آیا در شرکت نرم افزاری خوبی کار می کنید؟

این مقاله برای این هدف نوشته شده است که به شما برآوردی از شرکتی که در آن کار می کنید یا مدیریتش می کنید ؛ به دست آورید.خواندن آن تنها پنج دقیقه از وقت شما را می گیرد ولی ممکن است بعدها باعث صرفه جویی عمده ای در زمان و هزینه هایی که می کنید بشود.

- آیا از سیستم کنترل سورس بهره میبرید؟
- آیا میتوانید در یک مرحله، برنامه تان را build کنید؟
- آیا دارای build روزانه هستید؟
- آیا بانک اطلاعاتی از اشکالات (bugs)دارید؟
- آیا قبل از نوشتن کد جدید، به رفع اشکالات فعلی نرم افزار خود می پردازید؟
- آیا برنامه زمانبندیتان به روز است؟
- آیا دارای لیست مشخصات هستید؟
- آیا برنامه نویسان شما محیط آرامی برای کار کردن دارند؟
- آیا بهترین ابزارهایی را که وجود دارد میخرید؟

نشانی :
تهران - انتهای بلوار کشاورز - خیابان دکتر قریب - شماره ۸۵ - واحد ۳
تلفن: ۶۶۵۶۷۴۴۰ فکس: ۶۶۵۹۳۰۷۱
وب سایت: www.vestasoft.ir
نشانی اینترنتی: info@vestasoft.ir

- آیا بخش تست شما جداست؟
- آیا داوطلبان جدید در موقع مصاحبه، کد هم مینویسند؟
- آیا از آزمایش " خفت کردن در راهرو " استفاده می کنید؟

ویژگی شسته و رفته تست جوئل در این است که به راحتی میتوان به هر سؤال جواب بله یا نه داد. شما مجبور نیستید که تعداد خطهای کد در روز یا تعداد متوسط اشکال در هر قسمت را بشمارید. نقطه ضعف تست جوئل در این است که نباید از آن برای اطمینان از صحت نرم افزار نیروگاه اتمی خود استفاده کنید!

امتیاز ۱۲ عالی، ۱۱ قابل قبول و ۱۰ یا پایینتر نشان دهنده مشکلات جدی است. واقعیت این است که بیشتر موسسات نرم افزاری با امتیاز ۲ یا ۳ در حال فعالیت هستند و به کمک جدی نیاز دارند، چرا که شرکتهایی مانند Microsoftتمام مدت با امتیاز ۱۲ اداره می شوند.

البته، اینها تنها موارد مشخص کننده موفقیت یا شکست نیستند: مثلاً ، اگر شما تیم ماهری دارید که بر روی محصولی که هیچ کس آن را نمی خواهد کار میکند، خوب، مردم باز هم آن محصول را نخواهند خواست. از طرفی، تیم غیر عادی را میتوان تصور کرد که بدون انجام هیچ کدام از این کارها، نرم افزار فوق العاده ای تولید کند و دنیا را تغییر دهد. اما اگر همه شرایط برابر باشند، با رعایت این ۱۲ نکته، تیم منضبطی خواهید داشت که همیشه موفق به تحویل پروژه هایش میشود آن هم به موقع ! چیزی که در ایران کم دیده می شود.

آیا از سیستم کنترل سورس استفاده می کنید؟

من هم از پکیجهای تجاری کنترل سورس استفاده کردم، و هم از CVS که مجانی است ؛ و بگذارید به شما بگویم که CVS مناسب است. اما اگر سورس کنترل ندارید، فشار زیادی را برای اینکه برنامه نویسانتان بتوانند با هم کار کنند متحمل میشوید: برنامه نویسها راهی برای اینکه بدانند بقیه چه کرده اند، ندارند. و اشتباهات، به راحتی قابل بازگشت نیستند. و در آخر اینکه چون کد برنامه بر روی دستگاه تمام برنامه نویسان check out میشود، تا بحال نشنیده ام که پروژه های دارای سورس کنترل ، کد زیادی را به اشتباه از دست دهند.سیستم های SVN هم برای این کار وجود دارند که علاوه بر راه های کنترل سورس ؛ برای ایجاد ورژن برای هر سند الکترونیکی می تواند به کار رود.من خودم از Source Safe استفاده کرده ام و به نظر من در حد و اندازه های پروژه های معمولی و حتی کمی بزرگ ؛ خیلی پیچیده و گیر است.در حال حاضر از Tortoise SVN استفاده می کنم و ازش راضی هم هستم.البته باید سعی کنید که چند نفری روی یک قسمت کار نکنید و اگر کسی می خواهد که این کار را انجام دهد باید آن سند را قفل کند.

