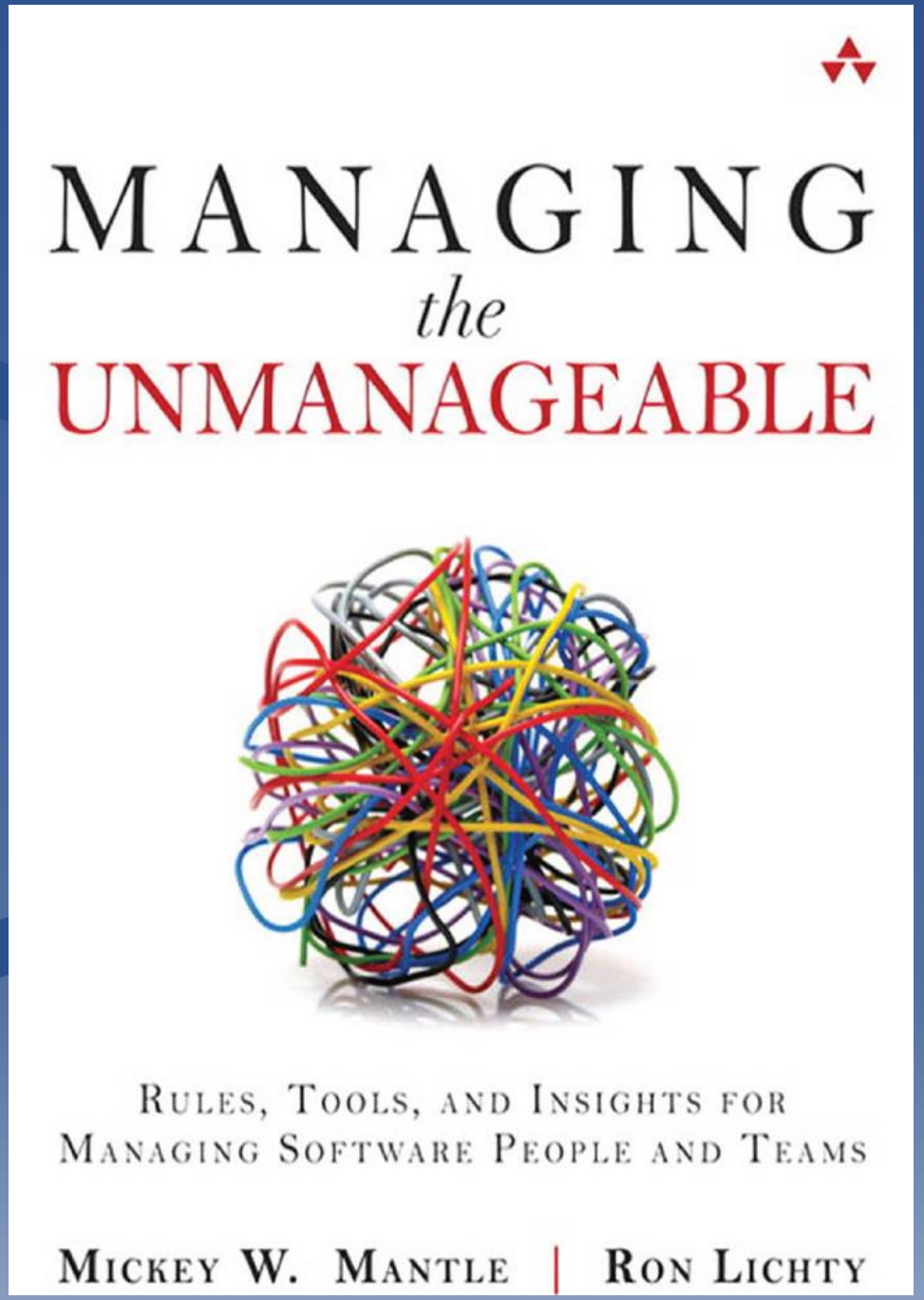
Managing the Unmanageable

Managing Software Development Teams

D. Vanderbist 15/04/2019



Software Development is difficult (1/1)

Why is it difficult?

- Software engineering is often more a craft than a science
- Anyone can become a programmer
- The effect of formalized frameworks (SDLC) on the results has been minimal it depends more on the collaboration of people

Why is it fun?

- Creating new things
- Things that are useful to others
- Fascinating complex puzzles
- Learning and doing non-repetitive tasks
- Tactile medium

Different type of people!

- Left-brained - Right-brained people:
  - Left: easy to manage
    engineering types
    engineers: formal education
  - Right: hard to manage
    art and more liberal types
    craftsman: training by doing

- Eagleson's Law applies: **"Any code of your own that you haven't looked at for six or more months might as well have been written by someone else."**

- Finding good software managers is hard because typically incentives do not work with free-spirited programmers: **work environment is more important than compensation** as motivator.

Understanding Software Developers (1/7)

Programming disciplines:

- Client programmers
- Server programmers
- Database programmers
- Web developers and other scripters

Client programmers:
Client programmers is that they tend to have assignments where they have near complete control over their resources.

Server programmers:
Server programs reside on machines that are generally remote from the end user, and most of these programs must be written to handle multiple activities from a number of clients at the same time. This creates a level of complexity that is often not part of the programs that client programmers must deliver. Server programs often must be written and deployed such that additional machines and resources can be added without changing the fundamental architecture of the program, which adds even more complexity to developing server programs.

Database programmers:
Database programmers are a different breed of cat from client or server programmers. Database programmers typically “speak” a completely different programming language, use completely different tools, and write programs that deliver distinctly different results from client or server programmers. Although DB became all SQL still need specialized DB knowledge to get job done i.e. expertise

Web developers:
Web developers, on the other hand, mostly use formatting markup (e.g., HTML, XML, CSS, ASP/JSP) and scripting tools (e.g., Perl, PHP, JavaScript) to accomplish their tasks. Some of their development is “cut, paste, and modify” (copying some code that works, sometimes modifying it to perform a different task).

Understanding Software Developers (2/7)

Types of programmers:

- System engineers/architects
- System programmers
- Application programmers
- Not really programmers

System Engineers/Architects:
System engineers/architects are the most highly skilled and experienced programmers on any development staff. The ability to understand the complex relationships among all the relevant system components (operating systems, communication systems, databases, online/offline access, security, hardware, etc.) requires extensive expertise in all the various technologies and systems.

System Programmers:
Most system engineers/architects started out as system programmers. System programmers understand how all the components in a system work, including the operating system(s) and communication system(s) on the client and/or server.

Application Programmers
Application programmers make up the vast majority of professional programmers, students, and hobbyists who call themselves programmers. Application programmers typically develop programs that are used directly by end users or whose results are used directly by end users.

Not Really Programmers
There are members of the technical staff on development teams who are referred to as “programmers” who are not really programmers. Some specify program or business logic using graphical user interface (GUI) tools that then generate user-accessible applications. Others create scripts or modify configuration files that result in customized content display. The chief distinction in the role these “programmers” serve is that they are using preprogrammed tools and applications and not directly crafting code.

Understanding Software Developers (3/7)

Domain Expertise:

- **In good economies**, organizations seek out technical people (as well as managers) with broad domain expertise, hoping that they will contribute out-of-the-box thinking and cross-pollinate ideas and approaches and best practices that have come into use in other domains.
- **In bad economies**, on the other hand, organizations are hunkered down, cutting costs, driving efficiency, limiting development to core domain functionality—and they tend to look for years and years of domain-specific expertise to reduce the risk of the few new hires they're able to make.

