The Ideal Programmer

Why They Don't Exist and How to Manage Without Them?

Mike Williams

(A.k.a. Grumpy Old Man)

History

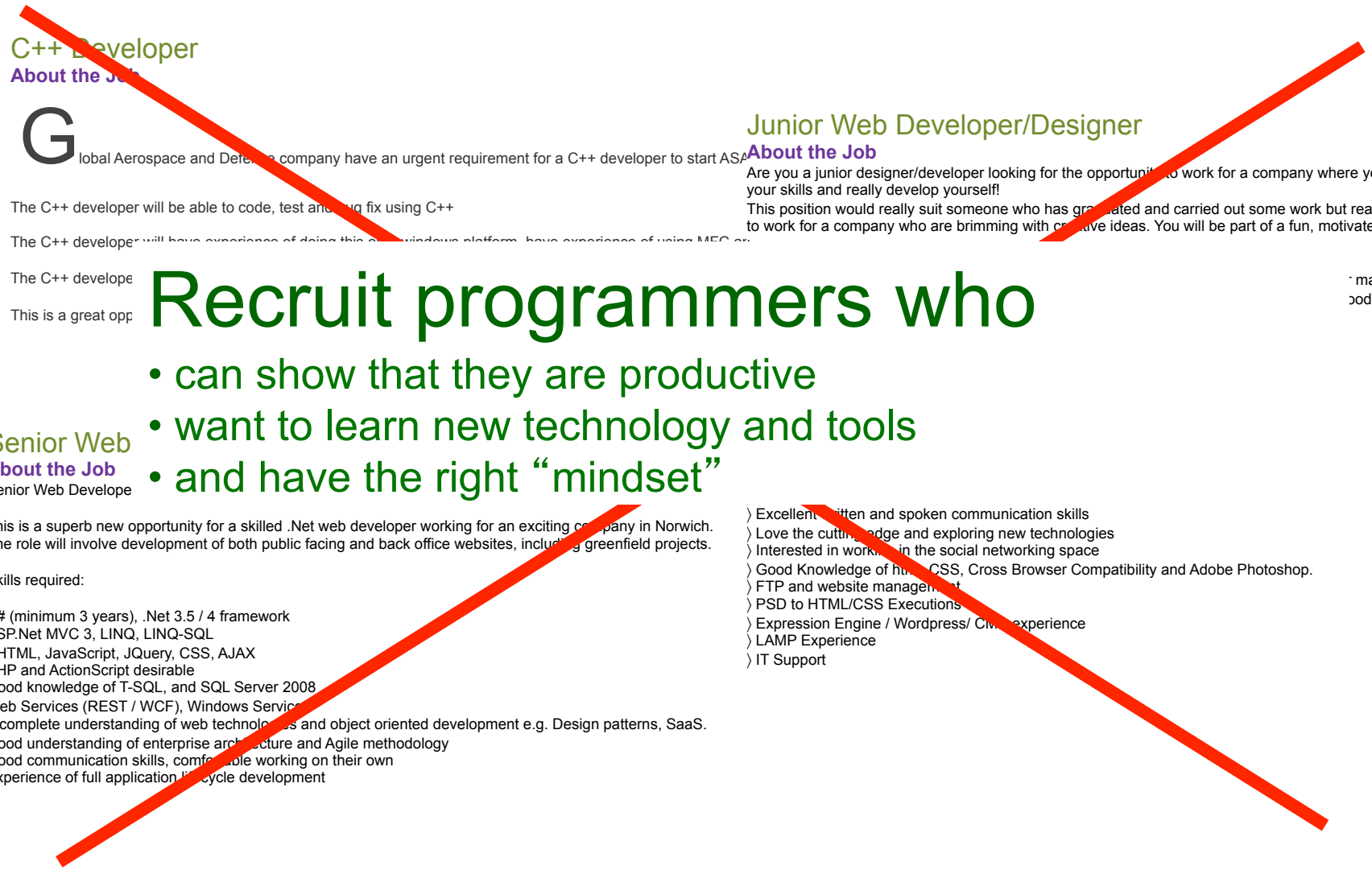
- Worked with software since 1967
  - First program in FORTRAN II on IBM1130
- Developed real time embedded telecoms software, mainly in assembly language and C
- Developed the first Erlang VM (JAM)
- Tried to “sell” Erlang for Ericsson
- Gave up trying to sell Erlang and have worked as a manager for about 20 years instead
  - Have managed both small units and large units (from 15 to 600 people), developing software

Ruminations

- Why are
  - Some programmers (Software Developers) so much better than others?
  - Some teams highly productive and self organizing and others require extensive management (processes)?
  - Some software projects very successful and some less successful?
  - Large software project very hard to run as a lot of small software projects?

