IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 2, 2019, accepted July 6, 2019, date of publication July 16, 2019, date of current version August 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929214

Cognitive and Hierarchical Fuzzy Inference System for Generating Next Release Planning in SaaS Applications

MUBARAK ALRASHOUD[1], ETESSAM HAZZA[1], FAYEZ ALQAHTANI[2], MUNEER AL-HAMMADI[3], ABDOLREZA ABHARI[4], AND AHMED GHONEIM[1]

[1]Department of Software Engineering, College of Computer and Information Science, King Saud University, Riyadh 11451, Saudi Arabia
[2]Department of Computer Science, College of Computer and Information Science, King Saud University, Riyadh 11451, Saudi Arabia
[3]Department of Computer Engineering, College of Computer and Information Science, King Saud University, Riyadh 11451, Saudi Arabia
[4]Department of Computer Science, Ryerson University, Toronto, ON M5B 2K3, Canada

Corresponding author: Mubarak Alrashoud (malrashoud@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University, Riyadh, Saudi Arabia, through the research group project under Grant RG-1440-135.

ABSTRACT The next release planning is considered as a cognitive decision-making problem where many stakeholders provide their judgments and opinions about the set of features that shall be included in the next release of the software. In multi-tenant Software as a Service (SaaS) applications, planning for the next release is a significant process that plays important roles in the success of SaaS applications. SaaS providers shall fulfill the evolving needs and requirements of their tenants by continuously delivering new releases. The first step in a release development lifecycle is the release planning process. This paper proposes a novel approach for the next release planning for multi-tenant SaaS applications. This approach is a prioritization approach that employs a hierarchical fuzzy inference system (HFIS) module to deal with the uncertainty associated with human judgments. The main objectives of the proposed approach are maximizing the degree of overall tenants' satisfaction, maximizing the degree of commonality, and minimizing the potential risk, while considering contractual, effort, and dependencies constraints. The performance of the proposed approach is validated against a one from the literature and shows better results from the perspective of overall tenants' satisfaction and adherence to the risk.

INDEX TERMS Fuzzy decision making, cognitive fuzzy inference systems, fuzzy inference system applications, software release planning, software as a service engineering.

I. INTRODUCTION

Utility (pay-per-use) computing has been a trend in recent years due to its ability to eliminate the overhead expenses of establishing IT infrastructure and management and maintenance efforts [1]. Multi-tenants Software as a Service (SaaS) is a utility model that provides a software as a shared cloud services that can be used via a thin client web-based application by a wide range of tenants [2]. The benefits of SaaS for tenants can be seen in cost reduction in terms of management, maintenance, and annual licensing [2]. From the provider's perspective, the benefits arise from serving a large number of tenants through a shared, centrally hosted service [1]. The success of SaaS applications is highly dependent on its tenants' satisfaction, which can be achieved via the fulfillment of their constant, continues, and evolving needs [2]. One of the significant difficulties that face SaaS providers is that they need to compromise some of the requested new features to balance the available resources, and at the same time, maximize the satisfaction of their tenants while fulfilling technical and contractual constraints. Consequently, assigning features to new releases shall be considered as a planning process that involves several factors. The process of Release Planning (RP) addresses decisions related to the selection and assignment of features to a set of software releases while considering the technical, resource, and risk constraints [3]. According to [4], [5], release planning is cognitively and computationally difficult. Therefore, human

The associate editor coordinating the review of this manuscript and approving it for publication was Yin Zhang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

intelligence shall be supported by computational intelligence to deal with the range of interrelated and complex decision factors. This paper proposes an approach of release planning for SaaS application. The proposed approach is a prioritization and cognitive that employs Hierarchal Fuzzy Inference Systems (HFIS).The main contribution of this paper is that it deals with release planning for SaaS applications as uncertain decision-making problem where human judgments and perspectives play the main role. Fuzzy inference system (FIS) depends on fuzzy reasoning to deal with uncertainty associated with human judgments. In FIS, the knowledge of the experts is converted to fuzzy rules. Then, the estimates that are provided by stakeholders about the uncertain attributes of the features are manipulated using these rules, which allow considering human expertise implicitly and automatically. We consider planning for only the next release of SaaS applications due to the extreme dynamics involved in SaaS applications, which significantly increases the likelihood of a wide range of changes being required within a short time period.

The rest of this paper is organized as follows: Related work is presented in Section 2. Theoretical preliminaries of this study are presented in Section 3. Section 4 presents the proposed approach for solving release-planning problem in SaaS application. Section 5 validate the propose approach by comparing its performance with an approach from the literature. Conclusions are presented in Section 6.

II. RELATED WORK

This paper utilizes fuzzy inference systems. These systems mimic the cognitive activity of the brain, which is based upon the relative grades of the information acquired by the natural sensory system [6]. We can say that using fuzzy inference system in solving release-planning problem mimics human cognitive in solving similar problems. It is useful to state that cognitive computing opened the doors for improving a wide range of smart services. Furthermore, the cognitive computing will make it possible to understand what happening in the world more deeply [7]. For examples, Hossain and Muhammad invested the development in big data oriented wireless, 5G, and Internet of Things technologies to enhance the healthcare services in [8]. They presented an emotion aware connected healthcare system. For emotion recognition, they fused the data captured by multimodality sensors in the IoT. The data processing is distributed in the cloudlet, the edge devices, and the cloud. A cognitive Internet of Things (CIoT)-based framework was proposed for image authenticity verification [9]. The proposed framework used opposite color local binary pattern (OC-LBP) for feature extraction and SVM for classification. An outperforming accuracy was reported for the proposed framework on two datasets. Zhang *et al.* presented a collaborative filtering (CF) based cross-domain recommender system to improve the performance of conventional CF-based recommender systems [10]. The user's similarity matrix for rating predictions in the target domain can be calculated based on the historical behaviors of the user in the auxiliary domain. The authors presented different ways for two domains fusion. In their proposed system, they addressed the challenge of cold-start users through association rules with social big data. The proposed method obtained superior performance over the conventional CF-based methods. A deep learning based multimodality system for emotion recognition was presented in [11]. To extract the features from the input signals, 2D CNN model is used for the audio spectrogram while 3D CNN model is used for the video signal. Extreme learning machine and support victor machine are used for feature fusion and classification respectively. The presented system obtained comparable performance. Hao *et al.* proposed emotion-aware video quality of experience (QoE) based on transfer learning [12]. They achieved better performance in video QoE by considering user's emotional reaction as an important factor in the watching experience. The users' emotion data are collected while watching the video. Their emotion obtained via transfer learning and HMMs. The video content emotion on the other hand are obtained through statistical methods. The video QoE is high when the similarity between the emotion of the users and the emotion of the content is high and vice versa. The authors found that individual users have different sensibilities to different influential factors. Based on that they presented emotion-aware personalized video QoE model with more enhanced prediction. In [13], cognitive computing technologies is applied to propose a new paradigm, CIoT, that absorb new capabilities in AI such as deep learning, the CIoT sensing system, data analytics, and cognition in providing human-like intelligence.

