عنوان: **آشنایی با آزمایش واحد (unit testing) در دات نت، قسمت 2**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۸ ۱۳:۴۶:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing

دلایل شانه خالی کردن از آزمایش واحد!

1- نوشتن آزمایشات زمان زیادی را به خود اختصاص خواهند داد.

مهمترین دلیلی که برنامه‌نویس‌ها به سبب آن از نوشتن آزمایشات واحد امتناع می‌کنند، همین موضوع است. اکثر افراد به آزمایش به‌عنوان مرحله آخر توسعه فکر می‌کنند. اگر این چنین است، بله! نوشتن آزمایش‌های واحد واقعا سخت و زمانگیر خواهند بود. به همین جهت برای جلوگیری از این مساله روش pay-as-you-go مطرح شده است (ماخذ: کتاب Pragmatic Unit Testing در سی شارپ). یعنی با اضافه شدن هر واحد کوچکی به سیستم، آزمایش واحد آن‌را نیز تهیه کنید. به این صورت در طول توسعه سیستم با باگ‌های کمتری نیز برخورد خواهید داشت چون اجزای آن‌را در این حین به تفصیل مورد بررسی قرار داده‌اید. اثر این روش را در شکل زیر می‌توانید ملاحظه نمائید (تصویری از همان کتاب ذکر شده)

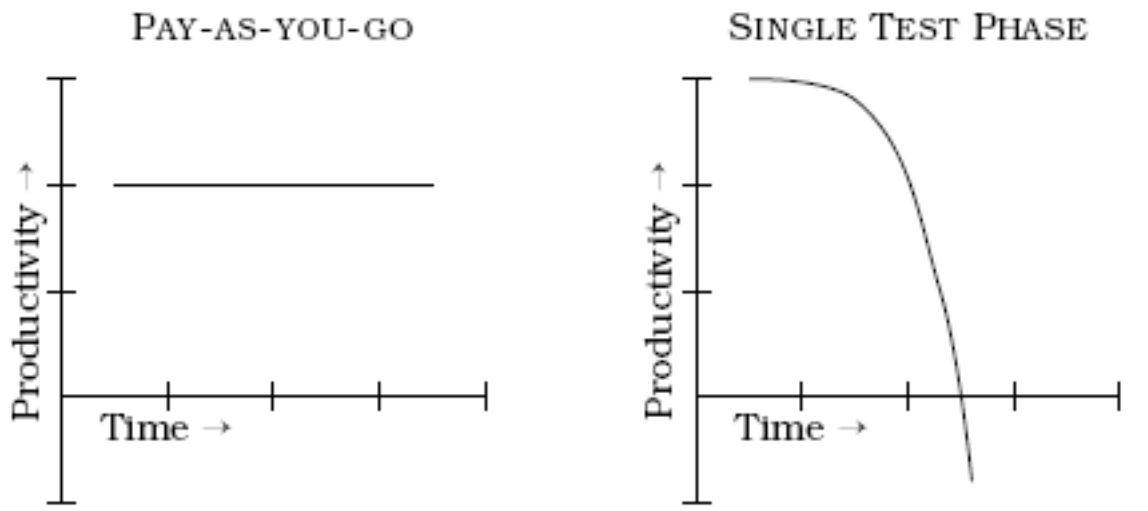


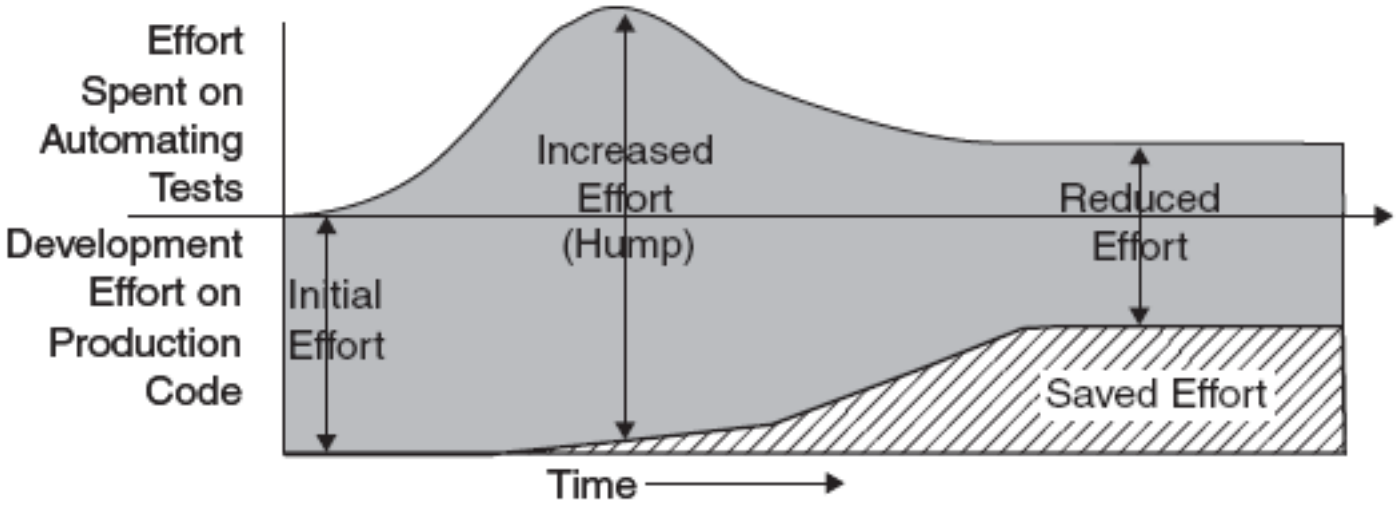
Figure 1.1: Comparison of Paying-as-you-go vs. Having a Single Testing Phase