Programmer proximity and relationships:

- In-house employees
- Geographically distant employees
- Contractors
- Contracted managed teams
- Outsourcing companies

In-house employees expect a career, not just a job. So, you must also worry about providing:

- Professional and career growth, and opportunities that will enable that growth
- Regular feedback
- Communication about company news and events
- An almost unlimited number of other expectations that you may have to discover

Understanding Software Developers (4/7)

Job Levels and Job descriptions:

Client Programmers	System Programmers	Database Programmers
Programmer 3	System Programmer 3	Database Developer 3
Programmer 2	System Programmer 2	Database Developer 2
Programmer 1	System Programmer 1	Database Developer 1
Senior Programmer 2	Senior System Programmer 2	Senior Database Developer 2
Senior Programmer 1/ Architect	Senior System Programmer 1/ Architect	Senior Database Developer 1/ Architect

Programmer 3 (Entry Level)

- Knowledge of Windows, Mac, or Linux
- Basic knowledge of good coding practices
- Aware of/interested in Internet technologies
- Aware of/interested in database technologies
- Knowledge of C/C++
- Ability to work in a team and take direction
- Can work with supervisor to plan tasks

Programmer 2 (Some Experience)

- Produced one or more commercial apps
- Proficiency on Windows, Mac, or Linux
- Experience with good coding practices
- Conversant with Internet technologies
- Conversant with database technologies
- Solid understanding of C/C++
- Self-motivated and can take direction
- Can independently plan tasks

Programmer 1 (Experienced)

- Produced two or more commercial apps
- Proficiency on two platforms
- Familiar with Internet technologies
- Familiar with database technologies
- Well versed in C/C++
- Good communication skills
- Self-motivated, minimal direction
- Good project-planning and schedule-estimating skills
- Recognizes problems and helps group adapt

Senior Programmer 2

- Produced two or more commercial apps
- Proficiency on two platforms
- Understands cross-platform issues
- Knowledgeable about Internet technologies
- Knowledgeable about database technologies
- In-depth knowledge of C/C++
- Strong communication skills
- Self-motivated, minimal direction
- Excellent analysis, project-planning, and schedule-estimating skills
- Watches for changing conditions and plans adaptations

Senior Programmer 1

- Produced two or more complex commercial apps or technologies
- Thorough knowledge of two platforms
- Understands cross-platform issues
- Excellent knowledge of Internet technologies
- Knowledgeable about database technologies
- Expert knowledge of C/C++
- Expert in software design practices
- Strong communication skills, industry relationships
- Self-motivated, works independently
- Excellent analysis, project-planning, and schedule-estimating skills
- Generates, enhances, and promotes new ideas
- Watches for changing conditions and plans adaptations

Understanding Software Developers (5/7)

Communication is key:

Communication is a factor of the inverse square of the distance domestically, and the inverse cube of the distance internationally.

—David C. Evans

We have seen too many

- Communication breakdowns just because someone is not sitting nearby!
- The farther apart people are the more difficult the problem.
    - down the hall,
    - in the next building,
    - in the next time zone,
    - halfway around the world),
- International difference:
    - extreme time zone differences,
    - language and accent challenges,
    - cultural barriers.

Contractors:

- By definition, “hired guns” who are called upon to do the task(s) you ask them to do in return for compensation.
- Contractors should have no implicit expectations. Any expectations must be spelled out in a contract you execute with the contractor before services begin.
- **Too often, you are looking to hire a full-time employee and either cannot find someone suitable to hire or are not sure you have found the right person to hire**. So you take on a contractor but begin treating the contractor like an employee. **This leads to implicit expectations that are akin to those of a fulltime employee.**

Understanding Software Developers (6/7)

Generational differences:

- **Baby Boomers** tend to be optimistic and loyal, and they long to stabilize their careers. Many Boomers are workaholics, willing to do more than is required to help their company and advance their careers.

- Members of **Generation X** tend to be cynical and have a self-reliance that has led them to embrace "free agency" over company loyalty. They view time as a commodity they are unwilling to share or give away.

- **Millennials** tend to be individualistic, yet with a group orientation. Ambitious and confident, they have a collective sense of optimism, tenacity, and heroic spirit, traits reinforced by the national unity following the 9/11 tragedy. Like the Gen Xers, the Millennials view time as a commodity they are unwilling to share or give away—they see work as something to do between the weekends.

Generation*	Year Born	Music	Mass Media	Technologies†	Characteristics†	Core Values§
Older Boomers	1945–55	Vinyl LPs	AM radio, broad-cast TV, news-papers	Analog (e.g., electric guitars), telephone, U.S. Mail	Willing to use technol-ogy, but often only to communicate with family and friends.	Rebel against conformity and strive for a perfectionist lifestyle based on personal values and spiritual growth.
Younger Boomers	1956–65	Cas-sette tapes	FM radio, cable TV, newspapers	PCs, FAX, e-mail	Comfortable with Internet, social media, and mobile; they em-brace technology but seldom fanatically.	Welcome team-based work and have had stable careers marked by loyalty to companies.
Gen X	1966–85	CDs	Cable TV, Web sites	Internet, e-mail, text messages	Love technology that helps them be inde-pendent, and digital stuff that improves their lives.	Economic and psychologi-cal "survivor" mentality; they tend to be skeptical of authority and cautious in their commitments. Ambitious and independent, they're now striving to balance the competing demands of work, family, and personal life.
Millennials	1986–2005	iPod/ Pandora	Web sites, Face-book, Twitter	Mobile, text mes-sages, Facebook, Twitter	Mobile is their defin-ing characteristic: texting, making party plans on the fly while out, carrying their identity around in their phones.	Coming of age during a shift toward virtue and values, they're attracted to orga-nizations whose missions speak to a purpose greater than a bottom line. They're technologically savvy with a positive, can-do attitude that says, "I'm here to make a difference."

Understanding Software Developers (7/7)

Personality Differences:

MBTI framework

Left-Brain versus Right-Brain People

- If you are a programmer or technical person, you've probably been labeled as being "left-brained," which is said to be more verbal, logical, analytical, and objective and is more appropriately called "left-brain dominant,"

Night versus Morning People

- What we do recommend is that you establish a set of "core hours" that you expect everyone to observe to ensure a minimum, reasonable level of communication across your entire team.

Cowboys versus Farmers

- Free souls vs methodological work

Heroes

- The challenge for you in managing heroes is that you can easily burn them out by expecting superhuman efforts all the time. Occasionally, yes, but all the time, no.

Introverts

- By drawing them out in meetings, and giving them positive reinforcement when sharing comments or insights, you can begin to build their confidence that they have something to contribute to the team.

Cynics

- Avoid at all cost hiring individuals who are deeply cynical. They can poison a development team and create havoc in your organization by sowing dissension and dissatisfaction where those feelings might otherwise never grow.

Jerks

- They may be brilliant, technically talented, great programmers. But brilliance doesn't justify the price you have to pay for having them in your organization. Having them in proximity is the problem.

Finding and Hiring Great Programmers! (1/8)

Good vs Great

- 20:1 differences in initial coding time
- 5:1 differences in code size
- 25:1 differences in debugging time
- 5.3 times more productive than teams of inexperienced bottom-tier programmers

A's hire A's. B's hire C's.
—Steve Jobs

When it comes to getting things done, we need fewer architects and more bricklayers.
— Colleen C. Barrett, President and Corporate Secretary of Southwest Airlines

Job Description:

- What is in it for you
- Company description
- Required skills
- Bonus skills
- Context
- Contact information

Finding and Hiring Great Programmers! (2/8)

PRINCIPAL PROGRAMMER, .NET

San Francisco/Oakland/ Berkeley — Specific location

(Note: significant telecommuting opportunity if desired) — Telecommuting?

