Received: 13 August 2023 | Revised: 1 April 2024 | Accepted: 29 May 2024

DOI: 10.1002/smr.2701

RESEARCH ARTICLE - METHODOLOGY

Journal of Software: Evolution and Process WILEY

Improving an emergency repair process with feature models

Karam Ignaim [1] | João M. Fernandes [2]

[1]Department of Software Engineering, Prince Abdullah bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University (BAU), As-Salt, Jordan

[2]Centro ALGORITMI/Dep. Informática, Universidade do Minho, Braga, Portugal

Correspondence
Karam Ignaim, Department of Software Engineering, Prince Abdullah bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University (BAU), As-Salt, Jordan.
Email: karam.ignaim@bau.edu.jo

Funding information
Fundação para a Ciência e a Tecnologia

Abstract

Whenever an emergency occurs, such as a change in the system operating environment that prevents regular functioning, one possibility to restore the system to normal operation is to perform immediately the emergency repair process (ERP). This paper introduces the feature-based emergency repair process (FbERP), which constitutes a feature-based improvement/extension to ERP. FbERP uses feature models to represent urgent change requests (UCRs) and traces each change to its location in the code base in order to realign documentation and code. Based on real-world change requests, we evaluated the proposed process using different techniques, including an empirical case study and a small survey. In the empirical case study, we compare FbERP with respect to ERP to check the effectiveness, efficiency, and usability of the proposed process. The results of the evaluation reveal that FbERP can be successfully used by software engineers to respond to UCRs, with an improvement over ERP.

KEYWORDS
emergency repair process, feature model, urgent change request

1 | INTRODUCTION

Managing software changes is one of the most essential aspects of software development and maintenance.[1–3] Change requests are common in software maintenance and can often be urgent in their nature. In order to effectively respond to change requests, a reliable and efficient process is necessary.[4,5] Change requests are normally addressed using the standard change request process.[3,6]

An urgent change request (UCR) to a system refers to a change request that requires immediate attention and implementation due to either the criticality or the impact to the business supported by that system. UCRs need to be prioritized and assigned to a dedicated team of software engineers who are expected to be knowledgeable about the system. The team must analyze the UCR, assess its impact to the software, and work on implementing the necessary changes within a specified, usually short, timeframe. This process is conducted to ensure that UCRs are addressed promptly and with a minimal disruption to the overall system functionality.

Change requests often involve issues that must be tackled urgently using the emergency repair process (ERP).[7] In order to maintain regular operations, UCRs are triggered, which may occur under specific circumstances, like:

1. when a severe system error occurs[8,9];
2. whenever changes to the operational environment of the system have undesired/unanticipated effects; and
3. when the business operating the system experiences unanticipated changes, such as new competitors or the introduction of new legislation that affects the system.

In situations similar to these ones, the urgency of the change may prevent the software engineers from following the formal change analysis process. ERP may involve the quick adoption of a workable solution rather than a formal solution (obtained through a formal change analysis

process) which aims to identify and address the root cause of the system fault.[3] It may be necessary to temporarily bypass specific procedures in order to restore functionality and minimize downtime. In addition, UCRs may necessitate immediate coordination and communication among various teams or departments to ensure a prompt resolution and prevent further system disruptions. These situations emphasize the need for a flexible and adaptable approach to effectively manage unforeseen system issues.

This paper presents the feature-based emergency repair process (FbERP), an extension to ERP. The process is designed for software engineers and system administrators to efficiently manage UCRs for software systems. FbERP provides a framework for (1) identifying UCRs, (2) ensuring the necessary coordination and communication between teams, and (3) aligning documentation and the code base in relation to each UCR. When software engineers intend to document the change in requirements and design, and an UCR necessitates additional emergency repairs, this misalignment may occur. Eventually, the original modification is forgotten, and the system documentation and code base are never realigned. With FbERP, software engineers can effectively manage unexpected problems and minimize outages, thereby improving the overall stability and dependability of their systems.

2 | RELATED WORK

Many research works address the challenges associated with change requests within the development of software systems. In this section, we are particularly interested in approaches where feature models (FMs) are used in maintenance/evolution scenarios for modifying a given system.

Botterweck et al.[10] propose the combined use an Evolution Plan and an evolution-oriented feature model (EvoFM). The basic idea of their approach is to model the evolution of software product lines (SPLs) by focussing on commonalities and variabilities within the FMs over time and describing these variabilities with a special form of FM.

