No free lunch for software after all

Michiel van Genuchten

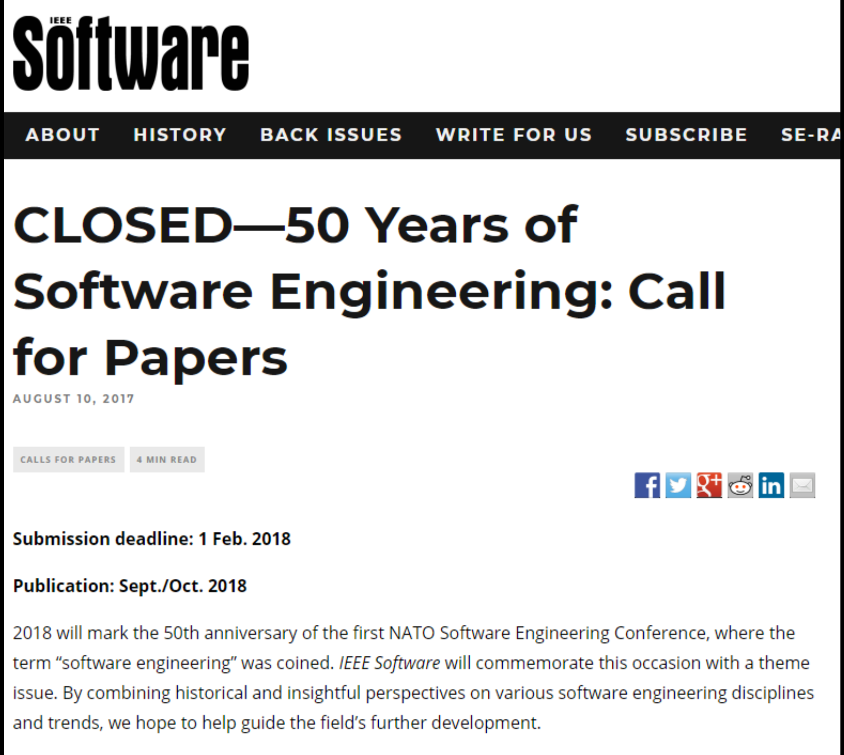
VitalHealth Software

genuchten@ieee.org

2018-03-26

1968

1972



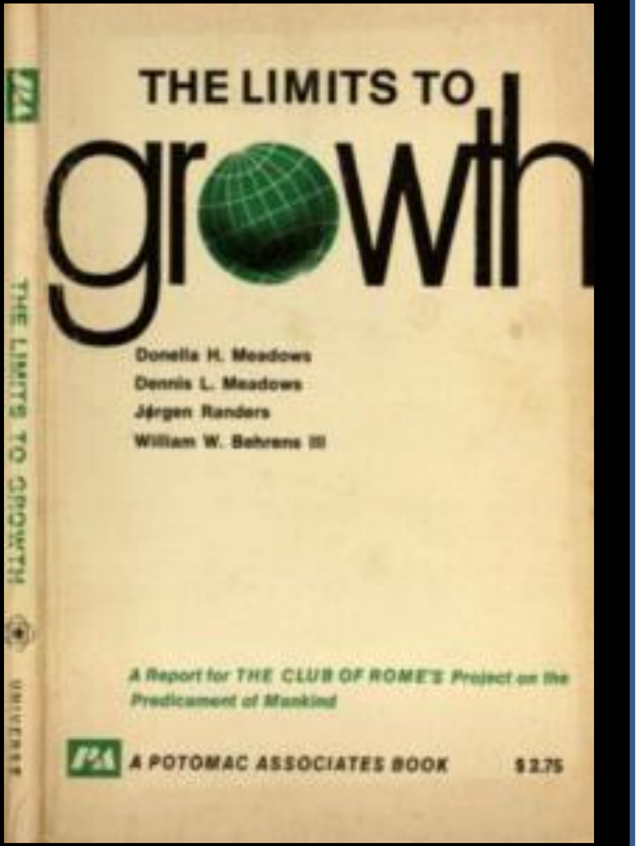
CLOSED—50 Years of Software Engineering: Call for Papers

AUGUST 10, 2017

Submission deadline: 1 Feb. 2018

Publication: Sept./Oct. 2018

2018 will mark the 50th anniversary of the first NATO Software Engineering Conference, where the term “software engineering” was coined. *IEEE Software* will commemorate this occasion with a theme issue. By combining historical and insightful perspectives on various software engineering disciplines and trends, we hope to help guide the field’s further development.



"One goal of this column is to build better quantitative insight into how software impacts various businesses. How the product uses the software and how the company built it are equally important."

40 columns in 2010-18

Bosch, RealNetworks

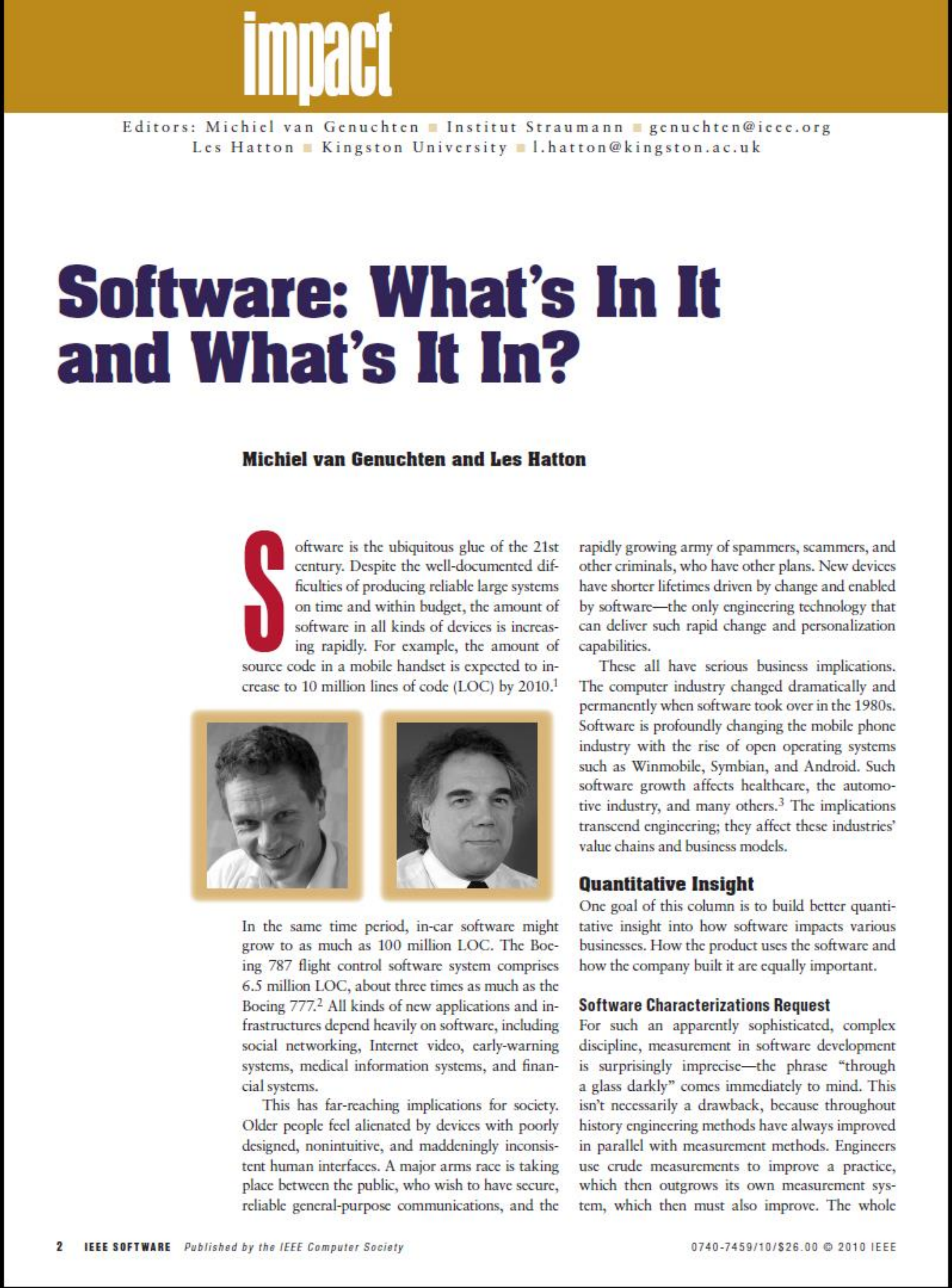
Philips, Honeywell

Hitachi, Uni of Queensland

Tomtom, Fujixerox

Microsoft, Shell

CERN, Oracle, Airbus, JPL, ESA



impact

Editors: Michiel van Genuchten ▪ Institut Straumann ▪ genuchten@ieee.org
Les Hatton ▪ Kingston University ▪ l.hatton@kingston.ac.uk

Software: What's In It and What's It In?

