Preliminary Analysis of Review Method Selection Based on Bandit Algorithms

Takuto Kudo
Kindai University
Higashi-osaka, Japan
1910370079g@kindai.ac.jp

Masateru Tsunoda
Kindai University
Higashi-osaka, Japan
tsunoda@info.kindai.ac.jp

Amjed Tahir
Massey University
Palmerston North, New Zealand
a.tahir@massey.ac.nz

Kwabena Ebo Bennin
Wageningen UR
Wageningen, Netherlands
kwabena.bennin@wur.nl

Koji Toda
Fukuoka Institute of Technology
Fukuoka, Japan
toda@fit.ac.jp

Keitaro Nakasai
Kagoshima College, NIT
Kirishima, Japan
nakasai@kagoshima-ct.ac.jp

Akito Monden
Okayama University
Okayama, Japan
monden@okayama-u.ac.jp

Kenichi Matsumoto
NAIST
Ikoma, Japan
matumoto@is.naist.jp

***Abstract*—To enhance the reliability of software, it is important is to review all software artifacts (e.g., design documents) to remove defects as earlier as possible. There are various review methods available, and project managers face the challenge of choosing a suitable method for their current projects. One of approaches to support the selection of review methods is to evaluate review methods beforehand, to identify the most effective method on average. However, past studies have not evaluated review methods thoroughly as the process can be time-consuming. We propose a bandit-algorithm (BA) based method to evaluate and then dynamically select a suitable review method (from a list of candidates). In our experiments, we assume that the proposed method is applied to design document review on basic design phase. We performed experiments based on a simulation, instead of using an actual dataset. On our simulation, when a review method is selected by our BA method, productivity (i.e., total development time) was improved by about 1.25 times, and it was the second highest among candidates of review methods.**

***Keywords*—Multi-armed bandit problem, online optimization, design document review, performance comparison**

I. Introduction

Most infrastructures of our society contain or use software, thus there urgent need to enhance the reliability of software. Software review is one of the important activities to ensure the reliability of software systems [2][6][12][15][18][20]. On the review activity, instead of executing the program, a developer reads review targets such as design documents and source code in order to find and locate defects.

Review method is selected by a project manager or quality assurance team [9]. There are various review method such as check-list based method, peer review, pass around, walk through, and inspections [9]. Required effort and their impact is different among these methods. Therefore, project managers face the daunting task of selecting a review method which is applicable and feasible for the specific project. To inform this selection, past studies [12][15][20] evaluated review methods, and identified methods which can be effective. However, they compared only two or three methods, and the evaluation is very limited. Hence, it is not obvious for project managers which review method is the best.

The goal of our study is to select an appropriate review method without such subjective experiments. To achieve the goal, we propose review optimization method based on bandit algorithm (BA). Assume that there are two slot machines whose expected reward is unknown, and 100 coins to be bet. Most simplest approach is to bet all coins to a machine. Instead of that, BA repeatedly bets a coin to a machine, and when the reward is low, the other machine is selected. Intuitively speaking, based on BA, developers review each page of design documents by different methods, and identify the best one during the review activity. BA has been successfully applied to optimize defect prediction models in prior studies [1][7][13].

In recent year, modern code review (MCE) - a lightweight review method -[17] has attracted attention. However, conventional review are still widely used in industry. For instance, an ISO standard about review of software design document [8] was recently established. Additionally, institutions [10][14] have collected data about design document review. Therefore, review of design document is still important.

II. Related Work

Review method evaluation: Laitenberger et al. [12] compared checklist-based reading and perspective-based reading on reviewing object-oriented design documents using the unified modeling language (UML). The study performed a subjective experiment with 18 participants, and concluded that the latter method found more defects, and the cost was lower, compared with the former method. Porter et al. [15] compared scenario-based with ad hoc and checklist=based methods on the review of software requirements specifications. They performed subjective experiment, and used 18 professional software developers as the subjects. They concluded that defect detection rate of the scenario method was higher than other methods. Thelin et al. [20] focused on a user's point of view, and evaluated review methods by subjective experiment. In the experiment, they compared usage based and checklist based method. As a result, the former method found defects which related to the view effectively and efficiently, compared with the latter method.

