applied sciences

MDPI

Article

Towards Design and Feasibility Analysis of DePaaS: AI Based Global Unified Software Defect Prediction Framework

Mahesha Pandit [1], Deepali Gupta [1], Divya Anand [2,3], Nitin Goyal [1,*], Hani Moaiteq Aljahdali [4], Arturo Ortega Mansilla [3,5], Seifedine Kadry [6] and Arun Kumar [7]

1 Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura 140401, India; mahesha.pandit@chitkara.edu.in (M.P.); deepali.gupta@chitkara.edu.in (D.G.)
2 Department of Computer Science and Engineering, Lovely Professional University, Phagwara 144411, India; divyaanand.y@gmail.com
3 Higher Polytechnic School, Universidad Europea del Atlántico, C/Isabel Torres 21, 39011 Santander, Spain; arturo.ortega@uneatlantico.es
4 Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 37848, Saudi Arabia; hmaljahdali@kau.edu.sa
5 Department of Project Management, Universidad Internacional Iberoamericana, Campeche 24560, Mexico
6 Faculty of Applied Computing and Technology, Noroff University College, 94612 Kristiansand, Norway; skadry@gmail.com
7 Panipat Institute of Engineering and Technology, Samalkha 132102, India; ranaarun1.ece@piet.co.in
* Correspondence: dr.nitingoyal30@gmail.com

Featured Application: DePaaS has the potential to be used as a global, shared platform for availing software defects prediction services by choosing appropriate base project, defect prediction model and prediction granularity. Over time, DePaaS can potentially become a rich source of defects metadata and provide deep insights into developing efficient software defects prediction models. It can promote inter-agency collaboration, data sharing, continuous improvement, and further research into application of artificial intelligence, genetic programming, and other techniques for solving key problems of software engineering.

Abstract: Using artificial intelligence (AI) based software defect prediction (SDP) techniques in the software development process helps isolate defective software modules, count the number of software defects, and identify risky code changes. However, software development teams are unaware of SDP and do not have easy access to relevant models and techniques. The major reason for this problem seems to be the fragmentation of SDP research and SDP practice. To unify SDP research and practice this article introduces a cloud-based, global, unified AI framework for SDP called DePaaS—Defects Prediction as a Service. The article describes the usage context, use cases and detailed architecture of DePaaS and presents the first response of the industry practitioners to DePaaS. In a first of its kind survey, the article captures practitioner's belief into SDP and ability of DePaaS to solve some of the known challenges of the field of software defect prediction. This article also provides a novel process for SDP, detailed description of the structure and behaviour of DePaaS architecture components, six best SDP models offered by DePaaS, a description of algorithms that recommend SDP models, feature sets and tunable parameters, and a rich set of challenges to build, use and sustain DePaaS. With the contributions of this article, SDP research and practice could be unified enabling building and using more pragmatic defect prediction models leading to increase in the efficiency of software testing.

Keywords: software defect prediction; cross-project defect prediction; DePaaS; defect prediction as a service; cloud-based defect prediction; software defect prediction service

Citation: Pandit, M.; Gupta, D.; Anand, D.; Goyal, N.; Aljahdali, H.M.; Mansilla, A.O.; Kadry, S.; Kumar, A. Towards Design and Feasibility Analysis of DePaaS: AI Based Global Unified Software Defect Prediction Framework. *Appl. Sci.* **2022**, *12*, 493. https://doi.org/10.3390/app12010493

Academic Editor: José Carlos Bregieiro Ribeiro

Received: 8 November 2021
Accepted: 4 December 2021
Published: 4 January 2022

1. Introduction

Studies have estimated that about 36 billion electronic devices would connect to the internet by 2021 [1] and about 54% of the world population would be online using software of various kinds [2]. The volume of software in terms of lines of code is estimated to be remarkably high—for example, the size of a typical iPhone application is estimated to be about 10–15 thousand lines of code, and the size of Google's code base to be about two billion lines of code [3]. Since the quality of human life has become dependent on computing devices and the software they run, it becomes quite important to ensure high quality within the high volume of software produced [3].

Building the software is a unique engineering effort and it involves four distinct characteristics of complexity, conformity, changeability, and invisibility, each of which has the potential to introduce defects in the software produced [4]. A research has listed multiple adverse economic and social situations caused by defective software including–"a software defect in converting measurement units" leading to the loss of NASA's Mars climate orbiter which was worth $125 million, "a numeric overflow error" crashing the Ariane 5 rocket whose development cost was about $8 billion and "a software defect" causing the largest non-nuclear explosion in the former Soviet Union [5]. A study shows that software quality assurance (SQA), an activity that helps preventing software defects such as the ones listed above, is quite expensive to deploy and operate [3]. Research shows that in the year 2018, about $4 trillion were spent on IT and telecom systems development [6] and about $2.84 trillion was attributed to the cost of poor software quality in the USA region alone [3]. The same research estimated that about 37% of the cost of poor quality is attributed to software failures and about 17% is to find and fix software defects [3].

Software quality assurance is a collection of activities performed by software developers to ensure that the software produced meets a specific quality standard. With research highlighting a high cost of poor software quality, there is a need to prioritize SQA activities to eliminate maximum number defects with a minimum spend on resources [7]. One such activity would be to isolate parts of the software that is more prone to defects. Such a process of determining areas of a software system that may contain software defects is called *software defect prediction* (SDP) [7]. It is also called as *defect prediction in software* (DeP) by some authors [8]. It should be noted that the ultimate objective of SDP is defect removal and SDP is not the only defect removal method [7]. There are several other defect removal methods including model checking [9], static analysis [10], fault localization [7], etc. The main difference between them and SDP is that these methods help identifying defects in the current codebase whereas SDP " . . . warns about future defect-prone areas . . . " as well [7].

SDP needs a dataset consisting of software features such as afferent coupling (Ca), Class Leaf Depth (CLD), Coupling between Objects (CBO), Efferent Couplings (Ce), Lines of Code (LOC), Number of Weighted Methods (WMC), Response for a Class (EFC), etc. These features could be captured either at class level or method level. An SDP model is built by selecting the best combinations of these software features. Employing a labelled training dataset (having an indication whether the software module is defective or not), the SDP model is trained to detect relation between input software features and the output defect indication. A trained SDP model is run on the software feature dataset of the software under development to classify software modules as defective or non-defective.

In earlier years, several statistical models were proposed to locate defect prone modules of software under development [11]. Recently, machine learning, search-based and hybrid artificial intelligence techniques are commonly proposed for SDP.

There is a call to apply SDP techniques in the early stages of software development lifecycle so that practitioners can prepare to test defect-prone modules with more rigor [12]. This is found to reduce software development and maintenance costs [13,14]. Some researchers have also defined "an ideal DeP model" [15]. Yet, SDP has not yet become a part of the software development process. The level of understanding of SDP models is low among software practitioners and managers [7]. Reasons for such low awareness could be attributed to multiple unsolved problems with SDP identified by researchers including Shi-

hab [7], Zhou [16], Bowes et al. [17], Porto et al. [18] and Son et al. [15]. However, the study did not find any literature that directly attributes low usage of SDP to any specific subset of SDP problems. However, the research found that in most cases, SDP was described as a stand-alone technique. The research did not find any practical attempt to describe SDP to software practitioners in a pragmatic, industry-friendly manner or allow easy access to a testbed of ongoing SDP research. Lack of industry participation could also be a strong reason for the fragmented state of SDP research.

To consolidate the ongoing fragmented SDP research, and to bring SDP closer to the industry practitioners, a cloud-based, multi-model SDP framework has been proposed. It is named as *DePaaS—Defect Prediction as a Service*. It is intended to serve as a global, unified platform that serves both researchers who build SDP models and software industry practitioners who consume defect prediction services provided by these SDP models. The proposed framework has the potential to help overcoming issues related to SDP's core objectives, input data quality, cross-project usage, SDP model performance and practical utilization by the software development community. The current research field-tested the idea of cloud-based SDP in many contexts and gathered deep insights into the way software practitioners perceive the defect prediction problem and the proposed DePaaS solution. The complex process of SDP was better understood by software practitioners when it was presented as a cloud-based service with a componentized architecture and a process flowchart.

The current research has produced a strong, detailed vision for a global, unified, AI based SDP framework called DePaaS. It has also obtained an early feedback on both SDP and DePaaS from the software practitioners. This article explains both results in Sections 3 and 4, respectively.

2. Research Design

This section explains the motivation for the research behind this article and key research questions.

2.1. Motivation for Research

Literature survey shows that many researchers have proposed on-premises, localized use of SDP models. Less than 10% of the literature shows that SDP models are deployed on a public cloud [19]. Current research did not find any major system equivalent to DePaaS except for one from IBM that proposed a software system called IGNITE to predict software defects using dynamic regression-based model with k-fold validation [20].

An enhanced and expanded effort is needed to develop SDP framework that is provided as a "Software Defect Prediction Service" on a public cloud for the use by the wider software development community.

2.2. Research Design

The current research was conducted to find answers for two research questions:

RQ1: What would be the design of DePaaS: a unified, global software defect prediction model which could be used by both SDP researchers and the software industry practitioners?
RQ2: Subsequent sections of this article present answers to these research questions.

Subsequent sections of this article present answers to these research questions.

3. DePaaS: Architecture and Design

This section explains use cases and the layered, modular architecture of DePaaS. It also examines technical feasibility of implementing DePaaS.

3.1. Users, Usage Contexts and Use Cases

3.1.1. Usage Contexts

The design of DePaaS enables conducting cross-project, cross-release, cross-version, cross-company, and one-off software defect prediction.

3.1.2. Users

DePaaS is designed as a unified platform intended to be used by both SDP researchers and software industry practitioners. On the SDP research side, authors of the SDP models could use DePaaS to offer their SDP models for testing and industry use. Such users are identified as "SDP Model Provider" on DePaaS. The platform supports "Dataset Provider" who uploads public and private datasets onto DePaaS. Once SDP models and datasets are uploaded to DePaaS, the industry practitioners, who are members of the software project under development could use DePaaS for software defect prediction. Such users are identified as "SDP Runners" on DePaaS. The platform has "DePaaS Admin" who administers the DePaaS platform by taking care of security, model availability, dataset availability and fine-tunes parameters of DePaaS workflow. The platform has a provision for independent "SDP Researchers" to observe the performance of SDP models. This role is intended to help spread awareness of SDP among both researchers and practitioners.