Many models and approaches in the literature aim to solve release-planning problems. Genetic Algorithm (GA) is used in many works in the literature to solve release-planning problem. In [4], Greer and Ruh proposed the EVOLVE method. It aimed at the iterative planning of incremental software development, producing a RP at each increment. The novelty of the approach is the employment of GA as an optimization to find an optimal or near-optimal solution. The factors of requirement's priority, effort constraints, requirement dependency, and stakeholders' weights are considered when planning for the next release. Ruhe and The [5] aimed at achieving maximum stakeholders' satisfaction through the gradual reduction of the problem's size and complexity. They believe that neither human nor computational intelligence alone be able to provide adequate decision support. Hence, they proposed a new hybrid approach (which they call it EVOLVE*) combining the strength of mathematical models with the subtleness of experts' knowledge and judgment. The advantage of the human intelligence is to handle soft, fuzzy, and implicit constraints, while the advantage of computer-based approach is to cover the size and complexity of the solutions. In [14], an extension of EVOLVE* is proposed. Q-EVOLVE II aims at balancing between development and test resources based on the list of candidate requirements, effort, and defects. It employs the tool ReleasePlanner with some quality aspects added to the model as a

plugging called W2RP. In [15], EVOLVE+ is introduced. It is an approach considers hard (such as budget) and soft (such as risk) constraints and objectives in the planning process. ELECTRE is utilized by EVOLVE+. Authors in [16] consider the dependencies between candidate requirements. In that proposed approach, theme-based RP is formulated as a bi-objective optimization problem that is tackled with Non-Dominated Sorting Genetic

Algorithm-II (NSGA-II). The aim is to deliver a group of features in a release that are inter-related, leading to better release value and customer satisfaction. Each solution of this optimization problem balances the preference between individual (stakeholder's prioritization) and theme-based planning objectives. Authors in [17] extend EVOLVE* in the perspective of clustering themes. The idea aims at offering features in a release in consideration to their semantic cohesiveness. The authors identified two categories of dependencies: direct (reflect a degree of similarity between features) and indirect (reflect a degree of dissimilarity between features). Directly dependent features have a degree of commonality amongst them, while indirect features occurred in a release is unfavorable. The approach provides three additional steps on top of the established planning process of EVOLVE*, due to its indirect management of themes while making release decisions. In [18], an extension of EVOLVE* presented to address the problem of feature generation and selection in industrial settings. The major idea of the solution proposal was to generate a system release that fulfills implementable business features; by implementing new/change software features. The model explicitly links business strategies with solution planning and solution development. The new features will be planned into optional and mandatory solutions. Fuzzy systems models are used for solving release-planning problem. In such models, release planning is considered as a fuzzy decision making problem, where human judgment relies on individual's understanding and perception of the problem plays roles in deciding the importance of features. Reference [19] considers release planning factors as fuzzy factors and extends EVOLVE* to FUZZY-EVOLVE*. The available and the required effort are denoted as fuzzy numbers. The objective function is formed as a fuzzy membership function. Fuzzy numbers operations are applied to the fuzzy objective function and fuzzy effort constraints in order to find the release plan that finds the optimal degree of satisfaction. In [20], the authors provide a solution to general release planning problem by prioritizing the features depending on their ranks generated by a FIS engine. In [21], Adaptive Network FIS (ANFIS) is used for generating release plans. In that model, the fuzzy rules obtained from the experts are adjusted using historical data about the previous release(s), which provides higher reliability than traditional FIS presented in [20].

Integer linear programming (ILP) is used to formulate release-planning problem. In [3], ILP is used to create a synergy between the art and science by integrating the generation of best solutions through the application of computational algorithms with the reliance on human capabilities to optimize among multiple alternatives. The latter is used to solve the illness of problem definition. In [2], Binary linear Programming (BLP), which is a special case of ILP, is used to propose a model for release planning in SaaS business applications. That model aims to identify the requirements that shall be included in the next release. The model is designed with the objectives of maximizing both the overall tenants' satisfaction, and the degree of commonality of the requested requirements, while minimizing its' implementation risks. In [22], ILP is used to formulate release planning in software product line (SPL). Because of the nature of SPL projects, new factors are involved (for example, resolving the conflicts between the requirements of core assets and the requirements of various products) [23] proposes a reduction approach for solving the Next Release Problem (NPR) using the Backbone-based Multilevel Algorithm (BMA). The algorithm employs multilevel reductions to iteratively downgrade the problem size to construct the final optimal set of requirements. The authors considered the factors of customer profits and requirements costs when constructing the solution. The goal is to maximize profits from a set of candidate requirements under budget constraint. The BMA framework is an iterative strategy, that converts the original problem into multiple levels of independent subproblems. It contains three phases: reduction, solving, and refinement.

