acmqueue

The Responsive Enterprise: Embracing the Hacker Way

Soon every company will be a software company

Erik Meijer and Vikram Kapoor

As of July 2014, Facebook, founded in 2004, is in the top 20 of the most valuable companies in the S&P 500, putting the 10-year-old software company in the same league as IBM, Oracle, and Coca-Cola.[3] Of the top five fastest-growing companies with regard to market capitalization in 2014 (table 1), three are software companies: Apple, Google, and Microsoft (in fact, one could argue that Intel is also driven by software, making it four out of five).

Hot and upcoming companies such as Über, Tesla, and Airbnb, which are disrupting the traditional taxi, car, and hotel markets, respectively, are also all fundamentally software companies.

Conversely, the bottom five companies, which lost market capitalization in the first half of 2014 (table 2), are mostly traditional enterprises in business for decades.

Given this information, the logical question to ask is Why are software-based companies taking over the world?[2] The answer is simply that powering companies by software allows them to be responsive and data-driven and, hence, able to react to changes quickly. To explain this, let's take an informal look at control theory.

TABLE 1: **Five Fastest-growing Companies**

Company	Market Cap	Year Founded
Facebook	118.80%	2004
Apple	55.90%	1976
Intel	48.50%	1968
Gilead Sciences	47.60%	1987
Microsoft	41.80%	1975

TABLE 2: **Bottom Five Companies that Lost Market Capitalization**

Company	Market Cap	Year Founded
GlaxoSmithKline	6.30%	1900 (mergers)
Citigroup	5.00%	1812
Philip Morris	4.90%	1900
Sanofi	4.30%	1973 (mergers)
WalMart Stores	2.60%	1962

In control theory, an *openloop* (nofeedback) control system computes the control input to the system using only the external input, without taking into account the current output of the system (figure 1). An openloop control system works well with full knowledge of a static world, but it falls apart when the environment evolves, or when there is no perfect model of the system under control. An example of an openloop system is the cab driver who after every trip returns to the same hotel to hang out with his fellow cabbies and smoke a few cigarettes, with a small chance of picking up new customers, and without taking into account that theater performances downtown have just finished and thus demand is elsewhere.

A *closedloop control* system takes into account the current output of the system and feeds this information back to the controller, which combines these measurements with the current external input to continuously correct and optimize the system under control (figure 2). Über, for example, knows the current traffic conditions, the exact supply and demand for transportation at any instant, and the historical price elasticity of customers; hence, it can optimize pricing for taxi rides in realtime and direct drivers to the locations with the highest demand for transportation.

In Microsoft Office, new features were always introduced and tested via extensive user studies—until Office 2007 when the existing user interface was replaced by a "ribbon," to the dismay of many users. When the start button disappeared in Windows 8, every Windows user in the world lost their accumulated muscle memory for commanding the Windows operating system. Such missteps occur when companies grow older and become deaf to feedback and external input. Companies that ignore feedback become *nocontrol* systems that produce output based only on their internal echo chambers (figure 3).

FIGURE 1

Open-Loop System

FIGURE 2

Closed-Loop System

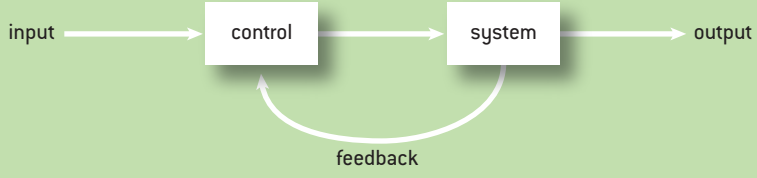
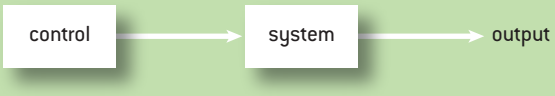


FIGURE 3 **No-Control System**



Internal processes and the desire to control risk start to overrule real innovation. Investors' pressure to optimize for quarterly results and the bottom line further encourage deafness. Last but not least, the workforce ossifies because of a recursive feedback system that over time ejects employees who are open to external stimuli in favor of people who worship legacy and process.

