Received: 10 March 2023 | Revised: 9 July 2023 | Accepted: 27 July 2023

DOI: 10.1002/sys.21715

REGULAR ARTICLE

Minimising conflicts among run-time non-functional requirements within DevOps

Souvick Das[1] | Novarun Deb[2] | Nabendu Chaki[3] | Agostino Cortesi[1]

[1]Ca' Foscari University, Venice, Italy

[2]Indian Institute of Information Technology Vadodara, Vadodara, India

[3]University of Calcutta, Kolkata, India

Correspondence
Souvick Das, Department of Informatics and Statistics, Ca' Foscari University, Venice, 30170, Italy.
Email: souvick.das@unive.it

Funding information
PNRR, Grant/Award Numbers: ECS 00000043, SPIN2021

Abstract
Significant contributions in the existing literature highlight the potential of softgoal interdependency graphs towards analyzing conflicting non-functional requirements (NFRs). However, such analysis is often at a very abstract level and does not quite consider the run-time performance statistics of NFR operationalizations. On the contrary, some initial empirical evaluations demonstrate the importance of the run-time statistics. In this paper, a framework is proposed that uses these statistics and combines the same with NFR priorities for computing the impact of NFR conflicts. The proposed framework is capable of identifying the best possible set of NFR operationalizations that minimizes the impact of conflicting NFRs. A detailed space analysis of the solution framework helps proving the efficiency of the proposed pruning mechanism in terms of better space management. Furthermore, a Dynamic Bayesian Network (DBN) - based system behavioral model that works on top of the proposed framework, is defined and analyzed. An appropriate tool prototype for the framework is implemented as part of this research.

KEYWORDS
NFR conflict, NFR contribution, NFR conflict minimality, NFR quality

1 | INTRODUCTION

DevOps is a logical continuation of the Agile software journey and combines Lean development strategies with the Theory of Constraints and the Toyota Kata movement[1] characterized by continuous development, continuous testing, continuous integration (CI), and continuous deployment (CD) of potentially shippable increments. The principles and foundations of DevOps have been found to be very fruitful in the software industry over the last decade. Apart from CI/CD activities, the DevOps ecosystem has also made heavy use of continuous monitoring tools - like the ELK stack[1] and Nagios.[22]

Non-functional requirements (NFRs) are critical in determining the quality of the services being delivered through a solution.[3,4] Existing literature[5,6] broadly classify NFRs (or quality attributes) into two categories. The first category consists of those NFRs which describe some property of the system at runtime - like *Availability*, *Performance*, and *Usability*. These quality attributes are called *run-time* NFRs. The other category of quality attributes are called *design-time* NFRs which describe some property for the design and development of the system - such as *Modifiability*, *Testability*, and *Portability*. In order to capture the interactions among NFRs a framework that uses Softgoal Interdependency Graphs (SIGs) was introduced in literature[7]. Such graphical interactions are based on the abstract characteristics of the NFRs involved. However, SIGs are unable to anticipate the collective run-time performance statistics of a set of NFRs that have

[1] https://aws.amazon.com/what-is/elk-stack/
[2] https://www.nagios.org/documentation/

been operationalized and deployed for a particular solution. In fact, the run-time characteristics for a set of NFRs depend on the operational infrastructure where the solution has been deployed.[8] When managing multiple NFRs in a system, conflicts may arise due to various factors such as shifts in the business environment, changes to operational infrastructure, and evolving requirements. These conflicts can pose significant challenges and require careful consideration to ensure the NFRs are satisfied in an optimal manner. Undiscovered NFR conflicts can have a detrimental impact on the performance of an operational system. For instance, if one requirement emphasizes high speed while another demands low resource consumption, finding a compromise may result in slower performance. Furthermore, undiscovered NFR conflicts may also introduce security vulnerabilities. For example, if one requirement prioritizes user convenience while another emphasizes strong security measures, compromises made to satisfy both may weaken the overall security posture. These scenarios highlight the significance of identifying and addressing non-functional requirement conflicts.Perceptions of NFR conflicts can evolve as data volumes increase and more efficient NFR operationalizations are developed. NFR operationalizations refer to possible solutions (or algorithms/techniques) that can be used to realize an NFR in the software component or product, such as *AES*, *DES*, *Blowfish* for the *Confidentiality* requirement. Higher-level SIGs may not always capture real-time operational environments. Continuous monitoring of run-time NFR characteristics is necessary to identify and effectively manage conflicts.

Another aspect of NFRs that has been studied in literature[5] is the quantitative assessment of these requirements, which involves two key concepts: *Response* and *Response Measure*. The *Response* specifies the activities that the system should perform when a stimulus arrives. The *Response Measure* tries to assess the degree of satisfaction for a given response. Many current approaches[9–11] for detecting conflicts in NFRs are based on potential conflict catalogs or rules. However, while these frameworks can effectively identify conflicts during software development, relying on catalogues or dependency rules results to be too restrictive, due to the dynamic nature of NFR conflicts.

The specific research gap that can be observed is that no data-driven framework has been introduced so far that quantifies and minimises the degree of NFR conflicts among NFR operationalizations based on the real-time response measures observed for each individual NFR.[12] In a more generic sense, there is no mechanism in place that can collectively identify the best possible set of NFR operationalizations which minimise the degree of conflicts between the participating NFRs.[13,14] The problem becomes even more complex when the business environment is dynamic and keeps evolving - both with respect to the desired set of NFRs and the corresponding operationalizations that are available.

This paper attempts to address this research problem by proposing a suitable and efficient NFR-conflict Minimization Framework based on real-time NFR response measures to provide conflict-minimal *NFR solutions*. An *NFR solution* refers to the solution that addresses all the NFRs that are involved with a particular software product. It is defined by the set of individual operationalizations that are used to implement (or satisfy) the participating NFRs. Notice that in the DevOps ecosystem, the possibility of continuously monitoring performance characteristics in real-time operational environments aligns perfectly with our objective to measure the run-time performance characteristics of NFRs. Our NFR -conflict Minimization Framework deals with *run-time* NFRs only. *Design-time* NFRs are clearly out of the scope of this research work.

The main contributions of this paper can be summarized as follows:

1. A mechanism for estimating the individual contribution of NFR operationalizations to their respective NFRs based on run-time measurement of certain properties associated with that NFR.
2. A mechanism to compute the relative conflicts between pairs of NFR operationalizations, parameterized on NFR priorities assigned by the developer.
3. A conflict minimization framework that provides the mathematical basis for quantification and derivation of conflict-minimal *NFR solutions* in the reduced solution space.
4. A tool interface that allows developers to set their input parameters and discover the conflict-minimal NFR solution using the proposed framework.

The contributions of the research paper are relevant to the developer community in several ways:

- *Better understanding of NFR conflicts*: The paper provides insights into the nature and types of NFR conflicts in software architecture.
- *Practical solutions for NFR conflict management*: The paper suggests techniques for managing NFR conflicts, such as monitoring run-time NFR characteristics and using adaptive middle-ware.
- *Improved software quality*: The paper helps create software that meets required quality standards by managing NFR conflicts that can impact software quality.
- *Increased productivity*: By providing practical solutions for NFR conflict management, developers can spend less time resolving conflicts and more time on developing software features, leading to increased productivity.

Overall, the contributions of the research paper are relevant to the developer community as they provide insights, solutions, and best practices for managing NFR conflicts in software architecture, which can improve software quality, productivity, and customer satisfaction within the DevOps framework.

The rest of the paper is organized as follows. Section 2 presents a motivating experiment to motivate and highlight the need for continuous monitoring of NFR Response Measures in the runtime environment. In the subsequent sections, the proposed NFR Conflict Minimization Framework has been elaborated. Section 3 presents an overview of the framework's architecture, while Section 4 explores the algorithms that are employed to implement the NFR Conflict Minimization framework. Additionally, the solution space management scheme has also been discussed in this section. The Section 5 demonstrates how the proposed framework manages the exponential growth of the solution space and deliver the conflict minimal solution. Section 6 presents a detailed review of the related works. The paper concludes with Section 7.

2 | MOTIVATING EXPERIMENTS

In order to motivate and highlight the need for continuous monitoring of NFR Response Measures in run-time operations environment, a simple yet detailed set of empirical experiments has been performed. This section documents the observations and inferences obtained from these experiments and, thereby, underlines the importance of the proposed solution.

The following details may be noted for better understanding the experimental settings.

- Our case study considers *Bandwidth Efficiency* as the only NFR required in the product.
- Data compression is considered for achieving *Bandwidth Efficiency*, assuming that compressed data requires less communication bandwidth, but other methods may also achieve the same goal.
- An empirical experiments with three specific NFR operationalizations (in this case, compression algorithms) - Lempel-Ziv (or LZ), ZIP, and Prediction by Partial Match (PPM) has been performed.
- As a testbench, five different publicly available datasets has been considered. These datasets are represented as $Data - i, 1 \leq i \leq 5$. These datasets are accessible in the GitHub Repository.[3]
- The dataset is split into four fragments: 10,000, 20,000, 30,000, and 40,000 records, respectively. Each fragment builds upon the previous one, reflecting the incremental generation of software usage data over time.

The summary of the detailed observations made during this empirical study have been documented in the following three subsections.

2.1 | Evolution of histograms with time

The first set of experiments are aimed at demonstrating the evolving nature of run-time performance measures and how they affect the frequency distributions of these measures.The experiments have been performed as follows:

- The compression of data records within each of the five data sets, namely $Data - 1$ to $Data - 5$, is performed for each compression algorithm.
- For each data set, the compression algorithm is applied on the four incremental partitions containing 10,000 to 40,000 records.
- For each histogram plot, the x-axis represents the percentage of compression achieved and the y-axis represents the fraction of data records that are compressed by a certain percentage.
- Compression percentage histograms for each ⟨*dataset*, *compression algorithm*⟩ pair are plotted together to assess if the frequency distribution of records remains consistent as the dataset size increases.
- The resulting histograms for all the fifteen (15) pairs - combining five (5) datasets and three (3) compression algorithms - are shown in Table 1.

Observation
We observe that, regardless of datasets and compression algorithms, the histogram shape changes as dataset size increases over time. This is visually evident when zooming into the histograms in Table 1, where different colored peaks of the x-axis indicate varying fractional distributions of records. It is important to note that, a threshold value may exist for an NFR's response measure, which must be met by a software system during runtime. Monitoring the variation of threshold values with histogram percentiles is required to determine the number of records that satisfy these thresholds.

2.2 | Variation of the threshold with fixed percentile

Building on initial insights, detailed experiments have been conducted to observe threshold value changes for growing datasets. The first set of experiments focuses on fixing the percentile to 70% for LZ, ZIP, and PPM applied on four fragments of dataset $Data - 1$. Using Matlab, the specific threshold values (C) are determined that achieve the 70th percentile. The resulting twelve histograms, along with their corresponding threshold values, are presented in tabular format in Table 2. Similar experiments were conducted for the remaining datasets ($Data - 2$, $Data - 3$, $Data - 4$, and $Data - 5$). The corresponding histograms and thresholds can be found in the GitHub Repository.[4]

Observation
Incremental dataset growth in our empirical experiments results in changing threshold values for achieving desired percentiles. This aligns with the evolving frequency distributions of histograms (documented in Section 2.1), indicating a corresponding adaptation of threshold values without anomalies.

2.3 | Variation of percentile with fixed threshold

The next set of empirical studies was conducted to observe the orthogonal behavior of the system. Experiments were conducted by fixing the threshold and identifying how it affects the percentile of the histogram when the data set grows incrementally with time. Three different

[3] https://github.com/Souvick-CU/PDANFR/tree/main/Dataset

[4] https://github.com/Souvick-CU/PDANFR

TABLE 1 Superimposed histograms of incremental fragments derived by the three different compression algorithms applied on the five data sets.

