JUNE 2018

www.computer.org

IEEE

IEEE computer society

IEEE COMPUTER SOCIETY: Be at the Center of It All

IEEE Computer Society membership puts you at the heart of the technology profession—and helps you grow with it.

Here are 10 reasons why you need to belong.

IEEE Computer Society—keeping you ahead of the game. Get involved today.

www.computer.org/membership

IEEE computer society

COMPUTING edge

IEEE COMPUTER SOCIETY computer.org • +1 714 821 8380

Printed with inks containing soy and/or vegetable oils

IEEE computer society

STAFF

Circulation: ComputingEdge (ISSN 2469-7087) is published monthly by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; voice +1 714 821 8380; fax +1 714 821 4010; IEEE Computer Society Headquarters, 2001 L Street NW, Suite 700, Washington, DC 20036.

Postmaster: Send address changes to ComputingEdge-IEEE Membership Processing Dept., 445 Hoes Lane, Piscataway, NJ 08855. Periodicals Postage Paid at New York, New York, and at additional mailing offices. Printed in USA.

IEEE Computer Society Magazine Editors in Chief

Computer
Sumi Helal, *Lancaster University*

IEEE Software
Diomidis Spinellis, *Athens University of Economics and Business*

IEEE Internet Computing
M. Brian Blake, *Drexel University*

IT Professional
Irena Bojanova, *NIST*

IEEE Security & Privacy
David M. Nicol, *University of Illinois at Urbana-Champaign*

IEEE Micro
Lieven Eeckhout, *Ghent University*

IEEE Computer Graphics and Applications
Torsten Möller, *University of Vienna*

IEEE Pervasive Computing
Marc Langheinrich, *Università della Svizzera Italiana*

Computing in Science & Engineering
Jim X. Chen, *George Mason University*

IEEE Intelligent Systems
V.S. Subrahmanian, *Dartmouth College*

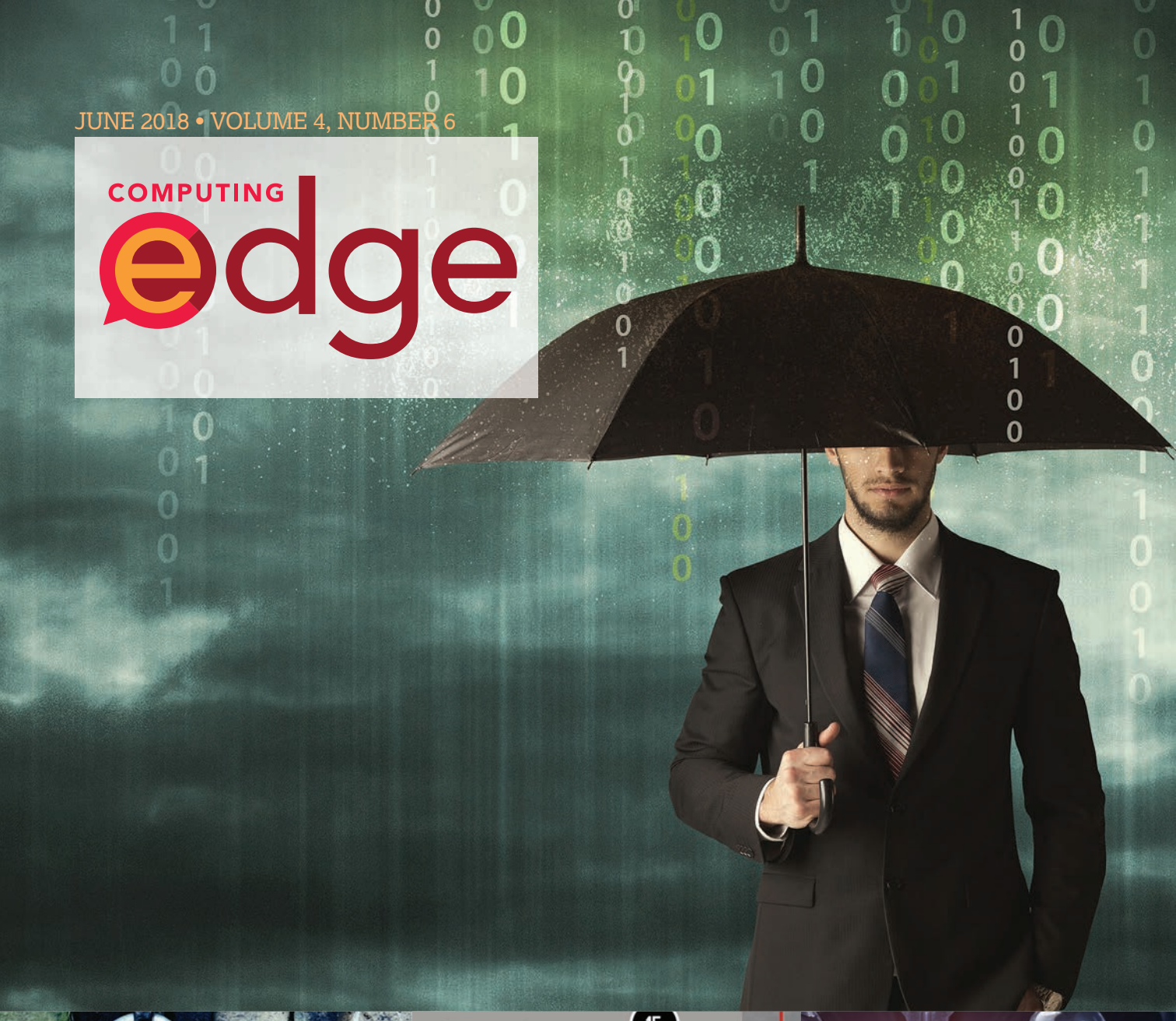
IEEE MultiMedia
Shu-Ching Chen, *Florida International University*

