Why Crunch Mode Doesn't Work: 6 Lessons

There's a bottom-line reason most industries gave up crunch mode over 75 years ago: It's the single most expensive way there is to get the work done.

by Evan Robinson

Executive Summary

When used long-term, Crunch Mode slows development and creates more bugs when compared with 40-hour weeks.

More than a century of studies show that long-term useful worker output is maximized near a five-day, 40-hour workweek. Productivity drops immediately upon starting overtime and continues to drop until, at approximately eight 60-hour weeks, the total work done is the same as what would have been done in eight 40-hour weeks.

In the short term, working over 21 hours continuously is equivalent to being legally drunk. Longer periods of continuous work drastically reduce cognitive function and increase the chance of catastrophic error. In both the short- and long-term, reducing sleep hours as little as one hour nightly can result in a severe decrease in cognitive ability, sometimes without workers perceiving the decrease.

Introduction

In the aftermath of ea_spouse's post on LiveJournal, quality-of-life conversations in the game development business have taken on a new life and a new urgency. Ea_spouse received thousands of comments to her original post — followed quickly by major media coverage. Thousands of people around the net participated in a vast, spontaneous discussion that explored issues like mandatory overtime, productivity, job portability, laziness, unionization, lawsuits and the general evil of corporations.

I've spent 20 years developing and managing software projects. Every year that passed — and every project I worked on — fueled my growing conviction that Crunch Mode is grossly, destructively, expensively inefficient. It's common sense that the more hours people work, the less productive they become. But, over time, I noticed that the productivity losses that result from working too many extra hours start taking a bigger toll faster than most software managers realize. As I dug around, I was stunned to discover that I was hardly the first one to figure this out: my observations have been common knowledge among industrial engineers for almost a century.

I've amassed a personal collection of source information over the past 15 years, this summary mainly includes information that you can readily find on the Web. I don't want you to take my word for it: I want you to be able to go out and read the original source material for yourself.
The History

In 1908 — almost a century ago — industrial efficiency pioneer Ernst Abbe published in *Gessamelte Abhandlungen* his conclusions that a reduction in daily work hours from nine to eight resulted in an increase in total daily output. (Nor was he the first to notice this. William Mather had adopted an eight-hour day at the Salford Iron Works in 1893.)

In 1909, Sidney J. Chapman published *Hours of Labour*, in which he described long-term variation in worker productivity as a function of hours worked per day. His conclusions will be discussed in some detail below.

When Henry Ford famously adopted a 40-hour workweek in 1926, he was bitterly criticized by members of the National Association of Manufacturers. But his experiments, which he'd been conducting for at least 12 years, showed him clearly that cutting the workday from ten hours to eight hours — and the workweek from six days to five days — increased total worker output and reduced production cost. Ford spoke glowingly of the social benefits of a shorter workweek, couched firmly in terms of how increased time for consumption was good for everyone. But the core of his argument was that reduced shift length meant more output.

I have found many studies, conducted by businesses, universities, industry associations and the military, that support the basic notion that, for most people, eight hours a day, five days per week, is the best sustainable long-term balance point between output and exhaustion. Throughout the 30s, 40s, and 50s, these studies were apparently conducted by the hundreds; and by the 1960s, the benefits of the 40-hour week were accepted almost beyond question in corporate America. In 1962, the Chamber of Commerce even published a pamphlet extolling the productivity gains of reduced hours.

But, somehow, Silicon Valley didn't get the memo. Ea_spouse writes:

> The current mandatory hours are 9am to 10pm — seven days a week — with the occasional Saturday evening off for good behavior (at 6:30pm). This averages out to an eighty-five hour work week [sic].

Actually, working 9am to 10pm, six days a week, plus 9am to 6:30pm one day a week comes out to \(6 \times 13 + 9:30 = 87.5\) hours per week — but after that many hours, who's still counting?

Electronic Arts is no different than many high-tech companies in this regard. For them — and anyone else who wants to increase their employees' productivity and sanity — let's take a look at some of the assumptions management makes regarding hours, output, efficiency, and production costs; and see how a century of industrial research has conclusively, consistently proven those assumptions wrong.

