Stanford eCorner

Revel in the Adventure of New Ideas [Entire Talk]

Steve Teig, *Tabula*

October 23, 2013

Video URL: http://ecorner.stanford.edu/videos/3203/Revel-in-the-Adventure-of-New-Ideas-Entire-Talk

Steve Teig, president and CTO of Tabula, believes entrepreneurs get the most from life by committing fully to "making work you love." In this expansive talk, Teig shares how to turn fear into a superpower, and weaves together insights from his career to explain the importance of always striving and why life is too short to not work with nice people.

Transcript

Well, thank you very much, Ravi, and thank you for this invitation to speak. I am genuinely honored to be here this afternoon. As you just heard, I am the Founder and the President and Chief Technology Officer of Tabula. Tabula is chip startup building a new kind of programmable device. Basically for those of you who are not so technical, a chip that can change its personality after the fact, where you can take after you manufacture this chip and say now you're powering an MRI machine, now you're powering a networking device and so on. For those of you who are technical, we're the thing after FPGA. And our longer-term agenda is to be the thing after microprocessor. To use this sort of fabric, this general approach to chip architecture to be able to build devices that have millions of objects running at gigahertz rates with a software model that allows multi-programming. Okay. Looking back I have been lucky enough to be the Chief Technology Officer of four successful startups before this one.

Two of which went public and two of which we sold. And most recently as Ravi mentioned that was the Chief Technology Officer at Cadence, a multi-billion dollar company in the software for designing chips. I have to say, I've really enjoyed all of my companies but I have to say also that Tabula is the most exciting and has been the most fun of all. Now as I thought about what I was going to talk about today, I thought well, I can talk about the future of computing which is a lot of what I've been thinking about in the Tabula or how to raise venture capital. I've raised hundreds of millions of dollars of venture capital in my life or even do some magic, although the straightjacket is at home. But decided instead with this being an entrepreneurship seminar that it would be fun to try to offer some precepts, some suggestions to some of the entrepreneurs of the future, for at least my perspective on how to do entrepreneurship. And then I realized well at least looking at the students in the room, I am roughly your parents age and we all know that when your parents say do this and don't do that, nobody pays any attention. The fact is your parents have a context that's causing them to make the suggestions that they are making and lacking that context, they just seem like arbitrary precepts and arbitrary principles and I didn't want to step on that particular landmine. That said, I do have 30 years of experience starting and building companies in a variety of fields and wondered if I couldn't find someway of talking about that entrepreneurship to you guys. So what I decided to do was to use my own history as a framework and to tell some stories based on the different companies I have been a part of, as a basis for presenting some themes, some lessons I've learned which I hope to be controversial at least in part, and yes some precepts with I hope the little bit of context that I'll be providing by showing where these different precepts showed up in my own history.

And yes, I am talking about looking back 30 years. And then I thought to myself, wow when I was a student if somebody had come to me who was my parents' age and said, I am going to tell you sonny about the future of technology and entrepreneurship looking back 30 years. Really, when Harry Truman was President, that that context was in someway going to be relevant to my future, but the fact is as quickly as technology is moving and as quickly as these companies are moving we're still strongly influenced by technology companies that were started 20 years, 30 years, even 40 years ago. Apple, which I

think we'd all agree is still a forefront company, was started in 1976. So more than 35 years ago. Oracle, 1977. Microsoft, which is still having an influence in 1975. Cisco which is still the basis of the hardware of the internet, 1984, so 30 years ago. Dell, also 1984. NVIDIA responsible for this auditorium, 20 years ago and they are certainly a leading edge company.

Linux was developed in 1991. And even Google which I think we would all agree is a forefront company is already 15 years old. So I don't feel too guilty about talking about stuff in 20 years or 30 years ago. What I am going to try in this story is to follow, is to illustrate a few beliefs such as the following rather long list but I'll repeat it at the end when it will make more sense I hope. I think it's important to try to change the world and I've tried to run my life that way. I've always started with a problem rather than with a technology. A problem that I think matters and that I think I might be able to solve particularly well, that matches whatever particular skills I might be able to bring to the table. To the extent I've developed technologies and I've tried to develop quite a few, I focused on technologies that I think have broader applicability, than their original area of application so that I can build a portfolio of technologies and technical approaches that I can use with each new thing that I try and take them. One of my core precepts is apply everything you know about everything, to every problem you encounter. And what that means is you want to try to find ideas you can borrow from other domains that you can apply to the problem at hand.

The more stuff you know, the more readily you can do that, so learning all the time, reading all the time, all kinds of stuff is important because you never know where you're going to see a connection that you didn't see before, that perhaps nobody else saw before. I also think it's important to seek rich and compelling metaphors. I think if you can find the right way to look at something, a good metaphor for the problem that you're working on, the math and technology and computer science will follow. I think it's important to challenge assumptions, other peoples assumptions and your own all the time. And it's not enough to have good ideas. You've got to build great products, craft matters, infrastructure matters. They are worth investing it. Make your products sparkle and have their physical implementation manifest what you think is beautiful about the underlying idea. Redirect your fear. When Michael Jordon was shooting the baskets, I can't imagine he was thinking to himself, oh my god, what if I missed this shot, I am going to look like an idiot on television.

Just shoot the damn basket. Focus your energy on the thing you are actually working on and you can give a 100% of your effect to that rather than protecting your ego, just in case. It's important to surround yourself with great teams, most of things one wants to take require a whole bunch of people. And that team should be talented, passionate, should be teaching you stuff and learning stuff from you and also they better be nice, because you're going to spend a lot of time with them and it's easy to undervalue niceness, but it actually matters. And finally, challenge yourself and your colleagues and have fun. Okay. With that long list let me jump in. I started my undergraduate career doing mathematics at Princeton. And I was okay at it, but at a certain point I had this horrifying thought. Suppose it turns out that I am great at mathematics and I am going to publish papers in algebraic number theory that six people in the world can understand.