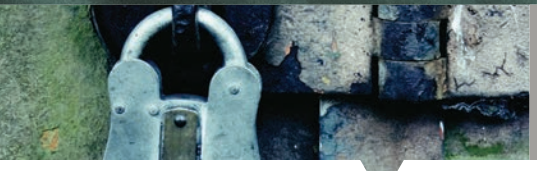
IEEE Annals of the History of Computing
Nathan Ensmenger, *Indiana University Bloomington*

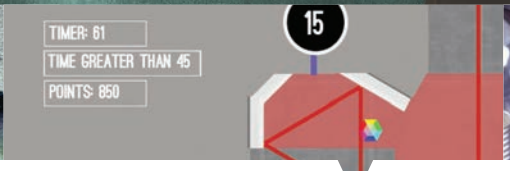
IEEE Cloud Computing
Mazin Yousif, *T-Systems International*

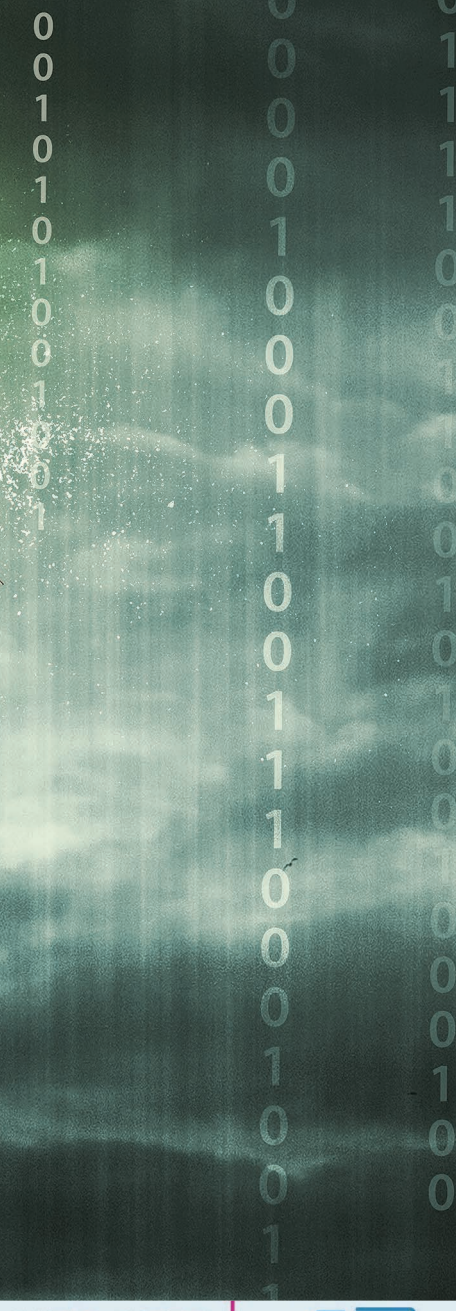
JUNE 2018 • VOLUME 4, NUMBER 6

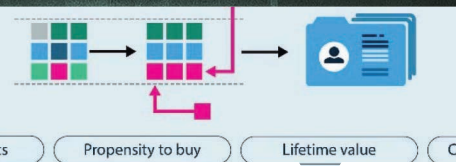
COMPUTING edge





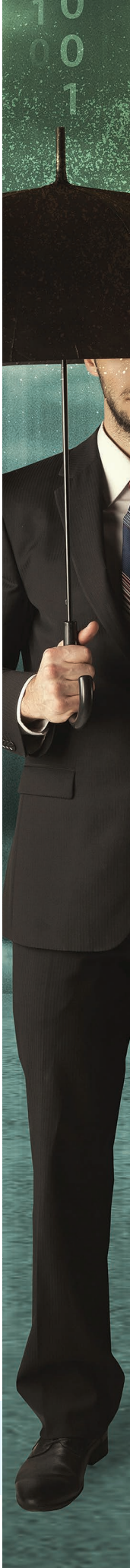






50

The Role of a Customer Data Platform



Magazine Roundup

Editor: Lori Cameron

The IEEE Computer Society's lineup of 13 peer-reviewed technical magazines covers cutting-edge topics ranging from software design and computer graphics to Internet computing and security, from scientific applications and machine intelligence to cloud migration and microchip design. Here are highlights from recent issues.

Computer

International Neuroscience Initiatives through the Lens of High-Performance Computing

Neuroscience initiatives aim to develop new technologies and tools to measure and manipulate neuronal circuits. To deal with the massive amounts of data generated by these tools, the authors of this article from the April 2018 issue of *Computer* envision the co-location of open data repositories in standardized formats together with high-performance computing hardware utilizing open source optimized analysis codes.

Computing in Science & Engineering

A Cyberplatform for Sharing Scientific Research Data at DataCenterHub

In this article from the May/June 2018 issue of *Computing in Science & Engineering*, the authors introduce DataCenterHub, a new solution for preserving, sharing, and discovering data produced by scientific research. Datasets are organized by experiments, with a simple common structure for

 Published by the IEEE Computer Society

metadata, file collections, and key parameters. Researchers associate annotations, reports, media, and measurements to each experiment, and interactive viewers interpret data by type and use so that they can be investigated before downloading. Parameters are extracted for discovery of key data otherwise hidden in files. DataCenterHub provides an alternative discipline-neutral solution, with the goal of helping researchers classify and share data for easy discovery and exploration.

IEEE Annals of the History of Computing