Michiel van Genuchten and Les Hatton

Software is the ubiquitous glue of the 21st century. Despite the well-documented difficulties of producing reliable large systems on time and within budget, the amount of software in all kinds of devices is increasing rapidly. For example, the amount of source code in a mobile handset is expected to increase to 10 million lines of code (LOC) by 2010.[1] In the same time period, in-car software might grow to as much as 100 million LOC. The Boeing 787 flight control software system comprises 6.5 million LOC, about three times as much as the Boeing 777.[2] All kinds of new applications and infrastructures depend heavily on software, including social networking, Internet video, early-warning systems, medical information systems, and financial systems.

This has far-reaching implications for society. Older people feel alienated by devices with poorly designed, nonintuitive, and maddeningly inconsistent human interfaces. A major arms race is taking place between the public, who wish to have secure, reliable general-purpose communications, and the rapidly growing army of spammers, scammers, and other criminals, who have other plans. New devices have shorter lifetimes driven by change and enabled by software—the only engineering technology that can deliver such rapid change and personalization capabilities.

These all have serious business implications. The computer industry changed dramatically and permanently when software took over in the 1980s. Software is profoundly changing the mobile phone industry with the rise of open operating systems such as Winmobile, Symbian, and Android. Such software growth affects healthcare, the automotive industry, and many others.[3] The implications transcend engineering; they affect these industries' value chains and business models.

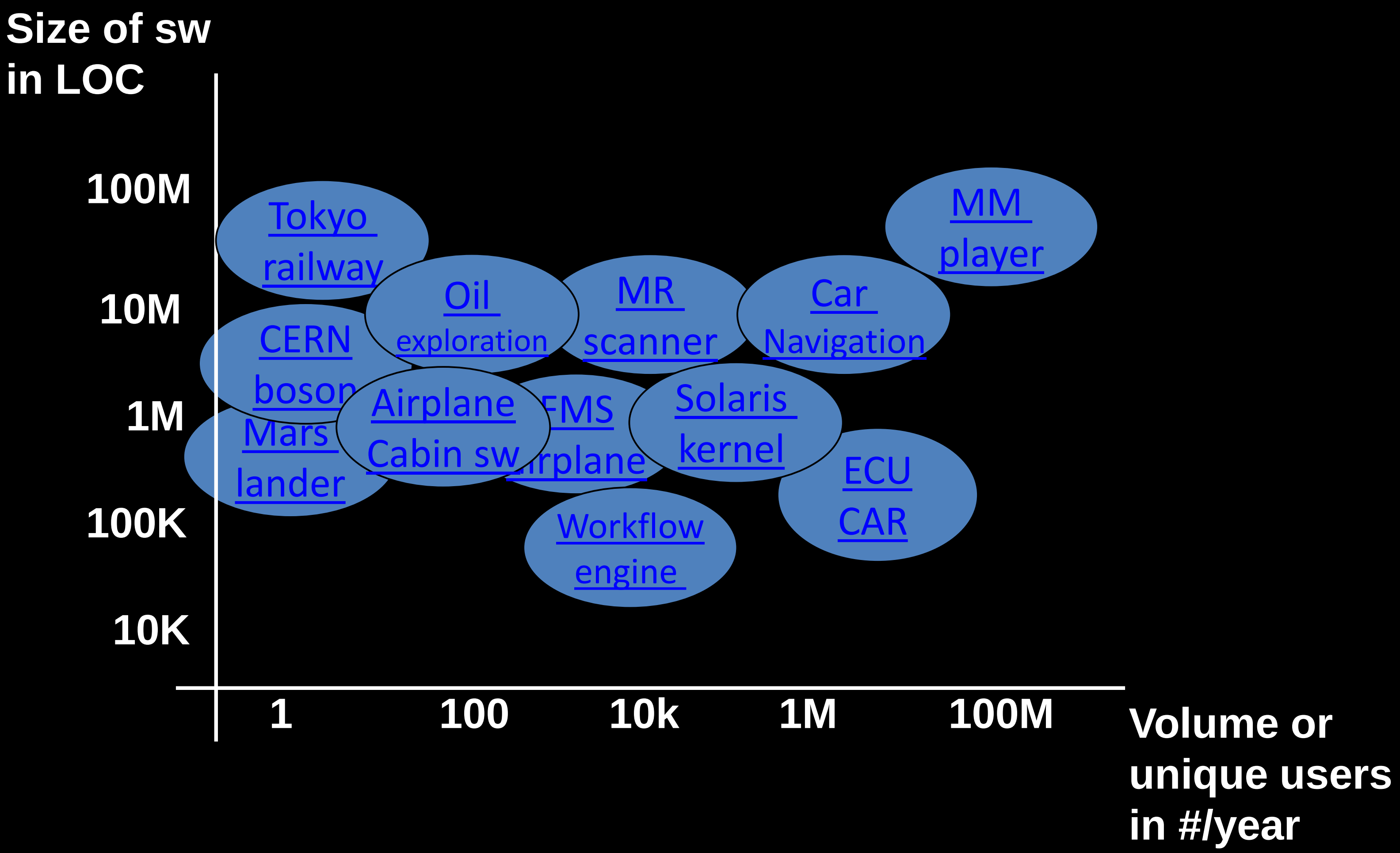
Quantitative Insight

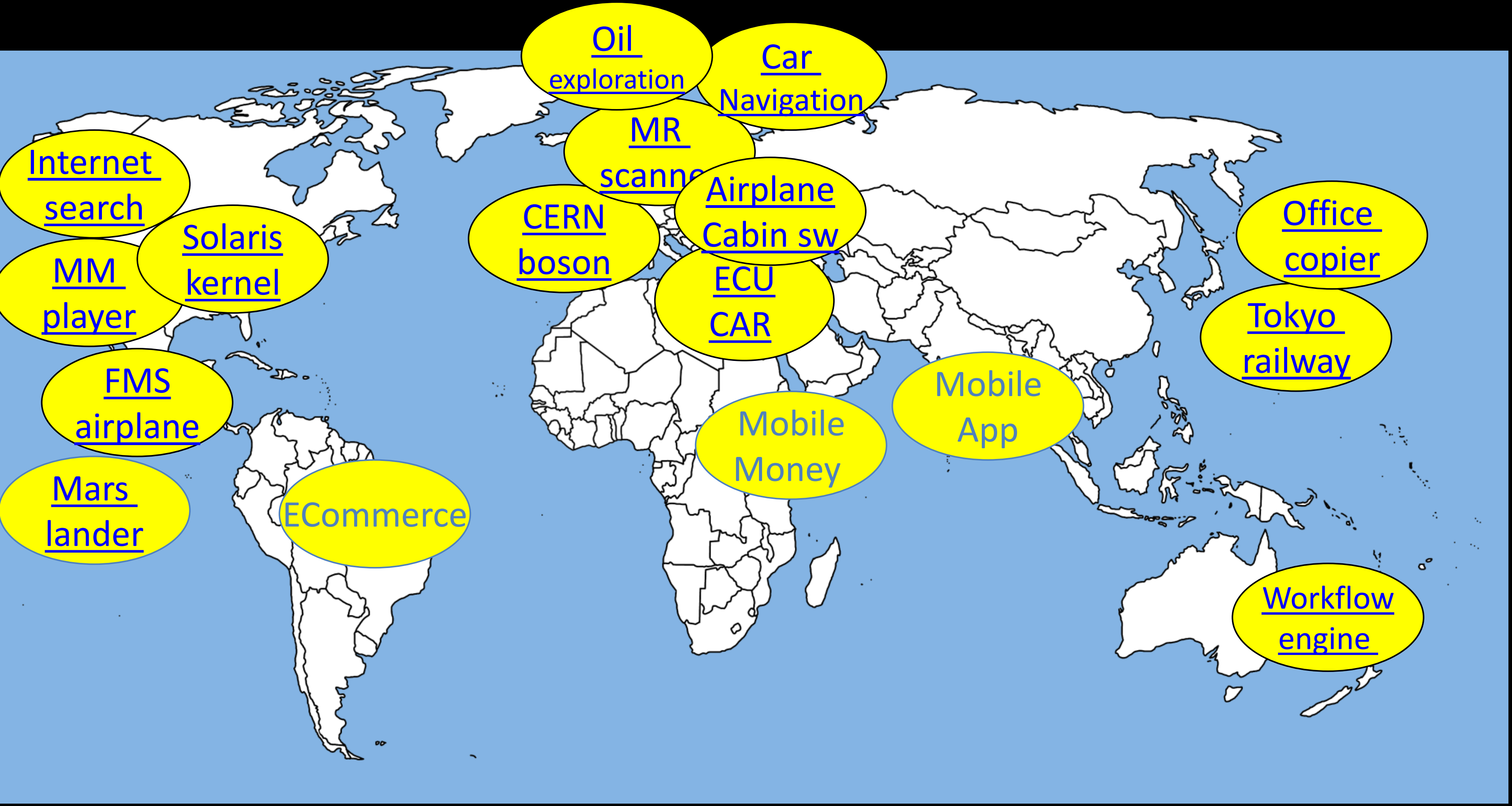
One goal of this column is to build better quantitative insight into how software impacts various businesses. How the product uses the software and how the company built it are equally important.

