Self-Elicitation of Requirements with Automated GUI Prototyping

Kristian Kolthoff
Institute for Enterprise Systems, University of Mannheim
Mannheim, Germany
kolthoff@es.uni-mannheim.de

Christian Bartelt
Institute for Enterprise Systems, University of Mannheim
Mannheim, Germany
bartelt@es.uni-mannheim.de

Simone Paolo Ponzetto
Data and Web Science Group University of Mannheim
Mannheim, Germany
simone@informatik.uni-mannheim.de

Kurt Schneider
Leibniz Universität Hannover
Hannover, Germany
kurt.schneider@inf.uni-hannover.de

Abstract

Requirements Elicitation (RE) is a crucial activity especially in the early stages of software development. GUI prototyping has widely been adopted as one of the most effective RE techniques for user-facing software systems. However, GUI prototyping requires *(i)* the availability of experienced requirements analysts, *(ii)* typically necessitates conducting multiple joint sessions with customers and *(iii)* creates considerable manual effort. In this work, we propose *SERGUI*, a novel approach enabling the Self-Elicitation of Requirements (SER) based on an automated GUI prototyping assistant. *SERGUI* exploits the vast prototyping knowledge embodied in a large-scale GUI repository through Natural Language Requirements (NLR) based GUI retrieval and facilitates fast feedback through GUI prototypes. The GUI retrieval approach is closely integrated with a Large Language Model (LLM) driving the prompting-based recommendation of GUI features for the current GUI prototyping context and thus stimulating the elicitation of additional requirements. We envision *SERGUI* to be employed in the initial RE phase, creating an initial GUI prototype specification to be used by the analyst as a means for communicating the requirements. To measure the effectiveness of our approach, we conducted a preliminary evaluation. Video presentation of *SERGUI* at: https://youtu.be/pzAAB9Uht80

CCS Concepts

• **Information systems → Information retrieval**.

Keywords

Requirements Elicitation, GUI Prototyping, Recommendation

ACM Reference Format:
Kristian Kolthoff, Christian Bartelt, Simone Paolo Ponzetto, and Kurt Schneider. 2024. Self-Elicitation of Requirements with Automated GUI Prototyping. In *39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24), October 27-November 1, 2024, Sacramento, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3691620.3695350

1 Introduction

Numerous techniques have been proposed and employed over the years with the aim of enhancing requirements elicitation in user-facing software systems development [8]. GUI prototyping serves as such an elicitation technique offering a mechanism for analysts to visualize their comprehension of the requirements and allowing customers to validate these through a tangible artifact. Additionally, these prototypes lay the groundwork for actively involving customers during the development phase. This involvement can spark meaningful discussions leading to the clarification and refinement of requirements [2, 9]. However, conducting RE using GUI prototyping requires the availability of experienced requirements analysts and creates considerable manual efforts. Furthermore, GUI prototyping is an iterative and time-consuming process, often requiring multiple joint sessions between analysts and customers, resulting in outdated, invalid requirements [11].

In this work, we propose *SERGUI*, a novel RE approach that enables customers to perform Self-Elicitation of Requirements (SER) for user-facing software systems through automated NL-based GUI prototyping and that provides an LLM-based GUI feature recommendation mechanism to promote the automatic elicitation of additional requirements. The general notion of SER refers to guiding customers to uncover their own requirements and has originally been introduced by *LadderBot* [10], thus reducing the effort of analysts in the initial RE phase and facilitating to closely integrate customers into the RE process. By extending the notion of SER with automated GUI prototyping, we exploit the benefits of GUI prototypes as a requirements specification artifact, facilitating to obtain fast customer feedback and achieving early clarification of requirements. In *SERGUI*, customers are guided through the GUI prototyping process by a rule-based dialogue assistant which provides matching GUIs and proactively recommends potentially relevant GUI features. Our *SERGUI* prototype, source code and evaluation datasets are all available at our accompanying repository [1, 12].