The Origins of the Architectural Metaphor in Computing: Design and Technology at IBM, 1957–1964

According to author David Halsted, the term "computer architecture" seems to have appeared first at IBM in the late 1950s and early 1960s, and then spread to establish itself within the computing community. What impact did the architectural metaphor have on computing during this early period? In this study from the January–March 2018 issue of *IEEE Annals of the History of Computing*, Halsted argues that computer architecture represents more than a simple terminological transfer from one field of practice to another. When the architectural metaphor entered the computing domain, it changed in ways that are specific to computing machinery. Computing absorbed the architectural metaphor into its practice and made (a subset of) architectural concepts its own.

IEEE Cloud Computing

Privacy-Preserving Image Processing in the Cloud

Millions of private images are generated in various digital devices every day. The consequent massive computational workload means more people are turning to cloud computing platforms for their economical computation resources. Meanwhile, privacy concerns arise over the sensitive information contained in outsourced image data. Lack of security and privacy guarantees becomes the main barrier to further deployment of cloud-based image processing systems. This article from the March/April 2018 issue of *IEEE Cloud Computing* studies the design targets and technical challenges in constructing cloud-based privacy-preserving image processing systems. The authors explore various image processing tasks, including image feature detection, digital watermarking, and content-based image search.

IEEE Computer Graphics and Applications

Toward Visual Avatars that Dress You Well and Impact Your Health

Imagine a future where you walk up to your wardrobe and a magic mirror displays your avatar, to whom you can talk about your planned activities for the day. You get visual suggestions about what to wear, taking all sorts of constraints into account, such as the clothes in your closet, weather conditions, and so on. Seeing your avatar in the suggested clothes could help you decide quickly what to wear, keep you warm or cool, and contribute to your health. Read more in this article from the March/April 2018 issue of *IEEE Computer Graphics and Applications*.