Software Characterizations Request

For such an apparently sophisticated, complex discipline, measurement in software development is surprisingly imprecise—the phrase "through a glass darkly" comes immediately to mind. This isn't necessarily a drawback, because throughout history engineering methods have always improved in parallel with measurement methods. Engineers use crude measurements to improve a practice, which then outgrows its own measurement system, which then must also improve. The whole

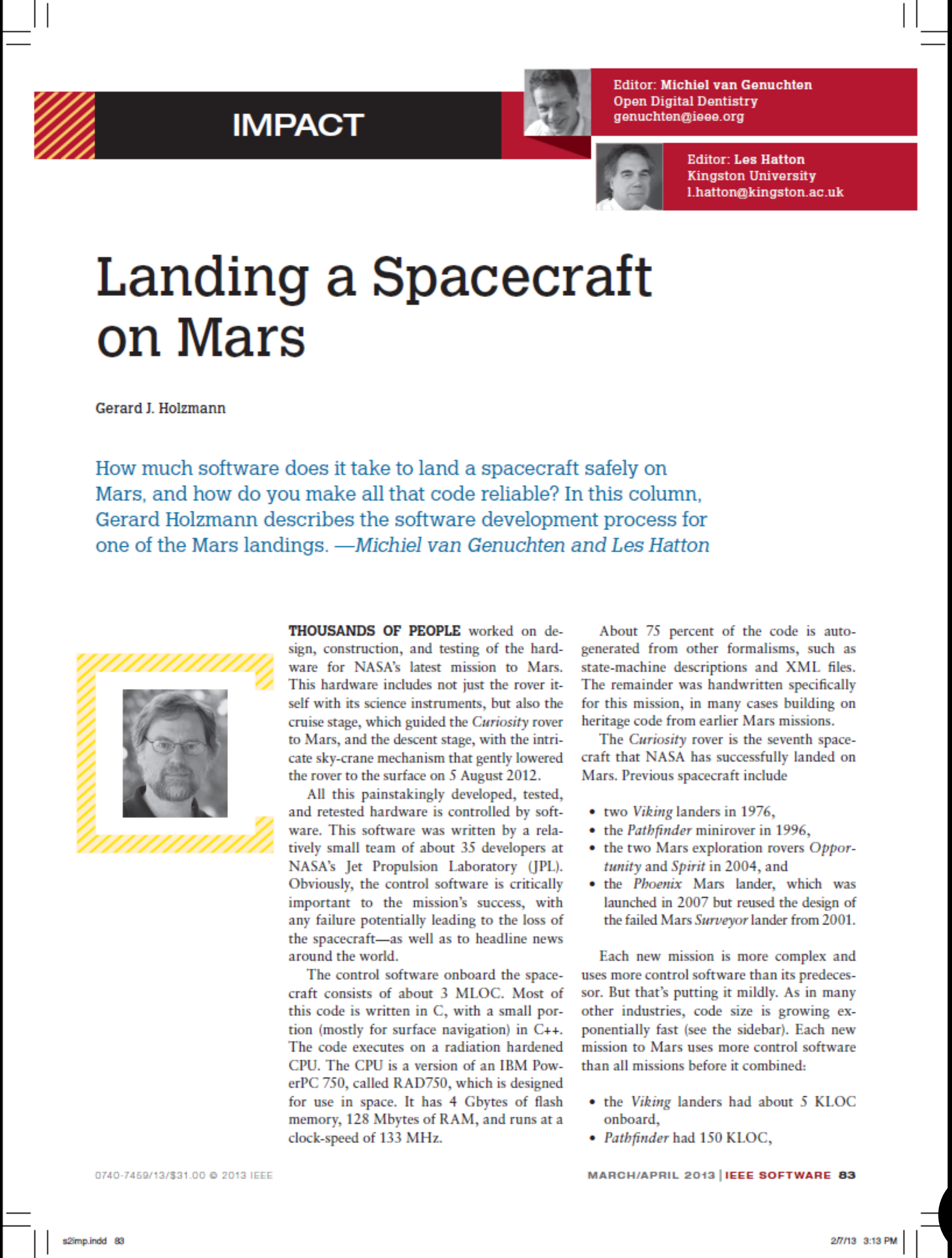
0740-7459/10/$26.00 © 2010 IEEE


Size of sw
in LOC
100M
10M
1M
100K
10K
1
100
10k
1M
100M
Volume or
unique users
in #/year
Tokyo railway
Oil exploration
MR scanner
Car Navigation
MM player
CERN boson
Airplane Cabin sw
FMS Airplane
Solaris kernel
Mars lander
ECU CAR
Workflow engine


Oil exploration
Car Navigation
MR scanne
Internet search
Airplane Cabin sw
CERN boson
Solaris kernel
Office copier
MM player
ECU CAR
Tokyo railway
FMS airplane
Mobile App
Mobile Money
Mars lander
ECommerce
Workflow engine

“To date, no significant anomalies have revealed themselves
in the flight software.”

Team of
35 people



IMPACT

Editor: Michiel van Genuchten
Open Digital Dentistry
genuchten@ieee.org

Editor: Les Hatton
Kingston University
l.hatton@kingston.ac.uk

Landing a Spacecraft on Mars

Gerard J. Holzmann

How much software does it take to land a spacecraft safely on Mars, and how do you make all that code reliable? In this column, Gerard Holzmann describes the software development process for one of the Mars landings. —*Michiel van Genuchten and Les Hatton*

THOUSANDS OF PEOPLE worked on design, construction, and testing of the hardware for NASA’s latest mission to Mars. This hardware includes not just the rover itself with its science instruments, but also the cruise stage, which guided the *Curiosity* rover to Mars, and the descent stage, with the intricate sky-crane mechanism that gently lowered the rover to the surface on 5 August 2012.

All this painstakingly developed, tested, and retested hardware is controlled by software. This software was written by a relatively small team of about 35 developers at NASA’s Jet Propulsion Laboratory (JPL). Obviously, the control software is critically important to the mission’s success, with any failure potentially leading to the loss of the spacecraft—as well as to headline news around the world.

The control software onboard the spacecraft consists of about 3 MLOC. Most of this code is written in C, with a small portion (mostly for surface navigation) in C++. The code executes on a radiation hardened CPU. The CPU is a version of an IBM PowerPC 750, called RAD750, which is designed for use in space. It has 4 Gbytes of flash memory, 128 Mbytes of RAM, and runs at a clock-speed of 133 MHz.

About 75 percent of the code is autogenerated from other formalisms, such as state-machine descriptions and XML files. The remainder was handwritten specifically for this mission, in many cases building on heritage code from earlier Mars missions.

The *Curiosity* rover is the seventh spacecraft that NASA has successfully landed on Mars. Previous spacecraft include

- two *Viking* landers in 1976,
- the *Pathfinder* minirover in 1996,
- the two Mars exploration rovers *Opportunity* and *Spirit* in 2004, and
- the *Phoenix* Mars lander, which was launched in 2007 but reused the design of the failed Mars *Surveyor* lander from 2001.

Each new mission is more complex and uses more control software than its predecessor. But that’s putting it mildly. As in many other industries, code size is growing exponentially fast (see the sidebar). Each new mission to Mars uses more control software than all missions before it combined:

- the *Viking* landers had about 5 KLOC onboard,
- *Pathfinder* had 150 KLOC,

0740-7459/13/$31.00 © 2013 IEEE

"To date, no significant anomalies have revealed themselves
in the flight software."