نوشتن آزمایشات واحد زمانبر هستند اما توسعه پیوسته آن‌ها با به تاخیر انداختن آزمایشات به انتهای پروژه، همانند تصویر فوق تاثیر بسیار قابل توجهی در بهره وری شما خواهند داشت.

بنابراین اگر عنوان می‌کنید که وقت ندارید آزمایش واحد بنویسید، به چند سؤال زیر پاسخ دهید:
الف) چه مقدار زمان را صرف دیباگ کردن کدهای خود یا دیگران می‌کنید؟

ب) چه میزان زمان را صرف بازنویسی کدی کرده‌اید که تصور می‌رفت درست کار می‌کند اما اکنون بسیار مشکل زا ظاهر شده است؟
ج) چه مقدار زمان را صرف این کرده‌اید که منشاء باگ گزارش شده در برنامه را کشف کنید؟

برای افرادی که آزمایشات واحد را در حین پروسه توسعه در نظر نمی‌گیرند، این مقادیر بالا است و با ازدیاد تعداد خطوط سورس کدها، این ارقام سیر صعودی خواهند داشت.



تصویری از کتاب xUnit Test Patterns ، که بیانگر کاهش زمان و هزینه کد نویسی در طول زمان با رعایت اصول آزمایشات واحد است

-2 اجرای آزمایشات واحد زمان زیادی را تلف می‌کند.

نباید اینطور باشد. عموما اجرای هزاران آزمایش واحد، باید در کسری از ثانیه صورت گیرد. (برای اطلاعات بیشتر به قسمت حد و مرز یک آزمایش واحد در قسمت قبل مراجعه نمائید)

-3 امکان تهیه آزمایشات واحد برای کدهای قدیمی (legacy code) من وجود ندارد

برای بسیاری از برنامه نویس‌ها، تهیه آزمایش واحد برای کدهای قدیمی بسیار مشکل است زیرا شکستن آن‌ها به واحدهای کوچکتر قابل آزمایش بسیار خطرناک و پرهزینه است و ممکن است سبب از کار افتادن سیستم آن‌ها گردد. اینجا مشکل از آزمایش واحد نیست. مشکل از ضعف برنامه نویسی آن سیستم است. روش refactoring ، طراحی مجدد و نوشتن آزمایشات واحد، به تدریج سبب طراحی بهتر برنامه از دیدگاه‌های شیءگرایی شده و نگهداری سیستم را در طولانی مدت ساده‌تر می‌سازد. آزمایشات واحد این نوع سیستم‌ها را از حالت فلج بودن خارج می‌سازد.

-4 کار من نیست که کدهای نوشته شده را آزمایش کنم!

باید درنظر داشته باشید که این هم کار شما نیست که انبوهی از کدهای مشکل دار را به واحد بررسی کننده آن تحویل دهید! همچنین اگر تیم آزمایشات و کنترل کیفیت به این نتیجه برسد که عموما از کدهای شما کمتر می‌توان باگ گرفت، این امر سبب معروفیت و تضمین شغلی شما خواهد شد.

همچنین این کار شما است که تضمین کنید واحد تهیه شده مقصود مورد نظر را ارائه می‌دهد و این‌کار را با ارائه یک یا چندین آزمایش واحد می‌توان اثبات کرد.

-5 تنها قسمتی از سیستم به من واگذار شده است و من دقیقا نمی‌دانم که رفتار کلی آن چیست. بنابراین آن را نمی‌توانم آزمایش کنم!

