KANER.COM **Cem Kaner, Ph.D., J.D.**[1] **kaner@kaner.com**
160 Rocket Lane #339 **408-244-7000 (Voice)**
W. Melbourne, FL 32904 **www.kaner.com** **408-244-2181 (FAX)**

RECRUITING SOFTWARE TESTERS[2]

Software Testing & Review Conference

San Jose, CA.

October, 2000.

I don't claim to be an expert on recruiting, but I have been reasonably successful at recruiting testers (and other software workers) over the last 17 years, have made a lot of mistakes, and learned a few lessons. This chapter collects some of those lessons.

Managers get work done through other people. The people that a manager chooses are the people who will achieve, or fail to achieve, the mission of the group and the tasks assigned by the manager. Recruiting staff is one of the hardest jobs for any manager. Hiring the wrong staff is, I believe, the worst mistake that a manager can make.

This chapter considers the following issues:

- A behavioral approach to gathering information
- Legal issues in recruiting and hiring
- Consensus-driven hiring
- A strategy for gathering information
- Characteristics of the person you're seeking
- Defining the position and characteristics needed for it
- Who approves the hiring
- How to find candidates
- What if you can't find candidates
- Sifting through resumes
- Evaluating the candidate's public materials
- The phone screen
- The application form

---

[1] Cem Kaner is Professor of Software Engineering at the Florida Institute of Technology. He is senior author of *Testing Computer Software* and of *Bad Software: What To Do When Software Fails*.

Kaner has worked with computers since 1976, doing and managing programming, user interface design, testing, and user documentation, teaching courses on software testing, and consulting to software publishers on software testing, documentation, and development management issues. He is also co-founder and co-host of the Los Altos Workshop on Software Testing and the Software Test Managers' Round Table.

Kaner also practices law, focusing on the law of software quality. He has been active (as an advocate for customers, authors, and small development shops) in several legislative drafting efforts involving software licensing, software quality regulation, and electronic commerce.

[2] I thank Jack Falk for many of the interview questions and for a detailed review and suggestions.

This is the sixth circulating draft of a chapter that will appear in the third edition of *Testing Computer Software*, by Cem Kaner, Bob Johnson, James Bach, Jack Falk, Hung Quoc Nguyen, and Brian Lawrence. This is near-final. The next draft will focus more on legal issues associated with recruiting, which are not competently dealt with in this one. The next draft will also fill in references and incorporate some exercises from Kaner's course on software tester recruiting. ***Comments and criticism are welcome, and should be addressed to kaner@kaner.com***

- Preparing for the interview
- Dividing the issues among staff
- Asking questions during the interview
- Reviewing work samples
- Reviewing sample test documentation (of yours)
- Setting up an audition
- Writing bug reports
- Using tests and puzzles
- Debates and controversial questions
- The problem of free consulting
- The post-interview meeting
- Feedback to the candidate
- Checking references
- Investigation (such as drug testing, credit checking, etc.)
- Red flags
- Making and closing the offer

There have been other thoughtful discussions of the recruiting of testers. I recommend Rothman (1998), Dustin et al. (1999, especially for GUI-level regression testers), and Black (1999).

A BEHAVIORAL APPROACH TO GATHERING INFORMATION (F xxx)

When you interview people, you interact with them. In the process, they reveal things about themselves that go beyond the tidy sections of their well-formatted resumes and the pat answers that they might have prepared for your standard questions. This can give you a better chance of learning how it would be to work with them. I recommend being alert to what candidates reveal to you in their behavior. I suggest that you design interviews to elicit behavior samples.

Here are some examples that come almost automatically during the interviewing process:

- Look at the candidate's resume and cover letter (if there is one) for her structure and organization. Is she sloppy? Disorganized? Does she communicate well? Does the candidate appear to be hiding anything?
- Listen to the message on the candidate's answering machine? Clear? Weird? Rambling? Are there oddities in the message that suggest that the candidate didn't listen to/debug the recording? (Caution: these are typically personal answering machines, and some talented people have creative or eccentric sounding messages. An odd message shouldn't change your decision to interview a candidate but it might still give you more insight into the person, help you develop questions, or interpret answers later in the interviews. Take some care in forming snap judgments. For example, a message that sounds rude, confrontational, and as though it was recorded by an intoxicated person might be just that, or it might be the normal speech of a decent, competent person who has a problem speaking.)
- Look at the candidate's web site and at the signature section of her e-mail messages. This is information that she has chosen to make public. How interesting is it? What does it tell you about how she organizes information, how carefully she gathers information, and how well she maintains the documents that she posts? You can find a lot more about the candidate on the web, way more than you'd expect. See Lane (1997), for example. CAUTION: I suggest that you avoid looking for credit-related or sexual-activity-related information, and that if you find it, you make a point and a policy of not reading it. (XXX ref to lawsuit report on credit reference.)
- When the candidate comes for the interview, is she dressed appropriately? Does she show up on time? If she's late, did she call in advance? Did she bring additional information (such as work samples, letters of reference, publications, etc.) with her? If so, were they relevant and well organized?

Along with these, I ask probing questions about the candidate's experiences. My goal is to get the candidate to tell me about something she has already done, rather than make up a hypothetical answer. So I don't ask,

> "What would you do with a product that came to you without specifications?'

Instead I ask,

> "Have you ever worked on a product that came to you without specifications? Tell me about the challenges this raised and how you handled them. (And then, as a follow-up question,…) What do you think you did particularly well in that situation? (And then…) What did you learn that will help you handle this better in the future?"

Behavioral questioning is a standard approach. Rosse and Levin (1997, p. 173) provide more examples of what they call situational and behavior-descriptive questions. Risser (1993, pp. 150-152) provides more examples and useful discussion of the value of behavioral questioning.

I also give tests. For example, when I interview an experienced tester, I want to learn how well she can write a bug report. So I give her a bug and ask for a bug report.

If you expect them to make presentations during the job, have them make a presentation. (XXX deMarco)

I also do some role playing. For example, when I interview a test manager, I want to learn how effectively she can defend her point of view in a discussion with an authority figure. So I will arrange things so that an authority figure (maybe me, maybe someone else) will challenge her viewpoint on something that she considers important.

In general, to learn how well someone will do something, try to set up a situation that lets you see how well they do it. To learn how someone will respond to something, try to set up a simulation that elicits their response.

LEGAL ISSUES (M XXX)

I'm a lawyer, but I don't know the laws governing interviewing. I cannot give you reliable legal suggestions, but I can point out some (not all) of the issues.

- Laws that were designed to deal with the long history of discriminatory practices by American businesses require employers to determine essential job functions involved in a position that they recruit for. Talk with your company's HR manager. You may have to list the essential job functions in every recruiting ad or help wanted notice placed by your company.
- Several books that I've read tailor their approaches to minimize the chance that the company can be successfully sued for discriminatory hiring practices. These authors would advocate a more rigidly structured approach to interviewing than I do, such as making sure that every candidate is asked the exact same list of questions. I tailor the details of my approach to the human being that I am interviewing. Different humans, with different interests, aptitudes and backgrounds, get different questions. However, this poses a risk—in a lawsuit alleging discriminatory hiring practices, an approach that is tailored to the individual is harder to defend.
- Independently of the legal issues, discrimination on the basis of race, gender, age, etc., is counterproductive for test groups because diversity is one of our key goals. Limiting diversity means limiting a testing group's effectiveness.
- Privacy concerns implicate another body of law. For example, you probably can't check a candidate's credit rating without her knowledge. You probably can't require an employee to take a pre-hiring polygraph (the alleged but far from reliable "lie detector") test.
- Check with HR if you plan to give tests to interview candidates. Under the Americans with Disabilities Act, you might be required to advise candidates of a test in advance, you might be

required to give the same test to all candidates for the position, and you might be required to plan to make accommodations for a candidate who has a disability.

- You must also be careful during the interview process to avoid making unintended promises. For example, suppose that you make a statement to a candidate that "No one here is fired unless they have a substance abuse problem." If you hire this person and he doesn't do any work, he can argue that you can't fire him unless he has a substance abuse problem. Similarly, you have to be careful about statements that you make about the nature of the job itself. You can clear all of this up with an appropriately worded offer letter and employment contract (or you can make life worse with a poorly worded one).

Risser (1993) lays out some of the issues, but you should recognize that even when you are dealing with an excellent book, laws differ across states and change over time. (xxx cross-reference matthew bender treatise on california employment law.) For guidance on the legal issues, consult your company's HR department. That's one of the reasons that you have an HR department.

CONSENSUS-DRIVEN HIRING (F xxx)

I follow three simple rules when hiring:

- Anyone in the company who wants to be part of the interview process for a candidate is welcome.
- Of the people who have interviewed the candidate, anyone in the testing group and any senior player from any other group who will work with the candidate can veto the hiring of this person.
- The veto policy must be actively managed so that vetoes will not be based on race, religion, family situation, gender, sexual orientation, age, disability, national origin, etc.

There are a few reasons for hiring by consensus.

- I believe that it is a more serious mistake to hire badly than to pass up a good candidate. The consensus process will sometimes exclude a good candidate, but it is likely to expose problems. I have been repeatedly humbled by hiring someone over the objection or strong reservations of one person, only to later discover the hard way that the candidate is a jerk, a sexual harasser, an incompetent, or just not a good fit. I am just not willing to make this mistake any more.
- Different interviewers bring out different answers and different characteristics from candidates. I want a broad process that encourages people with different viewpoints and interests to interview candidates. Encouragement requires letting people know that they're welcome and that their viewpoint will be carefully considered.
- Many of the people who I hire are "opportunity candidates" (see below). We are taking a risk hiring these people, or making some accommodations for them. I don't want to confront staff resentment later that this new person has some privileges that the complaining staff member lacks. For example, if someone can't work beyond a 40-hour week, I am likely to raise this during the post-interview evaluation meeting (see below). My question to the staff is direct.[3] I explain non-private details of the person's situation and ask whether the staff can live with the fact that this person will work shorter hours than they do. If they say no and I can't talk them around, then I can't hire this person. If they say that they'd resent a situation in which the candidate would get

[3] Jack Falk comments, "This reflects Kaner's personal style and it is not duplicatable. It works for him but it might not be useful for the readers." Some very successful managers make private arrangements, make it clear to everyone that the arrangements are private and not subject to challenge or discussion, and they find support from their staff based on a reputation for fairness and discretion that they establish in other ways. Additionally, some matters are personal and should be treated as confidential. Use your judgment and do what works for you in your relationship with your staff. The issue that we're raising is that you should have a plan for managing the implications of an employee's special circumstances. A consensus-based acceptance of them might be an appropriate and useful tool for managing them.

full pay but work fewer hours, I reassure them that I will calculate the person's salary with their limitations on hours of work factored in, and I ask if that would take care of their objection. If they say yes, then three months later, when someone says "Gee, how come Jane doesn't work as hard as I have to?", I can say, "You agreed to this. Now we all have to live with it."

- The people who participate in the hiring process become part of the support network for the candidate after she joins the company. Many people new to a company go through a rough period during their first few months as they learn cultural or technological issues the hard way. I want to be in a position to say, *We made the decision to bring this person in. Now let's help her succeed.* This is particularly important when it is the new manager who is having a rough time. She needs a staff who will rally around her and support her.
- On her first day on the job, it is made clear to the candidate that every person who interviewed her voted to hire her. She was carefully considered by all these people, and they all welcome her. Especially in a controversial position (welcome to testing), it is good to remember that the person you're arguing with today was one of the people who liked and respected you enough to vote to hire you. This can go a long way toward lessening the mistrust that sometimes develops between testers and programmers.[4]

Even in a small testing group, I don't require everyone to be part of the interview process. Some people just don't want to do this, or they don't want to do it for every candidate. That's fine. But they can't complain about the hiring decision later.

Junior staff are often hesitant to interview people because they don't know what to ask or because they're afraid to alienate the person they're interviewing. Some people like to interview the person who will become their supervisor. Others are uncomfortable. I encourage juniors to participate in interviews (partially because they have to get training in interviewing sometime, and there are only so many opportunities for this in a year), but I look for ways to make them comfortable. Here are some examples:

- The junior can silently watch an interview conducted by someone more senior. When I do this, as the interviewer, I introduce the observer to the candidate, explain that I am training this tester in interviewing by allowing him to watch some interviews, and ask the candidate's permission. If the candidate balks, I will ask the junior to leave. If the candidate is an individual contributor who is bringing technological skills, I might not hold this against her. Some very competent individual contributors are shy or awkward in groups. On the other hand, if the candidate who rejects the junior is interviewing for a management position, I will probe deeply into her attitude toward training and mentoring staff. She will probably not get the job.
- A small pack (maybe four of them) of juniors can take the candidate to lunch, accompanied by one mid-level member of the staff who is a good observer. The mid-level staffer will observe but not speak beyond the minimum required for politeness. I am particularly likely to do this with management candidates. The interview pack and I might even draft some questions before lunch, that they will ask during lunch. I encourage them to ask questions about the candidate's attitude toward training, education, and working conditions. If the candidate gets huffy ("You can't ask questions like that—I'm the manager, you're just the junior employee"), the candidate gets to go home early.

Note that both of these examples provide an opportunity for the candidate to exhibit behavior (the way she handles the situation, rather than what she say) that gives you insight that you probably can't get by

---

[4] Terminology note for the book as a whole. Should we call these people "programmers" or "engineers" or "developers" or some other title? We are reluctant to use the word "engineer" because it has become politicized, but in any case, we don't think that "engineer" applies to people who write code more than it applies to people who test it. At Florida Tech, for example, we teach several courses on software testing in the software engineering program and some of our best software engineering undergraduate and graduate students specialize in testing. Similarly, we think that testers *are* developers, so we don't choose to draw a distinction between testers and (other) developers. We call them programmers because testers interact with their code.

asking questions. You'll see a lot of examples like this running through this chapter, though I won't keep drawing your attention to them.

You'll also note that my interpretation and reaction to the behavior might be different from yours. I am a fan of Deming (1982). I believe that in employee relations, management should proceed from leadership and credibility, not from fear and power. I believe that it is normal for people's behavior to vary over time, for good people to screw up sometimes. I believe that many employee-made errors are induced by systematic weaknesses in their working situation (i.e. management-induced problems). I believe that most people want to do a great job. I look for management candidates who foster that greatness.

You might look for people who have other attitudes. I can't say that you're wrong. This chapter isn't to convince you either way on those issues. It is to help you gather data that will help you evaluate whether the candidate meets the vision you have. My interpretations are for illustration, not because they are necessarily the best ones for your situation.

The problem of discrimination is a more difficult one. First, let me stress that it is (even in Silicon Valley in 2000), a genuine problem. Asians, blacks, Hispanics, women, and people with disabilities that do not interfere with their work are still finding it hard to get work, to get treated with respect on the job, to get equal pay, and to get promoted. I have been personally reprimanded for hiring a black employee. I have had to deal with repeated and credible complaints from female staff members that they were being harassed by an executive and I have personally witnessed some of that unacceptable behavior. Asian immigrant colleagues of mine have faced offers of as little as half of the going rate paid to equally or less qualified whites. Black computer science students have told me about difficulties they've had getting internships or permanent positions. All of this is illegal, all of it is unacceptable, and somehow you have to convince a racist not to use a veto for a discriminatory purpose or your consensus-driven process becomes a roadblock (or a steep cliff) instead of a tool.[5]

How do I deal with this issue?

- Certainly, I publicly remind the testing group that they cannot discriminate on the basis of certain types of characteristics.
- Most of my work, though, is private. People make racist or sexist (or etc.) comments in private. Some people will also consistently speak against candidates of a certain kind, gradually revealing their colors. I talk with them privately and will cut off their veto power (and their opportunity to interview) if necessary. (I've also suggested that if a person doesn't like these rules, maybe he should find a more hospitable company.)
- I also deal with the discrimination problem, if I think that there might be one, by bringing prospective employees to the company as short term contractors. This makes it hard to recruit people who already have a job somewhere else, but if the pay is right, unemployed or underemployed or underpaid testers will gladly accept a six-week contract. I'll hire contractors on my own, or in conjunction with just one or two interviewers. The full group process doesn't take place until the contractor has been working for a few weeks. Now we have data on actual performance. A candidate who is doing a good job is harder to reject for spurious reasons.

A STRATEGY FOR GATHERING INFORMATION (m xxx)

You're looking for a candidate who has certain characteristics (knowledge, skills, abilities, and other characteristics—see below). Make a list of the desirable characteristics.

You can gather information about the candidate on these characteristics from several sources:

- Resume

[5] If you're looking for a job or for an improvement in your current job, we might have some useful advice for you about dealing with discriminatory practices in our chapter on job seeking.

- Phone screen
- Work samples
- Publications and other public materials
- Interview questions
- Tests
- Puzzles
- Group interview
- Behavior elicited by the interview
- References
- Investigative material (such as drug tests, credit reference checks, etc.)