آیا میتوانید در یک مرحله، برنامه تان را build کنید؟

منظورم این است: برای ایجاد نسخه قابل تحویل به مشتری از آخرین سورس، چند مرحله وجود دارد؟ در تیمهای خوب، یک اسکریپت وجود دارد که با اجرای آن، یک check out کامل صورت میگیرد، تمام کد کامپایل میشود، EXEها ساخته میشوند (در تمامی نسخه ها، زبانها و #ifdef ها) ، پکیج قابل نصب تولید میشود و بالاخره در فرم رسانه نهایی (CD) یا وبسایت یا (... ایجاد میشود.

اگر این رویه بیشتر از یک مرحله داشته باشد، مستعد اشتباه است. و هر چقدر به زمان تحویل نزدیکتر میشوید، احتیاج به چرخه سریعتری برای تصحیح "آخرینbug "، و ساختن EXE نهایی دارید. اگر کامپایل کردن کد، اجرای سازنده installer و بقیه کارها بیست مرحله به طول انجامد، دست به اشتباهات احمقانه خواهید زد.

فقط به همین علت، آخرین شرکتی که در آن کار میکردم، از WISE به InstallShield تغییر کرد: لازم بود که رویه ایجاد installer از روی یک script به صورت خودکار نیمه شبها توسط NT Scheduler اجرا شود و WISE چنین قابلیتی نداشت. (دوستان خوب ما در WISE به من اطمینان داده اند که آخرین نسخه شان توانایی build های شبانه را دارد(.

آیا دارای build روزانه هستید؟

وقتی که از کنترل سورس استفاده میکنید، گاهی پیش میآید که یک برنامه نویس چیزی را check in میکند که باعث شکستن build میشود. به عنوان مثال، یک برنامه نویس یک فایل سورس جدید اضافه کرده است و همه چیز روی دستگاه خودش درست کامپایل میشود، ولی یادشان میرود که فایل را به مخزن کد (repository)اضافه کند. دستگاه خودش را هم قفل کرده و بی توجه و خوشحال به خانه میرود. حالا کسان دیگری نیز نمی تواند کار کنند. آنها هم به خانه میروند، البته غمگین.

شکستن بیلد آنقدر بد (و رایج) است که درست کردن بیلد روزانه کمک میکند که چنین موضوعی ناشناخته نماند. در تیمهای بزرگ، یک راه این که مطمئن شوید که چنین مشکلاتی در اولین فرصت برطرف شوند، این است که بیلد روزانه را هر روز، هنگام ناهار انجام دهید. همه تمامی check inهایی را که میتوانند قبل از رفتن به ناهار انجام میدهند. بیلد، زمانی که همه برگشتند تمام شده است. اگر که همه چیز درست است، که خیلی خوب است! آخرین نسخه سورس توسط همه check out شده و کار ادامه پیدا میکند. اما اگر که عمل بیلد با موفقیت روبرو نشده باشد، افراد با نسخه سالم قبلی به کار خود ادامه میدهند.

در تیمExcel ، با قانونی داشتیم: هر کسی که build را میشکست، به عنوان تنبیه، مسئولیت نگهداری از بیلدها را عهده دار میشد. این انگیزه خوبی بود هم برای جلوگیری از شکستن بیلد، و هم راه خوبی بود برای این که همه (به صورت چرخشی) یاد بگیرند که رویه چطور است.

آیا بانک اطلاعاتی از اشکالات (bugs) دارید؟

برایم مهم نیست که در این مورد چه فکر میکنید. اما اگر حتی در یک تیم یک نفره مشغول تولید کد هستید و از بانک منظمی که تمامی ایرادات برنامه را لیست کند استفاده نمیکنید، حتماً کد با کیفیت پایین تحویل خواهید داد. برنامه نویسان زیادی فکر میکنند که میتوانند لیست اشکالات و باگها را در حافظه خود نگهدارند. چه مزخرفاتی! من در آن واحد بیشتر از دو یا سه باگ را نمیتوانم بخاطر بسپارم، و صبح روز بعد، یا با عجله ای که زمان تحویل دارید، همه به فراموشی سپرده میشوند. قطعاً باید به صورت رسمی و مکتوب لیست اشکالات را نگهداری کنید.

