

فصل اول

افسانه برنامه نویس نابغه

از آن جایی که این کتاب در مورد خطرات روابط اجتماعی در توسعه دهی محصولات خلاقانه است، به نظر منطقی می‌آید که تمرکزمان را روی تنها متغیری که شما قطعاً به آن کنترل دارید، بگذاریم؛ یعنی خود شما.

به طور ذاتی، انسان‌ها هیچ وقت کامل نیستند. ولی قبل ازینکه عیب و ایرادهای همکارهای خودتان را بسنجدید، لازم است که مشکلات و نقص‌های شخصی خودتون را درک کنید. ما می‌خواهیم که در مورد رفتارها، خلق و خو و واکنش‌های خودتان فکر کنید و در نتیجه این تفکر امیدواریم که ایده بهتری در مورد اینکه چطور می‌شود یک برنامه نویس کارآمد و موفق شد به دست آورید. نهایتاً شما انرژی کمتری را برای سرو کله زدن با انسان‌ها هدر خواهید داد و وقت بیشتری را صرف کدنویسی خواهید کرد.

نکته مهم و اصلی این فصل این است که شما درک کنید برنامه نویسی یک کار تیمی است. و برای این که در یک تیم مهندسی (یا هر تیم خلاقانه دیگری) موفق باشید، باید اخلاق و رفتار خودتان را روی سه محور اصلی تواضع، احترام و اعتماد برقرار کنید. قبل از این که جلوتر بريم، برای شروع بهتر است ببینیم برنامه نویس‌ها به طور کلی چطور رفتار میکنند.

کمک کن کدم رو مخفی کنم

در ده سال گذشته، هر دوی ما وقت به نسبت خوبی را صرف سخنرانی در کنفرانس‌های مربوط به برنامه نویسی کردیم. بعد از این که در سال 2006 سرویس Project Hosting را به صورت متن باز از طرف گوگل عرضه کردیم، با سوال‌ها و درخواست‌های زیادی روبرو شدیم. اواسط سال 2008 یه روند کلی در کل این سوال‌ها به وضوح مشهود بود:

- میشه لطفاً این قابلیت رو به Subversion اضافه کنید تا بعضی از branch‌ها از دید بقیه مخفی باشند؟
- من تونین این امکان رو فراهم کنید که پروژه‌های متن باز در ابتدا مخفی و خصوصی باشند ولی وقتی آمده شدن بشه اونا رو عمومی و در دسترس همه قرار داد؟
- من میخوام تمام کدم رو از اول بنویسم. میشه کمک کنید تمام نسخه‌ها و تاریخچه کدهای گذشته من پاک بشن؟

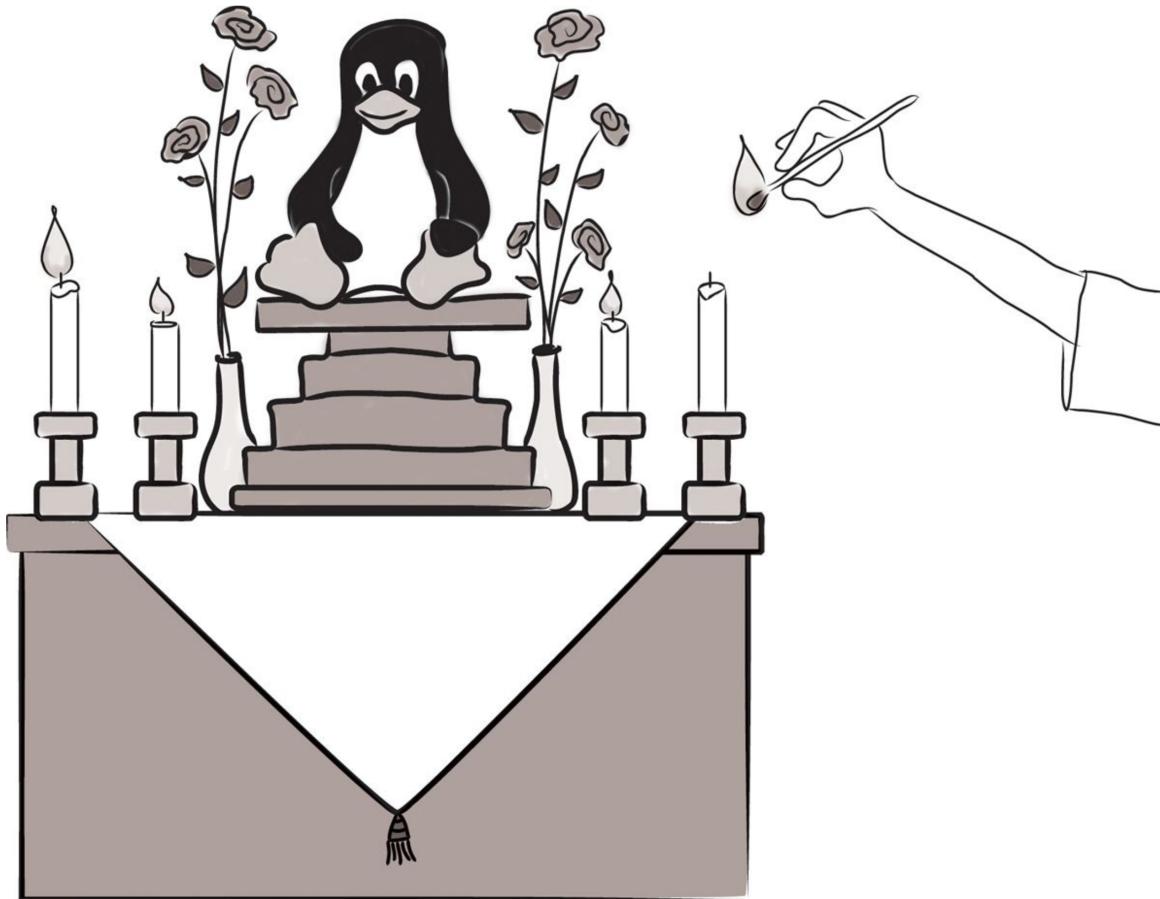
وجه مشترک تمام این درخواست‌ها رو متوجه می‌شوید؟
جواب مشخص است: عدم اعتماد به نفس.

افراد ازینکه بقیه کار ناتمام و در حال پیشرفت‌شان را ببینند، واهمه دارند. به نظر منطقی می‌آید و بخشی از طبیعت انسان هم همین است. هیچ کس دوست ندارد مورد انتقاد قرار بگیرد، مخصوصاً در مورد کارهایی که هنوز کاملاً تمام نشده‌اند. این نحوه نگرش افراد ولی یک سر نخ کلی به ما در مورد دنیای برنامه نویسی داد. این عدم اعتماد به نفس، در واقع یک نشانه هست از وجود یک مساله وسیع‌تر.

اسطوره نبوغ

هر دوی ما در طول سالهای دهه 90 در شیکاگو زندگی میکردیم و شاهد قهرمانی های متعدد تیم Chicago Bulls در بسکتبال بودیم. تلویزیون ملی پر شده بود از داستانهای افتخارات این تیم فوق العاده. ولی رسانه ها بیشتر روی چه چیزی تمرکز می کردند؟ ستاره خارق العاده تیم، مایکل جردن! نه کل تیم. هر بسکتبالیستی در دنیا دوست داشت که بتواند مثل MJ بشود. ما رقصیدن مایکل جردن با سایر بازیکن ها در زمین را می دیدیم. MJ همه جا بود: در تبلیغات تلویزیونی، یا حتی فیلم های در پیتی که در آن ها با شخصیت های کارتونی بسکتبال بازی می کرد. او یک ستاره بود. هر چه ای توی هر محله ای که بسکتبال بازی می کرد، آرزو داشت وقتی که بزرگ شد جای مایکل جردن را بگیرد.

برنامه نویس ها هم در تلاش برای پیدا کردن و پرستش بت های قهرمان، طبیعت مشابهی دارند. لینوس تروالدز، ریچارد استالمن، بیل گیتس، همه قهرمانانی هستند که با شاهکارهایشان دنیا رو تغییر دادند. لینوس تروالدز به تنها لینوکس را ساخت، مگر نه؟



ولی واقعیت این است که لینوس صرفا کد مقدماتی یه هسته مشابه Unix را به عنوان یک نمونه کار نوشت و آن را برای یک سری آدم ایمیل کرد. البته که کار کمی نبود و یک موفقیت بزرگ به حساب می آید. ولی کاری که کرد صرفاً یک قسمت کوچک از کل ماجرا بود. لینوکس صدها برابر بزرگ تراز این حرف هاست و توسط صدها شخص باهوش توسعه پیدا کرده. دستاوردهای لینوس این بود که این گروه از افراد را هماهنگ و رهبری کرد. لینوکس نهایتا نتیجه درخشنان این تلاش گروهی به حساب می آید. (ضمن این که خود Unix هم در اصل توسط یه گروه نخبه در Bell Labs نوشته شده بوده. نه صرفا توسط کن تامسون و دنیس ریچ).