Different sources are more effective for different characteristics. You can make a matrix to represent this. For example, yours might be structured like this:

Characteristics	Sources								
	Resume	Screen	Samples & Pubs	Interview Ques	Tests & Puzzles	Group	Interview Behavior	Refs	Invest.
Knowledge									
▪ **Testing courses**	x	x		x				x	
Skills									
▪ **bug reporting**			x		X			x	
Abilities									
▪ **team building**	x							x	
Other									
▪ **tolerance of ambiguity**				x			x	x	

For more on this type of matrix, see Rosse & Levin (1997), Chapter 6.

Along with determining what method(s) you'll use to gather the information, you have to decide who will gather it. For example, if you and several of your staff are interviewing the candidate, you will probably split up the characteristics. For example, one of you might check bug reporting while another focuses on tolerance of ambiguity.

CHARACTERISTICS OF THE PERSON YOU'RE SEEKING (F xxx)

As in so many aspects of software development, there is great value in thinking about your requirements first rather than wondering why you didn't meet them later.

And, as in so many other aspects of requirements analysis, you can take this too far and get paralyzed by analysis. The process of defining the job and the personal characteristics in advance has struck me as the greatest opportunity for analysis-paralysis in the hiring literature. Please read this through and think on it,

but remember the 80/20 rule—80% of the benefit will come from the first 20% of the work that you can do in this area.

What is the role of the testing group?

There's a certain amount of controversy about the role of the testing group. The characteristics of the people you seek depend on the role of your group:

- Some of the brightest people in the field say that the testing group's function is to assess and report the quality of the product. (Bach, 1997[6]; Rothman, 2000; xxx reference to Marick). Quality involves a wide range of attributes. Your staff will be assessing code, documentation, entertainment value, usability, performance, conformance to written and implicit customer requirements, hardware compatibility, and lots of other stuff. Additionally, if they are serious about publishing assessments, the group will benefit from staff who understand statistical theory and measurement theory.
- Other people in the field (me, for example) think that the typical role of the test group is to discover, report, and advocate for the repair of defects. I use Weinberg's (xxx) definition, "Quality is value to some person" and define a defect as a failure to meet a person's sense of appropriate quality for that product.[7] I encourage any person to file a defect report if the product fails to meet their sense of appropriate quality for that product, and I encourage the testers to consider the product from many angles, through the agendas and interests of many people.
- In some circumstances, the role of the group is more constrained. For example, consider a company that develops products under detailed contracts that incorporate long, precise specifications. The role of the testing group in those cases might be to assess the conformance of the product to the specification. Depending on the types of issues raised in the specification, you might or might not evaluate the usability, performance, hardware compatibility, etc.
- Another example of a constrained group is one whose mission is to search the code for coding errors. Their entire (or primary) question is whether the code as written implements the intent of the programmers (which may in turn be expressed in documents, such as specifications).
- At the other extreme is the quality assurance group, who set and enforce standards and facilitate software development process improvement efforts. (They may also spend time finding and reporting bugs).

I will assume that the group's charter is broad and focused on testing (such as the assessment and the discovery groups). If your group's charter is narrower than this, ask yourself whether it should be. Then search for a staff that collectively fills the range of skills, knowledge, and abilities that you need to fulfill your charter.

---

[6] From Bach, 1997: "Everybody shares the overall mission of the project-- something like 'ship a great product' or 'please the customer.' But each functional team also needs a more specific mission that contributes to the whole. In well-run projects, the mission of the test team is not merely to perform testing, but to help minimize the risk of product failure. Testers look for manifest problems in the product, potential problems, and the absence of problems. They explore, assess, track, and report product quality, so you can make informed decisions about product development. It's important to recognize that testers should not be out to 'break the code.' They should not be out to embarrass or complain, just to inform. Ideally, testers are human meters of product quality."

[7] This isn't the definition that I would use in court. There, we talk about a characteristic of the program that makes it unfit for normal use or that puts it into nonconformity with a contracted-for promise. But in a development shop, I am trying to help the company make an excellent product, rather than arguing about whether it is so bad that it is worth a lawsuit.

Strength in diversity

There is no single profile that fits the ideal software tester. Two testers who work on the same program will find different bugs. Diversity is essential. People with different skills, backgrounds, and sympathies will spot different classes of issues.

Additionally, the testing effort requires several strikingly different skill sets. For example, a few years ago, Jack Falk and I worked with a company that produced software to manage employee stock options. This application area is subject to complex government regulation (taxes, employee compensation, securities law, and sometimes lending law). If I had to build a small testing group for a company like that, I would shoot for a staffing mix like this:

- Senior tester or test manager with experience in business operations or human resources. This person has worn the shoes of the customer for this system. For a vertical application, I think this is essential.
- Senior tester or test manager with strong test planning skills. If this is the test manager, she needs excellent mentoring skills, because she won't have time to write the test documentation unless she is an individual contributor.
- Test automation hotshot, willing to serve as the group's tool builder.
- Talented exploratory / intuitive tester, someone who is really good at finding bugs by playing with the product.
- Network administrator. This person has the dual role of helping the other testers set up and deal with the ever-changing configurations that they have to test under, and designing configuration tests to determine whether the product will run on most of the systems in use by the product's customers.
- Attorney who is willing to wander through the various statutes and regulations looking for rules that the program must cover.

Some of these people have programming skill. Others have special knowledge of the application area or of platform or hardware. Others have special knowledge of the tools and techniques of black box testing.

I often see ads that require a specific profile of software testers: degree in software engineering or computer science, knowledge of the programming languages and tools in use at the hiring company, and some number of years of experience as a tester or as a programmer. These are nice credentials, but they would only solve *some* of the problems that must be addressed by this financial application company's testing group. Diversity is essential.

Several of the most effective testers that I've worked with have had no programming experience. On the other hand, most of the effective testing groups that I've worked with or consulted to have had at least one knowledgeable programmer on staff. Diversity is essential.

```
Repeat until hiring is completed:
                        {Diversity is essential};
```

Accepting testing as a way-station

The testing and technical support groups are the easiest technical entry points to a software development company, and therefore, many people come to testing on their way somewhere else. They want to become programmers, marketers, project managers, technical writers, whatever.

Over the years, a remarkable number of people have come to me with the glitter of somewhere else in their eyes and the willingness to stick it out in testing for an appropriate amount of time. If they offered the right mix of opportunities to me, I hired them even though I knew that they would go away eventually. (Then again, everyone goes away eventually.) A surprising number of those folks stayed in testing for a long time, or have come back since. Testing is a seductive field, once you come to know it.

But even the confirmed transients can add substantial value and substantial diversity to your group. Consider the following examples. These are real people, but the names have been changed:

- Joe was a marketer, who had reached as high as Director of Marketing and Sales. Between jobs, he decided to take a testing position (and so learn a great deal more about the realities of development) rather than moving right back into marketing. The job educated Joe a great deal. And Joe educated the other testers. He had plenty of ideas on how to make bug reports more persuasive, how to spot issues that would make the product harder to sell or support in the market and how to collect data to back up a bug report of the issue, and how to build credibility with groups outside of product development. He was a net gain to the group within weeks of being hired.
- Susan was a former VP of Marketing who heard about Joe and decided, between jobs, to learn more about product development too. She joined testing but only lasted in the group for a month or so before being transferred into a senior position in marketing, eventually becoming the company's VP. The testers didn't learn too much from Susan, but they had a great link into marketing.
- Sandy was a technical support supervisor with a software sales background. When her department was outsourced, she transferred into testing. Along with bringing a customer focus that was always welcome in this department, she brought strong scheduling, budgeting and status reporting skills. She also had a personable style (salespeople are so valuable in testing groups) and was able to present bad news without creating an interpersonal edge. A short time (perhaps two months) after joining the testing group, she became the supervisor of the company's largest test team, working on its most delivery-date critical project. Her status tracking, early warnings based on status, and her visible but non-threatening approach had a big effect. The product shipped, with good quality, a day ahead of schedule. About a year after joining the testing group, Sandy moved into a senior tech support role at another company. For the year that she was in testing, she was invaluable.
- Tony was a talented but self-trained programmer who wanted to work with a well-known programming team. Joining the testing group gave him an entrée to that team. If the opportunity arose, Tony would have been a valuable toolsmith in the testing group, and would have probably been quite happy in that role. He eventually rose to a senior level in the programming group. Tony's manager didn't manage Tony well. Tony spent the required time on his (testing) job, but he spent a lot of additional time working with the programmers, troubleshooting and fixing problems with a different product. The manager felt that Tony hung around with the programmers too much and with the testers too little, and that Tony collected and used a lot of information about the program but kept too much of it to himself. Tony (and his manager) might have benefited from a frank discussion about Tony's role that laid out ground rules while acknowledging and making more room for Tony's career interests.
- Cindy was a librarian who probably wanted to become a technical writer but got into a testing group first. As a tester, she was able to spot communication issues and training-related issues and explain them with clarity. She also excelled at writing testing documentation and she was effective as a supervisor and trainer of new testers. Over subsequent years (and subsequent companies), she moved back and forth between writing and testing assignments.
- Sean had been a project and a product manager, but he burned out on software and took a several year break from software development to manage construction projects. Eventually he was ready to come back to software development but software companies were hesitant to hire him. He

didn't have experience in the currently fashionable language and so his skills were seen as outdated. So he joined a test group to update his skills. Years later, he was still in testing, doing a great job, managing the test group of one of the industry's largest publishers.

There are lots of other stories like this—people who came to a testing organization with strong skill sets and a desire for a relatively short term (6 to 18 months) stint in testing, who quickly started to contribute tremendously to the groups that they joined or who would have contributed tremendously if they had been properly managed.

The mismanagement problem is a real one. Or, at least, it has been for me. On balance, I've been successful with staff who come to me with skills, on their way somewhere else. But when I've tried to jam them into standard molds, without recognizing and taking advantage of their special strengths, they've often been disappointed and disappointing. If someone's attractive features are special skills, then I am hesitant to hire that person unless those skills fit an identifiable near-term or intermediate-term need.

Opportunity hires

I want a great staff. This isn't easy. It's harder when the field is booming, as it is today, because there are fewer people than jobs. It's even harder when I work with a company that pays less than the market rate. (I founded the testing group at one company whose products were very exciting, but whose top tester salary matched the entry level rates of other software companies. I was successful in recruiting a topnotch staff, but it took creativity.)

When I can't offer the top money in the field or the most exciting product line or career path in the field, I have to find other incentives to attract solid talent. And so I look for people who have special needs that I can fill. These people have a strong skill set but they have a personal commitment to specialized interests that I can help them explore or they carry baggage that make other test managers reluctant to hire them. That leaves proportionally more of these people for me. If I can connect with this market (these markets) of people, and if I have a good job to offer, then I can have my pick of them. I call these people "opportunity hires." For them, the job that I offer is a special opportunity. For me, the skills they come with provide a special opportunity.

The people who I described in the last section, people experienced in some other area who want to spend six months or a year in testing and then move on, are an example of opportunity hires. There are a lot of other people who are relatively unpalatable to traditional hiring managers, who have a hard time finding a position even though they might be brilliant. Here are some examples:

- *Retirees, especially retirees who want to work only 20 or 30 hours per week.* People step down from their fast-paced, full-time jobs, and discover that they need money or that they've become bored. The ones that I'm thinking of are not looking for fast-paced jobs that might require lots of overtime. They're looking for an intellectually stimulating position that offers steady hours and some flexibility to spend time away when they have to deal with personal matters.
- *Pregnant women or single mothers with young children.* It might be illegal to discriminate against these people, but they're discriminated against anyway. I don't ask about candidates' family situations—if you do, the odds are that some day, you will be sued for it. But when someone raises the issue with me because they anticipate needing a leave of absence in the near future, or because they need unusually flexible hours, I'll talk with them about it. The fact is that at most Silicon Valley style software companies, people work long hours. And near the product release date, people work longer hours. Especially testers. If that's true at my company, I'll say so. I don't feel obliged to hire anyone who can't reliably put in those hours. But when someone with an extraordinary skill set and a positive attitude tells me that she'll gladly take a step down in responsibility and salary in return for strong control over her hours, I'd like to find a way to make this work.
- *Other family commitments:* Many middle-aged workers are facing a new responsibility. Their parents need time-consuming attention. As another example, non-custodial parents will often bargain for guaranteed vacation times and additional time off to be with their children. You can't

ask about family commitments beyond asking whether there is anything in the candidate's personal situation that would interfere with his regular attendance at work at the normal work pace that you expect of your staff. But if a candidate raises family commitments as a bargaining issue, I think that it is fair game to consider what accommodations can and cannot be made and whether this candidate's talents balance out the requested accommodations.

- *Returning to the workforce.* The prototypic case is the mother who comes back to software after investing several years in the care of her children. Sean (the project manager described above) is another example—he stayed commercially employed, but not in software. Similarly for the technical person who spends years trying out sales and marketing and realizes that she's happier in engineering. These people come to you with aptitude and seasoning but without current knowledge of tools, languages, and best practices.
- *Immigrants who speak English with a thick accent.* Testers must be able to fluently understand the language spoken by the product development team. I don't want to hire people who can't understand spoken or written English. But many immigrants understand more than they can say. I am reluctant to consider a candidate whose written grammar and spelling (especially if the mistakes are on the resume) are poor. And I am reluctant to consider a candidate who has no employment references within reasonable telephone distance (North America). But so is every other hiring manager. If a candidate looks as though he might be particularly talented or particularly smart, I'll try to find out more about him. Sometimes, this means hiring the candidate on a short term contract to see whether she can handle the job.
- *Career switch.* Some of my most successful hires have been of people who were leaving a mid-level to senior position in a different field in order to come to software. These aren't people who come to you and say, "I want to work in testing for a year before I move on." They're people who say, "I want to switch to software. Are you where I start?" They might well stay in testing. The key for these folks is often not the pay scale. It is the opportunity for training in a new profession. If you can/will offer to spend additional personal time (of yours, as the hiring/training manager) coaching this person, you can make a competitive offer even if the candidate will have to take a steep pay cut to come work for you.

In sum, opportunity hires are people who come to you with special needs or special circumstances. Their circumstances are odd enough that recruiters will often not help them, perhaps never presenting their resumes to employers or only presenting their resumes for unacceptably junior or low-paying positions. You can be in the position (I have been in the position) of being the only employer that the candidate has talked with, through a significant job search, who is willing to show respect for their talent and experience while making allowances for their special circumstances. That is, you might be able to be the only person who makes a serious job offer to a candidate who would normally appear to be substantially overqualified for the position you have open.

Opportunity as risk

Not every "opportunity" candidate is a good bet. Here are some examples of people who initially look like opportunities but on closer inspection turn out to be potential disasters. Again, the names are changed:

- Joe is an alcoholic. He says he wants to shift from technical (non-software) sales management into software. He looks great on paper: increasingly responsible jobs, good performance reviews, decent technical education, MBA, analytical and smart. But his real reason for looking for a new career is that he is screwing up so badly that he is just about to lose his current job and he is afraid that no one else in his current field will hire him.
- Sandy wants a programming job today. She'll accept a position in the testing group but starting on her first day on the job, she will do anything she can to wangle a transfer into the programming group.
- Jerry used to have a senior position. He will accept the job with you, but he will still expect to be treated with the kind of deference that he got used to in his last position(s). He will become

unhappy when people (such as programmers) don't take his advice and follow his directions. He will dedicate his time on the job to political intrigue rather than to finding and reporting bugs.

- Jane did well in her last career (perhaps even a software career, as a programmer or software marketer) but has no aptitude for testing. In your group, she will be bright, articulate, cooperative, hardworking and (unless you manage her very effectively) unproductive.
- Cecil is pursuing a graduate degree and will need a lot of time away from the job. Your staff are working far more overtime than they want to work and you are not in a position to accommodate their time-consuming personal needs (such as, finding time to take evening classes themselves). If you hire Cecil as a full-timer, your staff will come to resent him.

Minimizing opportunity hiring risk: Short-term contracts

When someone's moving into a new type of position or moving back into the workforce after a long time out, there is just no way to be sure that they and the job will be good for each other. You will discover that many wannabe-opportunities are no good and won't hire them, but for the person you seriously consider, you still face a substantial risk of being wrong.

Whenever possible, I try to hire people for a short term, such as a six-week contract. If the candidate works well and has the skill set that I want, I can then make a long term employment offer. But if she doesn't work out, I don't have to renew the contract, make a new contract, or extend an employment offer. (I make it very, very clear up front that the contract is complete at the end of six weeks, with no promises made as to consideration for future work.)

If the candidate doesn't work out, but is trying hard and gets along reasonably well with the other staff, I'll keep her for the full six weeks. This way, she gets to finish out the contract. This helps her in her next interview, when a prospective employer asks, "Why did you leave BugWare, Inc.? You were only there for six weeks." She can truthfully answer "It was only a six week contract. I completed it." And I can give her a reference that says the same thing.

Handled this way, the six week contract limits my risk as an employer and the candidate's risk. Either of us can walk away without complication.