2 Approach: SERGUI

SERGUI is an approach to automate the initial RE phase by leveraging a comprehensive GUI repository in combination with a GUI ranking and feature recommendation technique. Subsequently, an overview of our approach is given and depicted in Fig. 1. *SERGUI* is divided into multiple components: First, *(A)* an interaction model encompassing the essential interaction mechanisms between the customer and the automatic GUI prototyping assistant. Second, *(B)*

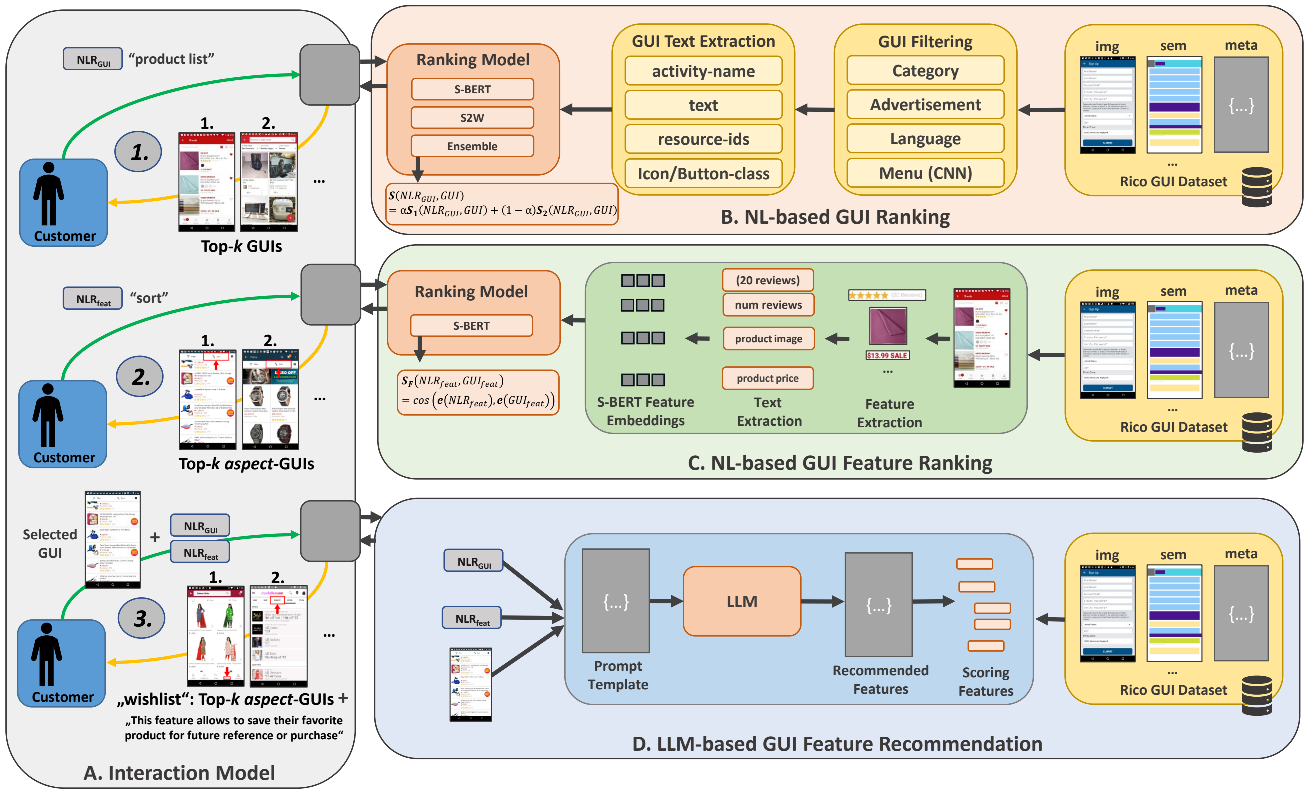


Figure 1: Overview of the *SERGUI* approach with *(A)* the interaction model showing the main interaction mechanisms, *(B)* NL-based GUI ranking approach, *(C)* the NL-based GUI feature ranking and *(D)* the LLM-based GUI feature recommendation

an NL-based GUI ranking technique building upon work of [4–6]. Third, *(C)* an NL-based GUI feature ranking mechanism. Fourth, *(D)* an LLM-based GUI feature recommendation mechanism to proactively suggest potentially relevant GUI features and simultaneously illustrate them matched on the GUIs from the top-k GUI ranking.