3.1.3. Use Cases

DePaaS provides services to both academic researchers and industry practitioners through its use cases. The initial design of DePaaS supports the following five use cases.

(1) *User Registration*: All roles, except "DePaaS Admin", register themselves into DePaaS. The "DePaaS Admin" comes as a built-in feature of the DePaaS platform. With this use case, academic researcher and industry practitioner will get a unique identity within DePaaS using which DePaaS services could be accessed.
(2) *Uploading SDP Models*: The "SDP Model Provider" uploads the SDP model by providing details such as model description, suitable usage contexts, suggested datasets, tunable parameters, model performance values, known issues, etc. The user also uploads the executable files of the SDP model.

With the help of this use case, DePaaS would provide an opportunity for the SDP Model Provider to showcase the novel SDP model. Simultaneously, the use case meets the industry demand for novel and improved SDP models.

(3) *Uploading Datasets*: The "Dataset Provider" uploads the defect dataset by providing details such as the description, information about the source of the dataset, available software features, suggested feature combinations, known issues with the dataset, etc.

With the help of this use case, DePaaS meets the data-demand of the software research community and the software industry. As the volume and diversity of the dataset increases, better SDP models could be developed.

(4) *Performing (or running) SDP and looking at results*: The "SDP Runner" runs the SDP workflow. The platform handholds the journey of the user through the SDP process.

With the help of this use case, DePaaS serves the software industry with vital information about software defects. It also serves the author of the SDP model by providing feedback about the quality of the SDP model.

(5) *Improving DePaaS*: At the end of each SDP run, the model performance is evaluated. The source dataset, feature sets, values of the tunable parameters, performance parameters are preserved for future analysis and improvements. "SDP Researchers" can analyse the historical SDP data and suggest new values for tunable parameters or suggest new combination of feature sets. They can also suggest ways to clean-up datasets and call for uploading/creation of novel SDP models. Such activities would improve the DePaaS platform, which serves the interests of both the academicians and the practitioners.

3.2. Functional Description

A functional description of the DePaaS platform is shown in Figure 1. It was developed to describe DePaaS to industry practitioners.

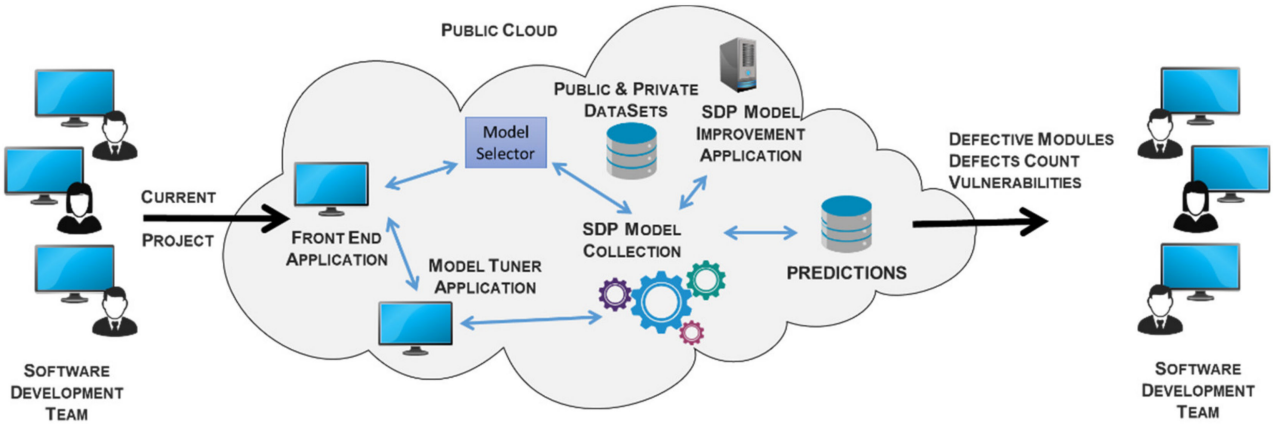

Figure 1. Functional diagram of DePaaS.

DePaaS shall be hosted on a public cloud as a publicly consumable service. It shall consist of a set of SDP models that can be accessed by software development teams using a front-end application. SDP models are built on feature sets from public datasets and cross-project/cross-version/cross-release defects data held on DePaaS. SDP models offer several AI based prediction techniques including–machine learning or search-based or hybrid techniques [8] and produce results of varying accuracy.

A model selector algorithm shall help selecting the most suitable SDP model matching the project on hand and such an algorithm might resemble the meta-learning solution proposed by Porto et al. [18]. DePaaS shall also have a model tuner application that shall help fine-tuning parameters of the chosen prediction model and to train-the model on a new dataset in near-real-time.

Software project teams, with the help of the front-end application, upload the meta-data of their current project including values for select software metrics. The front-end application hand holds the software development team through the SDP process which produces various outputs such as list of defective modules, module wise count of defects, severity of defects, etc. These results are offered to software development teams and are also stored in the DePaaS database.

DePaaS shall also contain SDP model improvement application that could be built to continuously improve the performance of DePaaS including the SDP model collection. It shall enable evolutionary learning within SDP techniques used in DePaaS and its continuous improvements shall help users to get the best performance over time.

As this schematic was developed for the consumption of software practitioners, other users such as SDP Model Provider, Dataset Uploader, DePaaS Admin, etc., are not depicted in this diagram.

3.3. SDP Models Provided by DePaaS

The literature of software defect prediction includes more than eighty (80) SDP models grouped into five classes: statistical, traditional machine learning, novel machine learning, search based and hybridised techniques.

- *Statistical SDP models* are built on techniques such as linear regression, logistic regression, naïve Bayes classifier, K-nearest neighbour, Bayesian networks, Discriminant Analysis, Correlation Analysis, Multivariate Adaptive Regression Splines (MARS), and Negative Binomial Regression (NBR), etc.

- *Traditional machine learning SDP models* are built on machine learning techniques such as decision trees, Bayesian Learning, Ensemble Learning, Evolutionary Learning, Neural Networks, Support Vector Machines, and Rule Based Learning, etc.
- *Novel machine learning SDP models* include Batch Nested Generalized Exemplar (BNGE), Exemplar-Aided Constructor of Hyperrectangles (EACH) and Rule Induction from a Set of Exemplars (RISE), Partial Decision Trees (PART), PrUning and BuiLding Integrated in Classification (PUBLIC), Rule Induction with Optimal Neighborhood Algorithm (Riona) and Simple Learner with Iterative Pruning to Produce Error Reduction (SLIPPER), etc.
- *Search-based (SBT) SDP models* include Bioinformatics-oriented hierarchical evolutionary learning (BioHEL), CO-Evolutionary Rule Extractor (CORE), CHC Adaptative Search for Instance Selection (CHC), Generational Genetic Algorithm for Instance Selection (GGA), Population-Based Incremental Learning (PBIL), Steady-State Genetic Algorithm for Instance Selection (SGA), Constricted Particle Swarm Optimization (CPSO), Linear Decreasing Weight PSO (LDWPSO), Real Encoding PSO (REPSO), Bojarczuk GP, Falco GP, Tan-GP, GA based Classifier System with Adaptive Discretization Intervals (GA-ADI), GA based Classifier System with Intervalar Rules (GA-Int), Incremental Learning with GA (ILGA), GA for solving problem MAX-F (OlexGA), Supervised Inductive Algorithm (SIA), Supervised Classifier System (UCS), X-Classifier System (XCS), and Memetic Pittsburgh Learning Classifier System (MPLCS), etc.
- *Hybridised (HBT) SDP models* Decision Tree- Genetic Algorithm (DT-GA), Oblique Decision Tree with Evolutionary Learning (DT-Oblique), Tree Analysis with Randomly Generated and Evolved Trees (TARGET), Genetic Algorithm with MSE estimation (GA_MSE), Intelligent Genetic Algorithm for Edition (IGA), Genetic based Fuzzy Adaboost (GFS-AB), Genetic Fuzzy Learning with Logitboost (GFS-LB), Logitboost with Single Winner Inference (GFS-MaxLB), Fuzzy rule approach based on a genetic cooperative-competitive learning (GFS-GCCL), Fuzzy Learning based on GP (GFS-GP), Fuzzy Learning based on GP Grammar Operators (GFS-GPG), Grammar Operators and Simulated Annealing (GFS-SP), Hierarchical Decision Rules (HIDER), Neural Network Evolutionary Programming (NNEP), Steady-State GA for Extracting Fuzzy Classification Rules from Data (SGERD), and Structural Learning Algorithm in a Vague Environment with Feature Selection (SLAVE), etc.

DePaaS, by design, can provide multiple SDP model describing its usage details and performance details. To start with, DePaaS aims to provide two best performing SDP models from three classes–machine learning, search-based and hybridised techniques. Malhotra has compared the performance of such novel machine learning, search-based and hybridised SDP models [21]. Performance parameters (G-Mean and Balance) of the top performing SDP models intended to be included in DePaaS is provided in Table 1 along with the Friedman score as determined by Malhotra.

Table 1. Relative Performance of SDP Models.

Class	Technique	G-Mean			Balance		
		Min	Max	Result of Friedman Test	Min	Max	Result of Friedman Test
ML	Slipper	0.24	0.88	33.6	21.43	86.39	28.5
	C4.5	0.33	0.89	31.81	12.01	94.34	31.2
SBT	Bojarczuk	0.47	0.86	38.8	37.49	75.72	26.5
	ILGA	0	0.89	20.4	29.12	90.36	27.2
HBT	DT-GA	0.40	0.86	32.1	36.54	94.34	34.1
	DT-Oblique	0.41	0.87	35.3	31.66	82.06	27.1

Thus, DePaaS shall include six best SDP models by design along with a facility to add any number of SDP models of various types. These six models could serve as the initial benchmark against which performance of new models could be compared.