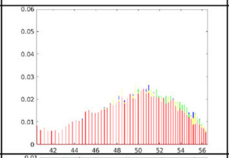
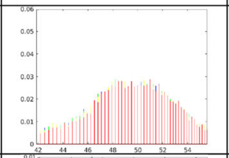
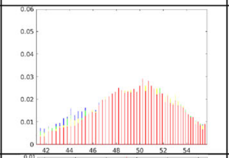
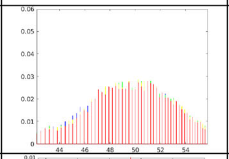
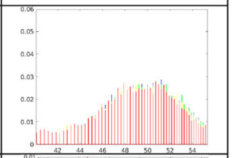
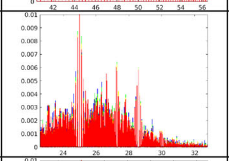
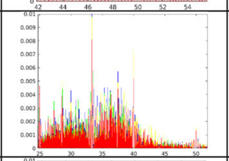
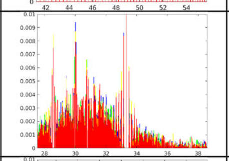
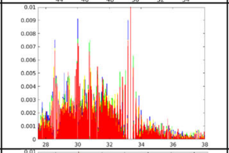
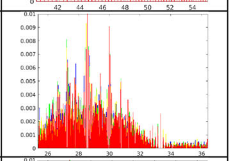
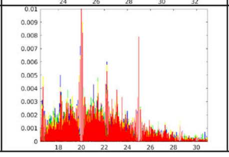
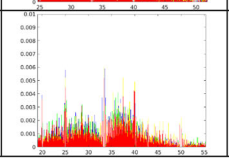
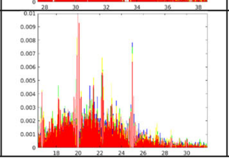
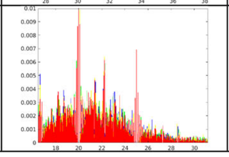
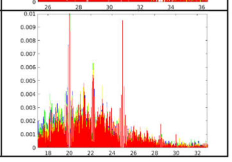
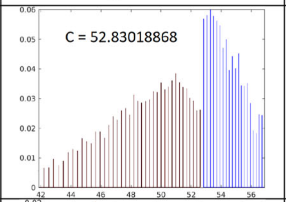
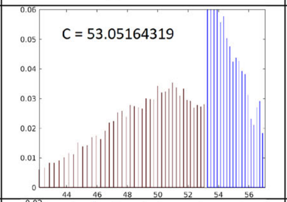
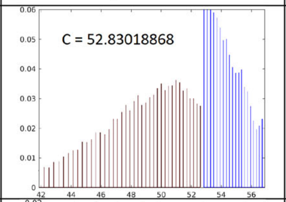
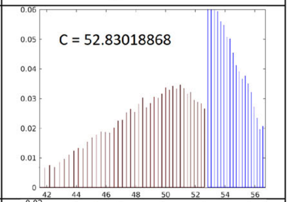
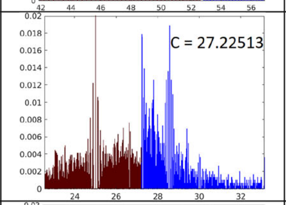
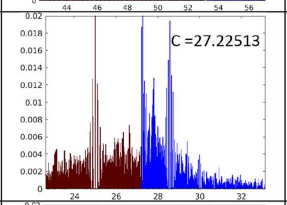
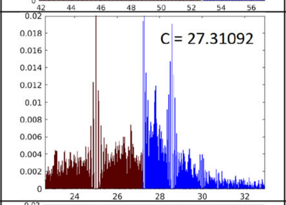
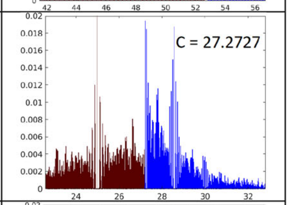
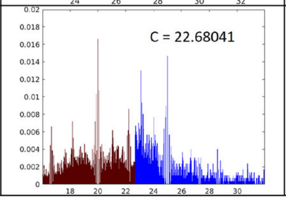
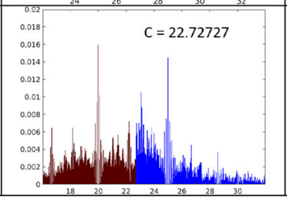
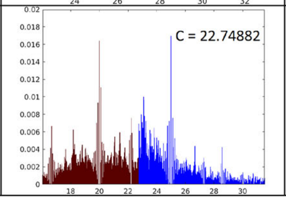
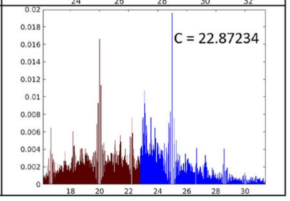
	Data - 1	Data - 2	Data - 3	Data - 4	Data - 5
Lempel-Ziv (LZ)					
ZIP					
Prediction by Partial Match (PPM)					

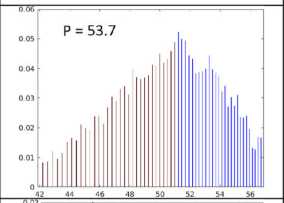
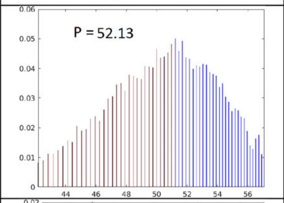
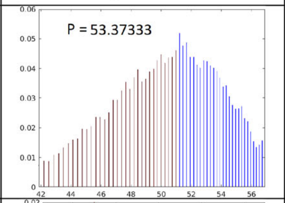
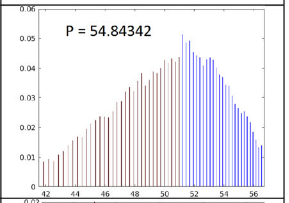
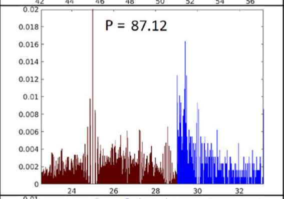
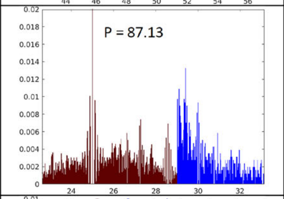
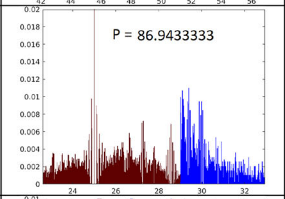
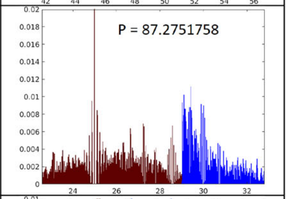
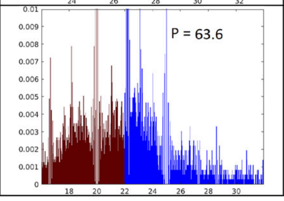
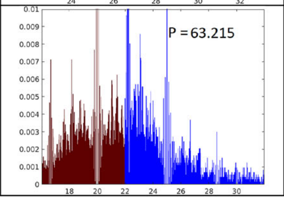
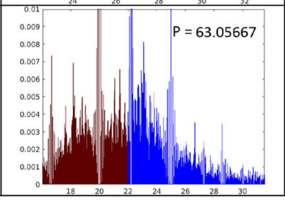
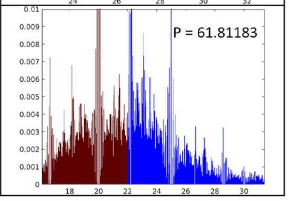
TABLE 2 Variation of threshold in the histograms generated for the four fragments of dataset *Data* − 1 with percentile $P = 70$ (fixed). The color of the histogram changes when the desired percentile is achieved.

	Data - 1 Fragment - 1	Data - 1 Fragment - 2	Data - 1 Fragment - 3	Data - 1 Fragment - 4
Lempel-Ziv (LZ)				
ZIP				
Prediction by Partial Match (PPM)				

thresholds of compression percentage for the three NFR operationalizations LZ, ZIP, and PPM have been considered. Based on the average performance of the individual algorithms, the thresholds are fixed at 51%, 29% and 22% for LZ, ZIP and PPM, respectively. The threshold remains fixed for each compression algorithm and for the four fragments of data set *Data* − 1. Next, again experiments on Matlab have been conducted and observed what is the percentile of satisfaction achieved by each algorithm. Here again we obtain twelve (12) different histograms which are plotted as a tabular structure in Table 3. The percentile (*P*) for each histogram, which satisfies the specific threshold, is mentioned on the respective histogram plots. Similar experiments have been performed for the four other data sets (*Data* − 2, *Data* − 3, *Data* − 4, and *Data* − 5) as well. All the data sets are provided in GitHub Repository.[5] The corresponding histograms and the percentile of satisfaction obtained against the specific threshold of compression percentage, are shown in the Annexure. The details are also provided in the GitHub Repository.[6]

[5] https://github.com/Souvick-CU/PDANFR/tree/main/Dataset
[6] https://github.com/Souvick-CU/PDANFR

TABLE 3 Variation of percentile in the histograms generated for the four fragments of dataset *Data* − 1 with specific threshold values (C). The color of the histogram changes at the threshold.

	Data - 1 Fragment - 1	Data - 1 Fragment - 2	Data - 1 Fragment - 3	Data - 1 Fragment - 4
Lempel-Ziv (LZ) C = 51				
ZIP C = 29				
Prediction by Partial Match (PPM) C = 22				

Observation

We may conclude from our empirical experiments that the percentile of records that satisfies a certain threshold also varies, as the data set increases gradually with time. This is also intuitively based on our initial experiments (described in Section 3.1) which demonstrate that the histogram shape varies with time as the data set grows in size. This entails that there should also be a change in the percentile that satisfies a specific threshold and no anomaly is observed in this respect as well.

From our experiments, we infer that there is a strong need to monitor the NFR response measures of a system as they keep evolving over time. This motivates our research and we proceed to explain the NFR Conflict Minimization Framework in the next Section.

3 | THE NFR CONFLICT MINIMIZATION FRAMEWORK

As depicted in Figure 1,[7] a NFR Conflict Minimization Architecture can be defined over the application development life cycle. The proposed architecture is "*data-centric*" in the sense that the best choice of NFR operationalizations (based on user defined NFR priorities) is evaluated from end-user data that is generated by the application itself. The proposed architecture allows the modification of an existing choice of NFR operationalizations (for conflict minimization purposes) by considering newly available operational data and the application's usage statistics in the most recent time window. In this section, a brief overview of the functioning of the NFR Conflict Minimization Architecture has been provided.

We consider a DevOps-based app development lifecycle where the application keeps evolving rapidly in iterations based on changing requirements and feedback coming from the end-users. DevOps lifecycles are characterized by CI/CD/CM (Continuous Integration/Continuous Deployment/Continuous Monitoring) pipelines. Once a minimum viable product (MVP) is delivered by the DevOps pipeline, we start generating usage statistics which can be closely monitored using continuous monitoring tools like the ELK stack, or Nagios. The proposed NFR Conflict Minimizer takes the usage statistics as input, along with the NFR catalogs containing NFR operationalizations, and their assigned priorities. The NFR Conflict Minimizer quantitatively evaluates the multiple NFR solutions (based on the most recent usage statistics) and identifies the one having the best quality metrics. This *data-centric* approach is one of main novelties of our proposed framework. The following sections (and the rest of the paper) elaborate on how the NFR Conflict Minimizer may be formalized and deployed in real-life scenarios.

Before introducing the NFR Conflict Minimizer and its formalization, let us consider a reference scenario that will be used to understand the individual components and their functionalities. All our examples are linked to this reference scenario. The scenario has been designed using *run-time* NFRs only, keeping in mind the scope of this work.

Reference Scenario

Let us consider a high-level goal *MessageTransfer* in the communication between two hosts. This goal is decomposed into three subgoals

[7] https://www.atlassian.com/devops

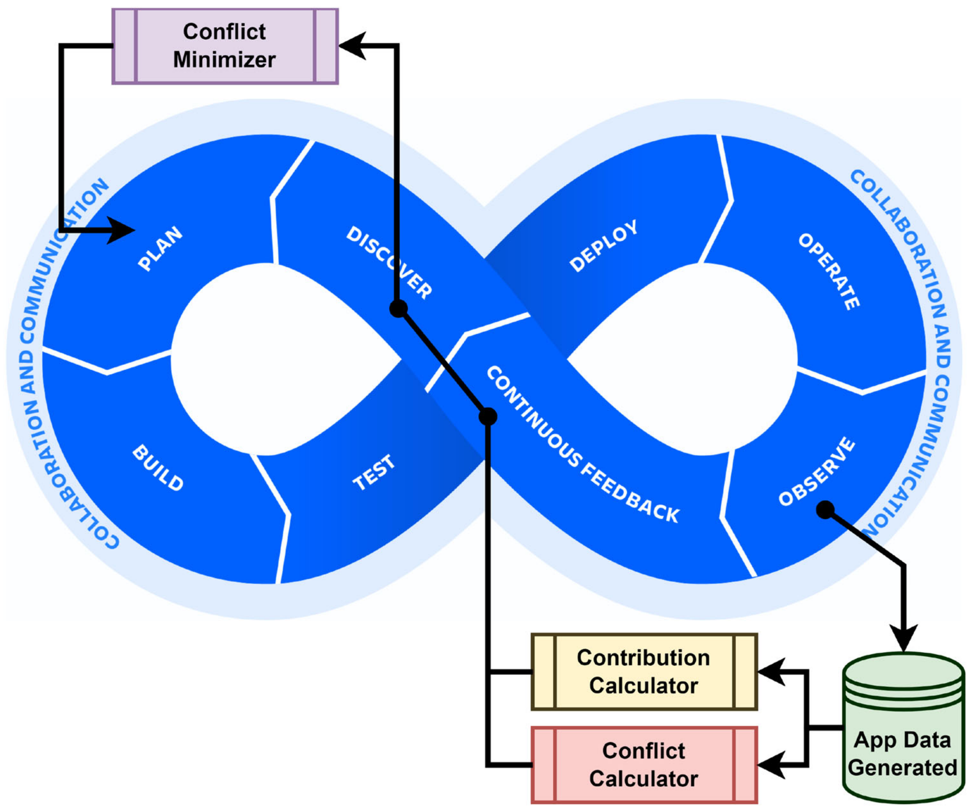


FIGURE 1 The NFR conflict minimization framework over the DevOps lifecycle. NFR, non-functional requirement.

- *CreateMessage*, *EstablishConnection*, and *SendMessage*. Consider three high-level NFRs, *Authentication*, *BandwidthEfficiency* and *Confidentiality*, that needs to be satisfied by the *MessageTransfer* goal. Let the NFR catalog for *BandwidthEfficiency* identify three different operationalizations - *Prediction by Partial Matching (PPM)*, *Context Mixing (CM)*, and *Lempel Ziv (LZ)*. Each of these operationalizations have their own unique performance characteristics like compression/decompression rate, and compression efficiency. Similarly, the NFR catalog for *Confidentiality* contains four encryption strategies - *Advanced Encryption Standard (AES)*, *Data Encryption Standard (DES)*, *Triple DES (3DES)* and *Blowfish* - and the possible *Authentication* strategies include *Fingerprint Scan (FnPr)*, *Face Recognition (FcRe)*, *Password (Pwd)*, and *Pattern Scan (PtrS)*. There are 48 choices of NFR solutions ($3 \times 4 \times 4$) that defines a spectrum ranging from *strongly conflicting* to *non-conflicting* solutions. The need for conflict minimization between NFRs becomes evident.

The factors which effect any NFR conflict minimization process are as follows:

1. A quantitative estimate of the level of satisfaction achieved by different operationalizations towards their corresponding high-level NFRs.
2. A quantitative estimate of how operationalizations interact (or conflict) with each other and across multiple high-level NFRs.
3. Different developers (or enterprizes) may prioritize the high-level NFRs differently, as per their requirements. A correlation mechanism that changes the quality of an NFR solution when the respective priorities are changed.