Guo and Wang[11] explore the consistent evolution of FMs to resolve the requested changes and maintain the FMs consistent. According to the definition of FMs, they first analyze the primitive elements of FMs and suggest a set of atomic operations on them. Then, they analyze and apply the semantics of change to FMs to support consistency maintenance during the evolution of the FMs. The resolution of a requested change to an FM requires obtaining and executing a sequence of additional changes derived from the requested change for keeping the consistency of the FM. Their approach limits the maintenance of an FM in a local range, affected only by the requested change, instead of the whole FM, which reduces the effort and improves the efficiency for the evolution of FMs.

An automated method, proposed by Maâzoun et al,[12] analyses the evolution of an FM, traces its impact on the design of the SPL, and offers a set of recommendations to ensure the consistency of both models. Their method defines metrics adapted to the evolution of SPLs to identify the effort needed to maintain the SPL models consistently.

Another relevant work in this field is the approach for the dynamic reconfiguration of software product families suggested by Gomaa and Hussein[13]. A reconfiguration pattern is a solution to a problem in a software product family where the configuration needs to be updated while the system is operational. It defines how a set of components participating in a software pattern cooperate to change the configuration of a system from one configuration to another.

While the previously mentioned articles and other relevant ones[14,15] are pertinent, our work is the first that we are aware of that advocates the use of FMs in the management of change requests as software systems evolve.

3 | FEATURE MODELS

An FM serves as a comprehensive representation of the features present in a software system and the relationships among them. An FM offers a systematic depiction of the features of a given set of software systems. FMs are widely applied in managing SPLs, offering a structured approach to handle variability and customization within a family of related products. They also serve as valuable inputs for generating diverse assets, including documentation, architectural designs, and code snippets. This versatility makes FMs instrumental not only in capturing and visualizing the inherent complexity of a software system but also in facilitating the downstream processes of development, ensuring consistency and coherence across various aspects of the software life cycle.

FMs are represented by feature diagrams, whose essential notations are shown in Figure 1. Features represent the functionalities or characteristics of the software product. Each feature can be either mandatory or optional. An optional feature may or may not be included in a particular

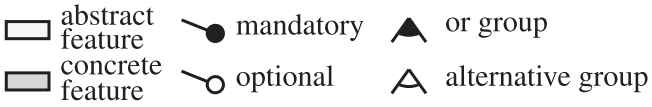

FIGURE 1 The notations of FMs.

product variant, while a mandatory feature is always present in every product variant. Abstract features represent general or high-level characteristics of the software system. They are used to structure an FM, but they do not have any impact at the implementation level. Contrarily, a concrete feature represents a specific functionality of the software system and is directly implementable. Typically, concrete features are at the lowest level in the hierarchy of the FM. Groups are used to represent variability among features. An alternative group represents a set of features, where only one feature can be selected for inclusion in a particular product variant. In an OR group, one or more features can be selected for inclusion in a product variant.

In the context of software development, an FM captures the different combinations of features that a set of software products can have. It helps in managing and visualizing the variability of features, allowing stakeholders to understand and configure the product based on their specific needs.[16] FMs are frequently represented using a tree-like structure, with the root node representing the core product and the child nodes representing optional or alternative features.[17] By selecting or deselecting specific features, it is possible to construct multiple product configurations or variants.

Figure 2 depicts an example of an FM. The FM represents various features and their relationships on a mobile phone. The root node is Mobile Phone, and it has different features representing different aspects of the phone, such as Calls, Messages, Screen, GPS, Software, and Media features. Calls, Messages, Screen, and Software are mandatory features, while Media and GPS are optional features. Basic, Color, and High resolution have alternative relationships with the Screen feature. The Media feature contains Camera and MP3, or both, because they form an Or relationship. Each feature (parent feature) can be further expanded into specific features (child features). For instance, the Software feature includes features like OS, UI, and Applications, indicating that these are related to software functionalities. The FM specifies the relationships and dependencies among the features. For instance, the Software feature has a parent-child relationship with the OS. In a real-world scenario, the FM for a mobile phone can be significantly more detailed and comprehensive, capturing a wide variety of features and their relationships. This FM provides a standardized representation of the numerous mobile phone features. Different combinations of these features can be chosen to produce various variants or configurations of the mobile phone, based on the needs and preferences of users.

4 | FEATURE-BASED ERP

This section presents how FMs can be used within ERP, since they assist software engineers in managing and prioritizing change requests. It is assumed that software developers have a basic understanding of FMs in order to profit from our approach. It is recommended to offer them training if they are unfamiliar with FMs. By doing this, the software engineers are able to prioritize tasks and to put suggested modifications into practice, which enhances the maintenance process as a whole.

