Paul Graham

Great Hackers

http://www.paulgraham.com/gh.html

What do hackers want? Like all craftsmen, hackers like good tools. In fact, that's an understatement. Good hackers find it unbearable to use bad tools.
...
Great hackers also generally insist on using open source software. Not just because it's better, but because it gives them more control. Good hackers insist on control.
...
Big companies think the function of office space is to express rank. But hackers use their offices for more than that: they use their office as a place to think in.
...
Along with good tools, hackers want interesting projects. What makes a project interesting?
Before ITA (who wrote the software inside Orbitz), the people working on airline fare searches probably thought it was one of the most boring applications imaginable. But ITA made it interesting by redefining the problem in a more ambitious way.

I think the same thing happened at Google. When Google was founded, the conventional wisdom among the so-called portals was that search was boring and unimportant. But the guys at Google didn't think search was boring, and that's why they do it so well.

Not: Peki neden bu insanlar sıkıcı problemleri zevkli uğraşlar haline getirmeyi başardı? Çünkü hüsnü zan içindeydiler. Güzel düşündüler, güzel gördüler.
...

The problem is not so much the day to day management. Really good hackers are practically self-managing. The problem is, if you're not a hacker, you can't tell who the good hackers are. A similar problem explains why American cars are so ugly. I call it the design paradox. You might think that you could make your products beautiful just by hiring a great designer to design them. But if you yourself don't have good taste, how are you going to recognize a good designer?

To drive design, a manager must be the most demanding user of a company's products.
Not: Bu fikir, benim iyi bir teknoloji firması kurmak için, önce kendimi iyi bir mühendis olarak yetiştirme amacımı meşrulaştırıyor.
...
It's pretty easy to say what kinds of problems are not interesting: those where instead of solving a few big, clear, problems, you have to solve a lot of nasty little ones. One of the worst kinds of projects is writing an interface to a piece of software that's full of bugs

The distinguishing feature of nasty little problems is that you don't learn anything from them. Writing a compiler is interesting because it teaches you what a compiler is.

So it's not just fastidiousness that makes good hackers avoid nasty little problems. It's more a question of self-preservation. Working on nasty little problems makes you stupid.

have the smart people work as toolmakers - böylece sıkıcı problemlerden uzaklaşır, ancak önemli katkıda bulunur.
...

Along with interesting problems, what good hackers like is other good hackers. Great hackers tend to clump together-- sometimes spectacularly so, as at Xerox Parc.
...
With this amount of noise in the signal, it's hard to tell good hackers when you meet them. I can't tell, even now. You also can't tell from their resumes. It seems like the only way to judge a hacker is to work with him on something.
...
One difference I've noticed between great hackers and smart people in general is that hackers are more politically incorrect. To the extent there is a secret handshake among good hackers, it's when they know one another well enough to express opinions that would get them stoned to death by the general public. And I can see why political incorrectness would be a useful quality in programming. Programs are very complex and, at least in the hands of good programmers, very fluid. In such situations it's helpful to have a habit of questioning assumptions.

Iyi yazılımlar sistematik planlamayla olmaz

http://www.paulgraham.com/usa.html
Paul Graham:

"Systematic" is the last word I'd use to describe the way good programmers write software. Code is not something they assemble painstakingly after careful planning, like the pyramids. It's something they plunge into, working fast and constantly changing their minds, like a charcoal sketch.

In software, paradoxical as it sounds, good craftsmanship means working fast. If you work slowly and meticulously, you merely end up with a very fine implementation of your initial, mistaken idea. Working slowly and meticulously is premature optimization. Better to get a prototype done fast, and see what new ideas it gives you.

Ideas for Startups

What people usually say is not that they can't think of ideas, but that they don't have any. That's not quite the same thing. It could be the reason they don't have any is that they haven't tried to generate them.

...

I also have a theory about why people think this. They overvalue ideas. They think creating a startup is just a matter of implementing some fabulous initial idea. And since a successful startup is worth millions of dollars, a good idea is therefore a million dollar idea.

If coming up with an idea for a startup equals coming up with a million dollar idea, then of course it's going to seem hard.

...

The fact is, most startups end up nothing like the initial idea. It would be closer to the truth to say the main value of your initial idea is that, in the process of discovering it's broken, you'll come up with your real idea.

The initial idea is just a starting point-- not a blueprint, but a question. It might help if they were expressed that way. Instead of saying that your idea is to make a collaborative, web-based spreadsheet, say: could one make a collaborative, web-based spreadsheet?

...