This paper deals with release planning for SaaS applications. It considers some factors that are related to the nature of SaaS application. For example, the type (or level) of service to which tenants subscribe is considered. Furthermore, In order to satisfy the most of tenants, the commonalities of features are taken into account in the planning process. We formulate release-planning problem for SaaS in as a prioritization problem where the features are assigned to the next release according to their ranks (that ranks are calculated by considering a set of factors that will be defined later in this paper). From the literature, we can see that the existing models to release planning employs human expertise to adjust the solutions that have been produced by mathematical approaches. So, these approaches can be used when there are limited numbers of features and stakeholders. However, this is not the case in SaaS applications. In SaaS applications, huge numbers of stakeholders located in different locations in the world participate in the planning effort, which makes difficult to adjust solutions manually. It is crucial to find an approach that allows release management to automatically and implicitly incorporate human expertise into the formulized solution of release-planning problems. Table 1 summarizes the reviewed models and with a comparison against our proposed approach.

III. PRELIMINARIES ABOUT FUZZY INFERENCE SYSTEMS

A fuzzy inference system (FIS) [24] is a function that utilizes linguistic rules to produce output. FIS module is composed of the following components:

TABLE 1. Summary of release planning solutions and formulation techniques.

Reference	Model	Formulation and Solutions Technique				Technical Constraints	Resource Constraints	Contractual Constraints	Problem Variables	No. of Releases
		ILP	AHP	GA	Other					
[4]	EVOLVE	✓		✓		Coupling and precedence	Effort of implementation		Requirement's priority, Effort constraints, Requirements dependency, Stakeholders' weights	Multi-Release
[5]	EVOLVE*			✓	Multi-criteria decision aid and heuristics	Coupling and precedence	Effort of implementation		-Requirements dependency, Effort constraints, Structure constraints, Stakeholder prioritization	Next two releases
[3]	The Art and Science of RP	✓				Coupling and precedence	Effort in Person-Hour, Cost		-Feature dependency, Stakeholders' interests, Available resources, Requirement's priority	Multi-Release
[15]	Backbone-based				BMA	Coupling and precedence	Cost of implementation		-Requirement costs, Customer priorities	Next Release
[6]	Q-EVOLVE II	✓				Coupling and precedence	Effort of implementation		-Requirement dependency, Quality constraints, Resource constraints, Stakeholders scores	Multi-Release
[9]	Theme-based				Multi-criteria decision aid and heuristics	Direct & Indirect Requirement dependency	Effort of implementation, Cost of implementation		-Advanced Requirement dependency, Resource constraints, Stakeholders scores	Multi-Release
[8]	Bi-objective			✓	NSGA-II	Coupling precedence	-Effort of implementation -Cost of implementation		-Requirement dependencies, Resource constraints, Effort constraints, Stakeholders priorities, Theme & feature values	Next Release
[2]	BLP	✓			BLP	Coupling precedence	-Effort of implementation	✓	-Tenants' Importance, Required effort, Available resources, Risk, Requirements dependency, Commonality of requirements, Tenant's weights	Future releases
[12]	Perception-Based				FIS	Coupling precedence	-Effort (person-days)		-Managerial constraints, Requirement dependencies, Effort constraints, Stakeholders' preferences, Stakeholders weights, implementation risk	Next Release
[13]	Adaptive network-based				FIS		-Available effort (person-days) -Required effort (person-days)		-Stakeholders' satisfaction -Risk -Resources Availability	Next Release
	The proposed prioritization approach for SaaS				✔	**Coupling and precedence**	**Effort of implementation**	✔	**-Tenants' Importance, Required effort, Available resources, Risk, Requirements dependency, Commonality of requirements, Tenant's weights**	**Next-release**

1)The knowledge base which includes the attributes of the input and output variables (the linguistic values and the meaning (parameters) of these linguistic values).

Example: risk is a linguistic variable, and we can define two linguistic values for risk variable: high-risk and low-risk. The meaning of low-risk may be defined as triangular fuzzy number with the following parameters [1 4 7], and the meaning of high-risk may be defined triangular fuzzy number with the following parameter [4 7 10]

2)The fuzzy rules which takes the following form: *IFX is ATHEN Y is B* where X is a linguistic input variable, A is a linguistic value of X. Y is a linguistic output variable and B is a linguistic value of Y.

Example: IF risk *is high THEN featurerank is low*

In a FIS module, there are usually many rules, and the maximum number of rule is the result of multiplying the cardinality of the sets representing the linguistic values.

Example: if X and Y are two linguistic variables and the number of values that X can take is 3, and the number of values that Y can take is 4, then the maximum number of rules is 12

(3)The inference process: this process is responsible for mapping the crisp input to crisp output. The following steps are applied:

- Evaluation of conclusion of each rule
- Aggregation of all rules
- Defuzzification which is the process that finds result of the inference process by calculating the center of gravity.

More information about FIS can be found in [24]

IV. PROBLEM DEFINITION

Let R^k be the current release of a SaaS application, T is a set of tenants who are subscribed to a SaaS application such that $T = \{t_1, t_2, \ldots \ldots, t_m$, $F = \{F_{t1}, F_{t2}, \ldots \ldots, F_{tm}$ be a set of sets features that are requested by the tenants, such that F_{ti} represents the features requested by a tenant t i. We define $F^* = \bigcup_{i=1}^{m} F_{ti} = f1, f2 \ldots f$n as the correlated set that includes all features requested by all tenants, where $n \leq \sum_{i=1}^{m} |F_{ti}|$.

A. PROBLEM VARIABLES

It is required to find F^{k+1} which is a set of features that will be added (or modified) in the release R^{k+1} (the next release),

such that $F^{k+1} \subseteq F^*$. F^{k+1} shall be selected from F^* with taking into account the following factors.

1) The important of each feature in F^* to each tenant in T: we define the matrix IMP with the dimension $n \times m$ such that $IMP(i, j)$ captures the importance of feature i to the tenant j.