Competitive salary, benefits, and options — Not equity only

Forensic Logic, Inc. (www.forensiclogic.com), is an early-stage, growth-oriented company looking for a highly productive senior developer with the ability to lead a team and set its technical direction based on tons of experience designing, coding, and scaling .Net and SQL Server high-volume Web and analytics applications.

Forensic Logic develops Web-based applications that provide law enforcement agencies with tools that facilitate increased officer safety, early detection of crime trends, and interagency search capabilities. The successful candidate will have a unique opportunity to work with massive data sets, both structured and unstructured, and extensive association, geospatial, timeline, and pattern analysis and visualization, and application of matching and ranking algorithms for solving crimes. The position will provide a growth opportunity to the right individual who will be part of a great team of talented and motivated coworkers.

Forensic Logic's culture values respect, teamwork, and collaboration in achieving leading-edge functionality balanced with high usability.

The product, company, and opportunity: sales!

Company culture: more sales!

JOB DESCRIPTION

- Provide technical team leadership, direction, and mentoring to the small, existing remote programming team.
- Form the nucleus of a second Bay Area development team.
- Apply incisive design and exceptional coding skill to knocking features off the products' extensive and growing features list.
- Lead periodic rapid refactorings that keep the application code fresh, flexible, and reusable.
- Help define team development and engineering best practices.
- Lead the team's implementation of best practices.

Specifically, what the person you hire will do . . .

REQUIRED SKILLS

- Strong Web application architecture and design skills
- Fast, clean, efficient code implementation
- In-depth knowledge of Microsoft .Net and SQL Server
- 3+ years' experience designing, developing, and scaling high-volume Web applications on .Net and SQL Server platforms
- Leadership and mentorship of other developers, junior and senior alike
- Team orientation; ability to participate in lively engineering debate, making a strong case for well-considered opinions, while listening to, appreciating, and critiquing the opinions of peers
- Ability to analyze and improve the scalability and performance of high-volume, information-rich Web applications
- Strong verbal and written communication skills
- Strong customer empathy and customer experience sensitivity
- Must be highly self-motivated, ambitious, flexible, self-sufficient, and high-energy
- 8+ years' programming experience

Specifically, the skills you require

EXPERIENCE WITH ANY OF THE FOLLOWING A PLUS:

- Algorithmic design and implementation; reasoning through algorithmic trade-offs
- Search/information retrieval
- Analytics, data warehousing, and business intelligence
- Information visualization
- Web services

The skills you consider a bonus

Some (but not extensive) travel will be required. — Travel?

Located adjacent to BART in the heart of the San Francisco Bay Area. — Location

www.ForensicLogic.com

Send résumés to:

Ron Lichty
VP, Products and Engineering
Forensic Logic, Inc.
RLichty@ForensicLogic.com

No phone calls

Principals only

Contact information

Finding and Hiring Great Programmers! (3/8)

Always be recruiting!

- If there are no positions open.
- If you find the superstar, create a business care for increasing the headcount

Recruitment budgeting:

- Paying commissions to headhunters
- Engaging an internal recruiter or retaining an external one for this or a group of hires
- Paying employees bonuses for making successful referrals
- Paying for advertising in daily and Sunday newspapers, in trade and technical journals, and online
- Organizing a special recruiting event, perhaps around the time and location of a conference focused on a key technology in which you need expertise
- Paying to fly in remote candidates to interview (and potentially paying for moving expenses to relocate them, should you decide to go that route) is usually a percentage of the new hire's first-year salary. The percentage was once 15 percent, but these days it is seldom less than 20, and 25 percent is not uncommon.
- Employee Referrals for a small bonus.

Finding and Hiring Great Programmers! (4/8)

CV's Screening

- Highlight the skills and tools you're looking for, where they appear.
- Draw arrows to gaps in employment history, so you can follow up with a question.
- Circle spelling errors, bad grammar, and sloppy formatting; you may end up making a decision between two candidates based on knowing that one can write well enough that you won't have to review every word.
- Note where candidates have changed jobs frequently; if you're looking for someone to stay on your team long-erm, you may need to formulate a question that elicits why a candidate jumped around.
- Hands-on experience using live coding across a conference call.

College degrees don't impress me, and lack of school doesn't scare me (see: Jobs, Steve, and Gates, Bill). At some point, when a person is far enough removed from school, the degree is all but meaningless. Experience is what matters most.- Eric Mull

Interviews:

- If the candidate is still interested
- Whether the candidate is interviewing with other companies (and what the time frame is with those companies, whether the candidate already has other offers, is considering them seriously, and when a decision must be made)
- What kind of job the candidate is looking for
- What the candidate considers to be their areas of expertise
- What compensation is expected
- Why the candidate is looking for another job
- The candidate's availability to start working for you
- Whether the candidate is willing to commute to your location if working in your offices is a requirement

What we learned was that asking candidates to write code and to answer questions about code is absolutely critical. —Steve Johnson

Finding and Hiring Great Programmers! (5/8)

Principal Programmer Interview Summary	Bill Smith	Cathy Llu	Arnold Lai	Lucy Miller	Andy Jones
Communicates designs effectively					
Listens					
Critiques others' designs					
Writing skills					
Customer Experience empathy/ awareness/design sense					
Intangible qualities					
Energy					
Flexibility					
Self-direction					
Smart					
Articulate					
Passionate					
Fit in with team					
Overall desire to work at our company					
Experience w/algorithmic design, coding, trade-offs					
Search/information retrieval					
Analytics, data warehousing, and business intelligence					
Information visualization					
Web services					
Sent us a follow-up thank you?					

Principal Programmer Interview Summary	Bill Smith	Cathy Llu	Arnold Lai	Lucy Miller	Andy Jones
Received résumé on					
Phone screen on					
First interview round on					
On time, early, or late?					
Second interview round on					
On time, early, or late?					
Bachelor's Degree (optional)					
Minimum 8 years programming experience					
Wrote first program ever in (year, language)					
Wrote first professional program in					
Experience with what languages					
Experience with what databases					
Minimum 3 years .Net programming experience					
Wrote first .Net program in					
Most recently wrote for .Net in					
Minimum 3 years SQL Server programming experience					
Wrote first SQL Server program in					
Most recently wrote for SQL Server v. (???) in (year)					
Web application architecture and design skills?					
Ability to analyze & improve scalability and performance					
Experience scaling high-volume, information-rich Web apps					
Fast, clean, efficient coder?					
Refactoring skills					
Has defined development and engineering best practices					
Experience leading and mentoring other developers					

Finding and Hiring Great Programmers! (6/8)

Interviewing:

- Make sure there is a whiteboard in the room. The eagerness to explain differentiates the talkers from the do-ers.

Interview questions:

- “What aspects of your last job did you most like?”
- “What were your colleagues and your management like?”
- “ Tell me about some of the things you and your supervisor disagreed about.”
- “What led you to leave the companies you previously worked for?”
- “What attracts you to our company?”
- “Why are you looking for another job now?”
- “What do you want to get out of your job?”

Learn to ask questions that are open-ended—that candidates can’t answer with a yes or no—questions like these:

- “Tell me about . . .”
- “How were you able to accomplish . . . ?”
- “What was your role in . . . ?”
- “ If you had led the development effort on that project, what would you have done differently?”
- “What best practices are you most fond of?”
- “What are your strongest technical strengths?”
- “What are your strongest nontechnical strengths?”
- “ If you think of the fabric of programming as triangular, with the points representing design, coding, and debugging, tell me about the part of the fabric on which you would most like to spend your time.”
- “ Where would you place yourself on a continuum where one end is developing gnarly algorithms and the other is developing customer focused UI?”
- “ Imagine a line. One end is leadership. On the other is teamwork that’s so fully collaborative that leadership is totally shared and no one on the team would be able to identify a leader. Where on that line would you place yourself?”
- “How would your manager describe you?”
- “ Tell me about your comfort level with asking for assistance from others.”
- “ Where do you fall on a continuum that ranges from highly structured, where your tasks are spelled out completely, to one that is entirely free-form and you have to make decisions, often without having all the information you’d like?”
- “How do you like to be managed?”