(Note: some companies prefer to contract for temporary workers through an agency. This keeps the employer/employee relationship a bit more distant and limits some of the potential legal complications associated with employment relationships. Other companies prefer to manage the relationship directly and pay the person more than she'd get after the agency takes its cut. If you don't know how to handle this, check with HR. They might have a strong opinion on either side.)

Minimizing opportunity hiring risk: Group buy-in

I discussed this above, in the section on consensus-driven hiring. It is important for your group to accept special arrangements. My position is that everybody has some special arrangements and everybody needs some special accommodations. The issue is to make this clear enough that people are tolerant of each other's situations.

Understand your tradeoff

Every person offers you opportunities and risks. In some cases, the contrast is more evident than in others and I've called those "opportunity hires."

It costs time and money and attention to manage and train any new person. If the opportunities that person offers you are real—you can take advantage of them *now*, then you have a reason to be flexible and to invest. If the opportunities that person offers are not ones that you *can* take advantage of right away, then you should be more cautious about making the hire.

DEFINING THE POSITION AND CHARACTERISTICS NEEDED FOR IT (m xxx)

Any book on recruiting will talk about this. Rothman's (1998) treatment is well tailored to the programming environment and quite thoughtful. She also provides sample descriptions of testing positions. Rosse & Levin (1997) provide a detailed generic treatment. Risser (1993) provides a good overview. Kaner, Falk, & Nguyen (1993) discuss some of the attributes of good testing candidates.

Defining tasks you need accomplished today

You're recruiting because there is some work that you want done. You're looking for someone to do it. It makes sense to list the specific tasks and the knowledge, skills or abilities required to do those tasks. You will then evaluate the candidate against the list. If he can't do what you need done today, you need someone else.

Defining tasks you need accomplished later

If you take a primarily task-oriented approach to defining the job, then there will be several tasks that aren't needed for today's product that will be important over the next year. What other things should the tester be able to do, over the next year or two, that aren't on the immediate needs list?

If this person isn't capable of doing some of these tasks now, can you train him? Can you spare the time? Do you have the skill? Does he have the aptitude?

Defining the strategic role of this person

Along with the specific tasks that you are trying to get done now, you should have a sense of what this person will do for you over the longer term. For example, are you trying to find an automation specialist? A test planner? Someone who will become a supervisor or manager after a few months of training? A brilliant opportunity (candidate with strong potential to become a senior manager) might be no opportunity at all if your goal is an automation hotshot.

List of individual characteristics

It can be very useful to list the individual characteristics that you will consider important for the job. You might want all of these for all jobs, but some are more important for some jobs than others. For example, diplomacy is probably a skill more needed by managers than by senior programmers.

Lists of characteristics are often broken into lists of KSAO (Knowledge, Skills, Abilities, Other). These are defined as follows:

- "Knowledge" is the body of information that the candidate will need in order to perform effectively. For example, a person might know or have training in test case design, maybe even a course in creating test matrices.
- "Skills" involve proficiency at a specific task. For example, a person might be really good at creating test matrices.
- "Abilities" involve potential to do a job. For example, a person might be a very bright, systematic, analytical thinker, who you would expect to be able to quickly become very good at creating test matrices.
- "Other" includes other characteristics that are important to the job, that aren't abilities. For example, a person might have personal integrity.

Here is a list of some of the skills, abilities, or other characteristics that are sometimes mentioned as relevant to success as a tester. This is a starting point, not the ultimate list for you. Don't look for all of these in one person. Pick a subset, define them the way that works for you, and add your own.[8]

- Alert
- Attentive to detail
- Analytical problem solver
- Architect (talented at designing systems, in breaking the system into achievable tasks, subtasks, and data)
- Arrogance (usually, less is better)
- Artistic (understands visual or audio presentation issues, can knowledgeably critique the esthetics of a design or product)
- Assertive (and willing to speak up when there is a problem)
- Auditor (compare situations against standards)
- Author (published, impressive, credible)
- Certified (for example, in quality engineering, quality auditing, software quality engineering, or software testing by the American Society for Quality; in quality analysis or testing by the Quality Assurance Institute; in a testing or quality related area by UC Santa Cruz Extension or a comparable university-based professional training program)
- Commitment as a person (keep promises, stick around)
- Commitment to a task (do what it takes)
- Commitment to quality
- Copes with difficult circumstances
- Copes well with rejection (for example, doesn't get depressed if several bugs go unfixed).
- Courageous (but not reckless) and willing to own the responsibility for his actions
- Creative
- Credible (people believe what he says)
- Curious (inquisitive, likes to explore and find things out)
- Customer focused
- Decision maker and problem solver (shows good judgment, realistic understanding of issues)
- Decisive
- Not very defensive (able to take criticism)
- Detail oriented
- Detective (if the programming staff cannot or will not brief testers on the program and if there are few official written descriptions, then at least some testers in your group will have to be good at digging up relevant information and briefing the others.)
- Diplomatic (able to convey bad news, criticism, or unreasonable requests in ways that don't offend and do encourage a desired response)

---

[8] When I teach a seminar on recruiting, I assign the following exercise: *Circle the 10 most important of the attributes listed here and add at least one more that is important for the position you're trying to fill.* Some of my former students have told me that this exercise was particularly useful for clarifying their focus.

- Editor (can effectively review, criticize and improve printed materials)
- Effective with junior testers
- Effective with senior testers
- Effective with test managers
- Effective with programmers
- Effective with non-testing managers
- Empathetic (able to appreciate other people's situations and viewpoints)
- Empirical frame of reference (learns by running experiments or poring over data)
- Empowering (promotes excellence and risk taking in other individuals)
- Energizing
- Fast abstraction skills
- Financially aware and sophisticated (for example, able to make economic arguments)
- Finds bugs (intuitive tester)
- Flexible (willing to shift takes or to take on new things)
- Goal setting
- Glue (promotes group cohesiveness)
- Humility
- Integrity (likely to keep commitments, unlikely to engage in dishonest conduct)
- Interpersonally perceptive (reads verbal and nonverbal behavior)
- Interviewer (good at getting information by asking questions)
- Investigative reader (good at pulling information from incomplete specs, from the Net, etc.)
- Leadership
- Likes (or at least, isn't driven crazy by) repetition
- Long term thinker
- Low probable ratio of noise bugs reported to significant bugs reported
- Meeting manager (skillfully facilitates or records (e.g. on flipcharts) other people's meetings)
- Mentor
- Multi-tasking (juggles multiple tasks well and can handle the pressure)
- Organizer and planner
- Persuasive
- Politically perceptive (reads the system)
- Policy and procedure developer
- Pragmatic
- Programmer (able to write good code, command technical respect of other programmers)
- Protective (stands behind, defends his staff, even when they're wrong)
- Punctual

- Scholarly (collects information, and is able to back up or evaluate arguments using data or credible opinions/statements of others)
- Sense of humor
- Spoken communication
- Strength of character (does the right thing even when it's personally costly or inconvenient)
- Subject matter expert, in the area being automated by the software
- Substance abuser (undesirable)
- Team builder
- Tolerant of ambiguity
- Tolerant of other approaches to managing projects, doing tasks and solving problems
- Troubleshoots well
- UI design (skilled at designing the appearance of features, etc., and a persuasive knowledgeable critic of the designs of others)
- Versatile (many abilities)
- Warm (interpersonally, makes the human environment more pleasant)
- Written communication
- Zealot (believes in The One True Way and insists that everyone else believe too. Not desirable in large quantities.)

I'll come back to the question of how to evaluate a person on these dimensions later. (xxx)

WHO APPROVES THE HIRING (F XXX)

Once you've found the right candidate, you probably still have to sell that person to your management, perhaps including your Human Resources department. These are the people you have to deal with at the end of the process, when you want to make the offer.

You should think about these people now, at the start of the hiring process, rather than later. If your Vice-President in charge of Getting In The Way has to approve a candidate, find out what she likes in candidates. Collect data during the interview process that can be used to show her that your favorite candidate has attributes that map onto her preferences. Knowing that VP's preferences will usually lead you to ask different (or at least additional) questions in interviews and might lead to you ask for (or hang onto) different types of work samples, or run different types of tests.

In courses that I've taught on recruiting, people have asked me time after time how I can speed up the hiring process. Some of them protest the thorough evaluation of candidates that I propose in this chapter, saying "Yes, but get real. Speed it up for hiring on web time." I have three responses:

1. *(Flippant.)* You can always speed up the process if you don't care whether you hire the right person.
2. *(Practical.)* You can do the multi-interviewer evaluation that I discuss here within a very few days. This doesn't have to stretch out. It's up to you.
3. *(Address the real problem.)* The longest delays that I see in hiring are usually attributable to management. The hiring manager qualifies the candidate in a matter of days and then has to wait from one to six weeks until the bureaucracy finally grants its approval. By this time, your candidate has accepted another offer. Assuming (as I do) that this is your biggest risk of delay, the more you take into account the objections, issues, etc. that will be raised by the Bureau of Delay, the faster you can get your candidate through.

If your actions are subject to being limited by slow decision makers, then at the start of the process you must ask "How can we put this resume on roller skates?"

HOW TO FIND CANDIDATES (F xxx)

You can advertise a particular position or you can advertise the general desirability of working for your company an ongoing basis, whether you have positions open or not. Some people prefer the ongoing approach because it generates a steady stream of resumes from people who pay attention to you or your company.

Ongoing advertising

The goal of ongoing advertising is to present an image of yourself, your group, and your company that makes people want to work with you. Do this by talking or writing about your technical views or your company's technology or management style. Most of these activities (publishing, teaching) are done for other reasons, but they have the effect of attracting people to your company. It should be obvious that it's important to avoid creating a false impression. Even if you succeed in hiring someone this way, it won't be a lot of fun working with them if the real situation doesn't match their expectations. Some of the usual types on ongoing advertising:

- Advertisements (paid spots) in newspapers, radio, TV, that promote your company generally rather than a specific position
- Books that you or your staff write
- Conference presentations
- Courses that you offer to the public or to the profession
- Newspaper or magazine articles about you or your company
- Newspaper, magazine or technical articles written by you or your staff about technical or management issues.

You can't afford the high cost of image advertising if you work for a small company. But if you are skilled at testing, you might well be able to teach courses on testing in the Extension (adult education) faculty of a local university or community college. Woefully few testing courses are taught in these faculties, partially because so few experienced people are willing to take the time to develop and teach the courses. If you teach well, and if you invite them to stay in touch, then many of your students will check in with you when they are looking for their next job.

Promoting a position

You can be much more effective at getting a stack of interesting resumes if you use multiple methods for publicizing your company and your currently open position than if you rely on one approach or medium.

You can advertise positions through:

- Announcements at conferences or meetings
- Announcements or ads in professional journals or newsletters
- Announcements (job listings) at employment services (your state unemployment agency, for example). (Yes, this can be useful. When BigCo. lays off 1000 testers, many of them will file an unemployment insurance claim. A notice with that government agency might capture their attention at the start of their job search.)
- Current employees (Some companies pay recruiting bonuses to staff—some pay as little as $250, others as much as $25,000. In a tight labor market, some companies hire half their staff by referrals.)

- Newspaper advertisements
- Postings on Net-based job boards and job-related newsgroups and e-mail lists
- Radio and TV advertisements
- Recruiters
- Spamming (if you want to find people who are willing to work for a spammer. Blecch.) (Note carefully that some of the best people will refuse to have *anything* to do with you if they hear about your opening via a spam.)
- Word of mouth announcements spread as rumors through the community of people you'd like to recruit from

Finding resumes on job listing boards

I have not recruited through job boards like monster.com. Reviewers of this chapter have given mixed reviews to such services.

- In some cases, employers pay a fee to review resumes and discover that the posted resumes are out of date or irrelevant. These folks feel that they spent too much for too little.
- In other cases, people find great resumes, contact the candidates and set up the first interview within a few days of beginning their search. Some of these folks say that they will probably never spend money on a newspaper help wanted ad or on a professional recruiter again.

It is worth seriously considering using a service like this. Your HR department might be able to find out for you which services have a good reputation in the HR community in your geographic area.

If you are considering trying one of these services, I strongly suggest that you publish your own resume on that service (maybe under a fictitious name) or that you get a good friend, who is looking for a job, to publish her resume on the service while you watch. As you watch, think about how average job candidates will interact with the service. For example:

- If the service requires you to type your data into a standard form (rather than accepting your resume), this is pretty time consuming. The odds are that you will make mistakes, leave stuff out, get dates wrong, make spelling mistakes. This resume will not be as polished as the one you wrote and edited offline. And the resumes you receive from other people on this service will not be polished either. Don't make the mistake of comparing the polish of these to the well-prepared resumes that people send you in the mail.
- If the service insists that you describe your qualifications in terms of a bunch of buzz words, you're not going to find many "opportunity hires" if you search using the typical buzz words. How are you going to search for talented but unusual candidates with this system?
- In general, ask yourself what questions the online service does *not* address in the online candidate description form. You'll want to be prepared to ask these during the phone screen.
- As a candidate, ask yourself how a company would discover your resume? Play the part of an engineer who is talented but not terribly skilled at marketing herself. What kinds of things might you leave out that would have made it easier for you to be found? As the searching manager, how will you find this candidate?
- As a candidate, ask yourself how you notice listings of companies on this service? How do they make themselves stand out? What made it easy for you to find other positions? There may be hundreds of openings listed. If you were to list yours, how would you make it visible and how long would it stay visible?

Posting openings on your corporate website

This is getting increasingly common and it makes so much sense. If someone is interested enough in your company to think about working for you, they are likely to look at your site for openings. The site itself tells them a lot about your company, simplifying your job of describing the company in your ad.

I strongly suggest that you make it easy for candidates to send you their resume via an email. If you require people to fill in a long application form over the web, just for a job at your company (which is definitely not the only company they are considering), they might decide that it is too time consuming and too much of a pain in the neck to apply to you. They might also consider this form a sign of disrespect for their time.

Consider using a dropbox for your email address, such as testmgr@yourcompany.com. If you change positions, the responses keep going to the right person (the new testmgr) rather than following you.

What do you say in an advertisement?

When I advertise:

- I want the right people to send me their resumes.
- I want the wrong people to send their resumes to someone else.
- I don't want people to misunderstand the job or the company.
- I don't want government regulators to tell me (or my company) that I am acting unlawfully or unethically.

Positions are often advertised ineffectively. An advertisement that lays out a generic job description is less likely to capture anyone's imagination, and is likely to discourage opportunity candidates (people with unusual backgrounds) from even applying.

The following examples were invented for this chapter. They might be reminiscent of ads that you have seen, but they were not based on any particular advertisement.

Here is an example of a poor advertisement:

> SOFTWARE TESTER. Hard-working, diplomatic, detail oriented, effective communicator. Great at finding and reporting bugs. Strong test planning skills. Automation experience desirable. UNIX and PC platforms. Must have B.S. in Computer Science or equivalent and five years of related experience.

Why would you want to work for this company? The ad doesn't say. What is special about this company? The ad doesn't say. What is interesting about this position? The ad doesn't say. Who will apply? My bet is that this ad will attract lateral hires (people who want to do the same job as they have today, but for more money, with a different boss). It will attract people who have standard backgrounds and it will attract underqualified people who don't worry about background (people who have consciously decided to apply for positions for which they are not qualified, at least on paper).

The next ad is another generic special that any HR Department can put together for you quickly:

> SOFTWARE TESTER. Come join our state of the art company, and define the leading edge in the testing of consumer software. Hard-working, diplomatic, detail oriented, effective communicator. Great at finding and reporting bugs. Strong test planning skills. Automation experience desirable. UNIX and PC platforms. Must have B.S. in Computer Science or equivalent and five years of related experience.
>
> We pay highly competitive salaries and have a superb benefits package. Work in the heart of Silicon Valley. Etc.

This ad promotes the company in general terms and the financial and lifestyle benefits of working for the company. It doesn't promote the career growth or the content of the job.

Here's an ad that could work:

> SOFTWARE TESTER: Bank applications, COBOL, SQL, Visual BASIC, Client/Server, etc. Financial application sophistication required. We are looking for excellent staff and will pay appropriately. Depending on experience and demonstrable skill, this position will pay $75,000 to $120,000 per year.

This is a generic financial position but is specific about its core selling feature. The employer will pay big bucks. (If $75-120K is no longer big bucks for this type of position, reread the ad with a bigger pair of numbers.) People who want the big bucks will apply for this job, whereas they might not apply for a job that offers "highly competitive salaries."

Here's another variation. In this case, the staff are the benefit, not the money:

> SOFTWARE TESTER: Bank applications, COBOL, SQL, Visual BASIC, Client/Server, etc. Every member of our group has sophistication in financial applications, and strong technical skills outside of testing (such as programming, data design, etc.) along with solid testing experience. Our department fosters a mutually supportive, growth environment. We work in teams and we make time to educate ourselves and each other.

Now consider this one.