In order to achieve these objectives, the proposed NFR Conflict Minimzer should perform a quantitative evaluation of the quality metrics for each NFR solution (representing NFR priorities and operationalizations chosen by the developer). The NFR Conflict Minimizer consists of three different sub-components as described below:

(a) The *Contribution Calculator* module is a static process that fetches the contribution value of chosen leaf-level operationalizations by accessing the NFR repository. It is static in the sense that these values do not change for individual applications and may be periodically updated by the repository manager based on newly generated usage statistics.

(b) The *Conflict Calculator* module is a dynamic process that evaluates the relative conflict values for a chosen set of NFR operationalizations. This module is dynamic because the conflict values are evaluated based on the assigned priorities and pre-defined threshold values for the satisfaction of each high-level NFR.

(c) The *Conflict Minimization Module* takes the contribution and relative conflict values to evaluate the feasibility of the chosen NFR solution and identify (or suggest) the least conflicting NFR solution. The developer can choose the proposed solution or he/she can choose to proceed with their selected solution of their choice.

The developer is also provided an interface to change the assigned priorities to NFRs and re-invoke the conflict minimization module to recompute the least conflicting solution based on changed NFR priorities.

In general, run-time NFRs associated with goals can be realized with the help of *operationalization traces*. An *Operationalization Trace* represents an instantiation of an NFR operationalization when applied over some goal. The relation between an NFR operationalization and its trace is similar to that between classes and their objects. When a trace is instantiated over a high-level (or non-leaf level) goal, then it can be recursively transferred to its lower level goals based on the type of decompositions.

Example 1. In the reference scenario above, let us suppose that the developer chooses *AES* as the operationalization of *Confidentiality*. *AES(MessageTransfer)* is the operationalization trace that gets instantiated for app generation. Since the *MessageTransfer* goal is further decomposed into three subgoals, the *AES(MessageTransfer)* trace can be realized by the following three sub-traces:

(i) *AES(CreateMessage)*: Encrypts the message to be transferred.
(ii) *AES(EstablishConnection)*: Key exchange between sender and receiver for decryption at the receiver's end.
(iii) *AES(SendMessage)*: Sends the encrypted message over the communication channel.

3.1 | Validation strategy

The proposed minimization framework in the DevOps cycle has been experimentally evaluated by providing evidence of the fact that the *Conflict Minimizer* makes use of the information collected dynamically by the Contribution and Conflict Calculator and allows to assign values to the conflicting NFRs that result in better performance of the overall system. This will be showcased in the experimental results presented in Section 4.3, where Table 9 compares the selected solution with respect to the alternative scenarios. The soundness of the minimization process relies on standard optimization computation over the quality functions corresponding to each feasible solution.

3.2 | The contribution calculator module

Let us have a goal model specification of the form "g `demands` $N_1, N_2, \ldots, N_k$", where g is a goal and $N_1, N_2, \ldots, N_k$ are high-level NFRs with catalogs stored in the NFR repository. If op_i denotes the operationalization chosen and π_i denotes the priority assigned for each high-level NFR N_i, then a possible NFR solution could be $S = \{\langle op_1, \pi_1\rangle, \langle op_2, \pi_2\rangle, \ldots, \langle op_k, \pi_k\rangle\}$ such that $\pi_i < \pi_{i+1}$ implies that N_i has higher priority than N_{i+1}. The trace can be defined for such an NFR solution as follows:

Definition 3.1 (Traces as Instantiation functions). Let $\mathbb{G}$ denote the set of goals within a goal model specification. The trace corresponding to the NFR solution $S = \{\langle op_1, \pi_1\rangle, \langle op_2, \pi_2\rangle, \ldots, \langle op_k, \pi_k\rangle\}$ on a goal $g \in \mathbb{G}$, with all operationalizations arranged in decreasing order of priority (i.e., $\forall i, \pi_i < \pi_{i+1}$), can be represented as: $op_k(op_{k-1}(\cdots(op1(g))))$ where $op_i(op_{i-1}(g))$ implies that the APIs for op_i are called after the APIs for op_{i-1} have been executed.

Example 2. Let us continue with our *Reference Scenario*. Let fingerprint authentication (*FnPr*), Lempel-Ziv compression (*LZ*) and AES encryption (*AES*) be the chosen operationalizations for the three high-level NFRs *Authentication*, *BandwidthEfficiency* and *Confidentiality*, respectively. If the developer assigns priorities 2, 4, and 7 to these NFRs, respectively, then the operationalization trace is represented as *AES*(*LZ*(*FnPr*(*CreateMessage*))). This implies that during message creation, APIs are first called for authenticating the sender using fingerprint scanning. This is followed by calling the API for compressing the message using Lempel-Ziv algorithm and, finally, the API for AES is called to encrypt the compressed message.

Every NFR is associated with six attributes - source of stimulus, stimulus, environment, artifact, response, and response measure. The response measure identifies the dimension along which the degree of satisfaction of that NFR is to be measured. For instance, the response measures for *Availability* could be fault detection time, fault repair time, availability percentage, and so forth. On the other hand, *Security* can have the dimensions number of attacks resisted, time required to detect an attack, percentage of system (or data) compromised, and so forth.[5]

The contribution of an NFR operationalization is a quantitative evaluation of that NFR in a specific dimension, identified by a corresponding response measure. This metric measures the positive impact of choosing a particular operationalization for a high-level NFR. The metric makes use of a contribution operator φ_P that estimates the degree of satisfaction achieved by that NFR operationalization with respect to the response measure P.

Definition 3.2 (Contribution Operator). Let '*op*' be an NFR operationalization that has to be satisfied for some goal $g \in \mathbb{G}$. The contribution operator φ_P is evaluated for the trace $op(g)$ and returns a value in the set of real numbers $\mathbb{R}$ with respect to some response measure P, that is,

$$\varphi_P : \tau \to \mathbb{R}$$

where τ denotes the end-user data set and $\varphi_P(t)$ is a quantitative measure of P for the data record $t \in \tau$.

TABLE 4 Data schema of the Hospital data set.

Data field	Data type	Data field	Data type
DRG Definition	VARCHAR2	Provider zip code	INTEGER
Provider Id	INTEGER	Hospital eegion descript.	TEXT
Provider name	TEXT	Total discharges	INTEGER
Provider dtreet address	VARCHAR2	Average covered charges	DECIMAL
Provider city	TEXT	Average Total payments	DECIMAL
Provider state	TEXT	Average Medicare payments	DECIMAL

Example 3. Let us assume a simpler version of the scenario discussed in Example 2 where *CreateMessage* needs to satisfy the high-level NFRs *Confidentiality* and *BandwidthEfficiency* only. Suppose the developer selects *LZ* for *BandwidthEfficiency* and *AES* for *Confidentiality*. The corresponding operationalization traces are *LZ*(*CreateMessage*) and *AES*(*CreateMessage*), respectively. Let the response measures associated with these traces be:

P_1 : "Latency introduced (in milliseconds) to apply *AES* on the user data"

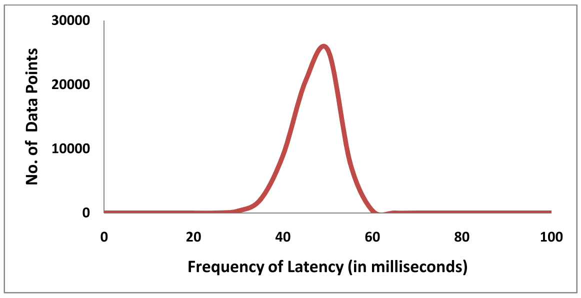
P_2 : "Compression efficiency (in %) achieved by *LZ* on the user data"

In order to estimate the values of φ_{P_1} and φ_{P_2}, we access the *Inpatient Prospective Payment System (IPPS) Provider Summary for the Top 100 Diagnosis-Related Groups (DRG) - FY2011.*[8] The data set has the schema shown in Table 4.

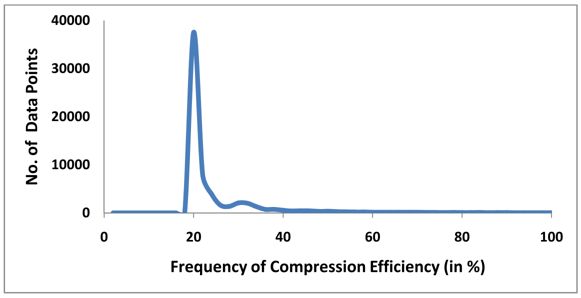
Each individual hospital record from the archive is accessed, and the *LZ* compression and *AES* encryption algorithms are applied separately to quantify the response measures P_1 and P_2. The latency introduced (for *AES*) and compression efficiency (for *LZ*) are recorded for each hospital data. The entire dataset contains more than 65,000 records. Figures 2A and 2B show the frequency distribution of the recorded latency and compression efficiency with respect to the entire dataset. We observe that the distributions take their peaks at **19–22 ms** for *AES* and **40% –50%** for *LZ*.

The contribution operator φ_P can be used to define the contribution function Φ_{op} associated with an NFR trace *op*(*g*). Let P_B (called the *Bounded Response Measure*) be defined from the response measure P as an inequality over a boundary value, B_V, that is, P_B is of the form $P(\leq, \geq, <, or >)B_V$. The contribution function Φ_{op} is an estimate of the mean degree of satisfaction of P_B by end-user data sets.

[8] Dataset: https://data.cms.gov/summary-statistics-on-use-and-payments/medicare-service-type-reports/cms-program-statistics-medicare-inpatient-hospital



(A) Latency of AES (in milliseconds)



(B) Compression Efficiency of LZ (in %)

FIGURE 2 Frequency distribution of data points while quantifying the response measures for *LZ* and *AES* individually. AES, Advanced Encryption Standard; LZ, Lempel-Ziv.

Definition 3.3 (Contribution Function). Let φ_P be the contribution operator associated with an NFR operationalization, P_B be the associated bounded response measure and τ be an end-user data set. The contribution function Φ_{op} is a two-argument function that can be defined as follows:

$$\boldsymbol{\Phi}_{op} : \varphi_P \times \tau \rightarrow [\![0, 1]\!]$$

that is, Φ_{op} takes the contribution operator and end-user dataset (or its subset) as arguments and calculates the fraction of records in that dataset that satisfy the bounded response measure P_B. Thus:

$$\Phi_{op}(\varphi_P, \tau) = \frac{|\{t| \text{ s.t. } \varphi_P(t) \vdash P_B, t \in \tau\}|}{|\tau|} \tag{1}$$

where $\varphi_P(t) \vdash P_B$ implies that $\varphi_P(t)$ satisfies the inequality of P_B.

Example 4. Continuing with the previous example, let the bounded response measures associated with P_1 and P_2 be defined as:

P_{B1} : "Latency introduced to apply *AES* $\leq B_{V1}$*ms*."

P_{B2} :"Compression efficiency achieved by *LZ* $> B_{V2}$%."

The peak values of the distributions (shown in Figure 2) can be used to estimate the values of B_{V1} and B_{V2} defined in P_{B1} and P_{B2}, respectively. Let us set B_{V1}=**20ms** and B_{V2}=**45%**. Since the boundary value estimates are derived from the frequency distribution peaks, it can be predicted that these values correspond to thresholds of 0.5, that is,

$Th_{AES} = Th_{LZ} = \mathbf{0.5}$. This implies that approximately 50% of the data points will satisfy the properties P_{B1} and P_{B2}, independently. In fact, we observe from our response measures that $\Phi_{AES} = 0.57$ and $\Phi_{LZ} = 0.512$ which makes the solutions acceptable, but only individually.

Note

An acceptance threshold Th_{op} has also been defined for each NFR operationalization and say that it is *acceptable* if $\Phi_{op} \geq Th_{op}$. Defining these thresholds are dependent on run-time environmental parameters. Correct estimation of these parameters is critical for the success of the framework and is non-trivial in nature.

We can also tune the thresholds by changing the boundary values B_{V1} and B_{V2}. Let us suppose that the desired threshold for *AES* is changed to **0.8** and for *LZ* to **0.6**. We observe from our response measure recordings that by setting B_{V1} to **29 ms** and B_{V2} to **43.8%**, we satisfy the thresholds and make the solutions acceptable, individually. The corresponding values of the contribution functions Φ_{AES} and Φ_{LZ} are 0.814 ($\geq$ 0.8) and 0.606 ($\geq$ 0.6), respectively.

We say that an NFR operationalization *op* is acceptable with respect to the bounded response measure P_B iff $\Phi_{op} \geq Th_{op}$, where Th_{op} represents some predefined threshold value for accepting the NFR trace. Conflicts occur when multiple NFR operationalizations interact and combine to provide a specific solution to the end-user. In order to identify conflicts, we need to first specify a formal mechanism for evaluating the contribution of combined operationalization traces.

