proceedings

MDPI

Proceedings

Machine Learning Methods for Inferring Interaction Design Patterns from Textual Requirements [†]

Viridiana Silva-Rodríguez [1,*], Sandra Edith Nava-Muñoz [1], Luis A. Castro [2], Francisco E. Martínez-Pérez [1], Héctor G. Pérez-González [1] and Francisco Torres-Reyes [1]

[1] Autonomous University of San Luis Potosí, 78300 San Luis Potosí, Mexico; senavam@uaslp.mx (S.E.N.-M.); fcoemtz@gmail.com (F.E.M.-P.); hectorgerardo@yahoo.com (H.G.P.-G.); francisco.torres@uaslp.mx (F.T.-R.)

[2] Sonora Institute of Technology (ITSON), 85000 Ciudad Obregón, Mexico; luis.castro@acm.org

[*] Correspondence: vsilva@alumnos.uaslp.edu.mx

[†] Presented at the 13th International Conference on Ubiquitous Computing and Ambient Intelligence UCAmI 2019, Toledo, Spain, 2–5 December 2019.

Published: 20 November 2019

Abstract: Ambient intelligence is one of the most exciting fields of application for pervasive, wireless, and embedded computing. However, the design and implementation of real-world systems must be conducted utilizing software engineering approaches. Some types of environments (hospitals, older adults homes, emergency scenarios, etc.) are particularly critical, especially in terms of the issues concerning expressing requirements, verifying and validating them, or ensuring functional correctness. To provide adequate ambient intelligence solutions, it is necessary to place special emphasis on obtaining, specifying, and documenting software requirements. To address this issue, our paper presents a model that integrates both requirements and design patterns. This is done through a natural language processing application in conjunction with other artificial intelligence algorithms. This work aims to support designers when analyzing text requirements and support design decisions. Our results were evaluated according to the cross-validated accuracy of predicting design patterns. The results obtained indicate that this approach could lead to good recommendations of design patterns, as it demonstrated an acceptable classification performance over the balanced dataset of requirements instances.

Keywords: interaction design; interaction patterns; requirements; machine learning

1. Introduction

Ambient intelligence (AmI) is the integration of ubiquitous computing, ubiquitous communications, and the application of user interfaces (UIs). The objective of AmI is to design and implement new systems that provide intelligent, personalized, and connected services [1]. One of the main components of an ambient intelligence platform is the way it relates to humans, and vice versa. To achieve this relationship, AmI can aim for this interaction to be performed in a non-intrusive way, minimizing explicit interaction. Thus, in situations in which explicit interactions are required, there is a need to design a set of physical interactions that include environment and communication, among other characteristics. For the set of interactions to be designed, we need to apply interaction design techniques.

Interaction in AmI has followed the user-centered design approach. The aim of this approach is not only to design products or user experiences, but to understand the relationship of the product with the motivations of the final user, its use, its context, and user needs. However, some types of environments (hospitals, older adults homes, emergency scenarios, etc.) are particularly critical, especially due to the issues concerning expressing software requirements, verifying and validating them, or ensuring functional correctness. This is due to the different behaviors and activities that are executed in these types of scenarios. To meet this challenge, and to understand and write the needs of the various scenarios, AmI designers have to collaborate with experts from other disciplines.

To provide adequate AmI solutions, it is necessary to place special emphasis on obtaining, specifying, and documenting software requirements. These requirements are based on normative, social, and technical aspects and must be transferred into functional requirements that can be used for system development. According to Coronato et al. [2,3], the specification of requirements allows designers to detect and eliminate faults from the beginning of the design process.

Within the software design process, some problems are recurrently found, for which it is valid to reuse solutions [4]. Much of the current related work on patterns stems from the work of Alexander et al. [5] covering the design and layout of buildings, towns, and communities. In software engineering and human–computer interaction, patterns do not just describe recurring situations and their solutions, but they describe a method for presenting the situation and solutions in a structured way [6].