What kind of life is that. And I felt for myself, that wasn't enough leverage and it made me not want to be a professional mathematician. Now it also happens that the Apple 2 came out when I was a freshman in college. And I could see immediately that was going to change everything, that computing was going to become pervasive that I needed to know more about this. So I bought an early Apple 2 become an early expert with the Apple 2 and did consulting for the educational testing service, the company that makes SATs. In fact my work there is on their front page, the cover of their annual report in 1981, I think. But I also could see, I was interested in lots of different stuff that computing was going to prevail a lot of different stuff, so I figured I could use computing to change careers whenever I felt like it. And I run my life exactly that way and that's why I switched to electrical engineering and computer science. The first job after school was at a startup company, a really ambitious startup company called Trilogy which was started Gene Amdahl and I joined them in 1982 as a contractor. Trilogy was doing what's called wafer scale integration, trying to make chips that were 10 centimeters on the side and trying to build an IBM compatible mainframe, a more reasonable thing to do in 1982 than now, using such technology.

And let's say they made me an offer I couldn't refuse and convinced me to join them as a contractor. In order to build this complicated chips and this complicated system, they had to simulate everything. And so they had lots and lots and lots of schematic diagrams describing the system they were trying to build and they would simulate that system with what's called a logic simulator. They asked me to write a compiler for the language that controlled their logic simulator, fine. When I looked at what they were trying to do with stimulation, I could see that simulating these schematic diagrams and running programs were really the same thing in disguise. Yes, schematic diagrams were a stranger computer language than C or in those days PASCAL, but they were programs nonetheless. And so what occurred to me, that if I only could up convert these schematic diagrams into PASCAL programs, I could use the PASCAL compiler with its optimizations and produce a custom simulator for everything one wanted to simulate that would be much, much higher performance and much simpler. And sure enough we built this thing and it was 700 times faster than the simulator that preceded it. And the technique we developed now called compiled code logic simulation is still, 30 years later, the standard technique for logic simulation. The things to observe here are that the parts of the system that hadn't been fully designed could still be described in C or PASCAL and so it allowed one to combine, low level descriptions of the parts of the hardware that were fully understand in schematics and abstract descriptions of the rest of the system and simulate them altogether to get high performance and also high generality.

The metaphor or viewing schematics as programs was the key that drove the whole effort. And craft matters, I would say in retrospect maybe 100X of that 700X came from the idea of compiled code and the remaining 7X is that we coded it, fanatically, very, very carefully to get all the performance out of it. It's important to build things carefully and well. Now one of the things from my days at Trilogy is that in 1983, there was an article in Science on what is now called simulated annealing, at that time a brand new general purpose combinatorial optimization technology. So a way of taking a function with 100,000 variables, a very complex function, and being able to try to find a near global optimum. The moment I saw the paper, I could see this was going to be a transformative technology and that I needed to learn a lot about it. The metaphor basically, their metaphor is you have this 100,000 variable function. Imagine that you represent this by 100,000 particles where the energy function of that system of particles matches the function you are trying to optimize. Simulate the system at a very high temperature and then very slowly cool it down until the whole thing coalesces into a very low energy crystal. This struck me as a really beautiful and powerful idea, but I also wanted to challenge their assumptions, because the fact is that the next level of questions one wanted to ask, in my opinion, were all inadequately answered in those early influential papers.

Really, what temperature should you start at? How much should you change the temperature by? How do you decide when you should change the temperature, once you do stuff? What kinds of transformation should you? Etcetera, etcetera. So I started working on the mathematics for that starting with their metaphor and trying to build a technology for general-purpose optimization. Now this served me well, because even though Trilogy went public, it became clear that they weren't going to be the transformative company we had all hoped and so with four other folks, three of them were colleagues at Tangent including my boss there, we started a company called Tangent at which I became the CTO. Tangent was a company that built software for chip design. At that time, there was the very beginning of a sea change in chip design. Before this time roughly, the way people would design chips is that somebody would manually specify where every transistor on the chip was going to go, where every wire was going to on the chip, completely crazy. I mean it obviously doesn't scale. And so somebody had the clever idea of well why don't we have small collections of transistors to build basic functions like AND and OR and PLUS and we're going to put them in a standard library of cells, cells that can go in rows and we're going to have rows of these components, separated by empty spaces called channels where we're going to put the wires, rows of components, wires, rows of components, wires as a way of making it easier to assemble chips. And the folks that started Tangent realized if only we could automate that procedure, automate the placement of where the cells were going to go and the routing of where the wires were going to go, we could make it possible for anybody to design a chip quickly and really have an impact on the electronics industry. So even though I knew absolutely nothing about placement or routing, they made me the CTO of this company and so I spent the summer of 1984 reading what was at time the entire literature on placements and routing.

I got all the proceedings from all of the relevant conferences and I sat by the pool at my apartment complex with 5 feet of tomes and just read from one end to the other and then we jumped in and tried to build our first product, Tancell, which as it turns out despite it coming from 1984, 1985 has been pretty influential. We built the first commercial analytical placer, which is now the standard technique for large-scale placement today 30 years later. We built the first commercial adaptive simulated annealing system, all that math I'd been doing on annealing paid off and I was able to use it to do optimization within Tancell. And that too is still used today and was used very popularly for about 20 years. So I tried to build technologies with leverage that general purpose optimization I was able to apply to the special purpose problem of placing cells. We had the first commercial timing driven placement route system, the first congestion based placement route system. These are all the standard philosophies for chip design even still today. Unfortunately even though we built Tancell, it worked, it was a good product, it was beating other products that were people were trying to develop. It didn't matter. We were ahead of the market.