یک مثال مشابه دیگر: آیا استالمن شخصا تمام مجموعه نرم افزارهای بنیاد نرم افزار آزاد^۱ را نوشته است؟ او نخستین نسل از نرم افزار Emacs را نوشت. ولی صدها شخص دیگه پشت سایر نرم افزارهایی مثل bash یا GCC هستند که روی لینوکس اجرا می شوند. استیو جابز کل تیمی را هدایت کرد که Macintosh را ساختند. و همینطور بیل گیتس،

اگر چه یک interpreter برای زبان برنامه نویسی Basic که قابلیت اجرا روی کامپیوترهای ابتدایی خانگی داشت را نوشت، ولی دستاوردها اصلی او تاسیس یک شرکت بزرگ بود که حول محوریت MS DOS به موفقیت رسید. با این وجود همه این افراد تبدیل به سمبول هایی شدند به نمایندگی از یه سری موفقیت جمعی.

ما یک جردن چطور؟ او هم همینطور. ما از او یک بت می سازیم ولی واقعیت این است که او به تنها یه تمام بازی های تیمشان را پیروز نشده. نبوغ واقعی او در این بود که می توانست به خوبی با هم تیمی هایش بازی کند. مربی تیمشان، فیل جکسون، بسیار زرنگ بود. ترفند مربی گری او شگفت آور است. او به خوبی می دانست که یک بازیکن به تنها یه نمی تواند باعث قهرمانی شود. بنابراین او یک تیم رویایی با محوریت مایکل جردن را گردآوری کرد. تیم او مثل ماشینی که خوب روغن کاری شده باشد، روان بود و اگر نه بیشتر ولی حداقل اندازه خود مایکل جردن، این تیم چشمگیر بود.

خوب پس چرا ما در تمام این داستان ها، از اشخاص بت می سازیم؟ چرا ما آدم ها توجه همون جلب محصولاتی می شود که یک آدم معروف از آنها تعریف کرده باشد؟ چرا دوست داریم لباس یا کفش مایکل جردن را داشته باشیم؟ شهرت، می تواند بخش خیلی بزرگی از دلیل این کار باشد. انسان ها به طور غریزی به دنبال رهبر و یا یک الگو می گردند تا از او یک بت بسازند و سعی کنند که شبیه او رفتار کنند. همه ما نیاز به قهرمانانی داریم که از آنها الهام بگیریم. دنیای برنامه نویسی هم از این قضیه مستثنی نیست. شهرت در دنیای تکنولوژی یک مفهوم کاملاً شناخته شده است و همه ما دوست داریم که بتوانیم چیزی را توسعه بدیم که دنیا رو عوض کنه، یا یک زبان برنامه نویسی فوق العاده طراحی کنیم.

ته دل همه ما آرزویی نهفته است برای اینکه روزی از ما به عنوان یک نابغه یاد کنند. رویا و خیال پردازی شما این است که روزی یک ایده ناب به ذهنتان خطور کند. به غار تنها یی خودتان بروید و برای هفته ها یا ماه ها، ایده ای که به ذهنتان رسیده را پیاده سازی کنید. سپس این ایده ناب را رها کنید در دنیا تا همه انگشت به دهان به نبوغتان خیره بمانند. همه همکارهای شما تحت تاثیر این هوش و ذکاوت استثنایی قرار بگیرند و به صف بایستند تا این نرم افزار را بینند. شهرت و ثروت هم طبیعتاً با همه این ها خودش را به شما نشان می دهد.

قصد جسارت نداریم، ولی واقع بینانه به قضیه نگاه کنیم، شما احتمالاً یک نابغه نیستید. بله ما مطمئنیم که شما یک خانم یا آقای باهوش هستید، ولی متوجه هستید که نابغه های واقعی چقدر کمیاب هستند؟ بله، متوجه هستیم که شما کد نویسی بلدید، که مهارت آسانی نیست و خوب شما را در گروه افراد به نسبت باهوش جامعه قرار می دهد. اصلاً حتی اگر فرض کنیم شما یک نابغه ناب هستید، باز هم این به تنها یی کافی نیست. نابغه ها هم اشتباه می کنند. همچنین، داشتن یک ایده ناب و مهارت خارق العاده برنامه نویسی به این معنی نیست که لزوماً نرم افزار نهایی به موفقیت خواهد رسید. این توانایی شما در همکاری کردن با بقیه است که می تواند باعث شود شما در کارتان یک آدم موفق یا ناموفق شوید.

کاملاً مشخص است که افسانه وجود داشتن یک نابغه خارق العاده همچنان از عدم اعتماد به نفس ما نشات میگیرد. بیشتر برنامه نویس ها از به استراتک گذاشتن کاری که به تازگی شروع کردند صرفاً به این دلیل واهمه دارند که حس میکنند همکارها اشتباه هاتشان رو می بینند و متوجه می شوند که آن ها یک نابغه تمام عیار نیستند. یک بار یک برنامه نویس در وبلاگ بن گفت:

من جدا اعتماد به نفس خودم را به کلی از دست میدهم وقتی حس می کنم ممکنه آدم ها کار من رو قبل ازینکه به اتمام برسه ببینند. حس میکنم که به شدت کار من رو قضاؤت خواهند کرد و فکر خواهند کرد که من چقدر احمقم.

این حس، به شدت بین برنامه نویس ها متداول است، و واکنش طبیعی این است که در غار تنها یی خودتان پنهان شده و مدام کار کنید و کار کنید و کار. هیچ کس سوتی های شما را نخواهد دید و شما همیشه این شانس را خواهید داشت که از کارتان وقتی پرده برداری کنید که کامل شده باشد. این قدر در خلوت خودتان پنهان کار کنید تا نتیجه بی نقص و بی عیب شود.

یک دلیل دیگر برای پنهان کار می تواند نگرانی از این موضوع باشد که یک برنامه نویس دیگر ممکن است از کار

شما ایده گرفته و قبل ازینکه شما فرصت کنید که اجرای آن را تمام کنید، او ایده شما را عملی کند. با مخفی کردن کارتان، یک جواری از ایده تون حفاظت میکنید.

می دانیم که الان احتمالاً به چه چیزی فکر می کنید: خوب که چی؟ آدم اجازه نداره هر طوری که دوست داشته باشه کار کنه؟!

خوب راستش نه! توی این مورد خاص ما قویاً اعتقاد داریم که این کار شما یک اشتباه بزرگ است. در ادامه دلیل این عقديه را توضیح می دهیم.

مخفي کاري در برنامه نويسي، عامل ضرر و زيان

اگر شما تمام کارتان را در خلوت تنهائي خودتان انجام بدھي، خطر شکست و همینطور احتمال عدم پيشرفت خودتون رو به شدت افزايش می دهد.

اولا، اصلا از کجا می دانيد که شما در مسیر درستی قرار داريد؟

تصور کنيد شما يك سازنده و طراح دوچرخه هستيد. يك روز يك اиде خارق العاده و ناب يك طرح تازه برای دنده دوچرخه به ذهنتان ميرسد. قطعاتي که لازم داريد را سفارش داده و هفته ها در کارگاهتان روی ساخت يك نمونه اوليه، سخت تلاش ميکنيد. وقتی همسایه شما، که اتفاقاً او هم در ساخت دوچرخه سررشيته دارد از شما درمورد کاري که ميکنيد سوال می پرسد، شما چيزی توضیح نمیدهيد. شما نمی خواهيد که تا زمانی که کارتan کامل و بی نقص تمام نشده، کسی در مورد پروژه شما چيزی بداند. چند ماه دیگر ميگذرد و شما برای ساخت نمونه کارتan دچار مشکل می شويد. ولی خوب چون تمام را رو مخفیانه کار کردید، کمک گرفتن از دیگران در اين مرحله دیگر غيرممکن است. يك روز ناگهان همسایه رو می بینيد که دوچرخه خودش را از پارکينگ بيرون آورده و روی دوچرخه اش يك دنده خيلي مدرن و خاص سوار کرده است. کاشف به عمل می آيد که اتفاقاً او هم روی يك اиде خيلي مشابه به اиде شما کار می کرده ولی از چند نفر دیگر از دوستان و همکاران خودش هم کمک گرفته است. اينجاست که شما دیگر اعصابتان به هم ميريزد. کار خودتان را به همسایه نشان می دهيد و او خيلي سريع به شما نشان ميدهد که کار شما کجاها چه اشکالاتی دارد. اشکالاتی که احتمالا همان هفته اول حل می شدند اگر همان ابتدا همسایه را در جريان پروژه خود گذاشته بوديد.



چند نکته در اين حکایت نهفته است. اگه شما اиде ناب خودتان را از بقیه مخفی نگه داريد و از نشون دادن کار در حال پيشرفتتan به بقیه خودداری کنيد، در حال قمار بزرگی هستيد. اشتباهات اساسی در طراحی های اولیه خيلي به راحتی پيش می آيند. ممکن است که خيلي راحت دوباره چرخ را اختراع کنيد! همچنین فواید استفاده از همکاري بقیه را نيز از دست می دهيد. دقت کرديد که چطور همسایه شما دنده دوچرخه را با کمک همکارهايis سريع تر از شما پياده سازي کرد؟

مردم به همين دليل است که قبل از ينكه در آب شيرجه بزنند، دمای آب را با نوك پنجه های پايشان چک ميکنند. قبل از شروع هر کاري لازم است که مطمئن شويد که اولاً شما روی موضوع درستی کار ميکنيد، ثانياً نحوه اجرای کارتan

صحیح است، و ثالثاً این کار قبل انجام نشده باشد. احتمال سوتی دادن در ابتدای کار بسیار بالاست. هر چه بیشتر از بقیه نظرشان را بپرسید، احتمال اشتباهات اولیه را کمتر میکنید.^۲

اینکه کارتان را از ابتدا با بقیه به اشتراک بگذارید و نظرشان را بپرسید، فقط باعث کاهش اشتباه و جلوگیری از کج رفتن نمیشود. بلکه باعث تقویت خصوصیتی در پروژه شما میشود که ما اسم آن را می‌گذاریم: ضریب اتوبوسی^۳!