3.4. Architecture

The conceptual framework of DePaaS described in Figure 1 is converted into a detailed technical architecture. The architectural components are grouped into five layers. Each layer shows modules, which are implementation units of software that provide a coherent set of responsibilities.

Five types of users are identified—SDP Researcher, SDP Model Provider, Dataset Provider, Project Team or End User or SDP Runner and SDP Admin. Internal data entities are also identified. The resulting technical architecture is shown in Figure 2.

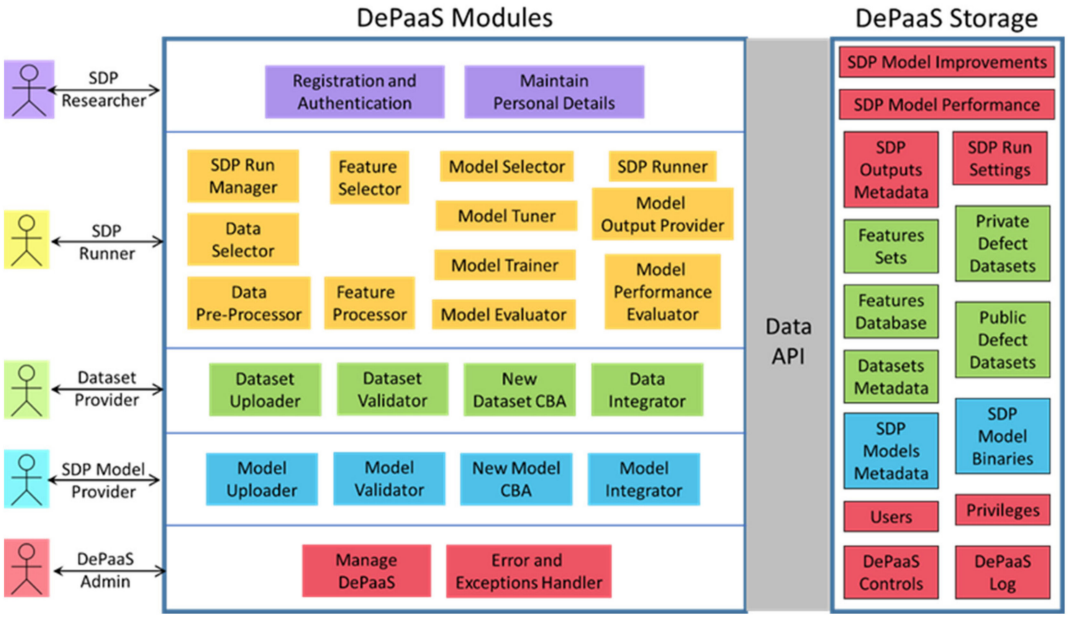

Figure 2. Layered, Modular Architecture of DePaaS.

To keep the high-level architecture simple to understand, messages passed between components are abstracted and not shown.

The architecture separates software modules and storage. At least five layers can be recognized in DePaaS architecture. The number of layers coincidently match with the number of users. However, there is no strict one-to-one correlation between users and layers even though Figure 2 implies it. Each layer is briefly described below:

- *Security Layer*: This layer consists of modules that register end users, maintains their profiles, and impose role-based security across the DePaaS platform at the application level.

DePaaS shall consist of multiple security controls, including *integration of active directory services* provided by the cloud service provider, dual factor authentication before uploading SDP model and datasets, formation of security groups to enforce common security controls and enforce perimeter network control using a firewall.

DePaaS shall enforce encryption, anonymization of names of the organization, business unit, project, software product, module file, package, class and methods. Each user shall have own containers which prevents sharing of data. While uploading SDP models and datasets, end users can mark specific data elements that they would like to hide from other users.

DePaaS shall seek minimal personal information of DePaaS users, and typically includes name, affiliation and email address.

Incidents related to data security and privacy are logged in DePaaS Logs and are periodically analysed by the DePaaS admin.

- *Data and Feature Set Layer*: This layer consists of modules that manage upload, validation, and integration of datasets. First the details of the dataset such as list of features are accepted, and then the dataset is validated for redundancy and relevance. A cost benefit analysis could also be performed to estimate the perceived value of integrating the new dataset. The dataset meeting the threshold for redundancy, relevance and cost is integrated into DePaaS.
- *Model Management Layer*: This layer consists of modules that manage upload, validation, and integration of SDP models. First the details of the SDP model such as learning technique, tunable parameters, performance parameters, etc., are accepted and the model is validated against acceptable thresholds of performance. A cost benefit analysis could be performed to estimate the perceived value of integrating the new SDP model. The SDP model meeting stated thresholds is integrated into DePaaS.
- *SDP Run Management Layer*: This layer consists of multiple modules that guide the industry practitioner to perform one run of software defect prediction. These modules together implement the SDP process shown in Figure 3.

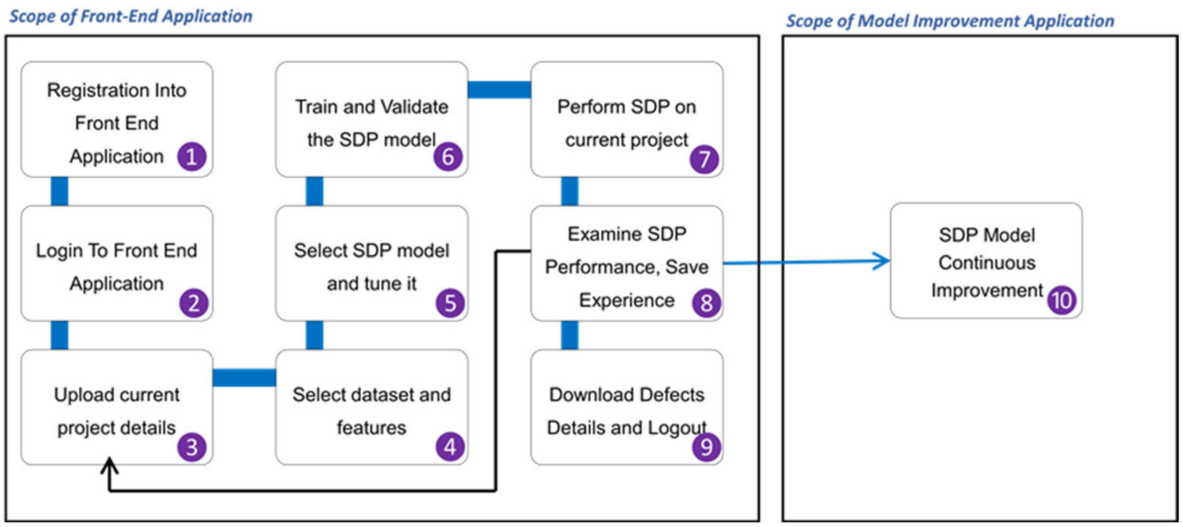

Figure 3. Novel DePaaS Process for use with DePaaS.

To initiate the use of DePaaS, the software development team shall register into DePaaS using the front-end application, choosing an appropriate commercial model such as pay-per-use. Then, the user (a member of the software development team) shall log into the front-end application and choose the most appropriate SDP model based on the nature of the current project and other relevant parameters. The model selector algorithm shall help this process by suggesting the most suitable SDP model/technique or a combination of multiple models/techniques, such as the one based on filter technique [22], or the other based on wrapper technique [23], or a hybrid model based on both filter and wrapper technique [24], or using any other recommendation algorithm [18]. The recommendation algorithm could also be made to choose the most appropriate set of reference projects in the CPDP context.

Once the SDP model is chosen, its parameters, such as granularity, shall be fine-tuned using the model tuner application. As software defect prediction could be done at different granularity, the end user shall specify the appropriate level at which the software defects need to be predicted. Choices of granularity could be subsystem, module, file, class, or method.

Then the user shall upload the source code consisting of software code units that needs to be classified as defective or not. Some models may not need the actual source code but work on the defect metadata. Moreover, some of the SDP models extract desired software metrics in real time either from uploaded source code or from the public datasets. In such cases, source code shall be uploaded prior to model selection.

In certain circumstances, the dataset and features need to be selected before selecting the defect prediction model. Such a pre-selection helps detecting issues in the dataset, such as, duplicate features. Additionally, pre-selecting datasets help counting the number of rows, which certain SDP models need as an input. In cases where the model might have a co-dependency on the dataset, the end user may not have a choice of datasets. In such cases, the dataset mandated by the SDP model is automatically chosen.

The chosen model shall then be trained and validated. For this purpose, the chosen dataset might be broken into three parts—the training part, validation part and the testing part. In the cross-project context, the model could be trained on one project's dataset and run on another project's dataset. The chosen model is initially fit on the training dataset. The chosen model might use its own learning method such as gradient descent or stochastic gradient descent. The model is run on the training dataset and results are compared with the desired target indicated by the chosen model. Based on the proximity of the output to the desired target, model parameters are adjusted. After successively fitting the model on the training dataset, the model is run on the validation dataset. This run provides a measure of evaluation of the chosen model. The model parameters are adjusted once again to improve the performance of the chosen model as desired. Finally, the chosen model is run on the test data set which provides a final measure of performance of the tuned model.

The input shall be processed as per the internal logic of the SDP model and unit-wise classification details (list of modules and a flag indicating whether the unit is defective or not) shall be produced by DePaaS as the output. Depending on the SDP model chosen, the output might consist of number of defects (defect count) and a set of source code lines marked as 'risky changes'. The end user shall view SDP outputs and download it in many formats as needed.

The end user could also examine model performance parameters such as Area under the Receiver Operator Characteristic Curve (AUC), G-Mean, Balance, etc., and choose to repeat the defect prediction process with the choice of the same model or by changing the granularity or by choosing an alternate SDP model or a context. DePaaS can provide the highest, lowest, and average performance of the chosen SDP model it has observed over time.

At the end of each usage cycle, DePaaS shall store its prediction experience—a set of statistics, including model performance, useful for further refinement of the model. Such parameters could be used in the future by the SDP model improvement application, which can be manually or automatically invoked after every run of the SDP model.