And that sea change to move to cells in this way, well it did happen, didn't happen till about three years after we built Tancell. So it was hard to make a really successful business out of this and we had to change course a little bit. Well it turns out there was another sea change happening in chips right about this time, this is in 1986 or so now where the folks who manufactured chips figured out ways that they could put multiple levels of wiring above the transistors rather than beside the transistors. They tried to fill the entire chip with these cells, a sea of cells or sea of gates as it came to be called. You have this jigsaw puzzle worth of cells filling up the space and then lots and lots of wires on top. Well that problem looked much harder than how to place components in rows. Nobody knew how to solve it, the people who invented that technology to make it possible to manufacture, people at other companies we didn't know how to solve it either, I definitely didn't know how to solve it either. But I did know that we had a general-purpose optimization technology that we could use as a bit of science laboratory to figure out how best to attack a difficult problem like this and it paid out. We actually figured out how to do it and we invented modern, what's called area based placement routes, the bases of the technology for how people have laid out chips ever since. We basically took that general combinatorial optimization technology that all the annealing stuff and kept evolving it and evolving it and refining it and improving it and it ended up being powerful enough to solve this problem.

We also built it on top of very carefully constructive scalable infrastructure. At that time, people were making chips with 5,000 cells or 10,000 cells but it wasn't very long, thereafter that there were 200,000 cells or 300,000 cells but we built the system that was able to scale enough, that we were able to ride wave and we more than doubled the capacity of what you could put on a chip of the given size with Tangate to that product. So that became quite successful, which in turn caused

Tangent to be acquired by Cadence, very large company. And the Cadence products from the mid 1980s or early 1990s of Gate Ensemble, Cell Ensemble, which generated more than $3 billion in revenue all have Tangate inside them. As it turns out, I didn't stay at Cadence very long, I started a biotech company instead. Now that sounds a little bit strange perhaps but it's less strange that it sounds. So while I was working on Tangate, this is 1986, 1987 is when people first started talking about the human genome project and I thought, well this is going to be transformative. This is going to be incredibly important and I don't know anything about it. So I went to the Stanford bookstore and got a book on general chemistry and organic chemistry and physical chemistry and molecular biology and genetics. And basically over the course of the next couple of years, tried to teach myself chemistry, biology, genetics, at least at an undergraduate level.

After doing this for a couple of years, I thought well surely the problem of discovering pharmaceuticals has been revolutionized by computer aided design in the same way the chip design has been revolutionized by computer aided design. But no, that wasn't the case at all. While there were very forward thinking people like Michael Levitt here at Stanford who were simulating molecules on the screen and doing science with the computer, nobody was attacking the engineering problem of engineering a pharmaceutical that has given properties using significant help from computation. And I thought to myself, well, why don't I take all the ideas that I've been using to engineer chips, to engineer molecules. The people who are doing this design of molecules are scientists who look at the molecules and think maybe I'll replace that nitrogen with a carbon and see if that makes my molecule more active. And while they are brilliant scientists in many cases and know far more chemistry that I will never know, they weren't engineers and they were trying to solve an engineering problem without using an engineering discipline. So to me engineering, whether you're engineering a bridge, engineering a chip, engineering a software, or engineering a molecule is at its core the same stuff. I think there were five main pillars to how engineering works; optimization, visualization, representation, generalization, which is machine learning, and simulation. And in the case of drug discovery, I figured I could use my placement and routing background treating the atoms as the cells and the treating the bonds as wires to try to find low energy structures for molecule with an optimization technology and it worked. For visualization, you see chemists play with these plastic models, to get a feel for how the molecules work.

But I realized with the computer we could make the plastic model actually simulate the correct forces on the molecule, give real intuition about what a molecule would really do but I wanted to provide an interface where you could grab the molecule on the screen and play with it. Like you know how when you are on the web and you see something in 3D and you move your mouse and it causes the thing to rotate, we were the first product to have that in it, that virtual track ball. For representation, I'd had lots of practice building huge databases for chip information. We used the same ideas to build molecular databases with billions of molecules in them and those molecular database structures are still used today. Generalization, we were perhaps the first to apply machine learning to the problem of drug discovery and we built a system called hypothesis generation that attempted to generate scientific explanations for the data that had been observed. The chemists would give us the molecules they had tried, how active they appeared to be and we would construct comprehensible models that were ranked models of possible explanations of the data that had been observed. And then simulation, using those models to estimate activity from those hypotheses. I know I am going to get a little bit technical, I'll back that off in just a second. Parenthetically, I went to venture capitalists, tried to raise money for this company and Michael Levitt of all people was brought in to do due diligence on me as you might know he just won the Nobel Prize in Chemistry here this year at Stanford and he blessed it thankfully and we did get the company funded, but I still remember this was more than 20 years ago, I still remember one of the questions he asked me which would you rather do win the Nobel Prize in Chemistry or make $5 million? And I thought for a little while and I said, I think I'd rather make the $5 million. Evidently he made the other choice and it worked out for him.

So the BioCAD technology, that was my company, BioCAD. That technology is still widely used. We sold BioCAD, the company to a company called Accelrus which still exists and is public. One of the things I realized in doing this biotech stuff is I was providing software for other people to design drugs. But if you've got a design technology that's actually superior, why would you give it to anybody else, why not use it to design stuff yourself. And so I started a pharmaceutical company. So in mid '90s now, I started a company called CombiChem and where I was the Chief Technology Officer and this was an attempt to change the world, to try to build a better scheme for generating molecules. The basic idea from CombiChem was something I had thought while I was still in BioCAD but didn't have a way to execute at BioCAD. It's an idea that I called Mastermind and I am not going to dive too far into this but I at least want to give a sense of it. Metaphors I think is really important.

