

Design2GarmentCode: Turning Design Concepts to Tangible Garments Through Program Synthesis

Supplementary Material

7. Validations in Design2GarmentCode

7.1. Rule-based Validation

Rule-based validation primarily addresses issues of completeness and hallucination during the MMUA’s generation process. With prompts generated by the DSL-GA containing over 100 questions, the MMUA often struggles to provide comprehensive answers in a single attempt. Additionally, due to the inherent hallucination tendencies of LLMs, some responses may fall outside the reasonable parameter range defined by GarmentCode. To mitigate these issues, we compare the MMUA’s responses against a predefined complete question space to verify whether all questions have been adequately addressed before program synthesis. Each response is further validated to ensure it falls within GarmentCode’s permissible parameter space. Questions with either missing or invalid answers are sent back to the MMUA for re-evaluation, with a maximum of two validation loops to refine the outputs.

7.2. MMUA Design Comparison

During design comparison we ask the MMUA to compare the output design image versus the design input and propose modification suggestions to DSL-GA to edit the generated pattern. Design comparison is especially useful for image-guided generation, where the output design image is rendered from the draped garment mesh under a similar viewpoint to the input image, we use TokenHMR [15] to estimate the camera pose and rough human pose from the input design image. The prompt used for design comparison is given in Figure 10.

8. Implementation Details

We use GPT-4V [6] for MMUA, and an instruction tuned version of Llama-3.2-3B for DSL-GA (Γ). The following sections contain the detailed explanation for the finetuned DSL-GA (Supp. 8.1), and training details for the Projector Ψ (Supp. 8.2).

8.1. Finetuning DSL-GA Γ_{ft}

To optimize the trade-off between computational cost and generation quality, we use Llama-3.2-3B-Instruct[52] as the base model for DSL-GA, fine-tuned over two epochs with LoRA (rank 16) and a learning rate of 5×10^{-4} on a dataset with 583 hierarchically defined NL-DSL pairs from GarmentCode’s public code repository. All code generation experiments were conducted on a single NVIDIA GTX

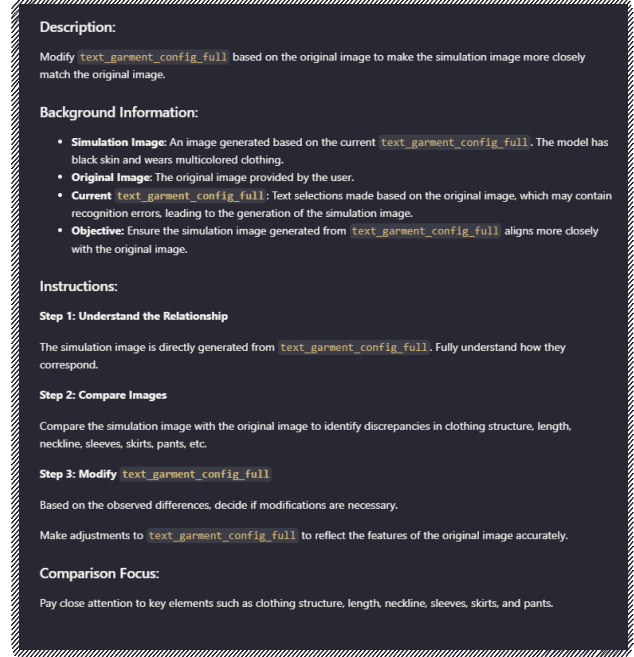


Figure 10. Prompt for MMUA during design comparison.

4090. For multi-modal understanding tasks, GPT-4V was employed as the designated agent.

8.2. Training The Projector Ψ

The projector Ψ is trained on the GarmentCodeData [32] dataset, which comprises approximately 115,000 garment samples draped on a standard A-pose body. We generate initial design descriptions for each sample using GPT-4V or rule-based inverse mapping from the ground truth design parameters for the sample, for example

```

if design.shirt.length.v > 1.0:
    return 'shirt_length_long'

```

The token sequence length is fixed at 122, which is equal to the number of design parameters in GarmentCode. The projection MLP and Transformer decoder are designed with feature dimensions of 128. The MLP consists of 4 intermediate layers, while the Transformer decoder includes 8 layers. Training is conducted using the Adam optimizer with a learning rate of 5×10^{-4} , a batch size of 16, and completed on a single NVIDIA GTX 4090 within 10 hours.