New technologies are the ingredients startup ideas are made of, and conversations with friends are the kitchen they're cooked in.

...

Ideas get developed in the process of explaining them to the right kind of person.

...

What would it even mean to make theorems a commodity? You got me. I didn't think of that idea, just its name.

You can't start with randomness. You have to start with a problem,

Let me repeat that recipe: finding the problem intolerable and feeling it must be possible to solve it.

...

that the best way to solve a problem is often to redefine it.

Redefining the problem is a particularly juicy heuristic when you have competitors, because it's so hard for rigid-minded people to follow.

...

Simplicity takes effort-- genius, even. The average programmer seems to produce UI designs that are almost willfully bad.

So if you want to start a startup, you can take almost any existing technology produced by a big company, and assume you could build something way easier to use.

Hamming: You and Your Research

http://paulgraham.com/hamming.html

In summary, I claim that some of the reasons why so many people who have greatness within their grasp don't succeed are: they don't work on important problems, they don't become emotionally involved, they don't try and change what is difficult to some other situation which is easily done but is still important, and they keep giving themselves alibis why they don't.
...
One of the characteristics of successful scientists is having courage. Once you get your courage up and believe that you can do important problems, then you can. If you think you can't, almost surely you are not going to.
...
When you are famous it is hard to work on small problems. This is what did Shannon in. After information theory, what do you do for an encore? The great scientists often make this error. They fail to continue to plant the little acorns from which the mighty oak trees grow. They try to get the big thing right off.
...
``How can anybody my age know as much as John Tukey does?" He leaned back in his chair, put his hands behind his head, grinned slightly, and said, ``You would be surprised Hamming, how much you would know if you worked as hard as he did that many years." I simply slunk out of the office!
...
What Bode was saying was this: ``Knowledge and productivity are like compound interest." Given two people of approximately the same ability and one person who works ten percent more than the other, the latter will more than twice outproduce the former.
...
The steady application of effort with a little bit more work, intelligently applied is what does it. That's the trouble; drive, misapplied, doesn't get you anywhere. I've often wondered why so many of my good friends at Bell Labs who worked as hard or harder than I did, didn't have so much to show for it.
...
There's another trait on the side which I want to talk about; that trait is ambiguity. It took me a while to discover its importance. Most people like to believe something is or is not true. Great scientists tolerate ambiguity very well. They believe the theory enough to go ahead; they doubt it enough to notice the errors and faults so they can step forward and create the new replacement theory. If you believe too much you'll never notice the flaws; if you doubt too much you won't get started. It requires a lovely balance. But most great scientists are well aware of why their theories are true and they are also well aware of some slight misfits which don't quite fit and they don't forget it. Darwin writes in his autobiography that he found it necessary to write down every piece of evidence which appeared to contradict his beliefs because otherwise they would disappear from his mind.
...
For those who don't get committed to their current problem, the subconscious goofs off on other things and doesn't produce the big result. So the way to manage yourself is that when you have a real important problem you don't let anything else get the center of your attention - you keep your thoughts on the problem.
...
If you do not work on an important problem, it's unlikely you'll do important work. It's perfectly obvious.
...
We didn't work on (1) time travel, (2) teleportation, and (3) antigravity. They are not important problems because we do not have an attack. It's not the consequence that makes a problem important, it is that you have a reasonable attack.
...
The great scientists, when an opportunity opens up, get after it and they pursue it. They drop all other things.
...
I notice that if you have the door to your office closed, you get more work done today and tomorrow, and you are more productive than most. But 10 years later somehow you don't know quite know what problems are worth working on;
...
The technical person wants to give a highly limited technical talk. Most of the time the audience wants a broad general talk and wants much more survey and background than the speaker is willing to give.
You should paint a general picture to say why it's important, and then slowly give a sketch of what was done
...

John Tukey almost always dressed very casually. He would go into an important office and it would take a long time before the other fellow realized that this is a first-class man and he had better listen.
He is going to dress the way he wants all of the time. It applies not only to dress but to a thousand other things; people will continue to fight the system.
...
I am an egotistical person; there is no doubt about it. I knew that most people who took a sabbatical to write a book, didn't finish it on time. So before I left, I told all my friends that when I come back, that book was going to be done! Yes, I would have it done - I'd have been ashamed to come back without it! I used my ego to make myself behave the way I wanted to. I bragged about something so I'd have to perform.