The 30 second tour of biochemistry and metabolism is as follows. You have in your body large molecules as molecules go, native protein. And those large molecules pick up smaller molecules on the scale of molecules and do chemistry on them. So it breaks some bonds, makes some bonds and so on. And by a series of such transformations, you can transform food into energy for example, or you can raise or lower your blood pressure and so on. The way most pharmaceuticals work is the idea is you can treat these large molecules as blocks that are picking up a small key and then handing it to the next guy and handing it to the next guy. What you want to do is find some lock where if you impede just that lock you prevent it from doing its job by stuffing some gum inside the lock, you selectively interfere with one biological process like raise in blood pressure. The catch is at least in those days, you don't get to see the lock. You have a biological test of some sort that will tell you how well the key fits the lock but not which parts of the key were good parts, nor what the lock looks like how are you going to design a

molecule under those circumstances. And I realized, this is rather like the game Mastermind, where somebody has a sequence of colored marbles that are hidden from you and you've got to determine what that sequence of marbles is.

So you create a sequence of colored marbles and the other person tells you, here is how many of your marbles were correct approximately. So it gives you a score for how well you did without telling you which part of your guess was a good part. If you're clever enough in designing your experiments which sequences you provide and clever enough in doing the inference based on the interest you get back, you can in fact infer in a systematic way what sequence of colors was hidden from you or you can use very analogous strategy to try to design a key for a lock you can't see. You make a bunch of keys, you try those keys, you see how well they fit and you use what you learnt to pick the most informative next set of keys. You couple an experimental design procedure to design an informative collection of keys with an inference procedure to try to learn from that collection of keys and the catch of course is the other guy can lie in this game, biological assays are very unreliable. So I want to play this 20 questions-like game but you have some probability of lying. Well it turns out that problem has a technical solution and we worked it out and we ended up hiring a bunch of chemists and a bunch of computer scientists and we built this Mastermind like system that we called the discovery engine, where we did what's now called active learning, it's something we invented independently, although a couple of years after Dave Angland thought of it at Yale, no web in those days, that's why I didn't know that until much later, and it turns out that by coupling experimental design with inference you can basically build an automated scientist. What we did in effect was automate the scientific method by the application of modern statistics and modern computer scientists and it worked. In those days, the biggest drug company was Merck. They took seven years on average to discover a drug and their success rate was about one in seven tries.

Our success rate was 11 out of 12 and our longest project took only 13 months, our average was less than a year. So it worked. And we discovered 11 different pharmaceutical candidates in a whole bunch of different areas doing drug discovery for hire for big drug companies. Based on this Mastermind metaphor, we were even beginning to apply this technology to materials design. The company went public. We did a partnership with Coupon Pharmaceuticals. They were impressed enough, they bought the company. We helped them discover a couple of drugs and that impressed Bristol-Myers Squibb enough, they bought that company for $7.8 billion. And that was time for me to go and do something else. And in fact I returned to chip design and I am not going to spend too much time on this, I joined a company called Simplex, another company in software for chip design, where I became its CTO.

And all I will say about Simplex is that I took the machine learning technology that I had developed in a biotech setting and just as I had with BioCAD taken chip ideas and applied them to biology. This time, I took the stuff I did for drug discovery and applied it back to chips. No one had applied machine learning technologies for chip design before as far as we could tell. Well it turns out it worked wonderfully. We were able to infer automatically models using the sort of Mastermind idea that could estimate with much higher accuracy. What the performance of things on a chip were going to be. We were five times as accurate as what preceded us. We were 100 times as fast as what preceded us. This is the leverage of developing technologies with broad applicability. General purpose optimization technology, general purpose machine learning technology, general purpose active learning technology, because they can be applied to problem after problem after problem and give you really powerful general purpose tools to attack even these really scary difficult problems.

At Simplex, certainly we were surrounded by a wonderful team as well. And the other thing we took on beside this higher accuracy business is we challenged the decades old assumption that the wires on chips are allowed to be only horizontal or vertical. We realized that if only you could have other directions, suppose you can have horizontal and vertical and horizontal and vertical that is to say and also 45 degrees and 135 degrees. So have different layers that had four different directions, you could build much smaller, much faster, much lower power chips. And there were all kinds of bogus reasons, you can't do this because none of which turned out to be real and so we had to reinvent the entire flow from front to back and something like 150 patents came out of that, not all of them were mine but a whole bunch from that effort. And we developed a scheme that generalized chip design to include diagonal wires as well as horizontal and vertical ones. We had an IPO in 2001. It was the most successful IPO of 2001. And then in 2002, we were acquired by Cadence from which it appears there is no escape if you do chip design. And I became Cadence's co-CTO.

Now, my job at Cadence was to try to find a vision for the future of Cadence. How do we make Cadence a 10 times bigger company than it used to be. Now Cadence sells software for people to design chips with, but the kind of chips people design with Cadence software are special purpose chips for special purpose applications, a chip in your cell phone, a chip in a Cisco router and so on. That's becoming so time consuming and so difficult even the big companies are scared of that. And I started to worry that it might not be the best thing for Cadence's long-term future to make - to be so dependent on people always jumping to the next technology and continuing to design at a rapid rate special purpose chips. I wondered couldn't you have a programmable chip, couldn't you have a more general purpose chip, where you can use the same chip for many applications by changing its personality. And as it turns out there were already two substantial companies doing exactly that, who have a technology called FPGAs that are basically programmable chips where you can view them with a different personality after they are manufactured. And I thought this is fantastic. FPGAs are going to be part of the future. And I had a look into them in more

detail, what I found was yes, programmability is fantastic and kudos to the FPGA companies for creating it, but the overhead for providing that programmability is absolutely intolerable.