The typical change request process, presented in Figure 3, consists of a series of activities beginning with the submission of a change request and concluding with a new version of the system that incorporates all requested changes.[2] During the change implementation phase of the typical process, the system specification, design, and implementation should be modified to reflect the system changes.[7] The new requirements are proposed, analyzed, and validated to correctly reflect the alterations to the system. The redesigned and implemented system components are then tested again. As part of the analysis of the change, the system prototyping may be carried out.

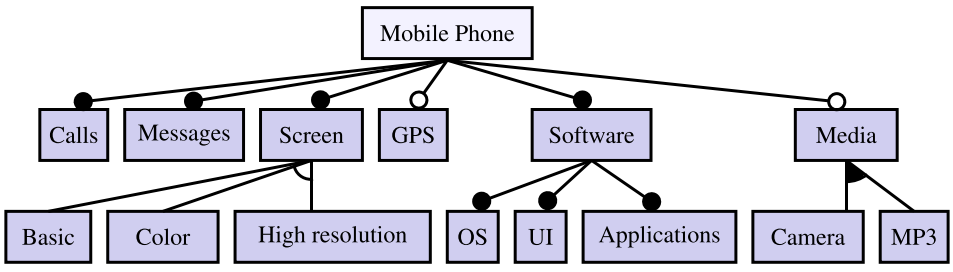

FIGURE 2 FM of the mobile phone.

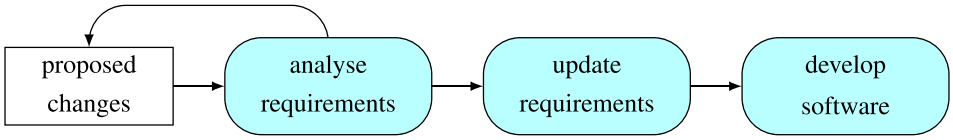

FIGURE 3 The typical repair process.[7], p. 239

When dealing with an UCR, due to the emergent/urgent nature of the situation, software engineers may not be able to follow the change analysis process that is established in their organization. Instead, they typically apply some approach similar to ERP.[7] We propose in those situations the adoption of FbERP, an extension to the traditional ERP.

The fundamental concept behind FbERP is to represent each UCR by an FM. The justification for using FMs, which are normally used for SPLs, is that they provide a clear and structured representation of the product features, making it easier for the software engineers to understand and prioritize the requested changes. FbERP also intends to facilitate feature traceability to the code base, aiming to ensure that the changes are reflected into all the development artifacts, like the documentation and the code base.

As shown in Figure 4, rather than modify the requirements and the design, the software engineers try to fix the program to solve the immediate problem.[18] As one can see in Figure 5, FbERP is an extension of ERP as it includes its three activities (marked with light blue background), but adds three new activities (marked with orange background). We next describe the six major activities that are part of FbERP:

1. **Create an UCR feature model (UCRFM)** that encapsulates the given UCR, using features and their relationships.
2. **Modify the existing FM**, when the requested change is approved and deemed necessary to reflect the change. This requires the addition or modification of features, dependencies, and relationships based on the UCRFM. The modified FM should represent the desired state of the system after implementing the requested change.
3. **Analyze the source code** by scrutinizing the existing FM of the software system and determining the impact of the requested change on its various features. Determine the dependencies, relationships, and potential conflicts that may result from the proposed change.
4. **Trace features to the code base** by distinguishing which specific code parts are impacted by the change request.[19] This requires establishing traceability links among the requested changes, underlying documentation, and code base, and documenting the traceability connections in a reference document. This eases comprehension of the rationale behind the changes and documents the decision-making procedure. This ensures that the changes are aligned with the system goals and can be readily reviewed or audited if necessary.
5. **Modify the source code base** and implement the UCR.[20] Once the change request is approved and all necessary preparations are made, proceed with implementing the change based on the UCRFM.
6. **Deliver the updated system** and deploy the software update to the production environment. Monitor the system attentively for any unanticipated problems that may arise as a result of the modification. Respond promptly to any concerns reported.

With the support of FMs, software engineers can analyze and manage essential change requests systematically while considering their impact on the overall software system. It enables them to make informed decisions, to establish effective priorities, and to maintain a clear understanding of the ongoing changes.

5 | RUNNING EXAMPLE

This section uses the project of an e-commerce platform as an example, to demonstrate the use of FMs to address UCRs. This platform was under development/maintenance by a team of software engineers. At some point in time, a critical bug was reported, and it needs to be fixed urgently. The bug affects the checkout process, where customers are unable to complete their orders. The UCR requires the software engineers to modify a specific feature temporarily until the bug is fixed.[21] In the following explanation, we present how FbERP can be used to handle this UCR.