IEEE Intelligent Systems

Investigative Knowledge Discovery for Combating Illicit Activities

Developing scalable, semi-automatic approaches to derive insights from a domain-specific Web corpus is a longstanding research problem in the knowledge discovery community. The problem is particularly challenging in illicit fields, such as human trafficking, where traditional assumptions concerning information representation are frequently violated. In this article from the January/February 2018 issue of *IEEE Intelligent Systems*, the authors describe an end-to-end investigative knowledge discovery system for illicit Web domains. They built and evaluated a prototype involving separate components for information extraction, semantic modeling, and query execution on a real-world human trafficking Web corpus containing 1.3 million pages, with promising results.

IEEE Internet Computing

Social Computing for Verifying Social Media Content in Breaking News

Social media is the place to go for both journalists and the general public when news events

break, offering a real-time source of eyewitness images and videos through platforms like YouTube, Instagram, and Periscope. Yet, the value of such content as a means of documenting and disseminating breaking news is compromised by the increasing amount of content misuse and false claims in social media. The authors of this article, featured in the March/April 2018 issue of *IEEE Internet Computing,* discuss cost-effective social computing solutions for user-generated content verification, which are crucial for retaining the value of and trust in social media for breaking news.

IEEE Micro

A Microarchitecture for a Superconducting Quantum Processor

The authors of this article from the May/June 2018 issue of *IEEE Micro* propose a quantum microarchitecture (QuMA). Flexible programmability of QuMA is achieved by multilevel instructions decoding, abstracting analog control into digital control, and translating instruction execution with nondeterministic timing into event trigger with precise timing. QuMA is validated by several single-qubit experiments on a superconducting qubit.

IEEE MultiMedia

Sensing Technologies for Monitoring Serious Mental Illnesses

Around 450 million people worldwide suffer from serious mental illnesses, which results in devastating personal outcomes and huge societal burden. Effective symptom monitoring and personalized interventions can significantly improve mental health care across different populations. However, traditional clinical methods often fall short when it comes to real-time monitoring of symptoms. Sensing technologies can address these issues by enabling granular tracking of behavioral, physiological, and social signals relevant to mental health. This article from the January–March 2018 issue of *IEEE MultiMedia* describes how sensing technologies can be used to diagnose and monitor patient states for numerous serious mental illnesses, and how the multimedia community can build on these technologies to enable efficient clinical decision making in mental health care.

IEEE Pervasive Computing

A Public Transit Assistant for Blind Bus Passengers

Public transit is the key to independence for many blind persons. Despite recent progress in assistive technology, public transit remains challenging for those without sight. To this end, the authors of this article from the January–March 2018 issue of *IEEE Pervasive Computing* developed a prototype mobile application that communicates information via Wi-Fi access points installed in buses and at bus stops to help blind bus passengers reach their destination. A user study of the system yielded insights into general accessibility issues for blind public transit riders as well as ways to improve the proposed system.

IEEE Security & Privacy

Privacy-Aware Restricted Areas for Unmanned Aerial Systems

Although drones are receiving a lot of attention from industry and academia alike, the protection of citizen privacy is still an open issue. The authors of this article from the March/April 2018 issue of *IEEE Security & Privacy* demonstrate how basic principles of information privacy could be integrated with existing infrastructure to build a framework for the dispatch of privacy-aware unmanned aerial systems (UASs). The proposed framework enables UAS operators to determine whether a selected flight path intersects with a restricted area by considering privacy preferences that can be configured by citizens.

IEEE Software

From Monolithic to Microservices: An Experience Report from the Banking Domain

Microservices have seen their popularity blossoming with an explosion of concrete applications in real-life software. Several companies are currently involved in a major refactoring of their backend systems to improve scalability. This article from the May/June 2018 issue of *IEEE Software* presents an experience report of a real-world case study from the banking

domain to demonstrate how scalability is positively affected by re-implementing a monolithic architecture into microservices. The case study is based on the FX Core system—a mission-critical system of Danske Bank, the largest bank in Denmark and one of the leading financial institutions in Northern Europe—for converting one currency to another.

IT Professional

To Blockchain or Not to Blockchain: That Is the Question

Blockchain has been considered a breakthrough technology—but does your company need it? In this article from the March/April 2018 issue of *IT Professional*, the authors discuss the advantages and disadvantages of blockchain technology using examples from the insurance sector, which can be generalized and applied to other sectors.