After having received the intended service, the end user would log out. The history of SDP model runs, and results would be saved for the future use. DePaaS would collect the feedback from the user before logging the user out. This feedback could include user's general comments about the overall experience of using DePaaS and specific comments about specific services that was provided to the user.

DePaaS would ensure that the software code and other details uploaded by the end user for defect prediction are discarded as per the privacy terms agreed with the end user. By design, DePaaS would retain the defects metadata that was generated during the SDP run. If desired by the end user, the names of the software modules, classes and methods could be encrypted before the actual storage.

The proposed DePaaS process could be integrated into the software development process used by the software teams. It could be invoked as early as possible—post coding and before planning of unit testing phase—of the software development process. It is recommended that the DePaaS process invoked before the integration or system testing

phase so that testing effort could be prioritised based on the prediction obtained from DePaaS.

- *Persistence Layer*: This layer consists of database as well as binary files–which are executable files of the SDP models. DePaaS modules access the DePaaS storage through the Data API, which separates data and executable code. Storage elements shown here could contain other sub-storages. For example: SDP Run Settings consists of dataset details, dataset pre-processing details, feature selection details, model tuning parameters, model training data, and possibly some runtime parameters such as number of iterations.

3.5. Advanced Algorithms

Software Defect Prediction is a novel idea for the industry to follow and implement. DePaaS implements several advanced algorithms to help end users to choose the best of SDP models, most appropriate feature sets for training SDP models, and optimum values for model tuning parameters.

- *Recommending Models:* The knowledge base of software defect prediction recognizes about eighty SDP models. End users might find it overwhelming to choose the best SDP model out of this large number of choices. As a means of help, these models could be ranked based on Friedman test scores of SDP model performance. However, an advanced algorithm that recommends the most suitable search based, or hybridized SDP models based on the parameters of the project on hand could be very useful. DePaaS implements such an algorithm.
- *Recommending Features and Feature Sets:* End users need to select a set of features (called feature sets) to train the chosen SDP model. As there are about eighty features that can be extracted from software code and defect datasets, choosing the most optimum set would be a difficult task for DePaaS users. Therefore, DePaaS provides an advanced algorithm to recommend features sets. Basili et al. [25] (1996) established that Chidamber and Kemerer's OO metrics show to be better predictors than the best set of "traditional" code metrics. Hence, they can be offered as the default feature set to train the SDP model. Other feature sets such as Henderson-Sellers' metrics [26], Martin's metrics [27] and QMOOD metrics suite [28] could also be offered by the recommendation algorithm.
- *Recommending Model tunable parameters*: Every SDP model implements a learning and/or searching algorithm. Performance of such algorithms could be influenced by tuning several parameters. Such parameters vary with the chosen SDP models. For example—common tunable parameters of GP SDP models include mutation probability (Mp), crossover probability (Cp), size of population (PS), number of generations (NG) and distance function (DS) [21]. Other parameters include number of rules, number of labels, convergence platform width, etc. [21]. For machine learning based SDP models, tunable parameters include confidence, number of leaf instances, maximum depth, number of labels, feature adjustment rate, number of nodes between prune, maximum number of nearest neighbour and probability of growth and pruning [21]. An advanced algorithm would recommend the most optimum value for these tunable parameters.

Thus, the advanced algorithms of DePaaS could successfully recommend SDP models, feature sets and tunable parameters of SDP models. Such algorithms would reduce the complexity of the SDP process. End users of DePaaS could accept recommendations of these algorithms initially and could take independent decisions as they gain experience with the SDP process.

3.6. Technical Feasibility

A review of the characteristics of DePaaS architecture indicates that the architecture is comprehensive, modular, layered and uses APIs to interact with other modules and

storage. It practices good practices of design including encapsulation, loose-coupling, and high-cohesion. The architecture is extensible as new modules can be added with relative ease. The architecture abstracts extensible concepts allowing for evolution of the platform. For example: Model Tuning not only contains details of tunable parameters for models included in DePaaS, but also enables easy inclusion of tunable parameters that could be identified in the future.

DePaaS could be implemented as a SaaS on a cloud platform-as-a-service such as Amazon Web Services (AWS). DePaaS would implement a multi-tenant architecture, with each application having N-tier architecture within. Alternatively, each application could be deployed as a microservice and be orchestrated with a managed Kubernetes service such as Amazon EKS.

The front-end application could be implemented using NodeJS—a language which can address large volumes of users landing on the platform. SDP models can be implemented using Python.

Each module of DePaaS needs to communicate with other modules. For example: the *model uploader* module needs to interact with *model validator*. Typically, such an interaction takes place by passing messages from one module to another. Considering the volume of messages passed between modules, message queues such as RabbitMQ could be used to manage communication between modules.

DePaaS needs a persistence layer to store user profiles, SDP models, defect datasets, SDP project details, intermediate and final results of SDP runs, model performance details, defects metadata and user feedback. Persistence facility could be implemented using multi-tenant database available on a cloud platform such as Postgres on AWS.

DePaaS needs a few services such as email, file-upload, billing, reporting, etc. Such services need not be implemented within the scope of DePaaS but could be availed from the hosting environment. Typically, providers of such services expose public, restful APIs which can be consumed to avail those services.

The architecture of DePaaS built as a SaaS would be horizontally scalable as new modules could be added and new features could be added to existing modules. DePaaS would also be vertically scalable as it is deployed on the cloud—a scalable infrastructure.

4. Industry Perception Study

To gauge the initial response of industry practitioners to DePaaS, a survey was conducted which also probed the awareness level of SDP among software practitioners. The DePaaS model was also explained with examples and practitioners were asked to offer their views on its need, usefulness and challenges associated with its construction, usage, and sustenance. This section explains the survey and its results in detail.

4.1. Details of the Study

This study was conducted on a large sample (n = 32) of experienced software professionals having experience of 10 to 15 years. The chosen sample was responsible for developing software work products and had the authority to deploy resources necessary to produce high quality, defect-free software products. They had the necessary educational background in computer science, and were aware of the software development life cycle, cost of quality, root causes for software defects, quality maturity models and the impact of high cost of software defect repair. The chosen sample was competent enough to comprehend SDP and judge its usefulness in an industrial setting. They were chosen using the technique of judgmental sampling and each respondent had consented to participate in this study.

Each respondent was interviewed by the research team. First, data for demographic details were collected and verified. Then, an overview of SDP (including known challenges) was given as a 30-min presentation. After this, respondents were asked for their level of understanding of SDP. Then, DePaaS was introduced as a cloud based, unified platform offering AI based models, guided process for software defect prediction. The usage

context, users and use cases were explained. The functional and technical architecture were explained. Any gaps in understanding of SDP and DePaaS were addressed and closed. Each respondent was handed over a questionnaire shown in Appendix A and were given seven to ten days to think about answers.

Once each respondent was ready with their answers to the questionnaire, a second interview was held in which answers to the survey questions were discussed and recorded. Results were tabulated and summarized. Whenever a single response for a question was desired, the median of Likert responses was computed. To arrive at certain conclusions, and to help with the statistical analysis of the solution, two nominal categories were combined.

Results were statistically analysed using 'one dimensional fitness test' using the Chi-square method. Null and alternate hypothesis were proposed for each question. Null hypothesis was rejected either by comparing probability value (p) with significance level (α) or by comparing Chi-square (χ^2) value with critical value. Once the results were found to be significant, the view of the group having highest response percentage was considered to be statistically significant.

4.1.1. Variables

Variables used in the study are tabulated in Table 2.

Table 2. Variables.

Input Variables	Output Variables
Name, age, gender, education, work-experience, designation, role, number of years of experience in the software industry, level of authority, SDLC knowledge, Knowledge of cost of quality, root causes for software defects, quality maturity models and the impact of high-cost of software defect repair, Open ended questions about challenges to build, operate and sustain DePaaS	SDP knowledge before presentation, SDP knowledge after presentation, Comfort Level with DePaaS understanding, Number of days of gap needed for assimilation, Response to each survey question, List of challenges to build, operate and sustain DePaaS.

4.1.2. Threats to Validity

- Threats to Construct Validity: Each of the research question needs a different type of validity. Face or logical validity—a superficial technique was used, which directly asked the responded whether the questions posed were relevant to the goals of the research or not. Questions posed were straightforward, in the sense that, the respondents directly answered the question. An expert review of the questionnaire could help establishing construct validity.
- Threats to conclusion validity: This research presents the median value of the sample pool response as the perceived awareness, perceived belief, perceived benefit, and perceived challenge of SDP and building DePaaS. Additional Chi-square test could be performed to compare the sample pool's responses with those of an industry expert, especially regarding the perceived benefits and challenges of SDP and building DePaaS.
- Threats to external validity: Anticipating a potential researcher bias in generalizing results, judgmental sampling was applied to diversify survey respondents. Software practitioners having a diverse background in terms of age, experience, number of software products developed, technical skills, educational background, etc. were chosen. Future repetitions of the study should consider the fact that the responses could be influenced by the researcher and a suitable statistical factor could be introduced to remove any potential bias.

4.2. Analysis of the Results of the Survey

Results of the survey are presented in multiple sections below along with the statistical analysis details.

4.2.1. Belief in Defect Prediction

The sample pool was asked about whether it believed that an artificial intelligence-based computer model, trained on software metrics, can classify software modules into two categories—defective or non-defective.

About 66% of respondents believed that defective modules could be identified by an intelligent computer program such as DePaaS (χ^2 = 4.1724, CV = 3.84, α = 0.05, p = 0.0411). Believers quoted advances in AI and machine learning as the primary reason to justify their belief whereas non-believers quoted poor-data, non-relation between metrics and defects, use of third-party software as main reasons not to believe in the ability of a computer program to predict defects. One respondent believed the defects are an inherent nature of the software and prediction is not as useful since each module is highly likely to be defective.

4.2.2. Awareness about SDP Technique

The sample pool was given an insight into two decades of research on SDP. It was informed that SDP has the potential to evolve as a robust process and become a step in the software development life cycle. Then, respondents were asked to rate the level of their knowledge about SDP.