In spite of the past studies, it is not perfectly clear which review method is the best. Past studies evaluated review methods based on subjective experiments. Such experiments are known to be time consuming, and therefore, it is difficult to compare many review methods simultaneously. Hence, past studies compared two or three methods. The number of comparison is considerably small, compared with other studies which compared methods without subjective experiment. For instance, a study which evaluated clone detection methods compared 30 methods [16]. Therefore, supporting review method selection is needed.

In our previous work, we applied BA for the selection of defect prediction models [1][7][13]. In [7], we used BA to select defect prediction model from four candidates, and BA dynamically selected from the candidates using Epsilon-greed. As a result, the prediction accuracy was the best or the second-best on three datasets. In [1], we used BA for cross-project defect prediction (CPDP). CPDP means a model is trained using data obtained from external projects. The study used BA to select the most suitable training project from a set of projects. In the experiment BA attained a higher accuracy than four baseline methods, on average. In [13], we used BA to select a suitable feature reduction technique, when building a defect prediction model. In the experiment, they used four candidates of reduction techniques. As a result, the prediction accuracy of BA was higher or equivalent than existing approaches on average.

Based on the successful results from applying BA for defect prediction, BA is expected to be effective when applied for review method selection. However, those previous studies evaluated BA to select defect prediction methods, and only one factor, prediction accuracy (e.g., AUC) was considered. While to evaluate BA on the review methods selection, both required review time (i.e., cost) and reduced time by finding defects should (i.e., quality of work) be considered, as explained in Section 4.

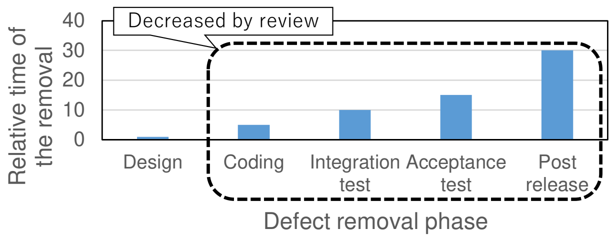

Fig. 1. **Relationship between defect removal cost and phase [19].**

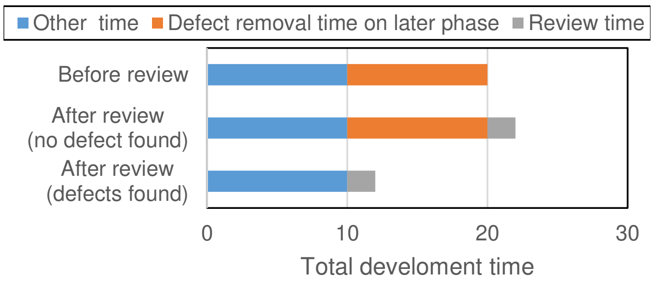

Fig. 2. **An example of cost and reward by review.**

TABLE I. AN EXAMPLE OF PROCEDURE BY BA-BASED SELECTION

BA for review method selection					
Reviewed page no.	**Review method**	**Total time**	**Defects**	**Avg. (cost + reward) of χ**	**Avg. (cost + reward) of ω**
Ordinal BA					
Number of play	**Slot machine**	**Total reward**	**Reward**	**Avg. reward of χ**	**Avg. reward of ω**
1	χ	1	No	1	0
2	ω	-7	Yes	1	-8
3	ω	-5	No	1	-3

III. BANDIT ALGORITHMS

BAs are proposed to solve multi-armed bandit problems. Those type of problems are often explained through an analogy with slot machines. Assume that a player has 100 coins to bet on several slot machines, and the player wants to maximize their reward. Instead of selecting only one slot machine and betting all 100 coins, BA suggests that a player bet only one coin on each slot machine. By calculating average reward of each machine after each betting, the player can then recognize which slot machine is the best (i.e., the highest average reward). The derivation of the problem is that each slot machine has an arm, and the arm is compared to a bandit who steals money from players. Multi-armed bandit problem seeks sequentially best candidates (they are referred to as arms) whose expected rewards are unknown, to maximize total rewards.

