عنوان: **Test Driven Development**
نویسنده: شهروز جعفری
تاریخ: ۱۳۹۲/۰۳/۰۸ ۲۲:۱۰
آدرس: www.dotnettips.info
گروه‌ها: Unit testing, TDD

نوشتن تست برای نرم افزار امری ضروریست، چه پس از تولید نرم افزار چه در حین تولید، در کل به وسیله تست می‌توان از به وجود آمدن باگ‌ها در هنگام گسترش دادن برنامه تا حد قابل توجهی جلوگیری کرد.
از معروف ترین روش‌های تست می‌توان عناوین زیر را نام برد:

Unit test

Integration test

Smoke test

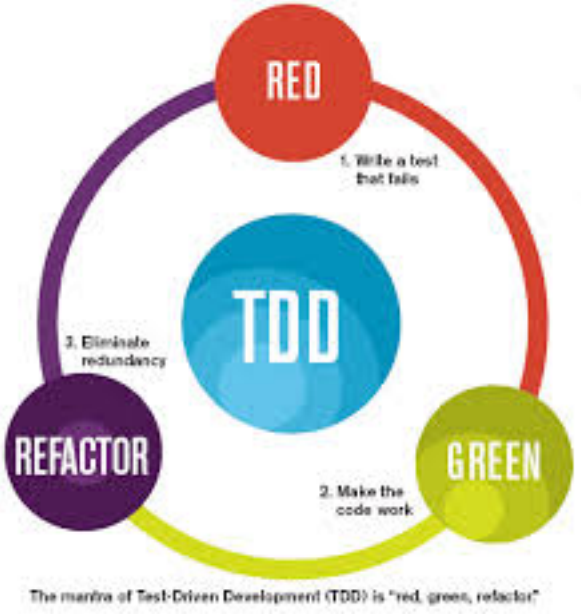
Regression test

Acceptance test

Test Driven Development
یک پروسه تولید نرم افزار است که برای اولین بار توسط Kent_Beck معرفی شد.
TDD شامل 4 مرحله کلی است:

نوشتن تست **قبل از نوشتن کد.**
کامپایل کردن کد و اطمینان از **Fail شدن** کامپایل.
پیاده سازی کد به طوری که تست ما پاس شود.
Refactoring

مراحل 4 گانه تست باید به صورت متناوب اجرا شوند.
البته بسیاری این 4 مورد را با عبارت red/green/ refactor نیز می‌شناسند.



همانطور که گفته شد در کل نوشتن تست باعث می‌شود که با اضافه شدن کدهای جدید در برنامه از به وجود آمدن باگ تا حدی جلوگیری شود.
اما مزایای TDD:

TDD باعث کاهش زمان تولید نرم افزار می‌شود.البته این حرف کمی عجیب است.(در ادامه بیشتر توضیح می‌دهم)

اعتماد شما نسبت به کد بالا می‌رود.
باگ کمتری تولید می‌شود بتابراین اعتماد مصرف کنندگان نیز نسبت به برنامه شما بالا می‌رود.
باعث نظم در کد می‌شود.
باعث انعطاف پذیری بیشتر در نرم افزار می‌شود.
ریسک تولید نرم افزار به علت باگ کمتر به حداقل می‌سد.
...

البته باید به این نکته نیز اشاره داشت که مایکروسافت تحقیقی انجام داده که بر طبق آن نوشتن کد به روش TDD می‌تواند 15 تا 30 در صد روند تولید نرم افزار را افزایش دهد ولی در عوض بین 40 تا 90 در صد می‌تواند از به وجود آمدن باگ جدید جلوگیری کند.
در بسیاری از محیطهای برنامه نویسی، نه تنها به این موضوع اهمیت داده نمی‌شود بلکه به طور کلی به اشتباه گرفته شده و حتی در پروژه هایی که تست نوشته می‌شود مفاهیم آن(که در بالا نام برده شده)جابجا شده و به اشتباه نام برده می‌شود.
هدف از نوشتن تست,تست کردن قطعات کوچک کد است,به عنوان مثال نباید تست به گونه ای باشد که یک متد با 300 خط کد را تحت پوشش قرار دهد.ابتدا باید کد به قطعات کوچک شکسته و بعد تست شود.
یک نمونه از متد تست:

```
[Test]
        public void TestFullName()
        {

            Person person = new Person ();
            person.lname = "Doe";
            person.mname = "Roe";
            person.fname = "John";

            string actual = person.GetFullName();
            string expected = "John Roe Doe";
            Assert.AreEqual(expected, actual, ”The GetFullName returned a different Value”);
        }
```

هدف از نوشتن این پست مقدمه ای بر شروع سری پست‌های TDD باستفاده از MVC.Net و فریم ورک قدرت مند تست Nunit است.