ضریب اتوبوسی: برابر است با تعداد افرادی که لازم است تا با یک اتوبوس تصادف کنند تا پروژه شما از بین برود.



دانش و اطلاعات چقدر بین افراد تیمان پخش شده؟ اگر شما تنها کسی هستید که جزئیات یک نمونه کار را بلد هستید، هرچند که ممکن است امنیت شغلی خوبی به نظر بیاید، ولی اگر روزی یک اتوبوس با شما برخورد کند، کل پروژه از بین خواهد رفت.

ولی اگر شما و دوستان با هم روی پروژه کار کرده باشید، عملاً ضریب اتوبوسی پروژه را دو برابر کردید. و اگر توانسته باشید که یک تیم را در طراحی و پیاده سازی نمونه کار دخیل کنید، وضعیت شما بسیار بهتر هم خواهد بود. پروژه صرفاً با ناپدید شدن یک نفر، از بین نخواهد رفت. درست است که احتمالاً اعضای تیم خدایی نکرده با اتوبوس برخورد نخواهند کرد، ولی اتفاقات پیش بینی نشده دیگری ممکن است پیش بیایند. مثلاً یک نفر ممکن است که ازدواج کند، یا به شهر دیگری منتقل شود، یا از شرکت استعفا دهد، یا مجبور شود از یک عضوی از خانواده که مریض شده مراقبت کند. نکته بحث این است که شما باید در مورد پروژه این آینده نگری را داشته باشید و با بالا بردن ضریب اتوبوسی تیم احتمال موفقیت پروژه را افزایش دهید.

صرف نظر از بحث ضریب اتوبوسی، مسئله سرعت کلی کار هم حائز اهمیت است. به راحتی می‌شود فراموش کرد که چقدر سرعت کار کردن انفرادی پایین است. خیلی پایین تر از میزانی که آدم ها دوست دارد به آن اعتراف کنند. وقتی تنها کار میکنید، چقدر چیز جدید یاد می‌گیرید؟ سرعت کارتان چقدر است؟ اینترنت دنیای اطلاعات است، ولی به هیچ وجه جای تجربه انسان ها رو نمیگیرد. کار کردن در تیم باعث می‌شود که خرد جمعی پشت یک پروژه افزایش پیدا کند. وقتی روی یک مساله ای گیر میکنید، چقدر از وقتتان را هدر می‌دهید تا خودتان را از چاله ای که در آن افتادید بیرون بکشید؟ حالا به این فکر کنید که چقدر این تجربه متفاوت خواهد بود اگر چند نفر از همکارانتان کنار شما باشند و در لحظه به شما بگویند که کجا اشتباه کرده اید و چطور می‌توانید مسئله را حل کنید. دقیقاً به همین دلیل است که اعضای تیم کنار هم می‌نشینند (یا بصورت دو نفره برنامه نویسی میکنند).

بیشتر اوقات شما به یک جفت چشم دیگر هم نیاز خواهید داشت.

اجازه دهید یک مثال دیگر بزنم. فرض کنید شما در حال کد نویسی با یک زبان برنامه نویسی که compiler دارد هستید. آیا روزها می نشینید و ده هزار خط کد می نویسید و وقتی مطمئن شدید که همه چیز را کامل و بی نقص تمام کردید، دکمه کامپایل رو برای اولین بار فشار می دهید؟! البته که نه! چه فاجعه ای به بار می آید. ما برنامه نویس ها با تغییرات کوچک که بتوانیم سریع نتیجه آنها را ببینیم، خیلی بهتر کار میکنیم. یکتابع می نویسیم، کامپایل می کنیم، یک تست به آن اضافه می کنیم، دوباره کامپایل می کنیم، کمی کد را تغییر می دهیم، مجدداً کامپایل می کنیم. اشتباهات تایپی و اشکالات کد را بلافضلله حل میکنیم. دائمآ نیاز داریم که به کامپایلر تکیه کنیم تا هوای ما را داشته باشد. این طور می توانیم کیفیت کدمان را بالا ببریم و نرم افزار را کم کم بهتر و قوی تر کنیم.

این نیاز به پیش بردن کار در تیکه های کوچک، نه فقط در سطح کدنویسی بلکه در سطح کل پروژه نیز وجود دارد. تغییرات اجتناب ناپذیرند و پروژه ها باید دائمآ خود را با این تغییرات تطبیق دهند. پروژه ها معمولاً با موانع از قبل پیش بینی نشده ای برخورد می کنند. گاهی اوقات کلاً متوجه می شویم که هیچ چیز طبق برنامه ای که در نظر داشتیم پیش نمی رود. نیازهای پروژه ناگهان تغییر می کنند. چطور ما می توانیم به سرعت برنامه ها و طرح هایمان را با تغییرات جدید تطبیق دهیم؟ جواب: با کار کردن بصورت تیمی.

اریک ریموند یک جمله معروف دارد:

وقتی تعداد چشم ها زیاد شوند، باگ ها و ایرادهای نرم افزار نمایان می شوند.

شاید این جمله را بتوان کاملتر و بهتر این طور بیان کرد: وقتی تعداد چشم ها زیاد شوند، پروژه شما از مسیر درست خارج نمی شود.

آدم هایی که توی غارتنهایی خود کار میکنند، روزی بیرون می آیند و می بینند که با وجود این که کارشان را به اتمام رسانندن، ولی دنیای بیرون غار تغییر کرده و ایده اولیه پروژه آنها دیگر در این دنیای جدید معنی نمی دهد.

مهندسين و دفترکار شخصي

بیست سال پیش، ذهنیت کلی آدم ها این بود که یک اتاق شخصی که بتوان در آن را بست، بهترین مکان برای یک مهندس است تا بتواند بالاترین کارایی را داشته باشد. فرض و گمان اون زمان این بود که این دفتر شخصی، باعث می شود که یک برنامه نویس مدت زمان بیشتری را بتواند برای خودش داشته باشد و فقط روی کد زدن تمکز کند. ولی ما فکر می کنیم نه تنها اتاق و دفتر شخصی برای بیشتر مهندسین نرم افزار اصلا ضروری نیست، بلکه حتی خطرناک هم هست. امروزه، نرم افزارها توسط تیم ها نوشته می شوند، نه اشخاص. داشتن یک کانال ارتباطی قابل اتکا با سایر اعضای تیم، حتی از داشتن اتصال پرسرعت به اینترنت هم مهم تر و ضروری تر است. تمام وقت دنیا را هم اکر بدون هیچ مزاحمی داشته باشی، اگر روی کار اشتباهی تمکز کرده باشی، وقتی رو تلف کردي.

متاسفانه به نظر می آید که شرکت های مدرن دنیای تکنولوژی، این روزها بعضاً از آن سمت بوم افتاده اند. وقتی وارد شرکت آن ها می شویم، ۵۰ یا حتی تا ۱۰۰ نفر را می بینید که در سالن های بسیار بزرگ در کنار هم کار می کنند. این طرح پلان آزاد و باز در دفاتر مهندسی این روزها موافق و مخالف های خودش را دارد. همه از کوچک ترین مکالمه ها خبردار می شوند و آدم ها برای صحبت کردن احساس راحتی نمی کنند چون نگران هستند که باعث اذیت و آزار بقیه شوند. این هم به اندازه دفترهای شخصی مضر است.

ما معتقدیم که حد وسط این دو حالت احتمالاً بهترین راه حل باشد. تیم ها را به گروه های ۶ تا ۱۲ نفره تقسیم کنید و هر تیم را در اتاق مخصوص خودشان قرار دهید. بدین صورت صحبت های خود را در کنار هم کار می کنند و کسی از حرف زدن خجالت نمی کشد و باعث آزار دیگران هم نمی شوند. قطعاً در این حالت هم مثل هر حالت دیگری، تک تک افراد همچنان نیاز دارند که جلوی سرو صدای مزاحم را بگیرند. به همین دلیل است که خیلی از آدم ها راه حل های خلاقانه ای پیدا کردنند تا بتوانند به بقیه نشان دهند که در کاری عمیق شدند و نمی خواهند که کسی مزاحم آن ها شود. ما در تیمی کار میکردیم که بین خودمان یک قانون گذاشته بودیم که هر موقع هر کسی با فرد دیگری کار داشت اسمش رو صدا میکرد و میگفت Breakpoint. اگر فرد مورد نظر فرصت صحبت کردن داشت با صندلی خود من چرخید و با هم صحبت میکردند و اگر فرصت نداشت، میگفت Ack به این معنی که در اولین فرصتی که پیدا

میکرد می آمد و با همکارش صحبت میکرد. تیم های دیگری را هم دیدیم که از هدفون های مخصوص Noise Cancellation استفاده می کردند تا سرو صدای محیط پیرامون رو حداقل کنند. به طور کلی خود عمل گذاشتند هدفون روی سر می تواند یک نشانه تلقی شود ازین که "الآن مزاح من نشوید مگر اینکه کارتان خیلی واجب باشد". حتی تیم هایی را هم دیدیم که از عروسک یا وسایل دیگری استفاده میکردند تا نشان دهنده در حال حاضر سرشان شلوغ است و یا در کاری عمیق شده اند.

