Zwinne eksperymenty w przemysłowym środowisku wytwarzania oprogramowania

na przykładzie praktyki programowania sterowanego ciągłymi testami

Marcin Kawalerowicz

Rozprawa doktorska

Promotor
dr hab. inż. Lech Madeyski, prof. uczelni

Wydział Informatyki i Telekomunikacji
Politechnika Wrocławska
2023

Streszczenie

Przeprowadzanie eksperymentów na projektach programistycznych w środowisku przemysłowym jest trudne i kosztowne. Cele interesariuszy nie muszą być zbieżne. Badacz może chcieć przeprowadzić kontrolowane eksperymenty, których efektem będą wiarygodne wnioski prowadzące do opracowania metod usprawniających proces wytwarzania oprogramowania. Właściciel projektu jest zainteresowany zwrotem z inwestycji, przy zastosowaniu nowych metod, przy możliwie najniższych kosztach ewaluacji nowych metod. Programista jest zainteresowany produkcją kodu wysokiej jakości z minimalnym nakładem pracy. Te, nie do końca zbieżne, cele mogą być zintegrowane przez proponowaną metodę zwinnego eksperymentowania w przemysłowym środowisku wytwarzania oprogramowania. Zwinne eksperymentowanie nie powinno wpływać na przebieg projektu, ale powinno pozwalać na efektywną pracę badawczą w czasie jego trwania.

Programowanie Sterowane Ciągłymi Testami (ang. *Continuous Test-Driven Development (CTDD)*) oraz Ciągła Predykcja Defektów (ang. *Continuous Defect Prediction (CDP)*) to dwie nowatorskie praktyki inżynierii oprogramowania zaproponowane w ramach niniejszej rozprawy. CTDD łączy ciągłe testowanie z programowaniem sterowanym testami. CDP jest kombinacją predykcji defektów oprogramowania i ciągłego informacyjnego sprzężenia zwrotnego do programisty. CDP zostało zawężone do Ciągłej Predykcji Wyników Budowy Oprgoramowania (ang. *Continuous Build Outcome Prediction (CBOP)*), która ma na celu przewidywanie wyników budowy i testowania na serwerze ciągłej integracji.

Celem niniejszej pracy jest opracowanie ram metody zwanej Agile Experimentation dla minimalnie inwazyjnego eksperymentowania w środowisku przemysłowego wytwarzania oprogramowania, która może być zastosowana w komercyjnych projektach wytwarzania oprogramowania, co pokazano na przykładach wprowadzania praktyk CTDD i CBOP.

W celu oceny CTDD i CBOP przeprowadzono serię eksperymentów typu single case (z jednym uczestniczącym programistą) oraz small-n (z niewielką liczbą uczestniczących programistów). Eksperymenty zostały zaplanowane przy użyciu szablonu GQM (goal, question, metric). Cel eksperymentu CTDD został zdefiniowany następująco:

> *Analiza* praktyki CTDD
> *w celu* jej ewaluacji
> *w odniesieniu* do jej zdolności do redukcji czasu tworzenia kodu produkcyjnego (rozumianego jako czas od momentu w którym projekt jest w stanie "czerwonym"(nie kompiluje się, testy nie przechodzą etc.) do momentu gdy znajdzie się on znowu w stanie "zielonym"(kompiluje się i wszystkie testy przechodzą))
> *z punktu widzenia* badacza i kierownika projektu
> *w kontekście* profesjonalnego programisty w rzeczywistym (przemysłowym) projekcie (projektach) tworzenia oprogramowania.

Cel eksperymentu CBOP został zdefiniowany w następujący sposób:

> *Analiza* praktyki przewidywania wyników ciągłej budowy oprogramowania (CBOP)
> *w celu* dokonania oceny tej praktyki
> *w odniesieniu* do jej zdolności w zmniejszaniu liczby nieudanych procesów budowy oprogramowania

z punktu widzenia badacza i kierownika projektu
w kontekście małej grupy profesjonalnych programistów pracujących w rzeczywistym (przemysłowym) projekcie tworzenia oprogramowania.

Definicje tych celów doprowadziły do sformułowania następujących pytań badawczych:

1. **RQ1** Czy praktyka CTDD zmniejsza czas tworzenia kodu produkcyjnego (rozumianego jako czas od momentu w którym projekt jest w stanie "czerwonym"(nie kompiluje się, testy nie przechodzą etc.) do momentu gdy znajdzie się on znowu w stanie "zielonym"(kompiluje się i wszystkie testy przechodzą))?
2. **RQ2** Czy praktyka CBOP zmniejsza liczbę nieudanych procesów budowy oprogramowania na serwerze Ciągłej Integracji (CI)?

W celu odpowiedzi na te pytania sformułowano następujące metryki:

1. czas programowania, o którym mowa w **RQ1** był mierzony za pomocą czasu RTG (Red-To-Green time) czyli czasu, który upływa od momentu kiedy oprogramowanie jest w „czerwonym" stanie (nie kompiluje się, któryś z testów nie kończy się sukcesem, itd.) do momentu gdy projekt przechodzi w stan „zielony" (kompilacja odbywa się prawidłowo, wszystkie testy kończą się sukcesem, itd.)
2. zdolność redukcji kończących się porażką procesów budowy, o której mowa w **RQ2** mierzona była za pomocą FBR (Failed Build Ratio), czyli stosunku liczby procesów budowy zakończonych porażką do wszystkich innych wyników budowy oprogramowania.