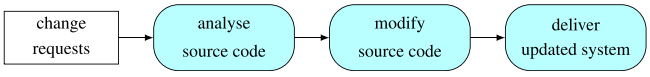

FIGURE 4 The emergency repair process.[7], p. 239

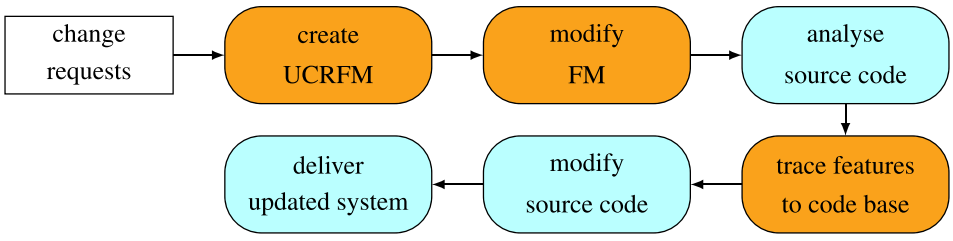

FIGURE 5 The feature-based emergency repair process (FbERP)

5.1 | Executing FbERP

The first step is to create an FM for the e-commerce platform. As presented in Figure 6, we assume that, for this example, the FM includes features like Product Search, Shopping Cart, Checkout, Payment Gateway, and Reviewers. This step also involves creating the UCRFM and determining which specific feature needs to be modified. In this case, it is the Payment Gateway feature that needs to be modified to address the bug under consideration. Secondly, the software engineers, when the requested change is approved, must modify the existing FM by refining it (Figure 6) with the UCRFM (Figure 7). The result of the composed FM is depicted in Figure 8.

At this point, the software engineers analyze the code base and identify which sections are affected by the change request. Next, the software engineers can implement the change request. In order to preserve consistency among the change requests, the documentation, and the code base, software engineers establish a traceability link from the change request to the code base. This makes it easier to apply any necessary emergency software fixes in the future. Finally, the software engineers deliver the modified software. By following the steps of FbERP, the software engineers can effectively handle UCRs using an FM.

Tracing the features of a change request, indicated in the corresponding UCRFM, to the code base involves identifying which specific parts of the code are affected by the requested changes. This process, which is described in detail in Ignaim et al,[22] is essential for understanding the scope of the change and ensuring that the modifications are made accurately and efficiently. For the e-commerce platform example, the mapping of the UCR to fix a bug in the checkout process involves the following steps. The software engineers need to (i) identify features of UCRFM. Next, they (ii) analyze the change request; for example, the change request specifies that the Payment Gateway feature needs to be modified to address the bug. Afterwards, the software engineers (iii) trace features to the code base. For that, they need to identify which parts of the code are related to the Payment Gateway feature. For the running example, we assume the following modules are associated with the Payment Gateway feature:

- The Payment Gateway module, which handles interactions with the payment gateway provider.
- The Checkout module, which manages the checkout process, including processing payments through the payment gateway.
- The Order module, which handles order creation and management, including payment status.

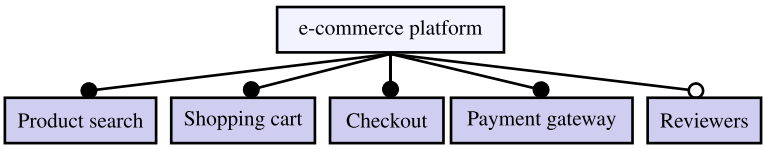

FIGURE 6 The FM for an e-commerce platform.

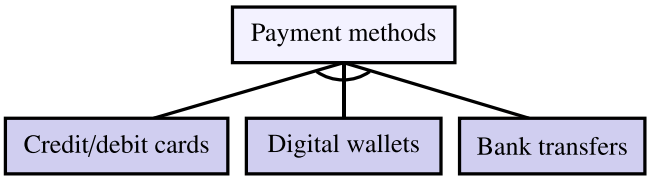

FIGURE 7 The UCR feature model.

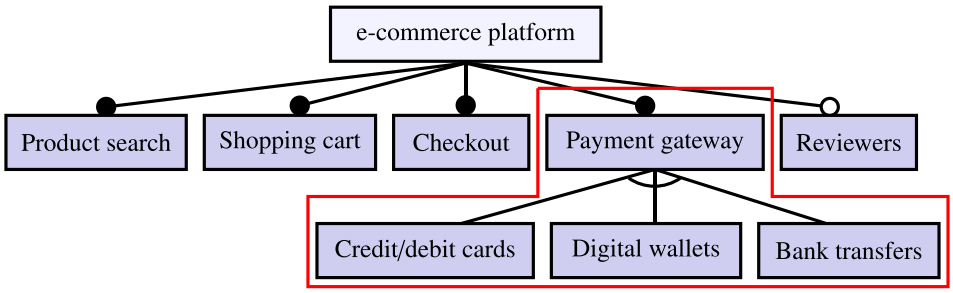

FIGURE 8 The existing FM after refining with the UCR FM.