I don’t have the answers,
but I have some thoughts.

Programmers
(a.k.a. Software Developers)



C++ Developer

About the Job

Global Aerospace and Defence company have an urgent requirement for a C++ developer to start ASAP

The C++ developer will be able to code, test and debug fix using C++

The C++ developer will have experience of doing this on windows platform, have experience of using MFC ar

The C++ develope

This is a great opp

Junior Web Developer/Designer

About the Job

Are you a junior designer/developer looking for the opportunity to work for a company where you can show case your skills and really develop yourself!

This position would really suit someone who has graduated and carried out some work but really want the chance to work for a company who are brimming with creative ideas. You will be part of a fun, motivated and developing

magazines based in

good career

Recruit programmers who

- can show that they are productive
- want to learn new technology and tools
- and have the right "mindset"

Senior Web

About the Job

Senior Web Develope

This is a superb new opportunity for a skilled .Net web developer working for an exciting company in Norwich. The role will involve development of both public facing and back office websites, including greenfield projects.

Skills required:

C# (minimum 3 years), .Net 3.5 / 4 framework
ASP.Net MVC 3, LINQ, LINQ-SQL
XHTML, JavaScript, JQuery, CSS, AJAX
PHP and ActionScript desirable
Good knowledge of T-SQL, and SQL Server 2008
Web Services (REST / WCF), Windows Service
A complete understanding of web technologies and object oriented development e.g. Design patterns, SaaS.
Good understanding of enterprise architecture and Agile methodology
Good communication skills, comfortable working on their own
Experience of full application lifecycle development

⟩ Excellent written and spoken communication skills
⟩ Love the cutting edge and exploring new technologies
⟩ Interested in working in the social networking space
⟩ Good Knowledge of HTML, CSS, Cross Browser Compatibility and Adobe Photoshop.
⟩ FTP and website management
⟩ PSD to HTML/CSS Executions
⟩ Expression Engine / Wordpress/ CMS experience
⟩ LAMP Experience
⟩ IT Support

The Ideal Programmer

- Technically competent and experienced
- Understands what technology is applicable
- Not afraid to try out new technology
- Understands the need for software architecture
- Has worked with both successful and unsuccessful project and understand why the successful projects worked and why the unsuccessful failed
- Documents his work with relevant and papers which will be useful to other people
- Refuses to do stupid things, write unnecessary documentation, use inappropriate software technology
- Understands that software needs to be maintained, probably by other people and prepares for it.
- Is both a team player and an individualist
- Can explain to other people what has been done
- Can understand vague requirements and turn them into strict well documented requirements and can check with the "customer" that this is what they want
- Can read long and semi-formal specifications and understand what is important and what can be left out
- Is able to help in recruiting new team members
- Teaches new team members and helps to integrate them into the team
- Is tolerant of the fact that individual team members have different skills
- Understands his/her own skills and weaknesses
- Regards criticism, pointing out of bugs etc as helpful and does not get annoyed
- Does not get "possessive" about code
- Understands that all vital parts of the software development process are equally important (requirements, systems works, programming, test, delivery, support)
- Accepts that parts of the code must be re-written
- Admits failure when things don't work out as expected and askes for advice when needed
- Realizes that architecture and code need to be elegant
- Understands when a system is good enough to deliver, and when it isn't
- Is prepared to work long hours when needed and is proud to deliver in time
- Is interested in customers' needs and wants to help meet them
- Enjoys programming

Too complicated, we will never find anybody who fits all these requirements

Intelligence or "The camel has two humps"

- A programmer must have the right sort of intelligence
  - http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf
  - *"Programming ability is not known to be correlated with age, with sex, or with educational attainment; nor has it been found to be correlated with any of the aptitudes measured in conventional 'intelligence' or 'problem-solving-ability' tests."*
  - *"despite the admonition of the computer science establishment to construct programs top down, experts build them bottom-up."*
  - *"the majority of good debuggers are good programmers, but not vice-versa"*
  - *"Programmers, who on the whole like to point and click, often expect that if you make programming point-and-click, then novices will find it easier. The entire field can be summarised as saying "no, they don't"."*

There are three types of people:

1. People who effortlessly learn to program and don't need any formal education
2. People who can be taught to program
3. People who haven't a clue what it is all about and never will be able to write a significant program

- The third group is by far the largest!

Reality

- The only safe way to determine a person's programming ability is by observing the person's programming performance in practice

A good programmer must enjoy programming!