IMPACT

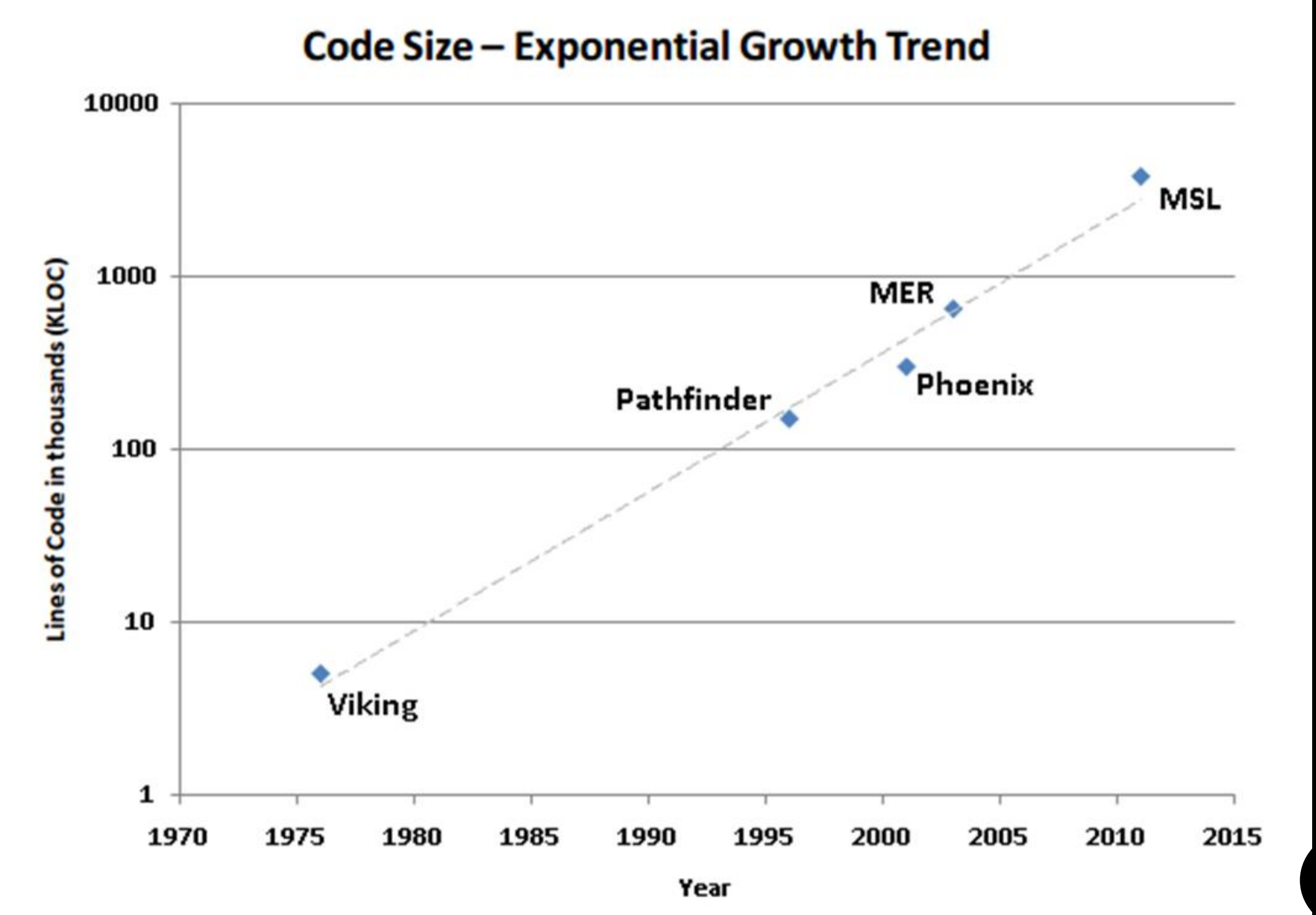
Editor: Michiel van Genuchten
Open Digital Dentistry
genuchten@ieee.org

Editor: Les Hatton
Kingston University
l.hatton@kingston.ac.uk

Landing a Spacecraft on Mars

Gerard J. Holzmann

Team of
35 people



Implications for quality

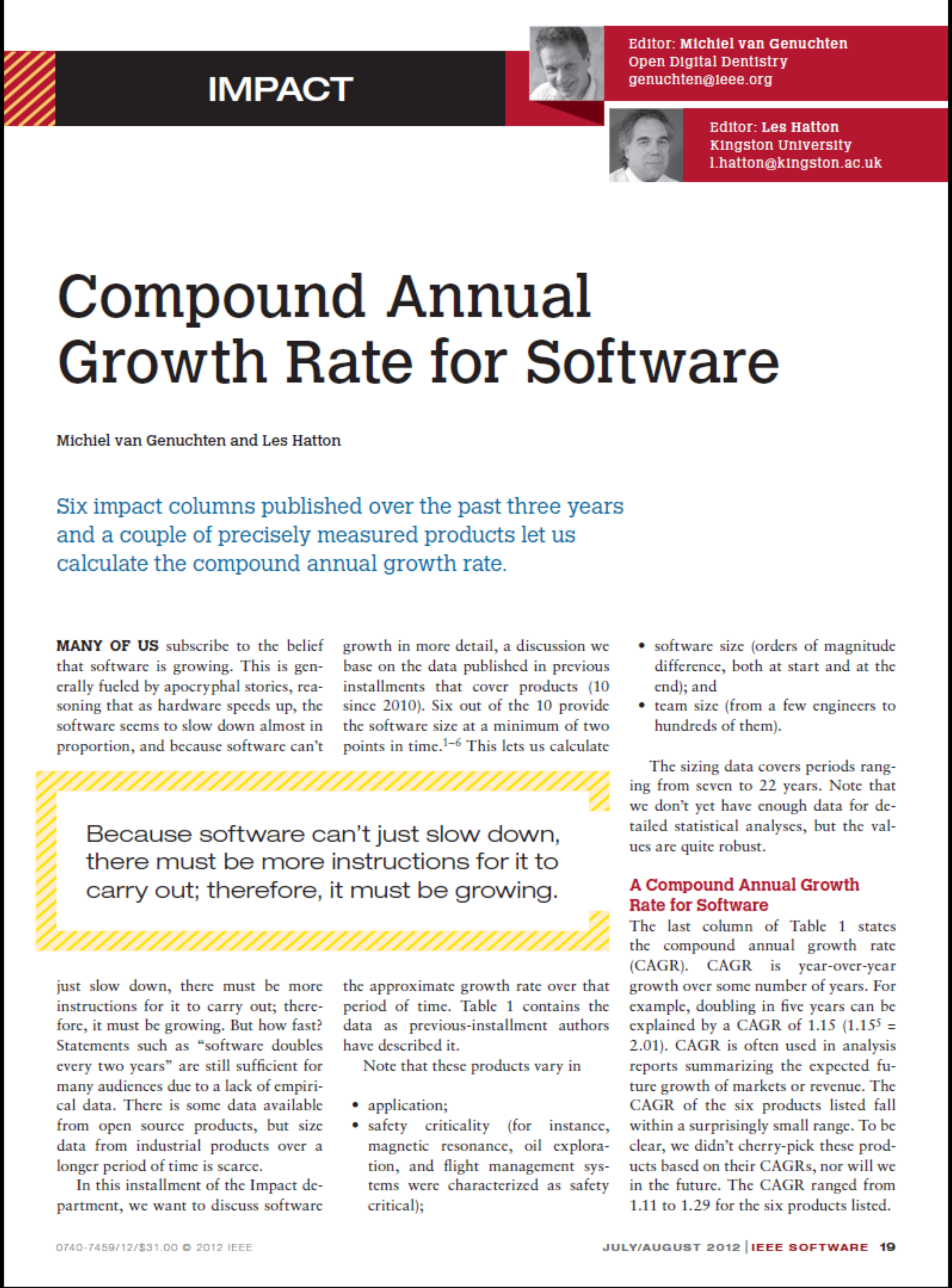
- Rocket science software without ‘anomalies’ is possible
- You do not need more people to do it right first time
- It is all based on proven processes and is hard work

CAGR = $(S/S0)^{1/r}$

Or $(1,15)^5 \sim 2$

Mars lander sw grew with 20 % a year over 36 years

Typical CAGR, validated with dataset of 500MLOC



Editor: Michiel van Genuchten
Open Digital Dentistry
genuchten@ieee.org

Editor: Les Hatton
Kingston University
l.hatton@kingston.ac.uk

Compound Annual Growth Rate for Software

Michiel van Genuchten and Les Hatton

Six impact columns published over the past three years and a couple of precisely measured products let us calculate the compound annual growth rate.

MANY OF US subscribe to the belief that software is growing. This is generally fueled by apocryphal stories, reasoning that as hardware speeds up, the software seems to slow down almost in proportion, and because software can't just slow down, there must be more instructions for it to carry out; therefore, it must be growing. But how fast? Statements such as "software doubles every two years" are still sufficient for many audiences due to a lack of empirical data. There is some data available from open source products, but size data from industrial products over a longer period of time is scarce.

> Because software can't just slow down, there must be more instructions for it to carry out; therefore, it must be growing.

In this installment of the Impact department, we want to discuss software growth in more detail, a discussion we base on the data published in previous installments that cover products (10 since 2010). Six out of the 10 provide the software size at a minimum of two points in time.[1–6] This lets us calculate the approximate growth rate over that period of time. Table 1 contains the data as previous-installment authors have described it.

Note that these products vary in

- application;
- safety criticality (for instance, magnetic resonance, oil exploration, and flight management systems were characterized as safety critical);
- software size (orders of magnitude difference, both at start and at the end); and
- team size (from a few engineers to hundreds of them).

The sizing data covers periods ranging from seven to 22 years. Note that we don't yet have enough data for detailed statistical analyses, but the values are quite robust.