Definition 3.4 (Combined Contribution Function). Let op_1, op_2 be two NFR operationalizations selected for two high-level NFRs N_1, N_2 and let π_1, π_2 be the associated priorities so that N_1 has higher priority than N_2, that is, $\pi_1 < \pi_2$. Let $\varphi_{P_1}, \varphi_{P_2}$ be the contribution operators for measuring op_1 and op_2 with respect to the response measures P_1 and P_2, respectively. Lastly, let P_{B1} and P_{B2} represent the bounded response measures derived from P_1 and P_2, respectively. The combined contribution function $\Phi_{op_1 \oplus op_2}$ can be defined as follows:

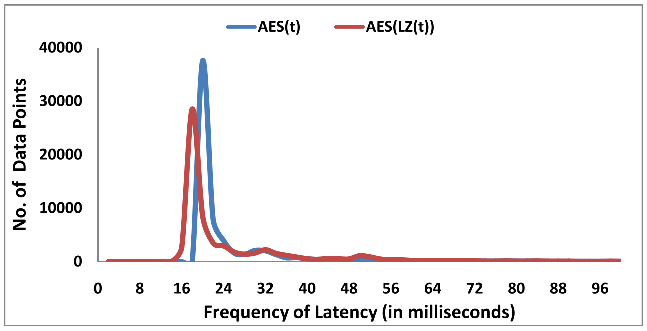
$$\Phi_{op_1 \oplus op_2} = \frac{|\{t | \varphi_{P_1}(t) \vdash P_{B1} \wedge \varphi_{P_2}(\varphi_{P_1}(t)) \vdash P_{B2}, t \in \tau\}|}{|\tau|} \quad (2)$$

Note

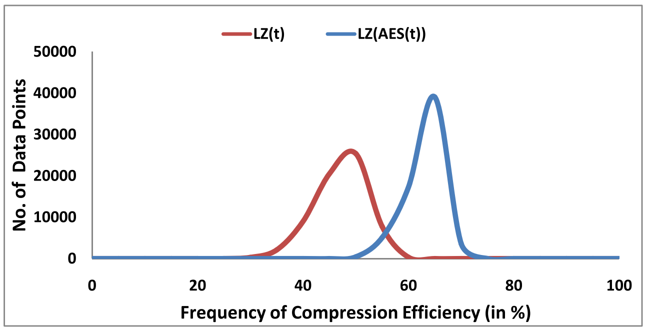
Let $Th_{op_1 \oplus op_2}$ denote the acceptance threshold for the combined trace. The derivation of the acceptance threshold for the combined trace is presented in Example 5. The operationalizations op_1 and op_2 are said to be in conflict if the combined contribution function does not satisfy the threshold, that is, $\Phi_{op_1 \oplus op_2} < Th_{op_1 \oplus op_2}$. Otherwise, the combined trace is acceptable and no NFR conflict exists. Also, if the associated priorities change and $\pi_2 > \pi_1$, then the combined contribution function is changed accordingly, as follows:

$$\Phi_{op_2 \oplus op_1} = \frac{|\{t | \varphi_{P_2}(t) \vdash P_{B2} \wedge \varphi_{P_1}(\varphi_{P_2}(t)) \vdash P_{B1}, t \in \tau\}|}{|\tau|}$$

This is because NFR priorities decide the order in which the NFR traces are generated.



(A) Latency of AES before and after applying LZ compression.



(B) Compression Efficiency of LZ before and after applying AES

FIGURE 3 Frequency distribution of data points while evaluating $\Phi_{LZ \oplus AES}$ and $\Phi_{AES \oplus LZ}$ in comparison to evaluating Φ_{AES} and Φ_{LZ} independently.

Example 5. Continuing from Example 4, the combined contribution of *AES* and *LZ* can be realized in either of two ways depending on the priorities assigned to NFRs *Confidentiality* and *BandwidthEfficiency*. The corresponding satisfaction criteria are as follows:

$$\Phi_{AES \oplus LZ} : AES(t) \vdash P_{B1} \wedge LZ(AES(t)) \vdash P_{B2}$$

$$\Phi_{LZ \oplus AES} : LZ(t) \vdash P_{B2} \wedge AES(LZ(t)) \vdash P_{B1}.$$

For the purposes of demonstration, both the combined contribution functions are evaluated. In reality, only one of them will be evaluated based on the assigned NFR priorities.

Using the same*Inpatient Prospective Payment System (IPPS)*[9] data set, both $\Phi_{AES \oplus LZ}$ and $\Phi_{LZ \oplus AES}$ have been evaluated. In Figure 3, the response measures of *AES* and *LZ* have been compared when applied independently (as shown in Figure 2) and when combined with the other. For instance, in Figure 3A, the blue line represents the frequency distribution of latency introduced when *AES* is applied on the raw data independently (from Figure 2A); whereas, the red line shows the frequency distribution of latency introduced when *AES* is applied after *LZ*. Similarly, in Figure 3B, the red line represents the frequency distribution of compression efficiency when *LZ* is applied on raw data

[9] Dataset: https://data.cms.gov/summary-statistics-on-use-and-payments/medicare-service-type-reports/cms-program-statistics-medicare-inpatient-hospital

TABLE 5 Variation of $\Phi_{LZ\oplus AES}$ with respect to different boundary values for P_{B1} and P_{B2}.

B_{V1}	B_{V2}	$\Phi_{LZ\oplus AES}$	Label ($Th_{LZ\oplus AES} = 0.5$)
19	45	0.254	*Conflict*
22	40	0.536	*Acceptable*
22	42.5	0.444	*Conflict*
25	40	0.583	*Acceptable*
25	45	0.346	*Conflict*

TABLE 6 Variation of $\Phi_{AES\oplus LZ}$ with respect to different boundary values for P_{B1} and P_{B2}.

B_{V1}	B_{V2}	$\Phi_{AES\oplus LZ}$	Label ($Th_{AES\oplus LZ} = 0.45$
20	65	0.0325	*Conflict*
22	60	0.458	*Acceptable*
22	62.5	0.19	*Conflict*
25	60	0.5	*Acceptable*
25	65	0.71	*Acceptable*

independently (from Figure 2B); the blue line represents the frequency distribution of compression efficiency when *LZ* is applied on data already encrypted by *AES*.

We conclude that the latency introduced by *AES* decreases slightly when applied on compressed data. This is evident from the frequency distributions shown in Figure 3A,B). On the other hand, the compression efficiency of *LZ* improves from 45% to 65% (approximately) when applied on encrypted data. These observations and peak values of the data distributions allow us to set the boundary values B_{V1}, B_{V2} as well as the threshold values $Th_{AES\oplus LZ}, Th_{LZ\oplus AES}$. Tables 5 and 6 show how the evaluation of the combined contribution operator changes depending on the values assigned to the boundary values of the bounded response measures P_{B1} and P_{B2}.

The thresholds assigned to the combined traces decide whether the traces are acceptable or in conflict. NFR conflicts are identified between *LZ* and *AES* whenever the combined contribution function fails to satisfy the threshold. Orthogonally, satisfying the threshold may require changing the boundary values so that more number of data records fulfil the satisfaction criteria. In case of conflicts, a system architect can try other combinations of operationalizations (of the high-level NFRs) and check if some combined trace satisfies the desired threshold with the given boundary values for the bounded response measures.

3.3 | The conflict calculator module

It is not sufficient to only identify the existence of a conflict. In order to minimize the NFR conflicts, we need to quantify the conflicts within combined traces. The *conflict calculator* module evaluates the relative conflict values based on some set of input data. The measure of conflict between any two NFR operationalizations depends on the deviation of the combined contribution function from the threshold value and also on the developer's prioritization of the corresponding high-level NFRs. Thus, the same choice of NFR operationalizations $\langle op_1, op_2\rangle$ can have different conflict values for different system developers based on their choice of priority values. With this notion, we proceed to define mean normalized priority and the conflict function.

Definition 3.5 (Mean Normalized Priority). Let $op_1, op_2, \dots, op_k$ denote the operationalization choices for k high-level NFRs and let π_i denote the priority value assigned to the i-th NFR. Each π_i lies in the range $[\![1, \overline{\pi}]\!]$ such that 1 represents the highest priority value and $\overline{\pi}$ represents the lowest priority value that can be assigned to NFRs. The mean normalized priority ω_k can be defined as -

$$\omega_k = \frac{\sum_{i=1}^{k} \frac{(\overline{\pi}-\pi_i)}{\overline{\pi}}}{k} = \frac{\sum_{i=1}^{k}(\overline{\pi}-\pi_i)}{k.\overline{\pi}} = \frac{k.\overline{\pi} - \sum_{i=1}^{k}(\pi_i)}{k.\overline{\pi}} = 1 - \frac{\sum_{i=1}^{k}(\pi_i)}{k.\overline{\pi}}.$$

Definition 3.6 (Conflict Function). Let op_1, op_2 be the selected NFR operationalizations of two high-level NFRs for which we have identified an NFR conflict, that is, $\Phi_{op_1\oplus op_2} < Th_{op_1\oplus op_2}$. The conflict function $\Delta_{op_1\oplus op_2}$ is directly proportional to the deviation ($\delta_{op_1\oplus op_2}$) from the threshold $Th_{op_1\oplus op_2}$, that is,

$$\Delta_{op_1\oplus op_2} = \omega_2.\delta_{op_1\oplus op_2}.$$

where ω_2 is the mean normalized priority (as defined in Definition 3.5) of the high-level NFRs corresponding to op_1 and op_2. The deviation $\delta_{op_1\oplus op_2}$ can be defined as -

$$\delta_{op_1\oplus op_2} = (Th_{op_1\oplus op_2} - \Phi_{op_1\oplus op_2})$$

The relative conflict values (or even the existence or absence of a conflict) depends on the assigned priorities in the combined NFR trace. This notion can be intuitively extended to more than two high-level NFRs.

Example 6. Continuing with Example 5, let the priorities assigned to *Confidentiality* and *BandwidthEfficiency* be 2 and 6, respectively, within the priority $[\![1, 10]\!]$. The threshold value $Th_{AES\oplus LZ}$ is 0.45, as shown in Table 6. Since *Confidentiality* has a higher priority than *BandwidthEfficiency*, the combined contribution trace $\Phi_{AES\oplus LZ}$ has been evaluated. Let us consider the conflict case when $B_{V1} = 22$ and $B_{V2} = 62.5$. The conflict function $\Delta_{AES\oplus LZ}$ is calculated as follows:

$$\Phi_{AES\oplus LZ} = 0.19$$

$$\delta_{AES\oplus LZ} = 0.45 - 0.19 = 0.26$$

$$\omega_2 = 1 - \frac{2+6}{2\times 10} = 1 - \frac{8}{20} = 0.6$$

$$\Delta_{AES \oplus LZ} = \omega_2 . \delta_{AES \oplus LZ} = 0.6 \times 0.26 = 0.156.$$

If some other developer assigns the priorities 1 and 3 to *BandwidthEfficiency* and *Confidentiality*, respectively, then $\Phi_{LZ \oplus AES}$ is evaluated. The threshold in this case is 0.5, as shown in Table 5. Let us consider the conflict case from Table 5 where $B_{V1} = 19$ and $B_{V2} = 45$. The conflict function $\Delta_{LZ \oplus AES}$ is calculated as follows:

$$\Phi_{LZ \oplus AES} = 0.254$$

$$\delta_{LZ \oplus AES} = 0.5 - 0.254 = 0.246$$

$$\omega_2 = 1 - \frac{(1+3)}{2 \times 10} = 1 - \frac{4}{20} = 1 - 0.2 = 0.8$$

$$\Delta_{LZ \oplus AES} = \omega_2 . \delta_{LZ \oplus AES} = 0.8 \times 0.246 = 0.197.$$

If another developer assigns priorities 4 and 7 to *BandwidthEfficiency* and *Confidentiality*, respectively, then for the above computation of the conflict function, only ω_2 will change. The conflict function is evaluated as follows:

$$\delta_{LZ \oplus AES} = 0.246$$

$$\omega_2 = 1 - \frac{(4+7)}{2 \times 10} = 1 - \frac{11}{20} = 1 - 0.55 = 0.45$$

$$\Delta_{LZ \oplus AES} = \omega_2 . \delta_{LZ \oplus AES} = 0.45 \times 0.246 = 0.11.$$

This is the notion of relative conflict values. A low priority NFR conflict (with priority levels 4 and 7) will have a lesser impact (= 0.11) on the overall conflict minimization problem as compared to a high priority NFR conflict (with priority levels 1 and 3) (=0.197). Orthogonally, we can state that the same NFR conflict can have different impacts depending on the priority values assigned by different developers.

3.4 | The conflict minimization module

Multiple NFR operationalizations, when applied on the same process or data resource (in some selected order), have a cumulative effect on that process or resource. This cumulative effect (which we call the Quality Function) plays a significant role in quantifying the overall quality of a solution.

Suppose we have a sequence of NFRs - like confidentiality, authentication, bandwidth efficiency, availability, performance, and so forth - which we would like to satisfy for some goal. The important question here is "*how do we check the existence of a conflict between all possible pairs of NFRs?*". In order to identify NFR conflicts between every possible combination of NFRs in the sequence, the notion of *ordered pairs* has been introduced. Each ordered pair of NFR operationalizations is checked in the Quality Function evaluation process.

Definition 3.7 (Ordered Pairs). Let $S = \{\langle op_1, \pi_1 \rangle, \langle op_2, \pi_2 \rangle, \ldots, \langle op_k, \pi_k \rangle\}$ be an NFR solution such that all op_i(s) are arranged in decreasing order of priority (i.e., $\forall i, \pi_i < \pi_{i+1}$). An ordered pair (within S) is denoted by the 2-tuple $\langle op_i, op_j \rangle$ such that $\pi_i < \pi_j$. Since all operationalizations are listed in decreasing order of priority, any operationalization op_i forms $(k - i)$ ordered pairs - one with each of the operationalizations that are to its right. These are $\langle op_i, op_{i+1} \rangle, \langle op_i, op_{i+2} \rangle, \ldots, \langle op_i, op_k \rangle$.

Example 7. Let us go back to Example 2. Based on the chosen operationalizations and assigned priorities for the three high-level NFRs, the NFR solution can be represented as $S = \{\langle FnPr, 2 \rangle, \langle LZ, 4 \rangle, \langle AES, 7 \rangle\}$. The set of all possible ordered pairs for S is given by $\{\langle FnPr, LZ \rangle, \langle FnPr, AES \rangle, \langle LZ, AES \rangle\}$.

We now proceed to define the quality function for an NFR solution based on all ordered pair of operationalizations that exist within the solution.