Notably, although we adopt a decoder-only Transformer architecture similar to DressCode, our innovative approach

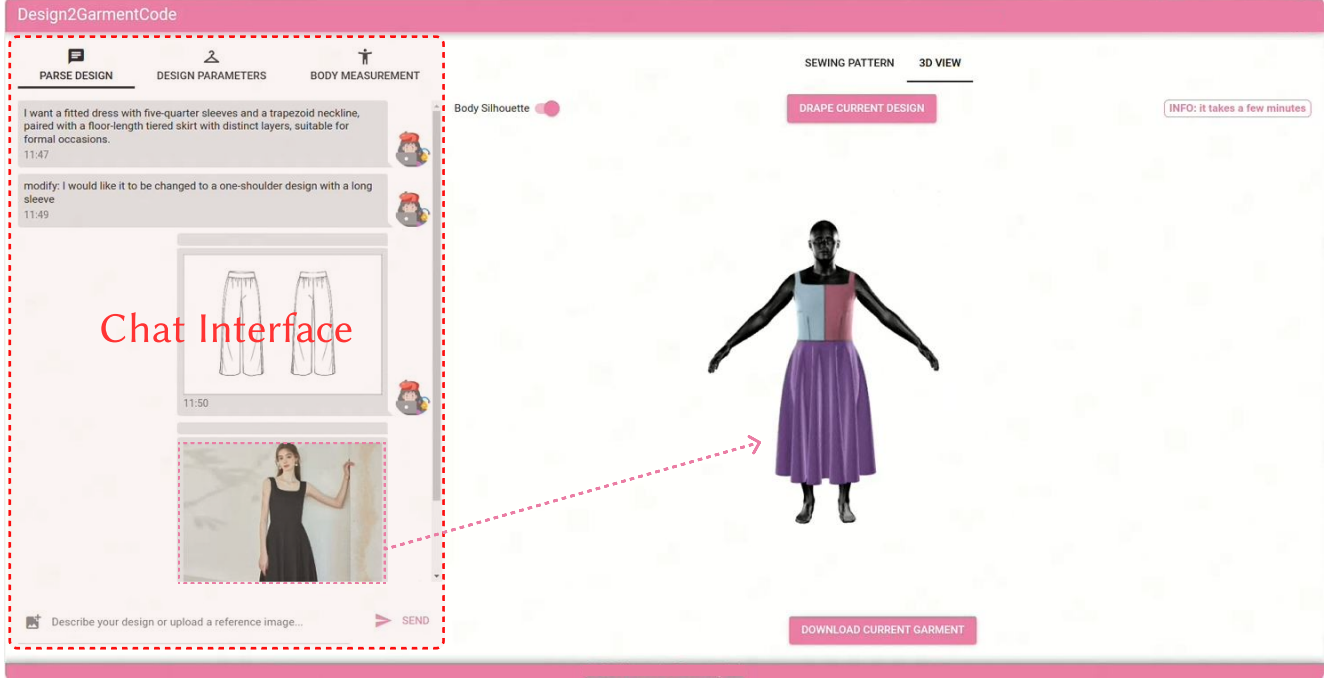


Figure 11. LMM-based interface for Design2GarmentCode, built upon the original GarmentCode GUI. The chat interface (left) allows users to provide natural language design descriptions or upload reference images or sketches, facilitating multi-modal design parsing into executable GarmentCode programs. We use the original GarmentCode execution engine to turn the generated program into 3D garments.