Recurring situations are similar for many AmI application development projects, for which using design patterns based on requirements seems to be a suitable solution. However, the literature about integrating both requirements and design patterns is scarce. Thus, their relation in practical contexts has not been sufficiently highlighted. This is due to questions that arise regarding the application of design patterns, such as How do you select the required design patterns? Moreover, How do you evaluate the process of using design patterns? The creativity and skills involved in designing UIs can be subjective and error-prone.

In this paper, a field-tested approach that combines both concepts is proposed. We present a model (IDPatternM) for inferring patterns of interaction design from the text processing of the requirements. This will allow designers to save time when analyzing text requirements and support design decisions. The proposal consists of the design of a recommendation system based on the IDPatternM model through the application of artificial intelligence (AI) algorithms.

In Section 2, related work is discussed to lay the foundations of the proposed model (IDPatternM). In Section 3, the process of searching for suitable design patterns for specific requirements is explained in detail, which is our main contribution. Section 4 presents the methodology that was followed in the development of the interaction patterns recommendation module. Section 5 describes the evaluation of the prediction of suitable design patterns for specific requirements. Finally, Section 6 concludes with a summary and a list of future work.

2. Literature Review

In recent years, some work has been published on the integration of both requirements and design patterns. We briefly present a selection of works that focus on integrating both requirements and design patterns.

Given the heterogeneity of current interactive systems, approaches such as model-based user interface development (MBUID) help reduce the gap between requirements and implementation. The above is achieved through the definition of models related to the UI that are captured and updated throughout the development life cycle. One of these models is the task model, which is useful when designing and developing interactive systems as it describes the logical activities that must be carried out to achieve the objectives of users [7]. However, the designer must enter specific metadata to relate to the information contained in the model-based system.

Several model-based tools have been developed to present and test a possible solution [8–11]. First, PD-MBUI [9] (UI based on patterns and UI based on models) proposes to integrate the patterns in the design process. The starting point of this approach is a clear description of the users' requirements, which includes answers to questions such as Who are the users?; What are their tasks?; In what environment will the system be used? The answers to these questions are essential to building models that reflect the users and their real needs in any scenario. However, they approach the problem by defining a structure for the design patterns and generalizing in the generation of an ontology, through which information is inferred based on an input defined by the designer.

Other approaches such as [8,10,11] consider the identification of patterns based on an ontology with the same structure of the design pattern and its design patterns catalog. Despite this, some approaches specifically address the natural language processing (NLP) of requirements. One of these is that of Navarro-Almanza et al. [12], who propose a model using deep learning techniques to represent text data in low-dimensional vector space and classify requirements in 12 different categories, which is lower than the NLP approaches to requirement classification. Moreover, design patterns are not addressed in this proposal.

There are other attempts at the automatic classification of software requirements using the PROMISE corpus, such as [13–15]. In general, approaches are based on NLP and information-extraction techniques to define characteristics for applying machine learning techniques. Some approaches are based on word frequencies and probabilistic methods, such as [16,17]. However, the requirements processing is not related to the design patterns or to supporting designers of interactive systems.

Gaffar et al. [18] propose a "disseminator pattern" that can be a person or a software agent and that interacts with the authors of the design patterns to support them in a semi-structured generation of the designs, in such a way that they become software tools that help the designers find, filter, and combine the patterns in a new design. The work of Gaffar et al. is based on an underlying XML database that, in turn, feeds back to it. Furthermore, it requires a good structuring of the data of the patterns and queries that represent most user needs when consulting design patterns. However, the requirements are not considered.

On the other hand, to support design decision making, which is subjective and error-prone, the approach presented in this paper consists of the construction of a recommendation system based on a requirements classification model to support designers in the process of designing interactive systems, through the inference of the related patterns in a set of requirements data. Instead of relying on manually crafted rules, text classification with machine learning learns to make classifications based on past observations. While we show prediction accuracy in the context of UI design, it is possible to extend our framework through a greater variety of requirements and design patterns. Therefore, it is possible to apply it in other application scenarios.

The works identified and described in the previous paragraphs are important but isolated efforts. To be able to specify the selection of patterns of interaction design, it is necessary to analyze what is needed and how to design an appropriate interactive system, for which the design patterns derived from requirements are useful to bridge gaps in the early phase of system development, where recurring requirements call for similar solutions. In the following section, a proposal for a model of interaction design pattern recommendation (IDPatternM) is described.