منظور ما را اشتباه برداشت نکنید. بله ما معتقدیم که برنامه نویس ها نیاز به بازه های زمانی بدون وقفه و مزاحم برای عمیق شدن در برنامه نویسی دارند. ولی در عین حال عمیقاً انتظار داریم که آن ها باید یک کانال ارتباطی قوی و همیشه در دسترس با سایر هم تیمی های خودشان داشته باشند. هنر واقعی، پیدا کردن حد وسط بین این دو نیاز است.

خوب نتیجه این شد که: ریسک تنها یک کار کردن ذاتاً از کار کردن با بقیه بیشتر است. به جای اینکه از این نگران باشید که یک نفر ایده کار شما را بذدد یا نحوه کار شما را قضاوت کند، خیلی بیشتر باید ازین بترسید که وقتتان را روی یک کار اشتباه تلف کنید. متاسفانه، مسئله مخفی کردن ایده ها فقط محدود به رشته مهندسی نرم افزار نیست. این یک مشکل فراگیر در همه رشته هاست. به عنوان نمونه، دنیای علم و دانش مثلاً قرار است که یک دنیای باز و آزاد از همکاری و تبادل اطلاعات باشد. ولی استیصال نیاز برای چاپ مقاله و رقابت برای دست یابی به بودجه های تحقیقاتی باعث نتیجه عکس شده است. متفکران بزرگ ایده های خود را با بقیه تقسیم نمی کنند. به سختی در تلاش برای پنهان کردن کارهایشان هستند و در غار تنها یک خود به تحقیق می پردازند و از ایرادهای کارشان بی خبر می مانند. نهایتاً مقاله خودشان را چاپ میکنند و طوری وانمود میکنند که انگار پشت کارشان تلاش و کوشش بی وقفه ای نبوده. نتیجه ها عموماً فاجعه بار هستند: یا به طور اتفاقی کار یک نفر دیگر را تکرار کرده اند، یا کارشان ایرادهایی دارد که متوجهش نبودند، و یا نهایتاً کاری را عرضه کردنده که هر چند در زمان شروع موضوعی ناب و جالب بوده، ولی الان دیگر بی معنی و بی فایده هست. میزان زمانی که توی این کار تلف میشود، بسیار غم انگیز اس. شما یکی مثل بقیه نباشید!

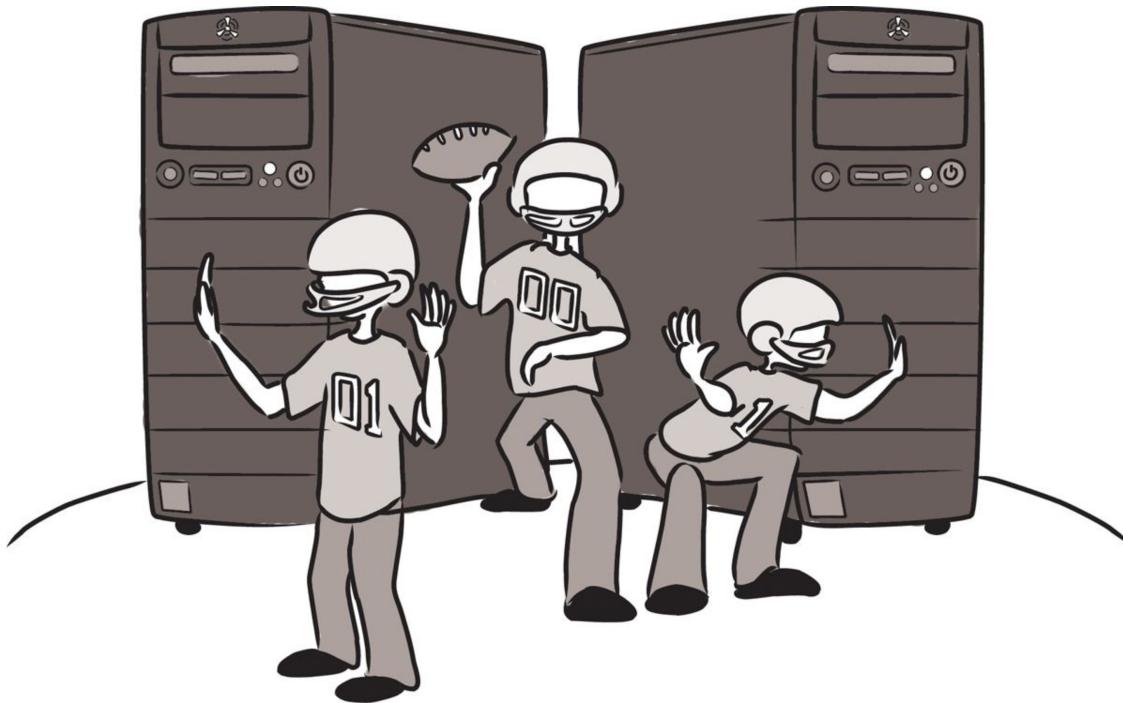
^۱ مخصوصاً اگه واقعاً سازنده دوچرخه باشید:

^۲ البته نظرسنجی بیش از اندازه در ابتدای کار پروژه می تونه خطرناک باشه. در مورد این موضوع در فصل های آینده بیشتر صحبت خواهیم کرد.

مهم، تیمه!

خوب اجازه دهید تا به عقب برگردیم و تمام این ایده هایی که مطرح کردیم را روی هم بگذاریم. اصل صحبت ما این است که در دنیای برنامه نویسی، فردگرایی خیلی به ندرت پیدا می شود. حتی اگر هم وجود داشته باشد، افراد فردگرا هیچ وقت به تنها یی به موفقیت های خارق العاده دست پیدا نمی کنند. پشت سر دستاوردهایی که توانسته اند دنیا را تکون دهند، جرقه یک ایده ناب به همراه تلاش قهرمانانه یک تیم منسجم نهفته است. هدف واقعی، ساختن یک تیم از فوق ستاره هاست، و این کار وحشتناک سخت است. بهترین تیم ها آن هایی هستند که از فوق ستاره هایی که دارند، هوشمندانه استفاده می کنند. عملکرد یک گروه به عنوان یک تیم، همیشه از حاصل جمع کارآئی تک تک افراد حاضر در تیم، بیشتر است.

به زبان ساده تر، برنامه نویسی یک کار تیمی است. ممکن است درک این موضوع سخت باشد چرا که تمام پیش فرض های ما از قهرمان سازی نابغه های دنیای کامپیوتر را به هم میریزد. ولی مثل یک شعار این جمله را تکرار می کنیم: برنامه نویسی، یه کار تیمیه!



وقتی در لانه تنها ی خودتان کد نویسی کنید، نابغه بودن کافی نیست. هیچ وقت با قایم شدن و در خفا روی یک پروژه سری کار کردن، دنیا را تکان نخواهید داد و نتیجه کارتان بقیه را شگفت زده نخواهد کرد. لازم است که حتما با بقیه کار کنید. رویا و هدف خودتان را با دیگران در میان بگذارید. کار را بین افراد تقسیم کنید و از کار همکارهایتان چیزهای جدید یاد بگیرین.

چه تعداد نرم افزار موفق یا قطعه ای از نرم افزار را می توانید در دنیا نام ببرید که بصورت گسترده توسط خیلی از افراد استفاده میشوند ولی حقیقتاً فقط و فقط توسط یک نفر نوشته شده باشند؟^۱ این شعار که برنامه نویسی یک کار تیمی است را بارها در جاهای مختلف این کتاب تکرار خواهیم کرد. یک تیم منسجم و کارآمد مثل طلا نایاب است و کلید اصلی دروازه موفقیت به شمار می آید. تمام تلاش شما باید در راستای رسیدن به این هدف باشد. و این دقیقاً موضوع اصلی کتابی است که در دست شماست.

^۱ ممکنه یک نفر بگوید LATEX!

ولی خوب به سختی می توان گفت که LATEX توسط کلی آدم در دنیای نرم افزار استفاده می شود. مگر اینکه

نویسنده‌گان مقالات علمی را هم در دنیای کامپیوتر و برنامه نویسی لحاظ کنیم.