2) The Tenants weight (TW): where the opinions of some tenants are considered more significantly than other tenants because, for example, of their business volume or loyalty. In order to measure the actual importance of a feature, the provided importance of features shall be linked with tenants' weights [25]. Let TW be an m elements vector where the element $TW(i) \in [0, 1]$ is the weight of tenant i, and $\sum_{i=1}^{m} W(i) = 1$.

3) The commonality of features: in order to satisfy maximum number of tenants with less possible effort, it is more efficient to include the features that are requested by most of tenants [26]. Let $COMMONALITY$ be an n elements vector such that $COMMONALITY(i)$ captures the commonality of features i, and $COMMONALITY(i) = \sum_{j=1}^{m} \left(i \in F_{tj}\right)$. For example, if a SaaS application is used by 100 tenants, and a new feature i is requested by 80 tenants then we say that the commonality of the feature i is 80

4) Risk of features implementation: risk of features shall be counted when planning for the next release [27]. The risk associated with a feature is the negative impact of implementing that feature on the quality, delivery time, or cost. Let $RISK$ be an n elements vector such that $RISK(i)$ captures the risk of features i, and the risk of features are provided by the development team.

5) Type of service: many SaaS applications are offered to the tenants in different types of service (for example, standard, professional, business). The tenants shall participate only on estimating the features that he can use. In other words, a tenant is eligible to provide his judgment only about the features that will be included in his level of service. We define S as $n \times m$ matrix, such that $S(i, j) = 1$ if tenant j is eligible to estimate feature i *else* $S(\mathrm{i}, \mathrm{j}) = 0$.

6) Effort: estimating available and required effort is an essential task in release planning [3]–[5]. Required effort of a feature is the effort needed to analyze, design, implement, and test that feature. We define $EFFORT$ as an n elements vector such that $EFFORT(i)$ captures the effort required to implement features i, and the efforts of features are provided by the development team. In addition, we define $AvailableEffort$ which is a numeric that captures the effort available to implement the release under consideration.

7) Technical dependencies: as in [3], [4], we consider two types of dependences: 1) coupling where if two features are technically coupled then they should be included at the same release. 2) Precedence where if a feature fi precedes a feature fj then fi should be delivered (or at least implemented and tested) prior to feature fj. We define $DEPENDENCIES$ as $n \times n$ matrix such that $DEPENDENCIES(i, j) = 1$ if features i, j are coupled, and $DEPENDENCIES(i, j) = 2$ if feature i precedes feature j and $DEPENDENCIES(i, j) = 0$ if there is are dependencies between i and j.

B. PROBLEM STATEMENT

In this paper, we consider the next release planning for SaaS applications as a prioritization process. The features are given ranks, and the features with the highest ranks are assigned to the next release while considering effort, type of service, and technical constraints. Let $RANK(i)$ be the calculated rank of the feature i. We want to find $F^{k1} \subseteq F^*$, which is a set of features that shall be included in the next release such that for features $i, j, (i \in F^{k1}) \wedge (j \in (F^{k+1})^c) \rightarrow RANK(i) \geq RANK(j)$ and $(F^{k+1})^c$ is the complement of F^{k+1}. F^{k+1} represents the features that will not be included in the next release. The following conditions are satisfied:

1) $\sum_{i=1}^{n} \left(\left(i \in F^{k+1}\right) \times EFFORT(i)\right) \leq AvailableEffort$

2) $S(i, j) = 1$ if tenant j is eligible to estimate feature i *else* $S(\mathrm{i}, \mathrm{j}) = 0$

3) $DEPENDENCIES(i, j) = 1 \rightarrow RANK(i) == RANK(j)$ and $DEPENDENCIES(i, j) = 2 \rightarrow RANK(i) \geq RANK(j)$

V. PROPOSED SOLUTION

Most of variables in release planning are exposed to uncertainty due to inadequate knowledge and ambiguity of features [28]. In addition, the provided judgments about importance, risk, and required effort of the features are subjective and human-dependent. FIS (and its extension HFIS) are appropriate techniques to handle the uncertainty caused by human factors. Furthermore, any solution for release-planning problems shall use human expertise supported by computational models. In many decision-making problems, fuzzy rules have proved highly effective in representing human expertise [29]. In this paper, we use HFIS of Mamdani types [24] as the core component to build the fuzzy rules, and then we combine the stakeholders' estimates using these rules to calculate a rank for each feature. The features with the highest ranks shall be assigned to the next release.

As it is illustrated in Figure 1, the proposed approach consists of four processes: data collection, preprocessing, ranking, and release plan generation.

A. DATA COLLECTION

The release management starts collecting and organizing the data about the factors of release planning process. The outputs of this process are Tenants' Weights (TW), type of service matrix (S), Importance of the features (IMP), risk of features ($RISK$), required effort ($EFFORT$), available effort ($AvailableEffort$), dependencies among features ($DEPENDENCIES$), and commonality of features ($COMMONALITY$). This data can be gathered using a web-based application developed to this purpose.

B. PREPROCESSING

The inputs to this process are Tenants' Weights (TW), type of service matrix (S), Importance of the features (IMP). We use these three data structures to produce a vector that contains the importance of feature while considering the weights of