- Wants to develop useful applications
  - Finds out what the customer needs and delivers it.
  - Sometimes customers don't know what they need
- Wants to continue being a programmer
- Hates interference by managers who don't understand software

A good programmer wants to learn new things

- Programming languages
- Tools
- Operating systems
- Applications
- Is prepared to make experiments
- Learns from mistakes

A good programmer understands the big picture

- Programming is a trial and error process
- Prepared to spend lots of time:
  - Testing
  - Writing **necessary** documentation
  - Working with, helping and teaching others
  - Maintaining and re-writing old code
  - Adding new features
- Tolerates the fact that capability and knowledge varies hugely between people and programmers

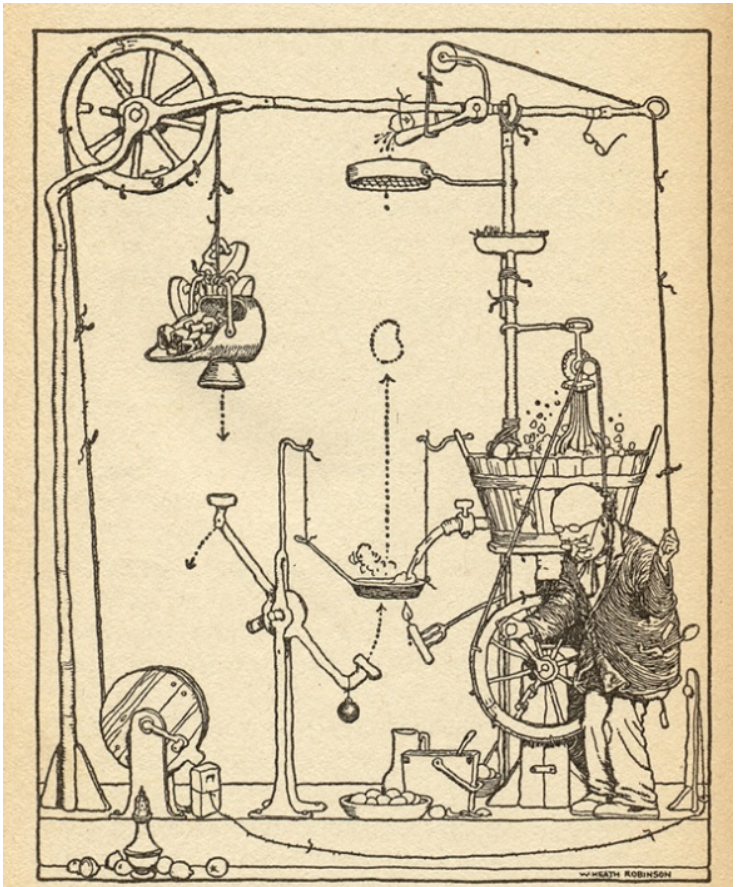
Teams & Projects – Essentials

- A software development team must have a clear and consistent vision of:
  - **What** are we going to develop?
  - **Why** are we doing developing it?
  - **How** are we going to develop it?

http://www.computer.org/csdl/mags/so/2012/03/mso2012030104-abs.html

What

- In some cases a detailed specification
- In some cases a standard
- Sometime a vague idea is enough, e.g.
  - A non SQL database
  - A chat server
- A vague idea of a multi-function product with lots of bells and whistles is a recipe for disaster!



The pancake-making machine

Why

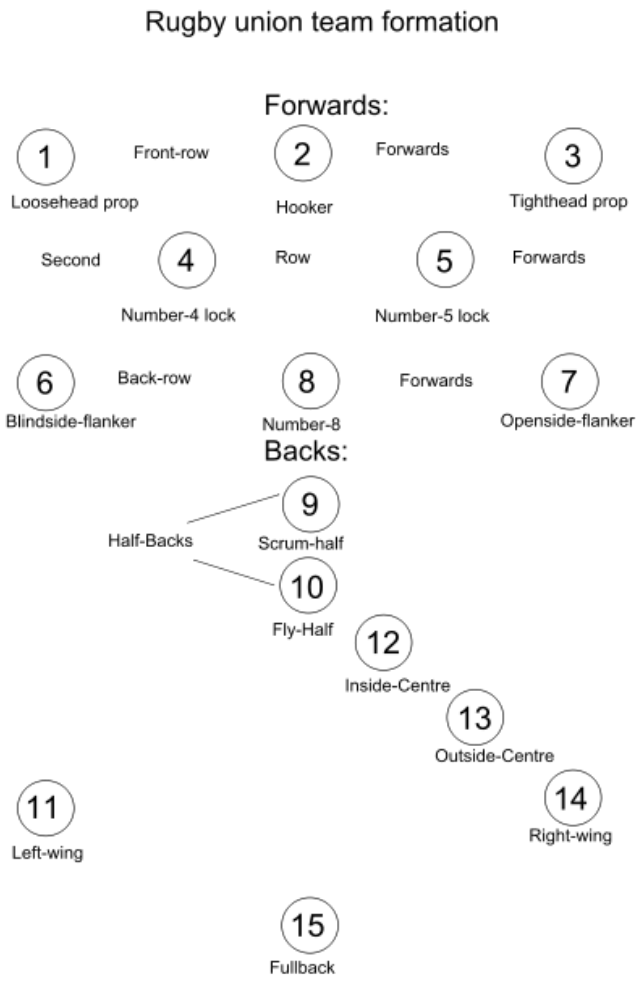
- To be motivated, you need to know why you are developing software
  - You need to see the big picture, where the (maybe tiny part) you are developing fits in and why it is useful
  - You need to believe that the final product serves a useful purpose