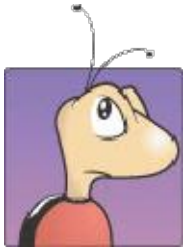
بانک باگ میتواند پیچیده و یا ساده باشد. یک بانک باگ سودمند باید حداقل اطلاعات زیر را در مورد هر باگ نگه دارد:

- مراحل کامل برای باز تولید اشکال
- رفتاری که انتظار آن میرود
- رفتار (ایراد داری) که واقعاً رخ میدهد
- فردی که رفع اشکال به او واگذار شده است
- آیا اشکال رفع شده است یا خیر

اگر پیچیدگی نرم افزار پیگیری باگها مانع از آن میشود که چنین کاری را انجام دهید، یک جدول پنج ستونه (با فیلدهای ضروری فوق) بسازید و شروع به انجام این کار کنید.

نشانی:
تهران - انتهای بلوار کشاورز - خیابان دکتر قریب - شماره ۸۵ - واحد ۳
تلفن: ۶۶۵۶۷۴۴۰ فکس: ۶۶۵۹۳۰۷۱
وب سایت: www.vestasoft.ir
نشانی اینترنتی: info@vestasoft.ir

آیا قبل از نوشتن کد جدید، به رفع اشکالات کنونی میپردازید؟



اولین نسخه Word تحت ویندوز، پروژه " نوحه مرگ " محسوب میشد که نوشتن آن به درازا کشید، فرجه ها دایماً به سر میرسیدند؛ افراد تیم، ساعتهای مسخره آمیزی کار میکردند، و پروژه باز ... و باز ... و باز به تاخیر میافتاد. استرس باور نکردنی بود. وقتی بعد از چندین سال، بالاخره محصول لعنتی اش بیرون آمد، مایکروسافت کل تیم Word را برای استراحت به جنوب مکزیک فرستاد ؛ و خودش به کنکاش روحانی جدی دست زد.

مایکروسافت متوجه شد که مدیران پروژه آنقدر بر حفظ " زمان بندی (schedule) " اصرار داشتند که برنامه نویسان مجبور به کد نویسی با عجله شده بودند، و بسیار بد کد می نوشتند -به این علت که فاز bug fixجز زمانبندی رسمی نبود. تلاشی برای پایین نگهداشتن تعداد خطاها وجود نداشت. حتی برعکس، روایت شده که یکی از برنامه نویسان که مسؤول نوشتن کد محاسبه ارتفاع خطوط متن بود، فقط نوشت return : 12;و بعد هم منتظر نشست تا در گزارش باگها بیاید که تابعش ، همیشه درست کار نمیکند. زمانبندی پروژه صرفاً تبدیل شده بود به لیستی از باگهایی که باید تولید میشد! بعدها، از این اتفاق با عنوان " متدولوژی عیوب نامحدود (infinite defects methodology) " یاد شد.

برای حل این معضل، مایکروسافت " متودولوژی کمترین عیوب (zero defects methodology) "را به صورت فراگیری اتخاذ کرد. بسیاری از برنامه نویسان داخل شرکت خندیدند - چون به نظر میرسید مدیریت به این نتیجه رسیده بود که با یک حکم سازمانی تعداد باگها را کم کند. اما در واقع، معنی " کمترین عیوب" در این است که در هر لحظه، بالاترین اولویت در رفع باگهاست، نه نوشتن کد جدید. اما چرا؟

به صورت کلی، هر چه برای تصحیح یک اشکال بیشتر معطل کنید، هزینه تصحیح آن (از لحاظ وقت و پول) بیشتر خواهد بود.

به عنوان مثال، وقتی که اشتباه تایپی انجام میدهید و کامپایلر آن را catch میکند، تصحیح آن کار اساساً ساده ایست. به همین ترتیب، بار اولی که کدتان را اجرا میکنید و در آن اشکالی میبینید، میتوانید بلافاصله آن را تصحیح کنید، چون همه کد در ذهنتان وجود دارد.

اما اگر در کدی که چند روز پیشتر آن را نوشته اید، ایرادی بیابید، یافتن محل دقیق آن مدتی طول خواهد کشید، البته وقتی که کدتان را باز خوانی کنید همه چیز بالاخره یادتان میآید و در یک زمان قابل قبول مشکل رفع میشود.

نشانی:
تهران - انتهای بلوار کشاورز - خیابان دکتر قریب - شماره ۸۵ - واحد ۳
تلفن: ۶۶۵۶۷۴۴۰ فکس: ۶۶۵۹۳۰۷۱
وب سایت: www.vestasoft.ir
نشانی اینترنتی: info@vestasoft.ir

اما بالاخره اگر در کدی که چند ماه پیش آن را نوشته اید باگی پیدا شود، به احتمال زیاد چیزهای زیادی را در مورد آن کد به فراموشی سپرده اید و تصحیح آن بسیار سختتر است. ممکن است مشغول تصحیح باگ کد کسی شده باشید که در آن لحظه برای مرخصی به جزایر دریای کارایب رفته باشد. در این صورت، تصحیح باگ به صورت یک علم در میآید : باید با درایت، وسواس، نظم و بدون هیچ تصوری از مدت زمانی که یافتن راه حل طول میکشد، عمل کنید.