ستون های سه گانه

پیش تر، منظورمان را از کار تیمی در مهندسی نرم افزار بیان کردیم. اگر کار تیمی، بهترین راه برای تولید قوی ترین نرم افزارهاست، چطور می توان یک تیم خوب را درست یا پیدا کرد؟ کار ساده ای نیست. برای رسیدن به مدینه فاضله در کار تیمی، ابتدا لازم است که سه مهارت اجتماعی که ما اسمشان را "ستون های سه گانه" میگذاریم را به خوبی یاد بگیرید و عمل کنید. این سه مهارت، فقط عامل روغن کاری چرخ های روابط اجتماعی نیستند. بلکه شالوده و بنیادی هستند که روابط تیمی سالم روی آنها استوار می شوند.

تواضع:

- شما مرکز دنیا نیستید. شما نه معصوم هستید و نه همه چیز دان. شما دید بازی نسبت به بهبود شخصیت خود دارید.

احترام:

- شما از ته دل به همکارانتان اهمیت می دهید. با آنها مثل انسان رفتاری میکنید و قدر مهارت ها و دستاوردهای آنها را می دانید.

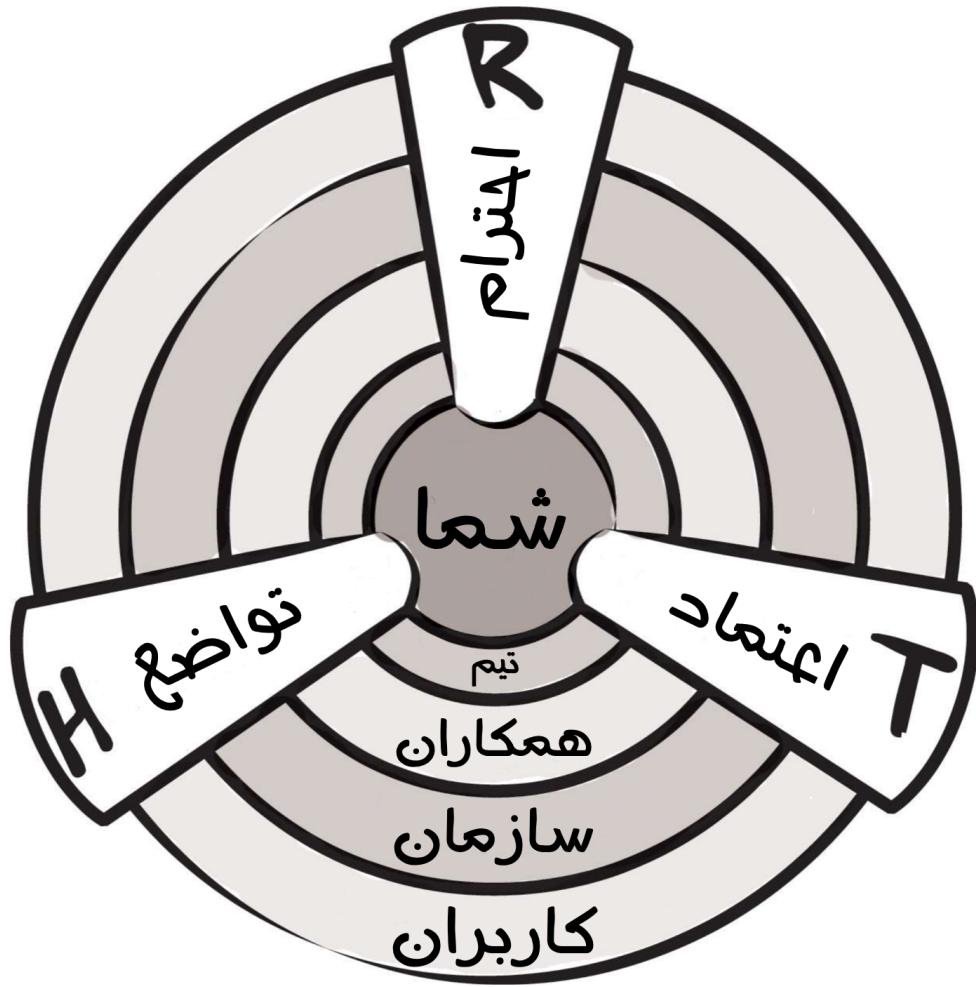
اعتماد:

- اطمینان دارید که سایرین، انسان های شایسته ای هستند که کار خودشان رو بلدند. شما به آنها این فرصت را می دهید که هرجا که مناسب بود اختیار امور را در دست بگیرند.

به مجموعه این اصول سه گانه، به اختصار ^۱HRT میگوییم و آن را هارت (Heart به معنی قلب) تلفظ میکنیم و نه هرت (Hurt به معنی آسیب و درد). چرا که هدف از این اصول، کاهش درد و آسیب رسانی به یکدیگر است.

قضیه ای که ما در طول این کتاب سعی در اثباتش داریم به همین سه ستون بر میگردد: تقریباً ریشه تمام درگیری های ارتباطی در تیم ها را می شود در عدم یا کمبود حداقل یکی از اصول تواضع، احترام، و یا اعتماد پیدا کرد. ممکن است ادعای سنگینی به نظر بیاید، ولی بیشتر به آن فکر کنید. یک رابطه اجتماعی زشت که باعث ناراحتی شما در زندگی شده را به خاطر بیاورید. در بطن این مجادله، آیا همه طرفین به اندازه کافی فروتنی و تواضع به خرج دادند؟ آیا رفتار آدم ها با هم بر پایه احترام و ملایمت بود؟ آیا بین افراد دخیل در صحبت، احترام دو طرفه ای وجود داشت؟

این اصول برای ما آن چنان حائز اهمیت هستند که کل کتاب را بر اساس آنها طبقه بندی کرده ایم. کتاب با تمرکز روی شما شروع میشود تا شما را قانع کند که اصول HRT را درون خودتون تقویت کنید و آنها را مرکز توجهتان در ارتباطاتتان قرار دهید. فصل اول کتاب در همین راستاست. در فصل دوم درباره ساختن یک تیم بر اساس ستون های سه گانه HRT صحبت خواهیم کرد. سپس درباره ساختن فرهنگ تیمی صحبت می کنیم که نقطه ای عطفی در فرایند ساختن یک تیم رویایی است. در مرحله بعدی تمرکزمان را روی آن دسته از همکارهایی میگذاریم که اگرچه ممکن است عضوی از هسته اصلی تیم ما نباشدند، ولی به طور روزانه با ما در ارتباط هستند. مثلا همکارانی که در تیم های دیگر کار میکنند یا داوطلبانی که صرفاً به پروژه ما کمک می کنند. خیلی از این افراد ممکن است نه تنها به اصول HRT پاییند نباشند، بلکه حتی شاید به طور اصولی انسان های سمی و خطربناکی باشند! اولین قدم این است که یاد بگیرید چطور تیمان را در مقابل این دسته از آدم ها محافظت کنید. هدف نهایی خارج کردن نیش و دندان این آدم ها از فرهنگ تیمی شماست، که خود این کار می تواند یک راه حل عالی برای گسترش تیمان باشد.



خیلی از تیم ها، عضوی از یک شرکت بزرگ هستند که خود محیط و فرهنگ شرکت می توانند برایشان به اندازه آدم های سمنی، دست و پاگیر و مخرب باشد. این که بلد باشیم که چطور بین مواعن بازدارنده محیط یک شرکت مانور بدھیم، می تواند مستقیماً به روی موفقیت یا عدم موفقیت تولید یک محصول نرم افزاری تاثیر بگذارد.

و در نهایت، نباید کاربران نرم افزارتان را فراموش کنید. گاهی خیلی راحت وجود آنها را از یاد می بردیم ولی کاربران، خون جاری در رگ های پروژه شما هستند. نرم افزار شما بدون افرادی که از آن استفاده کنند، هیچ هدف و مقصودی نخواهد داشت. تمام اصول HRT که باید بین افراد تیمان رعایت کنید، دقیقاً در نحوه ارتباط شما با مشتری هایتان نیز صدق می کند. فواید این نحوه بازخورد با مشتری، شدیداً زیاد خواهد بود.

یک لحظه توقف کنید!

احتمالاً وقتی این کتاب را برداشتید، انتظار نداشتید که در حال ثبت نام در یک جور گروه حمایتی که هر هفته جلسات دور همی میگذارد باشید. درک می کنیم. سر کله زدن با مشکلات روابط اجتماعی گاهی اوقات میتواند سخت باشد. کار کردن با انسان ها معمولاً آشفته، شلوغ، غیرقابل پیش بینی و آزاردهنده است. به جای تلاش برای برنامه ریزی و سیاست مداری در روابط اجتماعی، آدم و سوسه می شود که کلاً قضیه را فراموش کند. وقتی آدم می تواند با یک کامپایلر منطقی و قابل پیش بینی وقت بگذراند، چرا باید خودش را درگیر مسائل روابط اجتماعی بکند؟

توجه شما را به بخشی از سخنرانی معروف ^۲ ریچارد همینگ جلب می کنم:

با رفتار محترمانه، گفتن جوک و لطیفه و تلاش برای اینکه منشی من بخندد، کمک و سرویس فوق العاده بهتری دریافت میکنم. مثلاً یک روز به یک دلیل احمقانه یک سری سرویس در Murray Hill کار نمی کردند. نپرسید چرا، کار نمی کردند. من هم نیاز داشتم که یک کاری حتماً انجام شود. منشی من با یک نفر در Holmdel تماس گرفت، سوار ماشین شرکت شد، یک ساعت رانندگی کرد و کار مرا راه انداخت و برگشت. به

نوعی جواب همه رفتارهای محترمانه من با او و تلاش های من برای خنداندنش را داد. به وسیله یادگیری و مطالعه دقیق یک سیستم، شما یاد میگیرید که چطور کاری کنید که سیستم در راستای اهداف شما کار کند.