What Management Wants

What is management trying to achieve when it sends employees off on death marches? Do we honestly believe that the CEO of EA is happy that people are in the office 24/7 working their asses off?

Management wants to achieve maximal output from employees — they want to produce a (good) product as cheaply as possible. They also want to avoid hiring extra resources that increase the cost of the finished goods unless absolutely necessary. On the surface of it, Crunch Mode looks like the most obvious and logical way to reconcile these two desires.

Assuming output can be measured in discrete units, a manager who hasn't read the research may reason that if someone produces, say, 16 units of output in eight hours, they should produce 18 units in nine hours and 20 units in ten hours. To express that view as a simple equation, we can write:

\[
O = \frac{X}{Y} \times t
\]

Where \(O\) is total output, \(X\) is the given output during a benchmark number of hours, designated by \(Y\), and \(t\) is the actual
number of hours worked. In this hypothetical situation, increasing time $t$ is the simplest way to increase output $O$.

That assumption may be valid in the limited case where the hours of work are extended over a brief period, for example, to meet a looming deadline. But research — and long experience in other industries — have shown that the limits to such overtime spurts are reached sooner than most people realize. And when those limits are reached, the spurts turn into bogs.

**Hourly Productivity Is Important**

A more realistic view of worker output would take into account the changes in hourly output that result from a change in the length of the working day. Those changes result mainly from two sources: simple physical and mental fatigue that occurs in the later hours of a long day, and accumulated physical and mental fatigue that builds up over an extended period of long working days.

This more complex view can be represented by the following equation:

$$O = P(t_1, t_2, t_3 \ldots t_n)$$

Where $O$ is total output and $P()$ represents the changes in hourly productivity that occur over times $t_1$ to $t_n$. In this equation $P()$ is a function, not a constant. $P()$ will vary by worker, because some workers produce more than others. $P()$ will also vary by hour, because humans are not machines and do not do exactly the same amount of work in hour 14 of a job as they do in hour 1. Finally, $P()$ will vary according to the recent history of the worker, because people don't work as well the morning after a late night as they do the morning after a good night's sleep.

Sidney J. Chapman's *Hours of Labour* (1909), included (roughly) the following diagram:


Depicting $P$, the “long-period variations (with the length of the working day) of the marginal value of a fixed quantity of labour” $OX$ is increasing hours worked in a day and $OY$ is increasing value. If $On$ hours are worked, the total value produced is the area $Onda$. For a lengthier discussion, see [http://www.worklessparty.org/timework/chapman.htm](http://www.worklessparty.org/timework/chapman.htm). Observe that the height of the curve $P$ represents worker productivity (output per unit time at a given number of hours worked per day).

Astute readers will note that there is a point, $b$, where working more hours doesn't create more value. In fact, after $b$, each additional hour worked produces negative value. How can this be?

Chapman's diagram of the work curve assumes that a working day of a given length is maintained over a considerable period of time. Thus it incorporates both simple and accumulated fatigue into its model. At first the declines in output per hour simply reflect the effects of fatigue on both quantity and quality of work performed toward the end of a given day. But eventually daily fatigue is compounded by cumulative fatigue. That is, any additional output produced during extended hours today will be more than offset by a decline in hourly productivity tomorrow and subsequent days.

Even during a single “day” of extreme duration, output may come to a standstill as an exhausted employee becomes unable to function. Or output can turn negative as stupefied employees commit catastrophic errors that destroy previously
completed work or capital.

In factory terms, a worker's production rate decreases over time. A worker who is creating 10 widgets/hour at the beginning of a shift may be producing only 6/hour at the end of the shift, having peaked at 12/hour a couple of hours in. Over time, the worker works more slowly, and makes more mistakes. This combination of slowdown and errors eventually reaches a point of zero productivity, where it takes a very long time to produce each widget, and every last one is somehow spoiled. Assembly-line managers figured out long ago that when this level of fatigue is reached, the stage is set for spectacular failure-events leading to large and costly losses – an expensive machine is damaged, inventory is destroyed, or a worker is seriously injured.