> SOFTWARE TESTER. We test data communications software for the home market. Help us develop software that must be reliable, quick, and easy and fun to use.
>
> To effectively test our products, you will probably have to be able to read and write code. For this position, we are especially interested in people who know about set-top boxes, cable modems, TCP/IP, browser internals, or other current data communications implementation and design issues. We are also willing to meet testers (whether you can program or not) who are skilled in finding OS-level or device-level bugs, or who are skilled in performance measurement, client/server methods, or component level testing. Excellent communication skills are a must.
>
> Pluses include project management skills, experience creating and managing test plans to coordinate the work of several testers, and test automation experience.
>
> The successful candidate will probably have a degree in computer science and five years of software development or testing experience. In your cover letter, please indicate the types of evidence that you can provide that you can excel in a fast paced company as a technology-sophisticated software tester.

The ad conveys interest in technology and in the satisfaction of customers. Evidently, these are values of the group and (presumably) of the company. The ad might attract lateral hires (from other testing groups), or programmers who are willing to switch into a testing role in order to learn more about data communications.

The ad will also attract a few senior testers who want to brush up on technology. Their letters (the best of them) will admit to a lack of data communications experience, and to rusty programming skills, but will stress their project management and test planning skills. They will express great interest in this opportunity to learn about this new (to them) field, and will stress their willingness to work hard to achieve the learning.

In general, this ad tells people what they'll have to be (or become) good at, without demanding any particular credential (such as a degree). It will also attract the same clueless crowd of people who will send their resume to any ad that says "tester" or "programmer" and so you'll still have to throw those applications away.

One last example.

> SOFTWARE TESTER. Be one of the first employees in a software start-up. We offer challenging technology, long hours, and a stock option plan that will let you share in our success. It's too early to publicize the nature of our business, but we will carefully consider every resume that shows at least five years' experience in any aspect of software development, including at least two years' experience in software testing.

Each of these ads stresses what is special about the employer. Every well-marketed product or service carries a "unique selling proposition"—something special that is hard or impossible to find anywhere else. A reason to buy it. I apply that principle to the job advertisement as well. The company might be committed to customer satisfaction, hard driving pursuit of new technology, process management (ISO 9000-3 or CMM done by a company that believes in it and wants to benefit from it), job stability, family values, whatever. Telling candidates about the corporate mission and values, and the group mission and values helps people decide whether they are excited by you or not. Rosse & Levin (1997, p. 61) talk about this “unique selling proposition” in terms of being an “employer of choice,” a place that people will seek out. Rothman (1998) writes of positive and negative company factors that should be considered for advertising.

The ad will often also say something about what must be special about the candidate. I tend to be flexible on formal "requirements" like a degree in computing or accounting. I take some care to position these requirements as desirables, rather than as rigid requirements, unless I believe that they are absolute minimum requirements for the company at hand.

In special cases, I will be very specific. For example, I worked for an entertainment software company. Our style of testing was fast-paced, exploratory, and without the benefit of a specification. The product design was subject to late changes, as we (and others) criticized a product’s entertainment value, its appeal, its clarity of presentation and ease of use. During this period, certain DoD contractors and large IT employers were laying off technical staff. During interviews, many of these people insisted that our development methods were all wrong, that we would have to change all of them and that our group’s prime mission should be to facilitate that change. These people wouldn’t cut it in my organization. Most tiring were the follow-up calls asking how dare I not hire them, insisting that they were qualified for the job because they had 15 years of testing experience. It was hard to acknowledge that they did have the experience, while getting across the message that it was the wrong experience. Eventually, my ads read:

> Verifiable experience in development, support, or testing of software that was to be sold to or used by mass-market customers.

I still got the inappropriate resumes (fewer) and the follow-up phone calls, but it was faster and easier for me to say to a candidate that she was unqualified, as demonstrated by the lack of a qualification listed in the advertisement. The specific language in the ad satisfied people that they weren't being singled out when I turned them away. They might have thought I was a fool but they saw me as consistent, so they were willing to leave me alone.

I didn't adopt this wording because I don't like people with DoD or IT training. I adopted it because, at that time, in that market, I was getting flooded with resumes and follow-up calls from people who were not going to succeed in a certain class of job. They were wasting their time and mine.

Cook (1992) and Whitaker (1994) provide several other useful suggestions for advertisements.

Should you put your name in the advertisement?

Another issue in the content of the ad involves whether you should list your name as a contact point. I always do list my name. This wastes some of my time, because recruiters and more recruiters and more recruiters call me, and because some candidates call me. But I eliminate most of this issue while keeping the ad personal by saying:

> Send resumes to Cem Kaner, Manager of Software Testing, <<company address, company e-mail address>>. Please send inquiries and resumes by letter or by e-mail. I cannot handle inquiries and applications by telephone.
>
> Principals only please. Materials received from intermediaries, such as recruiters, will not be reviewed.

An advertisement with a human face will attract people who like to work for/with humans. Most ads are impersonal, so a personal touch stands out.

By the way, after you've been in the business for a while, people come to recognize your name. I never associate my name with an employer that I wouldn't commend to a friend. I never associate my name with a description of a job that I wouldn't give to an appropriately qualified friend. I never associate my name with an interviewing process that is designed in a way that it will demean or intimidate the candidates. Over the years, goodwill develops. People will apply for a position just because it's your name on the ad.

If I am interviewing on behalf of an employer who has staff difficulties, I'm honest about that with candidates, and I encourage other interviewers to be honest about it. The goal is to give candidates a "realistic job preview" (Rosse & Levin, 1997, p. 62). That doesn't mean that we try to advertise the company's faults or to discourage people from working at the company. And I don't necessarily put the company's weaknesses forward in the first phone screen. But it's important to make sure, before the candidate accepts a position, that she knows what she's signing up for.

WHAT IF YOU CAN'T FIND GOOD CANDIDATES? (M XXX)

Imagine publishing excellent ads, getting back a supply of resumes, and discovering that none of the people are worth considering. What should you do?

- The more urgently you need staff, the more strongly I urge you to get a mid-level or senior-level testing contractor. Let that person buy you time while you look for longer term staff at a more reasonable pace. You'll pay more per hour for a contractor, but when your urgent need for him ends, you can make him go away with no muss, no fuss, no guilt, and no lawsuits.
- The "opportunity hire" strategy is especially valuable in this situation. Look back through your resumes for hidden diamonds. Don't look for lumps of coal that you hope you can turn into diamonds.
- Try shifting to a different type of advertising. Use the web (or if you did use the web, use the newspaper), etc. Post notices at conferences, retain a recruiter, whatever.
- Carefully consider whether you have the time and the infrastructure that you need to hire a person who is much more junior than you would like. Can you train this person? Supervise him? If you think that you might be able to, ask next what qualifications you need. It is tempting to get someone who has basic testing skills, who could add some (albeit superficial) coverage to the testing effort from the start. As an alternative, consider what you'd like this person to be doing a year from now, and then ask what the hardest-to-learn aspects of that job would be. Then hire juniors who have that hardest-to-learn knowledge. For example, you might decide that a fresh B.Sc. in Computing who has a lot of coursework in real time systems, multi-threaded systems and fault tolerant designs can grow into a testing role better than a traditional black box tester can grow into the communications software troubleshooting role that you have in mind.

- Don’t hire a body to have a body. If you are so desperate that almost anyone will do, hire a junior contractor. Don’t lock unsuitable people into your department. That mistake will haunt you.

SIFTING THROUGH RESUMES (m xxx)

Most resumes will be rejected. Sort *quickly* through the resume pile to find the people who are worth calling. It's important to call good candidates *quickly* because they will probably be lost (someone else will hire them) if you delay.

I sort resumes into four piles: rejects, reject but keep in an active file, priority 2 and priority 1.

Rejects

Rejects get a form letter right away that says thanks, but no thanks. A common form letter says,

> "Your qualifications are impressive and we appreciate your effort in contacting us, but there is not a match between our requirements and your skills at this time."

I send a slightly-flattering letter like this to every reject, even the hopelessly underqualified and even the ones who have obviously lied on their resume (even the cretin who claimed to have authored a manual that I wrote). I am unfailingly polite. My goal is to spend a minimum of time and emotional energy on the rejects. My rejection is friendly and respectful for a few reasons. First, being told that they’re not going to get the job is bad enough. I’m not out to make the candidate’s life miserable, just to close this relationship. Second, even when the candidate is a liar or a creep and deserves worse treatment, I don’t see the payoff in making them mad. They get offended, make nasty phone calls, threaten lawsuits, and pester executives who then blame me for motivating this person to bother them. So I smile instead and send a note that simply encourages them to go away.

In many companies, the rejection letter is sent by the HR department. Check with them on company policy about rejection letters, but make sure that your company’s treatment of all candidates is polite. Over time, you will build a reputation as a recruiter and you want it to be a good one.

Here are examples of resumes that I will immediately reject:

- *Inappropriate behavior*, such as foul language, inappropriate gender references, or jokes in poor taste.
- *False statements or exaggerations.* Many resumes, perhaps 25%, contain lies or significant exaggerations. (xxx) I have no tolerance for these. By the way, I make these judgments quickly, and sometimes I might be wrong. I don't know of any requirement that I make a thorough investigation before privately concluding that someone made a false statement. For example, if a candidate claims to be an expert in Java but then misuses common phrases and appears to demonstrate fundamental ignorance of the platform, I won't spend time investigating further. I’ll simply reject the resume (but without sharing my private conclusion that this person is a liar.) On the other hand, I am always conscious of my duty to not discriminate against members of various protected groups—dismissal of a resume cannot be based on a stereotypic judgment like, "No woman could have led the testing of Word. She must be lying."
- *Clearly insufficient background.* If the ad calls for experience in testing or programming, and the resume doesn't list any, I usually reject it immediately. Not only is the candidate unqualified. He is evidently not reading or responding to the content of the ad. On the other hand, if the candidate writes a cover letter that admits that his experience is too thin, but says that he really wants the job, then I will read the resume more carefully. This tester is paying attention to what was said in the ad and responding to it head on. He is ambitious, he is being honest, and he is negotiating. I value those behaviors in a tester and so I will tend toward keeping him under consideration for some other position, if not this one.

- *Intolerance of my group's situation.* Some resumes state strong opinions about product development style or the mission of a testing group. This is not necessarily a bad idea but it has and should have two effects. It helps some readers recognize that this person might be a good fit. And it helps others recognize that they should interview someone else.
- *Spelling and obvious grammar errors.* Someone who doesn't take the time to check his own work (or get help from someone else) is unlikely to cut it as a tester. I make four exceptions to this generalization. First, if the candidate is a very recent immigrant, I might read the resume a bit more carefully before rejecting it. Second, if this is a very long resume, I tolerate an error or two as a normal bug rate. Third, if I downloaded the resume from an online service and I know that the resume was typed into an online form (which makes proofing and editing the resume difficult), then I'll ignore minor errors. And finally, if the candidate has clear, verifiably successful experience, then my prediction that he'll never be a good tester has been refuted. But in any of these cases, I'll question him closely and check his references carefully.
- *Remarkably insufficient information.* Some resumes convey so little information that I have no idea whether this tester is suitable for the position. If nothing in the resume tells me to be interested in the candidate, I reject it.

Reject but keep in an active file.

Some candidates are not appropriate for the current position but have distinguished themselves in some way that makes me want to keep their resume in an active file. For a different position, I might hire this person. The rejection letter for this person might say something like:

> "Thank you for your application. We have decided to consider other candidates for the position for which you applied, but we are impressed with your qualifications and will keep your application in an active file. We will contact you if a more suitable position becomes available in the near future."

Priority 2

This is a holding pile. I'm not enthusiastic about these candidates, but maybe they're more suitable for the job than I currently think. I won't reject them yet but I won't call them for interviews until I've explored the higher priority candidates.

Examples of candidates in this group:

- *Unusual situations.* Most of the "opportunity hires" start out in my second priority group.
- *Underqualified but within training distance.* This candidate doesn't meet the position's minimum knowledge / skill requirements, but it might be possible to train him into the role.
- *Recommended by someone significant.* This candidate was recommended by a staff member or a trusted colleague. Before I reject the resume (assuming that it is within the realm of possibility), I will look at it with some care.
- *Insufficient information*. Some resume styles (the very brief functional resume, for example) are uninformative. They hint at information about the candidate but provide little useful detail. I will almost always reject such a resume if the candidate is applying for a management-level position (he should have evaluated enough resumes as a hiring manager to know how worthless this one is). For a less senior position, I'll hang onto this resume *if* I don't have enough better candidates, and *if* some aspect of the resume makes it seem plausible that this candidate might be qualified for the position. My phone screen, however, will start with specific questions that are designed to quickly eliminate candidates who are unreasonable.

I've always had mixed feelings about the uninformative resumes. People who haven't looked for many jobs style their resumes according to advice they get, and they get bad advice from certain books and

recruiters. The advisor suggests that the candidate sprinkle the resume with buzzwords that a computer or an HR clerk can understand but keep to a minimum the information revealed in the resume, on the theory that employers usually use resumes to *screen out* a candidate rather than to qualify one. Under this theory, if a resume isn't eliminated, it will carry the candidate to an interview. In a sense, it is unfair to eliminate people who take this advice, because they are being rejected for something (taking bad advice from an apparently qualified person) that might not predict their success (or lack of it) on the job at all.

On the other hand, my experience is that most of these you-have-to-call-me-for-my-real-resume candidates are unsuitable. So why should I invest extra work calling them? Usually, I don't. But when it's hard to find suitable candidates, I'll work a little harder for anyone who might be suitable.

Priority 1

I call top priority candidates (for a phone screen) as soon as possible. Today.

EVALUATING THE CANDIDATE'S PUBLIC MATERIALS (F xxx)

Some candidates are published authors (in magazines, conferences, books). Some have web sites. Some are active on news groups or mailing lists (these are often archived). These writings reveal a lot about the candidate. For example, you are likely to learn:

- about the candidate's knowledge and writing style
- what the candidate claims in public about the nature of her job and experience (Is it consistent with the resume?)
- what issues seem to draw the attention of the candidate--what she thinks is important
- whether she engages in flame wars. What it takes in a debate to irritate her
- how much time she spends posting to newsgroups and (to the extent that you find out) to mailing lists and whether she is posting from a company e-mail address during normal business hours.

I prefer to read a candidate's material before a phone screen, but I often don't have time before the screen. I make the time to read the material before the face-to-face interview.

THE PHONE SCREEN (m xxx)

The usual point of a phone screen is to filter out candidates. The face-to-face interview is expensive and time-consuming for everyone. If this person is obviously unqualified or inappropriate, the sooner you close the call, the less time and money that you waste on him.

I also use the phone screen to learn more about the candidate, in order to better prepare for the face-to-face interview.

I allow ninety minutes per phone screen. The calls actually last between two and ninety minutes.

I have a list of issues/questions that I select from, and I often go through them in the order listed. But I might go directly to an issue if I have specific concerns or interests in that candidate.

For example, if I'm calling someone who submitted a low-information-content functional resume (he listed types of tasks that he did but didn't tell me where or when or provide much beyond buzzwords. He listed the employers and dates but didn't say what he did where), I might start by collecting a chronology. Where did you work? When? What Was your title? Who did you report to? What were your major accomplishments? Why did you leave? Or, I might start by asking about the claim or two that captured my attention. For example, I might say, "In your resume, you said that you have experience with QA Partner. Can you tell me when you use that tool and what you used it for?" If the answer is weak, and this was the key skill that led me to make the call, then I'll move into shutting down the call (see below).

Here's a list of questions that I put together over the course of recruiting a few test managers. I normally bring this list up on-screen, creating a candidate-specific file. I often type while I talk with the candidate. If I don't type, I take detailed written notes. I use a headset when I interview, so that my hands are free (for fast writing or typing) while we talk. You can get a good headset for $100. Your company can afford it.

If an issue is well answered in the resume, I won't spend much (or any) time on it in the phone call.

The questions listed here are questions that I am asking myself. They tell me what I'm trying to find out. I word the questions to the candidate differently for each candidate, following the flow of the discussion we've had so far and taking into account what I already know.

You might be required to ask all candidates the same questions in the same order. (Some companies that have been sued for discriminatory hiring practices, or that don't want to be, will adopt the rule that everyone during a screening is asked the same question.) If so, I recommend that you start the phone screen (after introductions) by telling the candidate that your practice is to ask everyone the same questions in the same order, apologizing in advance for any questions that cover issues well explained in their resume, or that reopen issues that they answered in a different response to a different question, or that seem out of context.

My list of questions for a test manager

Any question that you could ask a staff member is fair game for a management candidate. This list includes more questions than you will ask any candidate, and a lot of questions that you might find useful for building a list for any other position.

Part of your preparation for interviewing candidates could be to select a set of questions from this list (add a few of your own) that are appropriate for the position that you are trying to fill.

This list raises a wide range of issues that you would cover in your questions, but I've written the questions in third person *("how would he do this or that," rather than "how would you do this or that")* deliberately. That's because the questions are not intended as polished and immediately ready to use. You have to put them into your own language or they'll be artificial and ineffective.