Definition 3.8 (Quality Function). Let $S = \{\langle op_1, \pi_1 \rangle, \langle op_2, \pi_2 \rangle, \ldots, \langle op_k, \pi_k \rangle\}$ be an NFR solution such that all op_i(s) are arranged in decreasing order of priority (i.e., $\forall i, \pi_i < \pi_{i+1}$). The quality function ρ_k for this solution tuple can be defined as follows:

$$\rho_k(op_1, op_2, \ldots, op_k) = \sigma_k . \omega_k, \tag{3}$$

$$\text{where } \sigma_k = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} (\Phi_{op_i \oplus op_j} - \Delta_{op_i \oplus op_j}); \tag{4}$$

$$\Phi_{op_i \oplus op_j} : \text{ as per Definition 3.4,}$$

$$\omega_k : \text{ as per Definition 3.5., and}$$

$$\Delta_{op_i \oplus op_j} : \text{ as per Definition 3.6.}$$

Thus, for evaluating the quality function for an NFR solution, we need the contribution function ($\Phi_{op_i \oplus op_j}$) and the conflict function ($\Delta_{op_i \oplus op_j}$) values (derived from $\delta_{op_i \oplus op_j}$ and ω_2 values) for each ordered pair $\langle op_i, op_j \rangle$ within the solution S. The term σ_k is called the *effective contribution* of the NFR solution S. In brief, the quality function in the context of the framework refers to a measure of the effectiveness of a set of NFRs operationalizations in minimizing conflicts among them. The quality function takes into account the run-time performance statistics such as contribution and conflict metrics of NFR operationalizations and the priorities of each NFRs to identify the best possible set of NFR operationalizations that minimizes the impact of conflicting NFRs.

Observe that negative values of ρ_k signify non-feasible solutions whereas positive values signify feasible solutions. The conflict minimization problem tries to find the best solution from the set of feasible solutions.

Example 8. Continuing from Example 7, let the contribution function Φ, threshold Th, deviation δ, mean normalized priority ω_2, and conflict

TABLE 7 Parameters for all ordered pairs.

$\langle op_1, op_2\rangle$	$\Phi(Th)$	δ	ω_2	Δ
$\langle FnPr, LZ\rangle$	0.2 (0.6)	0.4	0.7	0.28
$\langle FnPr, AES\rangle$	0.45 (0.7)	0.25	0.55	0.1375
$\langle LZ, AES\rangle$	0.68 (0.5)	−0.18	–	0

function Δ values for all ordered pair of operationalizations be as listed in Table 7.

The quality function ρ_3 is evaluated as $\sigma_3.\omega_3$ where σ_3 and ω_3 are evaluated as follows:

$$\begin{aligned}\sigma_3 &= [(\Phi_{FnPr\oplus LZ} - \Delta_{FnPr\oplus LZ})\\ &+ (\Phi_{FnPr\oplus AES} - \Delta_{FnPr\oplus AES})\\ &+ (\Phi_{LZ\oplus AES} - \Delta_{LZ\oplus AES})]\\ &= [(0.2 - 0.28) + (0.45 - 0.1375) + (0.68 - 0)]\\ &= 0.9125.\end{aligned}$$

$$\omega_3 = 1 - \frac{\pi_1+\pi_2+\pi_3}{3\times\overline{\pi}} = 1 - \frac{2+4+7}{3\times 10} = 1 - \frac{13}{30} = 0.567.$$

Thus, $\rho_3 = \sigma_3.\omega_3 = 0.9125 \times 0.567 = 0.5174$.

The priorities $\langle\pi_1, \pi_2, \ldots, \pi_k\rangle$ chosen by the developer are associated to the high-level NFRs $N_1, N_2, \ldots, N_k$ and not their specific operationalizations. Thus, with different combination of operationalizations, different values of the quality function can be obtained without changing their respective priorities. Let $\mathbb{O}_1, \mathbb{O}_2, \ldots, \mathbb{O}_k$ represent the operationalization sets for the k high-level NFRs, respectively. The NFR solution space $\mathbb{S}$ is defined by the cartesian product of the operationalization sets, that is, $\mathbb{S} = \mathbb{O}_1 \times \mathbb{O}_2 \times \cdots \times \mathbb{O}_k$. Each member of $\mathbb{S}$ represents a valid NFR solution whose quality function ρ_k can be evaluated as shown above. The conflict minimization problem is designed to find the NFR solution that has the highest value of the quality function ρ_k.

Definition 3.9 (Conflict Minimization Problem). Let $\rho_k^{(i)}$ denote the quality function value of the i-th NFR solution from the NFR solution space $\mathbb{S}$ and let Th_{qual} be the quality threshold set by the developer. The conflict minimization problem max(ρ_k) can be formulated as:

$$\text{Maximize} \sum_{i=1}^{|\mathbb{S}|} \rho_k^{(i)}.x_i, \tag{5}$$

$$\text{where } \rho_k^{(i)} = \sigma_k^{(i)}.\omega_k \text{ for the NFR solution } S_i \in \mathbb{S},$$

$$\text{subject to the constraints, } 0 < Th_{qual} \le \sigma_k^{(i)} \tag{6}$$

$$x_i \in \{0, 1\}, \forall i, \tag{7}$$

$$\text{and} \sum_{i=1}^{|\mathbb{S}|} x_i = 1. \tag{8}$$

TABLE 8 Contribution and conflict matrices.

(a) Contribution function values				
$\Phi_{i\oplus j}$	**AES**	**DES**	**3DES**	**Blowfish (BF)**
PPM	0.268	0.635	0.183	0.58
LZ	0.213	0.26	0.65	0.72
CM	0.447	0.117	0.29	0.63
(b) Conflict function values				
$\Delta_{i\oplus j}$	**AES**	**DES**	**3DES**	**Blowfish(BF)**
PPM	0.139	0	0.19	0
LZ	0.172	0.144	0	0
CM	0.032	0.23	0.126	0

Abbreviations: AES, Advanced Encryption Standard; CM, Context Mining; DES, Data Encryption Standard; LZ, Lempel-Ziv; PPM, Prediction by Partial Match.

TABLE 9 Evaluation of quality functions ρ_2.

$\langle S_i, S_j\rangle$	Φ	Δ	$\sigma_2 = \Phi - \Delta$	$\rho_2(S_i, S_j) = \sigma_2.\omega_2$
$\langle PPM, AES\rangle$	0.268	0.139	0.129	0.1032 ($\sigma_2 < Th_{qual}$)
$\langle PPM, DES\rangle$	0.635	0	0.635	0.508
$\langle PPM, 3DES\rangle$	0.183	0.19	−0.007	−0.0056 (not feasible)
$\langle PPM, BF\rangle$	0.58	0	0.58	0.464
$\langle LZ, AES\rangle$	0.213	0.172	0.041	0.0328 ($\sigma_2 < Th_{qual}$)
$\langle LZ, DES\rangle$	0.26	0.144	0.116	0.0928 ($\sigma_2 < Th_{qual}$)
$\langle LZ, 3DES\rangle$	0.65	0	0.65	0.52
$\langle LZ, BF\rangle$	0.72	0	0.72	**0.576 (max ρ_2)**
$\langle CM, AES\rangle$	0.447	0.032	0.415	0.332
$\langle CM, DES\rangle$	0.117	0.23	−0.113	−0.0904 (not feasible)
$\langle CM, 3DES\rangle$	0.29	0.126	0.164	0.1312 ($\sigma_2 < Th_{qual}$)
$\langle CM, BF\rangle$	0.63	0	0.63	0.504

Abbreviations: AES, Advanced Encryption Standard; BF, Blowfish; CM, Context Mining; DES, Data Encryption Standard; LZ, Lempel-Ziv; PPM, Prediction by Partial Match.

Example 9. We go back to Example 6 where we had only two high-level NFRs - *BandwidthEfficiency* and *Confidentiality* - with assigned priorities 1 and 3, respectively, within the priority range $[\![1, 10]\!]$. Using definitions 3.4 and 3.6, let the combined contribution function and conflict function values for all possible combinational traces be as shown in Table 8, respectively. $\Phi_{i\oplus j}$ and $\Delta_{i\oplus j}$ signify that the row operationalizations (for *BandwidthEfficiency*) are applied before the column operationalizations (for *Confidentiality*).

The mean normalized priority ($\omega_2 = 0.8$) remains fixed for all combined traces. Using these values, we now proceed to evaluate the quality function ρ_2 for all possible NFR solutions and identify the conflict-minimal solution. Table 9 shows the computations for each of the 12 possible solutions (3 operationalizations of *BandwidthEfficiency* combined with 4 operationalizations of *Confidentiality*). The invalid solutions, which do not satisfy the quality threshold of the conflict minimization problem ($Th_{qual} = 0.4$), are mentioned in the column for ρ_2.

We see that the conflict-minimal solution is given by the combination trace $\langle LZ, BF\rangle$ with a quality function value of $\rho_2 = 0.576$. The NFR Conflict Minimizer performs the following checks:

1. If the developer selects a *non-feasible* solution (like $\langle PPM, 3DES\rangle$) having quality function value $\rho_2 < 0$, then the system aborts the app generation process and asks the developer to choose among alternate operationalizations.
2. If the developer selects a *feasible bad quality* solution (like $\langle CM, 3DES\rangle$) having $\rho_2 > 0$ but $\sigma_2 < Th_{qual}$, then the system *warns* the developer that the quality of the chosen solution is unacceptable. The developer must choose among alternate operationalizations.
3. If the developer selects a *feasible sub-minimal* solution (like $\langle LZ, 3DES\rangle$) having $\rho_2 > 0, \sigma_2 > Th_{qual}$ but $\rho_2 \neq max(\rho_2)$, then the system *informs* the developer of the existence of a conflict-minimal solution. The developer may choose to ignore the message or continue with the conflict-minimal solution.
4. If the user selects the *conflict-minimal* solution ($\langle LZ, BF\rangle$) such that $\rho_2 = max(\rho_2)$, then the system *informs* the developer that this is the best possible solution that exists.

Note

It is not necessary that every combination trace that has a non-zero conflict function value Δ, will be either *infeasible* or *bad quality*. In Table 9, the solution $\langle CM, AES\rangle$ has $\Delta = 0.032$ but neither $\rho_2 < 0$ nor $\sigma_2 < Th_{qual}$.

4 | THE NFR CONFLICT MINIMIZER ALGORITHMS

In the previous Section, the three modules of the NFR Conflict Minimization Framework have been introduced that are responsible for identifying the conflict-minimal NFR solution. Here we see how the *Conflict Minimization Module* is implemented as an algorithm. We assume that the contribution and conflict matrices have been populated by the *Contribution Calculator* and *Conflict Calculator* modules, respectively.

The NFR Conflict Minimizer uses two algorithms namely *Conflict-Minimized Solution* (Algorithm 1**Input**: The algorithm takes the following inputs:i The solution space $\mathbb{S}$ derived from the operationalization sets $\mathbb{O}_i$ of the k NFRs.ii k NFR priorities Π as selected by the developer. The range of the priorities is $[1, \overline{\pi}]$**Output**: Algorithm will return:i The conflict-minimal solution set $S_m \in \mathbb{S}$ that has the maximum quality function value ρ.) which calls another algorithm called *Effective Contribution* (Algorithm 2**Input**: The algorithm takes the following inputs:i $S = \langle s_1, s_2, \dots, s_k\rangle$: a solution tuple where $s_i \in \mathbb{O}_i$ and $\pi_i < \pi_{i+1}, \forall i$.ii $\mathbb{V}$: the set of contribution function values.iii $\mathbb{C}$: the set of conflict function values.iv Th_{qual}: quality threshold for finding the conflict-minimal solution.**Output**: Algorithm will return:i EC: the effective contribution of the solution tuple S if it is feasible and satisfies the quality threshold Th_{qual}.ii 0: otherwise.) to calculate the conflict-minimal solution for an NFR solution set. Algorithm 1 enumerates the entire solution set $\mathbb{S}$ consisting of all possible combination of NFR operationalizations. Algorithm 2 is invoked for every NFR solution, and the effective contribution of the solution is evaluated. We assume that the contribution function and conflict function values have been evaluated by the respective calculator modules. The user may interact with Algorithm 1 by changing the assigned NFR priorities, whenever the solution is not

ALGORITHM 1 Conflict-Minimal Solution

```
1:  procedure ConflictMinimalSolution$\mathbb{S}, \Pi$
2:      $\rho \leftarrow 0, Pr \leftarrow 0$
3:      for each $\pi_i \in \Pi$ do
4:          $Pr \leftarrow Pr + \pi_i$
5:      end for
6:      $W \leftarrow 1 - [Pr/(\overline{\pi} \times k)]$
7:      for each $S_i = \langle s_0, s_1, \dots, s_k\rangle \in \mathbb{S}$ do
8:          $E \leftarrow EffectiveContribution(S_i)$
9:          if $E = 0$ then
10:             continue
11:         else
12:             $R \leftarrow E \times W$
13:         end if
14:         if $R > \rho$ then
15:             $\rho \leftarrow R$
16:             $m \leftarrow i$
17:         end if
18:     end for
19:
20:     return$S_m, \rho$
21: end procedure
```

ALGORITHM 2 Effective Contribution

```
1:  procedure EffectiveContribution$S$
2:      $EC \leftarrow 0$
3:      for i=1 to $k$-1 do
4:          for j=$i$+1 to k do
5:              $EC \leftarrow EC + (\mathbb{V}[s_i][s_j] - \mathbb{C}[s_i][s_j])$
6:          end for
7:      end for
8:      if $EC \geq Th_{qual}$ then
9:          return$EC$
10:     else
11:         return0
12:     end if
13: end procedure
```