A Compound Annual Growth Rate for Software

The last column of Table 1 states the compound annual growth rate (CAGR). CAGR is year-over-year growth over some number of years. For example, doubling in five years can be explained by a CAGR of 1.15 ($1.15^5 = 2.01$). CAGR is often used in analysis reports summarizing the expected future growth of markets or revenue. The CAGR of the six products listed fall within a surprisingly small range. To be clear, we didn't cherry-pick these products based on their CAGRs, nor will we in the future. The CAGR ranged from 1.11 to 1.29 for the six products listed.

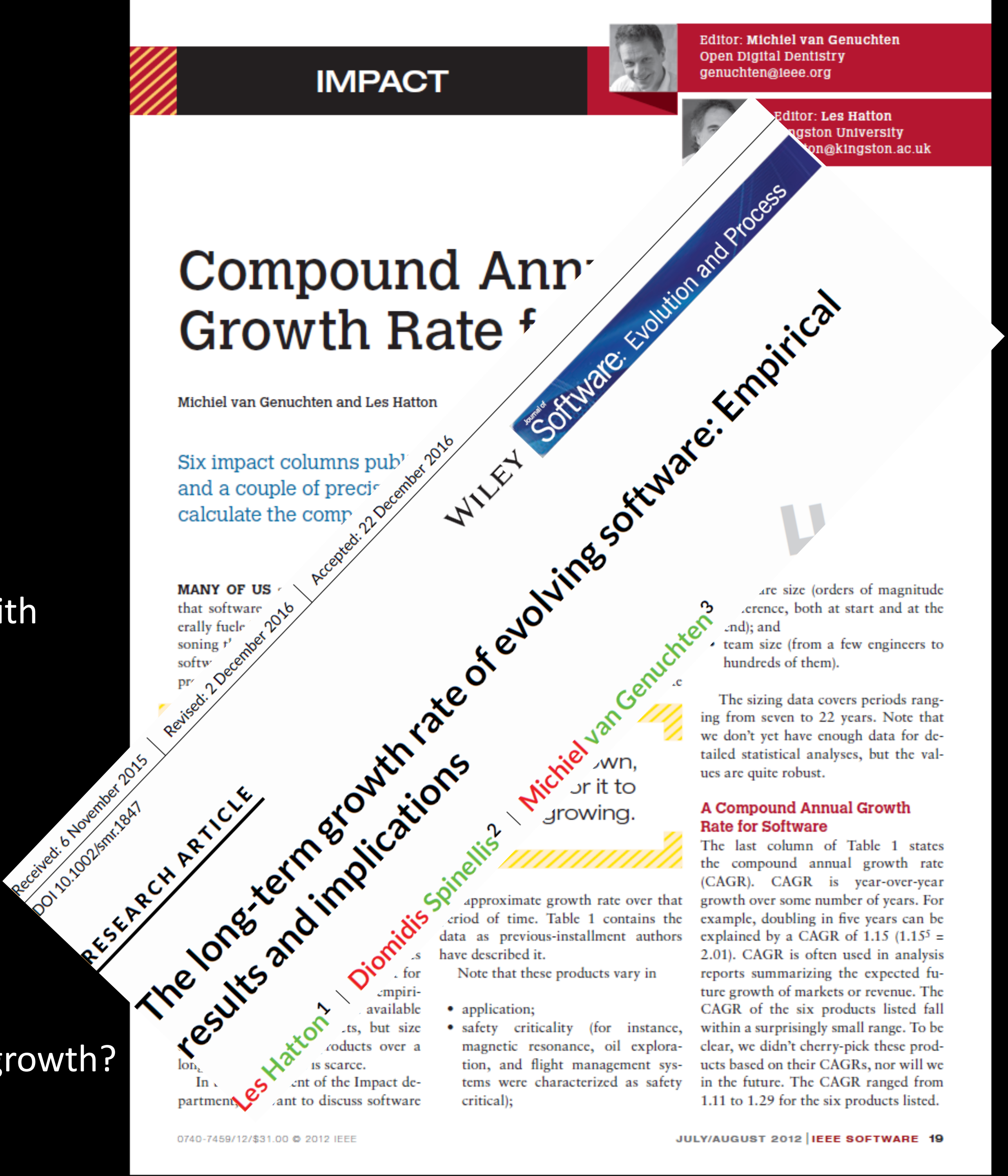
CAGR = $(S/S0)^{1/r}$

Or $(1,15)^5 \sim 2$

Mars lander sw grew with 20 % a year over 36 years

Typical CAGR, validated with dataset of 500MLOC

Have we found a limit to growth?



IMPACT

Editor: Michiel van Genuchten
Open Digital Dentistry
genuchten@ieee.org

Editor: Les Hatton
gston University
on@kingston.ac.uk

Compound Ann
Growth Rate f

Michiel van Genuchten and Les Hatton

Six impact columns publ
and a couple of preci
calculate the comp

MANY OF US
that software
erally fuele
soning t
softw
pr

s
for
empiri-
available
ts, but size
roducts over a
lon
is scarce.
In
ent of the Impact de-
partment
ant to discuss software

own,
or it to
growing.

approximate growth rate over that period of time. Table 1 contains the data as previous-installment authors have described it.

Note that these products vary in

- application;
- safety criticality (for instance, magnetic resonance, oil exploration, and flight management systems were characterized as safety critical);

ure size (orders of magnitude
erence, both at start and at the
nd); and

- team size (from a few engineers to hundreds of them).

The sizing data covers periods ranging from seven to 22 years. Note that we don't yet have enough data for detailed statistical analyses, but the values are quite robust.

A Compound Annual Growth Rate for Software

The last column of Table 1 states the compound annual growth rate (CAGR). CAGR is year-over-year growth over some number of years. For example, doubling in five years can be explained by a CAGR of 1.15 (1.15^5 = 2.01). CAGR is often used in analysis reports summarizing the expected future growth of markets or revenue. The CAGR of the six products listed fall within a surprisingly small range. To be clear, we didn't cherry-pick these products based on their CAGRs, nor will we in the future. The CAGR ranged from 1.11 to 1.29 for the six products listed.

0740-7459/12/$31.00 © 2012 IEEE

JULY/AUGUST 2012 | IEEE SOFTWARE 19

Received: 6 November 2015 | Revised: 2 December 2016 | Accepted: 22 December 2016

DOI 10.1002/smr.1847

WILEY

Journal of Software: Evolution and Process

RESEARCH ARTICLE

The long-term growth rate of evolving software: Empirical results and implications

Les Hatton[1] | Diomidis Spinellis[2] | Michiel van Genuchten[3]

Implications for requirements management

- The estimation problem is solved
- Roadmap can be tested for feasibility
- Quality suffers if one does not care about software size
- Software is not that complex; the real world is
- We can per year only map a limited additional percentage of the real world in sw
  - No matter which tools or programming languages we use
  - No matter how good the requirements are
  - No matter how smart the engineers are
  - No matter how many engineers we add

Editor: **Michiel van Genuchten**
Straumann
genuchten@ieee.org

Editor: **Les Hatton**
Kingston University
l.hatton@kingston.ac.uk

Short and Winding Road: Software in Car Navigation Systems

Han Schaminée and Hans Aerts

Car navigation started as an embedded product in expensive cars. Then it became an independent box, sold aftermarket. The volume that companies like TomTom have been able to accumulate now allows them to pursue the automotive original equipment manufacturers market. —*Michiel van Genuchten and Les Hatton*

TOMTOM IS KNOWN for *aftermarket* navigation systems—separate devices sold independently of the car. In April 2009, TomTom launched its first in-dash navigation system (built into the dashboard by the car manufacturer at the factory). The system is upgraded annually, allowing the introduction of new features and innovations over its lifetime. This column describes the product, its place in the automotive industry, and the development of its software.