The resume presentation

- Typos
- Consistency of presentation

If there are weaknesses in the presentation of the resume, I'll ask about them. For example, if there are typos or spelling mistakes, I might note them and ask why. The typo or spelling mistake might not have disqualified the candidate, but the candidate's response to this question might give me a reason to close the interview quickly. A response that no one cares about minor details, or a response that seems unusually defensive (the candidate asks, "How dare you ask about something like that?") will typically convince me to close the interview quickly.

Educational qualifications

- College or university studies
- Continuing education
- Books and publications read / written
- Conferences attended
- Professional societies—member / activity
- Standards committees—membership / what did they do

- Awards received by the candidate or his staff

How has this person learned about testing and about software development in general? I ask every testing candidate about education. The more senior position that the candidate is applying for, the stronger my feelings are that he should have been actively broadening and deepening his knowledge of the field.

I ask about each of these types of education. There are few university courses on software testing, and in some communities they are not the best available sources of education.

Even at the test manager level, the majority of candidates that I screen have never read a book on testing, never taking a university-level or university extension class on testing, and never attended a conference on testing. Their training has been completely in-house.

This group of questions also gives the candidate a chance to tell me about his commitment to the profession. I don't expect the candidate to be active in the IEEE, ASQ, ACM, etc.. I don't expect the candidate to work on the development of professional standards. But if the candidate does this, I want to know what he does, what he's been learning from it, and I want a sense of whether this candidate offers too much of a good thing. Some candidates expect to do this work on company time, for many hours per week. That might or might not be acceptable for the open position.

Employment history. For each employer:

- The basic data: company / dates / title / role / supervisor
- What kind of products he worked on?
- What interesting technology he used or developed (that he can talk about)?
- What worked well?
- What didn't work well?
- What he did that was special?
- What approach to continuous improvement?
- Why he left?

The question on continuous improvement is often informative. Sometimes I ask this up front, before asking about specific companies. I want to know how he monitors and improve his own work and the work of his staff (if he is a manager).

Approach to testing

- What is SQA? *(If he says, "huh?", I ask, what does the term "Software Quality Assurance" mean to him?)*
- What is the purpose of testing?
- What is the value of the testing group? How does he justify his work and his budget?
- Why should testers do testing in addition to testing done by the programmers? Has he ever seen or worked with development groups that didn't have testers? How did it work out?
- What is the proper role of the test group?
- What is the role of the test group vis-à-vis documentation, tech support, etc.?
- How much interaction with end users should testers have, and why? How should he learn about problems discovered in the field, and what should he learn from them?

- What does he know about testing tools? Can he give examples of glass box / black box / gray box tools? Has he used them? Did they work?
- How much test automation should a test group do, and what type and why?
- Development model? What should the programmers use? What should the test group use? *(If he doesn't understand the question about what development model the test group should use, ask what development model he uses for test automation. If he doesn't understand what "development model" means, ask him how he or his group have developed test automation in the past, what process they used, and what process he thinks they should have used.)*
- How did he get programmers to build testability support into the code?
- Role of bug tracking system (track defects? / flag personnel issues? / generate metrics? / record design bugs? / other roles?)

I ask these questions of supervisory candidates or of senior individual contributors. I am not looking for The One Right Answer about How Testing Should Be Done. I primarily want to know if this candidate has thought about these issues in any depth.

I am also be trying to learn whether his views are roughly compatible with the company's. Throughout this particular series of questions you see a bias of mine toward testing, with little regard to process standards. I listen to the answers about SQA and role to hear whether this tester will work happily in a group that does not follow a process standard like ISO 9000-3 or CMM.

What if you are recruiting a manager for a test group that does follow process standards and that sees the testing (QA) group as having a different primary role than testing? In that case, you should modify this list to meet the requirements and atmosphere of your company.

My intent with the tool / technology questions at this point is to learn the candidate's attitude about the role of technology in testing. I'll cover details later.

Knowledge of areas of testing

- What are the key challenges of testing?
- Has he ever completely tested any part of a product? How?
- Has he done exploratory testing effectively? *(What does he think exploratory testing is?)*
- Has he done specification-driven testing effectively?
- Should every type of business test its software in the same way?
- Economics of automation?
- Role of metrics in testing? *(What metrics has he used? Were they any good? Why? How did he know?)*
- Describe components of a typical test plan?
- Tools for interactive products?
- Tools for database products?
- Has he used / can he describe cause-effect graphs?
- Has he used / can he describe data flow diagrams?
- What is model-based testing? What are state diagrams? Has he used them?
- What is stochastic testing? In general, has he tried any approach to high-volume testing?

- What are the challenges, risks, and benefits of regression testing? When should it be used?
- When did he last have to focus on data integrity?
- What is root cause analysis? Has he done it? Should a test group do it? When?
- Does he know what API-level testing is? How much API-level testing has he done?
- What is load testing? In putting a system under load, how does he decide what patterns of activity should go into the load?
- A program under test interacts with users, with other programs, with hardware, and with the file system. Give examples of tests for each type of interaction.
- How should a group review software before final release?

Testing skills

- What steps does he take to perform a test?
- What is a test case?
- What is the most important type of bug? Why?
- What are the important conditions to test for?
- What is _____ and how would he test for each? *(race condition, boundary, control flow, etc.)*
- What kinds of tests has he developed?
- Has he written automation code? How did he test it? Document it? Put it under source control?

Windows (or other platform-specific) skills

- What tools work well on his platform? Which ones does he use? Example?
- What does he use for a reference for this platform?
- What types of defects are particularly common on this platform? Why?
- Has he written code for this platform?
- How does a typical program interact with the file system?
- What makes bugs particularly hard to reproduce on this platform?
- Has he done debugging on this system?

What's his approach to reporting bugs?

- On paper
- In person
- What does he do after he finds a bug? *(For example, looking for more bugs in the same place, looking for related bugs, troubleshooting or clarifying the bug?)*
- What if the bug is hard to reproduce?
- How should a tester respond when the programmers choose not to fix a bug?
- When should a test manager be brought into a discussion of a decision to defer a bug.

- Can he describe the typical types of bugs at his last company? *(Ask "Can you describe" rather than saying "Describe" because he might feel that such a description would breach his nondisclosure agreement with his last employer. Some companies consider this very sensitive information. Let the candidate make the decision about what he can say and what he can't.)*

Database testing

- Has he done any? When? Why?
- What ones did he test? How did he test them?
- What types of failures did he look for?
- What did he do (or what should one do) to measure testing coverage in a relational database with many ways to view data and many interrelationships among data items?
- What is a data flow diagram? When is it useful?

This list illustrates the questions that I ask. The actual list that I would use will depend on the company, application areas, etc.

The question, "Should every type of business test its software in the same way?" provides some indication about the candidate's open-mindedness and about the breadth of the candidate's actual education and exposure to the field. I hope to hear that life critical applications probably go through more rigorous testing and process management than here-today, new-version-tomorrow web-based applications. I would like to hear that different application issues call for different approaches. For example, the techniques that you apply a financial application (written in COBOL, doing fancy stuff with a huge database) differ from techniques to test the interactive competence of a word processor. If I'm lucky, I'll hear the candidate talk about the different paradigms of software testing (the different ways that people think about the core issues of the field). See Kaner & Bach (1999) for a discussion of this in the black box testing world. I don't think that there is one right partitioning of paradigms, but it is a mark of maturity in the field to recognize that two different groups can have substantially different views of what is a good approach to testing, and both can be right (given their context).

For a senior candidate (individual contributor or test supervisor), I want to find out what they think about various common issues in testing. How sophisticated is their thinking? Not whether I agree with them, but whether they have a well developed point of view. I also want to give them a chance to describe and evaluate the tools that they've used.

The data-oriented questions are examples of the types of questions that I ask in order probe sophistication in the testing of an application area. For a different class of applications, I'd ask different questions. For example, a highly skilled tester / test manager for interactive applications (games, word processors) might know little about high end data storage or financial applications. There's not much value in asking that person about databases and their test tools.

The "typical bugs" question is trying to get at the underlying question--"What kinds of problems with products are you used to dealing with?"

Interest and skill in this company's areas of application

- Product-category specific questions

This section gives the candidate a chance to show me that he is a subject matter expert.

Project Management

- How does he prioritize testing tasks within a project?

- How to develop a test plan and schedule? Tell me about bottom up vs. top down approaches?
- When should he begin test planning?
- When should he begin testing?
- Does he know of metrics that help estimate the size of the testing effort? How does he scope out the size of the testing effort?
- How many hours per day should a tester work?
- How should staff overtime be managed?
- How should his overtime be managed?
- How to estimate staff requirements?
- What to do (with the project tasks) when the schedule fails?
- How to manage conflict with programmers?
- How does he know when the product is well enough tested?
- How does he allocate (% of time) his (and his staff's) testing effort to different types of tasks?

These questions are primarily for mid-level to senior testers and for supervisors. At some point in seniority in many companies, a tester becomes largely self-managing. For example, the tester is assigned to a fairly large area of work and left pretty much alone to plan the size, type, and sequence of tasks within that area. Drucker (1966) wrote a remarkable book on time management, decision-making, prioritization, and survival skills for managers. Drucker includes any knowledge worker who has to manage his own time and resources within his definition of "executive." One of the successes in the development of my management style comes from learning to see the managerial nature of my mid-level individual contributors.

Staff relations

- What characteristics would he look for in a candidate for test group manager?
- What does he think the role of the test group manager should be? Relative to more senior management? Relative to other technical groups in the company? Relative to your staff?
- How do his characteristics compare to the profile of the ideal manager that he just described?
- How does his preferred style work with the ideal test manager role that he just described? What's different between the way he works and the role he described?
- What qualities are important in a tester? Who to hire in a testing group & why?
- Role of metrics comparing staff performance in HR management?
- How to estimate staff requirements?
- What to do (with the project staff) when the schedule fails?
- Tell about staff conflicts that he's handled?

This section is primarily for supervisory staff.

I ask the four test group manager questions only of management candidates. These four questions are enlightening, during the phone screen and during the face-to-face interviews. I think that it's entirely fair to ask these of someone who has management and hiring experience, and I expect thoughtful answers. Here are examples of some of the insights that I can get from these answers:

- *Suppose the candidate's picture of an ideal manager is dramatically different from his image of himself, or from the impression of him that you've built up during the interview. This can be a huge red flag. Not always. In some cases, for example, this reflects genuine humility. But a significant mismatch should make you think. And it should help you structure questions for the face-to-face interview.*
- *Suppose the candidate's description of the ideal manager exactly manages his perception / presentation of himself. This person might not be pathologically egotistical. He might just be trying to manage an interview in a way that puts himself in a good light—I think this is OK, as long as he doesn't lie or exaggerate. But again, it gives me a lead on future questions.*
- *Suppose the candidate's description of the ideal manager differs strongly from the expectations of your company. I expect to see some differences, but if there are fundamental differences in expected role or in expected relationship with the staff, then I will wonder whether this person can fit with the company. Wonderful, brilliant people might fit perfectly in some companies and poorly in others.*

Interpersonal skills

- Tell something about himself and what he's done in the past that he could bring to this position.
- Why does he want to work in testing?
- Tell about a difficult testing issue that was deferred and that he succeeded in getting fixed.
- How would he handle the situation if someone asked him to withdraw a bug that they wanted to go away but he wanted to fix? How would he deal with that person? Has he been in this situation? What happened?
- Tell about a project that was short on resources and how he handled it?
- Can he recall an instance in which he had to compromise his personal quality standards? What happened? How did he handle this?
- How does he see himself in a small (mid-size, big, whatever) organization?
- Does he do well in more (less) structured environments?
- If the work flow in a project is not well defined, how does he work out with other people what tasks are due to him (or from him) and when?

Goals

- What do he see himself doing next year? Five years from now? What is his plan for achieving that?
- Has he supervised other staff? How did he like it? *(Is the candidate interested more in a managerial track or a technical track?)*
- What testing tasks does he find the most interesting? What non-testing tasks does he find the most interesting?
- How would a position like this one fit with his lifestyle or his broader interests?

Attitudes

- Tell about an accomplishment that he is truly proud of.

- What motivates him to improve and progress in his career? Why is he motivated by these? What demotivates him? *(Behind these questions, I'm asking, "Is he self-directed?")*

Initiative

- What steps has he taken to enable him to be more effective in his job?
- Would he rather design and develop plans and procedures or implement and manage them? Why? Examples?
- Does he prefer to run projects himself or to get direction from others? How independently does he like to work and what challenges has independence posed? Examples?

Miscellaneous

- What professional situations have caused him to feel awkward?
- Define integrity
- What has he been criticized for in the past two years? How did he respond? What did he learn from it?
- Describe his most and least ideal boss.

Knowledge of the company

- What does the candidate know about the company?
- What questions does the candidate have about the position or the group?
- What questions does the candidate have about the company?

For example, if the company has a web site, I'll ask the candidate whether he has looked at it. Not many people will have looked at the site, but that's OK. By asking the candidate during the phone screen, I set up a follow up question for the face-to-face interview. If I ask this question during the phone screen, and he hasn't looked at the site by the time he comes for the interview, that's not a good sign.

I might also offer to send marketing materials or company profile materials to the candidate before the face-to-face interview. I will certainly agree to this if the candidate asks me for the material. This helps the candidate prepare his questions (he should have some).

THE APPLICATION FORM (F xxx)

When the candidate comes in for an interview (or you can mail or email it to her in advance), she should complete an application form. Your HR people probably have one handy.

The typical form sets the candidate's experience out in chronological order and asks a variety of other standard questions about her background. It is useful to interviewers to have this information in a standard format.

This form is particularly important when interviewing someone who gave you a functional resume. People often select that format in order to hide problems in their chronology or to make it easier to exaggerate what they have done across companies.

For more on standard application forms, see Rosse & Levin (1997, Chapter 7).

THE INTERVIEW: PREPARING FOR THE INTERVIEW (m xxx)

Over a period of one to three days (perhaps spread over a few weeks), a candidate will meet with up to ten (maybe even more) of your company's staff. This is expensive. It interferes with other work. You should prepare that time in advance, so that you spend it effectively.

Every interviewer needs some basic information, such as a copy of the advertisement, the candidate's cover letter (if there is one), and the resume. If the candidate's resume is light on detail, then your chronological notes from your phone screen are valuable.

There must be an interview schedule. People have to know when they will meet with the candidate and who they will bring the candidate to. You probably have to book the appropriate conference room in advance. You might want to make restaurant reservations in advance, so that your staff and your candidate don't waste time standing in line waiting for a table.

If you're going to give the candidate a demonstration of your products, you should have a machine set up and available for the purpose.

You will find it useful to review your notes and decide what classes of characteristics are most important to interview for. (See the discussion in the next section.)

There's value in having a brief pre-interview meeting to divide tasks and establish ground rules. For example, how independent do you want peoples' impressions to be? In your process, once someone has interviewed the candidate, can she discuss it with someone else who has interviewed the candidate? Can she discuss it with someone who has not yet interviewed the candidate? Under what circumstances can she discuss the interview with non-interviewers?

It's valuable also to educate your staff so that they recognize some common mistakes in reasoning or in appraising candidates, and so avoid them. For example, I try to help people understand:

- *Someone can be a wonderful person, very bright, and very competent, but still be inappropriate for the position at hand.* A rejection of the candidate's suitability for this position is not a rejection of that person.
- *A candidate can be weak in some areas even though he is superb in others.* (This is the problem of the "halo" effect.) For example, a person can be analytically talented, a solid mathematician, but incapable of putting together even a simple test plan, even a simple sequence of relevant tests. (Hard to believe? It was for me, too. But this is a real case.)
- *A candidate might be weak even though he has great credentials.* It's remarkable what glowing letters of reference a company will give a problem employee as part of the process of convincing the employee to go away. It's remarkable how many great companies some losers can amass on their resume. Once they start at a well-known, high quality (of testing) shop like Adobe or IBM or HP or Microsoft, other companies will hire them.
- *A candidate might be unacceptable even though he doesn't look unacceptable at first glance and you are desperate or sick of interviewing.* At some point, you might feel like you'll hire the next candidate who can prove that he can breathe. Don't do this. If you have to hire in desperation, bring on a short term contractor to fill the seat. Find the best contractor that you can, pay what you have to pay, and buy yourself some breathing room.
- *A candidate who acts poorly during the interview won't somehow improve when you hire him.* If he shows up late for the interview, with no good excuse, why do you think he'll keep these types of commitments later? If he makes promises that he doesn't keep, if he doesn't do his homework, if he fumbles the assignments or snaps at people under pressure, he's showing you what you're going to see later. He's showing you the person that you're going to work with. If you don't like that person, don't hire him.
- *A candidate might be acceptable even though he is not a perfect match for your fantasy of the perfect candidate.* Nobody will perfectly match your ideal candidate. Remember the wisdom of

the Rolling Stones: "You can't always get what you want, but if you try sometime, you might just find that you can get what you need." *Imperfections are normal. We are human. People will come to you with weaknesses as well as with strengths. You want to hire someone with the strengths that you need and the weaknesses that you know how to manage.*

THE INTERVIEW: DIVIDING ISSUES AMONG STAFF (F xxx)

You want to know what this person knows, and how she thinks, and what her skills are, and whether she's a decent human being, and whether you can work with her. That's a lot of questions. You have the huge list of nice-to-have characteristics that I provided above (plus others that you've added). Which of these are important? Make a much shorter list. For each item on the list, ask two questions (and get them both answered) in the pre-interview meeting:

- How are we going to find out about this?
- Who is going to find it out?