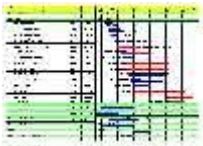
اگر در کدی که تحویل دادهاید ایرادی بیابید، متحمل هزینه باز هم زیادتر برای اصلاح آن خواهید شد.

پس اولین دلیلی که باید باگها را بلافاصله رفع کرد، کم کردن زمان لازم برای این کار است. دلیل دیگری هم وجود دارد: پیشبینی مدت زمان لازم برای نوشتن کد جدید، راحتتر از پیشبینی زمان لازم برای رفع یک باگ است. مثلاً اگر از شما بپرسم که چقدر زمان برای نوشتن کدی برای مرتب سازی یک فهرست لازم دارید، جواب نسبتاً دقیق میتوانید به من بدهید .اما اگر از شما بپرسم در جایی که IE 5.5 نصب شده باشد و کدتان دیگر کار نمیکند چقدر زمان لازم دارید تا باگ مربوطه را رفع کنید، بعید میدانم که حتی بتوانید حدسی بزنید !چرا که (بنابر تعریف صورت مساله) نمیدانید که منشأ مشکل کجاست. ممکن است سه روز وقتتان را بگیرد، ممکن هم هست که فقط دو دقیقه.

به بیان دیگر، اگر در برنامه زمانبندیتان تعداد زیادی باگی که باید رفع شوند، وجود داشته باشد، زمانبندیتان غیر قابل اعتماد است. اما اگر تمامی ایرادهای شناخته شده را تصحیح کرده اید و فقط کد جدید مانده، زمانبندیتان به طرز حیرت آوری دقیقتر خواهد بود.

نکته خوب دیگری هم که صفر نگهداشتن تعداد باگها در هر زمان دارد، عکس العمل سریعتر در برابر رقباست. بعضی برنامه نویسان این قضیه را به " آماده تحویل بودن" محصول در تمام لحظات، تعبیر میکنند. اگر رقیب شما قابلیت خفنی!! عرضه کرد، شما هم میتوانید آن امکان را فوراً به برنامه تان اضافه کنید و آن را تحویل دهید، بدون این که مجبور به تصحیح تعداد زیادی ایراد انباشته شده باشید.

آیا برنامه زمانبندیتان به روز است؟



میرسیم به بحث شیرین زمانبندی. اگر کدتان برای شرکتتان اهمیتی داشته باشد، پس حتماً به دلایل زیادی برای شرکتتان زمان اتمامش هم مهم هست. برنامه نویسان به صورت خیلی مفتضحی در زمینه زمانبندی، ترشرو هستند : سر قسمت مدیریت و بازاریابی فریاد میکشند " کار وقتی تمام میشود که تمام شده باشد" !

متاسفانه، این اصلاً به درد نمیخورد. برنامه ریزی های زیادی باید قبل از تحویل نهایی کد انجام گیرد : جلسات دمو ؛ تولید کاتالوگ ؛ ایجاد صفحه در وب سایت ؛ نمایشگاهها، تبلیغات و غیره. و تنها راه انجام این کارها، داشتن برنامه زمان بندی و به روز نگهداشتن آن است.

فایده حیاتی دیگر داشتن برنامه زمانبندی در این است که مجبورتان میکند تصمیم بگیرید چه قابلیتهایی را میخواهید در برنامه بگنجانید و این که امکانات با اولیت پایینتر را حذف کنید، و گرفتار بیماری featuritis نشوید(featuritis / scope creep / creeping featurism) .، و یا گرایش به ویژگیهای نو، بیماری طراحان و برنامه نویسان است ؛ طراحان یا برنامه نویسان مبتلا به این بیماری دوست دارند که به سیستم پیچیدهای بدون برنامه ریزی کافی امکانات و ویژگیهای نو اضافه کنند؛ و در واقع آن را - به صورت غیر اصولی - فقط پیچیده تر کنند.

من خودم در شرکتی کار می کردم که یک پروژه که قرار بود ۴ ماهه نوشته شود ؛ ۳ سال کشید و در انتها نیز Out Sourceشد و شرکت دیگری برنامه نویسی آن را به عهده گرفت.این پروژه پروژه حیاتی شرکت و دلیل به وجود آمدن شرکت بود ولی به دلیل نداشتن برنامه ریزی و مدیریت اشتباه در روند اجرا ؛ عملا شکست خورد.