A Short History of Car Navigation

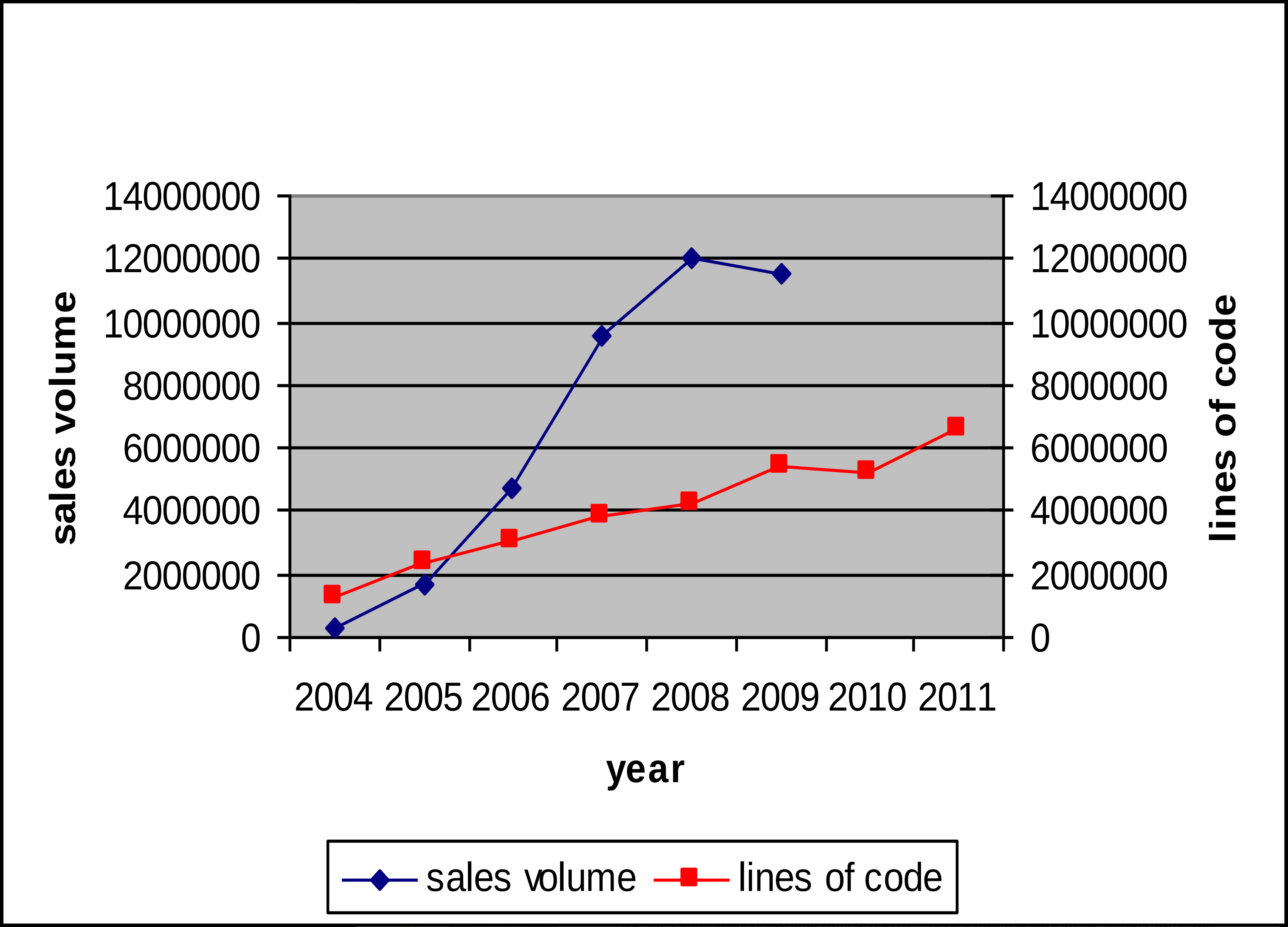
Car navigation started as a feature of expensive cars in the 1980s. The systems didn't use GPS and each car manufacturer's system was unique. It became a separate aftermarket device about 10 years ago, when the US made a more accurate GPS signal available for civilian use, thus making navigation less dependent on the vehicle itself. Today, TomTom's sales exceed 10 million devices per year, which is more than the two to nine million cars that the largest car manufacturers sell each year. This lets TomTom invest more in car navigation than the typical car original equipment manufacturer (OEM). Thus, TomTom can introduce new functionalities faster when the OEMs agree to use the synergy of aftermarket products.

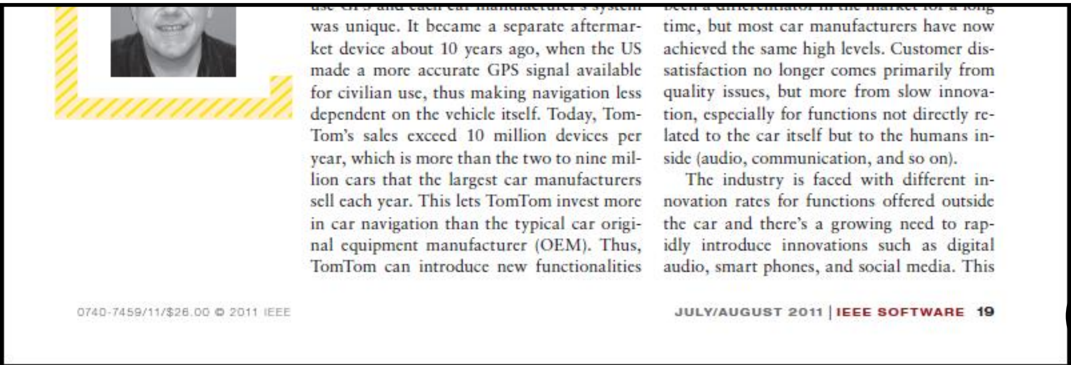
Handling Different Innovation Speeds

Over the last two decades, the automotive industry has gone through an impressive turnaround to ensure substantially higher quality levels. An example is the requirement to have less than a 0.01 percent failure rate at the car manufacturing line. A car consists of a large number of subassemblies from many different suppliers. The industry introduced rigid processes for architecture, validation, and supplier management to realize this breakthrough in quality. Better quality had been a differentiator in the market for a long time, but most car manufacturers have now achieved the same high levels. Customer dissatisfaction no longer comes primarily from quality issues, but more from slow innovation, especially for functions not directly related to the car itself but to the humans inside (audio, communication, and so on).

The industry is faced with different innovation rates for functions offered outside the car and there's a growing need to rapidly introduce innovations such as digital audio, smart phones, and social media. This

0740-7459/11/$26.00 © 2011 IEEE





use GPS and each car manufacturer's system was unique. It became a separate aftermarket device about 10 years ago, when the US made a more accurate GPS signal available for civilian use, thus making navigation less dependent on the vehicle itself. Today, TomTom's sales exceed 10 million devices per year, which is more than the two to nine million cars that the largest car manufacturers sell each year. This lets TomTom invest more in car navigation than the typical car original equipment manufacturer (OEM). Thus, TomTom can introduce new functionalities

been a differentiator in the market for a long time, but most car manufacturers have now achieved the same high levels. Customer dissatisfaction no longer comes primarily from quality issues, but more from slow innovation, especially for functions not directly related to the car itself but to the humans inside (audio, communication, and so on).

The industry is faced with different innovation rates for functions offered outside the car and there's a growing need to rapidly introduce innovations such as digital audio, smart phones, and social media. This

Software economics provides limits to growth

- Do not bother if your sw is not among market leaders
  - Markets can be defined in different ways
  - But many sw markets develop to global markets
  - Top 3 may have a future in a mature market
  - Embedded software has limited value
- Revenue should grow fast
  - If it costs nothing it is worth nothing
  - Where can the invoice be sent?
  - 30 percent revenue growth per year could be the minimum

Editor: Michiel van Genuchten
VitalHealth Software
genuchten@ieee.org

Editor: Les Hatton
Oakwood Computing Associates
leah@oakcomp.co.uk

No Free Lunch for Software after All

Anne-Francoise Rutkowski, Michiel van Genuchten, and Les Hatton

SOFTWARE HAS BEEN challenging the laws of economics for many years because its reproduction costs are zero. Building it the first time has always been a challenge, but afterward, copying has been all that's needed, and the sky has been the limit. It looked as if free lunches existed after all, as long as they were made of software and they got over the hurdle of an acceptable-quality first version. So, relatively small companies could serve large markets, and companies grew from small to huge in just a few years.

The Impact department, which has run since 2010 in *IEEE Software*, shows that software is ubiquitous and growing[1] and that the best opportunities arise for the companies with the most customers for the amount of software they write.[2] The more is indeed the merrier. However, this otherwise upbeat process has significant side effects.

The Economics of Software Also Benefits Criminals

As with medicines, users might think it appropriate for some software to describe the negative side effects, such as, "This software will thoroughly irritate you with its unnecessary defects," "Did we mention privacy is a thing of the past?," or "By installing our software, you might leave your system vulnerable to criminals who might steal your data, your money, or both—we don't know." In other words, zero reproduction costs cut both ways. The benefits that legitimate developers enjoy are exactly the same for people who want to use software for criminal purposes. Software can easily be turned into a weapon of mass deceit, as has been proven by spammers, phishers, and, recently, an automobile company.[3]

So, a series of battlegrounds has developed. In some cases, insufficient security and inadequate quality have turned users away from otherwise useful applications. For example, the Dutch elections are using paper and pen again after decades of electronic voting.[4] The responsible politicians concluded that the computer software was vulnerable and that going back to paper would thwart any cyberhacking bid. (Of course, some countries, such as the UK, never left paper-and-pen voting.)

However, software doesn't exist in a vacuum. As any software engineer knows, it's intrinsically linked with the hardware it runs on and the architectural design within which it sits. But users don't necessarily react just to the software; they react to the system as a whole. One consequence of this is that to users, the mantra is "computer failure." But to the software engineer, this

0740-7459/17/$33.00 © 2017 IEEE

As with medicines, software should describe the negative side effects

- “This software will thoroughly irritate you with its unnecessary defects”
- “Did we mention privacy is a thing of the past?”
- “By installing our software, you might leave your system vulnerable to criminals who might steal your data, your money, or both—we don’t know.”