Epsilon-greedy Algorithm: This algorithm chooses a random arm with probability epsilon. That is, it selects an arm whose average reward is the highest among arms with the probability 1 - ε ($0 \leq \varepsilon \leq 1$). When the value of ε is 0, arms are always selected based on the average reward of each arm. In contrast, when the value of ε is 1, arms are always selected randomly.

Upper Confidence Bounds: UCB algorithm [3] focus on not only average reward of each arm, but also the amount of information about each arm. For instance, when an arm is selected only once, we do not have enough information about the arm, and it is not clear whether the average of reward is valid or not. After the arm is selected repeatedly, we can know the valid average. UCB positively selects such less-information arms because they might have high average reward. To do that, UCB selects the arm where the value of r in the following equation is the highest:

$$r = x + \sqrt{\frac{2\ln t}{s}} \qquad (1)$$

In the equation, x denotes average reward of the arm, s is the number of times the arm is selected, and t is the number of total trials. The right most term denotes the amount of information of the arm. When s is small, the value r is large, and the arm is selected with a high probability.

Thompson Sampling: Thompson Sampling (TS) samples a value from the beta distribution for each arm - an arm is then selected when the value is the maximum [5]. The beta distribution is the distribution of the probability of “success” on a Bernoulli trial, and the distribution has parameters α and β. Assuming that “wins” occurs m times, and “losses” occurs n times on the trial, the estimated probability of the occurrence of "wins" will follow the beta distribution where α is $m + 1$ and β is $n + 1$. Using TS, the expected reward r is calculated as:

$$r = \text{random.beta}\ (m + 1, n + 1) \qquad (2)$$

where random.beta samples a value from beta distribution with $\alpha = m + 1$ and $\beta = n + 1$, m is the number of wins, and n is the number of losses. As a result, the expected reward of an arm increases with the increase in the number of wins and the decrease in the number of losses.

IV. Review Method Selection

Overview: Previous studies [4][11][19] indicated that cost to remove defects increases as software development advances. That is, the advantage of software review is that time to remove defects is suppressed by finding defects during early stages of the development, as shown in Figure 1. The disadvantage is that review requires additional effort (i.e., working hours). To apply BA to review method selection, we related review activities to the metaphor of slot machine of BA as follows:

- **One play on a slot machine** (needs one coin): reviewing one page of documents (needs certain working hours).
- **Reward of slot machine**: reducing development time by finding defects (i.e., time enclosed by a box in Figure 1).
- **Slot machine** (winning probability): the probability of finding defects are different with different review methods.

Cost and reward: We set cost and reward, based on the following assumptions:

- **Goal**: Shortening total development time.
- **Cost**: The total time is increased by the review time, when a page of the documents is reviewed by a method.
- **Reward**: The total time is decreased by finding defects, which are found with a certain probability by the method.

Assume that required review time is two hours, decreased time by finding defects (i.e., defect removal time on later phase) is 10 hours, and probability of finding defects is 30%. In this case, after reviewing one page, the total development time changes as follows (see Figure 2):

- **No defect found** (P = 70%): Total time += 2 hours
- **Defects found** (P = 30%): Total time += (2 - 10 hours)

Example of the procedure: Table I shows an example of the proposed BA-based selection method. We assume that review methods χ and ω are used, and required time for the review is one hour and two hours respectively.

1. Method χ is randomly selected, and the first page is reviewed. As a result, no defect is found, and the total working time increases by 1 hour.
2. Method ω is selected based on average working time, and the second page is reviewed. As a result, defects are found, and the total working time decreases by 8 hour (i.e., -10 + 2).
3. Method ω is still selected, and the third page is reviewed. As a result, no defect is found, and the total working time increases by 2 hours.

Parameter settings: To apply BA-based selection to actual software review, some of parameters show in Figure 3 and Table II should be settled. Table II shows the required actions and configurations of each parameter. The details of the parameters are explained in the next section.