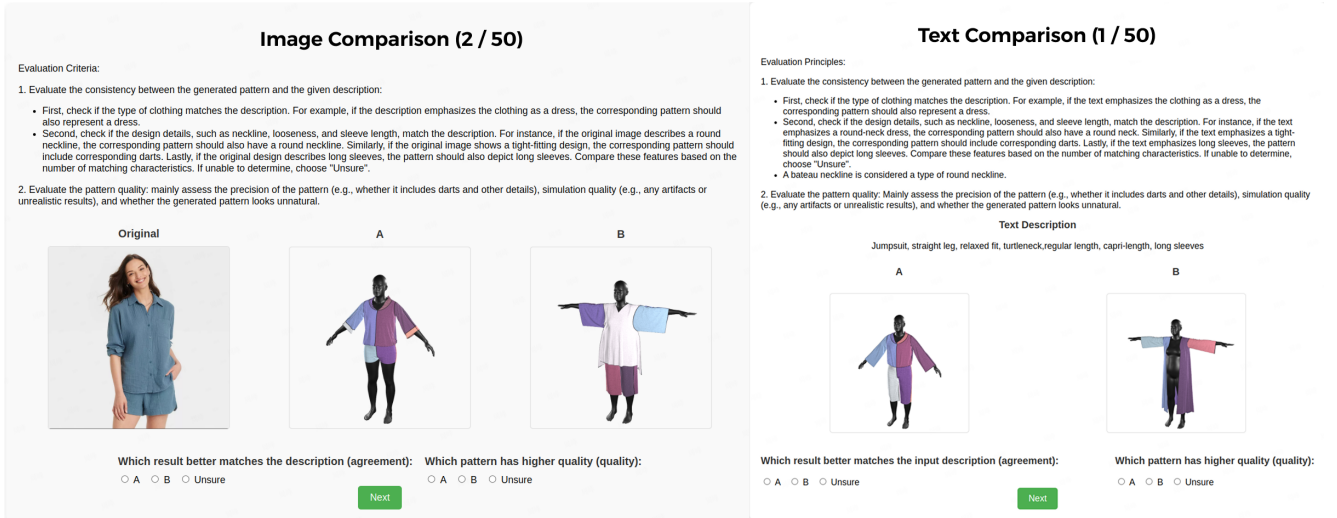


Figure 12. User study interface for evaluating sewing pattern generation quality. For each test input (Original image on the left for image-based evaluation or Text Description on the right for text-based evaluation), participants are presented with the simulation results of sewing patterns generated by Design2GarmentCode and a baseline method. Users are asked to evaluate the results based on two criteria: agreement with the input description and overall sewing pattern quality. If unsure, participants can select the "Unsure" option.

of quantifying sewing patterns through design parameters proves to be significantly more efficient and scalable. Specifically, with DressCode’s quantization scheme, the to-

ken sequence length is calculated as:

$$L_{seq} = N_p \times (N_e \times L_e + \|R\| + \|T\| + N_e \times \|S\|) + 2$$

where N_p , N_e denotes the maximum number of panels and edges respectively. $\|R\| = 4$ is the length of rota-



Figure 13. Example answers from Llama 3.2 3B when prompted with “How to draft a basic upper body bodice?”.

tion quaternions, and $\|T\| = 3$ is the length of 3D translation vector. $\|S\| = 4$ represents the per-edge stitching parameters containing a stitch tag and its existence indicator. L_e represent the length of quantified edge vectors, which might be 6 for cubic bezier curves and 4 for quadratic bezier curves. Using GarmentCodeData as an example, to fully cover GarmentCode’s modeling space, the required sequence length under DressCode’s method would be 13,951, with $N_p = N_e = 37, L_e = 6$, which will cost $\approx 1.5h$ to generate a single sewing pattern using DressCode, while our token sequence length is fixed at 122.

8.3. Inference Interface

For more convenient inference, we build an intelligent chat-based interface integrated into the original GarmentCode [31] GUI (Figure 11). The chat interface (left panel) enables users to provide natural language descriptions, upload reference images, or supply design sketches, facilitating multi-modal design parsing into GarmentCode-compliant programs which are then passed to the GarmentCode execution engine. The engine generates sewing patterns and 3D garment simulations (right panel). This interactive interface provides an intuitive environment for creating, editing, and refining sewing patterns, significantly improving accessibility for users without extensive expertise in parametric pattern-making. We provide a recording to demonstrate the inference process in [demo.mp4](#).