About 61% of respondents did not believe that they knew enough about SDP (χ^2 = 6.76, CV = 3.84, α = 0.05, p = 0.0093). About 16% of the respondents had a deeper knowledge of SDP. Only 3% of respondents were aware of advanced capabilities of SDP such as defects count prediction. None of the respondents was aware that the research has progressed to identify risky changes. None of the respondents believed that SDP could estimate the severity of defects.

4.2.3. Desirability of SDP

Explaining the architecture and features of DePaaS, the survey asked the sample pool whether it would desire DePaaS as an integral part of their software development process.

About 70% of respondents welcomed DePaaS as a key step in the software development process (χ^2 = 8.33, CV = 3.84, α = 0.05, p = 0.0039). However, the usage had pre-condition that the software code and/or defect metadata remain secure on the DePaaS platform. Respondents who did not desire DePaaS cited security concerns (loss of code, leak of information about data quality) as the primary reason not to use DePaaS.

This analysis finds that a cloud-based secure AI framework for SDP would be accepted by majority (70%) of software practitioners and though (66% of) software practitioners believe that defect prediction is possible due to advances in AI and machine learning techniques, not many ($\leq$3%) were aware that SDP can estimate defect count, defect severity and identify risky changes with the help of AI algorithms.

4.3. Feedback about Ability of DePaaS to Address SDP Challenges

The sample pool was briefed about typical challenges faced by SDP research and were asked to judge the potential of DePaaS to solve those problem. Results were tabulated and statistically analysed. They are presented in clusters in subsequent sections below:

4.3.1. Solving SDP Problems Related to SDP Objectives and Focus

The sample pool was briefed about SDP problems related to SDP objectives, SDP focus, and was asked whether DePaaS could help solve those problems.

About 62% of software practitioners indicated that DePaaS could help predicting defects across-projects (χ^2 = 8.91, CV = 3.84, α = 0.05, p = 0.0028). Majority of software practitioners believed that DePaaS does not help estimating defect severity (78% negative, χ^2 = 12.45, CV = 3.84, α = 0.05, p = 0.0004), predicting security vulnerabilities (78% negative, χ^2 = 12.45, CV = 3.84, α = 0.05, p = 0.0004), and highlighting risky code changes (83% negative, χ^2 = 17.29, CV = 3.84, α = 0.05, p = 0.0000).

4.3.2. Solving SDP Problems Related to Datasets

The sample pool was briefed about SDP problems related to datasets used in SDP and was asked whether DePaaS could help solve those problems.

According to responses of software practitioners, DePaaS promotes use of industry datasets (53% positive, $\chi^2 = 9.80$, CV = 3.84, $\alpha = 0.05$, $p = 0.0017$) since it would serve as a unified platform or a single place where commercial datasets could be shared for the common good of software defect prediction and prevention. About 69% of practitioners indicated that DePaaS could enforce and ensure data cleanliness as well ($\chi^2 = 7.54$, CV = 3.84, $\alpha = 0.05$, $p = 0.0060$). This can be achieved by introducing data cleanliness as an entry criterion for the dataset being uploaded to DePaaS. About 57% software practitioners indicated that DePaaS could help enforcing a uniform or a standardized way of documenting defect data—however, this belief was not found to be statistically significant ($\chi^2 = 1.20$, CV = 3.84, $\alpha = 0.05$, $p = 0.2733$). Most of the software practitioners (68%) were inconclusive about whether DePaaS would help addressing the problem of imbalanced datasets ($\chi^2 = 5.44$, CV = 3.84, $\alpha = 0.05$, $p = 0.0.19$).

4.3.3. Solving SDP Problems Related to Feature Selection

The sample pool was briefed about SDP problems related to feature selection and was asked whether DePaaS could help solve those problems.

Many respondents (45%) indicated that DePaaS could help feature selection. However, this belief was not found to be statistically significant ($\chi^2 = 1.64$, CV = 3.84, $\alpha = 0.05$, $p = 0.2008$). The front-end application could suggest optimal combination of features that could yield higher prediction accuracy. Such a suggestion could also be made based on the historical data of SDP model performance.

4.3.4. Solving SDP Problems Related to Building SDP Models

The sample pool was briefed about problems related to building SDP models and was asked whether DePaaS could help solve those problems.

Respondent data indicates that DePaaS could help enforcing a robust SDP model building methodology (56% positive, $\chi^2 = 6.55$, CV = 3.84, $\alpha = 0.05$, $p = 0.0105$) and comparing performance of multiple SDP models (60% positive, $\chi^2 = 6.76$, CV = 3.84, $\alpha = 0.05$, $p = 0.0093$). Respondents believed that the model improvement program of DePaaS could collect and analyse model performance which might help consistent performance of SDP models across multiple runs (58% positive) and across models (72% positive). There was no statistical evidence to support either of these beliefs ($\chi^2 = 3.00$, CV = 3.84, $\alpha = 0.05$, $p = 0.0833$ and $\chi^2 = 9.00$, CV = 3.84, $\alpha = 0.05$, $p = 0.0027$, respectively).

Respondent data indicates that DePaaS may not help improving prediction accuracy (58% negative, $\chi^2 = 4.84$, CV = 3.84, $\alpha = 0.05$, $p = 0.0278$) or incorporating security specific models (63% negative, $\chi^2 = 11.64$, CV = 3.84, $\alpha = 0.05$, $p = 0.0006$). Respondent data indicates that DePaaS could promote use of SBT and HBT models ($\chi^2 = 3.85$, CV = 3.84, $\alpha = 0.05$, $p = 0.0499$).

4.3.5. Solving SDP Problems Related to SDP Model Evaluation

The sample pool was briefed about SDP problems related to evaluation of SDP models and was asked whether DePaaS could help solve those problems.

Respondent data indicates that practitioners were inconclusive about whether DePaaS helps ensuring statistical validation of SDP models (77% neutral, $\chi^2 = 1.29$, CV = 3.84, $\alpha = 0.05$, $p = 0.2568$) or removal of author bias while building SDP models (71% neutral, $\chi^2 = 2.78$, CV = 3.84, $\alpha = 0.05$, $p = 0.0956$).

4.3.6. Solving SDP Problems Related to Practical Use of SDP

The sample pool was briefed about SDP problems related to practical use of SDP and was asked whether DePaaS could help solve those problems.

Respondent data indicates that DePaaS could help perform SDP in the CPDP context (72% positive, χ^2 = 8.91, CV = 3.84, α = 0.05, p = 0.0028). As a cloud-based AI service, DePaaS might be easier to comprehend (63% positive, χ^2 = 4.17, CV = 3.84, α = 0.05, p = 0.0411) with the help of a descriptive process.

4.4. Perceived Challenges to DePaaS

To understand the perceived challenges to build, use and improve DePaaS, three open ended questions were posed to the sample pool. Their responses are discussed in sections below.

4.4.1. Perceived Challenges to Build DePaaS

The sample pool was asked to list potential challenges to build DePaaS as a cloud-based AI framework having features listed Section 3 of this article.

Responses, as shown in Appendix B, indicate the presence of several challenges to build DePaaS. Primary concern expressed by 97% of respondents was the lack of clarity about the 'nature' of the defect identified by SDP models. Concerns were also expressed by 74% of the sample pool about the ability of DePaaS to identify GUI type of defects.

Ability to gather defect data, relevance of gathered data and lack of clarity about the theoretical connection between metrics and defects were also significant concerns of 74%, 94% and 52% of respondents, respectively.

Responses highlighted that building model selector (87%); model improvement application (29%) and even SDP model selection (45% positive) are complex tasks.

Respondents questioned the relevance of DePaaS for scripting languages (42%) and ability to cross the technology (programming language) barriers (68%).

4.4.2. Perceived Challenges to Use DePaaS

The sample pool was asked to list potential challenges to use DePaaS as per the 'De-PaaS process' explained in Section 3 of this article.

Responses to DePaaS usage related questions highlighted multiple concerns. Every respondent (100%) questioned security of uploaded software code, defect metadata and other details.

Responses indicated that choosing the appropriate project in CPDP context would be difficult (97%). This could be due to concerns about the model selector algorithm (87%). Similar concern was expressed about finding the data relevant for the chosen project (88%).

Respondents believed that using DePaaS may not easy as there are too many metrics to choose from (53%) which makes feature selection more complex (88%). Model selection and tuning need more skills (56%) which might need a deeper knowledge of programming and software metrics (53%). Some responses indicated that the DePaaS concept is too complex (25%) and difficult to teach (22%).

Some responses indicated the possibility of false alarms (66%) and non-repeatable results (72%). These challenges pertain to the SDP model chosen and not to the DePaaS framework.

4.4.3. Perceived Challenges to Sustain DePaaS

The sample pool was asked to list potential challenges to sustain DePaaS as a cloud-based AI framework and as a step within the formal software development process. In this context, 'sustenance of DePaaS' indicated continued use of DePaaS by the software development teams, addition of new SDP models to the DePaaS architecture and continuous improvement to the performance of SDP models.

Responses to questions about sustainability of DePaaS could provide deeper insights into the viability of DePaaS to become a step within the formal software development process.

Responses indicates that the definition of 'software defect' is likely to be inconsistent across software product teams and technologies (93%). Additionally, most respondents (97%) stated that DePaaS may not identify 'business defects' (cases where the customer requirements specifications were not met). These challenges pertain to specific SDP models as opposed to the concept or idea of DePaaS.

A cluster of responses (83%) indicated that the prediction accuracy might vary across models and might be very low for CPDP. However, only 30% of the respondents believed that the value addition of DePaaS is likely to be low.

Responses observed that software teams do not usually collaborate among themselves to share data and best practices of defects prediction or prevention (90%). In such circumstances, DePaaS might face survival challenges (30%). However, this can be overcome by sharing defect metadata.

77% of respondents believed that DePaaS could not possibly replace the experienced judgment of humans.

Overall, responses provided a rich set of challenges to be overcome while building, using, and sustaining DePaaS. This practitioner perspective is very a good contribution to further research in CPDP.

5. Conclusions

This paper has proposed DePaaS—Defect Prediction as a Service—a cloud-based, multi-model SDP framework. It is intended to serve as a global, unified platform that serves both researchers who build SDP models and software industry practitioners who consume defect prediction services provided by these SDP models.