3. IDPatternM: A Model of Interaction Pattern Inference Based on Software Requirements

IDPatternM is a model for the identification of design patterns based on requirements and is defined in this work (see Figure 1). The IDPatternM model arises from a case study [19] in which a series of metadata are considered in the design decision making by interactive system designers. As a point of entry to the model, a requirement is needed in text, which is entered into the text editor module (see Figure 1a). Here, a verification is carried out regarding the integrity of the metadata (e.g., the user, action, objects) and the grammatical structure for text processing.

Considering the advantages of NLP and formal representations, the proposal includes text verification based on a semantic parsing task [20]. For instance, given the requirement "A user shall be able to enter information on an incident, including location, description, and time period", we found that "user", "information on an incident", "location, description and time period", and "to enter" were specified as properties. The structure of these properties is similar to sentences that are grammatically adequate.

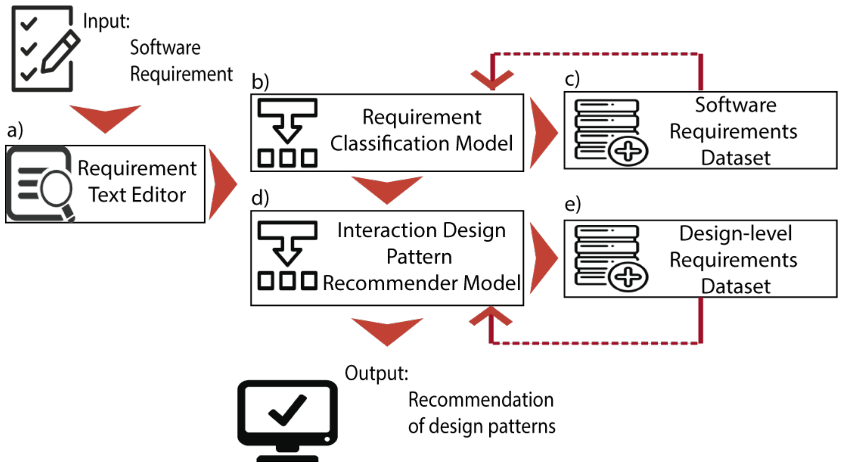

Figure 1. IDPatternM: Interaction pattern recommendation model based on requirements.

Once the requirement structure has been validated, this is classified into a type of requirement: functional or non-functional. The requirement classification model module (see Figure 1b) consists of a trained semi-supervised learning algorithm with a set of requirements data associated with a type of requirement (see Figure 1c), namely, the software requirements dataset. The functional requirements are the declarations of the services provided by the system, the way in which the system must react to particular inputs. An interactive system also involves non-functional requirements that describe quality criteria. However, this work focuses on functional requirements only, which specify what a system can do through interaction objects.

The requirement classification model module generates, as an output, a design pattern prediction. In this case, the example prediction is the "structured format" pattern, which is established as the following problem: "The user needs to enter data quickly in the system, but the format of the data must adhere to a predefined structure". Furthermore, the model will be fed back with the prediction.

The interaction design pattern recommender model (see Figure 1d) was generated with a design-level requirements dataset (see Figure 1e) on which AI algorithms were used for text classification. The following section describes each module of the IDPatternM model.

4. Methods

For the inference of interaction design patterns from the requirements text, a set of data requirements was established, for which texts were collected from various sources such as the PROMISE corpus dataset [21], which is the most popular dataset of software requirements. These requirements describe the actors, objects, and resources of the system over which they act, as well as the operations carried out by a user (transitive or non-transitive). Subsequently, these requirements were classified based on the design patterns of the Toxboe collection [22] (see Table 1).

The requirements dataset proposed contains 307 requirements related to design patterns: table filter (26), dashboard (20), search filters (22), sort by column (24), morphing controls (46), structured format (65), notifications (24), forgiving format (42), and input prompt (38).

Table 1. Patterns summary.