8.4. User Study Interface

To evaluate the quality of sewing pattern generation, we design a user study interface tailored for comparison (Figure 12). For each test input—either an original image (for image-based evaluation) or a text description (for text-based evaluation)—the interface presents participants with simulated garment results generated by Design2GarmentCode and a baseline method. Participants assess the results based on two criteria: **Agreement**, which measures how well the generated patterns align with the input description, and **Aesthetic**, which evaluates the structural integrity and aesthetic appeal quality of the patterns. An “Unsure” option is available for cases where a clear preference cannot be determined, ensuring unbiased and flexible feedback.

9. LMM Prompting Details

9.1. Pattern Drafting Test

As outlined in Sec. 3.2.1, a key prerequisite for Design2GarmentCode is the presence of embedded pattern-drafting knowledge in pre-trained large models. To assess this capability, we prompted models like O1-preview and LLaMa 3.2 3B Instruct with the question, “How to draft a basic upper body bodice?”. These models produced step-by-step drafting instructions in natural language, including commands such as: “STEP 1: Take Your Measurements,” and “STEP 2: Draw the Center Front Line, Draw the Shoulder Line, Draw the Armhole, Draw the Side Seam (Measure the distance from the underbust measurement and divide it by 4. Mark this distance from the armhole point down to the waist. Draw a vertical line to represent the side seam).” Figure 13 showcases sample outputs from Llama-3.2-3B Instruct which we used as baseline for DSL-GA.

9.2. Prompting for MMUA

Based on different input design modalities and tasks, we assigned five specific tasks to the MMUA.

Task 1: Identify the image, extract answers for each prompt question based on the image, and combine them to form the recognized clothing information. This task establishes the relationships between parameters and the questions corresponding to each parameter. It serves as the foundation for all subsequent tasks.

Task 2: Generate clothing information based on text. Building on Task 1, this task generates clothing prototype information according to user preferences.

Task 3: Retrieve existing clothing information and modify the clothing design according to the user’s ideas.

Task 4: Input stress test images along with the current clothing information from the text space. MMUA interprets the colors in the image as stress levels—red, yellow, or similar colors indicate areas that are too tight. MMUA dynamically adjusts the clothing information to reduce stress.



Figure 14. Limitations of Design2GarmentCode, including failed to modeling thin structures like halter-neck, unable to model unconventional bodices and stitching relationships are limited to one-to-one mapping.

Task 5: Compare previously generated clothing simulation images, their corresponding clothing information, and the original input image. Identify differences between the simulation and the original image, and dynamically adjust the clothing information to make the final simulation image more closely resemble the original.

As discussed in Sec. 3.2.2, due to the probabilistic nature of LMMs, the MMUA struggles to accurately estimate numerical values in the design configuration. Therefore, we limit the MMUA’s task to answering multiple-choice questions, with responses formatted as a selective parameter list. List1 illustrates example parameters before (`design_cfg_num`) and after (`design_cfg_slc`) modification. The complete prompt will be publicly available with Design2GarmentCode code base.

10. Limitations

A limitation of Design2GarmentCode is its current inability to substantially modify GarmentCode’s underlying structure and logic, which impacts the generation quality due to inherent constraints in GarmentCode’s design and modeling capabilities. For example, the range of upper garment patterns is limited, making it difficult to model personalized segmentations (Figure 14 (b)). Additionally, for designs like halter necks or strapless tops (Figure 14 (a)), GarmentCode cannot model fine straps, leading to potential simulation failures. These constraints restrict the system’s ability to accurately represent certain complex or customized garment designs.