A RESPONSIVE ENTERPRISE IS A SOFTWARE COMPANY

In the next decade every business will be digitized and effectively become a software company. Leveraging software, and, in general, computational thinking, to make a business responsive to change using a closedloop feedback system will be crucial to surviving in this new world where business = data + algorithms. Some examples of responsive companies follow.

Unlike traditional cab companies, Über does not own its fleet. Its value is in the realtime data and algorithms Über uses to transform this data into decisions. In a few years, Über will likely automate away all humans in the equation by replacing drivers with selfdriving cars, which themselves are possible only because of advances in software technology.

Netflix does not operate its own data centers but runs its whole operation on public cloud infrastructure. Simply put, cloud computing virtualizes hardware into software.[5] Instead of dragging in a physical computer or storage device, cloud computing achieves the same effect with a few lines of software code that in the blink of an eye allow companies to add compute and storage capacity, or to release surplus.

Software promotes agility by dramatically speeding up the feedback loop between output and input. In the past, companies could measure their performance every quarter, making it difficult to adjust quickly to changes in the environment. In contrast, Facebook ships new versions of its product multiple times a day, with enhancements and fixes determined by realtime feedback from actual use of the Web site. Companies such as Amazon and Booking.com continuously perform A/B testing or multiarmed bandit experiments on users to optimize purchase rates on their Web sites.

Amazon even makes its A/B testing technology available to third-party developers for free (https://developer.amazon.com/public/apis/manage/ab-testing). The same holds for other companies such as Facebook and Netflix, which make most of their internal software available as open source.

Continuous delivery and A/B or multiarmed bandit testing not only require new products to be rolled out quickly, but also mean that wrong decisions can be rolled back quickly. Accepting that decisions can be wrong, and acknowledging the errors, requires courage that traditional riskaverse enterprises tend to eliminate by introducing process.

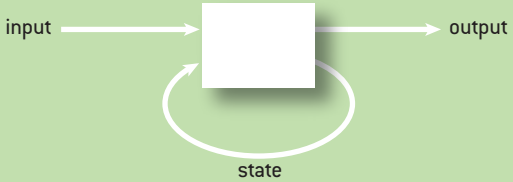
Responsive enterprises accept that failures will always happen and guard themselves from cascading failures by purposefully causing failures. Just the thought of creating a deliberate failure to make a system more robust against failure immediately sends many traditional middle managers over the edge. At Netflix, however, it is common practice to "let an army of monkeys loose" in the data center to pull out virtual cables and wreak havoc. Many agile development methods propose writing tests before writing code. It is impossible, however, to faithfully model the complexities of the environment in which deployed code runs in the test environment. Instead, software application failures should be treated like any other failures, deploying code in production immediately and rolling it back when problems occur.

Elementary psychology teaches us that in order for humans to learn (i.e., improve), feedback about their actions must be immediate. Smokers keep smoking because lighting up gives them immediate satisfaction, but the feedback effect of lung disease comes years later. Developers are humans, too, and, hence, can be stimulated to write better software by providing them immediate and physical feedback about the quality of their code. Companies can implement this feedback loop by making developers wear pagers that wake them up in the middle of the night in case problems arise and by making them responsible for the "whole stack" (also known as the DevOps model). Decoupling the development process from the operations side removes a valuable (recursive) feedback loop in the system and thus reduces the responsiveness of the system as a whole.

Another way of viewing a closedloop system is as a (nondeterministic) Mealy machine.[7] This is a finite-state machine where the next output value and the next state are determined by the current state and the current input (figure 4). The Mealy-machine view emphasizes an organization's "corporate memory" and posits that decisions on the current input may be influenced by a state that has been accumulated over many iterations using past inputs and outputs. At a meta level, however, the environment in which the enterprise operates will also change over time; hence, the corporate memory needs to be "reset" occasionally since it is not tracking relevant inputs anymore.

As a concrete example, take Microsoft, which since 1975 has accumulated an impressive corporate memory of shipping boxed products. While the world was craving boxed software, the Microsoft machine was able to optimize the output to generate spectacular profits. The environment changed, however, and began demanding cloud services, and suddenly the knowledge of optimizing for boxed software became an impediment to growth. Microsoft has had

FIGURE 4 **Mealy Machine**



to lay off thousands of people recently to "forget" the past and change to a "cloud first" direction.