2.1 Interaction Model

SERGUI is built as an interactive dialogue-based approach providing guidance through the GUI prototyping process, facilitating the close integration of customers. The interaction model encompasses three essential patterns that are repeated for each GUI in the application prototype. Initially, customers can specify their NLR for a particular GUI (NLR_{GUI}), which is responded by the approach with a top-k GUI ranking, as illustrated in Fig. 1 *A1*. This ranking represents the best matches as computed by the NL-based GUI ranking model. The GUI ranking showcases many potentially relevant GUIs encompassing potentially numerous variations, which already stimulates RE. Second, customers are enabled to specify additional NLR for individual features (NLR_{feat}), which is responded by the approach with a top-k ranking of *aspect*-GUIs, as depicted in Fig. 1 *A2*. Therefore, the approach presents a ranking of matching GUIs potentially containing the formulated GUI feature. If the customer finds a relevant *aspect*-GUI, then it can be selected. Subsequently, the GUI ranking is recomputed based on the new aspects and the *aspect*-GUI will be saved as part of the GUI prototype. Third, when the customer initially selects a GUI already satisfying many features plus the specified *aspect*-GUIs, then the approach utilizes the NLR_{GUI}, collection of NLR_{feat} and the selected GUI to proactively recommend potentially relevant GUI features, as shown in Fig. 1 *A3*. In an iterative fashion, the *SERGUI* approach will then present each GUI feature with a short textual explanation and the top-k ranking of potentially matching *aspect*-GUIs encompassing and visualizing the respective GUI feature. Similarly to before, a relevant *aspect*-GUI can directly be selected, which then will be added to the overall GUI prototype specification and exploited for GUI reranking. This GUI specification is then added to a simple linear app prototype, allowing customers to briefly skip through the app.

2.2 NL-Based GUI Ranking

To achieve NL-based GUI ranking, we mainly adopt the work of [6] and provide an extension for filtering and ranking models. The GUI ranking exploits the GUI repository *Rico* [3], the largest GUI dataset for mobile apps available. We follow the GUI filtering pipeline of [6]. To filter opened menus, we trained a CNN-classifier (three convolution and pooling layers) and provided as input a combination of both the original GUI screenshot image and the semantic annotation image as grayscale variants. We trained the CNN model for 6 epochs (adagrad optimizer and binary crossentropy as loss) and achieved a satisfying performance on a separate test set (*Precision*=.9818 / *Recall*=.7012). We focused on achieving a high precision, to avoid the removal of adequate GUIs. Overall, we filtered 23,817 GUIs (with 9,363 by the CNN) resulting in 48,402 GUIs remaining.

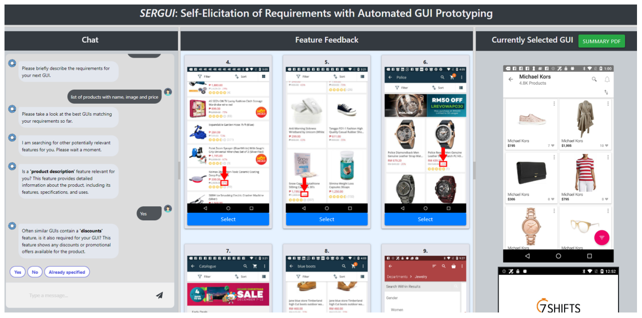
As a GUI ranking approach, we adopt the strong pretrained embedding-based *SentenceBERT* model from [6]. This model computes the ranking utilizing cosine-similarity between the embedded query and embedded textual representation of the GUI i.e. score $\mathbf{S}_1(NLR_{GUI}, GUI) = \cos(\mathbf{e}(NLR_{GUI}), \mathbf{e}(GUI))$ with $\mathbf{e}$ referring to the *SentenceBERT* embedding. Moreover, we extend this GUI ranking score by incorporating the *S2W* dataset [14], adding another artifact to the GUIs to score against. *S2W* is a large collection of manually crafted high-level NL descriptions of *Rico* GUIs, providing five descriptions per GUI from different annotators. For the *S2W* data, we similarly compute the average over the cosine-similarity between the query and all five descriptions i.e. score $\mathbf{S}_2(NLR_{GUI}, GUI) = \frac{1}{5}\sum_{i=1}^{5}\cos(\mathbf{e}(NLR_{GUI}), \mathbf{e}(S2W(GUI, i)))$. By creating an ensemble GUI ranking model using both the extracted GUI text and the high-level descriptions of *S2W* (see formula Fig. 1 *B*), we facilitate obtaining a more flexible and robust GUI ranking.