نگهداشتن) schedule برنامه زمانبندی)، الزاماً سخت نیست.

آیا دارای لیست مشخصات هستید؟

نوشتن لیست مشخصات (specifications) درست مثل استفاده از نخ دندان است: همه قبول دارند که کار خوبی است ولی هیچ کس حوصله اش را ندارد.

دقیقاً مطمئن نیستم که چرا ؛ احتمالاً به این علت است که برنامه نویسان از نوشتن مستندات متنفرند. در نتیجه، تیمهایی که همه اعضایش برنامه نویس هستند وقتی سراغ مسأله ای میروند، ترجیح میدهند که راه حلشان به زبان کد باشد تا به صورت سند. ترجیح میدهند که از اول شیرجه بزنند و کد بنویسند تا این که ابتدا یک لیست مشخصات درست کنند.

در مرحله طراحی، اگر به معضلی بر بخورید، به سادگی با تغییر چند خط میتوانید آن را حل کنید. اما زمانی که کد نوشته شده است، هزینه درست کردن معضلات بسیار گزاف است : هم از لحاظ عاطفی (چون مردم از دور انداختن کد خود متنفرند)، و هم از لحاظ زمانی، و به همین علت نوعی مقاومت در مقابل درست کردن معضل بوجود میآید. نرم افزاری که از روی لیست مشخصات (specifications) تولید نشود معمولاً نتیجه اش طراحی بد و زمانبندی خارج از کنترل است. به نظر میرسد که مشکل Netscape نیز همین بوده - چهار نسخه اول آن قدر شلم شوربا شد که مدیریت به طرز احمقانه ای تصمیم گرفت همه کد آن را دور بیاندازد و از صفر شروع کند. سر Mozilla هم دوباره همین اشتباه را مرتکب شدند، که محصول آن هیولایی خارج از

کنترل است که چند سال طول کشیده تا فقط به مرحله آلفا برسد.

تئوری مورد علاقه من این است که اگر برنامه نویسان را به یک دوره متمرکز نویسندگی بفرستید، یاد خواهند گرفت که نویسندگان با ذوقی شوند. راه حل دیگر، استخدام مدیر برنامه (program manager) زبردستی است که خود نوشته ها را تهیه کند. در هر دو حالت، باید قانون " هیچ کدی بدون مشخصات پذیرفته نیست " را به اجرا بگذارید.

چند سال پیش در شرکتی کار می کردم که قرار بود برنامه ای با تکنولوژی Tapi برای آنها بنویسم.ابتدا قرار شد برای آزمایش قابلیتها و یادگرفتن نحوه کار با Tapi یک برنامه کوچک بنویسم و بعد در صورت به نتیجه رسیدن و مطمئن شدن از اینکه این کار قابل انجام است ؛ نرم افزاری برای این موضوع طراحی و پیاده سازی شود.به دلیل نبود هیچ گونه مستندی در این زمینه و عدم ثبت قابلیتهای مورد نیاز ؛ ۳ ماه بعد مدیر عامل شرکت فکر می کرد که این همان برنامه نهایی است که به نتیجه رسیده و اکنون باید به بازار عرضه شود!

در نوشته چهار قسمتی من، هر چه که در مورد نوشتن مشخصات لازم دارید را میتوانید بخوانید.

آیا برنامه نویسان شما محیط آرامی برای کار کردن دارند؟

این موضوع - که با دادن فضای مناسب، ایجاد آرامش و محیط دنج (privacy) به کارمندان) ITیا اصطلاحاً - (knowledge workersبهروری افزایش می یابد ، به صورت گسترده ای مستند شده است. کتاب کلاسیک مدیریت نرم افزار، PeopleWare ، این موارد را بر میشمرد.

مشکل اینجاست: همه ما میدانیم که نیروی کار فنی با قرار گرفتن در جریانی که از آن به "در بحر موضوع رفتن " تعبیر میشود - بهتر کار میکنند ؛ این زمانی است که تمرکزشان کاملاً بر روی کارشان است و حواسشان کاملاً از محیط اطرافشان پرت شده است. احساس زمان را از دست میدهند و از طریق تمرکز مطلق، چیزهای عالی ای خلق میکنند. نویسندگان، برنامه نویسان، دانشمندان، و حتی بازیکنان بسکتبال در مورد حالت " در بحر موضوع فرو رفتن " میتوانند برای شما توضیح دهند.