Finding and Hiring Great Programmers! (7/8)

Ask for examples:

- " Think of a time when you knew you could not make a deadline. What did you do?"
- " What was the most interesting problem you faced in a former project? How did you solve it?"
- "Tell me about a time when you . . ."
- "Give me an example that illustrates your leadership style."
- " Think for a minute about the most stressful situations you've been in at work and tell me about the one you think was most stressful of all.
- What did you do to deal with it?"
- " Have there been times when you needed to formulate a new solution?
- Tell me about that time, and about what you devised."
- " Tell me about a best practice you played a role in getting your team to adopt."
- "Describe a time when you displayed extraordinary initiative."
- " Have you worked with a UI designer [product manager, business analyst . . .] to translate customer needs into technical requirements?
- Tell me about that collaboration."
- " Tell me about a time when your manager was annoyed with you or with your role on the team. How did you respond?"
- " Think about the teams you've been part of and tell me about a peak teamwork experience. What contributed to making that memorable?"
- " What have you done when you've had far too many tasks assigned to you than you can handle? When that's been the situation and you could see yet another task coming your way, what did you do?"
- " Tell me about a time when you successfully persuaded your manager or your team to adopt your position."
- " How have you handled making formal presentations in front of large and small groups? Tell me what that was like."
- " Have you had to present technical solutions to highly nontechnical audiences? How did that go?"
- " Describe a time when you advocated creating a better customer experience."
- " Describe a situation in which you had to tear down code and redesign and recode from the ground up."

Finding and Hiring Great Programmers! (8/8)

There are also key logistical questions you'll want to ask:

- "What will commuting to our offices from where you live be like for you?"
- "How much travel do you like (and how does that fit with the amount of travel I foresee in the job)?"
- "What are your compensation expectations?"
- "What was your base salary in your last job?"

Illegal to ask:

- Marital status
- Parental status
- Age (particularly whether 40 or over, but don't go there with anyone)
- Ethnicity or nationality
- Disability or perceived disability
- Religion
- Sexual orientation

Deciding:

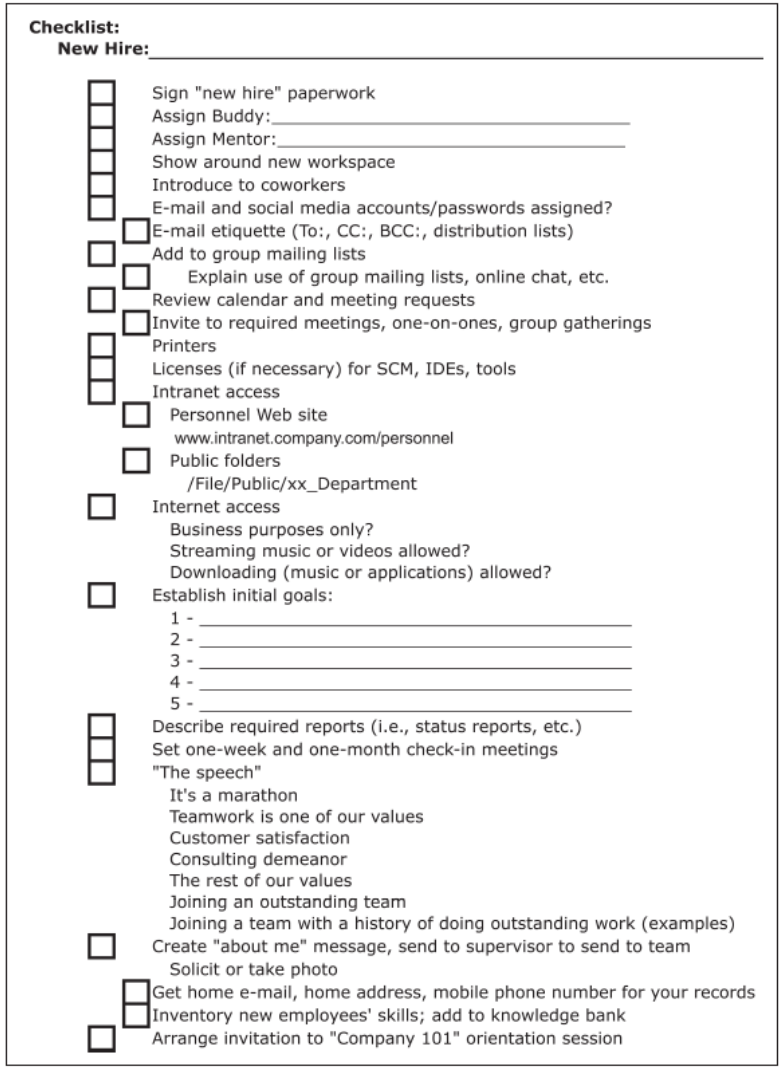
- If in doubt say no:
  If you're dithering, don't hire them. — Steve Burbeck,
  "Hire people you want to sit next to, both tomorrow and a year from now."
- Call references: ask the candidate to provide you with a list of references you can call.
- Know beforehand how long you're willing to give them: many candidates will ask for a few days or even a week or more to consider your offer. It's seldom the answer you want to hear, but it is reasonable.
- **Follow Up Until the Programmer Accepts:** sending candidates a signed offer letter in a FedEx package for them to sign and return speaks volumes to how important the candidates are to you and getting the offer in their hands is to them.

Getting New Programmers Started Off Right (1/3)

Get Them on Board Early

- Place a congratulations call.
  - Welcome them to the team.
  - Assign other team members to do the same.
- Preparing for Their Arrival
  - Know the required lead-time needed by core-services.
  - Prepare material and prepare access (physical/virtual)
- Launch buddy program
- Assign career mentor

First day:



Checklist:
New Hire:_____

- ☐ Sign "new hire" paperwork
- ☐ Assign Buddy:_____
- ☐ Assign Mentor:_____
- ☐ Show around new workspace
- ☐ Introduce to coworkers
- ☐ E-mail and social media accounts/passwords assigned?
- ☐ E-mail etiquette (To:, CC:, BCC:, distribution lists)
- ☐ Add to group mailing lists
- ☐ Explain use of group mailing lists, online chat, etc.
- ☐ Review calendar and meeting requests
- ☐ Invite to required meetings, one-on-ones, group gatherings
- ☐ Printers
- ☐ Licenses (if necessary) for SCM, IDEs, tools
- ☐ Intranet access
- ☐ Personnel Web site
  www.intranet.company.com/personnel
- ☐ Public folders
  /File/Public/xx_Department
- ☐ Internet access
  - Business purposes only?
  - Streaming music or videos allowed?
  - Downloading (music or applications) allowed?
- ☐ Establish initial goals:
  - 1 - _____
  - 2 - _____
  - 3 - _____
  - 4 - _____
  - 5 - _____
- ☐ Describe required reports (i.e., status reports, etc.)
- ☐ Set one-week and one-month check-in meetings
- ☐ "The speech"
  - It's a marathon
  - Teamwork is one of our values
  - Customer satisfaction
  - Consulting demeanor
  - The rest of our values
  - Joining an outstanding team
  - Joining a team with a history of doing outstanding work (examples)
- ☐ Create "about me" message, send to supervisor to send to team
  - Solicit or take photo
- ☐ Get home e-mail, home address, mobile phone number for your records
- ☐ Inventory new employees' skills; add to knowledge bank
- ☐ Arrange invitation to "Company 101" orientation session

