EasyChair Preprint

№ 8599

# Negative Transfer in Cross Project Defect Prediction: Effect of Domain Divergence

Osayande P. Omondiagbe, Sherlock A. Licorish and Stephen G. MacDonell



August 3, 2022

# Negative Transfer in Cross Project Defect Prediction: Effect of Domain Divergence

Osayand P. Omondiagbe
*Department of Information Science*
*Landcare Research and University of Otago*
New Zealand
omondiagbep@landcareresearch.co.nz

Sherlock A. Licorish
*Department of Information Science*
*University of Otago, Dunedin*
New Zealand
sherlock.licorish@otago.ac.nz

Stephen G. MacDonell
*Software Engineering Research Lab*
*Auckland University of Technology, Auckland*
*Department of Information Science*
*University of Otago, Dunedin*
New Zealand
stephen.macdonell@aut.ac.nz

***Abstract*—Cross-project defect prediction (CPDP) models are used in new software project prediction tasks to improve defect prediction rates. The development of these CPDP models could be challenging in cases where there is little or no historical data. For this reason, researchers may need to rely on multiple sources and use transfer learning-based CPDP for building defect prediction models. These data are typically taken from similar and related projects, but their distributions can be different from the new software project (target data). Although, transfer learning-based CPDP models are designed to handle these distribution differences, but if not correctly handled by the model, may lead to negative transfer. To this end, recent works have focused on building transfer CPDP models, but little is known about how similar or dissimilar sources should be to avoid negative transfer. This paper provides the first empirical investigation to understand the effect of combining different sources with different levels of similarities in transfer CPDP. We introduce the use of the Population Stability Index (PSI) to interpret whether the distribution of the combined or single-source data is similar to the target data. This was validated using an adversarial approach. Experimental results on three public datasets reveal that when the source and target distribution are very similar, the probability of false alarm is improved by 3% to 7% and the recall indicator is reduced from 1% to 8%. Interestingly, we also found that when dissimilar source data are combined with different source datasets, the overall domain divergence is lowered, and the performance is improved. The results highlight the importance of using the right source to aid the learning process.**

***Index Terms*—cross-project defect prediction, negative transfer, transfer learning, data shift**

## I. Introduction

Transfer learning cross-project defect prediction (CPDP) techniques are used commonly in new software projects to improve software defect detection [1]. This CPDP approach aims to minimise the distribution differences between the data of the two projects (source and target). Most importantly, when multiple source projects are used, researchers have shown that the performance of these CPDP models can be improved [2]. Also, researchers in the field of transfer learning have shown that data from other projects often yield better models than using only local data [3]. However, using different data sources more often results in data distribution differences, because the software modules in the different projects generate different defects. Although, previous work [4] has shown that by using a particular dataset known as a "bellwether", that is similar to the project dataset, a defect prediction model which performs better than using a CPDP model can be achieved Krishna et al. [4] noted that this bellwether dataset should be changed often as the performance deteriorate. This deterioration is a result of the learning process of transferring negative knowledge, which is known as negative transfer (NT). NT occurs when the distribution differences between the bellwether dataset and the project dataset become greater [5].

NT is a challenging problem in transfer CPDP and other transfer learning based systems [6, 5]. The effectiveness of any transfer learning system using external sources is dependent on three basic assumptions [6]; 1) the learning task between the two domains should be related, 2) the data distribution between the two domains should narrow and, 3) a reliable model which can extract the right information from both domains that can aid the learning process should be used. A software defect system which does not abide by any of these assumptions at any time may lead to NT. The first assumption is easy to follow, as it is easy to identify related tasks. For assumption 3, many models have been proposed ([6]) that can extract the right information from both domains. There is still no empirical study done in transfer learning CPDP to fully understand the impact of domain divergence in the learning process and to what extent domain divergence can be harmful. In this work, we conduct experiments on three publicly available software defect benchmark datasets to understand the relationship between domain divergence and negative transfer. We believe having such information could mitigate NT by choosing the best combination of model and source datasets.