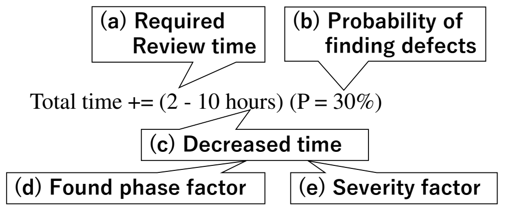

Fig. 3. **Paramaters for BA-based selection.**

TABLE II. Required actions and recognizability of parameters

Item	Parameter	Required action	Recognizable
(a)	Required Review time	Measured during test	After review
(b)	Probability of finding defects	Not required	After review
(c)	Decreased time	Estimated beforehand	No
(d)	Found phase factor	Estimated beforehand	No
(e)	Severity factor	Estimated beforehand	No
(f)	Number of trials	Not required	Before review

TABLE III. Required time (a) and probability of finding (b) on each method (based on [10])

Review method		α	β	γ	δ
Required time	**Value**	0.46	0.61	0.92	1.83
	Scale	0.75	-	1.50	2.00
Probability of finding	**Value**	0.14	0.28	0.35	0.52
	Scale	0.50	-	1.25	1.50

TABLE IV. Probability of finding defects and relative time to remove them for found phase factor (d) [19]

Phase	Coding	Integration test	Acceptance test	Post release
Probability of finding	0.16	0.53	0.09	0.23
Relative increased time	5	10	15	30

TABLE V. Probability of finding defects and time to remove them for severity factor (e) (based on [14][19])

Severity	Low	Medium	High
Probability of finding	0.490	0.355	0.155
Removal time	1	**2**	6

Additional procedure: BA-based selection needs additional procedure to apply actual software projects as follows:

- Switch review methods.
- Record the number of identified defects and the review time.
- Select a review method based on BA.

Reviewers are needed to switch review methods based on BA. However, most of review methods are a sort of guidelines, and therefore, it does not needs much effort for developers to accustom and switch the methods. The number of found defects and review time are recorded on most projects where quantitative management is conducted. Therefore, that does not

TABLE VI. TOTAL REDUCED TIME OF REVIEW METHODW AND RANDOM SELECTION

Review method	α	β	γ	δ	Random selection
Average	96.2	214.4	262.9	371.1	**236.1**
Median	99.2	203.1	238.4	379.3	**215.1**
Standard deviation	55.3	86.7	110.5	87.1	**131.0**

TABLE VII. TOTAL REDUCED TIME OF BA-BASED SELECTION

BA method	ε = 0	ε = 0.1	ε = 0.2	ε = 0.3	UCB	TS
Average	264.9	274.2	289.7	276.4	298.5	268.7
Median	284.8	287.8	280.7	250.3	299.8	262.5
Standard deviation	133.4	115.5	127.9	108.7	120.8	127.8

need additional cost on such projects. Selecting review method based on BA does not require additional effort for reviewers, once brief system for BA-based selection is made.

V. EXPERIMENT

Overview: In the experiment, we assumed that BA-based selection is applied to the review of design documents on the normal software design phase. The number of applied review method is four, and we named them *α*, *β*, *γ*, and *δ*. Note that the methods are imaginary ones for the experiment.

Instead of using actual dataset, we evaluated BA-based method by simulation. This is because we are unable to identify an actual dataset of design document review which can be used for this experiment. To set the parameters shown in Table II, we used the setting noted in [10][14][19] which shows summary statistics of dataset collected from software development companies We have done so to avoid setting parameters that make BA-based selection better intentionally. Based on the parameter settings, we generated artificial review dataset for the experiment. We made our replication package available online[1].

As the baseline for the evaluation, we selected one of review methods randomly. We repeated the experimental procedure 20 times, because ε-greedy selects arms randomly with the probability ε, and that could affect the results.

Performance of BA-based selection is evaluated based on total reduced time by the review. The time is calculated by the following equation:

$$\text{Total reduced time} = \text{reduced time by finding defects} - \text{required review time} \quad (3)$$

As explained in the previous section, the total reduced time is regarded as *total reward*, reduced time by finding defects earlier is regarded as *reward*, and required review time is regarded as *cost*. Therefore, when the reduced time is large, the method is regarded as effective to enhance productivity of the development project.