Getting New Programmers Started Off Right (2/3)

WELCOME DAY SCHEDULE FOR RON LICHTY'S NEW EMPLOYEE, Bob

9:00 a.m.	**Meet with Ron Lichty** Ron's office
9:30 a.m.	**Facilities Tour with Susan, our office manager** Meet in Lobby
10:00a.m.	**IT Training with Dennis** Your desk
10:30 a.m.	**HR Training with Gary** Conference Room: Days of Our Lives, 8th floor *Bring your new hire binder and a pen*
11:00 a.m.	**Meet with Career Manager, John Smith** John's office
12:00pm	**Lunch with Buddy - Jill Miller** Restaurant TBD - the firm is buying
1:30pm	**Meet with teammate on your first project, Arun Nguyen** Arun's cube
2:00 p.m.	**Meet with client-side teammate Johnny Arnold** Johnny's cube
2:30 p.m.	**Meet with best practices lead, Jim Starr** Jim's cube
3:00 p.m.	**Meet with business partner, Chris Barnstable** Chris's office
4:00 p.m.	**Time Tracking Tour with Jack, our finance guy** Jack's office

- A buddy-led **office tour** focused on making a round of introductions
- An **objectives kickoff** with the new hire's direct manager
- A **facilities tour** from the office manager: restrooms, office supplies, copy areas, printer and fax machine locations, conference room maps, water coolers, coffee and how to make more, snack machines, break rooms, safety information, fire escapes, and other important locations
- A **visit from IT** to review final computer, printer, network, e-mail, and Web configurations, walk through voicemail and VPN instructions, and
do Q&A
- A **benefits overview** delivered by HR; completion of the stacks of paperwork required to get new hires started these days, including the confidentiality agreement
- **Meetings with team members**—from other development leads to project and product managers—showing the new hire how to get to the servers, ideally on the new hire's own computer, as well as where in the file systems to find key team, department, and company assets; the corporate and department intranets; and bios and photos and directories of teammates and colleagues
- A long (hour-and-a-half) **lunch with the buddy or career mentor** (the other of the two should take the new hire to lunch later in the week);
each has been clearly instructed that the firm is paying and reminded if necessary how to expense lunch
- **A personal welcome by one or more senior managers** of the organization

Getting New Programmers Started Off Right (3/3)

Introductions

Company to New Hire:

- The corporate mission, vision, and values
- The metrics by which the department is measured
- The history of the company and its products, services, and business model
- An overview of the company's architecture and technology
- Important company milestones
- An organizational overview with relevant org charts, showing where the new hire fits in the employee universe (including in the context of employees just met)

New Hire to Company

- Welcome message:
  Get the new hire to write a couple of short paragraphs of personal information and past history, or interview the hire so that you can write it. You will incorporate a photo of the new hire as you craft this personal welcome message that you'll send out to the department, region, or company.

Ensuring Success

- Get them started on achievable tasks quickly.
- Collocate them with team members who can also act as mentors.
- Make sure they have immediate access to relevant documentation.

Effective Programming Manager: Managing Down (1/3)

Initial Expectations

First set of tasks:
Set up a local copy of the environment and sandbox

- Install the tools
- Load up the code to be worked on
- Compile the code
- Check the result against the team's official build to confirm

There are processes to walk your new hire through

- Your **established software development lifecycle**, and how projects are tracked and reported
- **Time recording**,
- **Vacation approval** and other time off
- The policy for **reimbursement of expenses**
- The **performance review process**
- Your new hire's **calendar** and the meetings to expect
- Ordering **business cards**
- Applications for **company credit cards** and **telephone credit cards**
- Telephone conference calling

Effective Programming Manager: Managing Down (2/3)

A good manager …

- Hires great programmers: otherwise you will loose time dealing with the issues they cause
- Turbocharges the Team: drive continuous improvement
- Creates a structure team: great programmers need to be surrounded by competent programmers who can be relied upon to do the day-to-day work of predictably making systems and products.
- Facilitates instead of dictates: a manager is maximizing his time and skills when he facilitates getting the right decision, rather than dictating it.
- Protects his team: protect his staff from the ebb and flow of problems, issues, and “opportunities” that wash over an organization on a daily basis.
- Manages information: **organizations are drowning in data, but starving for information.**
- Sets objectives: be careful when linking objectives with financial incentives
- Executes performance reviews: performance reviews are controversial as the lead to mixed results. However giving regular feedback is important.
- Delivers more positive than negative feedback.
- Knows when to cut losses: dismiss programmers than cannot live up to expectations. Do not tolerate non-performance because it drags the full team.
- Delivers results and celebrates successes.

Effective Programming Manager: Managing Down (3/3)

- People hate change but love progress
- You miss 100 percent of the shots you never take.
- Every hour of planning saves about a day of wasted time and effort.
- Prioritize. sometimes, it is urgent to wait.
- Don't boil the ocean.
- Common sense is not common practice.
- Leading by example occurs whether you like it or not.
- I praise loudly; I blame softly.
- Firefighters who get rewarded carry matches.
- For every 10-percent increase in problem complexity, there is a 100-percent increase in the software solution's complexity. That's not a condition to try to change (even though reducing complexity is always desirable); that's just the way it is.
- We have two ears and one mouth. Use them in this ratio
- Beware of Lunde's Law: Any time the senior developers on a project are replaced by a new team, the new team will eventually find a compelling reason for a total rewrite. And that new team is wrong.
- Work smart not hard.
- Software isn't released, it's allowed to escape

Effective Programming Manager: Managing UP (1/1)

Managing-Up is how you effectively manage your boss and those to whom he reports.

A good manager …

- Understands his boss: contributes to his boss' goals
- Packages communication: focusses on the important element without distracting with the details
- Understands the boss' boss: understand how your boss is evaluated
  - Improving quality
  - On-time delivery
  - Delivered Innovation
  - Created Products
  - Issues reporting
  - Realized Collaboration and partnering
  - Revenue and/or expenses
- Is an model employee: low maintenance, solved problems before they become an issue

Managing-Out is how you effectively manage relationships with your peers and others in departments in the organization outside of yours.

A good manager…

- Collaborates with the department
- Understands other departments
  - Finance (invoices, payments, and payroll)
  - Purchasing (purchase requests, vendor qualification, and expediting purchases)
  - Legal (contracts, intellectual property, mergers and acquisitions)
  - Human Resources (hiring, firing, performance reviews, internal and external salary equity, internships, and recruiting staff and contractors)
  - Marketing (market research, trade shows, corporate communications, press releases, collateral, and spiffs)
  - Sales (customer feedback, customer issues, customer connections, and the customer pipeline)
  - Technical support (critical customer issues, customer feedback, trends, and outstanding issues)

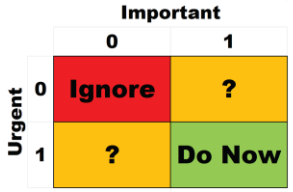
Effective Programming Manager: Managing Outside/Yourself (1/1)

Managing outside the company:

- Customers
- Technology providers
- Technology innovators and work disruptors
- Tools vendors and suppliers
- Government, trade, and international standards organizations
- Industry consortiums
- Professional organizations
- University educators
- Local connections

Managing yourself:

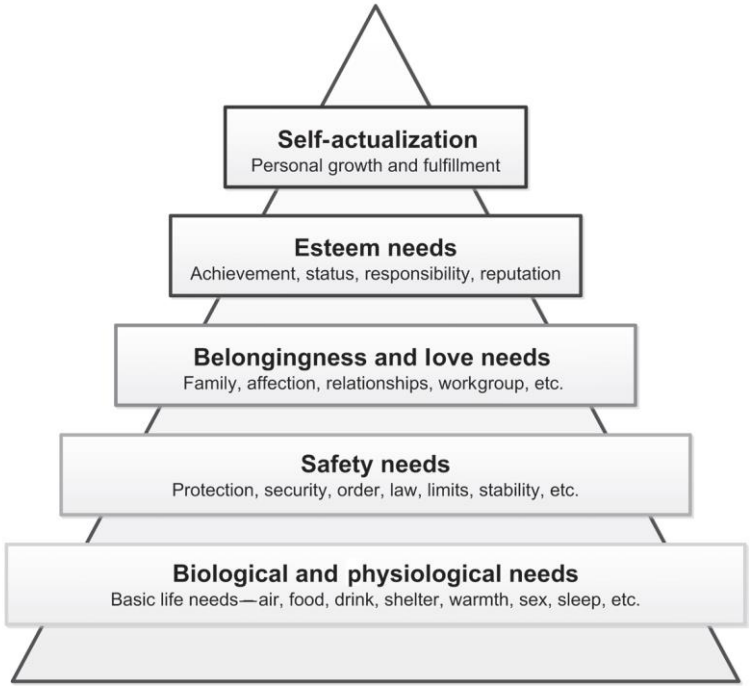
- Personal style: appearance, work ethics, behaviour
- Time and priority management: prioritize according to importance and urgency:



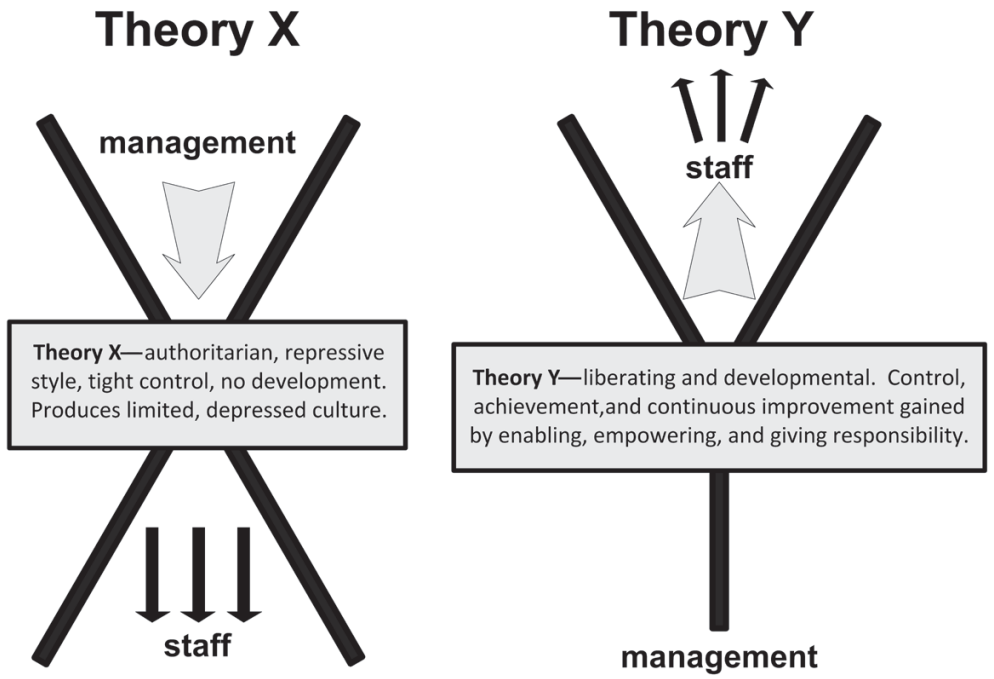
- Communications management
- Management practices:
  - Listen and pay attention to what is important;
  - Be part of the solution not the problem
  - Make progress every day
- Follow-up management
- Find a mentor

Motivating Programmers (1/5)

