Joel On Software

Fire and Motion

http://www.joelonsoftware.com/articles/fog0000000339.html

Once you get into flow it's not too hard to keep going. Many of my days go like this: (1) get into work (2) check email, read the web, etc. (3) decide that I might as well have lunch before getting to work (4) get back from lunch (5) check email, read the web, etc. (6) finally decide that I've got to get started (7) check email, read the web, etc. (8) decide again that I really have to get started (9) launch the damn editor and (10) write code nonstop until I don't realize that it's already 7:30 pm.

Somewhere between step 8 and step 9 there seems to be a bug, because I can't always make it across that chasm.bike trip For me, just getting started is the only hard thing.

...
When I was an Israeli paratrooper a general stopped by to give us a little speech about strategy. In infantry battles, he told us, there is only one strategy: Fire and Motion. You move towards the enemy while firing your weapon. The firing forces him to keep his head down so he can't fire at you.

...
It doesn't matter if your code is lame and buggy and nobody wants it. If you are moving forward, writing code and fixing bugs constantly, time is on your side.
...
Think of the history of data access strategies to come out of Microsoft. ODBC, RDO, DAO, ADO, OLEDB, now ADO.NET - All New! Are these technological imperatives? The result of an incompetent design group that needs to reinvent data access every goddamn year? (That's probably it, actually.) But the end result is just cover fire. The competition has no choice but to spend all their time porting and keeping up, time that they can't spend writing new features. Look closely at the software landscape. The companies that do well are the ones who rely least on big companies and don't have to spend all their cycles catching up and reimplementing and fixing bugs that crop up only on Windows XP.

Painless Software Schedules

Painless Software Schedules

http://www.joelonsoftware.com/articles/fog0000000245.html

Pek çok yazılım geç sevkiyattan dolayı pazarda başarısız olmuştur: Lotus 123. Yazılımı ilk hazırladıklarında o zaman yaygın olmayan 80386 mimarisine göre geliştirdiler. 16 ay boyunca yazılımın 8086 mimarisine uydurmak için uğraştılar. Bu sırada Excel 16 ay öne geçti ve 8086lar piyasadan kalktı.

Netscape, Apple MacOs ve benzer pek çok yazılım geç teslimattan dolayı pazar şanslarını kaybettiler.

Ancak çoğu programcı hiçbir zizelge yapmaz. Çünkü bu çok zokdur veya hiçbir işe yaramaz.

Kolay çizelge yapabilmek için:

1. Exceli kullan. Basit bir şekilde çalış.

2. Şu sütunlar bulunmalı:

Özellik (Feature), İş (Task), Öncelik, İlk Tahmin, Şimdiki Tahmin, Geçen Süre, Kalan Süre

3. Her bir özellik, çok sayıda işten oluşmalı.

4. Kodu yazacak programcı çizelgelemeyi yapmalı.

5. İnce çözünürlüklü işler tanımlanmalı (2-16 saatlik)

Eğer bir iş 40 saatlikse, yeterince parçalanmamış demektir.

6. İlk ve mevcut tahminleri ayrı ayrı takip et. Böylece hatalarından öğrenebilirsin.

7. Geçen süreyi her gün güncelle. Ancak bu iş 2 dakikayı geçmemeli.

8. Debug etme süresini çizelgeye dahil et.

9. Entegrasyon süresini çizelgeye ekle (kod gözden geçirme gibi)

11. Tampon süreyi ekle.

12. Yöneticilerin programcıların tahminlerini düşürmelerine izin verme.

13. Çizelgedeki işler, tahta bloklardır. Tahta blokları bir kutuya sığmazsa, bazı blokları çıkartırsın.

Bazı özellikleri çıkartmak iki yönden yararlıdır: Zamanı tutturmanı sağlar. Gereksiz özelliklerle yazılımı bozmanı engeller.

Advice for CS Students

http://www.joelonsoftware.com/Archive.html

Even on the small scale, when you look at any programming organization, the programmers with the most power and influence are the ones who can write and speak in English clearly, convincingly, and comfortably. XP ve Linuxu düşün.

By persuading other people, they get leverage. By writing clear comments and technical specs, they let other programmers understand their code, which means other programmers can use and work with their code instead of rewriting it.

I won't hire a programmer unless they can write, and write well, in English.

How Microsoft lost the API War

(By the way, for those of you who follow the arcane but politically-charged world of blog syndication feed formats, you can see the same thing happening over there. RSS became fragmented with several different versions, inaccurate specs and lots of political fighting, and the attempt to clean everything up by creating yet another format called Atom has resulted in several different versions of RSS plus one version of Atom, inaccurate specs and lots of political fighting. When you try to unify two opposing forces by creating a third alternative, you just end up with three opposing forces. You haven't unified anything and you haven't really fixed anything.)

So you've got the Windows API, you've got VB, and now you've got .NET, in several language flavors, and

don't get too attached to any of that, because we're making Avalon, you see, which will only run on the newest Microsoft operating system, which nobody will have for a loooong time. And personally I still haven't had time to learn .NET very deeply, and we haven't ported Fog Creek's two applications from classic ASP and Visual Basic 6.0 to .NET because there's no return on investment for us. None. It's just Fire and Motion as far as I'm concerned: Microsoft would love for me to stop adding new features to our bug tracking software and content management software and instead waste a few months porting it to another programming environment, something which will not benefit a single customer and therefore will not gain us one additional sale, and therefore which is a complete waste of several months, which is great for Microsoft, because they have content management software and bug tracking software, too, so they'd like nothing better than for me to waste time spinning cycles catching up with the flavor du jour, and then waste another year or two doing an Avalon version, too, while they add features to their own competitive software. Riiiight.