2.3 NL-Based GUI Feature Ranking

To achieve NL-based GUI feature ranking, we employ the top-k GUIs representing potentially relevant context for matching the features. Here, we restrict GUI features to individual GUI components. From the top-k GUIs, textual representations of the GUI components are extracted, including multiple texts for each component (displayed text, *resource-id* and semantic classes). These text representation are then utilized to match against NLR_{feat} again based on the cosine-similarity of the *SentenceBERT* embeddings (see Fig. 1 *C*).

2.4 LLM-Based GUI Feature Recommendation

Within the *SERGUI* approach, the GUI feature recommendations are based on few-shot prompting of an LLM (GPT-4) using the context of *(i)* the initial textual requirements denoted by NLR_{GUI}, *(ii)* a collection of already specified features by the customer as NLR_{feat} and *(iii)* an initially selected GUI from the top-k GUI ranking. With these contextual inputs, we fill in a prompt template. First, *(A)* we provide the *task instructions* asking the model to recommend the top-*30* GUI features given the described context. Next, *(B)* the template displays the initial requirements as NLR_{GUI}. Third, *(C)*, the initially selected GUI is provided to the model. Since the original XML-based GUI hierarchy contains a large amount of information, transforming the GUI hierarchy into a more abstract and focused representation is necessary. Each GUI component is transformed to a string of *"uicomp-text" (uicomp-type) (resource-id)* and then arranged in a two-level list based on grouping information (prompt details in [1]). To compute a GUI feature ranking, each of the predicated features is matched against the features in each GUI in the top-k GUI ranking. To obtain a confidence score that a GUI contains a predicated feature (NLR_{feat}) and to determine which GUI component within the GUI matches best (as *aspect*-GUI), we compute $\mathbf{S}_g(NLR_{feat}, GUI) = \max_{f \in GUI_{feats}} \mathbf{S}_F(NLR_{feat}, f)$. For each predicated GUI feature, we compute a feature score as $\mathbf{S}_{pf}(NLR_{feat}) = \frac{1}{k}\sum_{i=1}^{k} S_g(NLR_{feat}, GUI_k)$ over the top-k GUI ranking to estimate the feature coverage among top-k GUIs. The features are then recommended in the sorted order by utilizing the score $\mathbf{S}_{pf}$ and showing the top-k *aspect*-GUIs. Using a few-shot prompt, we generate text explanations for each predicted feature.



Please briefly describe the requirements for your next GUI.

list of products with name, image and price

Please take a look at the best GUIs matching your requirements so far.

I am searching for other potentially relevant features for you. Please wait a moment.

Is a **'product description'** feature relevant for you? This feature provides detailed information about the product, including its features, specifications, and uses.

Yes

Often similar GUIs contain a **'discounts'** feature, is it also required for your GUI? This feature shows any discounts or promotional offers available for the product.

Figure 2: Web app implementation of the *SERGUI* approach with a chat-section, workbench and GUI prototype preview

2.5 Feature-Based GUI Reranking

Based on positive feedback from the customer by selecting an *aspect*-GUI from the top-k *aspect*-GUI ranking, we can compute a reranking of the GUIs. Often customers may not explore much more than the top-*20* or top-*30* GUIs in the ranking, missing potentially relevant GUI features. By reranking the top-k GUIs, customers potentially find more relevant GUIs at the top of the GUI ranking or they can keep their original choice. We compute the GUI reranking as $\mathbf{S}_{RR}(GUI) = \beta\mathbf{S}(NLR_{GUI}, GUI) + (1-\beta)\frac{1}{|F|}\sum_{f \in F}\mathbf{S}_g(f, GUI)$ ensemble of GUI ranking and normalized sum over all feature scores.

