

Test-Driven Code Review: An Empirical Study

Davide Spadini, Fabio Palomba, Tobias Baum,
Stefan Hanenberg, Magiel Bruntink, Alberto Bacchelli

Test-Driven Code Review(TDR)とは

- プロダクションコードより先にテストコードを検証するコードレビュー方法である
- 実務者や非公式な文献から言及されているが体系的にまとめられた研究は存在していない
- 本研究ではTDRがコードレビューに与える影響を調査する

リサーチクエスチョン

- RQ1 : レビューワにテストコードを見せる順序はコードレビューの有効性に影響を与えるか？
 - 93人の開発者を対象に実験
- RQ2 : 開発者はTDRの実践をどのように認識しているか？
 - 103人を対象にインタビュー

RQ1 - 使用ツールについて

- ブラウザベースで実験
- コードレビューを視覚化
- 操作ログなどを集計

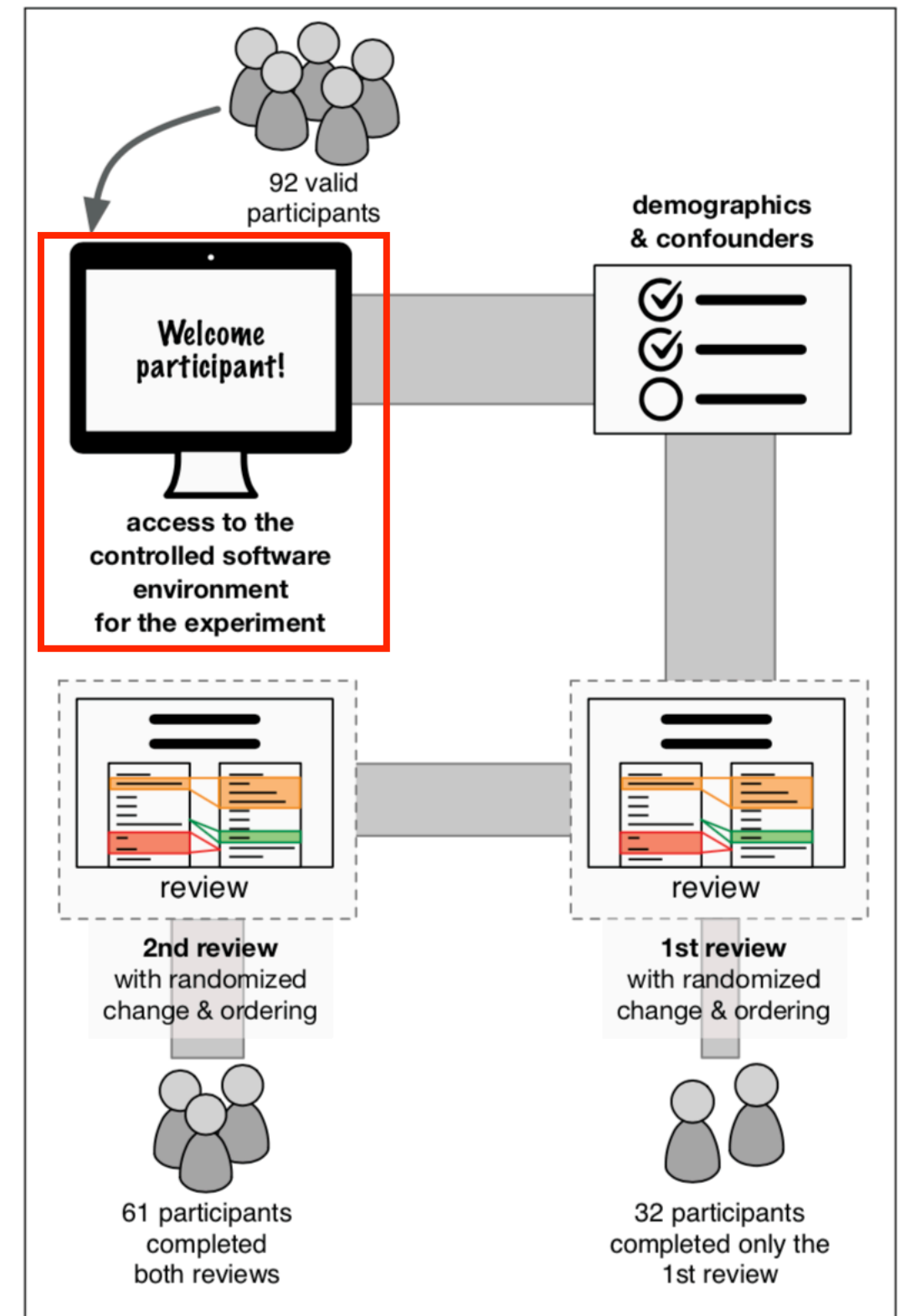


Figure 1. Experiment steps, flow, and participation

RQ1 - 事前調査

- 以下の項目を調査
- 参加者の役割
- Javaの経験
- プログラミングの慣習
- 実験日のこれまでの活動時間

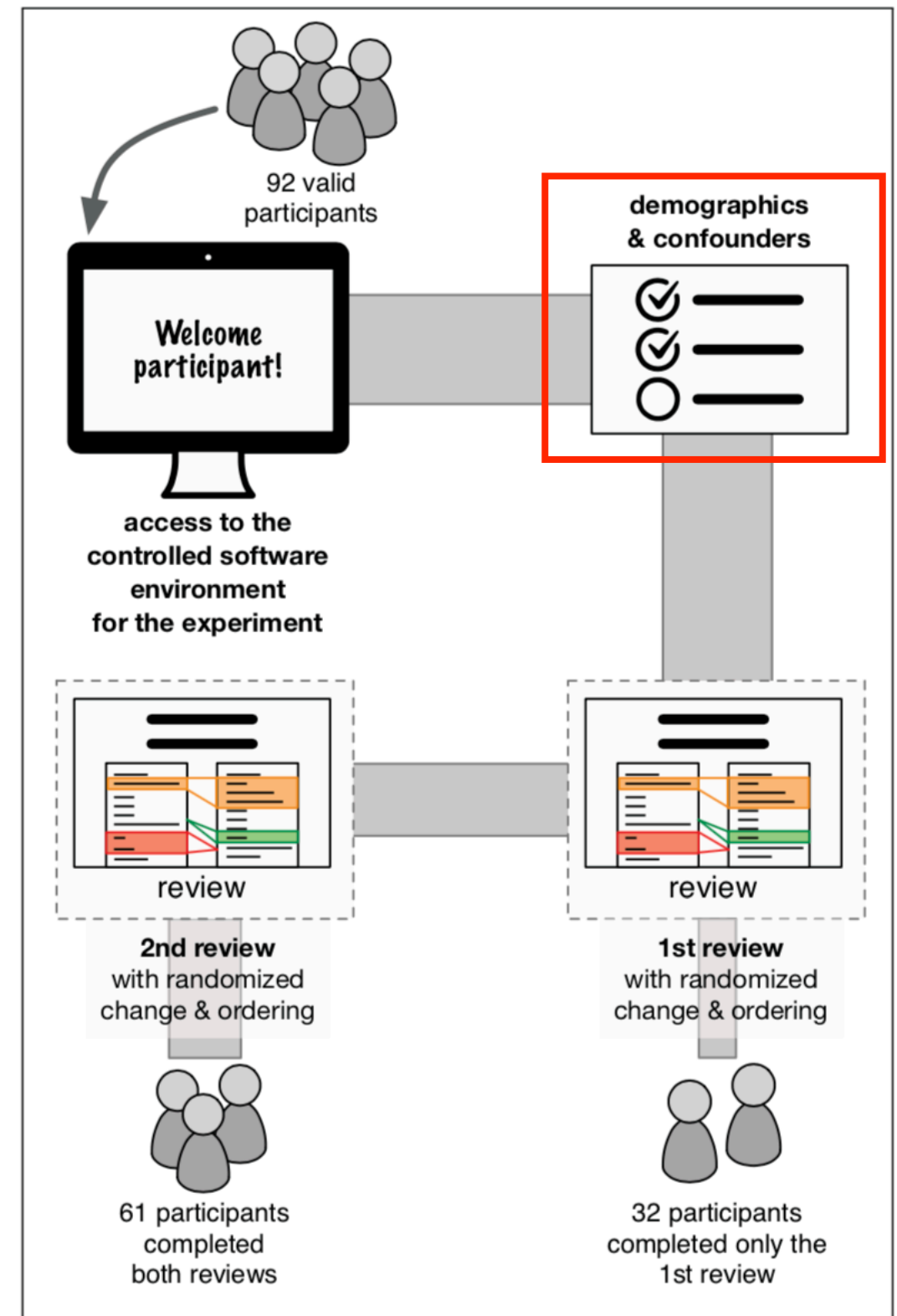


Figure 1. Experiment steps, flow, and participation

RQ1 - レビュー方法

- 3つのレビュー方法で調査
- Test-First
- Production-First
- Only-Production