It turns out that both then and now the FPGA companies only provide about 5% of the market for special purpose chips and the reason is that the overhead for programmability makes those programmable chips, incredibly large, incredibly expensive, incredibly slow. So the idea is beautiful, the execution not so much. And it led me to wonder, alright, can we do better, can I think of a way of building a better programmable chip than an FPGA. And it turns out, I came up with a way to do it. And realized that the architecture that made it possible to build a better programmable device, one that's much denser much faster that provides the performance of a special purpose chip and not too much overhead for the programmability whilst still preserving for programmability, that same architecture could be used for a computer and not just a special purpose chip that it provided the basis for a post von Neumann - I think a bunch of you are computer science people - a von Neumann computer, something where you would have a million computing elements or 10 million computing elements running at gigahertz rates where you can still program this in a software like way. So while I really wasn't going to do a fifth company, I just saw this opportunity, first of all the $100 billion opportunity for chips but also the chance to build a computer that went beyond the microprocessor and I had to do it. So I left Cadence and I spent a few months in my living room, trying to figure out how this is going to work and invented a series of technologies that we now called Space Time and Inside and Everywhere. And went off and founded Tabula, of which I am currently the President and Chief Technology Officer. That is building in fact a better programmable device, that can compete not just with these FPGAs sort of classical programmable devices but even with special purpose devices for a wide range of applications, but frankly my real agenda and please don't tell anyone is to build a better computer and to use this technology over the next several years to try to build a framework to get out of the serial von Neumann mindset into a better computing environment. Okay.

Let me take a step back literally. You might wonder, why do I bother starting companies. And my answer to that is because I know I am going to be dead soon. I know you're going to be dead soon and changing the world is really important to me and this is a way to do that. It's also because frankly I think almost all products are just lousy, even products that are supposed to be good are just lousy and it drives me crazy. And there are only so many of those products that any one of us can reasonably take on but I've tried to take on aspects of products we engineer that I think are engineered in a mediocre way and to apply some technology to make them better at least per my own taste. But the other reason I start companies is because frankly it's fun. It's incredibly exciting at least to me. Not just to build products which you can do it any company but to build the culture itself. When you have a small company, you can control collectively, what does it feel like to work at that company.

What kind of people work at this company. Make it a place that's a fun place to be around. And it is surprisingly intimate, exercise, you're not only building art in some sense collectively building wonderful products and building the company, but there is the intellectual challenge of working with these colleagues and frankly the emotional challenge of being spending many hours a day and many, many weeks at this effort. It's a really intense experience and a wonderful one. And I am still close with many of the people at my previous companies because it's hard to describe, there is an intensity to it, that's unique and to me wonderful. And you might also wonder, well how do I choose which company I am going to start and I hope it's clear from this rapid fire walk through my history that I've always tried to solve a problem that matters, excuse me whose solution matches my particular skill set. I've also tried to find a problem with a solution that will give me leverage to solve other problems down the road. So general tools I can use further on. So looking at Tangent, the idea was to help folks implement complex chips that was the problem we went after, but underneath, we developed general purpose, large scale optimization machinery that I've used in all kinds of different applications. At BioCAD the goal was to help people discover pharmaceuticals more quickly but we developed a general purpose body of machine learning technology that again has served me in many different areas.

At CombiChem, we ourselves discovered pharmaceutical quickly. And there the technique was to automate the scientific method, to develop general purpose active learning so we could have a scientist in a box to attack a wide range of scientific problems that came up. At Simplex, the objective was to help folks implement denser and faster and lower power chips, but we had to reinvent the entire chip implementation flow to deal with diagonal wires and ended up with 150 plus patents as a result invented all kinds of stuff. And at Tabula, the problem we're trying to solve is to help folks to design high performance chips, quickly and inexpensively with the underlying technology we're trying to develop of building a scalable post von Neumann computer and its associated software. So let me touch on a couple of the lessons, things I've learnt from the experience to-date. First, I think it's valuable to try to make connections that haven't been made before, even Albert Einstein is quoted as saying that very few ideas are made of whole cloth. So read a lot, read widely and try to find good metaphors that allow you to look at this problem as just a funny way of looking at this other problem, so you can borrow ideas from all over the place and they are not even necessarily other scientific ideas, ways of looking at problems will be a guide for what kind of math should you use, what kind of technology should you use and so on. I think metaphors are underrated and I think they are fundamental. Another point I'll make is that I've learned that making a decision at all and making it quickly and clearly often matters more than making the right decision. The fact of the matter is you're just not that smart, and I am not that smart and Steve Jobs wasn't that smart.

We work on incredibly complicated stuff in an incredibly complicated ecosystem, but the good news is that ecosystem is full of local optima. I just came back from safari in South Africa just a few weeks ago. And you see all kinds of really bizarre creatures when you are on safari in Africa and you see giraffes with their long necks and elephants with their long trunks and the rhinoceros with their horns, warthogs, very strange looking animals and so on. Well let's take the warthog, no the warthog is not a lion and it's not a leopard but it's been successful being for at least 25 or 30 million years. It's found a niche, it is a successful business, so to speak in the ecosystem of the animals that are on the savannah. And yes, it's possible that the original warthogs looked more like pigs and then they were being eaten too frequently by lions and leopards and so some of them were lucky enough to have tusks to defend themselves, excellent. So release 2.0 had tusks and allowed them to be more effective at being warthogs. You don't have to be a leopard, you can be a warthog and have a really successful business if you're willing to respond to your environment quickly. The key is figure out what you want to be, go after that. And if you see that the environment changes in front of you or that you picked wrong, change course a little bit.