Computing Now

The Computing Now website (computingnow.computer.org) features up-to-the-minute computing news and blogs, along with articles ranging from peer-reviewed research to opinion pieces by industry leaders.

Software, Software Everywhere

Software is all around us—it's in our smartphones, our cars, our TVs, our fitness trackers, and it's increasingly making up the critical infrastructure that supports hospitals, transportation, finance, government, and more. Systems that were once controlled by humans or machines are now depending on code. When the functionality of these systems has life or death implications, security is paramount. In this issue of *ComputingEdge*, author Gary McGraw shares the importance of protecting your software by developing and maintaining an inventory of all the software your organization uses in his *Computer* article "The New Killer App for Security: Software Inventory."

Software is also essential to most research today—a 2014 study showed that approximately 90 percent of academics surveyed said they use software in their research. The authors of *Computing in Science & Engineering*'s "Publish Your Software: Introducing the Journal of Open Source Software (JOSS)" present a publication that focuses on research software and its place in the scholarly publishing ecosystem.

Wearables are also powered by software. In *IEEE Pervasive Computing*'s "How Wearable Computing Is Shaping Digital Health," the author discusses how wearables enable personalized healthcare through a distributed information-sharing model that puts patients and users at the center, rather than providers and insurers. However, realizing the promise of wearables and digital health will require multiple parallel technological advances. Also from *IEEE Pervasive Computing*, "What Will We Wear After Smartphones?" looks at how the wearable field developed and explores where it's headed. According to the authors, wearable computing is entering its most exciting phase yet, as it transitions from demonstrations to the creation of sustained markets and industries, which should drive future research and innovation.

Another area that depends on software is gaming. The author of *IEEE Software*'s "Practices and Technologies in Computer Game Software Engineering" says that game developers from all fields (including game-based applications in entertainment, education, healthcare, defense, and scientific

 Published by the IEEE Computer Society 2469-7087/18/$33.00

research) share a common interest in how best to engineering game software, and examines techniques and technologies that inform contemporary computer game software engineering.

Robotics is an important emerging field where software is a key player. In "Emerging White-Collar Robotics: The Case of Watson Analytics" from *IEEE Intelligent Systems*, the author defines a white-collar robot as a set of computer-based capabilities that perform a task or set of tasks historically done by white-collar workers but without the need for human intervention. One example is Watson Analytics, which provides active software that helps investigate data. The author reviews Watson Analytics' capabilities, summarizes its strengths and limitations, and analyzes data in two case studies.

Finally, gathering and analyzing data is essential to the aforementioned topics, and software can alleviate the process. In *IEEE Software*'s "Different Databases for Different Strokes," the author provides an overview of current database-management-system technologies and suppliers, along with a case study of an Internet application. In *IT Professional*'s "The Role of a Customer Data Platform," the author explains that by providing access to data from numerous systems in one database and supporting the systems that can produce an appropriate customer experience, customer database platforms overcome the limitations imposed by fragmented point solutions and present a holistic approach to customer interactions.

The New Killer App for Security: Software Inventory

Gary McGraw, Synopsys

Unless you want to be the next data-breach headline, develop and maintain an inventory of all the software, including open source packages, that your organization uses.

More than half the battle of computer security is knowing what to protect. And more than half of knowing what to protect is knowing what you have. This applies not just to network assets like servers, desktop computers, mobile devices, and laptops but even more importantly to software. In this article I focus on software because in my view, software security is the most important subfield of computer security.[1]

BEFORE YOU CAN FIX IT, YOU NEED TO FIND OUT WHERE IT'S BROKE

How hard could it be to create and maintain an inventory of the software your organization uses? It's harder than you think. Most firms don't have an actionable software inventory—which puts them directly at risk.

In addition, the inventory problem is getting exponentially worse as development evolves because not everything that is code can be called an application. Components, libraries, frameworks, and all sorts of other things are also software that should be on an organization's security radar. Additionally, with the advent of agile, DevOps, CI/CD (continuous integration/continuous delivery), and other development philosophies and tool chains, there are hundreds or even thousands of automation scripts that make the approach work. Every one of those scripts is software that should go through the secure systems development life cycle, end up in the software inventory, and be subject to all the same risk-management decision making as giant flagship apps.

