How to conduct technical interviews?

I've interviewed approximately 60 developers since I started working and I still wander how I can improve my skills as a technical interviewer. I've never got a formation that explains how to conduct interviews. During the years, I've changed the way I interview people. I've also lowered my expectations since I'm not working on projects that requires world class developers.

French IT Market

Before speaking about recruitments, I must present the French IT market because it's very a very specific and peculiar market. **In France, technical jobs are underestimated**. It's true in many countries but this phenomenon is highly developed in France.

Only a very few corporations value experience in technical fields, especially in IT. This means that **beyond 35, if you're still a developer you're a "looser"** and you're likely to struggle beyond 40 to find a new job. Your salary quickly stops to grow since you become too expensive for a "simple job a junior can do". A friend of mine didn't get a developer position because he wasn't ambitious enough (meaning he didn't say he wanted to be a manager in 5 years).

As a result, **most good developers switch to management, functional jobs, leave France or become independents** (it's one of the rare way to stay in the technical field but it requires being more than a developer since it involves managing a one-person company).

French IT world is more or less a two-speed market with:

- contractor companies (like Atos, Cap Gemini, Accenture) that sell applications or consultants (meaning technical guys) to other companies,
- "client" companies (like Total, Sanofi, L'Oréal, BNP, Orange) that produce products or services for consumers.

It’s difficult to be hired by “client” corporations because French labor laws are over protective so that it’s very difficult for these corporations to fire a guy even if he is very bad. Instead, they prefer using expensive consultants they can fire whenever they want (the money goes to the contractor company, not the consultant’s pocket). Moreover, French society is elitist, **your Master’s Degree and the University** you studied **are very important to get a job** on these companies, **even 10 years after you get your diploma**. Therefore, many good IT guys end up unwillingly in a contractor company where they work as a consultant for the same “client” corporation during years without the good benefits of the corporation.

Note: there are also small companies and startups in France but many developers end up in a contractor company and a few in a “client” company.

I’m one of the lucky ones who’ve never worked for a contractor company, but I still wander about my future. Most candidates I’ve interviewed were consultants from contractor companies. Since consultants are expensive, client corporations want them to be “ready to use”. **The consultants’ resume is sometimes modified by the contractors** so that it matches exactly the needs of the client corporations. Many times, the consultant doesn’t pass through a technical interview, that’s why the contractor modifies the resume to easily sell the consultant. I have friends in contractor companies, some of them discovered their resume was modified just before an interview. Consultants are sometimes forced by the contractor to take a mission (even it’s far away or they don’t like the technologies of the mission) otherwise they’re fired. In fact, this is how contractor companies can easily fire people or make them want to leave.

Note: It’s not a black and white situation, I’ve met some people happy to work for a contractor company.

Sourcing

Before the interview there is the sourcing. We first need to provide a description of the technical skills were looking for and a description of the project we're working on. Then, the manager sends it to many contractor companies. In the rare cases of employee and intern recruitments, the manager sends it directly to the HR service.

I've often tried to reduce the requirements because it's impossible for someone to fit exactly what we're looking for. For example, in a previous project we asked for **someone who's good at 13 different technologies** including 3 very specific technologies… This kind of WTF requirements is a French particularity called the "**five-legged sheep**". In my current Big Data project, we're only looking for good and motivated java developers.

When we're dealing with consultants, there is no pre-selection done by an HR service, all the work is done by the technical team. The price of a consultant is decided by an HR service and doesn't depend on the technical skills of the consultant (though there are some very rare exceptions).

At the beginning I was carefully reading each resumes but after dozens of them I realized it was taking a lot of time, especially when you have a project to finish. I became a "keyword reader" guy, which is kind of ironic since I became the type of recruiters I used to despise. I look at a resume for less than 30 seconds and if I find it interesting then I look at it more carefully. I know that by doing so:

- I'm losing good candidates that don't know how to write a good resume,
- I'm losing good candidates that matches only a bit with our needs,
- I encourage contractors and candidates to fake their resume.

But I can't afford to spend too much time on this stuff, I'm a developer after all.

If the resume is matching exactly the position requirements I look carefully for the candidate on the Internet because it looks very suspicious. Otherwise, I still look quickly for the candidate on the net. When I have time and I see an unknown technology, I look about it too roughly understand it in order to be able to ask very simple questions.