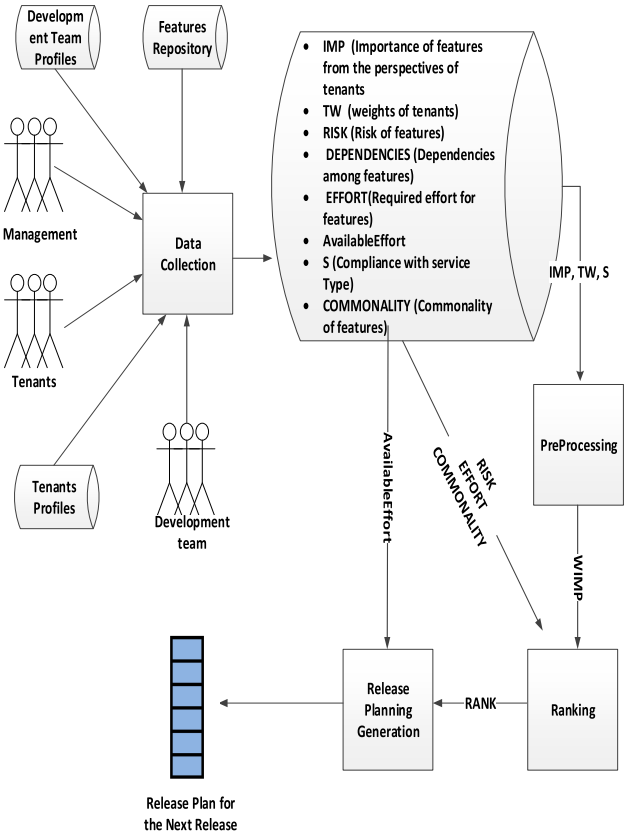

FIGURE 1. The proposed solution architecture.

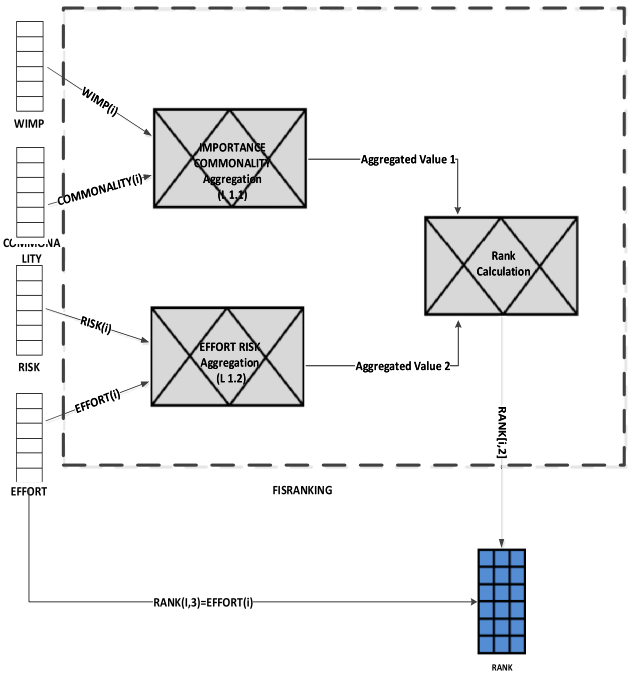

FIGURE 2. Ranking process.

tenants and their eligibility to judge the features. We define *WIMP* (weighted importance) as an n elements vector such that $WIMP = IMP. * S * TW$ where $(.^*)$ is the dot product operation of matrix.

C. RANKING

The ranking process uses a HFIS engine to calculate a rank of each feature. The inputs to RANKING process are:

WIMP, *RISK*, *EFFORT* and *COMMONALITY* vectors.

The output is $n \times 3$ matrix, and we call it *RANK*. It contains the information of the features after applying ranking process. The first column of *RANK* contains features identifiers $(1 \cdot \cdot n)$, the second will include the ranks of the features, and the third column contains the required effort of the features. The main component in ''Ranking'' process is the FIS engine, which we call *FISRANKING*. As Figure 2 shows, *FISRANKING* is HFIS that has two levels: the first level includes two parts: L1.1 which is the FIS-sub-module responsible for aggregating importance and commonality, and L1.2 which is the FIS-sub-module responsible for effort and risk aggregation. The second level receives the output of the two sub-modules in the first level to generate rank for features using predetermined fuzzy rules. It is important to state that the perspectives of the designer of the FIS engine is the main player in FIS approach. In other words, the degree of impact of a fuzzy variable depends on the way it is considered in the fuzzy rules. The elements of matrix *RANK* are defined as follows:

$RANK\ (\mathrm{i}, 1) = i,$

$RANK\ (\mathrm{i}, 2) =$

$FISRANKING\ (WIMP\ (i)\ , RISK\ (i)\ , EFFORT\ (i),$

$COMMONALITY\ (\mathrm{i}))$, and $RANK\ (\mathrm{i}, 3) = EFFORT\ (\mathrm{i})$.

The artifices of *FISRANKING* are defined as follows:

1)Four input linguistic variables: *IMPORTANCEVAR*, *RISKVAR*, *EFFORTVAR* and *COMMMNALITYVAR*. The linguistic values associated with these variables are defined by the experts. For example, in our experiments in section VI, we make $IMPORTANCEVAR = \{lowimp, midimp, highimp\}$ Each linguistic variable is associated with an input vector. *IMPORTANCEVAR* is associated with *WIMP*, *RISKVAR* is associated with *RISK*, *EFFORTVAR* is associated with *EFFORT*, and *COMMMNALITYVAR* is associated with *COMMONALITY*

2)If-then fuzzy rules: The rules shall be constructed in a way that increases the final rank of the highly important and high commonality features, and minimizes the rank of high risk and high effort features. The if-then fuzzy rules in this context are generated by the domain experts. When designing fuzzy rules, the likelihood of exponential increase in the total number of rules is a significant issue that should be taken into account. In our case, the maximum number of rules is:

$\prod_{\forall X \in FISVAR} |X|$ where $FISVAR = \{IMPORTANCEVAR, RISKVAR, EFFORTVAR, COMMONALITYV??R\}$. A huge number of rules ''may damage the transparency and

interpretation of FIS as humans are incapable of understanding and justifying hundreds or thousands of fuzzy rules and parameters'' [30].Therefore, it is necessary to use one of the techniques for rules reduction. In this paper, we chose the hierarchical fuzzy inference system [31], in which the fuzzy system is built in a hierarchical manner. Hierarchical fuzzy rules have proved to reduce the number of rules without affecting the approximation ability of the FIS [31]. In the hierarchy of fuzzy units, the outputs of lower levels are the inputs for higher levels. As it is stated, in our proposed model, we aggregate the inputs related to the development team and management using one FIS sub-module, and we aggregate the inputs related to the tenants using another FIS sub-module. After that, the outputs of these two intermediate FIS sub-modules are aggregated using a third FIS sub- module, which generates a rank for the feature under consideration