The most recent version of the Building Security in Maturity Model (BSIMM8; bsimm.com) reports that only 40 percent of participating firms (44 of 109) have an operational inventory of software deployments. The activity in question falls under Configuration Management & Vulnerability Management (CMVM) Level 2:

> ***CMVM2.3: Develop an operations inventory of applications.*** *The organization has a map of its software deployments. If a piece of code needs to be changed, Operations or DevOps can reliably identify all the places where the change needs to be installed.*

 Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

EDITOR **JEFFREY VOAS**
NIST; j.voas@ieee.org

Sometimes common components shared between multiple projects are noted so that when an error occurs in one application, other applications that share the same components can be fixed as well.

The aim of CMVM2.3 is clearly fixing broken code *everywhere* that it appears. As we'll see, this is an essential part of defense against attacks.

OPEN SOURCE ISN'T SECURE, AND THERE'S LOTS OF IT

Software exploitation is one of the most effective offensive cybersecurity weapons because there's so much broken software out there to exploit. Modern software is complex: hundreds of unique components, many of them dynamically marshalled on the fly when needed, work together to make an application tick. Many of these components are open source because speed is king and "minimum viable product" definitions are almost always based on functions alone. According to security firm Black Duck (full disclosure: this firm was recently acquired by my company Synopsys), 96 percent of the applications scanned in their 2017 analysis utilized open source software, and 67 percent of those applications had vulnerabilities that had been known for over four years on average.[2] Talk about a target-rich environment where "not invented here" sadly also means "not fixed here!"

Identifying open source software is clearly an important part of building and maintaining a functioning software inventory, which as we have already established is mandatory for increasing the maturity of a software security initiative. It's also a BSIMM activity, but in this case one practiced by only 23 percent of BSIMM8 firms (25 of 109). The activity falls under Standards & Requirements (SR) Level 2:

SR2.4: Identify open source. *The first step toward managing risk introduced by open source is to identify the open source components in use across the portfolio and really understand which dependencies they drag on stage. It's not uncommon to discover old versions of components with known vulnerabilities or multiple versions of the same component. Automated tools for finding open source, whether whole components or large chunks of borrowed code, are one way to approach this activity. An informal annual review or a process that relies solely on developers asking for permission does not generate satisfactory results. At the next level of maturity, this activity is subsumed by a policy constraining the use of open source.*

Identifying open source software is an important part of building and maintaining a functioning software inventory.

Of course, knowing that you have open source software and keeping track of it isn't sufficient if what you're using remains riddled with known vulnerabilities. Properly managing open source risk involves governance around choosing the open source software, obtaining it, and finally ensuring that everyone in your organization is always using up-to-date releases and fixing discovered problems (just as actually fixing, not just identifying, defects is required to make software better).

Even fewer BSIMM8 firms—9 percent (10 of 109)—control open source risk than identify open source use. The associated activity falls under SR Level 3:

SR3.1: Control open source. *The organization has control over its exposure to the vulnerabilities that come along with using open source components and their army of dependencies. Use of open source could be restricted to predetermined projects. It could also be restricted to open source versions that have been through an SSG [software security group] security screening process, had unacceptable vulnerabilities remediated, and are made available only through internal repositories. Legal often spearheads additional open source controls due to the "viral" license problem associated with GPL [GNU General Public License] code. Getting Legal to understand security risks can help move an organization to improve its open source practices. Of course, this control must be applied across the software portfolio.*

Clearly we have a software inventory problem. Just as clearly, open source software seems to play a major role in the problem.

WHEN THINGS GO BADLY WRONG

Don't just take my word for it when it comes to risk and open source software. Consider what happened to Equifax when an unpatched problem in a commonly used and very popular open source component turned out to be the root cause of the massive data

This article originally appeared in Computer, *vol. 51, no. 2, 2018.*

breach exposing financial data of as many as 143 million US consumers.

"Equifax has been intensely investigating the scope of the intrusion with the assistance of a leading, independent cybersecurity firm to determine what information was accessed and who has been impacted," the company wrote in a posting (help.equifax.com/s/article/What-was-the-vulnerability). "We know that criminals exploited a U.S. website application vulnerability. The vulnerability was Apache Struts CVE-2017-5638. We continue to work with law enforcement as part of our criminal investigation, and have shared indicators of compromise with law enforcement."