Required time and probability of finding: Required review time (item (a)) and probability of finding defects (item (b)) are different for each review method. In the simulation, we set them as shown in Table III. On review method *β*, we used the median of study [10] (gray shaded cells in Table III).

[1] https://zenodo.org/record/7246225

On the other methods, we multiplied the median of required time by *n*. On review method *α*, *γ*, and *δ*, we set *n* as 0.75, 1.5 and 2.0, respectively. When review time is long, the probability of finding defects is expected to be high. However, we assumed that the probability is not improved linearly. We multiplied the median of probability by *m*. On review method *α*, *γ*, and *δ*, we set *m* as 0.5, 1.25 and 1.5, respectively.

Values in the Table are included in interquartile range shown in [10]. Therefore, we considered the settings to be valid. Note that, this setting is not needed to apply BA-based selection. Instead of the setting, actual time of review should be measured.

Decreased time: Decreased time by review (item (c)) is same as increased time without review. The increased time is based on found phase factor (item (d)) and severity factor (item (e)). Therefore, decreased time was calculated by multiplying the two factors.

Defect identification phase factor: Table IV shows the probability of finding defects and relative time to remove them to settle item (d). Table IV is based on the numerical examples shown in study [19]. The value of the factor was selected based on the probability shown on the table.

Severity factor: Table V shows probability of finding defects and time to remove them to settle item (e). As probability of defect severity, we used the rate of defect severity after software release (within three months) shown in [14] (gray shaded cells Table V). As removing time for the medium severity, we used the median shown in [19] (bold number in the table). For other severities, we multiplied the median by *n*. On low and high severity, we set *n* as 0.5 and 3.0 respectively. The value of the factor was selected based on the probability shown on the table.

Number of trials: To evaluate BA, we need to set the number of trials (i.e., item (f)). Based on [14], the median of the page count of design documents was 160. The median of maximum team size was five on basic design phase [14], and we assumed that most of the team members review documents. Therefore, we assumed that each developer reviews 40 pages (i.e., page count / number of reviewers = 160 / 4), and set the number of trials to 40.

VI. RESULT

Conventional approach: Table VI shows descriptive statistics of the total reduced time of each review method and the random selection. The reduced time was the largest, when both required review time and probability of finding defects were the highest among methods (i.e., method δ). Therefore, if we select such method on actual development, we might achieve the highest productivity. However, as shown in Table II, they are not recognizable before review, and hence we should select a method from candidates randomly. The total reduced time of the random selection is the average of the total time of all methods.

Review method selection by BA: Table VII shows the total reduced time of each BA method. All of the reduced time was larger than random selection shown in Table VII. In BA methods, the average and the median of UCB was the largest. Specifically, when we compared UCB with the random selection, the median and the average were improved over 1.25 times. When we compared with each review method, UCB was the second highest among methods.

Therefore, in the simulation with our parameter settings, using BA-based selection with UCB, the total reduced time (i.e., productivity of development project) is greatly improved, compared with random selection. Also, it attains the second highest productivity among review methods. In actual development, reviewers might implicitly know which method tends to be better, as shown in Table III. However, unless the best method can be selected by the knowledge, it is better to apply BA-based selection with UCB, which can select the second best.

VII. Conclusion

In this study, we focused on review methods on software development, and proposed a new BA-based selection method to select an effective review method. To support the method selection, previous studies have tried to identify the best review methods based on subjective experiments. However, such evaluation is time-consuming without providing much clarity. However, BA-based selection evaluates review methods during review activity, and dynamically selects the best one. In the experiment, we performed the simulation based on actual software development dataset, to evaluate BA-based selection. For the BA, we used Epsilon-greedy, UCB, and Thompson sampling. We used total reduced time by the review (i.e., review cost - reduced time by finding defects). We prepared four candidates of review methods. As a result, we observed the followings:

- UCB was the highest performance among BA.
- UCB receded total development time by about 1.25 times, compared with random method selection.
- The effect of UCB was the second highest among review method candidates.

As future work, we will perform subjective experiment, and evaluate the extent in which the total time is reduced by BA-based selection.