The main contributions of this paper are as follows:

- To the best of our knowledge, we present the first study on the effect of domain divergence in CPDP
- We introduce a simple method for effectively measuring domain divergence in a transfer learning CPDP task.

The rest of this paper is organized as follows. In Section II we introduce related work in software defect prediction and negative transfer. Section III introduces our study design and research questions. Section IV analyses the experimental results and summarises the outcome. We next discuss our findings in Section V, before outlining potential threats to the validity of our outcomes in Section VI. Finally, we conclude this paper in Section VII.

## II. Related Work

Rosenstein et al. [7] was the first to discover the concept of NT. They noted that if two tasks or domains are too dissimilar, bias learned from the source tasks or domain will eventually affect the target task performance. Similarly, Wang et al. [6] pointed out that when the source and target domains are dissimilar and if learning is forced, the performance in the target domain is negatively affected. A recent survey on NT [8] shows that four main factors could lead to NT. These factors are; 1) domain divergence, 2) inadequate learning system, 3) source data quality and, 4) target data quality. Domain divergence was seen to be the root cause of NT, and if this is not taken into account at the feature, classifier, or target output level, then NT is likely to occur. From the literature, various strategies have been proposed to tackle the domain divergence between source and target datasets. Secure or distance methods were seen to be used when there is little or no similarity between the two tasks [9]. In cases where the similarity measure is medium, Zhang et al. [8] reported that it is best to handle the similarity with data, model or target transferability enhancement strategies.

In the domain of CPDP, Briand et al. [10] conducted the first study on such an approach with two Java systems. Their initial work did not report a successful outcome, and they reported that a more complex model was needed for a CPDP task. Zimmermann et al. [11] used a decision tree to conduct defect prediction experiments on 622 pairs of projects, and less than 3.4% of their experiments showed satisfactory performance. This could be attributed to the domain similarities, but their work did not investigate the root cause. Yu et al. [12] conducted empirical research to understand the importance of feature selection, where they discovered the importance of using the right source. Yu et al. [13] investigated the problem of irrelevant source data that could lead to negative transfer and proposed a data filtering method that is based on a semi-supervised clustering approach [14] to filter out bad source data. More recently, Zhao et al. [15] investigated the advantage of using multi-source data over a single dataset in CPDP. From their experimental results, it was seen that the number of times multi-source was favourable over a single dataset was minimal. Based on their outcome, they developed a method, that can resolve the problem of data distribution differences when using multiple source datasets at the same time.

The studies above all focus on either the importance of using multiple sources, selecting the right features or using a robust model. As using more source data could result in more redundant data, it is important to know to what extent domain divergence can affect the learning system. If this information is known beforehand, one could decide which data should be removed, added or corrected by the learning system. This issue is not yet investigated, so this paper aims to understand the full effect of domain divergence on the overall learning system.

## III. Study Design

The purpose of this paper is to investigate how domain divergence could hurt the performance in CPDP. To guide our investigation, we aim to answer the following research questions.

*RQ1. What is the effect of combining multiple sources in CPDP?*

**Motivation:** In CPDP, different methods [9, 6] have been proposed to solve the source data distribution differences issue. If multiple source are combined together, the distribution difference is likely to increase. To date, there is no empirical study to understand the extent of domain divergence that can cause harm to the learning process in CPDP. The findings from this RQ could help improve CPDP system by preventing negative transfer.

*RQ2. Does combining multiple sources in CPDP decrease the domain divergence?*

**Motivation:** Though previous studies have pointed out the benefit of using multiple source data [15], it is still unclear if combining related sources can decrease the total domain divergence and improve or hurt the overall learning performance of model. This question is designed to understand to what extent we should rely on combining different domains in CPDP.