Now, let’s talk about the interviews!

my experience as a candidate

Before speaking about how I handle interviews, let’s talk about some interviews I got as an interviewee. I haven’t done a lot of interviews but they all helped me to shape the interviewer I became. Here are 2 very different experiences I had as a candidate.

my worst experience

The worst interview I had was for a junior java developer position. At that time, my experience with java was a 6-month internship. I did an HR interview then a one-hour MCQ about java and was called a few days after for a technical interview. The technical interviewer asked me specific questions above my beginner level and was condescending. As a result I lost my self-confidence and wasn’t able to answer simple questions. During this interview, I really felt like a punching ball. If I had to do the same kind of interviews today, I think I would leave during the interview. Of course, I didn’t get the job (are you surprised?).

my best experience

As a candidate, the best interviews I had were at Microsoft. The interviewers were nice and the questions interesting (though it’s subjective). But here comes the problem: I had an HR call, a one-hour online exercise, a one-hour technical call (that I really liked), a 5-round interview (some where very fun), to discover during the middle of the 5-round interview that the job position was mostly about doing SQL queries (whereas I thought it would involve complex algorithms)… In the end I didn’t get the job because they rightfully feared I’d get bored and I was proposed to schedule another round (with less interviews) for a more technical job but I declined because it’s **exhausting** and **time consuming** (and at that time the future of Microsoft was uncertain).

Although I really liked these interviews, I don’t see the need for asking a very

good level in algorithms and data structures if the real job requires 5% of this level.

In my **professional** career:

- I've never coded my own sorting algorithm,
- I've never coded my own self-balancing tree,
- I've never used recursion,
- I've rarely created "touchy" algorithms.

Still, since most of my projects were dealing with a large amount of data, my understanding of time complexity helped me to optimize processes.

On my free time I sometimes look at algorithms and data structures because I like it (I'm currently looking at quantum algorithms, wonderful stuff!). But I understand all the developers that criticize the "google-like" interviews, especially the web developers: **why would you need to know dynamic programming or divide and conquer algorithms to do MVC, MVP or MVVM**?

I understand "google-like" interviews are a way to see how a candidate thinks but **someone who's not used to write algorithms will have great difficulty to answer** and that doesn't mean he's stupid. As a result, many "google-like" candidates are using books or sites (like glassdoor, careercup and geeksforgeeks) to brain dump many exercices.

my experience as an interviewer

For me, a good interview is a **nice technical discussion** where **both** I and the candidate **learn stuff**. At this end of the technical interview (that lasts between 1h and 2h), I must be able to tell if the candidate is good. That leads to another question: What is a good developer?

A good developer

There are as many definitions of a good developer as there are developers. When I started working, my definition was strongly focused on the (academic) problem

solving skills and the ability to quickly find a solution to a problem. But I realized that, unless you're working on a specific field, these skills are not important for day-to-day work. So, here is my current definition.

A good developer is someone who:

- Knows when to ask for help

By that I mean he knows when he's facing something too difficult so that it's better to ask someone than wasting days on a problem. To do so, the developer needs to **know his level** and **not being too proud** to ask for help. Moreover, he also needs to know when he's facing simple problems that can quickly be answered by thinking/google/stackoverflow instead of asking for help and wasting his coworker's time.

- Can quickly understand news languages and concepts

A project involves many technologies that change so a developer must be able to **quickly understand a new technology** (**not mastered it**, most of the time it's useless). I must admit the notion of "speed" is very subjective.

- Is logical

Logic is very important for developers but I've worked with developers who weren't logical. I'm not talking about super logic but good sense. For example, when working with data, we often need to process and filter data. I've seen some developers processing the data before filtering it…

I've worked with a developer who, instead of looking for a way to replace a word in many linux files (for example using sed or even coding a simple program), took 4 days to do it manually …

- Can resolve problems alone and learn from previous problems

I think it's important to be able to face a problem and being able to solve it alone. **If someone always needs help, he reduces the productivity of the team**. Of course, a brainless copy paste from google/stackoverflow is not solving a problem. I've also worked with developers who were asking for help and weren't able to see they had already faced this problem.

- Is willing to improve his code

The notion of “good code” is very subjective. For me a good function has to be read without scrolling. I’ve met people that prefer to put a full algorithm in the same function because it’s more readable for them. Neither of us is wrong, we just don’t have the same definition of what a good code is.

A good developer thinks **how to code** the requirements **in order to be** as **readable and maintainable** as possible. He also needs to be **critical about his work** and **willing to improve himself**.

- Can clearly express his ideas

This point is very important since a developer will often communicate with technical and non-technical guys.

- Has an average level for the technologies I’m looking for*

I don’t think knowing a specific language is mandatory to be a good developer. But, since I mostly interview consultants that are supposedly “ready to work”, they must have at least an average level. Moreover, having a technology in common helps me to see how deep the candidate has looked at the technology and sometimes how he learns.