Return of the Mac

http://www.paulgraham.com/mac.html

Quite small, but important out of proportion to its size. When it comes to computers, what hackers are doing now, everyone will be doing in ten years. Almost all technology, from Unix to bitmapped displays to the Web, became popular first within CS departments and research labs, and gradually spread to the rest of the world.

Writing, Briefly

http://www.paulgraham.com/writing44.html

Writing doesn't just communicate ideas; it generates them. If you're bad at writing and don't like to do it, you'll miss out on most of the ideas writing would have generated.

Write a bad version 1 as fast as you can; rewrite it over and over; cut out everything unnecessary; write in a conversational tone; develop a nose for bad writing, so you can see and fix it in yours; imitate writers you like; if you can't get started, tell someone what you plan to write about, then write down what you said; expect 80% of the ideas in an essay to happen after you start writing it, and 50% of those you start with to be wrong; be confident enough to cut; have friends you trust read your stuff and tell you which bits are confusing or drag; don't (always) make detailed outlines;

Undergraduation

http://www.paulgraham.com/college.html

What should you do in college to become a good hacker?

The way to be good at programming is to work (a) a lot (b) on hard problems.

How to Start a Startup

http://www.paulgraham.com/start.html

Üç şey gerekiyor:
- İyi insanlar
- Müşterilerin istediği bir şeyi yapmak
- Mümkün olduğunca az para harcamak

İyi bir startup üçünde de başarılıdır.

Bunların üçü de zordur, ama üçü de **yapılabilirdir**.

Fikir

Ben çoğu zaman rekabetin olmadığı bir alanı bulmaya çalışıyorum. Ama aslında pek çok zaman, mevcut ürünlerden daha iyisini yapabilmek de yeterli bir yenilik kaynağı olabilir. Belki ilk girenlerin sahip oldukları konum ve bilgiye göre geriden başlayabilirsin, ama yine de bu ihtimali de tümüyle gözardı etme. Örneğin, Google.

Temel fikir şu: İnsanların yaptığı bir şeye bak ve bunun beter olmayan bir şekilde yapılma yöntemini bul.

Fikirlerin değeri, sıfıra yakındır. Bunun bir örneği, firmaların zaman içinde ilk iş modellerini değiştirmeleridir.

İnsanlar

İyi insanlar, azimli olanlardır:

What it means specifically depends on the job: a salesperson who just won't take no for an answer; a hacker who will stay up till 4:00 AM rather than go to bed leaving code with a bug in it; a PR person who will cold-call New York Times reporters on their cell phones; a graphic designer who feels physical pain when something is two millimeters out of place.

Programcılar için üç test daha var: Gerçekten akıllılar mı, işi bitirebiliyorlar mı, çekilebilirler mi?

Aslında çekilebilir olmamaları, yeterince akıllı olmadıklarını, ancak kendilerini akıllı göstermeye çalıştıklarını gösterir.

İşletmecilerin startup kurmaları doğru değildir, çünkü iyi hackerlar bulamazlar.

Çoğu teknik adam, iş yürütmenin sıkıcı yönlerinden hoşlanmazlar. Bu yüzden bu işte yetersizlik gösterirler. Dolayısıyla, bu önyargıyı bir yerde kırmalı.

En zengin 500 kişi listesinde, MBA sahiplerinin sayısı son derece az. Teknik arka planı insanların sayısı daha fazla.

Bazıları müşterinin ihtiyaçlarına odaklanmayı yalnızca iş adamlarının yapabileceğini, teknik adamların sadece yazılım geliştirebileceğini, bunları tasarlayamayacağını söyler. Bu çok saçma. Programlamayı bilmek, kullanıcıları anlamayı engellemez.

Bu tip yanlışlar, sanırım istatistiksel korelasyonların ve önyargılarla oluşturulmuş iddiaların, sebep-sonuç ilişkileri yerine geçmesinden kaynaklanıyor.

Müşterilerin İstekleri

Bütün işletmeler açısından en önemli nokta bu. Lokantalar örnek. Bir lokanta iyi yemek yapıyorsa, kötü servis, kirlilik, gürültü her şey hoşgörülür.

Bütün batan startupların ortak özelliği budur.

Bu üç gerekliliğin dışında startupların yapmaları gereken dördünce bir şey daha var: **versiyon biri, olabildiğince erken çıkartmak**. Ancak bu zaten müşterilerin isteklerini tespit etmek için şart olan bir şey.

Hızlı prototipleme her zaman doğru olmayabilir. Ancak startuplar için doğrudur.