### A. Dataset

We selected three publicly available datasets (AEEEM, NASA and Open-source Java systems datasets). The AEEEM dataset was compiled by D'Ambros et al. [16] and has five open-source projects. Each project is Java-based, and they all have the same 61 different features, which include; 17 source code features, 5 previous-defect features, 5 entropy-of-change features, 17 entropy-of-source-code features, and 17 source code churn features. The NASA and Open-source Java systems datasets are collected and compiled by Jureczko and Madeyski [17] and are available in the PROMISE repository. They have been used extensively in software defect prediction research [18]. For general statistics of these datasets, please refer to Supplementary Material [1] (Table S1).

### B. Experimental Setup

We design 77 different experiments from three datasets (AEEEM, NASA and Open-source Java systems). We choose to use a simple model to avoid model complexity interference, so for this reason a simple KNN classifier was used, as was used in the work of Zhao et al. [15]. The experiment is in two folds; 1) single source and 2) multiple sources. For the single source experiment, we use each project in all three datasets as the target data, then, the source dataset was derived by alternating each project in the dataset (e.g., for the AEEEM dataset, when EQ was used as the target, the remaining dataset is alternated as the source dataset). For this single setup, a total of 30, 20 and 12 experiments were executed from the Open-source Java systems, NASA and AEEEM datasets respectively. For the multi-source experiments, we use each project in all 3 datasets as the target data, then we combine the rest of the projects in each dataset as the source (e.g., for the AEEEM dataset, when EQ was used as the target, the remaining datasets were combined as the source). A total of 15 multi-source experiments were executed. All experiments were conducted 5 times as used in the work of Bennin et al. [19] to reduce the impact of sampling bias and the results were averaged across the independent runs.

### C. Domain Divergence Measure

For the domain divergence measure, we use both the population stability index (PSI) [20] and adversarial approach.

*1) population stability index:* This is a type of information-theoretic measure to interpret whether the distribution of the source and target are similar. PSI was designed for the credit risk industry to monitor the distribution change in the data that was used to develop the credit risk model and validation data. It is now widely used in other domains, as it is closely related to well-established entropy measures [21] and can measure any change in the distribution of explanatory variables [20]. Intuitively, it is the number of information bits lost if we use the source instead of target and then use that information to go back to the source. This is different from Kullback-Leibler Divergence [22], which measures the number of bits lost from source to target.

Calculating PSI is done by using the following steps:

1) Divide the numeric features into a specific number of bins
2) Calculate % of instances in each bin based on the source sample.
3) Choose a cut-off point for all bins.
4) Apply the cut-off points to the target sample.
5) Generate the distributions for the target sample.
6) Calculate the sum statistics across all bins for both the source and target samples to calculate the PSI

$$PSI = \sum_{i=1}^{k} (T_i - S_i) * ln(\frac{T_i}{S_i}) \quad (1)$$

The general equation for PSI [21] is given in "equation (1)" where; $(T_i)$ is the observed relative frequency of occurrences of the response variable in the target datasets, and $(S_i)$ is the relative frequency of occurrences of the response variable in the source datasets, K is the categories numbered from 1 to K, "i" is the category values from 1 to K, and ln() is the natural logarithm. Based on the guidance of

[1] https://zenodo.org/record/6640190

Yurdakul [20] and our sample size for both domains, we choose a bin size of 20 to derive the following benchmark for our PSI:

- PSI $\leqslant$ 0.01 = little change has occurred
- 0.01 $\leqslant$ PSI $\leqslant$ 0.025 = the distribution has slightly shifted
- PSI $\geqslant$ 0.025 = the changes in distribution are significant

The PSI code and dataset are publicly available [2]. Next, we complement our PSI method with an adversarial approach.

*2) Adversarial method using the Kolmogorov-Simirnov test:* To prove the presence of distribution shift, we use an adversarial approach. Here, we detect covariate shift by training a classifier to discriminate between the source and target [23]. To do this, we assign a new variable called "class 0" and "class 1" to the source and target domain respectively. Next, we shuffle the dataset and split the datasets into testing and training before training a simple model with the training set. The model is then used to evaluate the test set, and a significance test is conducted to show that the result was statistically different from random chance. This process was executed 10 times. For the statistical test, we adopt the Kolmogorov-Smirnov (KS) test [22]. A high value of KS indicates that we can discriminate between the source and target.