Maslow's Hierarchy of Needs

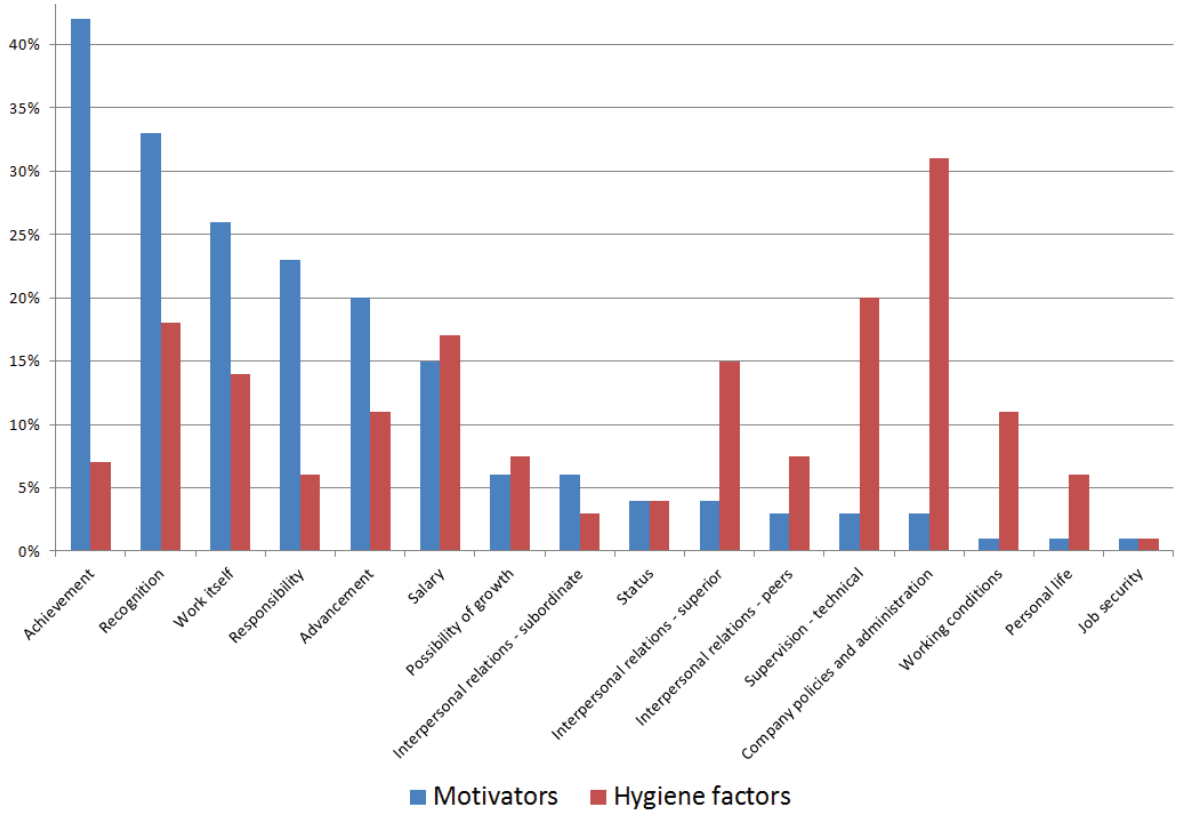


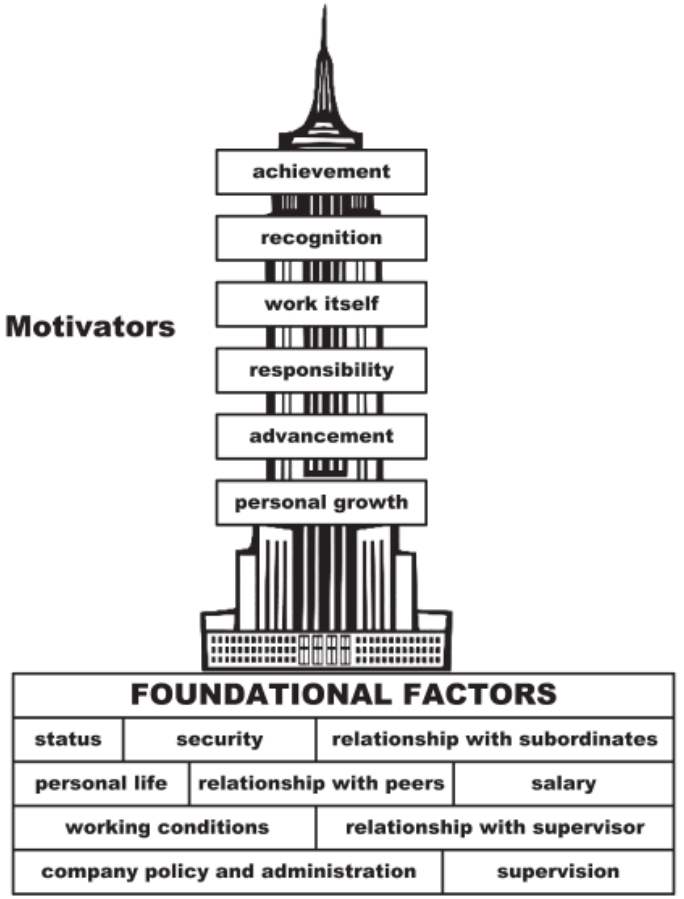
McGregor's X-Y Theory



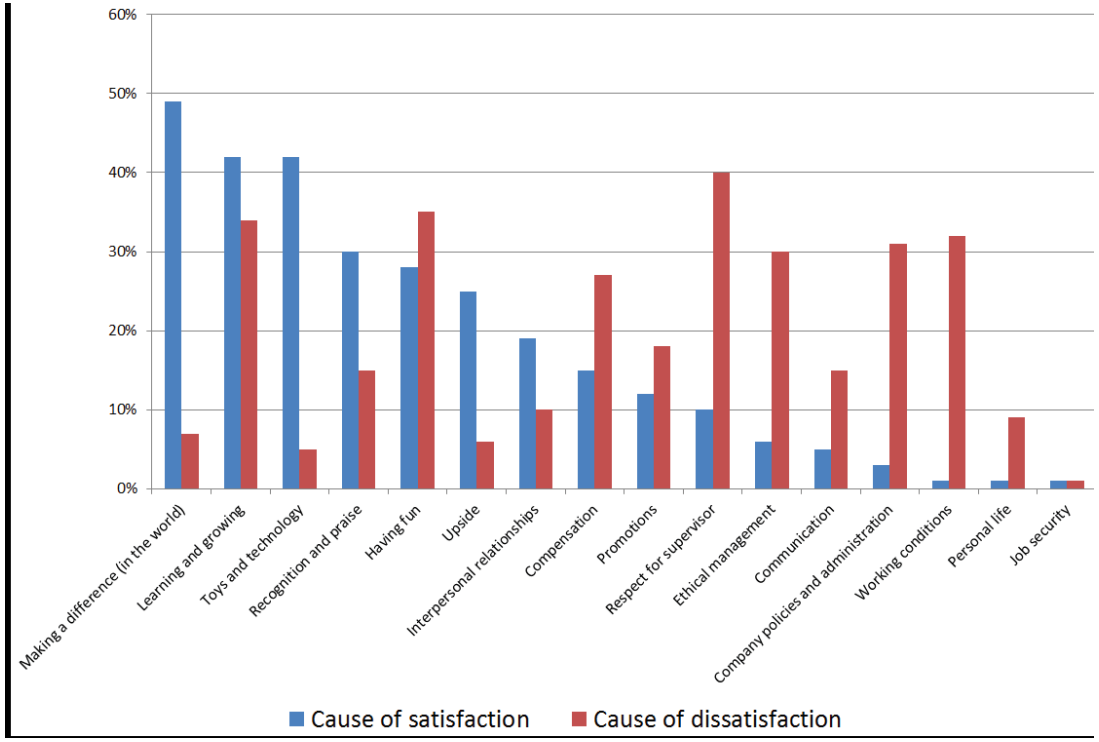
Motivating Programmers (2/5)

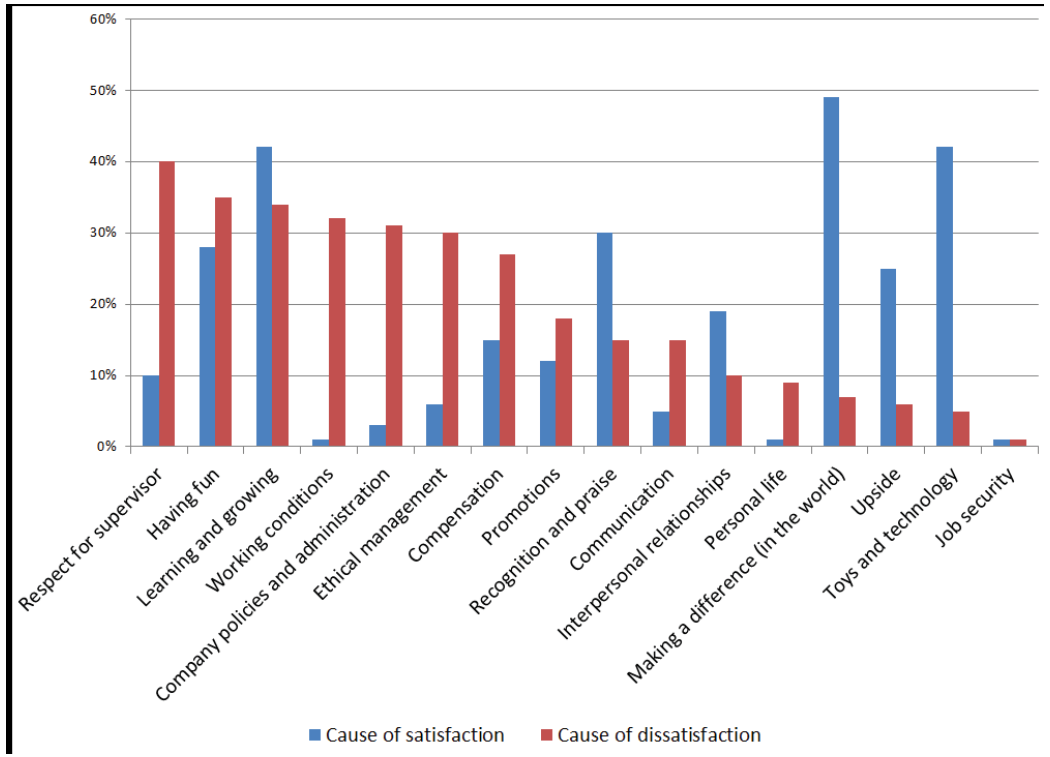
Herzberg's Motivation and Hygiene Factors





Motivating Programmers (3/5)





Motivating Programmers (4/5)

Key Foundational Factors (Hygiene):