This paper described the usage context, five types of users, five initial use cases of DePaaS and provided a layered, modular architecture as well. It described the structure and behaviour of architecture components. It established the technical feasibility of implementing DePaaS as a cloud-based software-as-a-service. It also provided a novel process for using DePaaS along with multiple recommendation algorithms to handhold end users.

This paper presented the first ever study of industry perceptions of software defect prediction. It presented statistically validated responses of software practitioners to questions that probed the extent of understanding of SDP among them. It established the need for DePaaS kind of universal SDP platform and proved that such a platform has potential to overcome many of the known problems of the field of software defect prediction.

The paper recommends implementation of DePaaS, continued research onto enhancing advanced recommendation algorithms of the global, unified defect prediction framework and extending the scope of the industry perception survey to cover a larger population.

6. Patents

Authors shall pursue a patent for detailed algorithms of DePaaS in the future.

Author Contributions: Conceptualization, M.P., D.G., D.A.; Methodology, M.P., N.G., D.A.; Validation, D.G., D.A., H.M.A.; Formal Analysis, H.M.A., N.G., A.K.; Investigation, H.M.A., A.O.M., A.K.; Resources, N.G., S.K.; Data Curation, N.G.; Writing—Original Draft, N.G., M.P.; Writing—Review Editing, N.G., H.M.A., A.K.; Supervision, N.G. and D.G., S.K.; Project Administration, A.O.M. and H.M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This project was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (D-61-830-1443). The authors, therefore, gratefully acknowledge DSR technical and financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study to voluntarily participate in the study and share their opinions.

Data Availability Statement: The study has made available the summary of the response data in the Appendices of this paper.

Acknowledgments: This project was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (D-61-830-1443). The authors, therefore, gratefully acknowledge DSR technical and financial support.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Questionnaire and Statistical Analysis of the Industry Perceptions Survey

Table A1. Belief in SDP and other questions.

Category and Question	Responses on Likert Scale and Statistical Analysis				
RQ2A (i): Belief in SDP					
QA1	**Do you believe that a computer program such as DePaaS can classify software modules (file/class/method/function/process) as defective or non-defective?**				
	Strongly Believe	**Believe**	**Neutral**	**Disbelieve**	**Strongly Disbelieve**
	7 (23%)	13 (43%)	1 (3%)	7 (23%)	2 (7%)
	Group 1: Believe = 23% + 43% = 66%, $N_1 = 7 + 13 = 20$ **Group 2:** Disbelieve = 23% + 7% = 30%, $N_2 = 7 + 2 = 9$ **H_0** = There is no difference in the way believers and disbelievers perceive the ability of DePaaS to identify software defects. **H_a** = Extent of belief in DePaaS to identify software defects is statistically different among believers and disbelievers. $\chi^2 = 4.17$, CV = 3.84, $\alpha = 0.05$, $p = 0.0411$, the observed value of believers (66%) is higher than the observed value of disbelievers (30%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief in DePaaS to identify software defects is statistically significant.				
RQ2A (ii): SDP Awareness					
QB1	**How deep is your understanding of SDP–concept, approaches, process, and outputs?**				
	Very Deep	Deep	Neither Deep nor Shallow	Shallow	Very Shallow
	1 (3%)	5 (16%)	6 (19%)	11 (35%)	8 (26%)
	Group 1: Aware = 3% + 16% = 19%, $N_1 = 1 + 5 = 6$ **Group 2:** Unaware = 35% + 26% = 61%, $N_2 = 11 + 8 = 19$ **H_0** = There is no difference in the depth in SDP knowledge among the two groups. **H_a** = Depth of SDP knowledge is statistically different among the two groups. $\chi^2 = 6.76$, CV = 3.84, $\alpha = 0.05$, $p = 0.0093$, the observed value of Group 2 (61%) is higher than the observed value of Group 1 (16%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Lack of SDP awareness among software practitioners is statistically significant.				
RQ2A (iii): Desirability of SDP					
QC1	**Would you desire to have a cloud-based AI framework for SDP as described, as a part of software development project?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	4 (13%)	17 (57%)	3 (10%)	5 (17%)	1 (3%)
	Group 1: Desirable = 13% + 57% = 70%, $N_1 = 4 + 17 = 21$ **Group 2:** Undesirable = 17% + 3% = 20%, $N_2 = 5 + 1 = 6$ **H_0** = There is no difference in the extent of desirability for DePaaS among the two groups. **H_a** = Desirability for DePaaS is statistically different among the two groups. $\chi^2 = 8.33$, CV = 3.84, $\alpha = 0.05$, $p = 0.0039$, the observed value of Group 1 (70%) is higher than the observed value of Group 2 (20%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Desirability for DePaaS among software practitioners is statistically significant.				

Table A1. *Cont.*

Category and Question	Responses on Likert Scale and Statistical Analysis				
	RQ2B (i): General SDP problems				
QD1	**Do you believe that DePaaS models could estimate defect severity?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	2 (6%)	3 (10%)	2 (6%)	21 (68%)	3 (10%)
	Group 1: Believers = 6% + 10% = 16%, $N_1 = 2 + 3 = 5$ **Group 2:** Non-Believers = 68% + 10% = 78%, $N_2 = 21 + 3 = 24$ $\mathbf{H_0}$ = There is no difference in the extent of belief among believers and non-believers that *DePaaS models could estimate defect severity.* $\mathbf{H_a}$ = Extent of belief among believers and non-believers that *DePaaS models could estimate defect severity* is statistically different. $\chi^2 = 12.45$, CV = 3.84, $\alpha = 0.05$, $p = 0.0004$, the observed value of Group 2 (78%) is higher than the observed value of Group 1 (16%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS models could estimate defect severity*'. The evidence to the contrary is statistically significant.				
QD2	**Do you believe that DePaaS could mark or label those parts of code that are vulnerable from security perspective?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	2 (6%)	3 (10%)	2 (6%)	20 (65%)	4 (13%)
	Group 1: Believers = 6% + 10% = 16%, $N_1 = 2 + 3 = 5$ **Group 2:** Non-Believers = 65% + 13% = 78%, $N_2 = 20 + 4 = 24$ $\mathbf{H_0}$ = There is no difference in the extent of belief among believers and non-believers that *DePaaS could label vulnerable code changes*. $\mathbf{H_a}$ = Extent of belief among believers and non-believers that *DePaaS could label vulnerable code changes* is statistically different. $\chi^2 = 12.45$, CV = 3.84, $\alpha = 0.05$, $p = 0.0004$, the observed value of Group 2 (78%) is higher than the observed value of Group 1 (16%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could label vulnerable code changes*'. The evidence to the contrary is statistically significant.				
QD3	**Do you believe that DePaaS could help defect prediction across projects?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	2 (7%)	16 (55%)	7 (24%)	3 (10%)	1 (3%)
	Group 1: Believers = 7% + 55% = 62%, $N_1 = 2 + 16 = 18$ **Group 2:** Non-Believers = 10% + 3% = 13%, $N_2 = 3 + 1 = 4$ A large part of the population (24%) had difficulty answering this question and chose to remain neutral. $\mathbf{H_0}$ = There is no difference in the extent of belief among believers and non-believers that *DePaaS could help defect prediction across projects*. $\mathbf{H_a}$ = Extent of belief among believers and non-believers that *DePaaS could help defect prediction across projects* is statistically different among the two groups. $\chi^2 = 8.91$, CV = 3.84, $\alpha = 0.05$, $p = 0.0028$, the observed value of Group 1 (62%) is higher than the observed value of Group 2 (3%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could help defect prediction across projects*' is statistically significant.				

Table A1. *Cont.*

Category and Question	Responses on Likert Scale and Statistical Analysis				
QD4	**Do you believe that DePaaS could highlight risky code changes?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	1 (3%)	2 (7%)	2 (7%)	22 (73%)	3 (10%)
	Group 1: Believers = 3% + 7% = 10%, $N_1 = 1 + 2 = 3$ **Group 2:** Non-Believers = 73% + 10% = 83%, $N_2 = 22 + 3 = 25$ **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could highlight risky code changes*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could highlight risky code changes* is statistically different. $\chi^2 = 17.29$, CV = 3.84, $\alpha = 0.05$, $p = 0.0000$, the observed value of Group 2 (83%) is higher than the observed value of Group 1 (10%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could highlight risky code changes*'. The evidence to the contrary is statistically significant.				
RQ2B (ii): Dataset problems					
QE1	**Do you believe that DePaaS could use industry datasets?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	3 (9%)	14 (44%)	12 (38%)	2 (6%)	1 (3%)
	Group 1: Believers = 9% + 44% = 53%, $N_1 = 3 + 14 = 17$ **Group 2:** Non-Believers = 6% + 3% = 9%, $N_2 = 2 + 1 = 3$ A large part of the population (38%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could use industry datasets*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could use industry datasets* is statistically different among the two groups. $\chi^2 = 9.80$, CV = 3.84, $\alpha = 0.05$, $p = 0.0017$, the observed value of Group 1 (53%) is higher than the observed value of Group 2 (9%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could use industry datasets*' is statistically significant.				
QE2	**Do you believe that DePaaS could help ensuring data cleanliness?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	2 (7%)	18 (62%)	3 (10%)	5 (17%)	1 (3%)
	Group 1: Believers = 7% + 62% = 69%, $N_1 = 2 + 18 = 20$ **Group 2**: Non-Believers = 17% + 3% = 20%, $N_2 = 5 + 1 = 6$ **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could help ensuring data cleanliness*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could help ensuring data cleanliness* is statistically different among the two groups. $\chi^2 = 7.54$, CV = 3.84, $\alpha = 0.05$, $p = 0.0060$, the observed value of Group 1 (69%) is higher than the observed value of Group 2 (20%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could help ensuring data cleanliness*' is statistically significant.				