نصحیتی که در این گفته نهفته این است: قدرت روابط اجتماعی را دست کم نگیرید. در مورد فریب کاری یا گول زدن افراد صحبت نمی کنیم. در مورد ساختن روابط مفید اجتماعی در راستای انجام دادن کارها صحبت میکنیم. همیشه عمر دوستی ها بیشتر از عمر پروژه ها خواهد بود.

^۱ حروف ابتدای کلمه انگلیسی برای هر یک از این سه ستون. یعنی: Humility, Respect, Trust

^۲ Richard Hamming - "You And Your Research"

<http://www.cs.virginia.edu/~robins/YouAndYourResearch.pdf>

اصول HRT در عمل

تاکنون این طور به نظر می آید که ما به روی منبر رفته و در مورد تواضع، احترام و اعتماد فقط حرف زده و موعظه کردیم. باید دید که چطور می توان این صحبت ها را در زندگی روزمره به واقعیت تبدیل کرد. در این بخش، به دنبال راهکارهای عملی ای هستیم تا به وسیله آنها یک سری رفتارهای شخصی را بازبینی کنیم. ممکن است خیلی از مثال هایی که می آوریم به نظر واضح و مبرهن بیایند ولی وقتی که در آنها عمیق شوید متوجه می شوید که بیشتر اوقات شما یا همکارانتان خطاکار هستید.

غرورت را کنار بگذار

این جمله با زبان بی زبانی عدم تواضع را به روی آدمی که خیلی رفتارهای فروتنانه ای ندارد، می آورد. هیچ کس دوست ندارد با کس همکار باشد که دائمًا فکر می کند مهم ترین انسان در اتاق است. حتی اگر واقعاً شما باهوش ترین آدم در یک بحث باشید، لازم نیست مدام هوشتان رو در چشم سایرین فرو کنید. به عنوان مثال، آیا در مکالمه ها دوست دارید همیشه حرف اول یا حرف آخر را شما بزنید؟ آیا فکر می کنید لازم است که درباره هر موضوع و مبحثی نظر بدھید؟ یا شاید احتمالاً آدمی را میشناسید که این طور باشد.

توجه کنید که تواضع و فروتنی به این معنی نیست که اجازه دهید بقیه سوار شما بشوند. اعتماد به نفس هیچ اشکالی ندارد. فقط مثل آدم های همه چیز دان رفتار نکنید. همیشه سعی کنید تمرکزان روی برانگیختن غرور تیمی باشد، نه فردی. به جای اینکه نشان دهید که شخصاً آدم فوق العاده ای هستید، سعی کنید حس موفقیت و غرور تیمی را بین همکارانتان تقویت کنید. یک مثال خوب در این راستا، بنیاد نرم افزار Apache است که کارنامه بلندبالای در ساخت تیم های منسجم در تولید نرم افزارهای مختلف دارد. تیم های این بنیاد به صورت فوق العاده ای به همکاری و کار گروهی خود می بالند و اجازه نمی دهند که یک شخص با فردگرایی برای پز دادن از تیم سوء استفاده کند.

غرور خودش را به روش های زیاد و متفاوتی نشان می دهد، و بیشتر اوقات مانع کارایی شما میشود. به قسمتی دیگر از سخنرانی ریچارد همینگ که در این راستا توجه کنید:

جان تاکی^۱ همیشه خیلی معمولی لباس می پوشید. وقتی وارد دفتر کار مهمنی میشد، ممکن بود مدت ها طول بکشد تا بقیه متوجه شوند که او چه شخص مهمی است و بهتر است به صحبت های او گوش دهند. برای مدت زمان زیادی، جان مجبور بود با این طرز رفتار بقیه کنار بیاید. غرور و تکبر کار بیوهوده ای هست. من هیچ وقت نگفته ام که جلوی دیگران کوتاه بیایید. بحث من این است که اگر با بقیه کنار بیایید و خوبزرگ بینی را کنار بگذارید، حتما سودش را می بینید. ولی بر عکس اگر غرور و تکبر را انتخاب کنید و معتقد باشید که همیشه حرف حرف شمامست، به مرور در طول عمر کاری خود هزینه اش رو می پردازید ... یادگرفتن سیستم های اطراف به شما کمک میکند که از آنها در راستای اهداف و نیازهای خودتان استفاده کنید. اگر نه، مجبورید دائمًا در جنگ با سیستم های دور و برтан باشید.

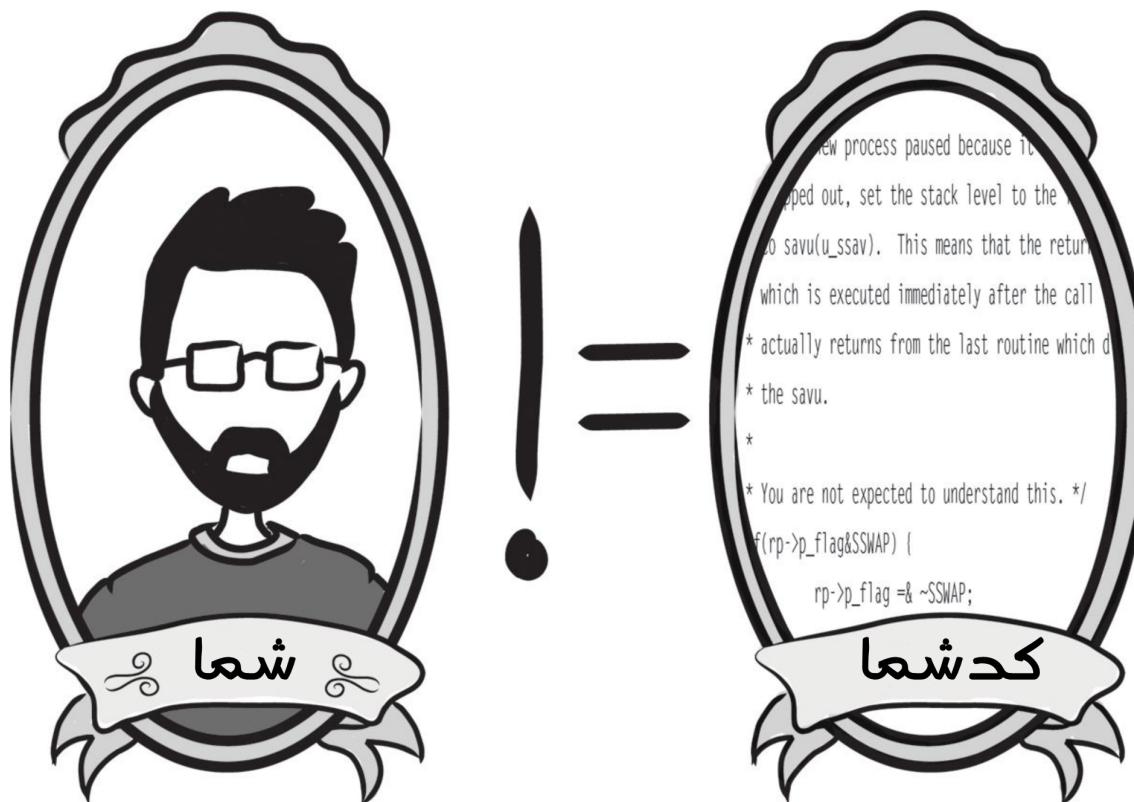
نحوه انتقاد کردن و انتقاد پذیری

داستانی در رابطه با شخص به نام جو که به تازگی کار جدیدی به عنوان برنامه نویس آغاز کرده بود، مطرح میکنیم. او همان هفته اول شدیداً روی کد نرم افزار و پروژه تمرکز کرد. از آن جا که برای کارش اهمیت بالایی قایل بود و به نحوه برنامه نویسی دقت داشت، به مرور از هم تیمی های خودش در مورد کارشان سوال کرد. محترمانه در مورد نحوه طرز فکر و پیش فرضیاتشان اطلاعات بیشتری درخواست کرد و تکه کدهایی را برایشان ایمیل می کرد تا هم نظرشان رو بپرسد و هم موبایله پیشنهاد بدهد که چطور می توانند منطق کشان را بهتر کنند. چند هفته که گذشت، به دفتر رئیس بخش احضار شد. جو با تعجب پرسید: "مشکل چیه؟ من چه کار اشتباہی انجام دادم؟" رئیسش با نگرانی گفت: "ما شکایت های زیادی در مورد تو شنیدیم. جو، به نظر میاد که رفتار تو با همکارانت خیلی تند و خشن است. چپ و راست از آنها ایراد می گیری. از تو ناراحت. لازمه که کمی ملایم تر باشی!" جو مات و مبهوت بود. در یه محیط مناسب و بر پایه اصول HRT ، همکاران جو باید با خوشحالی از نظرات و پیشنهادهای جو استقبال می کردند. ولی در

این مورد خاص، به نظر من آید که جو باید به خاطر عدم اعتماد به نفس همکارهای خودش ملایم تر رفتار من کرد و نظرات و پیشنهادهای خودش رو به صورت لطیف تری مطرح من کرد.