2.6 Prototyping Artifact

At the end of the GUI prototyping process, *SERGUI* produces an app summary including *(i)* a visualization of the overall app and *(ii)* for each GUI the main selected GUI, the collection of *aspect*-GUIs and an additional collection of textual requirements, representing recommended features that were relevant but no *aspect*-GUI was found. This GUI prototyping artifact produced by *SERGUI* can then be employed by analysts as a starting point for further elicitation.

3 Experimental Evaluation

To assess the performance of *(i)* the feature recommendation, *(ii)* relevance of matched *aspect*-GUIs and *(iii)* feature-based GUI reranking, we conducted a small preliminary evaluation. We recruited 12 annotators possessing technical backgrounds (BSc.:7|MSc.:5). In addition, the recruited participants had medium to high experience in software development (Mean:3.50|SD:0.90), as self-reported on a five-point Likert scale, allowing them to evaluate the relevance of the recommendation results. The evaluation consisted of three applications from diverse domains (*shopping*, *news* and *social*), each encompassing three GUIs, overall resulting in nine different GUIs. Each application was presented to the participants as a low-fidelity GUI prototype, containing solely a minimal collection of features to enable participants to recognize the notion of the GUI. On this basis, participants were asked to employ *SERGUI* for prototyping and evaluate the relevance of the results. Overall, 72 GUIs were prototyped during the preliminary evaluation (six per annotator).

3.1 GUI Feature Recommendation Relevance

To evaluate the feature recommendation relevance, 720 recommendations were made (10 per GUI) and we computed *MAP*, *MRR* and *P@k*. A *MAP* of .741 indicates a strong feature recommendation performance as approximately three out of four features are marked as relevant on average. Likewise, the *MRR* of .816 describes that the first relevant feature occurs at rank 1.22 on average indicating that the top-features are relevant. In addition, also the *P@k* values indicate a substantial recommendation performance starting from .691 (*P@1*) to .638 (*P@10*). This also shows that the feature scoring accordingly ranks more relevant documents higher as intended.

3.2 GUI Feature Ranking Relevance

To evaluate the relevance of the matched *aspect*-GUIs, we employed the identical setup as described before (10 features per GUIs with top-*15 aspect*-GUIs) and for each relevant feature (either answered by *yes* or by picking an *aspect*-GUI), we employed the rank of the selected *aspect*-GUI to compute respective ranking metrics. Since we only have at maximum one single relevant *aspect*-GUI, we computed the *MRR* and *HITS@k*. A *MRR* of .390 shows that that the first relevant *aspect*-GUI appears at rank 2.56 on average indicating a substantial performance in finding relevant *aspect*-GUI matches. Moreover, starting from a *HITS@1* of .270 represents that on average in 27% a relevant *aspect*-GUI could be found at the top-1 position. A *HITS@15* of .691 represents that on average our matching approach could find a relevant *aspect*-GUI in 69.1% within the top-15 *aspect*-GUI ranking. Therefore, for a large amount of recommended features our feature matching approach is capable of finding a relevant *aspect*-GUI visualizing the predicted feature.

3.3 GUI Reranking Performance

By comparing the *SentenceBERT* (*MRR:* .172) and *S2W* (*MRR:* .210) ranking models with the ensemble model (*MRR:* .429), we see that the ensemble considerably improves the GUI ranking performance. Based on the positive feature feedback (selecting an *aspect*-GUI), we computed a reranking of the GUIs. Participants could initially select a GUI and then are presented with feature recommendations. Afterwards, participants could either chose a new GUI from the updated ranking or keep their previous choice. In 68.05% of the cases, the participants selected a better matching GUI. The reranking score (initial - updated rank) is +61.89 ranks on average (SD:79.82|Min:-26|Max:335) indicating that the feature-based reranking can substantially improve the ranks of relevant screens.

