

*Nigel Mossman outlines the steps he takes when building an Agile project team. Getting the right team is one the critical factors in delivering a successful Agile project. The start of a new project means having to recruit a team, often from scratch and usually in a hurry.*

Identifying the team make up is the first step. Is the customer going to be on board from day one, or will a BA be required to act as the proxy customer? How many code engineers and what level of experience? It would be very unhealthy team that only comprised of Architects or graduates taken straight from university. Equally getting the right number of QA Engineers is key. Too smaller team leaves it vulnerable in the event of illness, too larger team will result in engineers tripping over each other when coding or QA Engineers (and the customer) in overload.

Write a job description (JD) for each role. Whether working through HR or a resourcing manager or direct with agencies or internet sites, they will need some idea of the skills and experience you are looking for. The JD for each role does not need to be overly long - perhaps like a ladies party dress – long enough to cover the subject but short enough to be interesting.

When writing the JD, you should give some thought to what possible applicants will have been doing in previous roles. This helps when reviewing CV's but also helps to identify the skills and capabilities that should be included in the JD. When writing the JD, check it against company standards and do remember all of the legislation that bars discrimination on grounds of age, gender, race, religion etc.

I normally write a short overview of the project (a paragraph) followed by a general description of the role – a sentence or two. This is then followed by a list of skills, some of which will be mandatory. Unless I have a good reason not to, I make domain make domain knowledge desirable rather than essential.

Next Step – Reviewing CV's. There are many books written on CV's and their construction. Given this, I never cease to be amazed by the variety of CV's I receive and the poor quality of many submissions. Many programmers CV's are not much more than lists of technologies and tools used on varioius projects. Structure is poor, spelling and grammar vary considerably. I have reviewed CV's from South Africa, Eastern Europe, Germany, Southern Europe, the USA and India – it's the same everywhere.

Based on CV presentation alone, you would reject a lot of very good candidates. You have to look beyond the poor presentation to try to pick out the right people. My general tips are look for people that can describe the project's they have worked on not just the

tools and technologies used. Look for people who've been working where you envisaged that they might when drafting the JD. I also look for gaps (lots of unexplained 3 month gaps can suggest illness, regular prison sentences or just plain disinterest in work). I also worry about CV's that are more than 4 pages long. As a general observation, I have found that CV's from Indian applicants can be very tend to be be verbose making it difficult to assess the suitability of the candidate.

Interviews. Even when I am selecting from a field of internal candidates, I always meet potential team members before inviting them to join the project. A telephone interview is a good way of establishing interest in a role, however you can only assess how a candidate interacts with people during a face to face interview. A video conference (VC) is a good alternative if you are recruiting off shore or candidates in difficult to access locations. I would be very reluctant to hire any body on the basis of a telephone call alone.

Structure and format are generally a matter of individual choice and (increasingly) have to comply with company standards. You should be aware that any notes you make can be requested by the candidate, especially if they feel you're refusal to hire was based on discrimination rather than objective criteria.

I generally try to involve no more than myself and one other person in the process. The other person will (ideally) be a practicing member of the discipline in question – i.e. A BA If interviewing a BA candidate. Depending on how comfortable and experienced the colleague is, we may interview together or separately. Interviewing together can be useful if you've had little time to prepare as while one is chatting to the candidate, the other can be framing up their questions.

Room preparation is also important. Always have a private space available for an interview; ensure it's clear of coffee cups and general office junk. I always try and avoid sitting directly across the table from a candidate (less confrontational). If interviewing with a colleague, do ensure that the candidate can look at both interviewers when speaking. It's nearly impossible to conduct a proper interview if the candiate has to turn his/her head through 180 degrees when replying to questions. These are important tips as they will help to sell the company to the candidate.

My personal technique is to outline the requirement for the role and describe project objectives at the start of the interview. This helps to relax the candiate; my real purpose is to see if, during the course of the interview, they relate their experiences in past roles to the requirements of the post.

When preparing for the interview process I work out a list of questions to put every candidate. I will then find a few extra questions tailored to individual candidates – for example checking gaps between engagements or to pick up on an interesting theme or project that the candidate has worked on. I prefer to ask competency based questions (e.g. “can you tell about a situation where you did designed a new architecture from scratch?”).

Years ago, when *C was it* and before the Internet, interviewing programmers in variably included a set list of tricky coding questions. Usually, you would ask the candidate to write a simple program or present them with several seemingly reasonable code fragments and ask them to identify the defects. The advent of Java with it’s many code packages and seemingly never ending list of frameworks means that this approach no longer works. I’ve found that nearly everybody passes the basic Java questions. Asking questions around specific code packages and frameworks is very hit and miss and has rarely served to differentiate candidates.

I have found it more productive to ask questions about how the candidate goes about coding, their attitude to quality, their engineering experience in general and what tool sets and frameworks they have used; which ones they liked and why. I also look for engineers that know how to go about solving a problem.

Using the same approach be applied to the other engineering disciplines. Asking BA’s about how they relate to customers, the way they gather and express requirements, their ability to work with code and test engineers and crucially their approach to organising stoies in the Agile context.

Although Agile is becoming more main stream, I still struggle to find an experienced field candidates. Candidates often claim to have Agile experience, in reality their projects have only used one or two Agile features (e.g. continuous integration). I generally look for candidates that show an interest in the subject and are able to talk with some enthusiasm on several aspects of Agile. Interest can be further highlighted by subscribing to forums, and having read Agile books. I’ve met many applicants that profess an interest in Agile, however during the interview it becomes clear that they read a page on a web site just before the interview.

I always conclude by allowing the candidate to ask any questions they have. This allows the candidate the chance to clarify anything they have not understood, or has not been addressed during the interview process. I’ve also found that the questions asked can provide more insite into the candidate and their interest in the role.

As I've been working with XP for the last few years, I often field questions about pairing. Some candidates state that they simply won't do it (many having never tried it!). My answer is that some amount of pairing is essential – the question is how much and the answer to that lies in discussion with the whole team. For example, I see little value in pairing on a maintenance project where the code base is widely understood by all members of the team, but would argue strongly for 100% pairing in the early days of a project creating new code.

Invariably, the final selection of candidates is based two factors; the factual evidence that shows that a candidate has the necessary skills and experience for the role and the more subjective observation about how well somebody will fit into a team. I can imagine HR managers being none to comfortable with my last comment, however, teams are about people and experience. Effective teams have to be able to interact with each other and possess the necessary technical skills.

It's not unusual for candidates to have to attend three or more interviews, attend assessment centers (which can last a day or more) and undertake various tests (including logic tests, psychometric tests, IQ tests etc). This takes up a lot of management time and, more importantly in a sellers market, will put many candidates off applying for roles. I know that I have declined interviews when I've heard about the amount of time and complexity involved in applying for a role (even for a contract). I believe that a fair and efficient recruitment process will be seen as a positive selling point in a competitive market. It will almost certainly give an edge over competing companies with long winded recruitment procedures.

Following basic good practice and keeping the selection process lean and fair will enable an Agile project manager to assemble a team that maximises the chances of a successful Agile project.

© Standard Data Systems Limited, June 2007

*-end-*