Every company should have a "tenth man" ("When nine people agree on something, it's the tenth man's responsibility to disagree no matter how improbable the idea")[6] or devil's advocate who injects a certain amount of randomness and chaos into the process. This prevents falling into the trap of getting stuck in a local optimum, or worse, making wrong decisions because changes in customer preferences were ignored and nobody dared to point out that the emperor had no clothes.

THE PEOPLE SIDE OF THE RESPONSIVE ENTERPRISE

From a people perspective, running a company powered by software is totally different from running a traditional enterprise. Once a company realizes that the road to success is to accept that a responsive enterprise is powered by software, it must apply open-loop feedback control to make the engine run smooth, and, moreover, it must deeply embrace developers as the engine for growth.

Software developers are nothing like traditional suitwearing corporate employees, and it is this aspect that is often the hardest for pointyhaired bosses to understand. Perhaps the difference is most succinctly expressed here: "You cannot race sheep, and you cannot herd racehorses;"[8] and "Domesticate developers like beekeepers domesticate bees."[1]

The "Seven Aspects of our Culture," as described in a Netflix slide presentation (Values are what we Value; High Performance; Freedom and Responsibility; Context, not Control; Highly Aligned, Loosely Coupled; Pay Top of Market; and Promotions and Development) provides a crisp formulation of people-management values for a softwarecentric enterprise.[4] Here is our take on four of Netflix's seven core values:

Context, not Control. Anyone who has performed improvised jazz or improv theater knows from experience that creativity can only flourish under strong constraints. Letting developers wander around without clear goals in the vastness of the software universe of all computable functions is one of the major reasons why projects fail, not because of lack of process or planning. If we give developers strict guidelines and orders about what to optimize for, and provide them immediate and realtime feedback about the consequences of their decisions on those goals, the competitive nature of geeks will take over and the system will quickly iterate toward an optimal solution. The only goal of middle management is to provide and enforce this context, and keep developers happy and productive within these welldefined bounds. We do not care about developers' opinions on the long-term strategy of the enterprise.

Pay Top of Market. In professional sports it is normal to spend many millions of dollars to attract and pay top players to assemble a team that can compete in the top of the division. To attract and retain the best developers (and to entice young people to pursue a career in hightech instead of becoming lawyers or stockbrokers), they need to be compensated well and given a big piece of the pie, based on the value they deliver and their limited shelf life, not on the number of hours they clock.

Highly Aligned, Loosely Coupled. Replacing the word *war* with *software* in the following excerpt from the *Fleet Marine Force Manual 1, Warfighting*, immediately brings to mind the day-to-day practice of software development:[9]

"*Software* is a complex endeavor. It is shaped by the human will. It is characterized by

friction, uncertainty, fluidity, danger and disorder. While the nature of *software* is constant, it remains unpredictable, and is affected by a mix of physical, moral and mental factors. While *software* has the characteristics of both art and science, it is primarily shaped by human experience."

The divisional organizational structure and operation of armies has been honed over many centuries to deal with such situations, and, hence, there is much to learn from the military. In particular, software development should follow the Philosophy of Command: "In order to support the fluid and chaotic nature of the battlefield, command must be decentralized. Subordinate leaders must use their own initiative to accomplish tasks which support their senior's intent."

Freedom and Responsibility. Tradition, as defined by Wikipedia, "is a belief or behavior passed down within a group or society with symbolic meaning or special significance with origins in the past." In a responsive enterprise, traditions have no role. In the traditional enterprise, traditions are encoded as process. Process is what causes organizational deafness, which leads to the downfall of a no-feedback system. Blind adherence to process also drives out creative people and rewards nonproductive bean counters. Process is necessary to produce adequate results with mediocre employees in the fast-food industry. The contrapositive is that in a high-tech company composed of first-rate hackers, there is no need for process.

No company embodies the hacker culture better than Facebook, and CEO and founder Mark Zuckerberg explained the idea eloquently in his letter to investors when Facebook filed for its IPO: "Hacker culture is also extremely open and meritocratic. Hackers believe that the best idea and implementation should always win—not the person who is best at lobbying for an idea or the person who manages the most people."[10]

TRANSFORMING INTO A RESPONSIVE ENTERPRISE

Software is an unstoppable force. To survive, enterprises must ride the wave instead of resisting change and ending up in the graveyard along with blacksmiths and agile coaches. Here are some guidelines to keep in mind.