In terms of knowledge workers, a programmer produces more good code and fewer bugs when well-rested. We take the first hour or so of the day getting into the groove. The next few hours tend to be our best ones. Later in the day, as we get tired, we get less done per hour — it takes a long time to fix a simple bug or add a simple feature that we would have handled in minutes earlier in the day. Pushed just a little farther — and it seems that much of the computer entertainment industry is working at this extreme most of the time — an overtired IT worker may trash valuable files requiring extra work to restore backups or have an accident on the way home that takes her offline for months.

**Lesson One, then, is this:** *Productivity varies over the course of the workday, with the greatest productivity occurring in the first four to six hours. After enough hours, productivity approaches zero; eventually it becomes negative.*

---

**Where's The Break-Even?**

If productivity essentially decreases over a working day, and working lots of hours results in reduced productivity, how do we establish a method to maximize total output, and figure out where the break-even point lies?

Unfortunately, quantifying knowledge worker output is a hard problem. I would love to be able to give a simple equation you can plug a few numbers into and pull out the magic number of hours each person should work to maximize their output. I can't, because even when such equations finally exist, it will be impossible to find and agree on the basic numbers to plug into them. Common programming measurements, like lines of code and function points are either easy to collect and of questionable value or difficult to define and collect. Useful measures like number of bugs created and number of bugs fixed are viewed with suspicion that they will be used unfairly in annual reviews (or gamed by clever programmers in anticipation of annual reviews or performance bonuses).

Artist output is easier by some measures (number of models or images) and as difficult by some others (subjective quality, look and feel, complexity of model).

Tester output is easy in one sense (number of unique bugs found), expensive in a second (code coverage), and extremely hard in a third (percentage of total bugs found).

Overall, most companies seem to have fallen to a least-common-denominator measure of team output. Either the game ships and sells — or it doesn't. While this is indeed the metric that matters most to shareholders, it's not terribly useful as a measure of productivity, especially daily or hourly productivity.

**Lesson Two, then is this:** *Productivity is hard to quantify for knowledge workers.*

So we are forced to draw analogies from other industries.

From the Work Less Institute of Technology, “*Psychophysics in Cyberia*” (written in response to the ea_spouse posting):

> It was over a century ago that Dr. Ernst Abbe conducted his observations on working time and output at the Zeiss Optical Works in Jena, Germany. Dr. Abbe, director of the plant, reduced the daily hours of work from 9
to 8 and kept careful records of daily output per worker before and after the change. What he found confirmed observations from throughout the 19th century: a moderate reduction in working time increased total output. In The Economics of Fatigue and Unrest, Philip Sargent Florence summed up the accumulated evidence to the 1920's:

"Reduction from a 12-hour to a 10-hour basis results in increased daily output; further reduction to an 8-hour basis results in at least maintaining this increased daily output; but further reductions while increasing the hourly rate of output, seems to decrease the total daily output."

Hugo Münsterberg's 1913 Psychology and Industrial Efficiency:

...Ernst Abbe, the head of one of the greatest German factories, wrote many years ago that the shortening from nine to eight hours, that is, a cutting-down of more than 10 per cent, did not involve a reduction of the day's product, but an increase, and that this increase did not result from any supplementary efforts by which the intensity of the work would be reinforced in an unhygienic way. This conviction of Abbe still seems to hold true after millions of experiments over the whole globe.

From the Work Less Party, Tom Walker's Prosperity Covenant:

That output does not rise or fall in direct proportion to the number of hours worked is a lesson that seemingly has to be relearned each generation. In 1848, the English parliament passed the ten-hours law and total output per-worker, per-day increased. In the 1890s employers experimented widely with the eight hour day and repeatedly found that total output per-worker increased. In the first decades of the 20th century, Frederick W. Taylor, the originator of "scientific management" prescribed reduced work times and attained remarkable increases in per-worker output.

In the 1920s, Henry Ford experimented for several years with work schedules and finally, in 1926, introduced a five day, 40 hour week for six days pay. Why did Ford do it? Because his experiments showed that workers in his factories could produce more in five days than they could in six. At every step along the way — in the 1840s, the 1890s and the 1920s — the consensus of business opinion insisted that shorter hours would strangle output and spell economic ruin.

Lesson Three is this: five-day weeks of eight-hour days maximize long-term output in every industry that has been studied over the past century. What makes us think that our industry is somehow exempt from this rule?

What About Short-Term Output?

