention using a prototype system for sweaters and cardigans.

We conjecture that fine tuning and flexibility for setting similarity thresholds at the different levels of the representation is crucial for the success of our hierarchical CBR in knitwear design. As the next step we are planning to continue our investigation on the frontiers of case merging.

7 ACKNOWLEDGEMENTS

We are grateful for the support of EPSRC and Sirdar Spinning Ltd. via a CASE studentship. We also thank Sue Batley-Kyle for her valuable help on knitwear design.

REFERENCES

[1] Eckert, C.M., Cross N., Johnson, J.H., 2000, Intelligent support for communication in design teams: garment shape specifications in the knitwear industry, Design Studies, 21/1:99-112.

[2] Ekárt, A., 2007, Evolution of lace knitting stitch patterns by genetic programming, Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation, 2457-2461.

[3] Tomiyama, T., D'Amelio, V., Urbanic, J. and El Maraghy, W., 2007, Complexity of Multi-Disciplinary Design, CIRP Annals - Manufacturing Technology 56/1:185-188.

[4] Urbanic, R.J. and El Maraghy, W.H., 2009, Using axiomatic design with the design recovery framework to provide a platform for subsequent design modifications, CIRP Journal of Manufacturing Science and Technology, 1/3:165-171.

[5] Richards, P. and Ekárt, A., 2008, Automating a Knitwear Design Process Using Case-Based Reasoning, 10th International Conference on The Modern Information Technology in the Innovation Processes of the Industrial Enterprises (MITIP) 2008.

[6] Richards, P. and Ekárt, A., 2009, Supporting Knitwear Design Using Case-Based Reasoning, CIRP Design Conference, Cranfield University, 2009.

[7] Aamodt, A. and Plaza, E., 1994, Case-based reasoning: Foundational Issues, Methodological Variations and System Approaches, AI communications, 7:39-59.

[8] Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T, Forbus, F., Keane, M., Aamodt, A., Watson, I., 2006, Retrieval, reuse, revision, and retention in case based reasoning, The Knowledge Engineering Review, 20/3:215-240.

[9] Price, C.J. and Pegler, I.S., 1995, Deciding Parameter Values with Case-Based Reasoning, 1st UK Workshop on Case-Based Reasoning, Salford, 121-133.

[10] Kolodner, J.L., 1993, Case-Based Reasoning Morgan Kaufmann Publishers Inc.
[11] Bergmann, R., Kolodner, J. and Plaza, E., 2005, Representation in case-based reasoning, The Knowledge Engineering Review, 20/3: 209-213.
[12] Craw, S., 2008, CM3016 Knowledge Engineering (Case-Based Reasoning), available online at http://athena.comp.rgu.ac.uk/staff/smc/teaching/cm3016, accessed 23 June 2008.
[13] Mejasson, P., Petridis, M., Knight, B., Soper, A., Norman, P., 2001, Intelligent design assistant (IDA): a case base reasoning system for material and design, Materials & Design, 22/3:163-170.
[14] Watson, I., 1999, Case-based reasoning is a methodology not a technology, Knowledge-Based Systems 12/5:303–308
[15] Prentzas, J. and Hatzilygeroudis, I., 2007, Categorizing approaches combining rule-based and case-based reasoning, Expert Systems 24/2:97-122.
[16] Arcos, J.L, Mulayim O. and Leake, D., 2008, Using introspective reasoning to improve CBR system performance, AAAI Metareasoning Workshop, p21-28.
[17] Tecuci, G., Boicu, M. and Cox, M.T., 2007, Seven Aspects of Mixed-Initiative Reasoning: An Introduction to the Special Issue on Mixed-Initiative Assistants, AI Magazine 28/2:11-18.
[18] Archer, A.F., 1999, A Modern Treatment of the 15 Puzzle, The American Mathematical Monthly, 106/9:793-799.
[19] Mitra, R and Basak, J, 2005, Methods of Case Adaptation: A Survey, International Journal of Intelligent Systems archive, 20/6: 627:645
[20] Watson, I and Perera, S, 1998, A hierarchical case representation using context guided retrieval, Knowledge-Based Systems 11:285-292.
[21] Purvis, L. and Pu, P., 1998, COMPOSER: A Case-Based Reasoning System for Engineering Design, Robotica, 16/3:285-295
[22] Long, J., Stoeckin, S, Schwartz, D.G. and Patel M.K., 2004, Adaptive Similarity Metrics in Case-based reasoning, Proceedings of Intelligent Systems and Control, Honolulu, Hawaii, USA, p260-265.
[23] Parmee, I., Cvetkovic, D., Bonham, C., Packham, I., 2001, Introducting prototype evolutionary systems for ill-defined, multi-objective design environments, Advances in Engineering Software, 32/6:429-441.
[24] Börner, K., 1994, Structural similarity as guidance in case-based design, in: S. Wess, K.D. Althoff, M. Richter (Eds.), Topics in Case based Reasoning, Springer, Kaiserslautern, 1993, pp. 197–208
[25] Likert, R., 1932, A Technique for the Measurement of Attitudes, Archives of Psychology, 140: 1-55.