For example:

- One interviewer might demonstrate the product to the candidate. This is partially an important courtesy to the candidate, but it also gives the interviewer a chance to watch how observant the candidate is. Does the candidate ask questions? Does she try things? Does she take notes? Is she interested?
- One interviewer might focus on the technical programming and design knowledge of the candidate who claims to be a competent programmer. The same interviewer (perhaps) or someone else might focus on the candidate's knowledge of specific test automation tools.
- If the candidate claims to have specific knowledge of your development environment, you might want to solicit one of your company's senior programmers to interview this candidate. In general, I like having programmers on the interview team (the tester has to be able to communicate with, establish rapport with, and build credibility with programmers). But when specific claims are made, it sure is helpful to have someone with specific expertise evaluate those claims.
- One interviewer might focus on the test planning aptitude of the candidate, perhaps using some test planning exercises.
- One interviewer might focus on the subject matter expertise of the candidate (if she claims to have such knowledge).
- One interviewer might ask questions focusing on how well a test manager candidate will train staff, support growth along their career paths, and provide them with growth opportunities. Maybe this person is also appraising negotiating skills, integrity, management of staff under difficult circumstances.
- One interviewer might serve as a guide, walking the candidate through the building, walking her from interview to interview, and answering any questions that she has. The guide is at a peer level to the candidate and makes it clear that she'll be glad to answer questions. The candidate might feel more comfortable asking questions of one person who feels more like a host than an interviewer. The guide answers the questions, but also reports the questions back to the group. Perhaps this person asks a few questions of her own, gently probing the inquisitiveness of the candidate.

Unless you explicitly note the issues, you won't even realize that you're missing many of them. Unless you assign them intentionally, you won't cover them all.

(Gosh, it's just like test planning.) (But of course. You are conducting a series of tests of a very complex subject matter. Coverage is an important issue for testing candidates, just as it is for testing software.)

No group is perfect at this the first few times they interview a candidate. But you can get better at it by walking through the issues again in the post-interview meeting.

THE INTERVIEW: QUESTIONS (f xxx)

I can't begin to list all of the interesting questions that you can ask in an interview. Some of my thoughts are reflected in the issues that I suggested for the phone screening. These are all good issues for the main interview. The main interview should also look at skills (by demonstration as well as by discussion) and detailed knowledge.

I refer to "issues" instead of questions because you can ask very different types of questions to get at the same issue.

Here are some of the key types of questions:

- Hypothetical vs. Behavioral
- Factual vs. Opinion
- Closed vs. Open-ended
- Traditional interview questions

Hypothetical questions

The hypothetical question is a what-if question. You describe a situation and ask how the candidate would deal with it. Normally, the candidate can ask you any questions that she considers appropriate, and then she frames her answer. You appraise the answer and, if you're paying attention, you also take note of the kinds of questions she asked. These tell you something about the analytical approach of the candidate.

I like to ask a few hypotheticals, not because I'm necessarily interested in the answer (I ask behavioral questions when I'm really interested in the answer), but because I am interested in seeing how the candidate gathers information. So my hypos lack some critical details.

And, of course, sometimes the answers are informative too. But often, the answers reflect what the candidate thinks you'd like to hear, or reflect an ideal situation rather than anything that the candidate has ever achieved or even attempted.

Behavioral questions

The behavioral question probes the candidate's actual experience.

For example, suppose that you want to appraise a candidate's understanding of bug tracking system design. You might ask:

(Situational)

> "Suppose that someone asked you to design their bug tracking system. What would you suggest as the most important characteristics of the system?"

Or you might ask (behavioral):

> "Have you ever had to design a bug tracking system? How did you go about deciding what were the most important characteristics of the system? What were they? Did you actually succeed in building them into the system? How well did the system work? What did you learn that would help you design a new system in the future?

Rosse and Levin (1997, p. 173) provide more examples of what they call situational and behavior-descriptive questions. Risser (1993, pp. 150-152) provides more examples and additional useful discussion.

When interviewing a test manager candidate, I spend time creating some behavioral questions that tell me how this person has handled problem employees. There are several variations, including the employee who makes a political mess by criticizing the product's (or the programmer's) quality at an inopportune time, or the employee who has been a great worker for a long time but has developed a drinking problem, or the employee who is disillusioned with the company and is spending more time visiting with other staff and complaining than on getting work done. I want to be able to predict how committed this manager is to supporting, defending, and growing her staff, and I also want to be able to predict how she will demand discipline when she must.

Factual vs. opinion questions

If I ask someone to tell me what they know about ISO 9000-3, I might be asking for their knowledge ("What is it?") or for their opinion ("Do you like it?") Ideally, I would know which question I'm asking before I ask it. If so, maybe I would ask the question better: "Can you describe ISO 9000-3?" or "What do you think of ISO 9000-3?"

Factual questions are important. At least one interviewer should ask several of them, to test the candidate's detailed knowledge of an area.

- Some candidates for a lab technician's role know a lot about printers, video cards, and other peripherals. If you don't ask, you'll never realize that one person is an expert (even if she is modest) whereas another person is only slightly knowledgeable (even if he is boastful).
- A candidate who claims to know a lot about QA Partner ought to be able to answer questions about its syntax, bugs, special capabilities, and the ways that people use it to create test suites optimized for different characteristics (maintainability, speed, re-use of scripts for foreign-language versions of the software, whatever). Don't just let the candidate tell you the (only) three things that she knows. Ask her questions that she doesn't volunteer the answer to.
- A candidate who claims to be active in the software quality community and interested in promoting the professional development of her staff ought to know who the main professional societies are, what the differences are between the certifications (such as ASQ's CQE or CSQE and QAI's CSTE and CQA), what conferences and courses are available to staff, etc.

Opinion questions are useful too:

- The candidate's opinion might be important. For example, if your company is committed to an ISO 9000-3 program, and the candidate thinks that there's nothing wrong with this standard that you couldn't fix with a shredder and a magnet, then you probably don't need to spend much more time interviewing her.
- Often, the question is not whether the candidate's opinion is right or wrong but whether the candidate forms opinions thoughtfully. For example, suppose that you ask the candidate about ISO, and she gives a negative opinion. Your next question might be, "Why do you think that?" Some people turn out to despise ISO 9000-3 because that attitude is fashionable in some circles or because their manager told them it was stupid or because they think that's what you want them to say. Some other people despise ISO 9000-3 because it was badly applied at a place they worked, and they've seen it badly applied elsewhere. They even read a book about it. Some people have a large set of thoughtless opinions. Others form opinions more carefully. If I'm hiring a lab tech, maybe I don't care. But if I'm hiring a senior tester, I want the one who is committed to knowing what she's talking about.

Closed and open-ended questions

A closed question calls for a yes or no answer, or a very short factual answer. Occasionally they are useful. Usually, they are the product of an untrained questioner. Closed questions often tell the candidate what answer you expect to hear, and so she just agrees with you because that's all you're calling for. Other closed questions are hostile in tone or nature (these are the questions used in cross-examinations in

court) and they make people defensive. The candidate feels as though you are trying to put words in her mouth. (You are.)

An open-ended question calls for a broader or more detailed answer. It calls for more input from the candidate and provides very little input from you.

For example, (closed question):

> "You agree, don't you, that maintainability is one of the most important characteristics of an automated test suite?"

For example, (open-ended question):

> "What do you think are the most important characteristics of an automated test suite?" (followed up with) "Why?"

Traditional interview questions

Here are some of the traditional "good" interview questions. Note that they are all open-ended, which is good.

- Tell me a bit about yourself.
- What are your strengths?
- What are your weaknesses?
- How would you feel if one of your subordinates was promoted to a position above yours?
- Why did you leave your last job?
- Why are you interested in joining our company?
- What makes you interested in software testing?
- Where do you see yourself X years from now?

It's probably worth having someone ask them, just to hear the answers, but many people practice canned answers to these things. Most people (of those who practice answers) practice their own answers, with their friends. But some people look for standardized answers that will be socially acceptable. Rothstein (1996, p. 13) (a book of standard answers) suggests the following answer to the question:

> "Q. How would you feel about one of your subordinates being promoted to a position above yours?
>
> "A. I guess it would depend on who it was and the circumstances in which it happened. If I had honestly felt that the person deserved the promotion, I might be a little jealous, but I'd also be among the first to congratulate him or her. But if I had reason to believe that it was due to backroom politics or personal favors, I'd probably be very angry."

By the way, let's turn this into a behavioral question.

> Q. Has a subordinate ever been promoted above you? (If yes) What happened? How did it feel? (If no) Has this happened to a friend of yours? How did they take it? How would you have felt in that situation?

THE INTERVIEW: WORK SAMPLES (M xxx)

When I schedule an interview, I ask the candidate if he has any work samples that he can bring in. I very carefully don't ask the candidate to bring in any secret documents. I ask if he has anything that he's done that he can share with me.

Confidential work samples

When the candidate arrives on interview day, I'll briefly look at what (if anything) he's brought. I won't look at the details yet. If there are any apparently confidential materials, I have to manage this issue with care. The conversation might go like this:

> Q. This is the test plan for your current product? Cool. Have you released the product yet?
>
> A. No, we'll probably release it in a few weeks.
>
> Q. Oh. Wow, I really appreciate your bringing this, but does your company consider these documents confidential? Would they mind if I looked at them?
>
> A. Well, yes, they probably would. But you asked for my work and this is what I have. What was I supposed to do?
>
> Q. I understand. I'm very sorry about confusing you. We never look at confidential documents. Would you mind putting them away and not showing them to my staff during the interviews? I could get into trouble with my management if my staff look at this.

Notice two things:

- My staff aren't going to see these documents and I haven't looked at them carefully enough to learn anything from them (beyond the fact that they are confidential).
- I've avoided blaming the candidate or calling him a dolt for bringing in company confidential documents. I want to minimize the extent to which the candidate feels awkward over this. I don't want him to blow the interview because he feels bad about this.

Now, behind this, I have to decide what to do about this candidate.

- If I decide that this candidate is probably naïve, then I won't hold this against him. We'll have a long talk about trade secrets at hiring time, and more training later.
- If this is a management candidate, he should know better than to make personal use of company secrets. The odds are high that I will disqualify this candidate.
- If I don't know what to think about this candidate, I'll find some way to probe further on questions of integrity.

One last thing to notice. I didn't ask for confidential materials, but I haven't made a point of saying "Don't bring confidential materials to the candidate." This is another case of letting the candidate show me who he is and what he does. By the way, if he asks, "Do you want me to bring confidential materials?", then of course I say, "No."

Usable work samples

Suppose that the candidate brings stuff that you can look at. Publications, or generic charts, or test documentation that his manager has agreed he can use. Then you want to review this material in detail.

- Read some of it. Ask questions about it. What is special about it? What was challenging?

- If this is a publication, ask for the story behind its development. Does the candidate publish frequently? Why did he publish this piece? What started him thinking about the problem? What research did he do for it? Why?
- If this is a set of test documentation, skim pieces of it and then ask the candidate to walk you through it. ("You" probably means, one of the senior testers on your staff.) What was challenging about developing test documentation for this product? What was particularly useful about this documentation? Did the candidate keep it up to date? How would he do a better job next time?
- If this is test documentation, probe it a bit. Try to think of the kinds of bugs that *could* come up in a product like this. Then ask what test cases would have revealed these bugs.
- Remember that some of the strengths, and some of the weaknesses, of this test documentation come from the tester and some of the others come from the tester's boss. Treat the candidate with respect, even if the document is poor. Ask what the constraints were on this project and what he would have liked to have done if he had more time.
- Ask what was the purpose of this document. How was it to be used? Did it meet the company's needs? This is a particularly interesting question if the document is weak because, in context, it might have been entirely satisfactory. Don't be hasty to form a negative judgment.
- Be courteous with this document, *especially* if it is weak. Firmly resist the temptation to lecture on how this should be done, or how you would do it at your company. This candidate is sharing something with you out of his private files. If you embarrass him over it, he'll remember that. This is still a small community. Don't make enemies.

THE INTERVIEW: SAMPLE TEST DOCUMENTATION (F XXX)

You might have documentation that is simple enough for the candidate to review. If so, have her inspect your document and criticize it. This is primarily interesting if the candidate claims to be good at auditing or inspecting testing documentation. Let her demonstrate her skill.

If you are going to use sample test documentation, I suggest that you create a standard set for interview purposes. There are some useful characteristics of the standard set:

- The sample has been "scrubbed" of confidential information. Maybe you will keep some confidential details, but you will remove or disguise or change the important ones.
- One sample should perhaps be long and detailed and formatted and long and very detailed. Make it obviously defective in some way, such as by deleting some sections that you think are important and that any good test auditor should think about.
- One sample should be short, perhaps three pages long. This might be a fragment of a longer document, but it should be complete enough to give the reader a sense of the quality of that document.

The advantage of using a standard set is that over time, you'll come to know the range of responses. It helps a lot to be able to compare the responses of a given candidate to others that have come from people you respect.

Additionally, there is value in giving these documents to your own staff and keeping track of their responses. Not because you are evaluating your staff. Don't do that. Instead, because it helps you understand where this candidate fits relative to your current staff.

Standardized documents help you avoid some mistakes. It is too easy, in a free-form evaluation that uses different documents for different people, to set your testing standards too high (thinking that your current staff or your peers meet them) or too low, or to interpret some answers as strange (even though they would be fairly common if you asked several people).

THE INTERVIEW: AN AUDITION (m xxx)

DeMarco & Lister (1987) recommend that you hold an "audition" for candidates. The candidate comes to the interview with a prepared 10 or 15 minute presentation on some aspect of past work.

I would include anyone who wants to see the audition as attendee and would allow them to ask a few clarification-type questions, but I would be cautious about the I-disagree-with-you type of questions. This can be a very threatening or intimidating situation for some candidates.

The more that this candidate will have to present material in public or the more experienced in presentation that this candidate is, the more appropriate I think that this interview style would be and the more willing I would be to allow critical questions. It is another type of sample of the candidate's work. As DeMarco & Lister put it, you want to see a juggler juggle before hiring him.

THE INTERVIEW: BUG REPORTS (F xxx)

It's fun to talk about how to write a bug report. It's interesting to have the candidate actually write one. I think that this is a particularly important test for testers who have a few months or a few years of experience. You'll find a lot of variation in how well they can do this, one of the most important and most basic parts of their job.

Find a reasonably straightforward bug in part of your software that is reasonably easy to understand. If none of your product's bugs fit that bill, get one from www.bugnet.com.

In my course on black box testing, I demonstrate a simple bug in Windows 95 Paint:

(1) Start the program.

(2) Color the background black.

(3) Zoom 200%.

(4) Select an area using the Freehand Select tool.

(5) Hit Del to delete the selected area.

RESULT: Either nothing gets deleted or some other area (lower and to the right) will be deleted instead.

In class, I also show a few additional (irrelevant) steps. I do some moving and deleting at 100% zoom first (nothing bad happens). I draw a circle and run all of my tests by selecting around the circle (even though the circle itself is completely irrelevant.) I show that the bug occurs with deleting, but not with moving, and that there is no failure if you move an area and then delete it. I draw the circle in the lower right corner, where it appears that nothing gets deleted (rather than the wrong area being deleted). Then I grow the window (so you can see the whole canvas) or select an area up and to the left, and the wrong area gets deleted. I demonstrate the bug, give students screen shots of all the steps (there are 15 screens), and ask them to write a bug report. I walk through the room answering individual questions (including "What would happen if I did this test … ?")

Even for this very simple bug, there are stunning individual differences among the students (who are usually experienced testers).

I've used this bug with perhaps 500 students by now. Some students can write a good bug report in 5 minutes. Others are still struggling after 30 minutes. Some write effective summaries and describe the bug step by step. Others, even some articulate people with 10 years experience, write a long, disjointed paragraph that is hard to understand.

It's easy to recognize a really good bug report, but even at leading software companies, a large percentage of the students (usually 6 months to 20 years of testing experience) don't write really good bug reports. It takes a while before you can tell the difference between a pretty good report, a not-so-bad report, a mediocre report, and a bad one.

Try your bug on your staff, so that you have a sample of reports for comparison, before trying it on interview candidates.