	Problem
Table Filter	The user needs to categorically filter the data shown in the tables by columns.
Dashboard	The user wants to digest data from multiple sources at a glance.
Sort By Column	The user needs to be able to sort the data in a table according to the values of a column.
Morphing Controls	The user wants to only be presented with controls available in the current mode.
Search Filters	The user needs to conduct a search using contextual filters that narrow the search results.
Structured Format	The user needs to quickly enter data into the system, but the format of the data must adhere to a predefined structure.
Notifications	The user wants to be informed about important updates and messages.
Forgiving Format	The user needs to quickly enter data into the system, which then in turn interprets the user's input.
Input Prompt	The user needs to enter data into the system.

Experiments

A model of textual requirements and design pattern classification (IDPatternM) was trained with the naive Bayes multinomial algorithm. The multinomial distribution usually requires the counting of entire entities. However, fractional counts like TF–IDF (term frequency–inverse document frequency) are also possible for the processing of requirements. This method [23,24] represents term frequency (TF) $\times$ inverse document frequency (IDF). In addition, weighting is commonly used in text extraction and information retrieval to evaluate the importance of a linguistic term (usually unigram or bigram) in a dataset. The importance of the term (weight) increases with the frequency of the term in the text, but it is compensated by the frequency of the term in the domain of interest (e.g., frequent words like "the" or "for" will be reduced).

To train a supervised classifier, the first step is to transform the requirement text into a vector of numbers. Subsequently, the vector representations were explored as TF–IDF weighted vectors. Once these representations of text vectors were generated, it was possible to train supervised classifiers with requirements and predict the design pattern in which they will be classified. After all of the previous data transformation, all features and labels (design pattern collection) were used to train the classifiers. There are several algorithms to perform training for this type of problem. The modality used was 75–25 (that is, 75% of the data from each participant was used for training and 25% for validation).

Once the requirements text classification model was executed, it was compared with other learning models, from which cross-validation was obtained based on its accuracy and the source of any potential problem. Comparative evaluation was carried out for the following four classification models: logistic regression, multinomial naive Bayes, vector linear support machine (LinearSVM, and random forests.

These four classification models were validated in the basic approach called cross-validation k-fold, which divides a training set into k smaller sets. In this case, a k-fold value of 5 was established. For each experiment, accuracy, precision, and recall are reported as classification performance measures.

5. Results

From the TF–IDF score, the terms that are most correlated with each of the design patterns were found using the chi-square test for feature selection. This test is used for categorical characteristics in a dataset. The chi-square between each requirement and the design pattern was calculated, from which the unigrams and bigrams of the entities with the best chi-square scores were subsequently displayed (see Table 2). Likewise, the test determines if the association between two categorical variables of the sample reflect their real association in the population.

Table 2. Unigrams and bigrams of the functional requirements dataset.

	Unigrams	**Bigrams**
Table Filter	summary, printable	printable summary, able display
Dashboard	display, events	shall provide, shall display
Sort By Column	recycled, audit	audit report, report shall
Morphing Controls	prevent, users	list players, allow player
Search Filters	bookmarks, search	allow user, shall search
Structured Format	delete, clinical	able create, able delete
Notifications	changes, notify	notify affected, shall notify
Forgiving Format	washing, functionality	leads washing, washing functionality
Input Prompt	record, allow	shall allow, shall record

These unigrams and bigrams demonstrate the congruence that exists in the definition of the problem of each design pattern with each text of functional requirement. For this, it is necessary to have a reliable dataset, that is, with classifications of functional requirements validated by expert interactive system designers.

After the data transformation process mentioned above, the classifiers were trained based on the requirements and type of design pattern. These were evaluated based on the logistic regression, multinomial naive Bayes, linear vector support machine (LinearSVM), and random forests algorithms. As a result, the LinearSVM and multinomial naive Bayes algorithms work better than the other two classifiers. LinearSVM presents a slight advantage, with a median accuracy of around 38% (see Table 3).

Table 3. Accuracy of classification algorithms.

	Logistic Regression	**Multinomial Naive Bayes**	**LinearSMV**	**Random Forests**
Accuracy	0.34	0.36	0.38	0.33

From this result with the model (LinearSVM), the confusion matrix was generated (see Table 4), in which the discrepancies between the predicted and real labels can be observed. The vast majority of predictions end in the diagonal (predicted equal to real). However, a series of erroneous classifications are shown, which are requirements that affect more than one design pattern.