وارد شدن به این حالت کار ساده ای نیست. اگر اندازه گیری کنید، میبینید که حدوداً ۱۵ دقیقه طول میکشد تا به حداکثر کارایی خود برسید. بعضی مواقع، زمانهایی که خسته هستید و یا به اندازه کافی برای آن روزتان خلاقیت به خرج دادهاید، دیگر نمیتوانید در بحر موضوع بروید و بقیه روز را به کارهای بیهوده، خواندن وب و چت کردن میگذرانید.

مشکل دیگر در اینجاست که از دست دادن تمرکز کار ساده ایست :سر و صدا، تماسهای تلفنی، ناهار را بیرون خوردن، آن پنج دقیقه ای که برای نوشیدن قهوه تا کافی شاپ صرف رانندگی میکنید، و مخصوصا مزاحمت ها و سؤالهای همکاران، همگی تمرکز را از بین می برند. اگر همکار شما از شما سؤالی بپرسد که یک دقیقه کارتان را

متوقف کند، ولی آنقدر حواستان را پرت کند که نیم ساعتی طول بکشد تا به حالت سابق برگردید، بهره وری کلی تیمتان در خطر جدی قرار دارد. اگر در محیط شلوغی کار میکنید که اعضای بخش بازاریابی یا پشتیبانی در آنها زیر گوش برنامه نویسان مشغول داد و بیداد هستند - بهروری شما سقوط بدی خواهد کرد.

در مورد برنامه نویسان (به نسبت بقیه knowledge workers و طیفهای دیگری که نیازمند تمرکز زیاد هستند)، قضیه سختتر است. بهره وری، به توانایی شعبده بازی با جزئیات زیادی در حافظه موقت بستگی دارد. هر نوع وقفه ای باعث بهم ریختن این جزئیات میشود. وقتی کارتان را از سر می گیرید، هیچ کدام از جزئیات (نظیر نام متغیرهای محلی یا این که تا کجای پیاده سازی الگوریتم جستجویتان را انجام داده بودید) یادتان نخواهد آمد و مجبور به مراجعه به کار قبلتان هستید که سرعتتان را میگیرد.

ریاضیات این مساله زیاد سخت نیست: فرض کنید (که شواهد هم این فرض را تایید میکنند (که اگر برای یک دقیقه هم یک برنامه نویس را متوقف کنید، پانزده دقیقه از بهره وریش کم می کنید. برای مثال، غضنفر و افشین(که هر دو برنامه نویس هستند) را در دو پارتیشن کنار هم قرار میدهیم (در یک فضای کاملاً Dilbert ای). افشین ؛ نام تابع کپی رشته در یونیکد را به خاطر نمیآورد. میتواند جواب آن را جستجو کند - که ۳۰ ثانیه طول میکشد، و یا این که از غضنفر بپرسد - که ۳۰ ثانیه طول خواهد کشید. خوب، چون غضنفر در کنارش نشسته است ترجیح میدهد که از او بپرسد. حواس غضنفر پرت میشود و ۱۵ دقیقه را از دست میدهد (برای این که افشین ۱۵ ثانیه صرفه جویی کرده باشد(.

حالا اجازه دهید که غضنفر و افشین را در دو اتاق (با در و دیوار) جدا بگذاریم. اکنون وقتی که افشین اسم تابع را به خاطر نمی آورد، میتواند جواب آن را جستجو کند (که همان 30ثانیه طول میکشد) و یا این که از غضنفر بپرسد که ۳۰ ثانیه طول میکشد و شامل از جای خود بلند شدن هم میشود (که با توجه به وضعیت جسمی معمول برنامه نویسان و مسایل دیگر ، کار ساده ای نیست!). بنابر این ترجیح میدهد که جوابش را جستجو کند. ۳۰ ثانیه وقتش تلف میشود اما ۱۵ دقیقه به نفع غضنفر است! وای!

آیا بهترین ابزارهایی را که وجود دارد میخرید؟

نوشتن کد در یک زبان کامپایل شده از کارهایی است که هنوز بر روی کامپیوترهای خروس نشان، نمیتوان با سرعت انجام داد. اگر فرآیند کامپایل، بیشتر از چند ثانیه طول میکشد، خرید جدیدترین و بهترین کامپیوتر در وقتتان صرفه جویی خواهد کرد. اگر کامپایل کردن حتی ۱۵ ثانیه طول بکشد، برنامه نویسها حوصله شان سر میرود و میروند به سراغ خواندن سایت ..... ، که آن هم تمام هوش و حواسشان را به خود خواهد برد و ساعتها بهره وری از بین می رود.

debugکردن کد GUI با یک مانیتور اگر غیر ممکن نباشد، بسیار سخت و دردناک است. اگر در حال نوشتن کد GUI هستید، داشتن دو مانیتور همه چیز را بسیار ساده خواهد کرد.