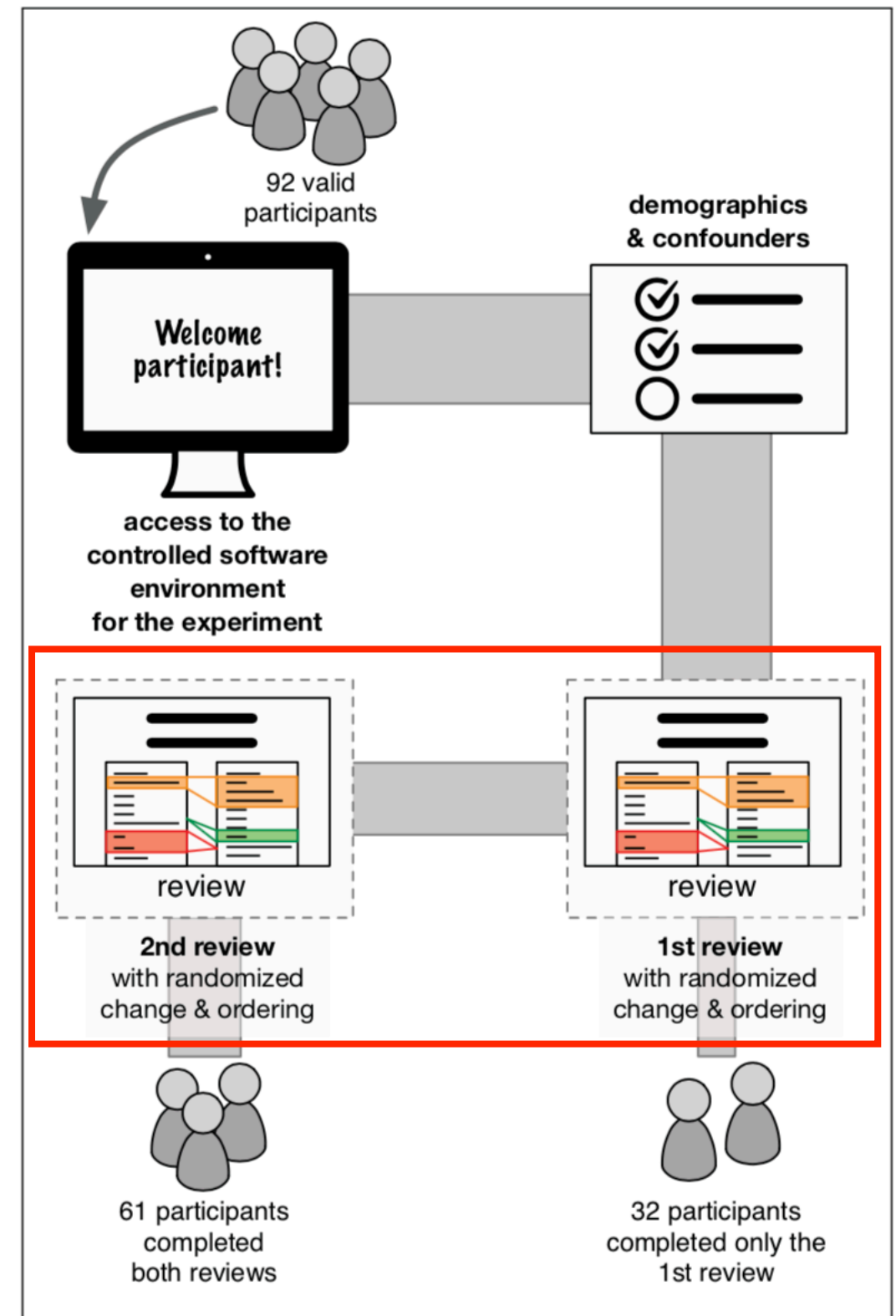


Figure 1. Experiment steps, flow, and participation

RQ1 - パッチについて

- JPACMAN-FRAMEWORKから以下の条件を満たす変更
をレビュー対象とする
 - 自己完結している
 - プロダクション, テスト両方を含む
 - 簡単すぎず, 難しすぎず
 - 疑わしい変更を含まない
- 2つのパッチが該当
 - 4 bugs and 1 issues, 5 bug and 3 issues

RQ1 - 統計量について

- レビュー方法を
中心にメトリクス化
- 各Propに対して
ロジスティック回帰

Table I
VARIABLES USED IN THE STATISTICAL MODEL

Metric	Description
<i>Dependent Variables</i>	
ProdBugsProp	Proportion of functional defects found in the <i>production</i> code
ProdMaintIssuesProp	Proportion of maintainability issues found in the <i>production</i> code
TestBugsProp	Proportion of functional defects found in the <i>test</i> code
TestMaintIssuesProp	Proportion of maintainability issues found in the <i>test</i> code
<i>Independent Variable</i>	