- Is passionate*

Though I like to see the passion for programming, I can’t blame a candidate if he doesn’t read or do code stuff on its free time. Moreover, working for a contractor company (which is what most French developers do) is passion killer. For me, **passion is not required to be a good developer** but it is to become a great developer, which is not what we need because we’re not doing rocket science (like 98% of the projects in IT). Of course, **being passionate doesn’t imply being good**. In fact, the best people I’ve worked with were doing IT only at work. But they were smart, knew their stuff and stayed longer when needed. Though I couldn’t share this passion with them, it was a real pleasure to work with them (I even became friend with some of them).

That being said, I’m currently working on Big Data where the technologies are new and quickly evolving. That’s why I currently prefer a passionate candidate because he’s more likely to stay up to date on the technologies (but it’s not mandatory).

The way I interview

The number of interviews depends on the type of positions, but most of the time:

- An intern will have an HR interview then a technical interview then an interview with a manager,
- A consultant will have a technical interview then an interview with a manager or all in one,
- An employee will have an HR interview then one or two technical interviews then one or two interviews with managers. I've only done it once since we rarely recruit employees.

I don't have any order in the way I do a technical interview. Sometimes I start by presenting my project, sometimes a let the candidate present himself. Most of the time I let the candidate (or another interviewer) choose the order.

When I have time, I read before the interview about the unknown technologies the guy used. I also prepare some questions when an experience inside the resume seems odd or if I'm interested by an unknown technology.

I often start by asking "anti-bullshit" questions about the technologies the candidate has used.

Then, I like asking broad questions about the technologies/architectures the guy used in his projects. Most of the time, I let the candidate present his experience and ask my prepared questions and/or my broad questions. I don't know what kind of broad questions I'm going to ask before the interview, it really depends on the candidate. But here is what I try to see during the discussion:

- how the candidate learnt a technology/subject,
- how far he learnt a technology/subject,
- the technical/management problems he faced and how he solved it,
- the reasons he chose a solution instead of another,
- the solutions he proposes if I add more constrains to a problem he faced,
- how he works with people,

- if the guy understands what he did during his previous experiences (technically and functionally),
- if the guy is understandable,
- how he behaves when he doesn't know an answer,
- how he behaves when I'm wrong.

I'm a big fan of paper and pencil so that both I and the candidate can express ours ideas. When the candidate can't answer a question, I try to ask other questions that will lead to my initial question. But if he still can't answer, I give him a possible answer I was expecting.

When I present my project, I often tell the candidate he can ask questions if he wants. I like to be asked questions or even challenged. It's a good way to see if the candidate can quickly think about problems and expose his point of view. But it happens rarely, I understand that I'm not a common interviewer so most candidates are not used to ask these kinds of questions.

I don't ask for real code because I don't believe I can learn from seeing someone coding for a short and stressful time-period. I would need at least half a day (a full day would be better) doing pair programming with a candidate on a **real problem** which is too much time for me and the candidate. Since I never tried, I might be wrong. Who knows, maybe next year I'll be asking FizzBuzz and Fibonacci to candidates.

I also don't believe in asking for a side-project. My few side-projects are ill-coded, unfinished and most of them are useless because they're not intended to be read by others and I don't have the time nor the will to write something clean and robust.

Moreover, asking for side-projects filters all the non-passionate developers and the passionate ones without side-projects, which means a lot of potential good candidates. As a candidate, I wouldn't work for companies that asks only for this type of candidates because I'd assume they would me to work on my free time.

Though I haven’t done hundreds of interviews, I’ve faced some situations multiple times.

The bullshiters

I have to distinguish 2 kinds of liars:

- those who lie a little,
- those who are total liars (the bullshiters).

Unlike some countries, we rarely check for references in France.

I don’t mind being lied about an experience if the candidate really has the level of his fake experience. But I’ve sometime interviewed total liars. I hate this kind of candidates because it’s a **waste of time for me and them**. On paper the guys are good but I’m **quickly disappointed** during the interview.

The last bullshiter I’ve interviewed told me he had developed a map-reduce job (Big Data stuff) that does complex mathematical stuff on 250 Giga Bytes of data in 50 seconds. I asked him again to be sure he wasn’t mistaking Giga Bytes with Mega Bytes but no, it was Giga Bytes. So, the guy worked on an overpowered platform that can read and process data at 5 Giga Bytes per second, not bad! Not to mention the overhead of Hadoop (a Big Data techno) that takes at least 5-10 seconds to run a job, plus the network and disk overhead of HDFS (a Big Data file system). So the guy lied to my face but wasn’t smart enough to find a realistic lie. As expected, he didn’t know the key components of a Hadoop cluster after one year “working” with Hadoop (which was twice my experience at the time). Of course, he “did” some machine learning stuff, but it was too complicated to be explained … a total waste of time.