THE INTERVIEW: TESTS AND PUZZLES (m xxx)

Several groups do some type of informal aptitude testing, using logic puzzles or numeric puzzles. I don't object to these, but I don't think that they are as informative as some people think they are. Here are some of my concerns:

- There are huge practice effects with logic and number puzzles. I used to do them with my daughter when she was about 12. She got pretty good with them. That didn't mean she was smarter, and it didn't make her a better tester. It meant that she was better at solving puzzles. These practice effects are the basis of the large industry of test preparation for SAT, LSAT, GRE and other standardized college admission tests. Practice effects (previous experience) last quite a long time and they are more pronounced in speeded tests. They are more pronounced in nonverbal tests and performance tests (Jensen, 1980). So, a person who looks really great on these tests might simply be more familiar with them. A person who looks like a dummy might have no experience solving them but (in my experience) be smart and an excellent tester anyway.
- Speed tests select for quick but not necessarily for thorough thinking. Mental rabbits. Tortoises sometimes design better products or better strategies for testing products.

A simple testing puzzle

Another old favorite among commonly used speed tests is Myers' (1979, p. 1) self-assessment. The candidate is given an extremely simple program and asked to generate a list of interesting test cases. The specific program involves an abstraction (a triangle).

I prefer this because it tests something that testers will actually do (analyze a program and figure out ways to test it). However, there will still be practice effects. Average testers who worked through Myers will probably do better than strong testers who have never seen the puzzle.

Additionally, I suspect that among skilled testers there will still be cultural differences in success with this test. I suspect that someone who is used to dealing with abstractions, such as geometric abstractions, or with logical relationships among numbers, is probably going to do better than someone who tests user interfaces or compatibility with devices.

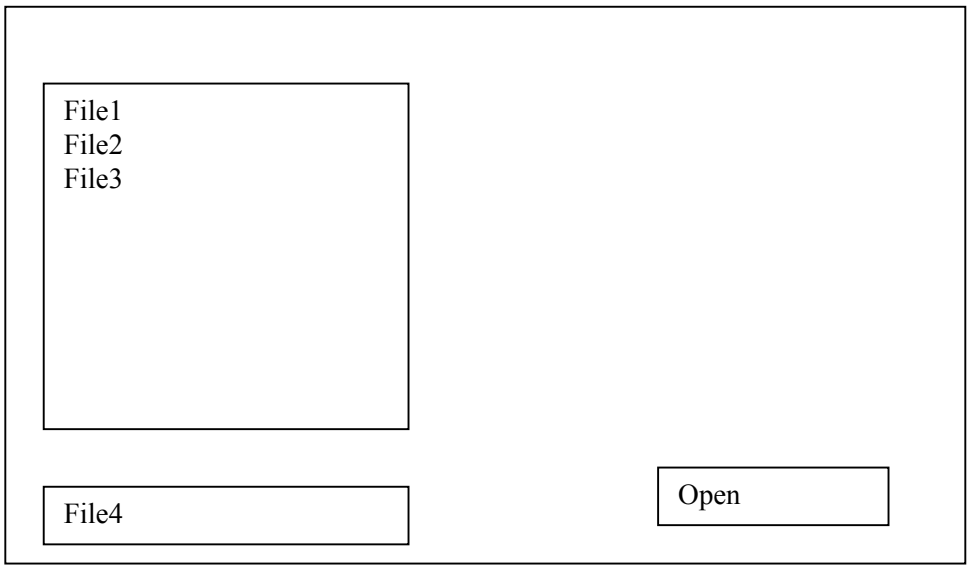
"Better" is a funny word, too. James Bach uses this example in one of his classes and did baseline interviewing of senior testers as part of his preparation for that class. Here's one of the answers that he got:

> Q. What tests should we run?
>
> A. I don't know. What will this program be used for? What are the consequences of failure?

Seems to me that we might test the program a little differently if we were using it to control a nuclear reactor rather than a computer game.

Another simple testing test

Here's an illustration of a test that I use that allows for cultural variation.



1. I draw a simple Open File dialog on a whiteboard.
2. I explain the dialog. This is an open file dialog. You can type in the file name (where it says File4 at the bottom) or you can click on the file name in the file list. Once you've selected the file, you can click on the Open button to open it.
3. I hand the marker to the candidate and ask him to tell me how he would test it. I make it clear that he can have as much time as he wants, and that many candidates take several minutes to think before they say anything.
4. The candidate can make notes on the whiteboard or on paper.
5. The candidate eventually begins presenting his thoughts. I listen, ask questions to clarify, but don't criticize and don't challenge. When the tester pauses, I let him be silent (to think) without saying anything. He can tell me when he's done. If it's ambiguous, I ask him if he has any other thoughts.

This is a remarkable test in the extent to which answers vary.

- One candidate might stay at the surface, pointing out every flaw in the design of the this dialog. (There is no Cancel button. There is no dialog title. There is no obvious way to switch directories. And on and on.)
- Another candidate might skip the UI issues altogether and try testing the opening of big files, little files, remote files (specified by paths that she types into the file name box, such as, d:\user\remote\fubar\File4), corrupt files, files with inappropriate extensions.

There are several other patterns. I think of these as cultural patterns because they reflect a cultural difference across platforms or user communities. For example, back in the days before Windows was a big deal, testers with Mac experience tended to focus on the user interface design, and testers with PC experience tended to focus on reliability of opening different file types from different places but were relatively blind to dialog box design. Testers with Amiga experience tended to focus on getting the thing to work under interesting conditions but they did little intentional testing of error handling, and they were less concerned with the niceties of the dialog design. For example, they would test large, existing files but

they wouldn't test files that are no longer there. PC testers were more likely to try to open a non-existent file or a file on an empty (no floppy in the) drive.

These patterns of response are based on dozens of interviews, back in 1987 and 1988: I was the founding manager of the Creativity Division's testing group at Electronic Arts, and then was a software development manager at Power Up Software when it was just founding its testing group. The variation was initially puzzling because candidates who seemed equally strong in other ways gave such wildly different answers. Jack Falk and Hung Quoc Nguyen helped me recognize the patterns, and the extent to which they were predictable from the tester's platform.

I didn't try Myers' triangle puzzle with these candidates, but my bet is that testers who were more concerned with logical data relationships would have done better than testers who were more concerned with UI-driven products. And yet testers with either of these backgrounds might have been equally bright and equally effective with the application that I would ask them to test.

As I came to recognize the variation in responses, I changed how I used this dialog:

1. I presented the dialog, gave the candidate the marker and whatever time he needed, and encouraged him to give me his thoughts.
2. Then I complimented him on his analysis (even if he did badly, I tried to be encouraging) and I showed him some other types of tests that he had missed. I explained that no one got all of the types of tests and that some people missed some issues because they were nervous or they thought they had been rushed. I spent most of the time showing different types of tests and suggesting why they might be interesting (what kinds of bugs they could find).
3. Then I erased the whiteboard, drew a Save File dialog that was just as badly designed (a few of the UI design flaws from before still there, some fixed and some new ones) and asked the tester to try again.

The *real* test was the second test. For this one, everyone had just received an initial practice test and some coaching, so differential practice effects were much reduced. Everyone had received feedback, been reassured that they weren't dolts, but had been told that they'd missed some things. Most testers were substantially less nervous the second time through.

My real question was whether this tester was responsive to my style of training. Could the candidate pick up my explanations and do a substantially better job the next time? If yes, and if the second analysis was pretty good, then I had a reasonable candidate (as measured by this test). If the second analysis wasn't much better than the first, then this candidate was unlikely to be hired. This might be a really bright, well intentioned, interesting person, but if he doesn't learn when I teach, he needs a different teacher.

Occasionally, I have dispensed with the second test because the candidate did impossibly badly during the first test, or was extremely defensive or argumentative during my explanation of alternative tests. Both of these have been rare, but they happen. Usually, this means that I'm done with this candidate. I'll spend a little more time looking for a polite way to sew up the interview, but he won't be hired.

Another simple test

This suggestion came from Henry Klein.[9] It is another excellent example of the ease with which you can create a simple, useful performance test:

> I like to present a testing candidate with a simple sales promotion that we might run on our web site, and then ask what they would do to test it. I've gotten many different answers to this, from the person who told me all

[9] Personal communication, January 30, 2000 by e-mail.

about how to automate it but couldn't come up with a single test case, to people who gave me wonderful test cases to people who clearly didn't understand what I was looking for as they had only run tests written by others. I like to follow that question with an imaginary bug their imaginary tests revealed, which I describe in some detail, including an extraneous fact or two, and ask them what the bug report should say. The two questions flow well together most times, and give me a quick feel for how they approach a problem and what their skill level might be for these two important testing tasks.

Note that he has standardized on the same test, using it several times. This helps him compare the answers.

My only concern with this example is the same concern that I've faced with my own simple tests—what if the candidate just isn't fast on his feet? What if the candidate is nervous? What if the candidate is awkward at oral presentation because his first language is not English? To what extent might I undervalue this candidate based on these factors?

While I think that Klein's test is strong and revealing on its own, I would consider modifying it into a four stage test. The first two stages are exactly as described. The third is the interviewer's feedback to the tester (good job, good thinking, here are some areas you might have handled differently). The fourth stage is a second advertisement or second web document with some of the old twists and a few new ones.

Regarding the language issue, by the way, I find it useful to echo back what a candidate says or to restate it and ask if that's what they mean.[10] If we can communicate well the first time, the candidate is less nervous and more fluent the second time.

More complex performance tests

Sometimes you know exactly what you want the candidate to do, it's a specialized task, and you don't much care if he is weak in other areas. In this case, I want to try to find a way to measure the candidate against the task at hand.

For example (details changed to protect confidentiality), a colleague and I interviewed a candidate for a senior position that involved performance-related testing of a complex product. This candidate had experience modeling complex systems, was very smart, had a solid technical background, had been in the business for years, and was very good at oral presentation. Despite that, I had some reservations. Ultimately, my colleague and I agreed to pose the candidate a puzzle that would be representative of the type of work that she would do.

1. We explained the task first. We would demonstrate the product before lunch and answer any questions that she had. We would run any test that she requested. Then when she was satisfied, we would go to lunch and she would explain her approach to testing any aspect (her choice) of the performance of the system. The candidate said that she understood, and she agreed to do it.
2. We did demonstrate the system and, if she would have asked any questions, we would have answered them.
3. We did go to lunch and we did discuss the performance of the system.

---

[10] Why bother? Why not just disqualify someone whose English is poor? Because a Russian immigrant with a Ph.D. in Mathematics might work brilliantly in your lab if you can figure out how to manage the interim period during which his English is improving. This is another example of an opportunity hire. You might or might not be able to manage it, but don't close it off without thinking about it.

In the particular case, the candidate constantly challenged the design of the product while we demonstrated it. She would explain how this or that was probably slowing up the system. We reminded her that the task was to (a) observe and then (b) figure out how to test, and not yet (c) file bug reports. She persisted, we reminded her again, she persisted, and over the next hour she didn't learn a lot about the system. When we went to lunch, she lectured us on what was wrong with the system, but said that she'd need more information before she could tell us how to test it.

This was a bright candidate, and until this part of the interview, she had a significant chance of being hired. But this candidate would not have survived at this company and therefore we chose not to hire her.

In short, if you can find a way to present a piece of the job that the tester will actually do, you can see how well the tester does it. You have to make it a fair test, by designing it in such a way that someone who doesn't know your product can still do well at it. That's challenging. But if you come up with a fair test, the behavior that you see can be very informative.

THE INTERVIEW: DEBATES AND CONTROVERSIAL QUESTIONS (m xxx)

When I interview a test manager candidate, sometimes I engage him in a debate or I recruit some other authority figure to engage him in a debate. My expectation is that test managers will have to stand up and be persuasive under difficult circumstances, in the face of contrary pressure from authority figures. I want some indication of how well this person can handle this. It is, in my view, an essential part of the job.

My procedure is simple:

1. Go through the usual interview questions, focusing on the role of the testing group, the use of technology, the importance of specs and test plans, and so on. Encourage the candidate to tell me what he thinks are the most important factors for success, or the things that he has particularly strong opinions about.
2. Eventually, I'll have a sense of what this candidate thinks (a) is really important and (b) he has thought about carefully. There are typically a few possible areas to discuss. I'll choose the one that I can most effectively handle on the other side.
3. For example, suppose that this candidate loves black box GUI automation. I'll comment that in my experience, it has been a waste of time. On the other hand, if he says that it's a waste of time, inefficient, impossible to maintain, then I become a diehard fan of GUI regression tools. In either case, I engage the candidate with questions and politely disagree with his answers, asking additional questions, often in the form of "But don't you think that . . .?" or "But what about this . . .?".

Here's how I appraise the results:

- If the candidate fumbles and stumbles and turns out to not know what he's talking about, I reject him. I only debate on an issue that the candidate has explicitly identified as an area of special knowledge, and I only disagree when he has expressed a strong opinion. If he doesn't have his facts straight, he was feeding me baloney. I have zero tolerance for lies and exaggeration. He's gone. (Of course, I don't call him an exaggerator. I close down the debate, continue the interview with friendly questions, smile, thank him, don't make him an enemy, and veto him in private.)
- If the candidate backs down and adopts my point of view, I get concerned. After all, this is the test manager and this is an opinion that he claims to hold dear. I'm persuasive, but not usually *that* persuasive. If he backs down, it's probably because I am an authority figure in the context of the interview. So what's going to happen in the Real Job when he has a strong opinion, raises it with his boss (or his boss's boss), and encounters some resistance? Will he back down? That's not always the right thing for a *test manager* (or a *software quality assurance manager*) to do.

- If the candidate gets obnoxious (personal attacks on my judgment, calls me stupid or ignorant, raises his voice, treats me without respect) then I predict that he will be ineffective (and maybe quickly fired) when his wisdom is challenged by an executive. Next candidate, please.
- If the candidate listens to what I have to say, acknowledges my points politely, accepts the occasional correction, but sticks to his guns while maintaining his cool, I like him.

Some candidates walk away from this part of the interview feeling that they were unfairly confronted by someone who is closed minded. They might go away and decide not to accept the job if it's offered or they might agree to accept the job but hold a grudge against the interviewer. I'm still learning how to handle this. Probably the best way is directly, to explain at the end of the interview that my style of interviewing is to allow a debate to develop in order to see how the candidate handles it. I then congratulate the candidate, appreciate his attentiveness and his approach to the discussion, and then make my evaluation and decision in private.

Even if the candidate misbehaves, it is important to recognize that this is a high pressure, difficult situation for many people. There is no point attacking this person or insulting this person for responding poorly to pressure. I do my best to smile and show appreciation to the candidate for coming and for working so hard in the interview, even if I consider his performance during the interview completely unacceptable.

I've talked with other test managers who use the debating approach. We stick with it because it is informative, but it is uncomfortable. Some other managers skip it but probe deeply with behavioral questions, like these:

> Q. Tell me about a time that you disagreed with your manager and stood up to her. What was the disagreement about, how did you handle it, and how did it come out? (If the candidate describes a success, follow up with a question asking if he ever stood up, worked hard on an issue, but failed to persuade. How did that feel?) (If the candidate describes an initial failure, follow up with a question asking about success.)
>
> Q. Tell me about a time that a product was shipped over your protest. How did you convey your dissent? What kinds of arguments did you make? Who did you make them to? Why did they fail?
>
> Q. Tell me about a time when you wanted to fire someone but your manager disagreed (or someone senior wanted you to fire one of your staff and you refused).

THE INTERVIEW: FREE CONSULTING? (f xxx)

Some companies use an interview as an opportunity to get free consulting. A few companies have a reputation for this. To an outsider, it seems that when they have a technical/managerial problem, they issue some invitations to senior testers to interview with them. The interviewers discuss this current problem and ask for the candidate's opinions. The worst of these companies drag the "interview" out for several days and then cut off communications as soon as they've gotten the advice / opinion / information that they wanted.

Don't do this. It's unethical. It's probably a violation of the minimum wage laws. (After all, these people are doing work for you at this point.) And it's probably fraudulent, if you deliberately interview people with the intent of getting their advice instead of with the hope of hiring them.

There is value in posing realistic puzzles, and your staff will learn a lot about how other people think about testing by participating in interviews. But be conscious of the line between interviewing (giving someone information on which they can base a hiring decision) and consulting (giving someone analysis

and/or advice about a current problem). Your company can build a reputation for itself that it probably doesn't want.

The reputational problem doesn't just extend to the company. If you're a manager in a company that abuses its interviewees, or that seems to, you mind find yourself treated pretty shabbily in some interviews when you look for your next job. I haven't done this and I like to think that I wouldn't do it (because it might deprive the company of a good candidate), but I've seen it happen and I don't think that it's unethical or unfair.

POST-INTERVIEW MEETING (M xxx)

Suppose that Joe, Sandy, Jane and Ted interview the candidate, in that order. There is value in getting their input, and a lot of value in their sharing their impressions with each other. The collective view of the candidate carries more insights than the sum of the individual views.

Feedback while the interview is in progress

When Joe is finished, I'll ask for his impressions, but I will ask him not to share them with Sandy until we meet at the end of the day. Similarly for Sandy, Jane, and Ted.

Some groups are close-knit, they don't like this, and perhaps they could adopt a different rule. Joe can talk to Sandy *after* she has completed her interview. But in this situation I would still want Joe and Sandy to both give me, independently, before talking to each other, a tentative Yes, No, or Maybe.