- Zero reproduction costs cut both ways. Criminals also benefit

Requirement of the year: GDPR

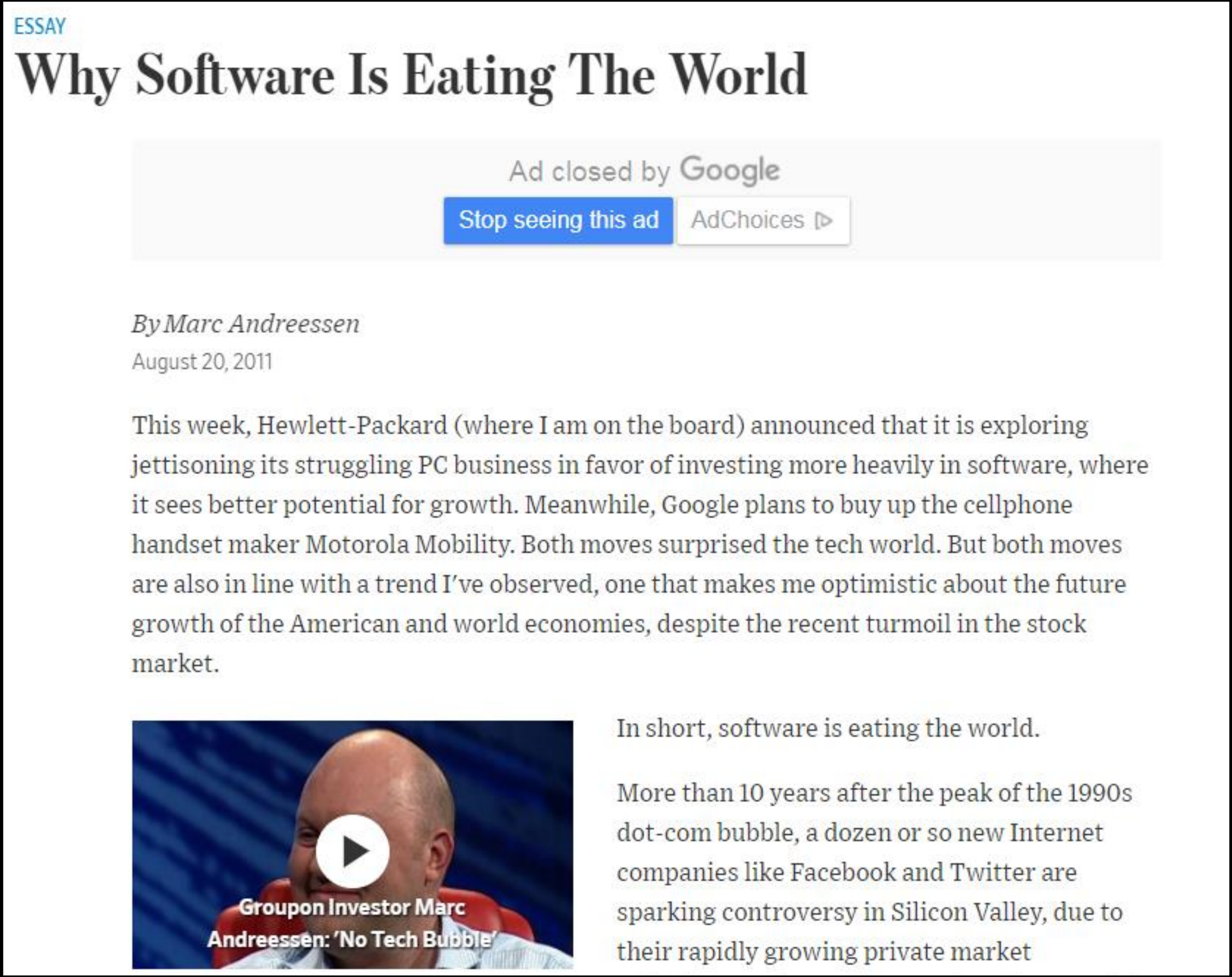
EU: General Data Policy Regulation

- Increased Territorial Scope (extra-territorial applicability)
- Penalties
- Consent
- Breach Notification
- Right to Access
- Right to be forgotten
- Data Portability
- Privacy by Design
- Data Protection Officers

See https://www.eugdpr.org/key-changes.html

Users turning away limits growth

- Defects lead to vulnerabilities which enables hacks
- Elections back to paper and less traffic on social media
- Building stuff that may lack a foundation
  - False negatives and software based recommendations
  - Autonomous driving and ‘pedestrian crossing street’



ESSAY

Why Software Is Eating The World

By Marc Andreessen

August 20, 2011

This week, Hewlett-Packard (where I am on the board) announced that it is exploring jettisoning its struggling PC business in favor of investing more heavily in software, where it sees better potential for growth. Meanwhile, Google plans to buy up the cellphone handset maker Motorola Mobility. Both moves surprised the tech world. But both moves are also in line with a trend I've observed, one that makes me optimistic about the future growth of the American and world economies, despite the recent turmoil in the stock market.

In short, software is eating the world.

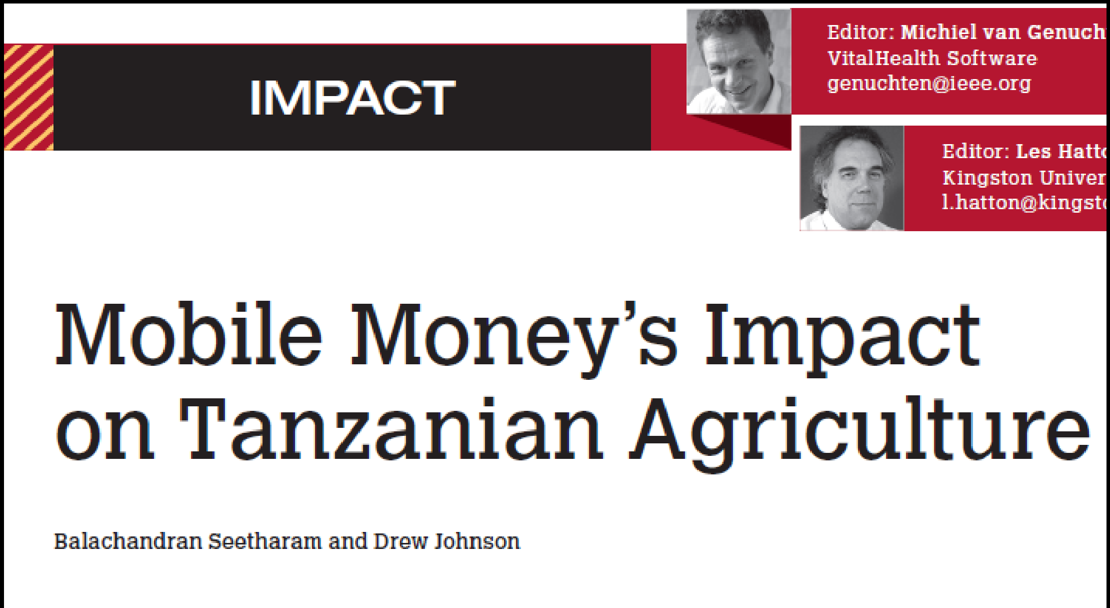
More than 10 years after the peak of the 1990s dot-com bubble, a dozen or so new Internet companies like Facebook and Twitter are sparking controversy in Silicon Valley, due to their rapidly growing private market

2018-03-26

For those who are eating the world

- Among the industries to be eaten
  - Healthcare
  - Automotive
- Market cap of large companies
  - Mobileye versus Opel (15B$ vs 4B$ in 2017)
  - Apple versus Healthineers (900B$ vs 31B$)
  - Google versus BMW (788B$ vs 67B$)
- Remember Motorola and Nokia?
- The legal structure is required to catch up to the digital age
  - Privacy laws like GDPR protect citizens
  - Antitrust law does not allow expanding monopoly to new markets

Example of the positive impact of sw



IMPACT

Editor: Michiel van Genuch
VitalHealth Software
genuchten@ieee.org

Editor: Les Hatto
Kingston Univer
l.hatton@kingsto

Mobile Money's Impact on Tanzanian Agriculture

Balachandran Seetharam and Drew Johnson

The 25th Impact column comes from Africa and s
how software enables the use of mobile money b
farmers and has significantly benefited agricultu
Tanzania. —*Michiel van Genuchten and Les Hatt*

TANZANIA, with a population of 48 million, has one of the fastest-growing economies in the world. Situated on Africa's east coast, the country has sustained an annual growth in GDP (gross domestic product) of 5 to 8 percent since 2000.[1] The World Bank predicts a continued annual GDP growth of 7 percent.[2] Agriculture, the backbone of the Tanzanian economy, constitutes 27 percent of the GDP and employs 80 percent of the