11. Additional Results

```

1  # Numerical parameters
2  design_cfg_num = [
3      # Meta Section
4      "meta.upper.v=Shirt",
5      "meta.wb.v=FittedWB", # waistband type
6      "meta.bottom.v=SkirtLevels",
7      "meta.connected.v=False",
8      # Waistband Section
9      "waistband.waist.v=1.05", # Fitted
10     "waistband.width.v=0.2", # Narrow
11     # Fitted Shirt Section
12     "fitted_shirt.strapless.v=False",
13     # Shirt Section
14     "shirt.length.v=1.55", # Long
15     ...
16 ]
17
18 # Selective parameters
19 design_cfg_slc = [
20     # Meta Section
21     "meta__upper__Shirt",
22     "meta__wb__FittedWB", # waistband type
23     "meta__bottom__SkirtLevels",
24     "meta__connected__False",
25     # Waistband Section
26     "waistband__waist__fitted", # Fitted
27     "waistband__width__narrow", # Narrow
28     # Fitted Shirt Section
29     "fitted_shirt__strapless__False",
30     # Shirt Section
31     "shirt__length__long", # Long
32     ...
33 ]

```

Listing 1. Example of original design configurations with numerical values and modified design configurations with only selective parameters.

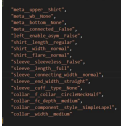
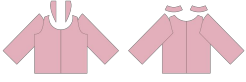




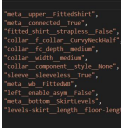
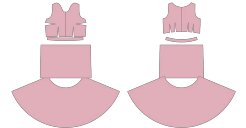

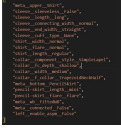



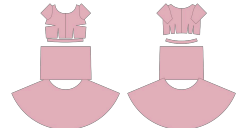

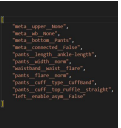
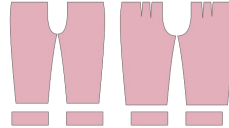


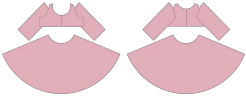


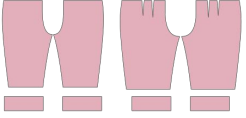


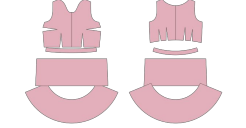

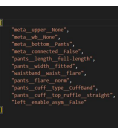
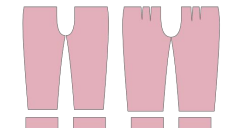

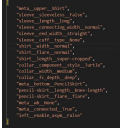



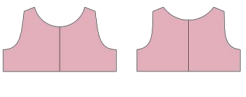


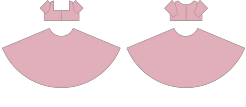




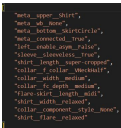
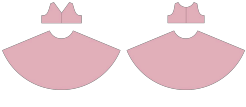


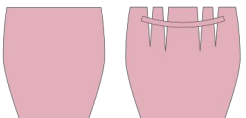

coat, long sleeves, long length				Jumpsuit, short sleeves, fitted, scoped length, knee-length, V-neck, straight leg			
Dress, empire waist, sweetheart neckline, layered skirt, sleeveless, floor-length				Outfit, long-sleeve shirt, regular length, mandarin collar, skirt, mini-length, tight fit, high waist			
Dress, floor-length, boat neck, layered skirt, tight fit, short sleeves				Pants, ankle-length, normal width, cuffed hem			
Dress, loose fit, scoop neck, knee-length, circle skirt, long sleeves				Pants, capri-length, normal width, cuffed hem			
Dress, mini-length, sleeveless, layered skirt, tight fit, halter neck				Pants, full-length, fitted width, cuffed hem			
Dress, long sleeves, knee-length, turtleneck, pencil style				Shirt, sleeveless, super-cropped length, henley style			
Dress, short sleeves, midi-length, square neck, A-line				Shirt, three-quarter sleeves, regular length, boat neck			
Dress, sleeveless, midi-length, V-neck, A-line_pattern				Skirt, above-knee length, tight fit, low waist			
Input text	Output from MMUA	Generated sewing pattern	Simulation results	Input text	Output from MMUA	Generated sewing pattern	Simulation results

Figure 15. Additional Text-guided generation results. From left to right: input text; output from MMUA; generated sewing pattern; simulation results.



Figure 16. Additional Image-guided generation results. From left to right: input image; output from MMUA; generated sewing pattern; simulation results.