4 Related Work

The *Fast Feedback* technique [11] reduces the number of necessary RE sessions with the customer by providing a tool-based support to rapidly and interactively create pen-paper prototypes combined with use cases and directly allows to incorporate customer feedback. However, still an experienced analyst is required to conduct the initial elicitation. Recent approaches *GUI2WiRe* [4, 5] and *RaWi* [6] support analysts to rapidly create GUI prototypes via NL-based GUI retrieval, however, cannot readily be employed by customers due to the lack of knowledge and experience with GUI prototyping and hence are dependent on analysts. In the RE approach in [13], analysts create initial prototypes based on a start-up meeting with the clients, subsequently enabling the customers to modify the prototype through a web-based tool. Similarly, the *Graphical Requirements Collector* [7] requests users to directly create their own GUI prototypes by constructing them bottom-up and augmenting them with textual requirements. This integrates customers closely into the process, however, due to the lack of guidance, requires customers to have technical knowledge and experience with GUI prototyping. Moreover, *LadderBot* [10] is able to conduct dialogue-based elicitation with customers by applying laddering, a structured interview technique. However, this automatic dialogue-based elicitation approach focuses on the collection of simple textual requirements solely and therefore cannot be applied for the initial elicitation with GUI prototyping. Consequently, this technique cannot provide fast feedback to customers on the basis of GUI prototypes like *SERGUI*.

5 Conclusion and Future Work

In this work, we proposed *SERGUI* as the first approach towards enabling the Self-Elicitation of Requirements with GUIs for customers. The evaluation results show that *SERGUI* is able to effectively support users during the GUI prototyping process in terms of providing relevant GUIs on the basis of NLR, recommending relevant GUI features and their visualizations. For future work, we plan to conduct a large user study to more comprehensibly evaluate our approach.

References

[1] [n. d.]. SERGUI - Self-Elicitation of Requirements with GUIs GitHub. https://github.com/SERGUI-Prototyper/SERGUI-Prototyping. Accessed: 2024-06-20.

[2] Michel Beaudouin-Lafon and WE Mackay. 2002. Prototyping development and tools. *Handbook of Human-Computer Interaction* (2002), 1006–1031.

[3] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 845–854.

[4] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2020. GUI2WiRe: Rapid Wireframing with a Mined and Large-Scale GUI Repository using Natural Language Requirements. In *35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20)*. ACM.

[5] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2021. Automated Retrieval of Graphical User Interface Prototypes from Natural Language Requirements. In *International Conference on Applications of Natural Language to Information Systems*. Springer, 376–384.

[6] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. 2023. Data-driven prototyping via natural-language-based GUI retrieval. *Automated Software Engineering* 30, 1 (2023), 13.

[7] J Michael Moore and Frank M Shipman. 2000. A comparison of questionnaire-based and GUI-based requirements gathering. In *Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering*. IEEE, 35–43.

[8] Klaus Pohl. 2010. *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated.

[9] Alon Ravid and Daniel M Berry. 2000. A method for extracting and stating software requirements that a user interface prototype contains. *Requirements Engineering* 5, 4 (2000), 225–241.

[10] Tim Rietz and Alexander Maedche. 2019. LadderBot: A requirements self-elicitation system. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 357–362.

[11] Kurt Schneider. 2007. Generating fast feedback in requirements elicitation. In *International working conference on requirements engineering: Foundation for software quality*. Springer, 160–174.

[12] SERGUI - Self-Elicitation of Requirements with GUIs - website [n. d.]. http://www.sergui-tool.com/sergui/. Accessed: 2024-06-20.

[13] Leonor Teixeira, Vasco Saavedra, Carlos Ferreira, João Simões, and Beatriz Sousa Santos. 2014. Requirements engineering using mockups and prototyping tools: developing a healthcare web-application. In *International Conference on Human Interface and the Management of Information*. Springer, 652–663.

[14] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile UI summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 498–510.