Finally, the software engineers (iv) document the tracing links from the features to the code base in a reference document, so that they can use the document to address future UCRs. By following these steps, software engineers can effectively trace features to the code base and implement the necessary modifications. This strategy enables them to comprehend the impact of the change, isolate the affected portions of the code, and address the UCR in a focused manner.

5.2 | Inserting the UCRFM

Once an UCR is approved, the software engineer needs to insert the UCRFM that represents that specific UCR into the existing FM. The software engineers can insert the UCRFM into a specific location (i.e., a target feature in the existing FM) by using the insertion operator suggested by Acher et al@˙.[23] For the UCR of the e-commerce platform, the software engineer can use the insertion operator to insert the respective UCRFM (Figure 7) into the existing FM (Figure 6) at the place where is the PAYMENT GATEWAY feature. This ensures that the UCR is properly addressed, without affecting other features of the e-commerce system (Figure 8).

6 | EVALUATION

To evaluate FbERP, we have conducted a case study in a real environment and have asked software engineers to implement five UCRs using both ERP and FbERP. We then compared the results and performance of software engineers in both cases. We used the following two evaluation techniques: an empirical study and a small survey.

6.1 | Empirical study

The study has one treatment, which is FbERP. The subjects are the software engineers who are working on repairing the objects. The authors found that the majority of the software engineers had a good comprehension of the basic concepts of FMs. The objects of this study are the UCRs for the e-learning platform. The software engineers were asked to address the five UCRs indicated in Table 1, using firstly ERP and then FbERP. In order to test the performance of both processes, we formulated the following hypotheses:

- H_0: FbERP **does not outperform** ERP.
- H_1: FbERP **outperforms** ERP.

Table 2 shows the effectiveness and efficiency of the software engineers while addressing UCRs with both processes. The last row of Table 2 also presents the performance improvement of software engineers while they were using FbERP. Effectiveness is calculated as the ratio between the number of right change requests that the software engineers performed and the total number of right change requests. Efficiency is calculated as the ratio between the number of right change requests that the software engineers performed and the total time spent on performing the change request. The results depicted in Table 2 reveal that FbERP can be successfully used by software engineers to address UCRs

TABLE 1 UCRs for the e-learning platform.

id	UCR
A	A request for an immediate bug fix to allow students to submit assignments seamlessly
B	A request for an immediate server upgrade to handle the increased traffic and ensure uninterrupted platform access
C	A request for an urgent correction and update to the video lecture to provide accurate information to the students
D	A request for an immediate fix to the user authentication system to ensure that students can access their accounts without any problems
E	A request for an immediate update to improve platform accessibility and ensure it complies with accessibility standards

TABLE 2 Results of the empirical study that includes five change requests.

	Number of right Change requests		Effectiveness (%)		Number of right Change requests		Time (Hours)		Efficiency (Change requests/hour)	
Software engineer	FbERP	ERP	FbERP	ERP	FbERP	ERP	FbERP	ERP	FbERP	ERP
#1	**4**	2	**80**	40	**4**	3	5.5	6.0	**0.73**	0.50
#2	3	3	60	60	**5**	3	5.5	5.0	**0.91**	0.60
#3	**4**	2	**80**	40	5	5	6.0	7.0	**0.83**	0.71
#4	**4**	3	**80**	60	**5**	3	5.5	5.0	**0.91**	0.60
#5	**4**	3	**80**	60	**4**	3	7.0	6.0	**0.57**	0.50
#6	**3**	2	**60**	40	4	4	5.0	6.0	**0.80**	0.67
#7	**4**	3	**80**	60	3	3	4.5	5.0	**0.67**	0.60
#8	**4**	3	**80**	60	**4**	3	4.5	5.0	**0.89**	0.60
#9	**4**	1	**80**	20	4	4	5.0	6.5	**0.80**	0.62
#10	3	**4**	60	**80**	3	**4**	5.5	5.5	0.55	**0.73**
Average	**3.7**	2.6	**74**	52	**4.1**	3.5	–	–	**0.77**	0.61

Note: Bold values indicate the best values when comparing FbERP and ERP.

TABLE 3 The questionnaire for evaluating FbERP.

id	Question
Q1	Does FbERP achieve its intended outcomes and goals?
Q2	Does FbERP consistently produce accurate and high-quality results?
Q3	Does FbERP support employee productivity and job satisfaction?
Q4	Is FbERP easy to use?
Q5	In FbERP, does the use of feature models is a source of difficulty for software engineers?