نظرات خوانندگان

نویسنده: رضا
تاریخ: ۱۳۹۲/۰۵/۲۰ ۰:۱۰

با سلام

با تشکر از مقاله خوبتون

خواستم ببینم پروژه وبی وجود داره که در اون قسمت‌های مختلف سایت رو با انواع تست‌های مختلف پیاده سازی کرده باشه (یا حداقل با روش unit test)؟

من از unit test استفاده می‌کنم ولی یه جورایی توش سر در گمم (تست‌ها رو می‌نویسم و عملکردش هم قابل قبوله ولی یه جورایی کدها خیلی بی نظم و بهم ریخته است)

نویسنده: حسینی
تاریخ: ۱۳۹۳/۰۲/۱۰ ۲۱:۴

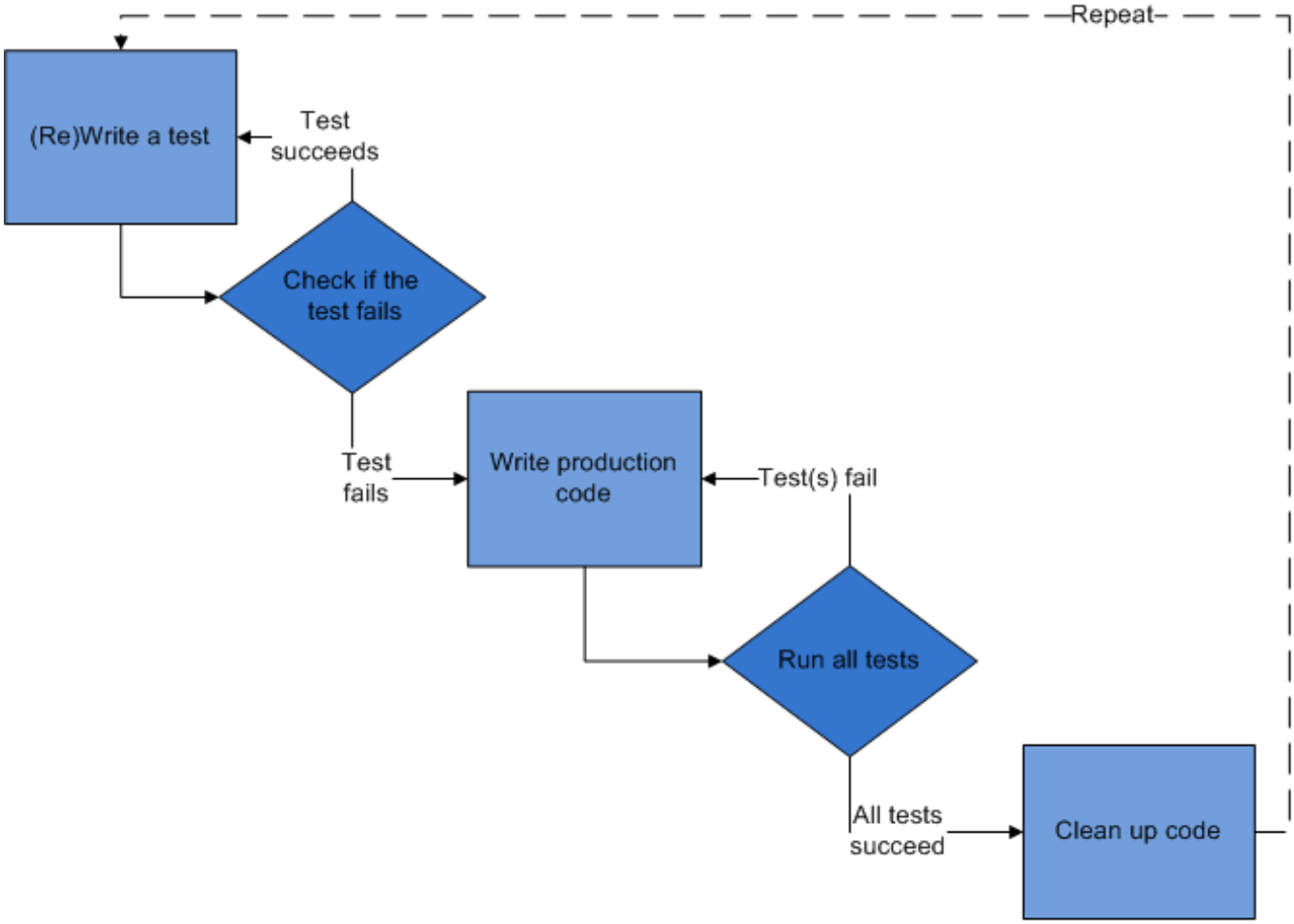
سلام

تفاوت TDD با unit testing چیه؟

همون مباحثی که برای tdd مطرح هست برای unit test هم مطرح میشه من تفاوت این 2 رو متوجه نمیشم.

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۳/۰۲/۱۱ ۱۱:۸

آزمون واحد بر می‌گردد به آنچه شما تست می‌کنید و TDD اشاره دارد به زمانی که تست می‌کنید،
در واقع فرض کنید برنامه‌ی ماشین حساب را توسعه داده اید، اکنون برای عملگر جمع تست می‌نویسید، این Unit Test هست. در TDD ، آزمون واحد شما توسعه و طراحی را پیش می‌برد، اگر مقالات مربوطه به TDD را مطالعه کنید، در TDD ابتدا بدون پیدا سازی هیچ ویژگی تست نوشته می‌شود.



به تصویر بالا توجه کنید، ابتدا تست نوشته شده، سپس کد محصول نوشته می‌شود..