Table A1. *Cont.*

Category and Question	Responses on Likert Scale and Statistical Analysis				
QE3	**Do you believe that DePaaS could help documenting defects in a uniform manner?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	5 (16%)	13 (41%)	2 (6%)	5 (16%)	7 (22%)
	Group 1: Believers = 16% + 41% = 57%, $N_1 = 5 + 13 = 18$ **Group 2:** Non-Believers = 16% + 22% = 38%, $N_2 = 5 + 7 = 12$ **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could help documenting defects in a uniform manner*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could help documenting defects in a uniform manner* is statistically different. $\chi^2 = 1.20$, CV = 3.84, $\alpha = 0.05$, $p = 0.2733$, the observed value of Group 1 (57%) is higher than the observed value of Group 1 (38%). Since $p > \alpha$ and $\chi^2 <$ CV, H_0 is true, and cannot be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could help documenting defects in a uniform manner*'.				
QE4	**Do you believe that DePaaS could handle imbalanced datasets?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	0 (0%)	1 (4%)	19 (68%)	6 (21%)	2 (7%)
	Group 1: Believers = 0% + 4% = 4%, $N_1 = 0 + 1 = 1$ **Group 2:** Non-Believers = 21% + 7% = 28%, $N_2 = 6 + 2 = 8$ A large part of the population (68%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could handle imbalanced datasets*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could handle imbalanced datasets* is statistically different. $\chi^2 = 5.44$, CV = 3.84, $\alpha = 0.05$, $p = 0.196$, the observed value of Group 2 (28%) is higher than the observed value of Group 1 (4%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could handle imbalanced datasets*'. The evidence to the contrary is statistically significant.				
RQ2B (iii): Feature selection problems					
QF1	**Do you believe that DePaaS could help selecting features in a formal manner?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	3 (10%)	11 (35%)	9 (29%)	5 (16%)	3 (10%)
	Group 1: Believers = 10% + 35% = 45%, $N_1 = 3 + 11 = 14$ **Group 2:** Non-Believers = 16% + 10% = 26%, $N_2 = 5 + 3 = 8$ A large part of the population (29%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could help selecting features in a formal manner*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could help selecting features in a formal manner* is statistically different. $\chi^2 = 1.64$, CV = 3.84, $\alpha = 0.05$, $p = 0.2008$, the observed value of Group 1 (45%) is higher than the observed value of Group 1 (26%). Since $p > \alpha$ and $\chi^2 <$ CV, H_0 is true, and cannot be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could help selecting features in a formal manner*'.				

Table A1. *Cont.*

Category and Question	Responses on Likert Scale and Statistical Analysis				
RQ2B (iv): Model building problems					
QG1	**Do you believe that DePaaS could help imposing a robust model building methodology?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	4 (13%)	13 (43%)	8 (27%)	4 (13%)	1 (3%)
	Group 1: Believers = 13% + 43% = 56%, $N_1 = 4 + 13 = 17$ **Group 2:** Non-Believers = 13% + 3% = 16%, $N_2 = 4 + 1 = 5$ A large part of the population (27%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could help impose a robust model building methodology*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could help impose a robust model building methodology* is statistically different among the two groups. $\chi^2 = 6.55$, CV = 3.84, $\alpha = 0.05$, $p = 0.0105$, the observed value of Group 1 (56%) is higher than the observed value of Group 2 (16%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could help impose a robust model building methodology*' is statistically significant.				
QG2	**Do you believe that DePaaS could help improving defect prediction accuracy?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	2 (6%)	5 (16%)	6 (19%)	14 (45%)	4 (13%)
	Group 1: Believers = 6% + 16% = 22%, $N_1 = 2 + 5 = 7$ **Group 2:** Non-Believers = 45% + 13% = 58%, $N_2 = 14 + 4 = 18$ **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could help improving defect prediction accuracy*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could help improving defect prediction accuracy* is statistically different. $\chi^2 = 4.84$, CV = 3.84, $\alpha = 0.05$, $p = 0.0278$, the observed value of Group 2 (58%) is higher than the observed value of Group 1 (22%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could help improving defect prediction accuracy*'. The evidence to the contrary is statistically significant.				
QG3	**Do you believe that DePaaS could promote use of search based (SBT) or hybrid techniques (HBT)?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	3 (10%)	15 (48%)	5 (16%)	3 (10%)	5 (16%)
	Group 1: Believers = 10% + 48% = 58%, $N_1 = 3 + 15 = 18$ **Group 2:** Non-Believers = 10% + 16% = 26%, $N_2 = 3 + 5 = 8$ **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could promote use of SBT and HBT models*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could promote use of SBT and HBT models* is statistically different among the two groups. $\chi^2 = 3.85$, CV = 3.84, $\alpha = 0.05$, $p = 0.0499$, the observed value of Group 1 (58%) is higher than the observed value of Group 2 (26%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could promote use of SBT and HBT models*' is statistically significant.				

Table A1. *Cont.*

Category and Question	Responses on Likert Scale and Statistical Analysis				
QG4	**Do you believe that DePaaS could help comparing performance of various SDP models?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	5 (16%)	14 (44%)	7 (22%)	3 (9%)	3 (9%)
	Group 1: Believers = 16% + 44% = 60%, $N_1 = 5 + 14 = 19$ **Group 2:** Non-Believers = 9% + 9% = 18%, $N_2 = 3 + 3 = 6$ A large part of the population (22%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could help comparing performance of various SDP models*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could help comparing performance of various SDP models* is statistically different among the two groups. $\chi^2 = 6.76$, CV = 3.84, $\alpha = 0.05$, $p = 0.0093$, the observed value of Group 1 (60%) is higher than the observed value of Group 2 (18%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could help comparing performance of various SDP models*' is statistically significant.				
QG5	**Do you believe that DePaaS could contain security specific SDP models?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	0 (0%)	3 (10%)	8 (27%)	16 (53%)	3 (10%)
	Group 1: Believers = 0% + 10% = 10%, $N_1 = 0 + 3 = 3$ **Group 2:** Non-Believers = 53% + 10% = 63%, $N_2 = 16 + 3 = 19$ A large part of the population (27%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could contain security specific SDP models*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could contain security specific SDP models* is statistically different. $\chi^2 = 11.64$, CV = 3.84, $\alpha = 0.05$, $p = 0.0006$, the observed value of Group 2 (63%) is higher than the observed value of Group 1 (10%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could contain security specific SDP models*'. The evidence to the contrary is statistically significant.				
QG6	**Do you believe that DePaaS could ensure that multiple runs of model yield same results?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	3 (10%)	15 (48%)	4 (13%)	5 (16%)	4 (13%)
	Group 1: Believers = 10% + 48% = 58%, $N_1 = 3 + 15 = 18$ **Group 2:** Non-Believers = 16% + 13% = 29%, $N_2 = 5 + 4 = 9$ **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could ensure that multiple runs of an SDP model would yield the same result*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could ensure that multiple runs of an SDP model would yield the same result* is statistically different. $\chi^2 = 3.00$, CV = 3.84, $\alpha = 0.05$, $p = 0.0833$, the observed value of Group 1 (58%) is higher than the observed value of Group 1 (29%). Since $p > \alpha$ and $\chi^2 <$ CV, H_0 is true, and cannot be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could ensure that multiple runs of a model yield same results*'.				

Table A1. *Cont.*

Category and Question	Responses on Likert Scale and Statistical Analysis				
QG7	**Do you believe that DePaaS could ensure that two SDP models yield same results?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	3 (11%)	17 (61%)	3 (11%)	3 (11%)	2 (7%)
	Group 1: Believers = 11% + 61% = 72%, $N_1 = 3 + 17 = 20$ **Group 2:** Non-Believers = 11% + 7% = 18%, $N_2 = 3 + 2 = 5$ **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could ensure that two DePaaS models yield same outputs for multiple runs*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could ensure that two DePaaS models yield same outputs for multiple runs* is statistically different among the two groups. $\chi^2 = 9.00$, CV = 3.84, $\alpha = 0.05$, $p = 0.0027$, the observed value of Group 1 (72%) is higher than the observed value of Group 2 (18%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could ensure that two DePaaS models yield same outputs for multiple runs*' is statistically significant.				
	RQ2B (v): Model evaluation problems				
QG8	**Do you believe that DePaaS could statistically validate SDP results?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	0 (0%)	2 (6%)	24 (77%)	3 (10%)	2 (6%)
	Group 1: Believers = 0% + 6% = 6%, $N_1 = 0 + 2 = 2$ **Group 2:** Non-Believers = 10% + 6% = 16%, $N_2 = 3 + 2 = 5$ A large part of the population (77%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers that *DePaaS could statistically validate SDP results*. **H_a** = Extent of belief among believers and non-believers that *DePaaS could statistically validate SDP results* is statistically different. $\chi^2 = 1.29$, CV = 3.84, $\alpha = 0.05$, $p = 0.2568$, the observed value of Group 2 (16%) is higher than the observed value of Group 1 (6%). Since $p > \alpha$ and $\chi^2 <$ CV, H_0 is true, and cannot be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS could statistically validate SDP results*'.				
QG9	**Do you believe that DePaaS models suffer from author bias?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	0 (0%)	2 (6%)	22 (71%)	5 (16%)	2 (6%)
	Group 1: Believers = 0% + 6% = 6%, $N_1 = 0 + 2 = 2$ **Group 2:** Non-Believers = 16% + 6% = 22%, $N_2 = 5 + 2 = 7$ A large part of the population (71%) had difficulty answering this question and chose to remain neutral. **H_0** = There is no difference in the extent of belief among believers and non-believers about *the presence of author bias in DePaaS*. **H_a** = Extent of belief among believers and non-believers about *the presence of author bias in DePaaS* is statistically different. $\chi^2 = 2.78$, CV = 3.84, $\alpha = 0.05$, $p = 0.0956$, the observed value of Group 2 (22%) is higher than the observed value of Group 1 (6%). Since $p > \alpha$ and $\chi^2 <$ CV, H_0 is true, and cannot be rejected at $\alpha = 0.05$. **Conclusion:** There is no statistical evidence to support the claim that the software practitioners believe that '*DePaaS is free from author bias*'.				