If the candidate is clearly failing, then I will send the candidate home early. All of a sudden we will discover that we have a rush project that we have to take care of. We explain this to the candidate, tell him that we are very sorry, but we can't finish this today.

I don't have infinite time to spend on interviews. When a candidate is no longer in the running, I want to stop spending money and time interviewing him. Those of us who interviewed him might briefly meet to discuss it, but I won't spend much time on this.

I've identified the desirability of terminating the interview at several points, because I am conscious that this is an expensive process that can't afford preventable waste. But please don't get the wrong impression. In my experience, most interview candidates make it through the entire day without being sent home early.

The end-of-day discussion

The typical candidate has stayed through the day, made good impressions and bad, and now we have to appraise him.

I start by asking for a tentative vote. Do most people like this candidate or not? Yes / No / Not sure. Then we go around the room and trade impressions. Sometimes, this results in a clear, quick decision (No) and so we break quickly and get back to our other work.

My next step is to pull out the list of issues that we were interviewing the candidate against. (See the section on dividing the issues among the staff, above.) We'll work through the list one at a time. For example, suppose that we interviewed a management candidate and we get to the issue of mentoring. Suppose, too, that a pack of juniors was given the task of finding out over lunch how good a mentor this candidate would be, and how helpful this candidate would be in assigning other senior staff as mentors:

- First, the juniors will report on what they asked and what they learned.
- Next (especially if the first feedback came from people who are just learning how to interview), I ask for anyone else's observations. It's often the case that the same issues come up, perhaps as side issues, in several interviews. It's also all too often the case that a candidate will say different things to different people. For example, the candidate might tell the juniors that training is very important, that it will be a priority, and that lots of senior staff time will be spent on coaching.

The same candidate might tell the next interviewer (a senior tester) that juniors are a pain in the neck, and that they require too much hand-holding. The candidate might promise to reduce the senior tester's training burden by hiring more senior staff or by subjecting the juniors to sink-or-swim self-training. Contradictions like these happen. Sometimes they are rooted in a misunderstanding. Other times, they reflect a two-faced candidate.

We walk through the list and by the end realize that we like this candidate a lot and are ready to hire (or to move on to the next stage, perhaps scheduling a final interview with executive staff) or we like this candidate but need more information (which we list, if we can), or that we don't like this candidate.

Sometimes one or two people have reservations that no one else has. We all like the candidate except for these one or two people. This can be difficult for everyone.

- If the interviewer has a firm negative opinion, based on observation and reasonable interpretation of what was said, the candidate is vetoed. Goodbye, too bad, oh well. You might not choose to adopt a consensus model, but I commend it highly.
- The interviewer might have a negative opinion based on a lack of information or on a misunderstanding. This might be dealt with by the in-meeting discussion. It is important that the interviewer be allowed to stick to her guns, and know that she is allowed to stick to her guns. Becoming convinced that she should shift from a veto to an abstention should be the result of "being convinced" and should not be the result of "being intimidated" or "being pressured."
- The interviewer might feel that she would change her mind if she learned certain additional information or if the candidate answered certain additional questions the "right" way. If the candidate is returning for another interview day, she could do her own follow-up interview or she could ask one of the other interviewers to ask the appropriate questions. Alternatively, the dissenting interviewer might agree that the issue can be explored as part of the reference checking process, as long as the right questions are asked and answered during the reference checks. If the answers come out the wrong way, of course, the dissenting interviewer can and (unless she has changed her mind for good reason) probably should veto the candidate.

One piece to keep in mind and to make clear to the group. The meeting doesn't provide the final decision. If everyone agrees to accept the candidate, that is a tentative approval. I still have to check references, and I will probably not broadcast the details of those references to everyone else. I still have to go through the mechanics of developing an acceptable offer. The offer could be blocked for various reasons as we go forward. But the group has spoken, saying that it is OK with them if we hire this person, and that's important.

Another decision that the group might make is that this candidate is acceptable but that the interviewing process is not yet closed. If we have six people scheduled for interviews, we might interview all six before making an offer to the first. The benefit is that you gain perspective when you can compare candidates. The risk is that the first candidate might have a job by the time you get around to offering him a position.

FEEDBACK TO THE CANDIDATE (m xxx)

Throughout this chapter, I've suggested that I don't give the unsuccessful candidate much negative feedback, especially when I decide not to hire him. I have several reasons for this:

- Some people find negative feedback insulting, even (especially) when they've asked for it and promised not to be offended. It is too easy to make an enemy through the interview process, and I have no desire to do that.
- Some people become angry and threaten me. I don't like it when people scream at me, or threaten to beat me up. When I was 18, I worked as an assistant manager in a store. Some of my edges were rougher back then than they are now. One person came to the store waving a pistol. He was

going to shoot me for insulting his wife. Some people are a little crazy, and if I don't know them, I don't know that I want to criticize them and learn just how crazy they or their spouse might be.

- Most people that I've given negative feedback to in the past have argued with me. They've tried to convince me that I was wrong, that they deserve the job, that I should extend the interview process and collect some more data. This is not useful to them or to me. It is just difficult.
- Some people argue with me that the basis for my decision to not hire them was inappropriate and in some way discriminatory. Now they're going to complain up and down the corporate chain of command that I am an evil person and they will quote me (or misquote me) to all and sundry. People have threatened me in this way, but it has never gone very far. Still, it is a risk that I would rather consciously manage by minimizing the amount of information that a rejected candidate can misinterpret and misuse.
- If I provide this type of feedback to some candidates but choose not to provide it to others, I am treating people differently in a way that might be characterized as unfair or discriminatory. The people who are most likely to react really badly to criticism seem to be the people who will demand the most forcibly that you give them the feedback if you give anyone the feedback. Adopting and following a minimal-feedback policy for everyone makes it easier to deal with the most troublesome people.
- This is not a culture that gives this kind of feedback. I don't get it when I interview, unless I get it from a friend, or a person who becomes a friend (and then gives me feedback from an interview long, long ago). My friends don't get it. The books that I read don't recommend it. And I don't have a legal duty to provide it. And finally (check with your company's HR to be sure), there is probably a company policy or preference against it.

Instead, I will reject someone by appreciating the time that they spent coming to us (they deserve that, no matter how awful they were after they arrived). And I appreciate their thoughtfulness (they must have done *some* thinking during the interview). And if I can think of anything else that I can honestly praise, then I will. And then I express my regrets, but we found someone else who was a perfect fit. Or I express my regrets but we decided that it just wouldn't work out. Sorry, we can't discuss the reasoning, that's company policy, you know those bureaucrats, but it was really great meeting you. The goal is to cleanly terminate the process, without insulting or hurting the feelings of the candidate.

CHECKING REFERENCES (F xxx)

Always check references. Some of my worst hiring mistakes would have been avoided if I had only been more thorough about checking references.

Despite the fact that most companies have a policy against giving references that contain more than name, dates of employment, and other strictly superficial factual information, many managers will give you additional information if you build some rapport with them and ask polite, direct questions. (I find that I have more success when I can call them at their home rather than at work.)

To the best of my knowledge, and I am not an expert in this field, the risk to an employer of providing an honest but negative reference has been vastly overblown. Lawsuits over this are, as far as I can tell, rare. There are statutes in several states that make it extremely hard for a former employee to prevail in such a suit and the courts are, as far as I know, pretty unfriendly toward these suits in the other states. Rosse & Levin (1997, especially pages 143-53) have a lot to say about this. Risser (1993, pp. 165-168) is a readable book about the law that provides advice on this. Again, I am not saying that in my opinion as an attorney, Rosse & Levin and Risser are correct. I have not done the level of research necessary to form a lawyer's opinion on this matter as it would apply to your company in your state.

I can't provide an extended discussion of the reference call—it would take as long as the discussion so far of interviewing. But my basics are pretty straightforward:

- I ask for factual information, checking the candidate's claims. This includes asking for a job description. *After* I get the employer's job description, I ask about specific tasks that the candidate mentioned, and whether these were significant parts of the candidate's job. (Sometimes they were not part of the job at all. Hmmmm.)
- I ask for examples of good performance. What were some of the memorable events that made you happy to have been working with this candidate.
- I ask what training the candidate received. If the candidate claimed that she attended specific courses (or whatever), then I ask whether the manager remembers these. If not, well, sometimes people forget these things. In my experience, the candidate was not necessarily incorrect in the resume in cases like this. But it's a tiny red flag that might combine with some other red flags.
- I ask questions that came out of the interview. For example, if we had some questions about an automation project that they candidate did, I might ask, "She told me about a product in which she was the lead automated test developer. I think this was BugWare 2000. Do you remember his role on that project?" If the reference-giver says yes (he might well not remember enough details to answer the question fairly), then I ask for a description of the candidate's role and the success of the automation project. If I get pablum (bland reassurance), I might ask my "real" question, "Let me tell you my concern. I've heard about a lot of test automation projects like this that failed because the test code wasn't maintainable enough. We didn't ask detailed enough questions to form an opinion about what happened on this project. Can you give me some additional insight?"
- I ask whether the candidate appeared to get along well with the other staff. I ask for examples.
- I ask why this person left, or what she said was the reason.
- I ask whether the employer would hire this person again, and why.
- And I ask whether there were any weaknesses in the person's performance.

Beyond these general points, here are a few specific comments.

Asking for negative feedback

When you call for a reference, you can certainly ask whether some aspects of the employee's performance that would make the employer reluctant to hire this person again. You can also ask what aspects of the employee's performance needed improvement. But please, understand that some people will be cautious about answering questions that call for direct criticism of the employee.

- When you call someone for a reference, that person doesn't know you. She doesn't trust you. She has no idea how mature you are, how experienced you are, how likely you are to keep what she says in confidence, and how thoughtfully you will interpret what she says.
- Many hiring managers jump on *any* criticism as the end of the world. I've been astonished by how badly people (managers or HR staff calling for references) have responded to identification of even relatively mild problems. Some very positive references from people who have been very enthusiastic about a former employee have been interpreted as negative because of the answer to a tell-me-about-the-employee's-faults question.
- Even if the current employee is great, and the flaws are weak, many managers have policies against answering this question. Some handle it by providing a virtually meaningless answer, pre-determined Pablum, often saying the same thing about different people. (Maybe you've heard this one? *"Oh, sometimes deadlines were a challenge, but deadlines are a challenge around here for everybody. He worked very hard."*)
- Others simply refuse to answer the question. That's what I do, and what some of my legal clients do, on my advice. The problem is a complex one because if I give an answer to this question for

Joe and refuse to give an answer for Sandy (whose performance was terrible) and Sandy someday sues me and claims, among other things, that my refusal was taken as an unfair criticism of her, then I am in a much simpler defensive position if I can say, "I answer the question the same way for everybody. I tell people that I'm not allowed to answer it, as a matter of company policy." When I've given refusals, the caller sometimes gets very huffy about it. One person interpreted it as a serious negative criticism of a candidate who is, in fact, the single best individual contributor tester that I have ever worked with. (I said that, and also that I had actively recruited this person into two subsequent companies after managing him in a first company, and that I would hire him again, at top wages, any time I had the opportunity. I said lots of other great stuff about this candidate, to no avail.) Please, when someone refuses to answer a question, realize that they are simply refusing to answer a question that they have probably been told not to answer or not to answer meaningfully. You are not entitled to an answer. You might be entitled to a pattern of answers that does not taken as a whole, intentionally mislead you. But you are not entitled to an answer to a difficult question from a stranger who has no reason to trust or respect you.

Letters of reference

When someone gives you a letter of reference, it often means exactly what it says. But sometimes, it is a negotiated document that carefully expresses everything good that a firing or contract-not-renewing or encouraging-an-employee-to-quit manager can say without lying while carefully avoiding mentioning all of the horrible things that this manager would like to say. If you call that manager for an interview, ask first if he wrote the letter. If he says yes, ask some follow-up questions.

- He might answer them by continuing to paint a glowing picture of the candidate.
- He might answer them by continuing to paint a favorable picture of the candidate, but under your smooth questioning, he might reveal some other useful information.
- He might answer them by filling in the gaps, when you ask specific questions. And so you realize that this candidate was not so good an employee after all.
- He might refuse to answer on the ground that company policy forbids it. This is odd, because he did write the letter, so company policy doesn't forbid some level of reference-providing. The underlying problem might be that there is a termination contract that promises that the only thing that the company will say about the former employee is what is in the letter. I'll ask straight out whether this is the problem. Sometimes the manager will tell me (yes, or no).

If someone sends me reference letters with their resume, I feel free to check those references before the face-to-face interview. Sometimes, I'll call one before the phone screen.

If someone gives me a list of references, I feel free to call them after the first face-to-face interview. I am likely to call one after this interview (while making the decision to call back for another interview), and call the others later, when I am making the hire/don't hire decision.

I will also call some other people who are not on the candidate's list. These might be other people who I know, who worked with the candidate. Or it might just be a call to HR at that company, checking employment dates, job description, salary, and asking for any additional information they can give (which, as a matter of company policy, might be nothing).

INVESTIGATION (M XXX)

<<The question in this section is, what kind of background checks are appropriate beyond reference checking. Obviously, the answers are different for jobs involving testing a computer game versus testing the design of the next generation of nuclear missiles. Probably I will make that point and send the reader to some references.

One obvious point: If the company has not yet done a web search to see what the candidate posts online, now is the time to do it.>>

RED FLAGS (f xxx)

Rosse & Levin (1997) have an excellent discussion of red flag issues (things that come up in the interview, the resume, or the reference checks that should make you think twice about hiring this employee). I don't agree with everything they said—In particular, they raise a red flag when candidates lay out vacation or attendance needs, without, in my opinion, spending enough time on the notion of opportunity hiring. But overall, they provide a good discussion that you might find useful to consider.

Walley & Smith (1998) provide another useful red flag discussion. Deception (lies, exaggeration) is widespread in interview responses and resumes. They focus on the types of ways that candidates mislead potential employers and ways (not all of which I would feel comfortable recommending) to discover this.

MAKING AND CLOSING THE OFFER (M XXX)

<<I'm still thinking through the coverage of this. I don't plan to help readers figure out how much to offer or how to negotiate it. The question that I'm wondering about is whether there is any useful advice that I can offer on speeding the process through management, getting the paperwork in shape, etc. >>

REFERENCES

Bach, J. (1997) "Working with testers", *Software Development,* March.

Black, R. (1999) *Managing the Testing Process*, Microsoft Press.

Cook, M.F. (Ed.) (1992) *The AMA Handbook for Employee Recruitment and Retention*, American Management Association.

DeMarco, T. & Lister, T. (1987) *Peopleware: Productive Projects and Teams.* Dorset House.

Deming, W.E. (1982) *Out of the Crisis,* MIT.

Drucker, P.E. (1966) *The Effective Executive*, HarperCollins.

Dustin, E., Rashka, J. & Paul, J. (1999) *Automated Software Testing: Introduction, Management, and Performance*, Addison-Wesley.

Kaner, C., J. Falk, & H.Q. Nguyen (1993, 2nd Ed., republished 1999) *Testing Computer Software*, John Wiley & Sons.

Jensen, A.R. (1980) *Bias in Mental Testing,* The Free Press.

Kaner, C. & J. Bach (1999) *Paradigms of Software Testing*

Lane, C. A. (1997) *Naked in Cyberspace: How to Find Personal Information Online*, Pemberton Press.

Myers, G. J. (1979) *The Art of Software Testing* John Wiley & Sons.

Risser, R. (1993) *Stay Out of Court: The Manager's Guide to Preventing Employee Lawsuits.* Prentice-Hall.

Rosse, J. & R. Levin (1997) *High-Impact Hiring.* Josey-Bass.

Rothman, J.R. (1998) *Hiring Technical People: A Guide to Hiring the Right People for the Job*, Rothman Consulting Group, Inc.

Rothman, J.R. (2000), "The influential test manager", *Software Testing & Quality Engineering,* Vol. 2, #2.

Rothstein, M. (1996) *Ace the Technical Interview,* 2nd Ed., McGraw-Hill.

Walley, L. & Smith, M. (1998) *Deception in Selection*, John Wiley & Sons.

Weinberg *Quality Software Management*, Volume 1.

Whitaker, K. (1994) *Managing Software Maniacs*, John Wiley & Sons.

ADDITIONAL REFERENCES

I'm still reading these. They look valuable, and I suggest that you look at them, but I haven't quoted them above. I've noted some of the useful characteristics of these books, but understand that I'm still reading them. They have gems that I haven't yet noticed.

Beatty, R.H. (1994) *Interviewing and Selecting High Performers*. John Wiley & Sons. *Good examples of questions and questioning styles.*

Constantine, L.L. (1995) *Constantine on Peopleware*, Yourdon Press / Prentice Hall. *Interesting discussions of consensus-based engineering and staff characteristics.*

Humphrey, W.S. (1997) *Managing Technical People*, Addison-Wesley. *The discussion of talented people will be useful for enriching the consideration of desirable staff characteristics.*

Irish, R.K. (1987, Revised 3rd Ed.) *Go Hire Yourself an Employer*, Doubleday.