There are local optima to be found all over the place, you don't have to be in this ridiculously farsighted visionary to see the whole picture at once. Have the picture, have a vision but be prepared to change your mind. Another thing I'll say in the hope of being the tiniest bit controversial is to some extent I think data is overrated. Now, obviously it may be a waggish about it: in a narrow domain you're trying to improve an algorithm of course data is fundamental, right, you take measurements how else can you tell how you're doing, but the more complex the problem we're going after the less data is going to help. If you're trying to make a big decision about the direction of your company, you cannot get enough data soon enough to make that decision well. And if you sit on your hands and wait in the hope that you'll get enough data to be able to make that decision correctly will be too late and you'll miss the market. At some level, you just need to go for it and see what happens. And yes, when the market response change course, be nimble clearly you don't want to have blinders on, but it's easy to get paralyzed by trying to gather too much data too soon and I think engineers as a community too easily fall into that really toxic trap. And then last of these, in my opinion, the most fun you could have at work is at a change-the-world startup. You're going to be dead soon.

Make it different. To paraphrase a recent movie, a million dollars isn't cool. Well I think a billion dollars isn't cool. Changing the world that's what's cool. And if you change the world to do it in some significant way the money is going to come. Make a difference. And to that end, it's not enough that you tolerate the uncertainty of building a company and tolerate the uncertainty of a startup. You've got to revel in it. It's like being in the roller coaster, when you put your arms up and go "ahh", it has to be fun that you don't know. It has to be fun that you have to respond to the uncertainty that surrounds you of building a company, of building great products, of building a team and changing the world, that's part of the thrill of it and there is just nothing like it as far as I am concerned.

Alright, before I wrap up, let me go back over this list of the precepts I started this talk with, now that you have a little context for why I believe them. As I hope this talk has made clear so far, I try very hard to change the world. It's very important to me and I hope it's important to you. I've always started with the problem that I think matters and that I think I can solve well that if it plays to my particular knowledge base and my particular skills. In developing technologies, I've tried to develop ones that have applicability beyond the current application to build a portfolio of rich technologies that I can then apply to whatever problem I see next. And as I mentioned early in the talk, one of my core precepts is apply everything you know about everything to every problem you'll encounter. So read a lot, seek rich compelling metaphors as a way of looking at problems in new ways and by new perspectives. It's important always to learn but it's also important always to teach, that's also part of the fun of being in a startup company is that constant back and forth teaching as part of the thrill. Surround yourself with a great team. People who are talented.

People who are passionate and people who are nice. I got to tell you, it wasn't until I was 40 that I think that I really embraced that having people that are nice is incredibly important. Earlier in my career too easily hired people that were clearly brilliant and not necessarily fun to work with and at a certain point you just say, you know, life is too short. Constantly challenge assumptions including your own assumptions make great products, craft matters infrastructure matters; invest in both because it will make your product sparkle. Redirect your fear. Use whatever protection for ego, you were going to use that energy, apply it towards the problem and it will make you stronger at attacking that problem, than you yourself think you are, give 100% in that sense. Challenge yourself and your colleagues and have fun. Revel in the adventure. So before I end and turn this over to you guys, I want to add a couple of precepts from someone who had a tremendous influence on me and that's my father. My father was a brilliant technologist and a brilliant executive.

And we were extremely close and he taught me many, many things and sadly he died when I was 13 years old. But before that he taught me many precepts that have served me well all this time. Among them and I remember being in five years old and hearing him say this, is pick what you love, because you're going to spend more time doing that than anything else in your life, more than with your family, more than sleeping, more than doing any of your hobbies, pick something you love. And the second, which he believed till his dying breath, is always try your best and I've always striven to do that as well. All right, with that thank you so much for your attention and I'll take whatever questions you might have. You've created so many new things. Do you think that innovative companies and true innovation comes out of creating new things or can it truly - or can it come out

of making existing innovations better? So the question was can innovation - is innovation always about creating new things or can it be based on existing innovations and making them better. Yeah, because you always have like so many patents and patents and stuff and so... So I think that that question lives on a spectrum rather than being a pair of opposites. As I said earlier, even Einstein says that almost nothing is invented out of whole cloth.

The fact is you're always standing on the shoulders of giants and midgets both. In trying to construct better technologies, you're always borrowing ideas and making connections people haven't made. So to the extent - so I do think it's possible to start with a pre-existing idea and a pre-existing technology if you can extend it in some unexpected way; applying it to a new domain, finding some new wrinkle on it, making a connection between this and that. In fact almost all of the patterns - I wouldn't venture to say all of the patents in the patent office - but certainly almost all of them are of the form, one of these things people already knew about with this particular thing added on top of it. Sometimes it's a dramatic thing and that's cool and sometimes it isn't and that's okay. Steve, can I ask you a question on the fear of jumping into something unknown? Sure. Because I think what's amazing for me is how you've jumped into these unknown fields whole-heartedly. What advice do you have for students or people who are prospective founders who are grappling with the fear of jumping into something that they don't know and also just speaking about the downside risk of jumping into entrepreneurship. Any tactical advice for getting over fear when it comes to doing this? Yeah, I'll say a couple of things about that. The first is - and this has always kind of made me laugh as long as I've lived in Silicon Valley which is 30 plus years now - fortunately for the people who live around here people who don't live around here generally speaking are scared to death to start the kinds of crazy companies that people who live around here start.

They overestimate the true downside of blowing it. So let's go through the exercise. So you go ahead and you start a company and you give it your all and you work on it for three years and at that point you can probably tell whether there might be something there or not. And let's imagine there isn't and the thing just fails. Presumably you were somebody pretty smart and motivated, you got the thing funded, you managed to build a team, you managed to build a product, you're going to get another job. You'll either start another company; you'll be able to join another company somewhere else. The objective risk, at least early in your careers which is where essentially all of you are, the objective risk is low. The worst thing that will happen to you is okay, so you'll get another job. You're going to be reasonably paid while you're at that start-up, you're going to learn a lot. The experience of trying to build a company and of trying to figure out how to make it fly will serve you whether that company flies high or not.