Like most startups, we changed our plan on the fly. At first we expected our customers to be Web consultants. But it turned out they didn't like us, because our software was easy to use and we hosted the site. It would be too easy for clients to fire them. We also thought we'd be able to sign up a lot of catalog companies, because selling online was a natural extension of their existing business. But in 1996 that was a hard sell. The middle managers we talked to at catalog companies saw the Web not as an opportunity, but as something that meant more work for them.

We did get a few of the more adventurous catalog companies. Among them was Frederick's of Hollywood, which gave us valuable experience dealing with heavy loads on our servers. But most of our users were small, individual merchants who saw the Web as an opportunity to build a business. Some had retail stores, but many only existed online. And so we changed direction to focus on these users. Instead of concentrating on the features Web consultants and catalog companies would want, we worked to make the software easy to use.

I learned something valuable from that. **It's worth trying very, very hard to make technology easy to use**. Hackers are so used to computers that they have no idea how horrifying software seems to normal people. Stephen Hawking's editor told him that every equation he included in his book would cut sales in half. When you work on making technology easier to use, you're riding that curve up instead of down. **A 10% improvement in ease of use doesn't just increase your sales 10%. It's more likely to double your sales**.

Küçük işletmeler için yazılım satarak başla. Satmak daha kolay olur. Ayrıca küçük bir işletmenin işlerini nasıl yaptığını gözlemle, böylece fırsat alanlarını bulabilirsin.

Ayrıca ucuz bir ürünü, daha güçlü hale getirmek, güçlü bir ürünü ucuzlatmaktan daha kolaydır. Teknolojide, low-end her zaman high-end'i uzun vadede yer. Sun, mainframelerde bunu yaptı. Unutma, bu Swatch'ın yöntemidir aynı zamanda. Alt piyasayı işgal ederek, üst tabakayı korumak.

Aynı işlemi ERP sahasında yapmak neden mümkün olmasın? Hangi yüksek teknoloji alanlarına ucuz alternatifler getirilebilir?

Kaynak Bulmak

Bazı startuplar kendi kendilerini finanse ederler: Microsoft gibi. Ancak bu çok zor, kendini finanse edebilmek için, danışmanlık işleri yapmak gerekir. Bundan ürün şirketine dönüşmek çok zordur.

Startup bir geç/kal koşusudur. Zengin olabilmek için, yaşayabilme ihtimalini maksimize etmek, hisse miktarını maksimize etmekten daha önemlidir. Dolayısıyla hisselerini satabilmek imkanı varsa, bu muhtemelen iyi bir seçenektir.

İlk melek sermayesinin değer verdiği şey, elinizde mevcut olan kodun değeri değildir.

What I didn't grasp at the time was that the valuation wasn't just the value of the code we'd written so far. It was also the value of our ideas, which turned out to be right, and of all the future work we'd do, which turned out to be a lot.

...

We talked to a number of VCs, but eventually we ended up financing our startup entirely with angel money. The main reason was that we feared a brand-name VC firm would stick us with a newscaster as part of the deal. That might have been ok if he was content to limit himself to talking to the press, but what if he wanted to have a say in running the company? That would have led to disaster, because our software was so complex. We were a

company whose whole m.o. was to win through better technology. The strategic decisions were mostly decisions about technology, and we didn't need any help with those.

This was also one reason we didn't go public. Back in 1998 our CFO tried to talk me into it. In those days you could go public as a dogfood portal, so as a company with a real product and real revenues, we might have done well. But I feared it would have meant taking on a newscaster-- someone who, as they say, "can talk Wall Street's language."

I'm happy to see Google is bucking that trend. They didn't talk Wall Street's language when they did their IPO, and Wall Street didn't buy. And now Wall Street is collectively kicking itself. They'll pay attention next time. Wall Street learns new languages fast when money is involved.

You have more leverage negotiating with VCs than you realize. The reason is other VCs. I know a number of VCs now, and when you talk to them you realize that it's a seller's market. Even now there is too much money chasing too few good deals.
...
So I'd advise you to be skeptical about claims of experience and connections. Basically, a VC is a source of money.

You may wonder how much to tell VCs. And you should, because some of them may one day be funding your competitors. I think the best plan is not to be overtly secretive, but not to tell them everything either.

Harcamamak

Patlama sırasında çoğu startup olabildiğince hızlı büyümeye çalıştı. Normalde bu müşteri sayısının büyümesi demektir, ancak onlar için, bu çalışan sayısının büyümesi demekti.

İlk gelen avantajından yararlanmak bile yanıltıcı. Google buna örnek. Kendi markalarını oluşturmak için para harcayan çok sayıda arama motorunu geride bıraktılar.