### D. Evaluation Performance Metrics

Probability of false alarms (PF) and recall or probability of detection (PD) were used as evaluation measures. PF is the percentage of non-defective instances that are misclassified within the non-defect class, while PD is intuitively the ability of the defect prediction model to find all the positive instances (i.e., defect classes). These measures were chosen based on a previous software defect study [19], where they noted that high recall and low false positive rate values are the more stable performance indicators for defect prediction. Intuitively, a defect predictor will "trade-off" between the PF and PD because the more sensitive the detector becomes, the more often the system is triggered and the PD values will be increased. Higher PD and lower PF values indicate a better defect detector.

## IV. Result

In this section, we present detailed experimental results for the indicators (PD and PF) as proposed in Section III-D.

### A. RQ1. What is the effect of combining multiple sources in CPDP?

Table I reports the average values of the two indicators for the KNN model on the NASA datasets. The "↑" after the PD denotes that the indicator needs to be maximized (higher the better), while the "↓" after the PF column denotes that the indicator needs to be minimised (lower the better). The values in the tables depicted in blue indicate that the value was the best in that experiment set. The complete results for the other two sets of experiments (AEEEM and Open-source Java systems datasets) are provided in the Supplementary Material[3] (Table S2 and S3). From the results, we note that when multiple sources were used, we were able to achieve the best result on 2 occasions (i.e., NASA experiment; when using "PC3" or "PC1" as the target and the remaining as the source) for the PD and PF indicators. Also, we noticed a lower value of PF in the AEEEEM result (i.e., when the target was "PDE"). More specifically, when we compared the outcome of using multiple sources with a target source, we were able to achieve an improvement of 3% to 7% in terms of PD and a reduction of 1% to 8% in terms of the PF score. It is also worth noting that the results with the highest values were from the experiment set with the lowest PSI measure (i.e., lower divergences).

[2]https://github.com/pascal082/dataShiftCPDP

[3]https://zenodo.org/record/6640190

### B. RQ2. Does combining multiple sources in CPDP decrease domain divergence?

To understand the extent of relying on different domains in CPDP, we looked at how the combination of different source domains affects the final model performance and the domain divergence. From the results, we noted 3 cases of when combining multiple sources resulted in lowering the domain divergence. These cases were seen in the "ant-1.7 → all", "PC3 → all" and "PC1 → all" experiments. This also increased the PD and decreased the PF values.

## V. Discussion

### A. RQ1. What is the effect of combining multiple sources in CPDP?

Our results show that when multiple source domains are merged, we observe that the number of times a multi-source CPDP is better than a single-source CPDP is lower. This outcome was also noted in the work of Zhao et al. [15]. On further examination of the PSI score, we noticed when the PSI is higher across all predictions, the PD is lowered and the PF is increased. We found that by alternating the source dataset, the performance and divergence measure from each experiment changes. This means one cannot know beforehand the best source to use. Although previous work has shown that more complex CPDP models [6] can handle domain divergence, there is still no evidence on what extent of this domain divergence is acceptable by these models. Here, it is evident that an increase in domain divergence decreases the detection performance. With the KS measure, we were able to correctly identify those sets of datasets that were flagged as having data shift by the PSI method.

### B. RQ2. Does combining multiple sources in CPDP decrease domain divergence?

From the experimental results, we noticed three instances ("LC → all", "PDE → all" and "camel-1.6 → all", etc) when multi-source was used and the domain divergence was greater than using a single source. Also, we were still able to derive a better defect model (i.e, higher PD and lower PF). This could be attributed to the fact that all information are useful, but some are more useful than others. Although a previous empirical study in CPDP has shown that the high variability of the distribution difference found in the source and target datasets can result in high levels of false alarm [24], the results in this paper indicate that when the right set of multiple source domains are combined, then the domain divergence gap is reduced. This was evident in 3 of the multi-source NASA experiments, where the PSI values were decreased when the target dataset was "PC1", PC3" and "PC4". Using the right combination of sources is simpler and faster than building a more complex model or transforming all features to a common latent feature space [25].