By tracking their software inventory and properly fixing broken code, Equifax could well have dodged a bullet and avoided billions of dollars in losses and being front-page news for all the wrong reasons. Oops.

KNOW THYSELF

Another perspective on the criticality of software inventory to security comes from the NSA's Tailored Access Operations (TAO) group, which undertakes offensive cyberactions and simultaneously develops a better understanding of what it takes to defend a network against a nation-state adversary.

In a talk at the USENIX Enigma conference in January 2016, TAO chief Rob Joyce explained that the organization does its work by knowing a target's software inventory better than the target itself.[3] Through proper reconnaissance, TAO builds up an understanding of what software is running on a target's machines (including open source, of course). When a new vulnerability is discovered in software that is part of the target's inventory, TAO is able to carry out initial exploitation. Game over.

Defending against this kind of inventory-based attack vector is as simple as developing and maintaining a better inventory of your own stuff than an adversary has. The good news is that organizations are in the best position to inventory their own stuff. The bad news is that not enough are doing so today.

By now, the importance of software inventory to security should be abundantly clear. Unless you want to be the next data-breach headline, develop and maintain an inventory of all the software, including open source packages, that your organization uses.

REFERENCES

1. G. McGraw, *Software Security: Building Security In*, Addison-Wesley, 2006.
2. Black Duck Center for Open Source Research and Innovation (COSRI), *2017 Open Source Security & Risk Analysis*; www.blackducksoftware.com/download/open-source-security-risk-analysis-2017.
3. R. Joyce, "Disrupting Nation State Hackers," video, YouTube, 28 Jan. 2016; www.youtube.com/watch?v=bDJb8WOJYdA&ab_channel=USENIXEnigmaConference.

GARY MCGRAW is Vice President of Security Technology at Synopsys as well as the author of 12 books on software security. He holds a dual PhD in cognitive science and computer science from Indiana University. Contact him through his website at garymcgraw.com.

Publish Your Software: Introducing the *Journal of Open Source Software (JOSS)*

Daniel S. Katz
National Center for Supercomputing Applications; University of Illinois, Urbana–Champaign

Kyle E. Niemeyer
Oregon State University

Arfon M. Smith
Space Telescope Science Institute

Editors:
Jeffrey Carver, carver@cs.ua.edu; Damian Rouson, damian@sourceryinstitute.org

Software is essential to most research today. We know this because we have asked researchers and examined their outputs. A 2014 study of UK Russell Group Universities[1] reports that approximately 90 percent of academics surveyed said that they use software in their research, and more than 70 percent said that their research would be impractical without it. About half of these UK academics reported that they develop their own software while in the course of doing research. Similarly, a 2017 survey of US National Postdoctoral Association members found that 95 percent used research software, and 63 percent said that their research would be impractical without it.[2] An initial study of software's role in journal articles examined three months of the journal *Nature* and found that during this period, of the 40 research articles *Nature* published, 32 mentioned software and averaged 6.5 software package mentions per article.[3]

However, even though software is a critical part of modern research, its publication, acknowledgement, and citation are not well-supported across the scholarly ecosystem.[4] Academic publishing has not changed substantially since its inception. Science, engineering, and many other academic fields still view research articles as the key indicator of research productivity, with research grants being another important indicator. Yet, the research article is inadequate to fully describe modern, data-intensive, computational research.

The *Journal of Open Source Software* (*JOSS*; joss.theoj.org)[5] focuses on research software and its place in the scholarly publishing ecosystem. Its goal is to make it easy for authors to publish a paper about their software, mostly focused on the software itself, and then be credited when this software is used based on the users citing the *JOSS* paper. Thus, the journal's selling point, as stated in the author guidelines, is as follows:

> *If you've already licensed your code and have good documentation, then we expect that it should take less than an hour to prepare and submit your paper to* JOSS.