How

- Different parts of a system may use different software technology
  - There must be a strategy of how it all fits together
  - Parts with similar functionality need to use the same technology
  - There must a reason for the choice of technology for the various different parts
  - Technology need to be decided based on what the team feels right, not obscure company policy
  - People must enjoy working with the software technology
  - Elegance is not optional (Richard O'Keefe)

Teams

- Since we don't have ideal programmers, a software development team must have as many of the aspects of an idea programmer as possible
- Must avoid the traditional large company fragmented approach
  - Each team member must be able to take on new other roles when necessary



By what about Agile?

- Good programmers and effective teams have been working in an Agile manner for man years before
  - SCRUM
  - XP
  - Kanban

  was invented.
- Scrum masters, backlogs, sprints etc are OK, but let's face it: There is too much religion!
- There are plenty of examples of highly efficient and technically competent software development teams who have found there own "Agile" ways of working.

The Guru

- Each project needs a Guru
  - Most important: Detailed knowledge of the application being developed or maintained
  - Understands the overall architecture of the design
  - Can explain the details to other people
  - Is, at heart, a programmer but maybe doesn't program on a day to day basis
  - If he/she can't answer a question, knows who can answers and learns themselves.

The all-rounder and experimenter

- Understands the software technology being used.
- Can
  - jump in and help with any part of the product
  - work with any of the software technologies being used
- Does experiments with parts of the product to see to test improvements in:
  - Simplicity and elegance
  - Performance
  - User friendliness
  - Development efficiency

The organiser and enthusiasm bringer

- Has a plan for how and when the various parts need to be ready and how they be integrated.
- Is at heart a programmer and can understand when things are working and when they are not
  - Is able to start remedial action without humiliating anyone
- Maintains the “What, Why and How” spirit of the project
  - Human skills as important as software skills

The mechanic

- Keeps the development infrastructure working
  - Servers and backup
  - Version control
  - Programming
  - Requirements
  - Build
  - Test framework (both automated and manual)
  - Delivery
  - Mail, Wikies, other communication
  - Bug handling systems

The super tester

- Understands the application nearly as well as the Guru
- Can work out all the peculiar things which can happen to make the system go wrong
- Is able to explain in details why the faults he/she find have occurred and how to reproduce them
  - Makes other happy that he/she has found a fault
- Can often suggest remedies or “quick fixes” to keep the project on track.
- Checks test coverage and suggests way to improve coverage.

The librarian, integrator and maintainer

- Sometimes the same as the “mechanic”
- Maintains the over-all version control strategy and checks that things are checked in correctly
- Maintains the test, build and delivery system so that anybody can produce a complete “delivery system” in a few minutes if and when needed

The documenter

- Understands the application nearly as well as the Guru
- Understands what needs to be documented **and what doesn't**
- Understands the needs and knowledge of those who read the documentation
- Is able to write clearly, grammatically, concisely and pedagogically and is able to help other people to do so
- Is a programmer at heart

The madmen (aka programmers)

- All good programmers are mad
  - You have to be mad to be prepared to work in a trial and error fashion
- A good programmer and “turn his/her hand” to any aspect of software design.
- A good programmer is an individualist who is capable of working with other people

A question you need to ask

- Is there a process in place (either formally documented, or well known which describes how:
  - requirements are found
  - requirements are broken down into systems architecture
  - parts of the architecture are programmed
  - parts are tested
  - system is built
  - system is tested
  - system is delivered
  - system is maintained
  - version control is done
  - etc
- If you don't have this, you are in trouble.
- If you believe you can slavishly follow such a process, you are in deeper trouble.
  - See A Rational Design process: How and Why to Fake it. David L. Parnas http://web.cs.wpi.edu/~gpollice/cs3733-b05/Readings/FAKE-IT.pdf