Sadece marka oluşturarak farklılaşmaya çalışan firmalar, aslında yenilik fırsatlarının bulunduğu sektörleri işaret ederler.

Biz yavaş büyüdük. Bu yüzden her işi yaptık. Satışta iyi değildim, ama teknik destekte iyiydim. Müşteriyle konuşurken hatayı düzeltiyordum. Kulaktan kulağa büyüdüğümüz için, ilk kullanıcı kümemiz bizi **kendi başlarına bulabilecek kadar akıllı insanlardı**. Bir startupın ilk aşamalarında, **akıllı kullanıcılar kadar değerli bir varlık yoktur**. Onları iyi dinlerseniz, kazanan ürünü nasıl yapacağınızı size söylerler.

...

Google understands a few other things most Web companies still don't. The most important is that you should put users before advertisers, even though the advertisers are paying and users aren't. One of my favorite bumper stickers reads "if the people lead, the leaders will follow." Paraphrased for the Web, this becomes "get all the users, and the advertisers will follow." More generally, design your product to please users first, and then think about how to make money from it.

Bir VC firmasından milyonlarca dolar alırsan kendini zengin hissedersin. Ancak değilsindir. Zengin olmak, gelirlerle ilgilidir, yatırımlarla değil.

Profesyonel görüntüye takılmamak lazım. Profesyonellik görüntüyle değil, iş yapmayla ilgili olmalıdır.

En önemli para harcanmayak şey, eleman almaktır. Yeni eleman almak, bir şirketin yapabileceği en kötü şeydir. Çalışanlar, sürekli tekrarlanan masraflardır. Ofis mekanını genişletmeni gerektirirler. Seni yavaşlatırlar. Bir fikri

direkt uygulamak yerine, onunla ilgili toplantı yapmanı gerektirirler. Ne kadar az insan, o kadar iyidir.

İnsanlar niçin böyle bir hatayı bu kadar yoğun yapıyorlar. Bu insanların kendi zayıflıklarını gizlemek için uyguladıkları bir yöntem. Herkes, sizi kaç kiçi çalıştırıyorsunuz sorusuyla değerlendirir.

Etkileyici görünmek, etkileyici olmaktan farklı bir şeydir ve daha önemsizdir.

Siz Yapmalı mısınız?

Bir şirket başlatmalı mısınız? Bunu yapmak için uygun kişi misiniz?

Çok sayıda insan, startup kurmaya zannettiğinden daha fazla uygundur. Ben de öyleydim. Ancak bu fikir en başta beni korkutuyor. Ancak bunu yapmak zorundaydım, çünkü bir Lisp hackerıydım.

Beating the Averages

For years it had annoyed me to hear Lisp described that way. But now it worked to our advantage. In business, there is nothing more valuable than a technical advantage your competitors don't understand. In business, as in war, surprise is worth as much as force.

And so, I'm a little embarrassed to say, I never said anything publicly about Lisp while we were working on Viaweb. We never mentioned it to the press, and if you searched for Lisp on our Web site, all you'd find were the titles of two books in my bio.

The kind of power programmers care about may not be formally definable, but one way to explain it would be to say that it refers to features you could only get in the less powerful language by writing an interpreter for the more powerful language in it. If language A has an operator for removing spaces from strings and language B doesn't, that probably doesn't make A more powerful, because you can probably write a subroutine to do it in B. But if A supports, say, recursion, and B doesn't, that's not likely to be something you can fix by writing library functions.

And if you're writing a short, throwaway program, you may be better off just using whatever language has the best library functions for the task. But in general, for application software, you want to be using the most powerful (reasonably efficient) language you can get, and using anything else is a mistake, of exactly the same kind, though possibly in a lesser degree, as programming in machine language.

Why Smart People Have Bad Ideas

Work people like doesn't pay well, for reasons of supply and demand. The most extreme case is developing programming languages, which doesn't pay at all, because people like it so much they do it for free.

İş yürütmenin kendisi zor değil. Esas problem teknik olarak yetkin olmakta. Çünkü piyasadaki ürünlerin çoğunluğu yazılım geliştirme noktasında zayıf.

Rekabetten kaçınmak için özellikle fakir bir pazar seçmek büyük bir hata.

Wall Street Journal gibi bir dergiyi okumak, çözülmesi gereken problemlerle ilgili fikir verir.

Müşterilerin gereksinimlerini anlamak, hackerların çözdükleri diğer problemlerden daha kolaydır.