

Stress Management in the Computer Programming Field
Kyle Brady
<http://www.kyle-brady.com>

Abstract

Examination and analysis of stress, and the management of, in the computer programming industry, with an interest in corporate-sponsored methodologies.

In the world of computer programming, stress is well understood to be a factor of daily life for its practitioners. Generic computer users understand what it means to be frustrated with computers, and architecting the software itself is merely an extension of the same frustrations by an order of magnitude. However, despite the common knowledge of the high levels of stress for programmers and software engineers alike, the very corporate structures that employ them tend to plead ignorance, as if the problems would solve themselves – even if a majority of the stressors are corporate-inspired.

The most crucial point in understanding computer programming is that it requires a level of concentration, intelligence, and broad-base knowledge alien to most other professions – a deadly combination of engineering precision and abstract artistic creativity. The reasons are broad and many, not to mention technical in nature, but such activities are extremely demanding and “can strain the limits of human cognition, stress, and fatigue” (Galinsky et. al 1993). Architects of software come to expect this over their training and careers, many of which begin during the teenage years, and therefore internalize it as a job hazard, rather than an issue potentially worth addressing in a concrete manner.

The interactions between users and the software must be accounted for, requiring large volumes of study and analysis of humanity’s tendencies, and is not accomplished overnight. In these days of modernity, the act of programming has evolved significantly from the days of punch cards a half-century ago, but is still limited: every action, analysis, and reaction must be hand-written, lest a bug appear to only frustrate the end user. Development is typically done inside a special environment that facilitates software creation and debugging, but encourages a reactionary coding style: write, test, fix, and repeat until deemed functional. A growing movement is interested in what they have deemed “ahead-of-time” programming, which would “allow the programmer to offload as much of the memory burden and mental work as possible”, with the intended effect of “greater satisfaction and lower levels of stress and fatigue” (Snell). While it has proven helpful in limited studies, as well as achieving granular adoption in some

development environments, this solution ignores what many consider to be the exacerbating stress factor: management.

In various studies of programmer stress, a common theme of high stress levels, causing various problems, appeared. Depending on the stage of the project, the symptoms varied: project planning saw high anxiety and depression, code writing saw irritability and low morale, and debugging/testing saw all possible manifestations. The problem, it seems, is one that most programmers could have easily articulated without the need for costly research. Management of programmers is not typically effective, and tends to cause unnecessary problems. Many believe this is because the management theories used are based on “techniques derived from the manufacturing industry” (Glass), and revolve around arbitrary value markers such as lines written per week, features implemented, or percentage of debugging completed. One such researcher, Fujigaki, states that “the software process is a learning and communication process”, implying that the employee-manager relationship should have a more fluid model, since software itself is very fluid in its creation.

Others may argue that programmers themselves are the cause, since they are a different brand of individuals, as previously mentioned, with character traits apparent across the board, regardless of employer, industry, gender, nationality, or even generation. If an average person were asked at random what the characteristics of a programmer were, they are likely get it right: a high need to achieve, to reach high levels of job satisfaction and personal accomplishment, with very little emphasis on socialization (Zawacki). With these traits in mind, it would appear that managing structures based on a need for social appreciation would be problematic, leaving some to suggest moving such individuals with a higher valuation of social life to leadership positions, as a middle-man between the personality clashes.

Regardless of the true source, the fact remains that stress for computer programmers tends to reach high levels that are often maintained for their entire careers. With minor deviations, such as modern Internet giant *Google* (“Top 10 Reasons to Work at Google”), most organizations ignore the stressful conditions under which their employees operate, refusing to help alleviate these problems even when they can, and do, easily escalate into larger situations. In the interviews with the two employees at *CIENA* – one a former programmer turned specialized support analyst (Mike), another a high-level programmer with customer-facing duties (Shawn) – they echoed similar sentiments regarding the dearth of stress management options, regardless of whether they believed such options would prove useful.

Both individuals expressed personal methods through which de-stressing was possible, such as exercise, but that the daily worklife did not allow for such luxuries – no massages, no guided meditation, and no yoga rooms. Despite not having corporate-sponsored stress management options, Shawn and Mike both expressed a love for what they are involved in, even going so far as to say that perhaps the stress factor was not novel and could be considered part of the job description.

As the world becomes exponentially more digital with each passing year, the need for the software industry to evolve becomes even greater. Whether the issue is employee management,

development methodologies, or the job itself, computer programming will continue to become more difficult and stressful, and the projects ever larger, more complex and grand. If a middleground is not reached for those involved in the production and architecture of software, the consequences for the humanity's technological future, and the world-at-large, could be momentous.

References

Fujigaki, Y. (1993, July). Stress analysis: A new perspective on peopleware. *American Progress*, 6(7), 33-38.

Galinsky, Traci L., Roger R. Rosa, Joel S. Warm, William N. Dember. (1993). Psychological determinants of stress in sustained attention. *Human Factors*, 35(4), 603-614.

Glass, Robert L. (1997, April). Practical programmer: The ups and downs of programmer stress. *Communications of the ACM*, 40(4), 17-19.

Google. (2009). *Top 10 reasons to work at Google*. Retrieved April 30, 2009, from <http://www.google.com/intl/en/jobs/lifeatgoogle/toptenreasons.html>

Snell, James L., Ph.D. (1997). Ahead-of-time debugging, or programming not in the dark. *International Conference on Software Technology and Engineering Practice*, July.

Zawacki, R. (1993, November). Motivating IT people in the 90s: An 'alarming drop' in job satisfaction. *Software Practices*, 3(6), 4-5.