The use of wrong keywords

Again I have to distinguish 2 cases:

- the case where I start to speak about a keyword in the candidate's resume and he quickly tells me that he used this technology/concept only a bit or a long time ago.
- the case where the candidate doesn't understand a keyword in his resume.

I remember interviewing a guy who had worked on a "real-time" banking application during multiple years. In my opinion, real-time applications in Java are one of the rare cases when you really need to understand how a JVM (Java Virtual Machine) works and the underlying garbage collection algorithms. I've never worked on real-time applications but I'm interested by JVM stuff because it uses clever mechanisms (you can read my article about JVM memory zones). So, I was happy to interview the guy to learn new stuff. It turned out he wasn't working at all on a real-time application and he had no idea how a JVM works. His "real-time" constrain was to answer a service call in a few seconds … I was very disappointed but I kept it to myself.

In this kind of situation, I can't directly blame the candidate because I don't know if it's him or his contractor that put the mysterious keyword.

The anxious guy

It's very easy to see if someone is anxious: you can quickly detect it by the tone of his voice, the way he look at you or some repetitive movements (especially the fingers and foots). I often try to break the ice by being kind or asking very simple questions, but it doesn't often work. I find it very difficult to interview a nervous candidate because I can't tell if he doesn't answer because he is stressed or because he doesn't know the answer. I prefer focusing about projects the candidate did and add new constrains to them because I'm in his **comfort zone** and (I think) I'm more likely to get an answer (despite the stress). But I can't do that every time since I need a high level of concentration to challenge someone in his comfort zone.

When I disagree with an other interviewer

I've interviewed alone a few times but most of the times with one or two interviewers (another technical interviewer and/or a manager). And here comes the situations where I disagree with the other interviewer, for example:

- An interviewer was intentionally stressing a candidate to see how he was handling stress. In my opinion, an interview is stressing enough. I was feeling bad for the candidate but I couldn't express my strong disagreement during the interview.
- Another situation I faced was when a candidate didn't understand the other interviewer's questions but neither did I: the questions didn't make sense technically speaking! Again, for the sake of the interview I acted as if I understood the questions and tried to "rephrase" the questions to the candidate.
- the last situation I remember is when another interviewer asked (what I think are) stupid questions like "what's the difference between JUnit 3 and JUnit 4" or asking for a specific word/function/regexp.

Awkward situations

Since the sourcing has always been done as a team, I haven't always decided who we should interview. I've end up a few time with a very awkward situation:

- the situation where I know before the interview the guy won't get the job since his experience doesn't fit with our needs. I was right (the 3 times) and felt sorry for the candidate to waste his time.
- I interviewed a candidate I had strongly refused a year before for the same project. It happened just once but it's a 8/10 on the awkwardness scale.

The last awkward situation is typically French and is due to the way contractor firms behave toward consultants: I've interviewed a few candidates that weren't interested by the job and were doing the interview just because they were forced by the contractor.

When I’m wrong

Being a technical interviewer is not an easy task. You see many candidates with different technical backgrounds. Sometimes when a candidate describes his project, it feels like it’s not possible. I always ask for more precision then skip to another point. But I record exactly what the candidate just told me and check on the Internet after the interview (or ask some coworkers). For example, a candidate told me he was using Struts (a web Java framework) to develop WAP interfaces (the ancestor of mobile internet). During the entire interview I thought the candidate was bullshiting me (and therefore I was biased during the interview) but after looking on the Internet, it turned out it was possible!

Evolution of the process

I don’t know if it’s just the French market but many senior developers I’ve interviewed were at best very average candidates. This is why we recently added in my current project a MCQ with basic to average questions about java and object concepts. If someone doesn’t pass a certain threshold we don’t do a face to face interview (which, I recall lasts 1h to 2h). I don’t really like it because I know some good developers that couldn’t answer this MCQ but it’s a way to filter the bullshiters and the candidates whose resume are “over optimized”.

Unfortunately, the projects I’ve worked on had their ups and downs. That led good developers I had recruited to leave after 6 months (and honestly, I understand them). I now put more weight on the feeling the candidate is likely to leave (which is very subjective).

To conclude

In the end of the face to face interview, if I can say “I could work with this guy” and “this guy seems good” then it’s a yes. I’ve sometime had to struggle for a candidate I believed in but not the other(s) interviewer(s). I’ve never been wrong when I felt a candidate was good but I have a high rejection ratio, that’s why I am lowering my thresold. I’ve sometimes accepted a candidate I wasn’t sure about because we desperately needed manpower and the other(s) interviewer(s) liked the guy, which led to good and bad developers.

If you’ve never did interviews, you now know what’s going on behind the scenes (or at least a possible way).

If you’re a technical interviewer, I’d be glad to read how you do it.

Whether you’re an interviewer or an interviewee, what do you think of the way I do it?