D. RELEASE PLANNING GENERATION

The first step in this process is that the values of the *RANK* matrix are tuned in order to satisfy the dependencies constraints. For coupling constraints, the ranks of the features that are coupled are adjusted to have the same rank, formally,$DEPENDENCIES\ (i, j)\ ==\ 1\ \rightarrow\ RANK\ [i, 2]\ =\ RANK\,[j, 2]$. For precedence constraints, if feature i precedes feature j then the rank of i should be greater than the rank of j. Formally, $DEPENDENCIES\ (i, j)\ ==\ 2\ \rightarrow\ RANK\ [i, 2]\ =$ RANK[j, 2] $+\ \epsilon$ where ϵ is a very small number that only affect the features that have precedence relationship. ϵ can be calculated by sorting RANK according to the second column (ranks). After that we find the minimum difference between two ranks (let us call *minrank*) and make $\epsilon\ <\ minrank$(for example we can make $\epsilon\ =minrank\,/10$).

After this tuning process, a greedy approach is applied; that is, the features with the highest rank are added to the release plan. This process is continued for as long as the available effort is not exceeded.

VI. VALIDATION

This section compares the proposed HFIS-based approach for release planning for SaaS with the Binary Linear Programming-based (BLP) approach presented in [2]. As in [2], we compare the two approaches using two performance metrics: the overall tenants' satisfaction, and the adherence to the risk. The overall tenants' satisfaction shows how much tenants are satisfied with the application. The more tenants' satisfaction, the more success of the application is. The adherence to risk shows the degree of the quality of the new release. As it is stated earlier, the risk of a feature is the impact of this feature on the schedule and the quality of the release. If the release plan is adhered to risk, then it considers the quality and time factors. In order to define the overall satisfaction, we should first define a formula to measure the degree of satisfaction for each tenant. Let ds_i be the degree of satisfaction for tenant ti. We define $Sats\ :\ T\ \times\ P(F^*)\ \times\ P(F^*)\ \rightarrow\ [01]$ as a function that calculates the degree of satisfaction for a tenant as follows:

$$ds_i = Sats\left(i, F_{ti}^{k+1}, F^{k+1}\right) = \frac{\sum_{k=1}^{n}\left(IMP(k,i)\times\left(f_k\in F^{k+1}\right)\times\left(f_k\in F_{ti}^{k+i}\right)\right)}{\sum_{j=1}^{n}\left(IMP(j,i)\times\left(f_j\in F_{ti}^{k+1}\right)\right)}$$

such that $P(F^*)$ is the power set of F^* and $ds_i \in [1, 0]$. In this calculation, F_{ti}^{k+1} is the release plan generated according to the perspective of t_i. Note that F_{ti}^{k+1} is just used to measure the degree of satisfaction of a tenant t_i about the generated release plan (F^{k+1}). This formula calculates the ratio of the cumulative importance of the features that found in the set $F_{ti}^{k+1}\ \cap\ F^{k+1}$ to the cumulative importance of the features that in the set F_{ti}^{k+1}. If $F_{ti}^{k+1}\ \subseteq\ F^{k+1}$, then the degree of satisfaction of tenant t_i is 1(100%), and if $F_{ti}^{k+1}\ \cap\ F^{k+1}\ =$ $\emptyset$, then degree of satisfaction is 0. The overall satisfaction ($OSat$) can be calculated as the additive weighting of the degree satisfactions of the tenants.

$$\mathrm{OSat} = \sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{ds_i} \times \mathrm{W\ (i)}$$

The maximum value OSat can take is 1, which means that the degree of satisfaction is 100%. The lowest value is 0, which means a degree of satisfaction of 0%.

Furthermore, the quality of the projected release plan can be evaluated by measuring the degree to which it considers the risk factor. Maximum adherence to the risk factor is desired. Let $\mathrm{F_{Risk}^{k+1}}$ be the release plan produced by fulfilling technical, contractual, and resource constraints, and by completely considering the risk factor, which means that the features with the lowest risk are assigned to the next release. We define the adherence to the risk factor (Ad_risk_factor) as follows:

$$Ad\_risk\_factor = \frac{|F_{risk}^{k+1}\ \cap\ F^{k+1}|}{|F_{risk}^{k+1}|}$$

Ad_risk_factor is the ratio of the number of features in $\mathrm{F_{risk}^{k+1}}\cap\mathrm{F^{k+!}}$ to the number of features in $\mathrm{F_{risk}^{k+1}}$. We say that a release plan has completely adhered to the risk factor when Ad_risk_factor $= 1$.

We use Matlab code to implement HFIS and BLP-based approaches. We use Monte Carlo simulation to model the probability of $OSat$ and Ad_risk_factor. We use normal distribution to generate random values of importance, available effort, required effort, commonality, service level and eligibility of tenants, and the technical constraints. After that, we run two scenarios.

Scenario 1: This scenario considers few numbers of tenants and features. The number of features which to be planed are varied from 10 to 20 features, and the number of tenants is ranged from 5 to 20. The aim of Scenario 1 is to compare between the proposed approach and the referenced one when there are limited numbers of features and tenants (which is not the case in most SaaS applications).

Scenario 2: This scenario considers medium to huge numbers of tenants and features. The number of features which to be planed are varied from 50 to 200 features, while the number of tenants is ranged from 50 to 500. The aim of Scenario 2 is to compare between the proposed approach and the referenced one when there are medium to huge numbers of features and tenants (which is the case in most SaaS applications). To be realistic regarding available effort, in all scenarios, we make the available effort takes a random value is in the range of 30% to 80% of the total required effort.

In each scenario, we generate 1000 release plans. For each release plan, and using the two approaches, we calculate the cumulative average of the overall satisfaction and the adherence to risk.