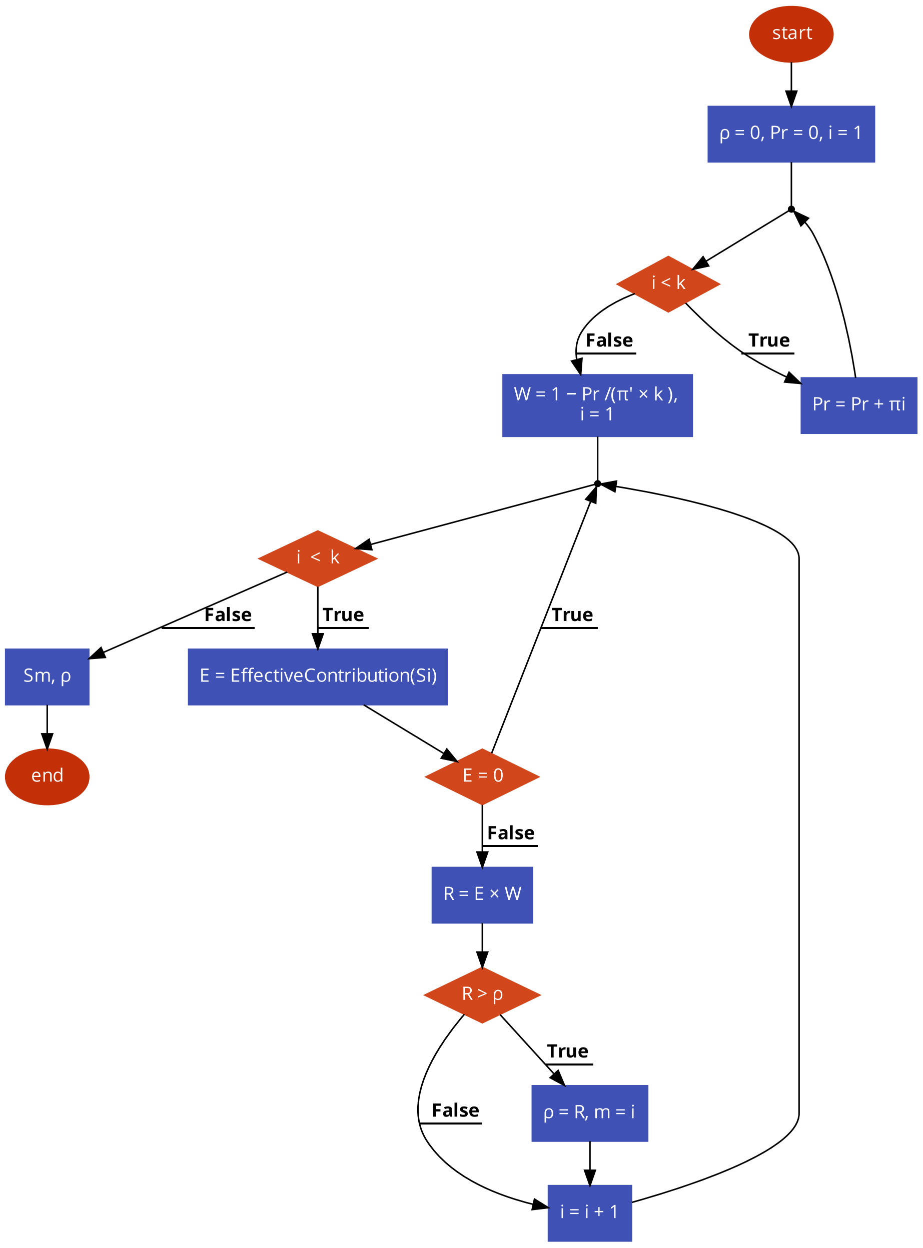


FIGURE 4 Flowchart of conflict-minimal solution.

considered satisfactory. A flowchart of the entire process is shown in Figure 4 for better understanding.

4.1 | Managing the solution space

An obvious problem associated with the proposed conflict-minimal solution is the *pairwise-evaluation* of NFR conflicts that results in an exponential growth in the size of the NFR solution space $\mathbb{S}$. In this section, a weighted scheme over NFR priorities and conflicts has been proposed that results in effectively managing the solution space. However, before going into the details of this weighted scheme, let us revisit the data-centric NFR Conflict Minimization Architecture shown in Figure 1. We can see that the DevOps app development life cycle has a feedback component which computes the *Conflict-minimal NFR Solution* provided by the NFR Conflict Minimizer. This feedback is provided periodically based on the usage statistics that have been obtained (or generated) since the last periodic update. The conflict-minimal solution can lead to a redesign of the existing solution which can be sent to all users of the app as a version update. Thus, as more data are generated by the app, the conflict-minimal solution may keep changing.

4.1.1 | The basis

Let us consider that $S(t)$ represents the solution in the current iteration and we want to derive the conflict-minimal solution for the next iteration - denoted by $S(t + 1)$. The evaluation process is parameterized with

TABLE 10 Conflict function values.

$\Delta_{i\oplus j}$	AES	DES	3DES	Blowfish(BF)
PPM	0.139	0	0.19	0
LZ	0.172	0.144	0	0
CM	0.032	0.23	0.126	0

Abbreviations: AES, Advanced Encryption Standard; CM, Context Mixing; DES, Data Encryption Standard; LZ, Lempel-Ziv; PPM, Prediction by Partial Match.

the static NFR priorities (denoted by Π) and the dynamic NFR conflicts computed in the previous iteration (denoted by $C(t)$), that is,

$$S(t+1) = f(\Pi, C(t))$$

With this periodic updation process, a proposal for the management of our solution space is as follows. Only for the first time when the app is being developed (i.e., for $t = 0$), NFR conflicts have to be evaluated for all pairs of operationalizations. For example, consider Table 8 (part b)(repeated here for illustration purposes) where NFR conflicts between all possible pairs of *Confidentiality* and *BandwidthEfficiency* have been computed.

However, for every successive iteration (or periodic update), the solution space can be reduced by using our proposed weighted scheme. The scheme is based on weights w_1 and w_2 assigned to the two parameters NFR Priority and NFR Conflict, respectively, by the app maintenance team. Both w_1 and w_2 are normalized in the range [0,1]. Determination of these weights is an overhead for the administrator and needs to be defined accurately for the success of the framework. With these settings, we now elaborate on how the weighted scheme works (Table 10).

4.1.2 | The weighted scheme

Let us assume that the maintenance team assigns $w_1 = 0.85$ and $w_2 = 0.4$. Now $w_2 = 0.4$ implies that we will keep 40% of the least conflicting solutions pairs. If we look at Table 8 (part b), we see that out of the 12 possible solution pairs, five are without conflicts. So we look at the seven conflicting solution pairs. Since $0.4 \times 7 = 2.8$ (which rounds off to 3), we keep the three least conflicting solution pairs and discard the four pairs with higher conflict values. The underlying presumption here is that conflicting solutions at the previous iteration (at $t = i$) may become conflict-free in the next iteration (at $t = i + 1$) since conflict values are calculated from usage statistics. The solution space reduces as shown below (Table 11):

Next, NFR Priority on the reduced solution space has been considered. Let the normalized priorities of *Confidentiality* and *BandwidthEfficiency* be 0.8 and 0.6, respectively. *Confidentiality* has four NFR operationalizations and, thus, the normalized priority is multiplied with the number of operationalizations available, that is, $0.8 \times 4 = 3.2$, which rounds off to 3. This is further multiplied by w_1 as $0.85 \times 3 = 2.55$ (which also rounds off to 3). So we keep the three best solutions AES, 3DES, BF. Similarly, for *BandwidthEfficiency*, $0.6 \times 3 = 1.8$ (rounds off to 2) and $0.85 \times 2 = 1.7$ (which rounds off to 2). So we keep the two best solutions PPM, LZ. Thus, we further reduce the solution space by keeping only those pairs whose members have been selected in the above mentioned process. The reduced solution space is shown in the Table 12.

TABLE 11 Conflicting solution pairs after removing higher conflict values.

$\Delta_{i\oplus j}$	AES	DES	3DES	Blowfish(BF)
PPM	0.139	0	■	0
LZ	■	■	0	0
CM	0.032	■	0.126	0

Abbreviations: AES, Advanced Encryption Standard; CM, Context Mixing; DES, Data Encryption Standard; LZ, Lempel-Ziv; PPM, Prediction by Partial Match.

TABLE 12 Reduced solution space.

$\Delta_{i\oplus j}$	AES	DES	3DES	Blowfish(BF)
PPM	0.139	■	■	0
LZ	■	■	0	0
CM	■	■	■	■

Abbreviations: AES, Advanced Encryption Standard; CM, Context Mixing; DES, Data Encryption Standard; LZ, Lempel-Ziv; PPM, Prediction by Partial Match.

The app management team can decide to change the weights in the upcoming iteration of the periodic update process. Depending on the weights, the current solution space $S(t)$ can either increase or decrease in size in the next iteration. However, this growth or reduction in the solution space happens in a controlled environment.

Note

Adding a new NFR (adds a new dimension) or adding a new operationalization (to an existing NFR) for the next iteration becomes quite easy. Pairwise computations need to be performed for only those pairs which are active in the current iteration only.

4.1.3 | Special cases: $w_1 = 0$ or $w_2 = 0$

If $w_1 = 0$ or $w_2 = 0$, then in both cases we do not go for the multiplication process mentioned in the weighted scheme. This is because a weight of zero has different implications. $w_1 = 0$ signifies that we accept the solution space reduction achieved by normalized NFR Priorities and we do not want to reduce it further. In this case, multiplication would result in reducing the solution space of all dimensions to *zero* - which does not have any consequences. On the other hand, $w_2 = 0$ signifies that we do not discard any conflicting solution in the next iteration. Multiplication would also result in *zero* which would imply that there would be no reduction in solution space based on NFR Conflicts. However, if one of the weights is *non-zero*, then we proceed to modify the solution space (using the corresponding parameter whose associated weight is *non-zero*) as described in the weighted scheme.

5 | EXPERIMENTS

In this section, the evaluation of our proposed NFR Conflict Minimization and Weighted Solution Space Pruning algorithms is conducted on a more complex use-case scenario consisting of five NFRs. The selected NFRs are *Confidentiality*, *BandwidthEfficiency*, *Authentication*, *Usability*, and *Performance*. The five NFRs have been assigned the priority values one (for *BandwidthEfficiency*), three (for *Confidentiality*), five (for *Authentication*), six (for *Usability*), and seven (for *Performance*), respectively. For each of these NFRs, we have the following sets of operationalizations:

i. *Confidentiality*:{*AES*, *DES*, *3DES*, *Blowfish*}
ii. *BandwidthEfficiency*:{*PPM*, *LZ*, *CM*}
iii. *Authentication*:{*Password*, *Pattern*, *FingerPrint*, *FaceRec*}
iv. *Usability*:{Touch Based GUI, Conversational UI, Menu-Driven Interface}
v. *Performance*:{Progressive Web App Implementation, Native Implementation, Hybrid Implementation}

Based on these priority values, our NFR Conflict Minimization algorithm evaluates the pairwise contribution and conflict values for each *ordered pair* of NFRs.

The size of the solution space can be calculated by taking the Cartesian product of the individual operationalization sets. For the above example, the size of solution space comes to $4 \times 3 \times 4 \times 3 \times 3 = 432$. The solution space can be visualized in Figure 5A. The figure is to be interpreted as follows:

i. There are five parallel planes identified using red dashed lines. Each plane represents one NFR.
ii. The planes are positioned from top to bottom in decreasing order of NFR priority.

    For our example, the topmost plane represents *BandwidthEfficiency* (priority 1), the second plane from the top represents *Confidentiality* (priority 3), followed by *Authentication* (priority 5) and *Usability* (priority 6). The bottom-most level represents *Performance* (priority 7).
iii. The black dots on each plane corresponds to the operationalizations available for that NFR.

    For our example, there are three dots on the topmost plane for *BandwidthEfficiency*, four dots on the next plane for *Confidentiality*, and so on.
iv. Edges are set up in a top down manner, that is, a vertex on any plane can connect with any vertex of any plane that comes below it.

The above guidelines apply to all the sub-figures in Figure 5.

5.1 | Reduction in solution space

The original solution space, depicted in Figure 5A, is pruned using the threshold value $Th_{qual} = 0.4$ and weights of $w_1 = 0.8$ for NFR Priority and $w_2 = 0.4$ for NFR Conflict, respectively. Based on the solution space management mechanism described above, the reduced solution space is obtained as shown in Figure 5B. Some of the interesting things to observe in this figure are as follows:

i. Pruning based on NFR conflict is realised by deleting edges from the solution space.
ii. Pruning based on NFR priority is realised by deleting vertices from the solution space.
iii. The number of vertices in the topmost level (Level 1) remains the same.
iv. The number of vertices in Levels 2 and 4 get reduced by one.
v. The number of vertices in Level 3 and 5 get reduced by two.
vi. Unlike the original solution space, the reduced solution space is not a complete graph due to edge deletion by NFR conflict.
vii. Not all edges appearing in the reduced solution space form a part of some valid solution. An edge $\langle v_1, v_2 \rangle$ becomes a part of some valid solution, only if there exists three more vertices v_3, v_4, and v_5 such that all the edges $\langle v_1, v_3 \rangle$, $\langle v_1, v_4 \rangle$, $\langle v_1, v_5 \rangle$, $\langle v_2, v_3 \rangle$, $\langle v_2, v_4 \rangle$, $\langle v_2, v_5 \rangle$, $\langle v_3, v_4 \rangle$, $\langle v_3, v_5 \rangle$, and $\langle v_4, v_5 \rangle$ are also present.

Figures 5C–F show the only four solutions that are possible in the reduced solution space of Figure 5B.