1. *Embrace that software is at the heart of everything you do.* In the 19th and 20th centuries, machines and logistics were the pivots of most organizations. If you had the best machines and supply lines in your industry, you were the winner. In the 21st century having the best software is the path to business glory—that is, your software is directly and continuously shaped by what the market actually wants and needs (closedloop, datadriven design decisions driving implementation).

Software lifts physical limitations on the speed with which services can be created and changed. Delivery channels such as mobile devices and widespread Wi-Fi technology eliminate physical delivery borders that had to be overcome in the past. The consequence of the new reality of a softwaredriven enterprise is that CEOs have to deeply understand software (and the hackers who create it) and how to incorporate it into their business model in order to run successful organizations. It is not surprising that some of the most successful CEOs (Mark Zuckerberg, Larry Page, Bill Gates, and Larry Ellison) have developer backgrounds.

2. *Organize yourself as a fractal closedloop realtime feedback system at each level of the organization, with bidirectional feedback loops between layers.* Organizations tend to hold on to the "proven" business model that made them successful in the past, but in the new era of software-driven enterprises, most business models of the past are invalidated. The revolution behind the responsive

enterprise is in the continuous feedback loop, which, in combination with creativity, relentless experimentation, and datadriven decision making, unleashes the ability for enterprises to learn and react to their environments in realtime. Organizations need to undertake a softwaredriven business model whereby static business plans are replaced by strategic thinking about market opportunities. These hypotheses are then translated into A/B experiments that are tested in the market on actual customers and analyzed in realtime to check whether the hypothesis was valid or not.

While a traditional business plan looks forward based on (untested) historical assumptions, the softwaredriven business model is based upon continuously interpreting data derived from experiments and feedback.

3. *Effective organizations are hierarchical, like it or not.* For some reason companies seem to keep ignoring the success of the hierarchical structure used by the military, which has evolved from a "survival of the fittest" mentality. Instead, they keep experimenting with new organizational structures such as "functional" or "flat" when sticking with a divisional organization would make more sense.

Translated into the concepts introduced here, companies should be recursively decomposed into communicating closedloop control systems where the context (goals) of each inner loop is set by the immediately enclosing loop, but how those goals are realized is left to the discretion of the teams that make up the inner loop. As each inner ring delivers feedback to the enclosing ring, the organization as a whole becomes a single closedloop control system. Note that in programming, divide and conquer is also one of the most effective ways to solve complex problems, so it makes a lot of sense to apply that same technique to structure an organization.

4. *Developers are the engines of growth and are responsible for the tactical level of the operation.* As a responsive enterprise is recursively decomposed, it bottoms out at squads of about 10 developers, or in Amazon.com speak, "two-pizza teams." These hacker squads are shoveling the coal below deck, keeping the ship running. They are the ones who actually create the software that powers the organization. Therefore, keeping hackers happy is the most important people aspect of the responsive enterprise. People management becomes hacker management. Developing software has been compared with juggling eggs—that is, to maintain a productive flow, developers should have nothing on their minds other than the code they are working on. Management should provide developers with all the software and hardware they need to do their work, with no questions asked.

Developers should be compensated like sports stars, not only because they are the ones who effectively create the value of the company, but also because the intensity of coding takes a toll on the bodies and minds of developers. Just like competitive athletes, they simply burn out by the time they reach their mid-30s.

5. *Middle management only provides operational context linking strategy with tactics.* In many traditional companies management often is synonymous with (subtle) control—control of (assumed) risk, control of behavior, control of the uncontrollable. Instead of making things happen, middle management should *allow* things to happen by setting clear boundaries and goals, and then just let go. Managers need to change from sheep herders to beekeepers. That means no standup meetings, no burn-down charts, no weekly status reports, and, especially, no planning poker.

Not many people are fit to become beekeepers. They have the highly responsible job of finding exactly the right constraints under which developers are most productive. Placing too many constraints on developers will make them unhappy and, hence, unproductive; too few constraints will make them lose focus and, hence, be unproductive. Of course, the constraints under which developers operate should align perfectly with the overall strategy of the company, and status information must be properly fed back up the chain, putting even more pressure on the judgment ability and improv skills of middle managers.