Table 4. Confusion matrix of LinearSVM algorithm in the requirements classification.

Predicted									**Actual**
TF	**DB**	**SByC**	**MC**	**SFilters**	**SFormat**	**Notif**	**FFormat**	**IPrompt**	
2	0	1	2	0	0	0	0	0	**TF**
3	1	0	1	0	0	0	0	0	**DB**
1	0	2	1	1	0	0	2	0	**SByC**
1	0	0	12	0	4	0	0	1	**MC**
2	0	2	2	1	1	0	0	1	**SFilters**
0	0	0	2	0	20	1	0	0	**SFormat**
0	0	0	2	0	1	4	1	0	**Notif**
0	0	0	2	0	2	0	6	0	**FFormat**
0	0	0	3	0	2	0	3	5	**IPrompt**

Finally, a classification report and a representation of the main classification metrics by class (a type of design pattern) (see Table 5) were obtained. The classification metrics provide deeper insight into the behavior of the classifier, in terms of the overall accuracy, that can mask the functional weaknesses in a class of multi-class problems.

Table 5. Metrics report for the requirements classification by design patterns.

	Precision	**Recall**	**f1-Score**	**Support**
Table Filter	0.22	0.40	0.29	5
Dashboard	1.00	0.20	0.33	5
Sort By Column	0.40	0.29	0.33	7
Morphing Controls	0.44	0.67	0.53	18
Search Filters	0.50	0.11	0.18	9
Structured Format	0.67	0.87	0.75	23
Notifications	0.80	0.50	0.62	9
Forgiving Format	0.50	0.60	0.55	10
Input Prompt	0.71	0.38	0.50	13
avg / total	0.59	0.54	0.52	98

The classification reports that 59% of data was predicted accurately. The "dashboard" (100%) and "notification" (80%) design patterns obtained were predicted best. However, the number of samples for each one was low.

6. Conclusions

This research work describes a model for the recommendation of design patterns based on requirements (IDPatternM) for developing interactive systems in AmI. Due to situations in which explicit interactions are required, it is necessary to design a set of physical interactions that include the environment and communication, among other characteristics. The use of design patterns supports compliance with the principles of usability and user experience, thus promoting compliance with standards and the use of best practices. However, it is a challenge to find the right pattern when the requirement is ambiguous or the context of use is not specified. Due to these problems, this work proposes a IDPatternM to initially formulate grammatically correct requirements so that they can be processed and related to design patterns. Thus, they have to be translated, as well as classified according to their functionality to associate them with the corresponding pattern, according to a knowledge base.

For the evaluation of the IDPatternM module, a study was carried out with four learning models, in which its accuracy and the potential source were obtained as metrics. Although the results of the experiments determined that the linear vector support machine method is the most appropriate, a challenge in this work is to determine the knowledge base because for the moment, it is formed based on the literature and not on opinions of experts in the field.

The scope of the application developed is for a collection of patterns that cover the design of UIs. Even so, the IDPatternM model is more ambitious in the sense of having a knowledge base and considering different collections of patterns to cover the requirements of any type of system. As a future work, we propose to evaluate the recommendation module with expert designers who work with design patterns in their daily practice.

Acknowledgments: The authors wish to thank the National Council of Science and Technology (CONACYT) for its support with grant No. 246970.

References

1. Cook, D.J.; Augusto, J.C.; Jakkula, V.R. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive Mob. Comput.* **2009**, *5*, 277–298.
2. Coronato, A.; Pietro, G.D. Formal Design of Ambient Intelligence Applications. *Computer* **2010**, *43*, 60–68. doi:10.1109/MC.2010.335.