Microsoft grew up during the 1980s and 1990s, when the growth in personal computers was so dramatic that every year there were more new computers sold than the entire installed base. That meant that if you made a product that only worked on new computers, within a year or two it could take over the world even if nobody switched to your product. That was one of the reasons Word and Excel displaced WordPerfect and Lotus so thoroughly: Microsoft just waited for the next big wave of hardware upgrades and sold Windows, Word and Excel to corporations buying their next round of desktop computers (in some cases their first round).

The Joel Test: 12 Steps to Better Code

http://www.joelonsoftware.com/articles/fog0000000043.html

SEI´ın hazırladığı SEMA adlı değerlendirme kriterlerinin özeti.

12 tane soru.

0)Do you use source control?
1)Can you make a build in one step?
2)Do you make daily builds?
3)Do you have a bug database?
4)Do you fix bugs before writing new code?
5)Do you have an up-to-date schedule?
6)Do you have a spec?
7)Do programmers have quiet working conditions?
8)Do you use the best tools money can buy?
9)Do you have testers?
10)Do new candidates write code during their interview?
11)Do you do hallway usability testing?

Can you make a build in one step?

Tek bir script ile her şey temelden hazırlanır.

İlk üç adım: Sürekli Entegrasyonun adımları.

Bug Veritabanı:

Buglar sürekli takip edilmeli. Basit olmalı ama en azından şu verileri içermeli:

- complete steps to reproduce the bug
- expected behavior
- observed (buggy) behavior
- who it's assigned to
- whether it has been fixed or not

Yeni kod yazmadan önce bugları düzelt:

Microsoft Word. Ölüm yolculuğu. Sürekli zaman baskısından dolayı sürekli yeni fonksiyonalite eklemek. Ancak bu arada mevcut bugların sürekli yolları tıkaması.

Sakin çalışma ortamı:

Dikkat dağılmamalı. 15 saniye kazanmak için insanlar bölünmemeli. Bir insanın flow denilen yoğun odaklanma evresine geçmesi 15 dakika alıyor.

Koridor kullanışlılık testi:

Koridorda gördüğün insanlara uygulamanı test ettir. Kolay kullanılabiliyor mu, onlar açısından öğren. Kullanışlılık problemlerinin %95´i bu şekilde halledilebilir, diyor yazar.

Bedava UI tasarımı kitabı: http://www.joelonsoftware.com/uibook/chapters/fog0000000057.html

Acısız Bug Takibi

http://www.joelonsoftware.com/articles/fog0000000029.html

Gerçek bir bug takibi uygulaması kullanılmalı diye tavsiye ediyor Joel. Excel tabloları gibi uygulamaların takip etmeyi sağlayamayacağını söylüyor. (**Starteam, FogBugz, bedava yazılımlar**)

Veritabanı geliştiricileri emaille haberdar eder, değişikliklerden.

Örnek bir bug takip raporu:

file:///F:\Belgelerim\Notlarim\Java%20Notlarim\Resimler\Makaleler\Bug.doc

Bir bug raporunda bulunması gereken üç olmazsa olmaz şey:

1. Bugı **yeniden üretmenin** adımları
2. Ne görmeyi **bekliyordun**
3. Bunun yerine **ne oldu**

Bugı açan kişi birine atayabilir. Atanan kişi de çözebilir veya başkasına atayabilir. Ancak bugın kapatılmasını sadece açan kişi yapabilir.

Farklı insanlar çözüldüğünü-onarıldığını düşünerek bırakabilir. Açan kişi tekrar dener. Eğer hala bug duruyorsa, o zaman yeniden aktive eder.

Çözülmüşlük durumları:

fixed, won't fix, postponed, not repro, duplicate, or by design

Versiyon takibinin bulunduğu varsayılıyor.

Acısız Yazılım Takvimleri

http://www.joelonsoftware.com/articles/fog0000000245.html

Basit bir araç kullan. Excel gibi. Project kullanma, çünkü bu bağımlılıklara odaklanır.

file:///F:\Belgelerim\Notlarim\Java%20Notlarim\Resimler\Makaleler\11.gif

Özellik / İş bölüştürmesi yap. Özellikler işlerden oluşsun.

İşler 2 - 16 saat arası olsun. Yani çok ufak.

Çok küçük parçalara bölmek, tahminleri güçlendirir. Çünkü falan metodun ne kadar süreceğini bilmezsin. Ama diyalog oluşturmanın, splash oluşturmanın ne kadar süreceğini tahmin edebilirsin.

Geçen zaman sütununu her günün sonunda güncelle. Bu iş iki dakikada yapılabilmeli. Sürekli saat takibi yapmana gerek yok. Tahmin et. 8 saat üzerinden dağıt, başka işlerle uğraşmış olsan bile.

Eğer işler yetişmiyorsa, yetişecek gibi davranma, zorlama. Bunun yerine öncelikli olmayanları bir sonraki sürüme bırak. (Belki bunlara hiç ihtiyacın olmayacak bile.)