If 40-hour weeks offer the most reasonable long-term arrangement for maximizing output, can we expect to get short-term gains from short periods of longer workdays or extended workweeks?

In a word, briefly. You can get more work out of more hours for several days to a couple of months, depending upon how much longer the workday is.

It is intuitively obvious that a worker who produces one widget per hour during an eight-hour day can produce somewhere between eight and 16 widgets during a 16-hour day. As we've seen, that's the essential logic behind Crunch Mode's otherwise inexplicable popularity. But worker productivity is largely dependent upon recent history. From the Executive Summary of Scheduled Overtime Effect on Construction Projects, published by The Business Roundtable in 1980:

Where a work schedule of 60 or more hours per week is continued longer than about two months, the cumulative effect of decreased productivity will cause a delay in the completion date beyond that which could have been realized with the same crew size on a 40-hour week.

Productivity drops when working 60-hour weeks compared with 40-hour weeks. Initially, the extra 20 hours a week makes up for the lost productivity and total output increases. But the Business Roundtable study states that construction productivity starts to drop very quickly upon the transition to 60-hour weeks. The fall-off can be seen within days, is obvious within a week...and just keeps sliding from there. In about two months, the cumulative productivity loss has declined to the point where the project would actually be farther ahead if you'd just stuck to 40-hour weeks all along.

(The same report cites studies that show total output while working eight-hour days is either 16% or 20% higher than total output working 9-hour days.)

So, yes, Crunch Mode can increase output over the short term. But, at 60 hours per week, in no case should "the short term" be defined as anything more than eight weeks long. At that point, the costs strongly begin to outweigh the advantages. Not only have you lost all the gain those increased hours bought; you've also got tired, angry, burned-out workers. When you return them to a 40-hour week, their output will be sub-par for some time while they recover.

At 87.5 hours per week? Lacking hard data, I would estimate that productivity would drop to under 50% of baseline within a month. An extra 47.5 hours per week (more than double “normal” hours) would provide a large initial surge of extra output.

**Lesson Four is this:** *At 60 hours per week, the loss of productivity caused by working longer hours overwhelms the extra hours worked within a couple of months.*

---

**The Sleep Factor**

There's another, shorter window that needs consideration in assessing the useful limits of Crunch Mode. That is: How long can someone be productive if they're not getting enough sleep?

Colonel Gregory Belenky, the Director of the Division of Neuropsychiatry at Walter Reed Army Institute of Research, does research for the Pentagon on maximizing the productivity and alertness of soldiers under combat conditions. In his 1997 paper, *Sleep, Sleep Deprivation, and Human Performance in Continuous Operations*, he found that:

Laboratory studies show that mental work declines by 25% during each successive 24 hours of continuous wakefulness. Sleep-deprived individuals are able to maintain accuracy on cognitive tasks, but speed declines as wakefulness is extended.

...  
In our study, FDC [artillery Fire Direction Center — *ER*] teams from the 82nd Airborne division were tested during simulated continuous combat operations lasting 36 hours. Throughout the 36 hours, their ability to accurately derive range, bearing, elevation, and charge was unimpaired. However, after circa 24 hours they ... no longer knew where they were relative to friendly and enemy units. They no longer knew what they were firing at. Early in the simulation, when we called for simulated fire on a hospital, etc., the team would check the situation map, appreciate the nature of the target, and refuse the request. Later on in the simulation ... they would fire without hesitation regardless of the nature of the target.

...  
At 15 days into the simulation the 4 hour sleep/night battery is firing less than a third of the rounds that the 7 hour sleep/night battery is firing.

**Lesson Five is this:** *Continuous work reduces cognitive function 25% for every 24 hours. Multiple consecutive overnights have a severe cumulative effect.*

---

**Cognitive Decay and Error Rates**
One of the biggest productivity sinks created by Crunch Mode is the increase in the number of errors produced. While most errors will be easily fixed, there will be some that could cost all of the output you’ve gained by crunching. The longer you crunch, the greater your odds of creating a big, expensive, schedule-busting monster.

Programmers, artists, and testers aren't paid for their bulging muscles and phenomenal ability to move mass from point A to point B. They're paid for their brains. Longer hours or, especially, insufficient sleep (as little as 1-2 hours less per night) does serious damage to their ability to use those brains productively.