Treatment	Type of the treatment (TF, PF, or OP)
<i>Control Variables</i>	
<i>Review Details</i>	
TotalDuration	Time spent in reviewing the code
IsFirstReview	Boolean representing whether the review is the first or the second
Patch	Patch 1 or 2
<i>Profile</i>	
Role	Role(†) of the participant
ReviewPractice	How often(†) they perform code review
ProgramPractice	How often(†) they program
ProfDevExp	Years of experience(†) as professional developer
JavaExp	Years of experience(†) in Java
WorkedHours	Hours the participant worked before performing the experiment

(†) see Table III for the scale

RQ2 - インタビューの参加者

- 以下の表の開発者
+ ソーシャルメディアなどを通じて回答者を募集
- コードレビューに関する一般的な質問後,TDRに関する質問

Table II
INTERVIEWEES' EXPERIENCE (IN YEARS) AND WORKING CONTEXT

ID	Developer	Reviewer	Working context	Applying TDR
P1	8	8	OSS	Almost never
P2	3	3	Company A	Almost Always
P3	15	15	Company A	Almost Always
P4	10	10	Company B	Almost Never
P5	10	5	Company C	Always
P6	3	2	Company D	Sometimes
P7	16	16	Company E	Always
P8	4	4	Company F / OSS	Sometimes
P9	3	3	Company G / OSS	Never

RQ1 - 参加者について

- コメントのないレビューや時間をかけすぎているレビュー（異常値レビュー）を取り除く

Table III

PARTICIPANTS' CHARACTERISTICS – DESCRIPTIVE STATISTICS - N. 92

	Dev	Student	Researcher	Architect	Analyst	Other
Current role	61% (56)	16% (15)	12% (11)	5% (5)	3% (3)	2% (2)
Experience (years) with	None	<= 1	2	3-5	6-10	>10
- Java prog.	13% (12)	5% (5)	7% (6)	21% (19)	34% (31)	21% (19)
- Profess. dev.	5% (5)	11% (10)	13% (12)	18% (17)	28% (26)	24% (22)
Current frequency of	Never	Yearly	Monthly	Weekly	Daily	
- Programming	0% (0)	0% (0)	3% (3)	17% (16)	79% (73)	
- Reviewing	15% (14)	7% (6)	16% (15)	22% (20)	40% (37)	