در یک محیط حرفه‌ای مهندس نرم‌افزار، انتقادات هیچ وقت شخص نیستند. بلکه بیشتر اوقات در راستای بهتر شدن محصول نهایی مطرح من شوند. نکته ریز اساسی این است که شما و همکارانتان بتوانید یک انتقاد سازنده نسبت به یک کار ابتکارآمیز را از توهین به شخصیت طرف، تفکیک دهید. حمله به شخصیت آدم‌ها هیچ سودی ندارد و تقریباً غیرممکن است که آدم بتواند با بت آن کاری انجام دهد. ولی انتقاد سازنده، باعث کمک به پیشرفت می‌شود. مهم‌تر از همه یک انتقاد سازنده همیشه همراه با احترام مطرح من شود چرا که براساس علاقه به بهبود کار و پیشرفت طرف مقابل خواهد بود. اگه واقعاً شخص برای شما محترم باشد، همیشه در انتخاب لغات و عبارات دقت من کنید. نراحت در انتخاب کلمات و جملات مفید در انتقاد و پیشنهاد، مهارتیست که فقط با تمرین زیاد به دست من آید.

در مقابل، شما نیز لازم است که هنر انتقاد‌پذیری را یاد بگیرید. یعنی نه تنها لازم است که در مهارت و توانایی‌های خود فروتن باشید، بلکه باید اعتماد داشته باشید که همکارانتان قلباً خیروصلاح و پیشرفت شما (و البته کار و پروژه) را در نظر دارند. برنامه‌نویسی هم مهارتیست که مثل هر توانایی دیگر با تمرین و ممارست بهتر می‌شود. اگر همکارانتان در مورد فرضی توانایی شعبده‌بازی شما پیشنهاد یا نظری میداد، آیا این نظر را یک حمله شخصی به هویت خود تلقی من کردید؟ فکر من کنیم، یعنی امیدواریم که این طور نباشد. به طور مشابه، کدی که شما من نویسید یا محصول خلاقانه ای که خلق من کنید برابر با شخصیت و ارزش هویت شما نیست. تکرار می‌کنیم: شما، کدی که من نویسید نیستید! شما هم این را تکرار کنید: شما، چیزی که خلق من کنید نیستید. نه تنها لازم است که خودتان این قضیه را باور کنید، بلکه باید همکارانتان را هم قانع کنید که انسانها، کدی که من نویسند، نیستند.



به عنوان مثال، این نحوه غلط حرف زدن با یک همکار با اعتماد به نفس احتمالاً پایین است:

"بابا تو کل منطقی که روی این متدهای پیاده سازی کردی اشتباهه، باید از فلان مدل استاندارد که همه هم من دونن استفاده من کردی."

این نحوه انتقاد، پر است از الگوهای اشتباه: اولاً می‌گویند که کارش کلاً "اشتباه" است (انگار که دنیا سیاه و سفید است). طلبکارانه از او میخواهند که کاری که کرده را مطابق نظر شما تغییر دهد و او را متهم می‌کنند که چیزی را ساخته که در تضاد با باور و اعتقاد همه آدم هاست (به او حس حماقت القا میکنید). جوابی که دریافت خواهید کرد، یک عکس العمل عموماً احساسی و تدافعی خواهد بود.

روش بهتر برای انتقال همان منظور ولی با کلمات مناسب‌تر می‌تواند این طور باشد:

"بین این مطقی که توی این متد پیاده سازی شده یکم من رو گیج کرده. به نظرم شاید فلان مدل اگه پیاده سازی بشه باعث خوانایی بهتر کد و نگهداری راحت‌ترش بشه."

دقت کنید که چطور با فروتنی، مشکل را به خودتان مرتبط می‌کنید نه به شخص مقابل. او اشتباهی مرتكب نشده، بلکه این شما هستید که در فهم کد او دچار مشکل شدید. پیشنهاد شما صرفاً برای این است که طرف مقابل به شما کمک کند تا کد را بهتر متوجه شوید و احتمالاً پروژه نیز راحت‌تر به اهداف بلند مدت خود برسد. همچنین، شما از شنونده، طلب کار نیستید. به همکاران این فرصت را می‌دهید که با صلح و آرامش نظر و پیشنهاد شما را قبول نکند. بدین صورت موضوع بحث حول محور کد باقی می‌ماند و وارد موضوع شخصیت و هنر برنامه نویسی افراد نمی‌شود.

یادگیری سریع از اشتباهات

یک داستان معروف و شاید کلیشه در دنیای بازار در مورد مدیری وجود دارد که با یک اشتباه بزرگ باعث می‌شود که شرکتی که در آن کار می‌کند ۱۰ میلیون دلار ضرر کند. روز بعد با ناراحتی وارد دفتر شده و شروع می‌کند تا وسایلش را جمع کند. همان‌طوری که انتظار داشت به او می‌گویند که مدیرعامل در دفترش منتظر است. با ناراحتی و آرامی وارد دفتر شده و یک برگ کاغذ به مدیرعامل تحويل میدهد. وقتی مدیرعامل می‌پرسد که این چیست؟ جواب میدهد: "استعفانامه منه. مگه نمیخواین من رو اخراج کنید؟" مدیرعامل در کمال ناباوری پاسخ میدهد که "اخراج کنم؟! من همین الان ۱۰ میلیون دلار خرج تجربه و آموزش تو کردم. حالا اخراجت کنم؟!"^۲

طمئن‌آ اغراق در این داستان زیاد است ولی نکته این است که مدیرعامل این داستان به خوبی می‌داند که اخراج مدیری که اشتباه کرده، نه تنها ۱۰ میلیون دلار ضرر را بر نمی‌گرداند بلکه باعث می‌شود شرکت یک مدیر خوب که به خاطر تجربه ای که به دست آورده مطمئن‌آ اشتباهش را تکرار نخواهد کرد را نیز از دست بدهد.

یک از شعارهای مورد علاقه ما در گوگل این بود: "گزینه شکست، همیشه روی میز موجوده." باور شرکت این بود که اگر شما هر از گاهی شکست نخورید، یعنی به اندازه کافی در کارتان ابتکار نداشتهید و ریسک نکردید. شکست خوردن، یک فرصت طلایی است برای یادگیری و بهتر شدن در تلاش‌های بعدی. یک جمله که به توماس ادیسون نسبت داده شده، این طور می‌گوید: "من اگر ده هزار راه پیدا کنم که هیچ کدام به نتیجه نرسند، شکست نخورده‌ام. دلسرب نمی‌شوم، چون هر تلاش اشتباهی که تموم می‌شود، یک قدم من را به هدف نزدیک تر می‌کند."

در بخش X Google، که بیشتر پروژه‌های آینده‌نگرانه مثل Google Glass و ماشین‌های خودران در آنجا انجام می‌شود، شکست خوردن تعمدآ جزوی از سیستم تشویقی پیش‌بینی شده است. اعضای تیم‌ها دائمآ ترغیب می‌شوند تا ایده‌های دیوانه‌وار همدیگه را به چالش بکشانند. هر کس به تعداد دفعاتی که در طول یک زمان مشخص، عدم موفقیت یک ایده جدید را ثابت می‌کنه، پاداش می‌گیرد و حتی با سایرین رقابت می‌کند. فقط وقتی که هیچ کس نمی‌تواند تحت هیچ شرایطی یک ایده جدید را رد کنه، ایده وارد پیاده‌سازی اولیه می‌شود.

راز یادگیری از اشتباهات این است که خوب و دقیق تجربیات را ثبت کنید. در دنیای کسب و کار به این نوع بررسی علت و معلول وقوع یک اشتباه، "کالبدشکافی" گفته می‌شود. کالبدشکافی‌های اشتباهات خود را مکتوب کنید. این کار را با دقت مضاعفی انجام دهید و اطمینان حاصل کنید که نوشته شما یک لیست به درد نخور از عذر و بهانه‌های الک نباشد. یک کالبدشکافی مناسب، همیشه درس‌هایی که آدم از اشتباه یاد می‌گیرد را دقیق توضیح میدهد و کارهایی که لازم است در نتیجه این دروس، شروع شوند یا تغییر کنند را لیست می‌کند. حتماً این نوشته را جایی در دسترس قرار داده و مطمئن شوید که کارهای که نوشته شده بیگیری خواهند شد. به یاد داشته باشید که نوشت

اشتباهات کمک میکند تا سایرین نیز (چه الان و چه در آینده) از اشتباهات شما درس بگیرند و تاریخ را تکرار نکنند. ردپاهای خودتان را پاک نکنید. بر عکس آنها را شفاف و پررنگ کنید تا راه نمایی برای بقیه باشند.