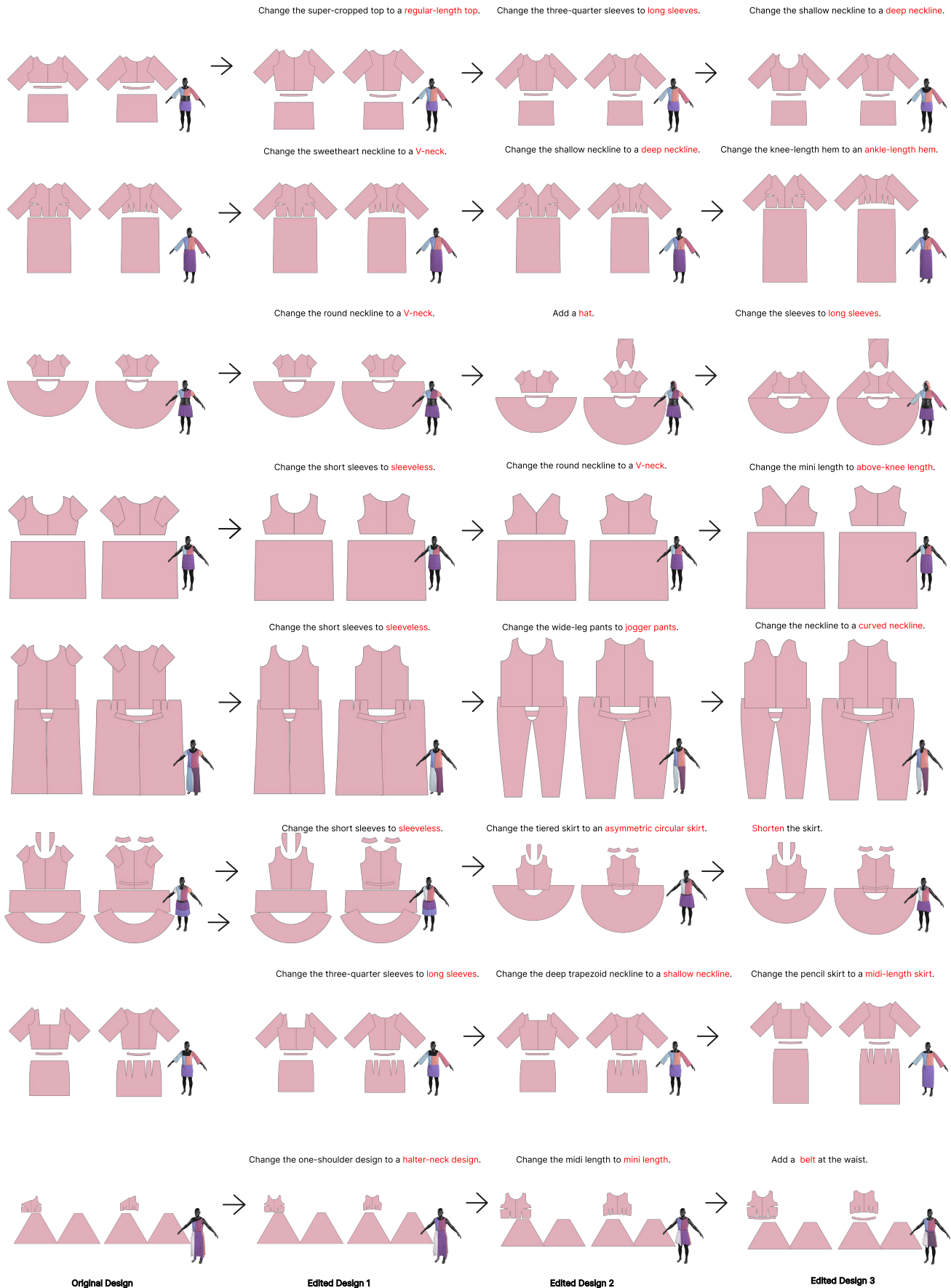


Figure 18. Additional sewing pattern editing results. Starting from the original sewing pattern on the far left, the system applies user instructions to edit the pattern. The left side of each arrow represents the original pattern, while the right side displays the edited result.