Hugo Münsterberg's 1913 *Psychology and Industrial Efficiency*:

> It has been well known for a long while how intimate the relations are between fatigue and industrial accidents. … it can be traced everywhere that in the first working hours in which fatigue does not play any considerable role, the number of accidents is small, and that this number sinks again after the long pauses.

Colonel Belenky points out that the consequences to soldiers of losing as little as one hour of sleep per night include “[r]educed … higher order mental abilities that sustain situational awareness…[r]educed individual and unit effectiveness, errors, accidents…”

It's a good thing knowledge workers rarely have to worry about “friendly fire.”

From *Sustained Reduced Sleep Can have Serious Consequences*:

In a study on the effects of sleep deprivation, investigators at the University of Pennsylvania found that subjects who slept four to six hours a night for fourteen consecutive nights showed significant deficits in cognitive performance equivalent to going without sleep for up to three days in a row. Yet these *subjects reported feeling only slightly sleepy and were unaware of how impaired they were*. [emphasis mine — ER]

The Los Angeles Times, *Sleepy Medical Interns Called a Road Hazard*, by Karen Kaplan, January 13, 2005:

Studies have shown that being awake for 21 hours impairs drivers as much as having a blood-alcohol concentration of 0.08, which is the legal limit for noncommercial drivers in the U.S.

It's ironic. Most software companies will fire an employee who routinely shows up drunk for work. But they don't think twice about putting the fate of this year's silver bullet project into the hands of people who are impaired to the point of legal drunkenness due to lack of sleep. *In fact, they will demand that these people work to the point of legal impairment as a condition of continued employment.*

The risks are real -- and the errors made can be truly catastrophic. From *The Promise of Sleep* by Dr. William Dement, pp 51-53:

The night of March 24, 1989 was cold and calm, the air crystalline, as the giant *Exxon Valdez* oil tanker pulled out of Valdez, Alaska, into the tranquil waters of Prince William Sound. In these clearest of possible conditions the ship made a planned turn out of the shipping channel and didn't turn back in time. The huge tanker ran aground, spilling millions of gallons of crude oil into the sound. … In its final report, the National Transportation Safety Board (NTSB) found that sleep deprivation and sleep debt were direct causes of the accident. … The direct cause of America's worst oil spill was the behavior of the third mate, who had slept only 6 hours in the previous 48 and was severely sleep deprived.

The final report of the *Rogers Commission* (on the Space Shuttle Challenger accident) said that the decision to launch made during a critical teleconference was flawed. The Human Factors Analysis section suggests that lack of sleep “may have contributed significantly.”

If the bit rot caused by sleep deprivation can lose battles, kill patients, beach oil tankers, and blow up space shuttles, consider what it can do to your $15 million game project.
Lesson Six is this: *Error rates climb with hours worked and especially with loss of sleep*. Eventually the odds catch up with you, and catastrophe occurs. When schedules are tight and budgets are big, is this a risk you can really afford to take?

### What's It All Mean?

It comes down to productivity. Workers can maintain productivity more or less indefinitely at 40 hours per five-day workweek. When working longer hours, productivity begins to decline. Somewhere between four days and two months, the gains from additional hours of work are negated by the decline in hourly productivity. In extreme cases (within a day or two, as soon as workers stop getting at least 7-8 hours of sleep per night), the degradation can be abrupt.

Many of the studies quoted above come out of industrial environments, and it may be argued that the more creative mental work of programmers, artists, and testers is fundamentally different. In fact, *it is different*, and Colonel Belenky explicitly addresses that:

> In contrast to complex mental performance, simple psychomotor performance, physical strength and endurance are unaffected by sleep deprivation.

The ability to do complex mental tasks degrades *faster* than physical performance does. Among knowledge workers, the productivity loss due to excessive hours may begin sooner and be greater than it is among soldiers, because our work is more affected by mental fatigue.

When *ea_spouse* wrote:

> The current mandatory hours are 9am to 10pm — seven days a week — with the occasional Saturday evening off for good behavior (at 6:30pm).

she was telling us that the entire team her husband worked on was working at far less than their optimal productivity. They had been working 60+ hour weeks for months already, before management tried to kick them into an 87.5-hour-per week super crunch.