بیشتر برنامه نویسان باید یک زمانی فایلهای bitmap را (برایicon ها وtoolbar ها) دستکاری کنند و اکثراً ویرایشگر مناسب برای این کار ندارند. استفاده از MS Paint برای این کار بیشتر شبیه یک شوخی است، که البته غالب برنامه نویسها از همین روش استفاده میکنند.

در محل کار قبلی ام، adminشبکه دایماً برای من spam میفرستاد و غر میزد که من بیشتر از ۲۲۰ مگابایت فضای مجازم، روی سرور جا اشغال کرده ام. من روزی جواب دادم که با توجه به قیمت هارد دیسک، هزینه فضای مورد استفاده کمتر از هزینه دستمال کاغذی من است و صرف کردن حتی ده دقیقه از وقتم برای کوچک کردن دایرکتوری ام یک هدر دادن بهره وری است.

تیمهای تولید نرم افزار درجه یک، برنامه نویسان خود را شکنجه نمیکنند. حتی موضوعات جزیی اعصاب خرد کن ناشی از داشتن ابزار نامناسب، جمع میشود و برنامه نویسها را بد اخلاق و ناراحت میکند. یک برنامه نویس بد خلق، یک برنامه نویس با کارایی پایین است.

یک نکته دیگر هم اضافه کنم... برنامه نویسها را به راحتی میتوان با رشوه دادن - بوسیله جدیدترین و با حال ترین وسایل - تطمیع کرد. این برایتان خیلی ارزانتر از دادن حقوق مکفی تمام می شود!

به یاد دارم که چند سال پیش در شرکتی کار می کردم که همیشه حقوق ها را دیرتر از حد معمول می داد. من همیشه آخر ماه به دفتر مدیر عامل می رفتم و یک ساعت از وقت هردوی ما به این موضوع اختصاص می یافت که چرا حقوق من دیر شده است.هنوز هم جمله آن مدیر در گوشم هست که به من می گفت " کار کردن خاک خوردن دارد " و من هم می گفتم : "اگر می خواستم خاک بخورم ؛ برای خودم و به صورت پروژه ای کار می کردم.علت کار کردن من در شرکت شما این است که می خواهم سر زمان مقرر ؛ درآمد خود را داشته باشم".همیشه هم صحبتها با این جملات خاتمه می یافت که "ما که فقط همین ماه حقوق را کمی دیر دادیم و شما همیشه حقوق خود را سر وقت دریافت می کنید!" و من هم می گفتم "من این جمله را چند ماه است که می شنوم"

حالا که به آن روزها فکر می کنم می بینم که من و تمامی کارکنان شرکت از چند روز مانده به آخر ماه مدام به زمان دادن حقوق فکر می کردیم و چون سر وقت داده نمی شد این فکر کردن ۱۰ الی ۱۵ روز طول می کشید که به صورت توانی ؛ تمام انرژی ما صرف این موضوع می شد.حتی کارکنان سر این موضوع با هم بحث می کردند و طنز می ساختند.در صورتی که مدیر عامل می توانست با پرداخت به موقع حقوق ؛ بهره وری را در آن شرکت بالا ببرد.البته من هنوز هم از اوضاع شرکت خبر دارم و می دانم که به جز این قضیه ؛ چند قضیه کوچک بهبود یافته اند.

آیا بخش تست شما جداست؟

اگر در تیم شما افرادی که وقتشان اختصاصاً برای تست کردن باشد - حداقل یک نفر برای هر دو یا سه برنامه نویس - وجود نداشته باشد، شما یا محصولات باگ دار تحویل خواهید داد؛ و یا این که با پرداخت حقوق قسمت تست نرم افزار به خود برنامه نویسان، پولتان را هدر میدهید. چون برنامه نویسان بیشتر از پشتیبانان ؛ حقوق می گیرند.خساست در زمینه افراد جهت بخش تست، آن قدر صرفه جویی احمقانه ایست که من واقعاً تعجب میکنم چرا اکثر مردم نمیفهمند. ضمنا برنامه نویس ها معمولا حالتهای بحرانی را در نظر نمی گیرند و با کاربر در تعامل نیستند و مشکلات کاربر را نمی دانند.نکته دیگری هم که در این وسط وجود دارد این است که برنامه نویسان معمولا همه چیز را از یک روش و آن هم روشی که خودشان انجام می دهند ؛ آزمایش می کنند.بنابراین اگر می خواهید نرم افزاری را آزمایش کنید ؛ هیچ وقت این کار را به برنامه نویس آن واگذار نکنید.