گزارش کالبدشکافی خوب، شامل موارد زیر است:

- تاریخچه اتفاقات: از زمان پیدا شدن اشتباه یا حادثه تا راه حل
- عامل اولیه اتفاق یا اشتباه
- برآورد دقیق هزینه و آسیب ایجاد شده
- مجموعه موارد تحت اقدام سریع برای غلبه فوری بر مسئله
- مجموعه موارد جهت اقدام برای جلوگیری از اشتباه مشابه در آینده
- درس هایی که از اتفاق گرفته شده

برای یادگیری، زمان بگذارید

سیندی یک ستاره بود. یک مهندس نرم افزار فوق العاده که حقیقتاً در کارش استفاده شده بود. به راحتی به عنوان مدیرفنی پروژه ارتقا یافت و مسئولیت های بیشتری را روی دوش خودش احساس کرد. خیلی زود، مشاور و مربی اعضای تیمش شد و به آنها راه و روش مهندسی را آموختش می داد. در کنفرانس ها و سمینارهای مختلف سخنرانی می کرد و به آرامی مدیریت چند تیم مختلف دیگر را نیز به عهده گرفت. اگرچه از این که در کارش خبره شده بود لذت من برد، ولی اندک اندک در کارش بی حوصله شد. یک جایی در این مسیر، یادگیری موضوعات جدید را فراموش کرده بود. حس باهوش ترین و ماهرترین بودن در هر جمعی، خیلی آرام تازگی خود را برای او از دست داد. با وجود تمام نشانه های موقتی، جایی یک چیزی هنوز خالی بود. یم روز که مشغول کار بود، متوجه شد که تخصصی که دارد دیگر خیلی در دوره جدید به کار نمی آید. همه درگیر موضوعات تازه تر شده بودند و دنبال تخصص های جدید تر بودند. سیندی چه اشتباهی مرتکب شده بود؟

تعارف که نداریم، خیلی خوش میگذرد وقتی آدم داناترین فرد یک جمع باشد. مربی گری برای بقیه می تواند واقعاً ارزشمند باشد. ولی مشکل این جاست که وقتی شما به یک ماکزیموم نسبی در تیمتان دست پیدا میکنید، یادگیری را کنار می گذارید. وقتی یادگیری را فراموش کنید، حوصله شما در کار سرخواهد رفت. عادت به رهبری تیم خیلی راحت تبدیل به یک اعتیاد می شود. فقط با تواضع و فروتنی می توان مسیر را عوض کرد و راه را برای یادگیری موضوعات جدید باز کرد. مجدداً اینجا هم تواضع و تمایل برای یادگیریست که اهمیت دارند. از حاشیه امن خودتان خارج شوید و خودتان را برای یادگیری و پیشرفت به چالش بکشانید. در طولانی مدت آدم خوشحال تری خواهید بود.

یادگیرید صبور باشید

سال ها پیش فیتز روی پروژه ای برای تبدیل مخزن های کد CVS به Subversion (و بعدها Git) کار می کرد. به قدری CVS بدقلق بود که دائماً فیتز با باگ های جدید مواجه میشد. یک از دوستان قدیمی فیتز، به اسم کارل قبل ها روی CVS خیلی کار کرده بود و به خوبی با آن آشنا بود. این شد که فیتز و کارل تصمیم گرفتند که با هم روی این پروژه کار کنند و باگ ها را برطرف کنند.

وقتی که دونفره شروع به برنامه نویسی کردند با یک مشکل مواجه شدند: فیتز از پایین به بالا به مسائل نگاه میکرد. او در هر مشکلی شیرجه میزد و با سعی و خطأ و امتحان کردن روش های مختلف سعی میکرد که قضیه را حل کند. کارل اما از بالا به پایین صورت مسائل رو می شکافت. او سعی میکرد که به همه متدها و کدهای موجود نگاه کلی داشته باشد و سپس مشکل را پیدا کند. نتیجه این تفاوت در نگاه، کشمکش های مختلف و بعض اجر و بحث های شدید بین این دو نفر شد. به قدری این مشکل برای آنها خسته کننده شد که دیگر حاضر نبودند همزمان و دونفره برنامه نویسی کنند. لازم به ذکر است که آن دو دوستهای دیرینه ای بودند که برای همیگر اعتماد و احترام خیلی بالای قابل بودند. این موضوع در کنار صبر و بردازی بالای اونها باعث شد که بتوانند راه دیگری برای همکاری پیدا کنند. تصمیم گرفتند که با هم و دونقره کد نویسی کنند تا وقتی که به یک باگ برخورد کنند. آن وقت برای حل اون باگ، از هم جدا میشدند تا هر کس تهایی روی مسئله کار کند. وقتی مشکل حل میشد، مجدداً برنامه نویسی دو نفره خودشون را ادامه میدادند. صبوری این دو نفر، همراه با تمایلشان برای همکاری، نه تنها سرنوشت پروژه را نجات داد، بلکه دوستی آنها را نیز پایدار کرد.

تأثیرپذیر باشید

هرچه آدم تاثیرپذیری باشید، شخص تاثیرگذار تری نیز خواهید بود. هرچه آسیب پذیرتر باشید، قوی تر به نظر خواهید آمد. این جمله ها در نگاه اول تناقض های عجیبی به نظر می آیند. ولی همه ما حداقل یک نفر را اطراف خودمان میشناسیم که بی نهایت آدم کله شقی است! هر چقدر هم که آدمها تلاش کنند تا قانعش کنند، همیشه مرغش یک پا دارد. معمولاً سرنوشت این طور آدمها چخ میشود؟ تجربه ما نشان داده که بیشتر اوقات مردم آنها را به عنوان موانع در نظر میگیرند که باید دور زده شوند. بقیه به نظرات یا مخالفت هایشان خیلی محل نخواهند گذاشت. قاعده‌تاً نمی خواهید که شما هم به چنین آدمی تبدیل شوید. پس این موضوع را خوب در ذهن خودتان فرو کنید: هبیج اشکالی ندارد که یک نفر نظر شما را تغییر بدهد. موضوعاتی که حاضرین به خاطر آنها مبارزه کنید را با دقت انتخاب کنید. یادتان باشد که برای اینکه حرف شما به خوبی شنیده شود، ابتدا لازم است که شنونده خوبی برای حرف های بقیه باشید. قبل از این که نظر خودتان را بیان کنید یا در مورد مسئله‌ای با قطعیت تصمیم بگیرید، به نظرات بقیه با دقت گوش کنید. اگه شما دائماً تصمیم خودتان را عوض کنید، آدم بی اراده ای به نظر خواهید آمد.

اما در مورد موضوع آسیب‌پذیری، باید گفت که این مهارت در نگاه اول خیلی عجیب به نظر می رسد. اگر یک نفر به ضعف های خودش یا به عدم دانشش در یک موضوع خاص معرفت باشد، مگر نه اینکه بقیه او را کمتر جدی میگیرند؟ آسیب‌پذیری یک ضعف است و باعث خراب شدن دیوارهای اعتماد میشود. مگر نه؟

خیر، اصلا درست نیست! پذیرش اشتباهاتتان یا اعتراف به اینکه کاری رو به خوبی انجام ندادید، باعث پیشرفت شما در طولانی مدت میشود. واقعیت این است که تمام اصول HRT در این قضیه قرار دارند. پذیرش اشتباه حاصل تواضع و بیانگر مسئولیت پذیری شمارست. همچنین نشانه ای است بیانگر این که شما می توانید به بقیه اعتماد کنید. و در مقابل بقیه هم به شما، صداقت، و قدرتتان احترام بیشتری خواهند گذاشت.

گاهی اوقات بهترین کاری که می توانید انجام دهید فقط این است که بگویید: "نمی دونم."



همه می دانند که سیاست مدارهای حرفه ای هیچ وقت به اشتباه خودشان یا به عدم اطلاع از یک موضوع خاص اعتراف نمی کنند. حتی وقتی که کاملاً برای همه، اشتباہشان واضح و میرهن باشد. برای همین است که هیچ کس یک کلمه از حرفاپیشان را باور نمی کند. این رفتار سیاست مدارها شاید به این دلیل است که دائماً از طرف احزاب مخالفشان تحت حمله هستند. ولی در دنیای مهندسی، هیچ دلیلی وجود نداره که شما دائماً در حالت تدافعی قرار داشته باشید. هم تیمی شما، همکار شماست، نه رقیب شما.

^۱ John Tukey ، ریاضیدان معروف آمریکایی و مبدع الگوریتم Fast Fourier Transform

^۲ روایت های زیادی ازین داستان مطرح شده که به شرکت ها و مدیرعامل های متفاوتی نسبت داده شده اند.

قدمهای بعد

اگر تا اینجای کتاب با ما همراه بودید، در مسیر رسیدن به "هنر کار کردن با دیگران" قرار دارید. شما قطعاً باید با تمرکز و بررسی رفتارهای شخصی خودتون شروع کنید. وقتی مهارت هایی که در این فصل بیان شد را در زندگی روزمره خودتان استفاده کنید، متوجه خواهید شد که همکاری شما با هم تیمی هایتان طبیعی تر خواهد شد و کارآیی تیمتون بالاتر خواهد رفت.

تغییرات مهم همیشه با خود شما شروع می شوند و به دیگران سرایت پیدا می کنند.

در فصل آینده، در مورد این موضوع صحبت خواهیم کرد که چطور می توان فرهنگ تیمی را بر اساس اصول HRT ارتقا داد.