## VI. Validity Threats

### A. Addressing threats to validity

**Construct validity:** This paper uses probability of false alarms and recall to assess prediction performance on imbalanced data, since there are no universally agreed upon performance measures. There is a potential threat to our experimental results from the two metrics we consider in this paper. Nevertheless, most empirical studies uses this procedure [19]. **External validity:** We acknowledge that the original datasets might contain some errors in the labels, that may introduce threats to the defect prediction evaluation. We also acknowledged a threat to external validity in terms of whether conclusions can be generalised, since software engineering studies are prone to variability [26]. These threats have been minimised by experimenting with different projects, ranging from open source and academic projects.

TABLE I
PD AND PF FOR THE NASA EXPERIEMENT

| PSI ↓ | Target | Source | PD ↑ | PF ↓ | KS-test (shift) | PSI | Multi-Source | PD ↑ | PF ↓ | KS-test(shift) |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.03 | PC3 | CM1 | 0.837 | 0.501 | Yes | | | | | |
| 0.04 | PC3 | PC1 | 0.849 | 0.498 | Yes | 0.01 | PC3 → $all$ | 0.869 | 0.488 | No |
| 0.04 | PC3 | PC4 | 0.846 | 0.503 | Yes | | | | | |
| 0.03 | CM1 | PC3 | 0.853 | 0.565 | Yes | | | | | |
| 0.08 | CM1 | PC1 | 0.856 | 0.565 | Yes | 0.04 | PC3 → $all$ | 0.853 | 0.489 | Yes |
| 0.01 | CM1 | PC4 | 0.856 | 0.465 | No | | | | | |
| 0.04 | PC1 | PC3 | 0.877 | 0.545 | Yes | | | | | |
| 0.08 | PC1 | CM1 | 0.887 | 0.544 | Yes | 0.014 | PC1 → $all$ | 0.912 | 0.415 | No |
| 0.03 | PC1 | PC4 | 0.887 | 0.544 | Yes | | | | | |
| 0.04 | PC4 | PC3 | 0.796 | 0.583 | Yes | | | | | |
| 0.01 | PC4 | CM1 | 0.857 | 0.482 | No | 0.02 | PC4 → $all$ | 0.826 | 0.483 | Yes |
| 0.03 | PC4 | PC1 | 0.827 | 0.492 | Yes | | | | | |

**Internal validity:** The experimental results in this study may have been influenced by a few uncontrolled factors. For instance, there could have been unexpected error in calculating the PSI. We sought to reduce such threats by executing the experiment five times. Finally, while we recognise the threats above, we anticipate that our study still contributes novel findings to the domain of CPDP and research could use this outcome for selecting the right domain source.

## VII. Conclusion and Future Work

The use of multiple but related sources in CPDP could improve the performance of CPDP, but can sometimes increase the domain divergence. This domain divergence lead to NT. To tackle this NT, it is recommended to address the domain divergence from the model, data, or learning phase. To this end, most recent works have focused on developing a robust model, but little or no understanding is known about how this divergence may further exacerbate when multiple sources are combined, and how this could affect the learning process. Accordingly, we consider understanding the problem of domain divergences and NT in CPDP. We introduced the use of the Population Stability Index to measure the similarity between the domains. Experiments on three software defect benchmark datasets reveal that when domain distribution difference are very similar, the probability of false alarm is improved by 3% to 7% and the probability of detection is reduced from 1% to 8%. Also, our results revealed that when different source data with varied domain divergence were combined, the domain divergence is lowered in some cases, and the performance was improved. These results highlight the importance of using the right domain source to aid the learning process. A plausible next step could be to use a measure such as the characteristic stability index to identify and eliminate features that increases domain divergence.