In most times, places, and industries over the past century, managers who worked their employees this way would have been tagged as incompetent — not just because of the threat they pose to good worker relations, but also because of the risk their mismanagement poses to the company's productivity and assets. A hundred years of industrial research has proven beyond question that exhausted workers create errors that blow schedules, destroy equipment, create cost overruns, erode product quality, and threaten the bottom line. They are a danger to their projects, their managers, their employers, each other, and themselves.

Any way you look at it, Crunch Mode used as a long-term strategy is economically indefensible. Longer hours do not increase output except in the short term. Crunch does not make the product ship sooner — it makes the product ready *later*. Crunch does not make the product better — it makes the product *worse*. Crunch raises the odds of a significant error, like shipping software that erases customer's hard drives, or deleting the source tree, or spilling Coke into a server that hasn't been backed up recently, or setting the building on fire. (Yes, I've seen the first three of these actually happen in the last, bleary days of Crunch Mode. The fourth one is probably only a matter of time.)

Managers decide to crunch because they want to be able to tell their bosses "I did everything I could." They crunch because they value the butts in the chairs more than the brains creating games. They crunch because they haven't really thought about the job being done or the people doing it. They crunch because they have learned only the importance of *appearing* to do their best to instead of *really* doing their best. And they crunch because, back when they were programmers or artists or testers or assistant producers or associate producers, that was the way they were taught to get things done.
But it's not the only way. In fact, the literature shows, over and over again, that it is the very worst way. And that's the bottom-line reason most industries gave up crunch mode over 75 years ago. Managers, shareholders and employees all stand to benefit from time-tested management practices that will deliver better products, sooner, less expensively -- and with less wear and tear on human resources and public reputations.

Acknowledgments

I'd like to specifically acknowledge:

- Tom Walker of the Work Less Institute for his help in providing documents and reviewing this article;
- Sara Robinson for multiple reviews and revisions, not to mention putting up with incessant talk about dusty old research;
- The IGDA for providing me a bigger soapbox.

Appendix: Collected Sources

As I noted, almost all the documents cited here are available on the Web. While much more data is available in the print literature, I more or less deliberately selected from online studies that the reader can access and use immediately. Here is a collected list of documents I have either quoted from or consulted extensively.

- EA spouse
- Psychology and Industrial Efficiency, Hugo Muensterberg, 1913, available at Classics in the History of Psychology, maintained by Christopher D. Green, York University, Toronto, Canada.
- Prosperity Covenant, Tom Walker.
- Sleep, Sleep Deprivation, and Human Performance in Continuous Operations: Colonel Gregory Belenky, Director of the Division of Neuropsychiatry, Walter Reed Army Institute of Research, U.S. Army Medical Research and Materiel Command.
- Sustained Reduced Sleep Can have Serious Consequences, Linda Cook, NINR, March 2003.
- Sleepy Medical Interns Called a Road Hazard, Los Angeles Times, Karen Kaplan, January 13, 2005. Archived at Mischievous Ramblings
- Mischievous Ramblings, Evan Robinson, “It's Not Just Abusive, It's Stupid!”
- Mischievous Ramblings, Evan Robinson, “Can People Really Program 80 Hours a Week?”
- Mischievous Ramblings, Evan Robinson, “Staying Awake”

IGDA Resources/Links

- Discuss quality of life issues in the discussion forums
- Quality of Life Summit: An IGDA Think-Tank

- "Quality of Life in the Game Industry: Challenges and Best Practices" white paper
- IGDA open letter: Quality of Life Issues are Holding Back the Game Industry
- IGDA quality of life advocacy and resources

Author Bio

Evan Robinson started in the game business at 19 as a developer for TSR. By 22, he was building computer games as an independent developer for EA. In the two decades since then, he has been a grunt programmer, lead engineer, technical director, director of engineering, process consultant, and technical auditor for some of the industry's best-known companies. Evan's publications and frequent GDC presentations -- starting in the conference's early years -- have established him as one of the industry's voices on best software engineering practices and profitable programmer management. He lives in Vancouver, BC.

Copyright © 2005 Evan Robinson.

The opinions expressed in this article do not necessarily represent the IGDA.