So I think people vastly overestimate the true risk that they are taking. Now with that said, I don't really know a way to say this to you that you will really be able to internalize because I know I wouldn't have accepted it at your age, except having experienced it myself but I'll try to say this anyway. What I have found for myself in being willing really to take the plunge, really not to protect my own ego, really to think I am going to bet it all and if it fails, I am going to feel terrible, is that I am able to do things that I myself don't think I can do. What I found the first time I did that, was that problems that I thought were too hard for me were nearer to my grasp than I thought because the third of my effort - third of my - whatever, of my effort that I was spending protecting myself - oh, what if this goes wrong, all that stewing about all the things that could go wrong, by taking that third of my effort and applying it to the problem, I was now 1.5 times as powerful as I would have been if I hadn't done that. And I found that it was like a super power; that it ended up making me and my team in whom I inculcated the same idea, it made us a more powerful engineering team than we thought we were. So I realize that unless you've done this that might not resonate but it certainly matches my own experience. Being the innovator and the brain child with everything you do and also being the businessman and being the front of these companies, how did you find the balance between being the businessman and being the innovator and was there ever a time that maybe being a businessman is difficult? The question is how did I strike a balance between being a business person and being an innovator and did I ever have difficulty striking that balance; not having enough time for both. It's a good question and the fact is my emphasis has been much more on the technology side than it has been on the pure business side. I have in fact personally raised a lot of money in all of my different companies because I come across as the maniac you see before you with a certain level of passion about the stuff I'm working on and when they bring in experts to look at what we're doing it helps the potential investor to see that we know what we're talking about. But in fact I have chosen that at almost all of my companies to bring in business people, not because I didn't think I could do the business part honestly but because that's not where I get the greatest leverage.

There are other people who are great at business and in fact some who are way better at it than I am. I have more personal leverage I think working on technology. So with that said, as it happens looking at Tabula specifically. When I started the company, I was CEO and I was CEO and CTO. And in retrospect that was a mistake. Because for exactly the reason you are implying, I simply couldn't spend enough time on the two different things and if I had to do it over again, I would choose differently. At the very least, I would have built more of an executive staff sooner with VPs of engineering and stuff like that or I eventually brought in a world class CEO who is our CEO today, Dennis Segers and he and I are partners where he is the CEO and I am effectively the CTO. But I waited longer than I should have to do that and Tabula took longer than it ought to have in part because I made that bad decision. You seem to have educated yourself in a number of different fields which were not what you originally learned about in University. How do you go about doing about? So the question was I have taught myself

about areas that I didn't learn about in college, how did I go about doing that? I would wager you already know the answer to that question.

I would wager that in the classes you are taking that much of the teaching you're doing, despite the excellent teaching obviously in Stanford, much of the teaching you're doing is teaching of yourself. You take text books and study them, you look at things on the web, et cetera, and I do a lot of that as well. I just - I love ideas. I love learning new stuff. And I approach this without any sort of magic. I just read papers, read textbooks. But I really, really enjoy it so I spend a tremendous effort on it because of the thrill of learning new metaphors, learning new ideas, being able to walk in Galileo's shoes for just a moment and see how he saw the world, that kind of stuff. Yes? When is the right time to start a company? When is the right time to start a company? Wow. I'll tell you at least how I've done this. It's a good question and I'll give it a serious answer.

For all of my talking about redirect your fearand I believe that, I think it's trueI am much more careful than it would appear to an outsider in the companies that I have started. The fact is before I've started any of my companies, I was sure. The venture capitalists who invested weren't necessarily sure, the team that joined with me weren't necessarily sure. I was sure. And I don't see that frivolously at all. The fact is, for example with Tabula which is really involved technology, I spent months doing nothing but thinking about that to work out not all of the problems, certainly the hundreds of people we have been working with have contributed immensely to this project over the last years, but at least the top level problems, I got far enough to convince myself I knew I had a solution that would work, that I knew that there weren't giant obstacles in the way and that I could see that there was a finish line out there, even if I hadn't mapped it all the way up. I waited till I was sure. And before I put my ass on the line and go to a venture capitalist and say give me millions and millions of dollars, in my heart of hearts, I can't be certain the company will succeed - there are so many exogenous forces there that I can't control - but I can at least be sure the thing I am saying I can deliver, I can deliver. I know how to build it. I know it's going to work.

I know it's going to be better than any of the solutions I've currently encountered. I spent a long time dotting the i's on what I see the endogenous risks of can we build this product, how much engineering effort is it going to take, how hard are these technical problems really before I make the wager. But once I know I can deliver it, then it's really a matter of do I think there is a market there and that sometimes you can tell and sometimes you can't. It really depends. I try to ask experts and get at least some feel for it. For my comments about data before, there is only so much you can learn but I at least try to have conversations with perspective customers and other experts in the field to get some feel for it. But once I know I can build it and once I have some sense that if I could build it there would be something there, at least for me, that's when I go for it. And can I just go a little bit deeper on that, Steve? Would you start a company right out of college or would you go through an education phase of learning under somebody else and then an execution phase of starting something else? It depends. I don't think there is anything wrong with starting a company right out of college. I think it depends.

If you have an idea and where you can tell you have something that's technically sound, well, that you can tell you're solving a problem that matters, you can tell you have a technically sound solution for attacking it and you have at least some reason to believe by talking to others that if you could only build this thing, it would solve the problem and it would make a different, why not. I mean I think there is value in apprenticing as it were underneath people for a couple of years but I don't know that it's fundamentally required even though I see value in it. Yes, go ahead. Yes. So you're saying you were sure that you felt confident you could start this company but between that and having a finished product there is some different roads you go down. So can you talk about what you've done to move faster through the build, measure, learn feedback loop to try to drive to finish product. Right so let me try to parrot that back. The path from having an idea to having an actually finished product can be full of twist and turns, what have I done to try to accelerate that path of being able to learn efficiently. Part of this - a big of this I think is being really scientific about it. That is many engineers, smart though they are, aren't systematic in their experiments.