## Acknowledgement

This research was partly supported by an Internal Research fund from Manaaki Whenua — Landcare Research, New Zealand. Special thanks are given to the Informatics Department for their ongoing support.

## References

[1] S. Herbold, "Training data selection for cross-project defect prediction," in *Proceedings of the 9th international conference on predictive models in software engineering*, 2013, pp. 1–10.

[2] Z. Sun, J. Li, H. Sun, and L. He, "Cfps: Collaborative filtering based source projects selection for cross-project defect prediction," *Applied Soft Computing*, vol. 99, p. 106940, 2021.

[3] F. Peters, T. Menzies, and L. Layman, "Lace2: Better privacy-preserving data sharing for cross project defect prediction," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 801–811.

[4] R. Krishna, T. Menzies, and W. Fu, "Too much automation? the bellwether effect and its implications for transfer learning," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 122–131.

[5] L. Gui, R. Xu, Q. Lu, J. Du, and Y. Zhou, "Negative transfer detection in transductive transfer learning," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 2, pp. 185–197, 2018.

[6] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, "Characterizing and avoiding negative transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 293–11 302.

[7] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *In NIPS'05 Workshop, Inductive Transfer: 10 Years Later*, 2005.

[8] W. Zhang, L. Deng, L. Zhang, and D. Wu, "A survey on negative transfer," 2020. [Online]. Available: https://arxiv.org/abs/2009.00909

[9] B. Cao, S. J. Pan, Y. Zhang, D.-Y. Yeung, and Q. Yang, "Adaptive transfer learning," in *proceedings of the AAAI Conference on AI*, vol. 24, no. 1, 2010, pp. 407–412.

[10] L. C. Briand, W. L. Melo, and J. Wust, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE transactions on Software Engineering*, vol. 28, no. 7, pp. 706–720, 2002.

[11] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *ESEC-FSE'09 - Proc. Jt. 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng.*, 2009.

[12] Q. Yu, J. Qian, S. Jiang, Z. Wu, and G. Zhang, "An empirical study on the effectiveness of feature selection for cross-project defect prediction," *IEEE Access*, vol. 7, pp. 35 710–35 718, 2019.

[13] X. Yu, M. Wu, Y. Jian, K. E. Bennin, M. Fu, and C. Ma, "Cross-company defect prediction via semi-supervised clustering-based data filtering and mstra-based transfer learning," *Soft Computing*, vol. 22, no. 10, pp. 3461–3472, 2018.

[14] L. Lelis and J. Sander, "Semi-supervised density-based clustering," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 842–847.

[15] Y. Zhao, Y. Zhu, Q. Yu, and X. Chen, "Cross-project defect prediction considering multiple data distribution simultaneously," *Symmetry*, vol. 14, no. 2, p. 401, 2022.

[16] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, no. 4, pp. 531–577, 2012.

[17] M. Jureczko and L. Madeyski, "Towards identifying software project clusters 1054 with regard to defect prediction," in *Proceedings of the 6th International*, vol. 1055, p. 9.

[18] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the nasa software defect datasets," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.

[19] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 534–550, 2017.

[20] B. Yurdakul, *Statistical properties of population stability index*. Western Michigan University, 2018.

[21] A. Becker and J. Becker, "Dataset shift assessment measures in monitoring predictive models," *Procedia Computer Science*, vol. 192, pp. 3391–3402, 2021.

[22] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[23] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[24] Z. He, F. Peters, T. Menzies, and Y. Yang, "Learning from open-source projects: An empirical study on defect prediction," in *2013 ACM/IEEE international symposium on empirical software engineering and measurement*. IEEE, 2013, pp. 45–54.

[25] M. Long, J. Wang, G. Ding, W. Cheng, X. Zhang, and W. Wang, "Dual transfer learning," in *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012, pp. 540–551.

[26] V. R. Basili, F. Shull, and F. Lanubile, "Building knowledge through families of experiments," *IEEE Transac on Software Engineering*, 1999.