Table IV

DISTRIBUTION OF PARTICIPANTS' REVIEWS ACROSS TREATMENTS

	TF	PF	OP	total
Patch1	31	32	29	92
Patch2	28	29	34	91
total	59	60	63	

RQ1 : 実験結果

- レビューワーにテストコードを見せる順序はコードレビューの有効性に影響を与えるか？
- 顕著な違いは以下の2つ
 - プロダクションに含まれる保守性の問題の割合
 - テストに含まれる不具合の割合

Table V

AVERAGE PROPORTION OF BUGS AND ISSUES FOUND, BY TREATMENT, AND REVIEW TIME. COLORED COLUMNS INDICATE A STATISTICALLY SIGNIFICANT DIFFERENCE BETWEEN THE TREATMENTS ($p < 0.01$), WITH THE COLOR INTENSITY INDICATING THE DIRECTION.

	Proportion of found				Time
	production		test		
	bugs	maintIssues	bugs	maintIssues	
TF	0.28	0.08	0.40	0.18	7m11s
PF	0.33	0.21	0.17	0.13	6m27s
OP	0.28	0.18			5m29s

RQ1 - 実験結果

- Test-Firstは"TestBugsProp", "ProdMaintIssuesProp"の両方で統計的に有意と判明した

Table VI
REGRESSIONS FOR 'PRODMAINTISSUESPROP' AND 'TESTBUGSPROP'

	TestBugsProp			ProdMaintIssuesProp		
	Estimate	S.E.	Sig.	Estimate	S.E.	Sig.
Intercept	-0.7314	1.9242		-0.0471	1.2191	
TotalDuration	0.1664	0.0618	**	0.0462	0.0259	.
IsFirstReview 'TRUE'	-0.9213	0.5344	.	-0.1554	0.2241	
Patch 'P2'	1.9296	0.5688	***	-2.8474	0.4579	***
Treatment 'PF'				0.0975	0.2386	
Treatment 'TF'	1.1792	0.4639	**	-1.2468	0.3908	**
ReviewPractice	0.0675	0.2082		0.2598	0.1389	.
ProgramPractice	-0.6608	0.4685		-0.2180	0.2938	
ProfDevExp	-0.0982	0.1951		-0.2953	0.1211	*
JavaExp	-0.0182	0.1512		0.0366	0.0631	
WorkedHours	-0.0225	0.0817		0.0403	0.0405	
... (†)						

significance codes: '***' $p < 0.001$, '**' $p < 0.01$, '*' $p < 0.05$, '.' $p < 0.1$
(†) Role is not significant and omitted for space reason

RQ1 - レビューの質の調査

- レビューの質を評価するため、外部から2人の評価者を用意
- 1 から 5 で評価し、Cohen's kappaで一致率を調査
 - production issues ($\kappa = .69$)
 - test issues ($\kappa = .78$)
 - production bugs ($\kappa = 1$)
 - test bugs ($\kappa = .77$)
- "レビュー方法間で品質に違いはない"と判断

RQ2:TDRの短所

- テストはそれほど重要でないと認識されている
 - "プロダクションの変更を先に知りたい"
- テストはプロダクションの振る舞いについてあまり教えてくれない
 - "まずプロダクションがどういう動作をするか知りたい"
- テストの品質が低い
- レビューツールでサポートされていない

RQ2:TDRの長所

- **ブラックボックスビューを与えてくれる**
 - "プロダクションの振る舞いの概要を教えてくれる(ドキュメントのように使える)"
 - "複雑なアルゴリズムを検証するときに入出力の確認がすぐにできる"
- **テストの品質を上げる**
 - "テストが先に見られると分かると意識して書くようになる"

まとめ

- 実験とインタビューによりTDRを体系的に調査
- TDRにより質は下がらず、見つかるbugやissueの割合も変わらない
- TDRの適用に問題があると考えている開発者は多い