Acknowledgment

This research is partially supported by the Japan Society for the Promotion of Science [Grants-in-Aid for Scientific Research (C) and (S) (No.21K11840 and No. 20H05706).

References

[1] T. Asano, M. Tsunoda, K. Toda, A. Tahir, K. Bennin, K. Nakasai, A. Monden, and K. Matsumoto, “Using Bandit Algorithms for Project Selection in Cross-Project Defect Prediction,” Proc. of International Conference on Software Maintenance and Evolution (ICSME 2021), pp.649-653, 2021.

[2] A. Aurum, H. Petersson, and C. Wohlin, “State-of-the-art: software inspections after 25 years,” Journal of Software: Testing, Verification and Reliability, vol.12, no.3, pp.133-154, 2002.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time Analysis of the Multiarmed Bandit Problem,” Machine Learning, vol.47, pp.235-256, 2002.

[4] B. Boehm, and V. Basili, "Software Defect Reduction Top 10 List," IEEE Computer, vol.34, no.1, pp.135–137, 2001.

[5] O. Chapelle and L. Li, “An empirical evaluation of thompson sampling,” Proc. of International Conference on Neural Information Processing Systems (NIPS), pp.2249-2257, 2011.

[6] M. Ciolkowski, O. Laitenberger, and S. Biffl, “Software reviews, the state of the practice,” IEEE Software, vol.20, no.6, pp.46-51, 2003.

[7] T. Hayakawa, M. Tsunoda, K. Toda, K. Nakasai, A. Tahir, K. Bennin, A. Monden, and K. Matsumoto, “A Novel Approach to Address External Validity Issues in Fault Prediction Using Bandit Algorithms," IEICE Transactions on Information and Systems, vol.E104.D, no.2, pp.327-331, 2021.

[8] International Organization for Standardization (ISO), Software and systems engineering — Work product reviews, ISO/IEC 20246:2017, ISO, 2017.

[9] Information-technology Promotion Agency (IPA), Guidebook of devemopment methods to achieve high reliablility software, IPA, 2011 (in Japanese).

[10] Information-technology Promotion Agency (IPA), The 2018-2019 White Paper on Software Development Projects, IPA, 2018 (in Japanese).

[11] C. Jones, Applied Software Measurement: Global Analysis of Productivity and Quality, McGraw-Hill, 2008.

[12] O. Laitenberger, C. Atkinson, M. Schlich, K. Emam, "An experimental comparison of reading techniques for defect detection in UML design documents," Journal of Systems and Software, vol.53, no.2, pp.183-204, 2000.

[13] M. Tsunoda, A. Monden, K. Toda, A. Tahir, K. Bennin, K. Nakasai, M. Nagura, and K. Matsumoto, “Using Bandit Algorithms for Selecting Feature Reduction Techniques in Software Defect Prediction,” Proc. of Mining Software Repositories Conference (MSR 2022), pp.670-681, 2022.

[14] S. Ohiwa, T. Oshino, and S. Nakai, Analysis of Software Project Data Repository, Economic Research Association, 2020 (in Japanese).

[15] A. Porter, and L. Votta, “Comparing Detection Methods For Software Requirements Inspections: A Replication Using Professional Subjects,” Empirical Software Engineering, vol.3, pp.355-379, 1998.

[16] C. Ragkhitwetsagul, J. Krinke, and D. Clark, “A comparison of code similarity analysers,” Empirical Software Engineering, vol.23, pp.2464-2519, 2018.

[17] C. Sadowski, E. Söderberg, L. Church, M. Sipko and A. Bacchelli, “Modern Code Review: A Case Study at Google,” Proc. of International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), pp.181-190, 2018.

[18] F. Shull, I. Rus, and V. Basili, “How perspective-based reading can improve requirements inspections,” IEEE Computer, vol.33, no.7, pp.73-79, 2000.

[19] G. Tassey, The Economic Impacts of Inadequate Infrastructure for Software Testing, National Institute of Standards and Technology, 2002.

[20] T. Thelin, P. Runeson, and C. Wohlin, “An experimental comparison of usage-based and checklist-based reading,” IEEE Transactions on Software Engineering, vol.29, no.8, pp.687-704, 2003.