آیا داوطلبان جدید در موقع مصاحبه، کد هم مینویسند؟

آیا شما حاضرید یک شعبده بازی را بدون این که چند حقه برایتان اجرا کند، استخدام کنید؟ معلوم است که نه. آیا برای عروسیتان، آشپزی که غذایش را نچشیده باشید، استخدام میکنید؟ بعید میدانم. (مگر این که خاله بزرگتان باشد و بترسید که تا آخر عمر از شما به دل بگیرد و برنجد(.

با این وجود، هر روز، برنامه نویسهایی بخاطر داشتن resume جالب یا به این خاطر که مصاحبه گر از گپ زدن با او لذت برده، و یا ..... استخدام می شوند. یا باید به سوالات ساده ای مانند " فرق متد و خصوصیت چیه؟ " یا "فرق کرم و ویروس کامپیوتری چیه؟" یا "برنامه نویسی چند لایه به چه صورت انجام می شه؟" که با خواندن مستندات قابل پاسخگویی است، جواب دهند. واقعاً نباید برایتان اهمیتی داشته باشد که داوطلب، هزاران نکته پیش پا افتاده را حفظ کرده است یا نه، بلکه باید توانایی او در تولید کد برایتان مهم باشد .از همه چیز بدتر، سؤالات " معما گونه " است : آن دسته از سؤالاتی که پاسخ دادن به آنها غیر ممکن است ولی وقتی جواب را میدانید بدیهی به نظر میرسند. مثلا "فرق Padding و Margin چیست؟".بعضی از مصاحبه کنندگان برای نشان دادن توانایی خود و ضایع کردن مصاحبه شونده لیست بلند بالایی از سوالات و معماهای برنامه نویسی تهیه می کنند تا مصاحبه شونده حساب کار دستش بیاید! اگر شما هم این خصوصیات را دارید لطفا از این به بعد مصاحبه نکنید.

لطفاً از این کارها نکنید! و هر کاری که میکنید، حتماً از مصاحبه شونده بخواهید که برایتان کد بنویسد .

آیا از روش " خفت کردن " استفاده می کنید؟

در تستhallway usability ، شما یقه اولین فردی را که از راهرو رد می شود،را می گیرید؛ و مجبورش می کنید که بنشینند پای برنامه ای که همین الان نوشته اید. اگر با پنج نفر این کار را تکرار کنید، ۹۵٪ مشکلات کار با برنامه تان (usability problems) را کشف خواهید کرد.

طراحی " واسط کاربر " خوب آنقدرها هم که فکر میکنید سخت نیست، حتی برای این که مشتریها عاشق برنامه تان بشوند و آن را بخرند، واجب هم هست.

اما مهمترین نکته در مورد واسطهای کاربر (user interfaces) این است که اگر برنامه تان را به پنج یا شش نفر نشان دهید، به سرعت بزرگترین مشکلات کاربران را خواهید دانست Jakob Nielsen .مقالهای دارد که در آن توضیح میدهد چرا این چنین است. خلاصه این که حتی اگر در زمینه UI واقعاً ضعیف باشید، با آزمایشهای قابلیت استفاده راهرویی که شرح آن رفت و خرجی هم ندارد، UIتان واقعاً بهبود پیدا میکند .

چهار استفاده برای این تست:

۱. موسسه نرم افزاری خود را بسنجید تا بدانید که در آنجا باید چگونه کار کنید!
۲. اگر مدیر یک تیم نرم افزاری هستید، با استفاده از این تست چک کنید که آیا تیمتان با تمام توانش کار میکند یا نه؟ اگر امتیازتان ۱۲ است، بهتر است که برنامه نویسانتان را به حال خودشان رها کنید و انرژیتان را روی عدم مداخله بخش مالی و فروش شرکت در کار برنامه نویسها متمرکز کنید.
۳. اگر میخواهید جایی استخدام شوید، از کارفرمای جدیدتان بپرسید که چه امتیازی در این تست به دست میآورند. اگر خیلی پایین بود، مطمئن شوید که اختیارات درست کردن اوضاع را دارید و الا وجودتان بی حاصل خواهد بود.
۴. اگر سرمایه گذاری هستید که در حال برنامه ریزی و سنجش تیمتان میباشید و یا شرکتی نرم افزاری هستید که قصد ترکیب با مجموعه نرم افزاری دیگری را دارید، این تست وضعیت را به صورت سر انگشتی به شما خواهد گفت.