3. Coronato, A.; Paragliola, G. A structured approach for the designing of safe AAL applications. *Expert Syst. Appl.* **2017**, *85*, 1–13. doi:10.1016/j.eswa.2017.04.058.
4. Bradley, M.; Kristensson, P.O.; Langdon, P.; Clarkson, P.J. Interaction Patterns: The Key to Unlocking Digital Exclusion Assessment? In *International Conference on Applied Human Factors and Ergonomics*; Springer: Cham, Switzerland, 2018; pp. 564–572.
5. Alexander, C. *A Pattern Language: Towns, Buildings, Construction*; Oxford University Press: Oxford, UK, 1977.
6. Riehle, D.; Züllighoven, H. Understanding and using patterns in software development. *Tapos* **1996**, *2*, 3–13.
7. Calvary, G.; Coutaz, J. Introduction to model-based user interfaces. *Group Note* **2014**, *7*, W3C.
8. Thanh-Diane, N.; Vanderdonckt, J.; Seffah, A. UIPLML: Pattern-based engineering of user interfaces of multi-platform systems. In Proceedings of the 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), Grenoble, France, 1–3 June 2016; pp. 1–12.
9. Seffah, A.; Taleb, M. Tracing the evolution of HCI patterns as an interaction design tool. *Innov. Syst. Softw. Eng.* **2012**, *8*, 93–109.
10. Vanderdonckt, J.; Simarro, F.M. Generative pattern-based design of user interfaces. In Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems, Berlin, Germany, 20 June 2010; pp. 12–19.
11. Engel, J.; Märtin, C.; Forbrig, P. *A Concerted Model-Driven and Pattern-Based Framework for Developing User Interfaces of Interactive Ubiquitous Applications*; LMIS@ EICS: Duisburg, Germany, 2015; pp. 35–41.
12. Licea, G. Towards supporting Software Engineering using Deep Learning: A case of Software Requirements Classification. In Proceedings of the 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT), Mérida, México, 25–27 October 2017; pp. 116–120.
13. Delgado-Solano, I.P.; Núñez-Varela, A.S.; Pérez-González, G.H. Keyword Extraction from Users' Requirements Using TextRank and Frequency Analysis, and their Classification into ISO/IEC 25000 Quality Categories. In Proceedings of the 2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT), San Luis Potosi, México, 24–26 October 2018; pp. 88–92.
14. Ghotra, B.; McIntosh, S.; Hassan, A.E. Revisiting the impact of classification techniques on the performance of defect prediction models. In Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, Italy, 16–24 May 2015; Volume 1, pp. 789–800.
15. Ghotra, B.; McIntosh, S.; Hassan, A.E. A large-scale study of the impact of feature selection techniques on defect classification models. In Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, 20–21 May 2017; pp. 146–157.
16. Casamayor, A.; Godoy, D.; Campo, M. Semi-Supervised Classification of Non-Functional Requirements: An Empirical Analysis. *Inteligencia Artif. Rev. Iberoamericana De Inteligencia Artificia* **2009**, *13*, 35–44. ISSN 1137-3601, doi:10.4114/ia.v13i44.1044.
17. Cleland-Huang, J.; Settimi, R.; Zou, X.; Solc, P. The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. In Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis/St. Paul, MN, USA, 11–15 September 2006; pp. 39–48. doi:10.1109/RE.2006.65.
18. Gaffar, A. Application-agnostic interactive big data: Managing HCI complexity at the source. *Int. J. New Comput. Archit. Their Appl. (IJNCAA)* **2014**, *4*, 1–16.
19. Silva-Rodríguez, V.; Nava-Muñoz, S.E.; Martínez-Pérez, F.E.; Pérez-González, G.H. How to Select the Appropriate Pattern of Human-Computer Interaction?: A Case Study with Junior Programmers. In Proceedings of the 2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT), San Luis Potosi, México, 24–26 October 2018; pp. 66–71.
20. Diamantopoulos, T.; Roth, M.; Symeonidis, A.; Klein, E. Software requirements as an application domain for natural language processing. *Lang. Resour. Eval.* **2017**, *51*, 495–524.
21. The promise repository of empirical software engineering data, 2015. Available online: https://zenodo.org/record/268452#.Wp8O4uejnIU (accessed on 24 May 2019).
22. User Interface Design Patterns, 2007. Available online: http://ui-patterns.com/ (accessed on 24 May 2019).

23. Salton, G.; Yang, C.S. On the specification of term values in automatic indexing. *J. Doc.* **1973**, *29*, 351–372.
24. Jones, K.S. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* **2004**, *60*, 493–502.