اگر واقعا نمی‌دانید که این کد قرار است چه کاری را انجام دهید به طور قطع الان زمان مناسبی برای کد نویسی آن نیست!

-6 کد من کامپایل می‌شود!

باید دقت داشت که کامپایلر فقط syntax کدهای شما را بررسی کرده و خطاهای آن‌را گوشزد می‌کند و نه نحوه‌ی عملکرد آن‌را.

-7 من برای نوشتن آزمایشات حقوق نمی‌گیرم!

باید اذعان داشت که به شما جهت صرف تمام وقت یک روز خود برای دیباگ کردن یک خطا هم حقوق نمی‌دهند! شما برای تهیه یک کد قابل قبول و قابل اجرا حقوق می‌گیرید و آزمایش واحد نیز ابزاری است جهت نیل به این مقصود (همانند یک IDE و یا یک کامپایلر).

-8 احساس گناه خواهم کرد اگر تیم فنی کنترل کیفیت و آزمایشات را از کار بی کار کنم!!

نگران نباشید، این اتفاق نخواهد افتاد! بحث ما در اینجا آزمایش کوچکترین اجزا و واحدهای یک سیستم است. موارد دیگری مانند functional testing, acceptance testing, performance & environmental testing, validation & verification, formal analysis توسط تیم‌های کنترل کیفیت و آزمایشات هنوز باید بررسی شوند.

-9 شرکت من اجازه اجرای آزمایشات واحد را بر روی سیستم‌های در حال اجرا نمی‌دهد.

قرار هم نیست بدهد! چون دیگر نام آن آزمایش واحد نخواهد بود. این آزمایشات باید بر روی سیستم شما و توسط ابزار و امکانات شما صورت گیرد.

پ.ن.
در هشتمین دلیل ذکر شده، از acceptance testing نامبرده شده. تفاوت آن با unit testing به صورت زیر است:

آزمایش واحد:
توسط برنامه نویس‌ها تعریف می‌شود
سبب اطمینان خاطر برنامه نویس‌ها خواهد شد
واحدهای کوچک سیستم را مورد بررسی قرار می‌دهد
یک آزمایش سطح پائین (low level) به شمار می‌رود
بسیار سریع اجرا می‌شود
به صورت خودکار (100 درصد خودکار است) و با برنامه نویسی قابل کنترل است

اما در مقابل آزمایش پذیرش به صورت زیر است:
توسط مصرف کنندگان تعریف می‌شود
سبب اطمینان خاطر مصرف کنندگان می‌شود.
کل برنامه مورد آزمایش قرار می‌گیرد
یک آزمایش سطح بالا (high level) به شمار می‌رود
ممکن است طولانی باشد

عموما به صورت دستی یا توسط یک سری اسکریپت اجرا می‌شود
مثال : گزارش ماهیانه باید جمع صحیحی از تمام صفحات را در آخرین برگه گزارش به همراه داشته باشد

ادامه دارد...

نظرات خوانندگان

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۰/۰۹ ۰۷:۴۸:۰۰

سلام از این که هر روز وبلاگت رو اپدیت می کنی متشکر از این بحث هم خیلی خوشم می اد ادامه بده

نویسنده: Alex's Blog
تاریخ: ۱۳۸۸/۰۱/۲۵ ۱۵:۴۹:۰۰

سلام آقای نصیری
یه خواهش ازاتون داشتم اونم این بود که من این روزا مجبورم یه روالی رو برای performance & environmental testing تهیه کنم. بخاطر همین میخواستم ازتون خواهش بکنم اگه منابعی در این مورد دارین یا نرم افزارهایی برای این تستها سراغ دارین بهم معرفی کنید. (من بیشتر روی performance سیستمها می خوام کار کنم)
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۱/۲۵ ۱۶:۲۳:۰۰

سلام
در مورد تست کارآیی مشخص نکردید که چه پلتفرمی مد نظر شما است. اگر دات نت مد نظر است، نرم افزار شرکت red gate در این زمینه حرف اول را می‌زند:
www.red-gate.com/Products/ants_profiler/index.htm
شرکت سازنده resharper هم یک محصول دیگر در این مورد دارد:
/www.jetbrains.com/profiler

در مورد سایر پلتفرم‌ها هم کمابیش هست. profiler و code profiling را جستجو کنید.

نویسنده: Alex's Blog
تاریخ: ۱۳۸۸/۰۱/۲۶ ۰۸:۴۷:۰۰

ممنون از لطفتون
منظورم پلت فرم دات نت بود.
بازم ممنون.