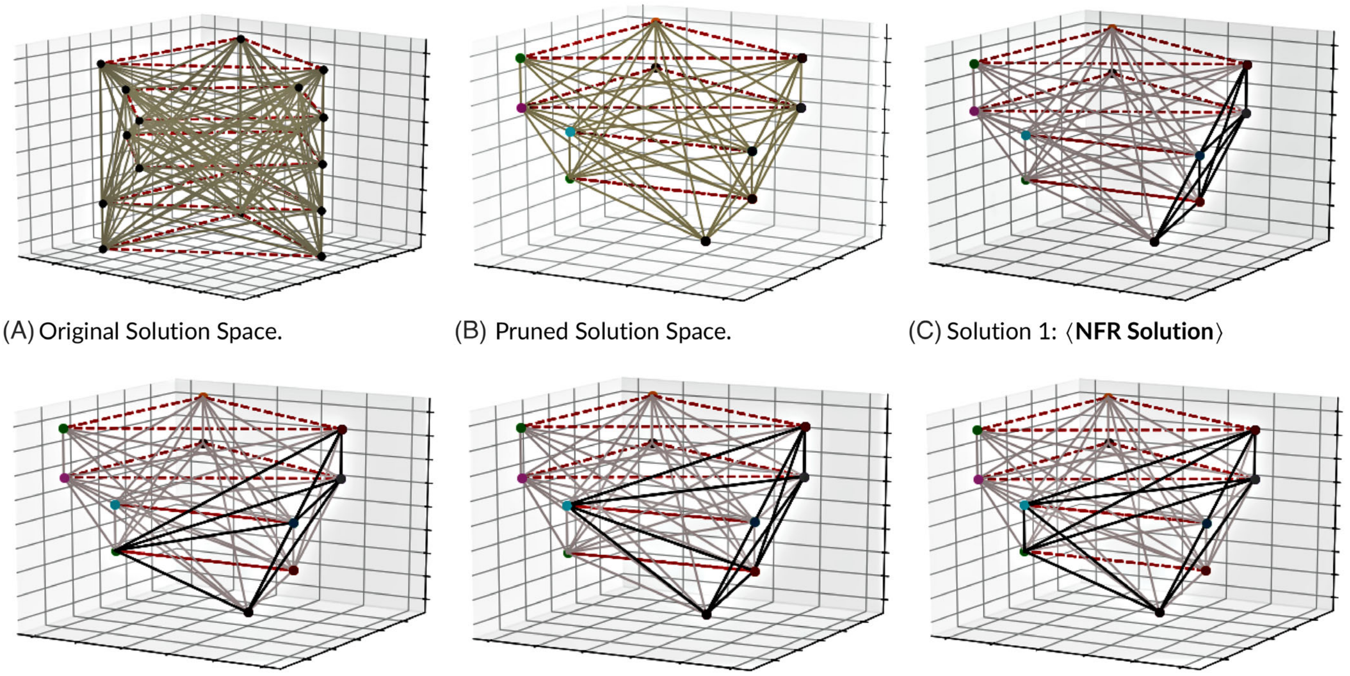
5.2 | Annexure: The NFR conflict minimization tool

Please refer to the annexure submitted with the paper. It documents a step-by-step process of using our tool which takes all the NFR related information from the user, and based on the evaluated contribution and conflict functions, the tool evaluates the conflict-minimal choice of NFR operationalizations. The tool also shows how the solutions space may be reduced by setting the weights (w_1 and w_2) and threshold (Th_{qual}) values. Finally, the tool allows the user to visualize the valid solutions in the reduced solution space.

All tool sources for reproduction purposes are made available in a GitHub project.[10] We also have a video tutorial of the case study.

Table 13 shows the experimental evaluation of the system and explain how the system selects the best fit solution. For instance, at a given time instance t_i, we have dataset D_i. In Table 13, column 1 shows four increments of the dataset for four successive time instances. In this experiment, compression efficiency is measured for each of the operationalizations on the current dataset and compression efficiency that satisfies the given percentile criteria is also considered.

[10] https://github.com/GRL2APK/OptimizationTool



(A) Original Solution Space. (B) Pruned Solution Space. (C) Solution 1: ⟨**NFR Solution**⟩

(D) Solution 2: ⟨**NFR Solution**⟩ (E) Solution 3: ⟨**NFR Solution**⟩ (F) Solution 4: ⟨**NFR Solution**⟩

FIGURE 5 In the above figures, (A) Original solution space; (B) Pruned solution space with $Th_{qual} = 0.4$, $w_1 = 0.8$, and $w_2 = 0.4$; (C)–(F) Solutions within the reduced solution space.

TABLE 13 Experimental evaluation with respect to compression efficiency of bandwidth efficiency non-functional requirement operationalizations.

Dataset size	Threshold	Percentile	Compression efficiency(LZ)	Compression efficiency(ZIP)	Compression efficiency(PPM)	Distance (LZ, Threshold)	Distance (ZIP, Threshold)	Distance (PPM, Threshold)
10K	19	0.6	51.69082126	26.51933702	21.67487685	32.69082126	7.519337017	2.674876847
20K	23	0.5	50.98039216	25.99118943	20.83333333	27.98039216	2.991189427	−2.166666667
30K	47	0.7	52.83018868	27.31092437	22.74881517	5.830188679	−19.68907563	−24.25118483
40K	45	0.6	51.69082126	26.60550459	21.78217822	6.690821256	−18.39449541	−23.21782178

Abbreviations: LZ, Lempel-Ziv; PPM, Prediction by Partial Match.

Table 13 presents compression efficiencies for different operationalizations. For example, at t_1, we got compression efficiencies 51.69%, 26.51% and 21.67% for LZ, ZIP and PPM, respectively.

Selection of the best fit solution is preceded by measuring the distance between the obtained efficiencies and the desired threshold. The distance calculation for each operationalization with respect to the given threshold is accomplished by measuring simple difference. The distance measures computed for our experiment are shown in columns 7–9 of Table 13. For instance, at timestamp t_1, the distance measure between the threshold and compression efficiency is 32.69 for LZ, 7.51 for ZIP and 2.67 for PPM.

A procedure that identifies the best fit operationalization based on minimum positive distance is also defined. For example, for the first iteration, PPM has the minimum distance. Hence, for this iteration, our (Dynamic Bayesian Network) DBN-based adaptive system selects PPM as the most suitable operationalization. The system identifies the best fit by finding the minimum among all positive distances, while ignoring the negative distances. This is because negative distances imply that the particular NFR operationalization cannot meet the desired threshold. However, if it is observed that every operationalization has negative distance, then the system finds the maximum of all negatives to suggest which operationalization comes closest to satisfying the threshold. In Table 14, we present four iterations and illustrate how our DBN-based system computes compression efficiencies at every iteration and adapts to the best fit operationalization in the next time slice.

We also provide a GitHub[11] link containing all the datasets and experiment data for our study. A user manual is also provided in

[11] https://github.com/Souvick-CU/PDANFR

TABLE 14 Experimental Eevaluation with respect to latency of confidentiality non-functional requirement operationalizations.

Dataset	Threshold	Percentile	Latency (AES)	Latency (DES)	Latency (BF)	Distance (AES, Threshold)	Distance (DES, Threshold)	Distance (BF, Threshold)
10K	67	0.6	81	39	38	−14	🟩 28	29
20K	85	0.5	80	35	28	🟩 5	50	57
30K	42	0.7	69	41	30	−27	🟩 1	12
40K	26	0.6	67	30	18	−41	−4	🟩 8

Abbreviations: AES, Advanced Encryption Standard; BF, Blowfish; DES, Data Encryption Standard.

the repository in order to replicate the experiments and observe the behavior of the system.

6 | RELATED WORK

Managing conflicts among NFRs involves mainly three activities- conflict identification, conflict analysis and conflict resolution.[15] In this section a brief overview of related works have been presented in order to address these three activities. Paja et al.[8] propose STS-ml modeling language which is able to detect possible conflicts between security requirements and actor's business policies. This model supports to represent transmission of Role, Permissions, Documents, Information. In 2012, Affleck, Krisna[16] extend the NFR Framework which supports quantitative analysis of the framework. Leaf-softgoals as well as operationalizations are also assigned some weights and the contributions towards their parents are also evaluated. Hu et al.[7] integrate NFR framework with their work that uses SIG to represent interdependencies among softgoals. Conflict detection and automated reasoning have been achieved by using SWRL (Semantic Web based Rule Language)[17] conflict detection guidelines and NFRs correlation in semantic web.[18] A tool which is able to detect conflict between NFRs based on semantic relationships in the ontologies has been presented in literature[13]. J. Y. Bang et al.[19] present collaborative-design framework (FLAME) that efficiently and continuously detects design conflicts. The framework can classify design conflicts and identify risks behind the design conflicts. In literature[20] authors present a catalog which contains conflicts among NFRs of UbiComp and IoT applications. Another perspective of the paper is construction of the catalog at design time so that the behavior of the operationalizing softgoals can be simulated using some executable models. A framework has been introduced that accepts a pair wise NFRs conflict characterization and manages NFR metrics and measures as parameters to provide quantitative reasoning about NFR conflict is presented in literature[21]. In another work,[11] a model has been proposed in order to identify conflicts and dependencies through scenarios and use-cases. A conflicting NFRs tradeoff framework (CNTF) and the corresponding method are proposed by X. Zhang et al.[22] They have used fuzzy set theory to assess NFRs provided by stakeholders and plots the degree of satisfaction of each conflicting NFR strategies in tradeoff curves. In literature[23], the authors categorize relative conflicts between NFRs in three different categories: *absolute conflict*, *relative conflict* and *never in conflict*. The approach maintain different stages to develop a catalogue. Sadana et al.[10] introduce a framework illustrates some typical interrelationship between NFRs to analyze conflicts among NFRs. In another work,[24] NFRs are fragmented in the form of multi-entity Bayesian network fragments. As a consequence, when changes arise, the system produces a Bayesian network to measure the impact of the system's conditions on the NFRs.

Egyed et al.[25] introduce an approach that eliminates only false conflicts and cooperation without involving any interdependencies between requirements. Finally they generate a shorter weighted list of possible conflicts and cooperation. A framework has been presented in literature[14] is able to identify conflicts between security and privacy and provides necessary methods to resolve and mitigate those conflicts. In literature[26], the authors have presented a systematic approach to allow software architects to support reasoning about uncertainty about the impact of alternatives on stakeholders' goals. They calculate the consequences of uncertainty through Monte-Carlo simulation and shortlist candidate architectures based on expected costs, benefits and risks. Another research[27] extends the QUAMOCO[28] quality model to specify feature dependent NFRs. The authors provide tool support for the operationalization of this quality model which is useful for DevOps contexts. The research proposed by literature[29], investigates correlations between Invisibility and user interaction-related NFRs in UbiComp and IoT systems. Through research methods like interviews and questionnaires, it systematizes the definition of 110 correlations and provides guidance for software engineers in selecting strategies to fulfill Invisibility and related NFRs.[30] proposed a machine learning-based model for detecting conflicts between NFRs in a Software Requirement Specification (SRS) document. By utilizing text preprocessing, vectorization, and classification techniques, the model achieves 84.74% accuracy. This approach offers a potential solution to address requirement conflicts, which are a common cause of software failure, ultimately aiding in the design, testing, and maintenance of software systems. Another work[31] proposes an actor-based goal convergence method that automatically detects and resolves conflicts between goal models from different domains, forming a unified goal model. The paper also presents a prototype system that visually supports the convergence of goal models and identifies grammar mistakes and conflicts, including non-functional, operational, and resource conflicts.

TABLE 15 Comparison of our framework with existing solutions.

Research work	Conflict detection	Quantitative analysis	Conflict-minimal solution	Conflict mitigation
7	✓	×	×	×
8	✓	×	×	×
13	✓	×	×	×
14	✓	×	×	✓
16	✓	✓	×	×
20	✓	×	×	×
21	✓	✓	×	✓
11	✓	×	×	×
22	✓	✓	×	✓
23	✓	×	×	×
10	✓	✓	×	×
24	✓	×	×	×
25	✓	×	×	×
27	✓	×	×	×
29	✓	✓	×	✓
30	✓	×	×	×
31	✓	✓	×	✓
Our approach	✓	✓	✓	✓

Majority of the existing works present conflict detection frameworks that uses potential conflict catalogs or conflict rule bases.[32–34] However, the notion of NFR conflicts is very dynamic and it does not seem logical to hard code them in the form of catalogs or rule bases. Even for the same software application, different end-users may prioritize the associated NFRs differently. There has been work in the literature where researchers have tried to quantify the impact or contribution of NFRs as well as propagate them throughout the SIGs.[7] However, these quantifications are based on human values specified by the requirement engineers. The large volumes of data generated by the mass deployment of any IoT (Internet-of-Things) application are used to quantify the contribution and relative conflicts. Instead of using static rules and catalogs, our proposed framework mathematically evaluates relative conflicts based on end-user priorities. The space of possible solutions can also be dynamically pruned based on these priorities. The proposed framework is capable of deriving the conflict-minimal solution within this reduced solution space (Table 15).

7 | CONCLUSION

In this work, an NFR management framework has been documented that identifies the best possible combination of NFR operationalizations while minimizing the impact of conflicts between the participating NFRs. The proposed framework works with only *run-time* NFRs whose response measures can be collected from usage statistics at run time. The framework is dynamic in the sense that it keeps updating the conflict-minimal NFR Solution with respect to changes in the NFR solution space - the set of participating NFRs, their possible operationalizations, combined contributions, pair-wise conflicts, and so forth - over time. It is worth mentioning that the framework does not have any significant overhead on the targeted system. The only requirement of the framework is the periodic analysis of App usage statistics.

As part of our future work, we plan to extend our work for *design-time* NFRs as well. This may not be trivial as *design-time* NFRs decide the architecture of the software being developed and combining their measurements with run-time characteristics is quite challenging. We are also working towards getting a feedback of our tool when used in real-world projects.