Odpowiedź na pierwsze pytanie badawcze RQ1 wydaje się być pozytywna. Zaobserwowano niewielki pozytywny wpływ stosowania praktyki CTDD w porównaniu z praktyką TDD w odniesieniu do czasu RTG.

Wyniki przeprowadzonych badań nie dają podstaw na do pozytywnej opdowiedzi na pytanie badawcze RQ2. Wbrew intuicyjnym oczekiwaniom, użycie CBOP wydaje się zwiększać FBR.

Badania nad CTDD i CBOP zostały przeprowadzone zgodnie z rygorami Agile Experimentation przy minimalnym wpływie na sam projekt biznesowy. Wyniki eksperymentów stanowią oryginalne rozwiązanie problemu naukowego i pozwalają na wyciągnięcie istotnych wniosków i zastosowanie ich w codziennej praktyce wytwarzania oprogramowania w sferze gospodarczej.

Mariusz Wykowska

12.06.2023

Using Agile Experimentation in an Industrial Software Development Environment

to study Continuous Test-Driven Development

Marcin Kawalerowicz

PhD dissertation

Supervisor
Lech Madeyski, PhD, DSc

Faculty of Information and Communication Technology
Wrocław University of Science and Technology
Poland
2023

Abstract

Background Conducting experiments on software projects in an industrial environment is difficult and expensive. The objectives of the stakeholders do not necessarily coincide. A researcher may want to conduct controlled experiments that result in reliable conclusions, leading to methods for improving the software development process. The project owner is mostly interested in a return on investment, thus using and evaluating the new methods at the lowest possible cost. The software developer is interested in producing high-quality code with minimal effort. Those, not entirely convergent, goals are addressed by a novel software engineering method called Agile Experimentation. Experiments conducted in an industrial software development environment according to this method should not affect the course of the project while allowing for effective research work during the project.

Continuous Test-Driven Development (CTDD) and Continuous Defect Prediction (CDP) are two novel software development practices proposed as a part of the work in this dissertation. CTDD combines Continuous Testing with Test-Driven Development. CDP is a combination of defect prediction and continuous feedback. CDP has been simplified to form a lightweight implementation named Continuous Build Outcome Prediction (CBOP), which aims to forecast the build and test results on the Continuous Integration server.

Aim The objective is to develop an Agile Experimentation method for minimally invasive experimentation in industrial software development environments that can be applied to commercial software development projects and to test CTDD and CBOP practices under its rigour.

Method A series of single-case (with a single participating software developer) and small-n (with a low number of participating software developers) experiments were performed to evaluate CTDD and CBOP. The experiments were scoped using the GQM (goal, question, metric) template. The CTDD experiment goal was defined as follows:

> *Analyze* the CTDD practice
> *for the purpose of* evaluation
> *with respect to* its ability to reduce the time of development of production code (understood here as the time that elapses from the moment when a software project enters a "red" state of failure (not compiling, unit tests failing, etc.) to the moment when the project is "green" once again (it compiles, and all the tests are passing))
> *from the point of view of the* researcher and project manager
> *in the context of* a professional software developer in a real-world (industrial) software development project.

The CBOP experiment goal was defined as follows:

> *Analyze* continuous build outcome prediction (CBOP) practice
> *for the purpose of* evaluation of the practice
> *with respect to* its ability to reduce the number of failing builds
> *from the point of view of the* researcher and project manager
> *in the context of* a small group of professional software developers working on an industrial-grade software project.

These goal definitions led to the formulation of the following research questions:

1. **RQ1** Does CTDD practice reduce the time of development of production code?
2. **RQ2** Does CBOP practice reduce the number of failing builds on the Continuous Integration (CI) server?

To answer those questions, the following metrics were formulated:

1. The ability to reduce the development time in **RQ1** was measured in Red-To-Green time (RTG) — the time that elapses from the moment when a software project enters a "red" state of failure (not compiling, unit tests failing, etc.) to the moment when the project is "green" once again (it compiles and all the unit tests are passing).
2. The ability to reduce the number of failing builds **RQ2** was measured in Failed Build Ratio (FBR) — the proportion of the failed builds to all other build results on the CI server.

Results The results of the experiments suggest a positive answer to **RQ1**. There is a small positive effect of using CTDD practice compared with the TDD practice regarding the RTG time – RTG is reduced. The answer to the second research question **RQ2** seems to be negative. Contrary to intuitive expectations, the usage of CBOP increases FBR.

Conclusions The research on CTDD and CBOP was conducted under the rigour of Agile Experimentation, with a minimal impact on the business-driven project in a real (i.e. industrial) software development environment. The results of the experiments constitute an original solution to a scientific problem and allow for meaningful conclusions to be drawn and applied to software development practice in the economic sphere.

12.06.2023