They do a run, let's say in the software world, they'll do a run and they will call that an experiment. No. What hypothesis are you testing? What are you going after, what do you hope to learn from this? You have limited resources: use those resources efficiently by asking the best question you can think you ask. So I tried to mitigate what I see as the risks as quickly as possible by building prototypes to a lot of prototyping. Reading papers so I can learn from others' successes and failures. And trying to diagnose carrying if something didn't turn out the way I expected, I really try to dig to the bottom of it and I encourage my colleagues to do the same. Why didn't this work? There is something we're not understanding: why not? What's wrong with our approach? So if you can make every step you take count. It actually doesn't take that many steps often to get to something that works. That's the best I can do. Yes? So it sounds like more than once you've jumped into entirely different industries and taught yourself a lot of the things that were required to innovate in that space.

So I am curious as a student what's the value in institutionalized education that you had at Princeton and what we have here? So the question is that more than once I seemed to have switched fields by teaching myself stuff, so given that what was the value of institutionalized education of the sort that I had at Princeton. Indescribable. Going to Princeton was utterly wonderful as I imagine that going to Stanford is for all kinds of reasons. First of all, being able to spend a substantial fraction of

my time for years in a row studying, learning all kinds of crazy stuff and I took courses all over the place in all kinds of crazy stuff besides math and computer science. I mean really all kinds of crazy stuff. But I also met tons of smart people and I loved the fact - the thing I loved most in some ways about Princeton - is I met all these people who were every bit as passionate about ancient Greece or contemporary music as I was about math and then eventually computer science. So they would hold forth and lecture me about why this particular - why Herodotus is cool, just as I would hold forth and lecture about why operating systems are cool. There is - it helped reinforce this having ideas from all over the place that I can use for future things. Also frankly, Princeton's math department is outstanding. I've got a - and the computer science as well.

I got a very strong grounding in the fundamentals in mathematics anyway and the state of computer science as of that time. Happily computer science is a young enough field. It was actually possible to learn much of what there was at the time 30 years ago and certainly that's served me very well. I've worn you all out. When you talk in metaphor to venture capitalists, how did they take that? So the question is when I talk in metaphor to venture capitalists how do they take that. And the answer is it is one of the absolute best things to do with venture capitalists. It's one of the secrets - and I've had a lot of practice at this - it's one of the secrets to doing a great job with venture capitalists. In my opinion, it's the real reason they should believe me, not just the apparent reason that they do. It's why this technology is really going to work and it isn't because of the equation on the board, it's because of the idea that underlines that equation. It helps them understand for real because the metaphor is the reality for me.

It helps them understand for real why this is going to and why it's going to work. I absolutely depend on metaphor especially with these very technical things in communicating to the venture capitalists. It's been a successful strategy for me but I would have to say it's been a successful strategy for them. I think they're right to want the metaphor. To what extent having been based in Silicon Valley has been a difference for you? I'm sorry, I didn't understand. To what extent having been based in Silicon Valley has made a difference for your career? To what extent is being based in Silicon Valley made a difference for me? Well I haven't been based anywhere else but my opinion is it's been an enormous difference. Silicon Valley is an extraordinary, extraordinary place for all kinds of different reasons, a tremendous assortment of people with passion and technical expertise across the board, a whole venture capital infrastructure and so on. But to my own taste, number one is that Silicon Valley understands, in a way no other place I've ever been does, that the only crime is not to strive. It's consistent with my father's precept of always try your best. If in a more conservative place like New York or Switzerland, not that I - and I love New York and I love Switzerland - but if you fail you look bad and you don't get money again and you don't get to try stuff again.

And so there is an engrained fear of failure that is to some extent rational. In Silicon Valley, if you really tried hard, if you built something interesting and you really tried hard, if it didn't fly, it didn't fly. As long as you strove, as long as you really gave it everything and had rational basis for believing it was going to fly. The community in Silicon Valley is forgiving in a good way and says well, you took a real shot at this, take a real shot at something else and it might fly. I think - I've never seen that anywhere else. What do you think computer chips will look like in 20 years? Like so the quantum computing, is that going to be the future or? So what do I think computer chips will look like in 20 years? Is quantum computing the future? This is kind of a long story. Let's see, what do I think? I think a couple of things. Touch on quantum computing first. I do think quantum computing will matter but I don't think it's going to be everything. I think quantum computing will play some sort of role in co-processing for specific applications.

I am not sure I believe yet that quantum computing will be everything but I would be delighted to be wrong about that. 20 years is an eternity so it's hard to say certainly but I will at least try looking ahead, say oh, 10 years which is about the best hope one might have still. I think silicon will have runs its course. I think we'll be seeing graphene or carbon nanotubes or some other sort of carbon based scheme. Certainly I think we'll see three dimensional assemblages. We're seeing the very beginnings of that already to try to make the wires shorter. More important than that: I think that the philosophy of computer architecture is going through a significant change. It's something I myself am trying to help facilitate but, whether Tabula pulls this off or not, I think computer architecture 10 years from now will be importantly different from now, the approach to software will be importantly different. The belief that we'll be able to take a von Neumann, a serial microprocessor, lash a whole bunch of them together in multi-core and that someone is going to figure out how to program it, I think is nonsense. And that's just not going to happen.

It's not going to happen because it is non-physical to pretend that it doesn't cost anything to get data from a memory into a computing element. And until that's accounted for, either in the programming model or in the compiler or both, we will not be able to scale computing and that's going to change in the next 10 years. We're actually out of time. Oh, we're out? Okay. Sorry, I know we can go on for an hour. But please join me in thanking Steve Teig.