TABLE 4 Four-point Likert scale weights and abbreviations.

Scale point	Weight	Abbreviation
Strongly agree	+2	SA
Agree	+1	A
Disagree	−1	D
Strongly disagree	−2	SD

with an improvement in effectiveness and efficiency of 22 percentage points and 26%, respectively. This proves that, when dealing with UCRs, software engineers may have higher performance with FbERP.

6.2 | Small survey

We prepared a small survey with five questions (see Table 3), related to the characteristics of FbERP. Questions Q1–Q4 are formulated in a positive sense, since, when the response is an agreement, that implies a positive perception in relation to FbERP. Contrarily, Q5 is formulated negatively, since, when the response is an agreement, that implies a negative perception with respect to FbERP. Because we wanted the user to express an opinion, we collected responses using a forced-choice four-point Likert scale, in which there is no secure "neutral" alternative. Table 4 presents the response options, which range from strongly agree to strongly disagree, and also their respective numerical values.

The questionnaire was distributed to five companies. It was answered by ten of their software engineers, who read the manual that explains the proposed process. We analyzed the answers of software engineers (see Tables 4 and 5). From one side, the analysis shows that the software engineers agree on adopting FbERP in addressing UCRs (i.e., user friendliness and applicability). On the other side, they complained about the difficulty of the FM, namely the software engineers who were unfamiliar with this technique.

TABLE 5 Answers provided by 10 software engineers to the five questions and their analysis.

	Software engineers										Mean of responses	Point scale
Question	1	2	3	4	5	6	7	8	9	10		
Q1	+2	+1	+1	+2	+2	+1	+1	+2	+2	+2	+1.6	SA
Q2	+2	+2	+2	+2	+1	+1	+2	+2	+1	+2	+1.7	SA
Q3	+1	+1	+2	+2	+1	+2	+1	+2	+1	+1	+1.4	A
Q4	+1	+1	+1	+2	+2	+2	+2	+2	+2	+2	+1.3	A
Q5	+1	+1	+1	+1	+2	+1	+1	+1	+1	+2	+1.2	A

Regarding Q1, the software engineers strongly agree on the ability of FbERP to achieve the intended outcomes and goals of responding to UCRs in an effective and efficient manner. For Q2, the software engineers believe that FbERP consistently produces accurate and high-quality results that enable them to address UCRs while keeping the documentation and the code base aligned. The software engineers feel, with respect to Q3, that FbERP supports their productivity and job satisfaction. The software engineers affirmed, with respect to Q4, that FbERP is easy to apply and use. Finally, for Q5, the software engineers agree that the use of FMs is a source of difficulty in FbERP. FMs can be a powerful instrument for managing the features of a set of software products. However, they also present challenges that make them difficult to use, particularly for large systems with numerous features and relationships, in which it is normally hard to comprehend the relationships among the features.

6.3 | Results

The results in Tables 2 and 5 provide evidence that enables us to accept the alternative hypothesis (H_1) and to reject the null one (H_0). This implies that FbERP can be used effectively and efficiently to address UCRs and to improve the performance of ERP. The improvement in performance of software engineers in performing UCRs using FbERP (last row in Table 2) and positive feedback from the software engineers regarding the ease of use (rows 5 in Table 4) and applicability (rows 2–4 in Table 4) of FbERP further strengthen the case for its applicability in addressing UCRs. The complaints about the difficulty in using FMs (row 6 in Table 4) can be tackled through training and providing support to software engineers to familiarize them with the technique. Overall, the study suggests that implementing FbERP can lead to significant improvements in addressing UCRs and enhancing the performance of existing processes.

7 | VALIDITY THREATS

This section discusses potential threats to the validity of our study and steps we have taken to minimize or even to mitigate them. The threats are discussed in the context of the four types of threats to validity based on a standard checklist: internal, construct, conclusion and external validity[24].

Internal validity refers to the extent to which a cause-and-effect relationship established in a study cannot be explained by other factors. In this study, we controlled for potential confounding variables by randomly addressing change requests with both ERP and FbERP. This increases the internal validity of the findings and suggests that any observed effects could be attributed to FbERP. However, there may still be other uncontrolled variables that could influence the results, such as individual differences in the skills of software engineers or the specific context in which the evaluation took place. An important limitation of this study is that the evidences are taken from the maintenance of a specific e-learning platform, in which software engineers addressed UCRs. This situation is not representative of all possible types of software maintenance tasks. Thus, while the study provides evidence of the effectiveness of FbERP within a specific context, we recommend caution when interpreting the results in other contexts.