Table A1. *Cont.*

Category and Question	Responses on Likert Scale and Statistical Analysis				
	RQ2B (vi): Practical usage problems				
QH1	**Do you believe that DePaaS could be widely used in your company?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	5 (16%)	18 (56%)	2 (6%)	4 (13%)	3 (9%)
	Group 1: Believers = 16% + 56% = 72%, $N_1 = 5 + 18 = 23$ **Group 2:** Non-Believers = 13% + 9% = 22%, $N_2 = 4 + 3 = 7$ **H_0** = There is no difference in the extent of belief among believers and non-believers about *wide use of DePaaS within a company*. **H_a** = Extent of belief among believers and non-believers about *wide use of DePaaS within a company* is statistically different among the two groups. $\chi^2 = 8.53$, CV = 3.84, $\alpha = 0.05$, $p = 0.0035$, the observed value of Group 1 (72%) is higher than the observed value of Group 2 (22%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among software practitioners that '*DePaaS could be widely used in a company*' is statistically significant.				
QH2	**Do you believe that DePaaS could help practitioners to gain a better understanding of SDP?**				
	Very Likely	Likely	Not Sure	Unlikely	Very Unlikely
	7 (22%)	13 (41%)	3 (9%)	6 (19%)	3 (9%)
	Group 1: Believers= 22% + 41% = 63%, $N_1 = 7 + 13 = 20$ **Group 2:** Non-Believers = 19% + 9% = 28%, $N_2 = 6 + 3 = 9$ **H_0** = There is no difference in the extent of belief among believers and non-believers about *the ability of DePaaS to help understanding of SDP*. **H_a** = Extent of belief among believers and non-believers about *the ability of DePaaS to help understanding of SDP* is statistically different. $\chi^2 = 4.17$, CV = 3.84, $\alpha = 0.05$, $p = 0.0411$, the observed value of Group 1 (63%) is higher than the observed value of Group 2 (28%). Since $p < \alpha$ and $\chi^2 >$ CV, H_0 is false, and can be rejected at $\alpha = 0.05$. **Conclusion:** Belief among the software practitioners that '*DePaaS helps better understanding of SDP*' is statistically significant.				

Appendix B. Challenges to Build, Use and Sustain DePaaS

Table A2. Challenges to build, use and sustain DePaaS.

Question Identifier and Tag	Challenge	Responses	%
RQ2C (i) DePaaS building challenges	Nature of defects detected is not clear	30	97%
	Difficult to pin-point defective method/code	30	97%
	Datasets may not reflect project dynamics	29	94%
	Model selector algorithm is not clear-each project needs are different	27	87%
	Company policy prevents sharing project data	23	74%
	Cannot detect certain types of defects example—GUI	23	74%

Table A2. *Cont.*

Question Identifier and Tag	Challenge	Responses	%
RQ2C (i) DePaaS building challenges	Not sure how the concept works across projects of different programming languages	21	68%
	Cannot see association of metrics to defects	16	52%
	Which SDP models to put into DePaaS?	14	45%
	Concept might work for OOAD but not scripting languages	13	42%
	Too difficult to build model improvement application	9	29%
RQ2C (ii) DePaaS usage challenges	Concern about security of code and data	32	100%
	Concern about presence and finding relevant project in CPDP context	31	97%
	Difficulty in feature selection	28	88%
	Lack of relevant data to train models	28	88%
	Not repeatable findings	23	72%
	Potential false alarms	21	66%
	Needs high skills to select and tune the model	18	56%
	Needs deep computer science knowledge (programming and metrics)	17	53%
	Too many metrics to choose from	17	53%
	Training the model might take time	9	28%
	Concept is too complex	8	25%
	Difficult to teach	7	22%
RQ2C (iii) DePaaS sustenance challenges	Inability to spot business/product defects which are most subjective in nature	29	97%
	Definition of defect is unclear-it might vary across user, technology, and product	28	93%
	Slow evolution of SDP models	27	90%
	Software teams do not cooperate to the extent needed	27	90%
	Prediction accuracy is variable and could be very low for CPDP	25	83%
	Cannot replace practitioner's judgment and experience	23	77%
	No clarity on how third-party modules are supported	19	63%
	Not sure how does DePaaS cope with design and code evolution	13	43%
	Cannot beat developer insight about defect-proneness but defect count might help	12	40%
	Low value addition	9	30%
	Not clear as who would maintain DePaaS	9	30%

References

1. Statista, Number of IoT Devices 2015–2025. Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 19 June 2021).
2. CAST, Research Labs | CAST. Available online: https://www.castsoftware.com/resources/research-library/research-labs (accessed on 10 March 2019).
3. Krasner, H. Research Report: The Cost of Poor Quality Software in the US: A 2018 Report | CISQ—Consortium for Information & Software Quality. Available online: https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2018-report/index.htm (accessed on 10 March 2019).
4. Shihab, E. *An Exploration of Challenges Limiting Pragmatic Software Defect Prediction*; Queen's University: Kingston, ON, Canada, 2012.
5. Harley, N. 11 of the Most Costly Software Errors in History. Available online: https://raygun.com/blog/costly-software-errors-history/ (accessed on 10 March 2019).
6. Murray, S. IDC Forecasts Worldwide IT and Telecom Spending to Slow After Last Year's Rebound; Economic Risks Have Increased. Available online: https://www.businesswire.com/news/home/20180621005079/en/IDC-Forecasts-Worldwide-IT-and-Telecom-Spending-to-Slow-After-Last-Year%E2%80%99s-Rebound-Economic-Risks-Have-Increased (accessed on 19 June 2021).
7. Brooks, F.J. *Mythical Man-Month, The: Essays on Software Engineering, Anniversary Edition*; Addison-Wesley: Boston, MA, USA, 1995.
8. Kitchenham, B.; Charters, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Available online: https://userpages.uni-koblenz.de/~{}laemmel/esecourse/slides/slr.pdf (accessed on 21 June 2021).
9. Jhala, R.; Majumdar, R. Software Model Checking. *ACM Comput. Surv. CSUR* **2009**, *41*, 1–54. [CrossRef]
10. Synopsys, Synopsys Software Security | Software Integrity Group. Available online: https://www.synopsys.com/software-integrity.html (accessed on 19 June 2021).
11. Arisholm, E.; Briand, L.C.; Johannessen, E.B. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *J. Syst. Softw.* **2010**, *83*, 2–17. [CrossRef]
12. Catal, C.; Diri, B. A systematic review of software fault prediction studies. *Expert Syst. Appl.* **2009**, *36*, 7346–7354. [CrossRef]
13. He, H.; Garcia, E.A. Learning from Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284. [CrossRef]
14. Radjenović, D.; Heričko, M.; Torkar, R.; Živkovič, A. Software fault prediction metrics: A systematic literature review. *Inf. Softw. Technol.* **2013**, *55*, 1397–1418. [CrossRef]
15. Son, L.; Pritam, N.; Khari, M.; Kumar, R.; Phuong, P.; Pham, T. Empirical Study of Software Defect Prediction: A Systematic Mapping. *Symmetry* **2019**, *11*, 212. [CrossRef]
16. Zhou, Z.H. *Ensemble methods: Foundations and Algorithms*; CRC Press/Taylor & Francis: Boca Raton, FL, USA, 2012.
17. Bowes, D.; Hall, T.; Petrić, J. Software defect prediction: Do different classifiers find the same defects? *Softw. Qual. J.* **2018**, *26*, 525–552. [CrossRef]
18. Porto, F.; Minku, L.; Mendes, E.; Simao, A. A Systematic Study of Cross-Project Defect Prediction with Meta-Learning. Available online: http://arxiv.org/abs/1802.06025 (accessed on 20 June 2021).
19. Catal, C.; Erdogan, M.; Isik, C. Software Defects Prediction in the Cloud. In Proceedings of the 21st Conference of Open Innovations Association FRUCT, Helsinki, Finland, 6–10 November 2017; Available online: https://ieeexplore.ieee.org/servlet/opac?punumber=8241162 (accessed on 20 June 2021).
20. Williams, A.; Patra, J.; Das, S.; Jayaraman, I. IGNITE Defect Predict Provides Early Insights to Prevent Application Failure. Available online: https://www.ibm.com/downloads/cas/4XOVXPDB (accessed on 20 March 2021).
21. Malhotra, R. An extensive analysis of search-based techniques for predicting defective classes. *Comput. Electr. Eng.* **2018**, *71*, 611–626. [CrossRef]
22. Rodger, J.A. Toward reducing failure risk in an integrated vehicle health maintenance system: A fuzzy multi-sensor data fusion Kalman filter approach for IVHMS. *Expert Syst. Appl.* **2012**, *39*, 9821–9836. [CrossRef]
23. Aparisi, F.; Avendaño, G.; Sanz, J. Interpreting Out-of-Control Signals of MEWMA Control Charts Employing Neural Net-works. *Int. J. Comput. Electr. Autom. Control Inf. Eng.* **2010**, *4*, 24–28.
24. Huda, S.; Abdollahian, M.; Mammadov, M.; Yearwood, J.; Ahmed, S.; Sultan, I. A hybrid wrapper–filter approach to detect the source(s) of out-of-control signals in multivariate manufacturing process. *Eur. J. Oper. Res.* **2014**, *237*, 857–870. [CrossRef]
25. Basili, V.; Briand, L.; Melo, W. A validation of object-oriented design metrics as quality indicators'. *IEEE Trans. Softw. Eng.* **1996**, *22*, 751–761. [CrossRef]
26. Henderson-Sellers, B. *Object-Oriented Metrics: Measures of Complexity (Facsimile ed.)*; Pearson College Div., Prentice-Hall, Inc.: Hoboken, NJ, USA, 1995.
27. Martin, R.C. Object oriented design quality metrics: An analysis of dependencies'. *Rep. Object Anal. Des.* **1995**, *2*, 1–8. Available online: https://linux.ime.usp.br/~{}joaomm/mac499/arquivos/referencias/oodmetrics.pdf (accessed on 20 June 2021).
28. Bansiya, J.; Davis, C. A hierarchical model for object-oriented design quality assessment'. *IEEE Trans. Softw. Eng.* **2002**, *28*, 4–17. [CrossRef]