Figure 3 and 4 shows the results of Scenario 1 (few numbers of features and tenants). We can see that HFIS-based approach gives better results in the degree of overall satisfaction, while BLP-based outperforms the HFIS-based in the adherence to risk.

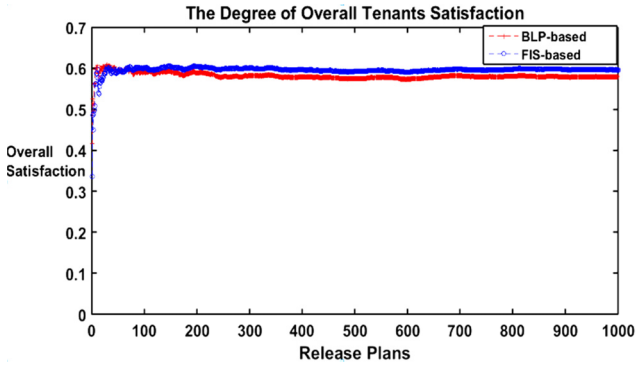

FIGURE 3. Overall satisfaction (Scenario 1).

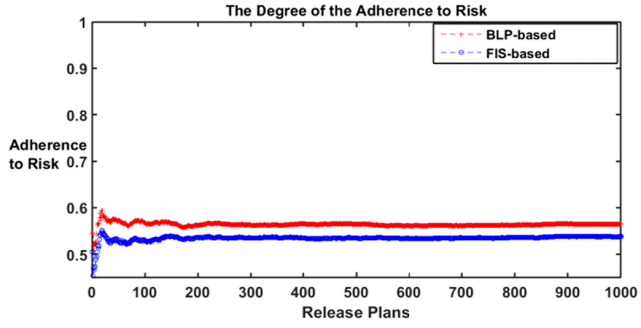

FIGURE 4. Adherence to risk (Scenario 1).

Figure 5 and shows the results of Scenario 2 (medium to huge numbers of tenants and features). We can see that HFIS-based approach outperforms BLP-based in both metrics.

This experiments show that the HFIS-based approach achieves better results when there are huge numbers of tenets and features, which is the case in most SaaS applications.

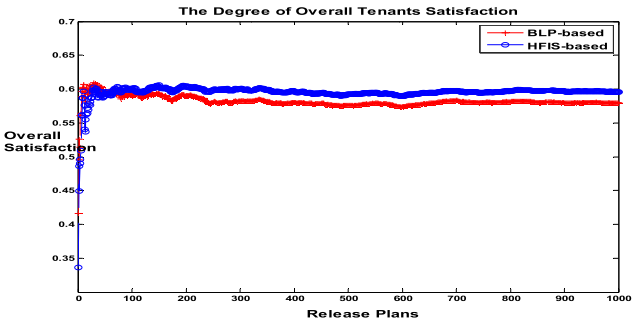

FIGURE 5. Overall satisfaction (Scenario 2).

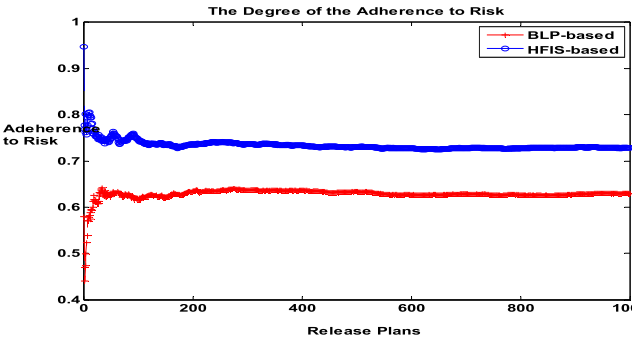

FIGURE 6. Adherence to risk (Scenario 2).

VII. CONCLUSION

The next release planning is the proceess of selecting the most promising features that will be included in the next release. The selected features shall maximize tenants satisfaction and consider the factors of risk, technical, effort, and type of service constatints. The variables that control SaaS applications include: the importance of each feature as perceived by the different tenants, the decision weights of the tenants, the potential risks along with the required effort that are associated with each feature as estimated by the members of the development team, the available effort allocated to deliver the release as estimated by release management, the technical dependencies among features, type of service documents, and the degree of commonality of features. In this work, we propse a novel approach for release planning to tackle the ''next release'' planning problem in SaaS applications. This approach is aprioritization approach that employs a Hirarichical Fuzzy Inference System (HFIS) in order to generate a rank for each feature. The rank of a feature represents its priority among other features. After that, the ranks are adjusted to reflict the dependencies constraints. Then, with considering required and avialable effort, the features are sorted and prioritized using a greedy approach. The features with the highest ranks are assigned to the next release. The propsed approach shows promosing results comaperd to the binary linear programmimg approach from the perspective of overall satisfaction and adherence to risk.

As a further line of research, the architecture of SaaS applications can be considered when release plaiing process

is conducted. For example, many applications use services from other vendors which makes the risk of integration a key factors on the planning process. Furthermore, the reliability of information can be considered by using another input from the stakeholder that express their sureness about their judgments. The application of Z-numbers [32] can be used in this case.