Understanding and appreciating the hacker's mind is impossible if you are not a hacker yourself. Managing hackers can be done only by people with hacker backgrounds, just like the best sports coaches once were top athletes; and like successful coaches, successful middle managers should be compensated handily, but ejected quickly if they prove to be ineffective at making their teams win.

6. *Decision making is mostly driven by data as opposed to status and seniority.* When developers write software, they use tools such as debuggers, profilers, and static checkers to improve the quality and performance of their code. When it comes down to decisions around code, only the numbers and data generated by those tools count. The same is true for a softwaredriven enterprise. Control information that flows down the chain and feedback information that flows up the chain must be backed by real data and measurements, and great care must be taken to avoid injecting any subjective interpretation that pollutes the raw data—until the moment where human interpretation and creativity are required. Every layer in the organization must have a data scientist/data wrangler (which is really just a specialized developer role) to help mine, interpret, and visualize data to help in decision making.

7. *The senior leadership team sets strategic (macro) long-term direction.* The responsive enterprise is a divisional organization where the strategic direction comes solely from the senior management team. Senior managers are responsible not only for making sure the feedback engine executes smoothly, but also, more importantly, for implementing a meta feedback loop that makes sure that the enterprise is listening to the correct input signals from a constantly changing external world, that the output produced by the enterprise is still the desired output, and, last but not least, that the feedback they receive remains relevant for their decision making. As a consequence, senior management is also responsible for overall people management such as acquiring new capabilities to react to external changes, expelling obsolete capabilities in the organization, and flushing corporate memory.

CONCLUSION

Sooner than you may think, every company will be a software company. The obvious way to run a software company is as a meta software application, recursively structured as a layer of commuting closedloop feedback systems, using a strictly layered architecture modeled after the time-proven hierarchical structure of armies and applying software-inspired profiling and debugging techniques to optimize the profitability of the enterprise. On the operational side, instead of *talking* about code, companies should follow the "hacker way" and focus on *writing* code using continuous improvement and iteration, leveraging the best developers they can afford.

REFERENCES

1. Card, O. S. 2008. How software companies die. *Windows Sources*: 208; http://www.netjeff.com/humor/item.cgi?file=DeveloperBees.
2. Denning, S. 2014. Why software is eating the world. *Forbes* (April 11); http://www.forbes.com/sites/stevedenning/2014/04/11/why-software-is-eating-the-world/.
3. Dogs of the Dow. 2014. Largest companies by market cap; http://www.dogsofthedow.com/largest-companies-by-market-cap.htm.
4. Hastings, R. 2008. Netflix culture: freedom and responsibility; http://www.slideshare.net/reed2001/culture-1798664.
5. Helland, P. 2012. Condos and clouds. *ACM Queue* 10(11); http://queue.acm.org/detail.cfm?id=2398392.
6. Legge, A. 2014. What *World War Z* can teach you about critical thinking. *Evidence Mag*; http://evidencemag.com/world-war-z/.
7. Mealy, G. H. 1955. A method for synthesizing sequential circuits. *Bell System Technical Journal* 34(5): 1045–1079.
8. Thomas, D. 2008. Developing expertise: herding racehorses, racing sheep. *InfoQueue*; http://www.infoq.com/presentations/Developing-Expertise-Dave-Thomas.
9. U.S. Marine Corps. 1997. *Warfighting*. U.S. Government Printing Office; http://www.marines.mil/Portals/59/Publications/MCDP%201%20Warfighting.pdf.
10. Zuckerberg, M. 2012. Mark Zuckerberg's letter to investors: the hacker way. Reprinted in *Wired*; http://www.wired.com/2012/02/zuck-letter/.

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

ERIK MEIJER (emeijer@appliedduality.com) is the founder of Applied Duality and professor of big-data engineering at Delft University of Technology. He is well known for his contributions to programming languages such as Haskell, C#, Visual Basic, Hack, and Dart, as well as his work on big-data technologies such as LINQ and the Rx Framework.

VIKRAM KAPOOR (v.kapoor@prowareness.nl) is the founder of Prowareness and iSense. He was voted "IT CEO of the year" in the Netherlands by his peers in 2013. The mission of Prowareness is to increase the success and happiness of people and organizations by facing challenges together.