Regarding the estimation of the effectiveness/efficiency of FbERP, and the comparison with ERP, the software engineers first completed the tasks with ERP and then with FbERP. Repetition of a task may affect the time necessary to complete it. Additionally, the software engineers may embrace the same perspectives regarding FbERP, and we may lose the ability to get correct feedback if this occurs. This risk was addressed by dividing the empirical study into two sessions and doing each individually. We performed the first session and waited a period of five weeks to perform the second session. This may still be an issue in terms of familiarization, since one may consider this period not sufficiently long.

Construct validity is related to the accuracy and correctness of the implementation of FbERP. In this case, it is important to ensure that the each part of the system affected by a change request is accurately represented in the FM and that the tracing mechanism correctly identifies the

corresponding code locations. To address this threat, one should provide details about how the FM is constructed and how the accuracy of the tracing mechanism is verified. Additionally, one should consider conducting further experiments or using different datasets to validate the obtained results.

Conclusion validity is about the ability to draw significant correct conclusions. which are reached through rigorous and repeatable treatment. We analyzed qualitatively whether the adoption of FbERP improved. We attempted to reduce the bias by seeking support from the data gathered from the empirical study and the survey. Therefore, the conclusions presented in this article are traceable to data.

External validity refers to the generalizability of the results beyond the specific context in which the evaluation was conducted. In this case, we used real-world UCRs to evaluate FbERP, but it is unclear whether the same results will be observed in different software development environments or with different types of change requests. Additionally, the sample size of the change requests used in the evaluation may not be representative of all possible change scenarios. Therefore, we recommend being careful when applying FbERP in specific settings.

8 | CONCLUSIONS

The occurrence of UCRs is inevitable in various contexts. This work proposes FbERP, an improvement to ERP that addresses the occurrence of UCRs by introducing feature modeling techniques. The basic idea of ERP is to model change requests using FMs and trace changes to their locations in the code base. We evaluated FbERP using real-world change requests and compared it against ERP. For the experiments described in this paper, the results show that FbERP, when compared to ERP, achieves an improvement of 22 percentage points in effectiveness and 26% in efficiency and supports employee productivity and job satisfaction. The results also show that FbERP can be used effectively and efficiently to address UCRs with high user satisfaction. The evaluation process conducted with software engineers confirms that the use of FMs helps them to manage and prioritize the UCRs, because they are able to visualize and understand the dependencies among different features. By systematically analyzing the impact of each UCR, software engineers can ensure that modifications do not inadvertently introduce new bugs or conflicts.

In general, nothing impedes FbERP to be applicable to all types UCRs, but it seems more appropriate to some UCR types than others. For instance, FbERP is certainly well adapted to UCRs that imply improving a given software product that involves features that are already represented in the FM of that product, but not included in the software product itself (e.g., the features in an alternative or OR group). In any case, it is up to the software engineers to identify the context of each specific software product and evaluate whether FbERP may bring benefits.

With this study, we have opened up new opportunities for further exploration on the use of feature modeling in addressing UCRs. We have provided observations that can be beneficial for practitioners and researchers in the field of software engineering. We believe that there are many ways to continue this work, like applying FbERP to more projects in order to gain more insights on its advantages and disadvantages and consequently improve its ingredients.

ACKNOWLEDGMENTS

We would like to thank the companies for taking part in the evaluation process. The case study would not be complete without their continuous support and cooperation. This work was supported by FCT–Fundação para a Ciência e Tecnologia within the R&D Units Project Scope UIDB/00319/2020.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available in the supplementary material of this article

ORCID

Karam Ignaim https://orcid.org/0000-0001-8650-8021
João M. Fernandes https://orcid.org/0000-0003-1174-1966

REFERENCES

1. Lacroix M, Lavency P. The change request process. In: 2nd International Workshop on Software Configuration Management (SCM 1989); 1989:122-125. https://doi.org/10.1145/72910.73357
2. Stojanov Z, Dobrilovi D, Perii B. An approach in modifying submission phase of change request process. In: 6th International Symposium on Intelligent Systems and Informatics (SISY 2008); 2008. https://doi.org/10.1109/SISY.2008.4664923
3. Efe P, Demirors O. A change management model and its application in software development projects. *Comput Stand Interf*. 2019;66:103353. https://doi.org/10.1016/j.csi.2019.04.012
4. Siebert S, Paton RA, McCalman J. *Change Management: A Guide to Effective Implementation*: SAGE; 2015.
5. Heydari M, Lai KK, Fan Y, Li X. A review of emergency and disaster management in the process of healthcare operation management for improving hospital surgical intake capacity. *Mathematics*. 2022;10(15):2784. https://doi.org/10.3390/math10152784