ACKNOWLEDGMENTS

Work partially supported by iNESTInterconnected NordEst Innovation Ecosystem, funded by PNRR (Mission 4.2, Investment 1.5), NextGeneration EU-Project ID: ECS 00000043, and by SPIN2021 Ressa_Rob, funded by Ca'Foscari University.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ORCID

Souvick Das https://orcid.org/0000-0002-3314-2537
Nabendu Chaki https://orcid.org/0000-0003-3242-680X

REFERENCES

1. Rother M. *Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results*. McGraw-Hill Professional Publishing; 2009.
2. Barth W. *Nagios: System and Network Monitoring*. No Starch Press; 2008.
3. Mairiza D, Zowghi D, Gervasi V. Conflict characterization and analysis of non functional requirements: an experimental approach. In: *Intelligent Software Methodologies, Tools and Techniques (SoMeT), 2013 IEEE 12th International Conference on*. IEEE, 2013a:83-91. https://doi.org/10.1109/SoMeT.2013.6645645
4. Poort ER, De With PHN. Resolving requirement conflicts through non-functional decomposition. In: *Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on*. IEEE; 2004:145-154. https://doi.org/10.1109/WICSA.2004.1310698
5. Bass L, Clements P, Kazman R. *Software Architecture in Practice*. Addison-Wesley Professional; 2003.
6. Gilson F, Galster M, Georis F. Extracting quality attributes from user stories for early architecture decision making. In: *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE; 2019:129-136.
7. Hu H, Ma Q, Zhang T, et al. *Semantic Modelling and Automated Reasoning of Non-Functional Requirement Conflicts in the Context of Softgoal Interdependencies*. Vol 9. IET; 2015:145-156. https://doi.org/10.1049/iet-sen.2014.0153
8. Paja E, Dalpiaz F, Giorgini P. *Modelling and Reasoning About Security Requirements in Socio-Technical Systems*. Vol 98. Elsevier; 2015:123-143. https://doi.org/10.1016/j.datak.2015.07.007
9. Affleck A, Krishna A. Supporting quantitative reasoning of non-functional requirements: a process-oriented approach. In: *2012*

International Conference on Software and System Process (ICSSP). IEEE; 2012a:88-92.
10. Sadana V, Liu XF. Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements. In: *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*. Vol 1. IEEE; 2007:215-218. https://doi.org/10.1109/COMPSAC.2007.73
11. Martinez GG, Del Carpio ÁF, Gómez LN. A model for detecting conflicts and dependencies in non-functional requirements using scenarios and use cases. In: *2019 XLV Latin American Computing Conference (CLEI)*. IEEE; 2019:1-8.
12. Eismann S, Walter J, von Kistowski J, Kounev S. Modeling of parametric dependencies for performance prediction of component-based software systems at run-time. In: *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE; 2018:135-13 509.
13. Liu CL. Cdnfre: Conflict detector in non-functional requirement evolution based on ontologies. *Comput Stand Interfaces*. 2016;47:62-76. https://doi.org/10.1016/j.csi.2016.03.002
14. Alkubaisy D. A framework managing conflicts between security and privacy requirements. In: *Research Challenges in Information Science (RCIS), 2017 11th International Conference on*. IEEE; 2017:427-432. https://doi.org/10.1109/RCIS.2017.7956571
15. Mairiza D, Zowghi D, Nurmuliani N. Managing conflicts among non-functional requirements. In: *Australian Workshop on Requirements Engineering*. University of Technology; 2009.
16. Affleck A, Krishna A. Supporting quantitative reasoning of non-functional requirements: a process-oriented approach. In: *Proceedings of the International Conference on Software and System Process*. IEEE Press; 2012b:88-92. https://doi.org/10.1109/ICSSP.2012.6225987
17. Horrocks I, Patel-Schneider PF, Boley H, Tabet, S., Grosof, B., Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission* 2004;21(79):1-31. https://www.w3.org/Submission/SWRL/
18. Berners-Lee T, Hendler J, Lassila O. The semantic web. *JSTOR*. 2001;284:34-43. https://www.w3.org/standards/semanticweb/
19. young Bang J, Brun Y, Medvidović N. *Collaborative-Design Conflicts: Costs and Solutions*. IEEE; 2018:25-31.
20. Carvalho RM. Dealing with conflicts between non-functional requirements of UbiComp and IoT applications. In: *Requirements Engineering Conference (RE), 2017 IEEE 25th International*. IEEE; 2017:544-549. https://doi.org/10.1109/RE.2017.51
21. Mairiza D, Zowghi D, Gervasi V. Conflict characterization and analysis of non functional requirements: an experimental approach. In: *Intelligent Software Methodologies, Tools and Techniques (SoMeT), 2013 IEEE 12th International Conference on*. IEEE; 2013b:83-91. https://doi.org/10.1109/SoMeT.2013.6645645
22. Zhang X, Wang X. *Tradeoff Analysis for Conflicting Software Non-functional Requirements*. IEEE Software; 2019:156 463–156 475.
23. Mairiza D, Zowghi D. Constructing a catalogue of conflicts among non-functional requirements. In: *International Conference on Evaluation of Novel Approaches to Software Engineering*. Springer; 2010:31-44. https://doi.org/10.1007/978-3-642-23391-3_3
24. Saeed AAA, Lee SW. Non-functional requirements trade-off in self-adaptive systems. In: *2018 4th International Workshop on Requirements Engineering for Self-Adaptive, Collaborative, and Cyber Physical Systems (RESACS)*. IEEE; 2018:9-15.
25. Egyed A, Grunbacher P. *Identifying Requirements Conflicts and Cooperation: How Quality Attributes and Automated Traceability can Help*. Vol 21. IEEE Software; 2004;50-58. https://doi.org/10.1109/MS.2004.40
26. Letier E, Stefan D, Barr ET. Uncertainty, risk, and information value in software requirements and architecture. In: *Proceedings of the 36th International Conference on Software Engineering*. ACM; 2014:883-894. https://doi.org/10.1145/2568225.2568239
27. Haindl P, Plösch R. Towards continuous quality: measuring and evaluating feature-dependent non-functional requirements in DevOps. In: *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE; 2019;91-94.
28. Wagner S, Lochmann K, Heinemann L, et al. The Quamoco product quality modelling and assessment approach. In: *2012 34th International Conference on Software Engineering (ICSE)*. IEEE; 2012:1133-1142.
29. Carvalho RM, Andrade RM, Oliveira KM. How developers believe invisibility impacts NFRs related to user interaction. In: *2020 IEEE 28th international requirements engineering conference (RE)*. IEEE; 2020:102-112.
30. Abeba G, Alemneh E. Identification of nonfunctional requirement conflicts: machine learning approach. In: Berihun ML, ed. *Advances of Science and Technology*. Springer International Publishing; 2022:435-445.
31. Peng Y, Li B, Wang J, Liu Z. An approach of crossover service goal convergence and conflicts resolution. In: *2020 IEEE World Congress on Services (SERVICES)*. IEEE; 2020:225-230.
32. Roy M, Deb N, Cortesi A, Chaki R, Chaki N. Requirement-oriented risk management for incremental software development. *Innov Syst Softw Eng*. 2021;17(3):187-204. https://doi.org/10.1007/s11334-021-00406-6
33. Roy M, Deb N, Cortesi A, Chaki R, Chaki N. CARO: a conflict-aware requirement ordering tool for DevOps. In: *29th IEEE International Requirements Engineering Conference (RE) 2021*. IEEE; 2021:442-443. https://doi.org/10.1109/RE51729.2021.00061
34. Roy M, Das S, Deb N, Cortesi A, Chaki R, Chaki N. Correlating contexts and NFR conflicts from event logs. *Software and Systems Modeling*. 2023:1-24. https://doi.org/10.1007/s10270-023-01087-4

How to cite this article: Das S, Deb N, Chaki N, Cortesi A. Minimising conflicts among run-time non-functional requirements within DevOps. *Systems Engineering*. 2023;1-22. https://doi.org/10.1002/sys.21715

ANNEXURE 1 - MODELING THE BEHAVIOR OF THE SYSTEM

The system behavior can be modeled as a sequence of state action sequences. There are three key aspects in the dynamically evolving environment described above: (a) the generated data set upto time instant t_i, (b) the NFR characteristics observed at time t_i, and (c) the NFR operationalization which will be deployed between time t_i and t_{i+1}. We model these temporal state-action sequences using Dynamic Bayesian Networks (DBNs). DBNs are chosen over other modeling techniques for its generalized and powerful modeling notations which are apt for modeling the run time adaptations of the operations environment.

A DBN is a stochastic model describing a sequence of possible events. It is a state based model which represents the state of each variable at discrete time intervals. The network consists of a series of time slices, where each time slice indicates the value of each variable at time t_i. Some of the assumptions for our proposed modeling framework are as follows:

- The state variables at time t_i depend only on the state variables at time t_{i-1} (and other variables at time t_i).

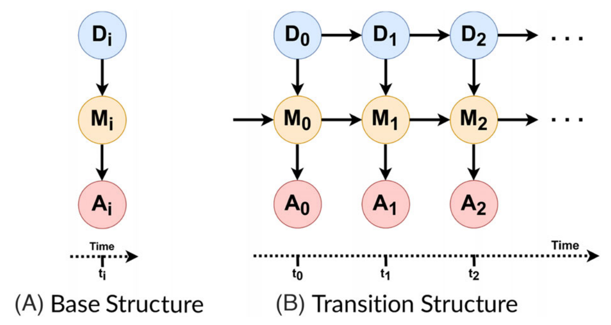


FIGURE A1 Dynamic Bayesian Network model for the system.

- It is a stationary process. The structure and parameters of the model do not change over time.

In order to define the DBN model for the dynamic adaptation of NFR operationalizations based on desired NFR characteristics, we need to define the *base network* and the *transition network* which are fundamental structures within a DBN model. The base network is shown in Figure A1A. It consists of the following components:

- D_i: The cumulative data generated till the i-th time instant t_i.
- M_i: The set of parameters that define a state at time t_i. Each parameter is defined by a 2-tuple of the form $\langle name, value \rangle$.
- A_i: It is the action to be taken at time t_i that best fits the desired performance characteristics. It uses the parameters listed in M_i (calculated from the data D_i) to check if they satisfy the desired percentile of NFR satisfaction.

The transition network is composed of the temporal base networks shown in Figure A1B. The transition network can be described as follows:

- A state transition occurs when the next time slice expires.
- The edge from D_i to D_{i+1} represents that the cumulative data recorded in D_{i+1} is derived from D_i plus the incremental data generated in the $(i+1)$-th time slice.
- The edge from M_i to M_{i+1} represents the change in desired performance characteristics associated with the NFR.
- The temporal base network at the end of each time slice observes the parameters of the NFR characteristics and compares them with the desired NFR satisfaction criteria to decide the operationalization for the next slice.
- The transition network continues as long as the system is up and running.

The DBN model depicted above captures the dynamic NFR characteristics of the system and, based on changing NFR satisfaction criteria, it automates the process of choosing alternate NFR operationalizations for meeting those criteria (Algorithm A1).

ALGORITHM A1 Best Fit NFR Operationalization Selection

Input: $\mathbb{O}_p$ is the set of all operationalizations of Bandwidth Efficiency NFR. D_i is the dataset at a given time instance t_i. $\mathbb{P}$ is the percentile criteria and $\mathbb{T}_h$ is the threshold criteria.
Output: System will select best fitted NFR operationalization (B_f)

```
1:  procedure Main$\mathbb{O}_p, D_i, \mathbb{P}, \mathbb{T}_h$
2:      $CE \leftarrow \langle name, value \rangle$
3:      for $O \in \mathbb{O}_p$ do
4:          $C_e \leftarrow Compression\_Efficiency(D_i, O, \mathbb{P})$
5:          $CE \leftarrow CE \cup C_e$
6:      end for
7:      $D_o \leftarrow \langle name, value \rangle$
8:      for $e \in CE$ do
9:          $d \leftarrow CalculateDistance(e, \mathbb{T}_h)$
10:         $D_o \leftarrow D_o \cup d$
11:     end for
12:     $B_f \leftarrow FindBestFit(D_o)$
13: end procedure
14: procedure FindBestFit$D_o$
15:     $flag \leftarrow 0$
16:     for $d \in D_o$ do
17:         if $d > 0$ then
18:             $flag \leftarrow 1$
19:         end if
20:     end for
21:     if $flag = 0$ then
22:         return Max($D_o$)
23:     else
24:         return Min($D_o$)
25:     end if
26: end procedure
```

AUTHOR BIOGRAPHIES

Souvick Das is a Research Associate at the Department of Environmental Science, Informatics, and Statistics, of Ca' Foscari University, Venice, Italy and is currently pursuing Ph.D. in Computer Science and Engineering from University of Calcutta, India. He has received B.Sc and M.Sc in Computer Science from West Bengal State University, India in the year 2012 and 2014 respectively. He has qualified UGC NET-JRF in the year of 2016.

Novarun Deb is an Assistant Professor at the Indian Institute of Information Technology, Vadodara, India. Dr. Deb was a Research Associate at the Department of Environmental Science, Informatics, and Statistics, of Ca' Foscari University, Venice, Italy, for more than two years. He has done his Masters and Ph.D. in Requirements Engineering from the Department of Computer Science and Engineering, University of Calcutta.

Nabendu Chaki is a Professor in the University of Calcutta, Kolkata, India. He is sharing seven international patents including four US patents. Dr. Chaki has authored seven books and nearly 200 Scopus Indexed papers in Journals and International conferences. He is the founder Chair of ACM Professional Chapter in Kolkata. He was active in 2009-2015 for developing international standards in Software Engineering and Service Science as a Global (GD) member for ISO-IEC.

Agostino Cortesi is a Professor of computer science in Ca' Foscari University, Venice, Italy. He has extensive experience in the area of static analysis and software verification techniques, with particular emphasis on security applications. He published more than 150 papers in high level international journals and proceedings of international conferences. Currently, he serves as co-Editor in Chief of the book series "Services and Business Process Reengineering" published by Springer Nature.