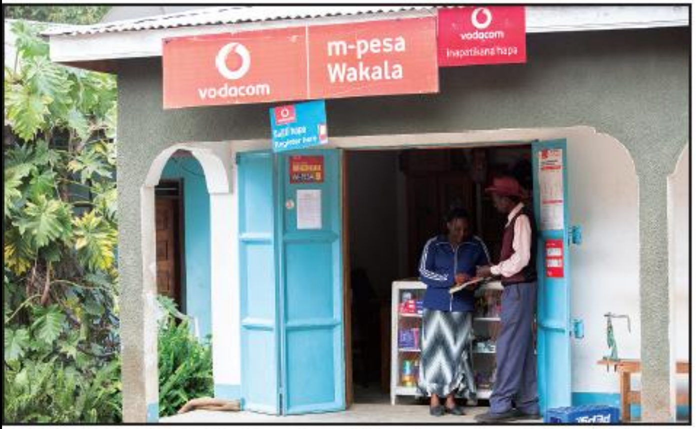
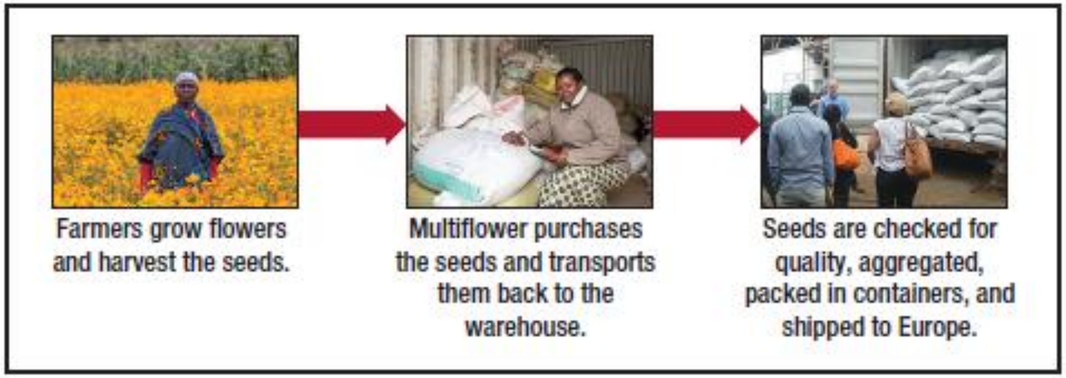


FIGURE 1. A typical kiosk in Tanzania at which customers can withdraw and deposit money from digital wallets, using their mobile phones.



Farmers grow flowers and harvest the seeds.

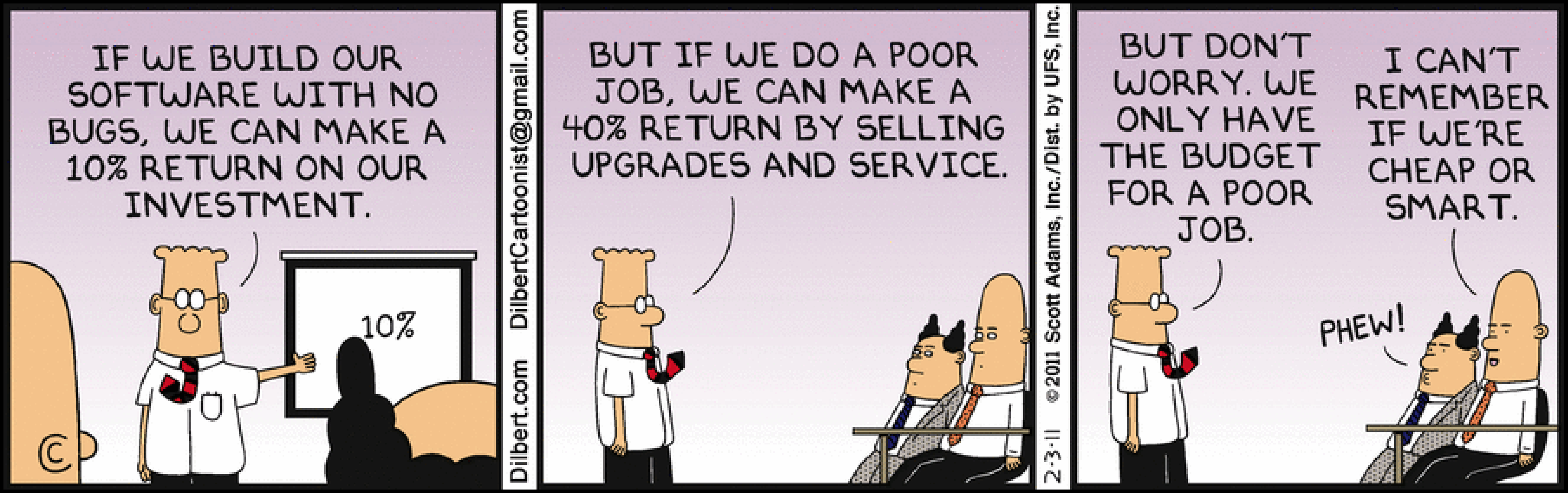
Multiflower purchases the seeds and transports them back to the warehouse.

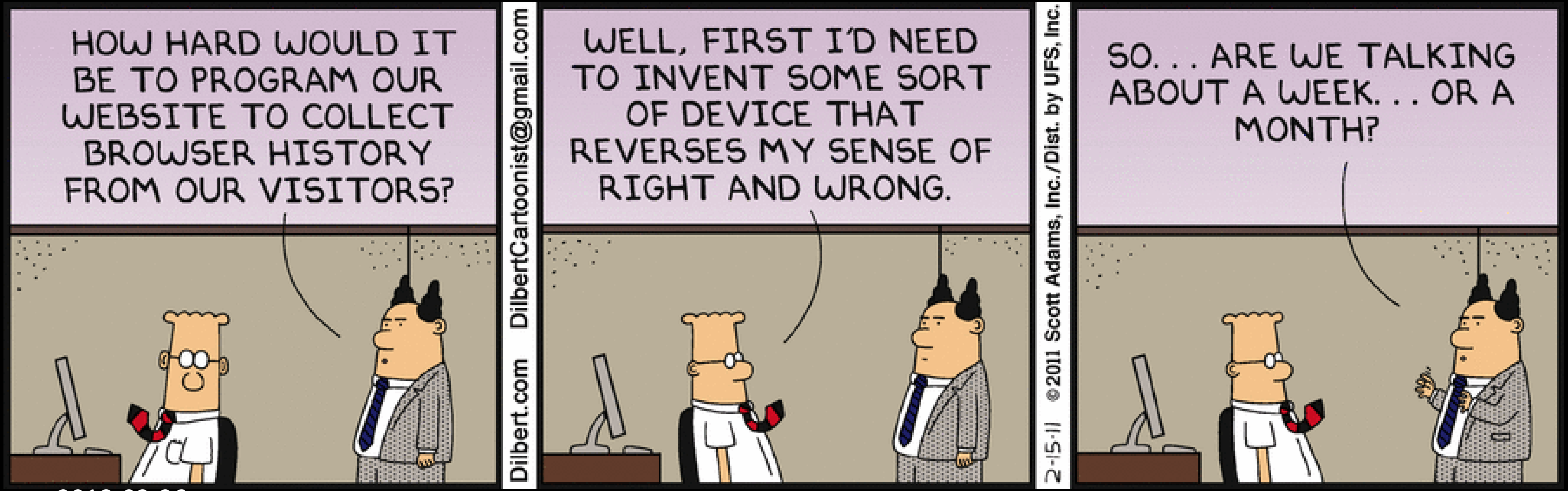
Seeds are checked for quality, aggregated, packed in containers, and shipped to Europe.

FIGURE 2. The agriculture workflow of Multiflower, an agricultural exporter headquartered in Arusha, Tanzania.

Conclusions

- Software has quite an impact
- There are limits to growth
- Important future requirements are:
  - Economical
  - Legal
  - Ethical


IF WE BUILD OUR SOFTWARE WITH NO BUGS, WE CAN MAKE A 10% RETURN ON OUR INVESTMENT.
10%
Dilbert.com DilbertCartoonist@gmail.com
BUT IF WE DO A POOR JOB, WE CAN MAKE A 40% RETURN BY SELLING UPGRADES AND SERVICE.
2-3-11 ©2011 Scott Adams, Inc./Dist. by UFS, Inc.
BUT DON'T WORRY. WE ONLY HAVE THE BUDGET FOR A POOR JOB.
I CAN'T REMEMBER IF WE'RE CHEAP OR SMART.
PHEW!


HOW HARD WOULD IT BE TO PROGRAM OUR WEBSITE TO COLLECT BROWSER HISTORY FROM OUR VISITORS?
Dilbert.com DilbertCartoonist@gmail.com
WELL, FIRST I'D NEED TO INVENT SOME SORT OF DEVICE THAT REVERSES MY SENSE OF RIGHT AND WRONG.
2-15-11 ©2011 Scott Adams, Inc./Dist. by UFS, Inc.
SO. . . ARE WE TALKING ABOUT A WEEK. . . OR A MONTH?

2018-03-26