- Respect for supervisor
  - Gain technical respect
  - Respect Others
  - Lead example
  - Manage and Coach
- Having fun
  - Camaraderie
  - Informal contacts
- Learning and growing
  - Provide environment with opportunities
- Good working conditions
  - “No jerks” allowed rule
  - Work-life balance
  - Personal space
- Sane company policies and administration
  - Communicate vision, policies and rules
  - Protect against organizational distractions
- Ethical management
  - Character: integrity and trusted
  - Attitude: serving and responsible
  - Excellence: continuous improvements
  - Competency: expertise and effectiveness
  - Conduct: respectful and loyal
- Fair compensation
  - Compensate fairly: Make sure if you publish your staff’s compensation you would not be ashamed
  - Promote appropriately

Motivating Programmers (5/5)

Key Motivating Factors:

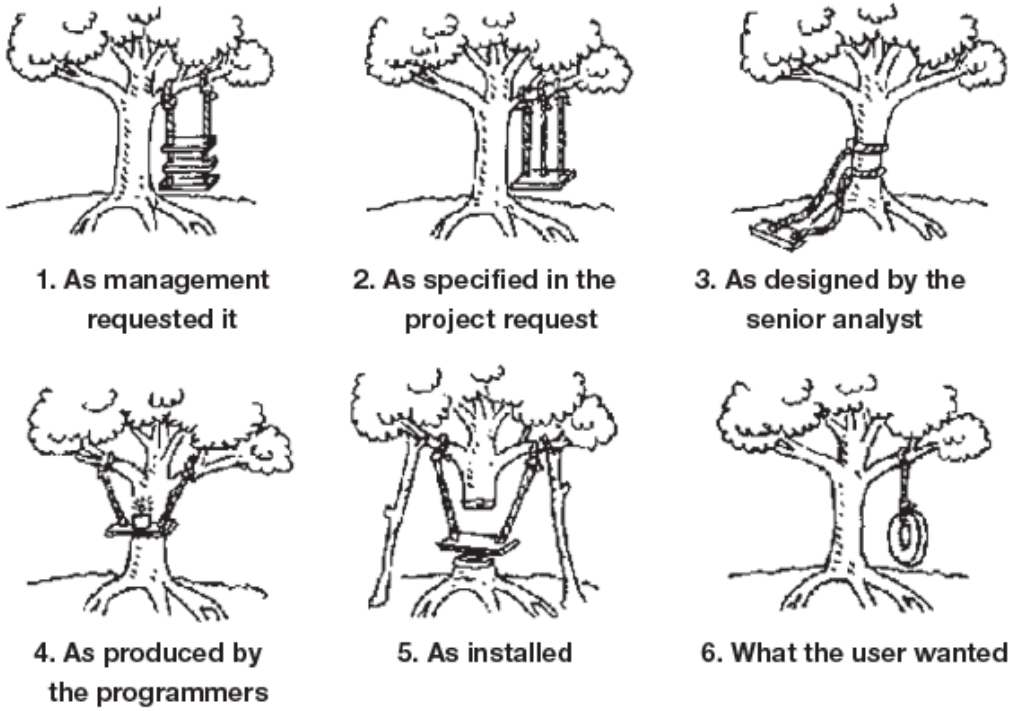
- Making a Difference in the World: have an impact
- Learning and Growing: continue to learn
- Toys and Technology: equipped with state-of-the-art tools
- Recognition and Praise: extra mile is appreciate
    - If anything goes bad, I did it. If anything goes semi-
    - good, we did it. If anything goes real good, you did it.
- Having Fun with Your Staff
- Upside or other incentives:
    - Higher salary
    - Significant bonus
    - Stock options (or equivalent)
    - Strong company growth
    - Increased perks

Creating a Successful Programming Culture (1/1)

Characteristics of a successful programming culture:

- Mutual Respect: you don't have to like but you must be able to work collaboratively.
- Innovation: unless your team is charged with research, new unproven technologies can be ratholes.
    - To encourage innovation, you must encourage risk.
    - To encourage risk, you must tolerate failures.
    - To tolerate failures, you must be patient.
    - To communicate your encouragement of innovation, you must reward dedication, not success.
- Standards: you should expect your team to follow standards.
    - Requirements
    - Coding
    - Code quality
    - Use of automated document generation
    - Development process
    - Build process
    - Test case tracking
    - Traceability
    - Check-ins (e.g., code review before check-in)
    - Source control
    - Open source use
    - Intellectual property (IP)
- Delivery: shipping or going live, on-time completion needs to be explicitly valued.
- Communication: ensure that everyone is moving in the right direction.
- Fix the problem, not the blame.
- Communication Among Virtual Teams: setting up monitors and cameras to give remote employees more "presence"
- Fairness: rewards (and critique) will be delivered fairly, not arbitrarily.
- Empowerment but balanced
- I inspect what I expect.
- Professionalism: doing what you say you'll do.
- No Jerks and Bozos: no tolerance for unacceptable behavior that threatens the productivity of the team.
- Excellence: honoring, demanding, and expecting excellence
- Teamwork and Collaboration: high-performance teams, balance between rewarding teams and rewarding heroes,
- Passion
- Customer Focus: "It's the Customer Experience, Stupid!", gaining a customer perspective is critical.
    - The ultimate metric that I would like to propose for user friendliness is quite simple: If this system were a person, how long would it take before you punched it in the nose?
- Learning; code is never finished nor perfect, programmers have to learn when to let it go.
- Environment: don't penny wise and pound foolish, don't waste hour to explain a few dollars on a much needed tool

Successful Software Delivery (1/4)



- Defining the project:
  - Clear requirement and assumption
  - Iterations are critical
  - Requirements explain the what not the how.
  - Seek to delight customers
  - Use ballpark and magnitude for effort don't waste time on details.
  - Recognize Nonnegotiable Dates
  - Inspire the team

Successful Software Delivery (2/4)

- Planning the work:
  - Break the Project into Features, and Prioritize Them
  - Do risk-first development.
  - Break Features into Tasks and Sub-Tasks
  - Engage Your Team in a Bottom-Up Estimate
  - Assemble Task Estimates into a Project Estimate
    - One very conservative industry rule of thumb is that programmers typically spend no more than 55 percent of their time coding.
  - Look for the Limitations on Estimation
  - Get Agreement Around the Risks, Not Just the Schedule
- Development is a triangle: good, fast, cheap—pick any two!
- Always present three options—three alternatives to choose among.
- Allocate Sufficient Time for Unit and Project Testing
- Estimation Is a Unique Challenge Every Time. Variations caused by:
  - Programming paradigm (procedural to object-oriented = 20 percent design/80 percent development to 50/50)
  - Methodology (Agile result in lower QA overhead)
  - Size of project
  - Target platform and delivery mechanism
- Determine the Pace of the Project

Successful Software Delivery (3/4)

- Kicking off the plan:
  - Organize a project kickoff
  - Create a definition of done
  - Define success
    - Stakeholders scorecard
  - Establish a project workbook
    - Product Development Lifecycle

- Executing the work:
  - Design the Work
  - The perfect is the enemy of the good.
  - Hold a Design Review
  - Complete a Prototype to Inform the Design
  - Set Agreed-Upon Milestones
    - PoC, Alpha, Beta, Release Candidate, General Availability
  - Confirm That Regular Check-In Meetings Have Been Set
  - Actively Drive Development
    - Ensure That Agreed-Upon Standards and Requirements Are Met
    - Leverage Test-Driven Development
    - Hold Stand-Up Meetings
    - Insist on Code Reviews

Successful Software Delivery (4/4)

- Running the End Game:
  - No New Features
  - Run the Product
  - Be Prepared to Declare Success and Start on the Point Release
  - Know When to Cut Your Losses
  - OEM and International Versions
- Delivering the Software:
  - :Celebrate
  - Retrospect:
    - What did we do well?
    - What could we do better?
  - Share
  - Refactor