REFERENCES

[1] B. Sengupta and A. Roychoudhury, "Engineering multi-tenant software-as-a-service systems," in *Proc. 3rd Int. Workshop Princ. Eng. Service-Oriented Syst. (PESOS)*, 2011, pp. 15–21.
[2] M. Alrashoud, L. Ahmed, and A. Abhari, "Binary linear programming-based release planning for multi-tenant business SaaS," in *Proc. Int. C* Conf. Comput. Sci. Softw. Eng.*, New York, NY, USA, 2014, Art. no. 15.
[3] G. Ruhe and M. O. Saliu, "The art and science of software release planning," *IEEE Softw.*, vol. 22, no. 6, pp. 47–53, Nov./Dec. 2005.
[4] D. Greer and G. Ruhe, "Software release planning: An evolutionary and iterative approach," *Inf. Softw. Technol.*, vol. 46, no. 4, pp. 243–253, 2004.
[5] G. Ruhe and A. N. The, "Hybrid intelligence in software release planning," *Int. J. Hybrid Intell. Syst.*, vol. 1, nos. 1–2, pp. 99–110, 2004.
[6] M. M. Gupta, "On fuzzy logic and cognitive computing: Some perspectives," *Scientia Iranica*, vol. 18, no. 3, pp. 590–592, 2011.
[7] Y. Zhang, L. Peng, Y. Sun, and H. Lu, "Editorial: Intelligent industrial IoT integration with cognitive computing," *Mobile Netw. Appl.*, vol. 23, pp. 185–187, Apr. 2018.
[8] M. S. Hossain and G. Muhammad, "Emotion-aware connected healthcare big data towards 5G," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2399–2406, Aug. 2018.
[9] M. Felderer, J. Ho, A. Beer, and G. Ruhe, "Industrial evaluation of the impact of quality-driven release planning," in *Proc. 8th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Sep. 2014, Art. no. 62.
[10] Y. Zhang, X. Ma, S. Wan, H. Abbas, and M. Guizani, "CrossRec: Cross-domain recommendations based on social big data and cognitive computing," *Mobile Netw. Appl.*, vol. 23, no. 6, pp. 1610–1623, Dec. 2018.
[11] M. S. Hossain and G. Muhammad, "Emotion recognition using deep learning approach from audio-visual emotional big data," *Inf. Fusion*, vol. 49, pp. 69–78, Sep. 2019.
[12] Y. Hao, J. Yang, M. Chen, M. S. Hossain, and M. F. Alhamid, "Emotion-aware video QoE assessment via transfer learning," *IEEE Multimedia*, vol. 26, no. 1, pp. 31–40, Jan./Mar. 2019.
[13] Y. Zhang, X. Ma, J. Zhang, M. S. Hossain, G. Muhammad, and S. U. Amin, "Edge intelligence in the cognitive Internet of Things: Improving sensitivity and interactivity," *IEEE Netw.*, vol. 33, no. 3, pp. 58–64, May/Jun. 2019.
[14] G. Ruhe, "A systematic approach for solving the wicked problem of software release planning," *J. Soft Comput.*, vol. 12, pp. 95–108, Jan. 2008.
[15] D. Ameller, C. Farré, X. Franch, and G. Rufian, "A survey on software release planning models," in *Proc. Int. Conf. Product-Focused Softw. Process Improvement*, 2016, pp. 48–65.
[16] M. R. Karim and G. Ruhe, "Bi-objective genetic search for release planning in support of themes," in *Search-Based Software Engineering* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Cham, Switzerland: Springer, 2014.
[17] N. Agarwal, R. Karimpour, and G. Ruhe, "Theme-based product release planning: An analytical approach," in *Proc. 47th Hawaii Int. Conf. Syst. Sci.*, 2014, pp. 4739–4748.
[18] G. Zorn-Pauli, B. Paech, T. Beck, H. Karey, and G. Ruhe, "Analyzing an industrial strategic release planning process—A case study at roche diagnostics," in *Requirements Engineering: Foundation for Software Quality* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Berlin, Germany: Springer, 2013.
[19] W. Shen, "Software release planning with fuzzy objectives and constraints," M.S. thesis, Dept. Elect. Comput. Eng., Univ. Calgary, Calgary, AB, Canada, 2005.
[20] M. Alrashoud and A. Abhari, "Perception-based software release planning," *Intell. Automat. Soft Comput.*, vol. 21, no. 2, pp. 175–195, 2015.
[21] M. Alrashoud and A. Abhari, "Planning for the next software release using adaptive network-based fuzzy inference system," *Intell. Decis. Technol.*, vol. 11, no. 2, pp. 153–165, 2017.
[22] M. I. Ullah and G. Ruhe, "Towards comprehensive release planning for software product lines," in *Proc. 1st Int. Workshop Softw. Product Manage. (IWSPM)*, vol. 2006, pp. 51–55.
[23] J. Xuan, H. Jiang, Z. Ren, and Z. Luo, "Solving the large scale next release problem with a backbone-based multilevel algorithm," *IEEE Trans. Softw. Eng.*, vol. 38, no. 5, pp. 1195–1212, Sep./Oct. 2012.
[24] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Studies*, vol. 7, pp. 1–13, Jan. 1975.
[25] O. Saliu and G. Ruhe, "Supporting software release planning decisions for evolving systems," in *Proc. 29th Annu. IEEE/NASA Softw. Eng. Workshop (SEW)*, Apr. 2005, pp. 14–24.
[26] H. J. La and S. D. Kim, "A systematic process for developing high quality SaaS cloud services," *Cloud Computing* (Lecture Notes in Computer Science), vol. 5931. Berlin, Germany: Springer, 2009, pp. 278–289.
[27] G. Ruhe and D. Greer, "Quantitative studies in software release planning under risk and resource constraints," in *Proc. Int. Symp. Empirical Softw. Eng. (ISESE)*, 2003, pp. 262–270.
[28] A. Al-Emran, P. Kapur, D. Pfahl, and G. Ruhe, "Studying the impact of uncertainty in operational release planning—An integrated method and its initial evaluation," *Inf. Softw. Technol.*, vol. 52, pp. 446–461, Apr. 2010.
[29] W. Di, X. Zeng, and J. Keane, "A survey of hierarchical fuzzy systems," *Int. J. Comput. Cognition*, vol. 4, no. 1, pp. 18–29, 2006.
[30] R. R. Yager, "On a hierarchical structure for fuzzy modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 4, pp. 1189–1197, Jul. 1993.
[31] C. Wei and L.-X. Wang, "A note on universal approximation by hierarchical fuzzy systems," *Inf. Sci.*, vol. 123, pp. 241–248, Apr. 2000.
[32] L. A. Zadeh, "A note on Z-numbers," *Inf. Sci.*, vol. 181, no. 14, pp. 2923–2932, 2011.