6. Sancak IE. Change management in sustainability transformation: a model for business organizations. *J Environ Manag*. 2023;330:117165. https://doi.org/10.1016/j.jenvman.2022.117165
7. Sommerville I. *Software Engineering*. 9th ed.: Pearson Education; 2011.
8. Motwani M, Sankaranarayanan S, Just R, Brun Y. Do automated program repair techniques repair hard and important bugs?. *Empir Softw Eng*. 2018; 23(5):2901-2947. https://doi.org/10.1007/s10664-017-9550-0
9. Silva RK, Farias K, Kunst R, Dalzochio J. An approach based on machine learning for predicting software design problems. In: XIX Brazilian Symposium on Information Systems (SBSI 2023); 2023:53-60. https://doi.org/10.1145/3592813.3592888
10. Botterweck G, Pleuss A, Dhungana D, Polzer A, Kowalewski S. EvoFM: Feature-driven planning of product-line evolution. In: Workshop on Product Line Approaches in Software Engineering (PLEASE 2010). ACM; 2010:24-31. https://doi.org/10.1145/1808937.1808941
11. Guo J, Wang Y. Towards consistent evolution of feature models. In: 14th International Conference on Software Product Lines (SPLC 2010), Lecture Notes in Computer Science, vol. 6287. Springer; 2010:451-455. https://doi.org/10.1007/978-3-642-15579-6_35
12. Maóoun J, Bouassida N, Ben-Abdallah H. Change impact analysis for software product lines. *J King Saud Univers - Comput Inform Sci*. 2016;28(4):364-380. https://doi.org/10.1016/j.jksuci.2016.01.005
13. Gomaa H, Hussein M. Dynamic software reconfiguration in software product families. In: 5th International Workshop on Software Product-Family Engineering (PFE 2004), Lecture Notes in Computer Science, vol. 3014. Springer; 2004:435-444. https://doi.org/10.1007/978-3-540-24667-1_33
14. Hallsteinsen S, Stav E, Solberg A, Floch J. Using product line techniques to build adaptive systems. In: 10th International Software Product Line Conference (SPLC 2006); 2006:141-150. https://doi.org/10.1109/SPLINE.2006.1691586
15. Clotet R, Dhungana D, Franch X, Grünbacher P, López L, Marco J, Seyff N. Dealing with changes in service-oriented computing through integrated goal and variability modelling. In: 2nd International Workshop on Variability Modelling of Software-Intensive (VAMOS 2008); 2008:43-52.
16. Ignaim KM, Alkharabsheh K, Ferreira AL, Fernandes JM. A concrete product derivation in software product line engineering: a practical approach. *Int J Comput Appl Technol*. 2022;70(3-4):225-232. https://doi.org/10.1504/IJCAT.2022.10054830
17. She S, Ryssel U, Andersen N, Wąsowski A, Czarnecki K. Efficient synthesis of feature models. *Inform Softw Technol*. 2014;56(9):1122-1143. https://doi.org/10.1016/j.infsof.2014.01.012
18. Arcuri A, Yao X. A novel co-evolutionary approach to automatic software bug fixing. In: IEEE Congress on Evolutionary Computation (CEC 2008); 2008:162-168. https://doi.org/10.1109/CEC.2008.4630793
19. Eisenbarth T, Koschke R, Simon D. Locating features in source code. *IEEE Trans Softw Eng*. 2003;29(3):210-224. https://doi.org/10.1109/TSE.2003.1183929
20. Rajlich V, Wilde N. The role of concepts in program comprehension. In: 10th International Workshop on Program Comprehension (WPC 2002); 2002: 271-278. https://doi.org/10.1109/WPC.2002.1021348
21. Jin M, Shahriar S, Tufano M, Shi X, Lu S, Sundaresan N, Svyatkovskiy A. Inferfix: End-to-end program repair with LLMS. In: 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023); 2023:1646-1656. https://doi.org/10.1145/3611643.3613892
22. Ignaim K, Fernandes JM, Ferreira AL. An industrial experience of using reference architectures for mapping features to code. *Sci Comput Program*. 2024;234:103087. https://doi.org/10.1016/j.scico.2024.103087
23. Acher M, Collet P, Lahire P, France R. Composing feature models. In: 2nd International Conference on Software Language Engineering (SLE 2009), Lecture Notes in Computer Science, vol. 5969. Springer; 2009:62-81. https://doi.org/10.1007/978-3-642-12107-4_6
24. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. *Experimentation in Software Engineering: An Introduction*: Springer; 2000.

How to cite this article: Ignaim K, Fernandes JM. Improving an emergency repair process with feature models. *J Softw Evol Proc*. 2024; e2701. https://doi.org/10.1002/smr.2701