2469-7087/18/$33.00 © 2018 IEEE Published by the IEEE Computer Society June 2018

JOSS is

- **Built on GitHub.** We use GitHub so that we can take advantage of numerous of its features (for example, the use of issues for reviews; rapid interaction between author, reviewer, and editor; and notifications) and authors' and reviewers' general familiarity with GitHub.
- **Open access.** Papers submitted to and accepted by *JOSS* are open access (CC-BY licensed), and authors retain the copyright to their work. The software components of the publications are licensed under an Open Source Initiative (OSI) approved license.
- **Intended for research software.** *JOSS* attempts to be general to all research software, not just for science or any single field. The main question we ask of submitters is, "Is it likely that users of this software will want to cite it?"
- **Intended to be easy to use for authors.** Authors with well-documented code in a public code repository simply need to create a directory in that repository that contains a short paper in Markdown format; in that paper, they then declare the authors of the software, the purpose of the software, and any needed references.
- **Intended to be easy to use for reviewers.** Reviewers work through a checklist of items related to *JOSS*'s conflicts of interest policy, the code and documentation in the software repository, and the software paper itself, as further discussed in the next section.
- **Supported by NumFOCUS.** NumFOCUS (numfocus.org) is a 501(c)(3) nonprofit that supports and promotes world-class, innovative, open source scientific computing. It provides a forum for such projects to work through issues, and provides a mechanism for these projects to accept funding. Open Journals (http://theoj.org) is a collection of open source, open access journals, of which *JOSS* is the flagship publication. Other journals based on the same model are in varying stages of spinning up.
- **Endorsed by and affiliated with OSI.** The OSI (opensource.org) is a global nonprofit organization that protects and promotes open source software, development, and communities by championing software freedom in society through education, collaboration, and infrastructure, stewarding the Open Source Definition (OSD) and preventing abuse of the ideals and ethos inherent to the open source movement. In support of this mission, *JOSS* requires that the software it publishes uses an OSI-approved license.
- **Indexed.** *JOSS* is listed on Sherpa/Romeo, and *JOSS* papers are indexed by ADS (http://adsabs.harvard.edu), which in turn is indexed by Google Scholar.
- **Connected to other code reviewing systems.** For submissions of software that has already been reviewed under rOpenSci's rigorous onboarding guidelines,[6] *JOSS* does not perform further review; the editor in chief fast-tracks such submissions to acceptance. And this could be extended to work with other trustworthy systems.

JOSS PROCESS

The typical *JOSS* submission and review process (shown in Figure 1) follows the steps described below.

1. An author submits an article, including a link to software, to *JOSS* using the web application and submission tool. The article is a Markdown file named paper.md, visibly located in the software repository (in many cases, placed together with auxiliary files in a paper directory).

2. Following a routine check by a *JOSS* administrator, a "pre-review" issue is created in the joss-reviews (https://github.com/openjournals/joss-reviews) GitHub repository. In this pre-review issue, an editor is assigned who then identifies and assigns a suitable reviewer. The editor then asks the automated bot Whedon to create the main submission review issue.

3. The reviewer conducts the submission review in the issue by working through a checklist of review items:

- agreement with JOSS policies on conflicts of interest and code of conduct;
- general checks to ensure that the source code is available, an OSI-approved license was used, the software version in the paper and the repository match, and the submitter is an author of the software;

- checks on functionality, including installation and performance claims, if appropriate;
- checks on documentation, which should include a statement of need for the software, installation instructions, example usage, and documentation of functionality, tests, and community guidelines; and
- checks on the paper itself, including ensuring that the author list is reasonable, the statement of need is included, and sufficient and well-structured references are present.

The author, reviewer, and editor discuss any questions that arise during the review, and once the reviewer completes the checks, he or she notifies the submitting author and editor. *JOSS* reviews are discussions—in the open within a GitHub issue—between the reviewer(s), author(s), and editor. Like a true conversation, discussion can go back and forth in minutes or seconds, with all parties contributing at will. This contrasts with traditional journal reviews, where the process is merely an exchange between the reviewer(s) and author(s) via the editor, which can take months for each communication and, in practice, is usually limited to one to three exchanges due to that delay.[7] The reviews are subject to a code of conduct (https://github.com/openjournals/joss/blob/master/CODE_OF_CONDUCT.md); both authors and reviewers must confirm that they have read and will adhere to this code of conduct, during submission and with their review, respectively.

4. After the review is complete, the editor asks the submitting author to make a permanent archive of the software (including any changes made during review) with a service such as Zenodo or Figshare, and to post a link to the archive in the review thread. This link, in the form of a DOI, is attached to the submission.

5. The editor in chief produces a compiled PDF, updates the *JOSS* website, deposits Crossref metadata, and issues a Crossref DOI for the submission.

6. Finally, the editor in chief updates the review issue with the *JOSS* article DOI and closes the review. The submission is then accepted into the journal.

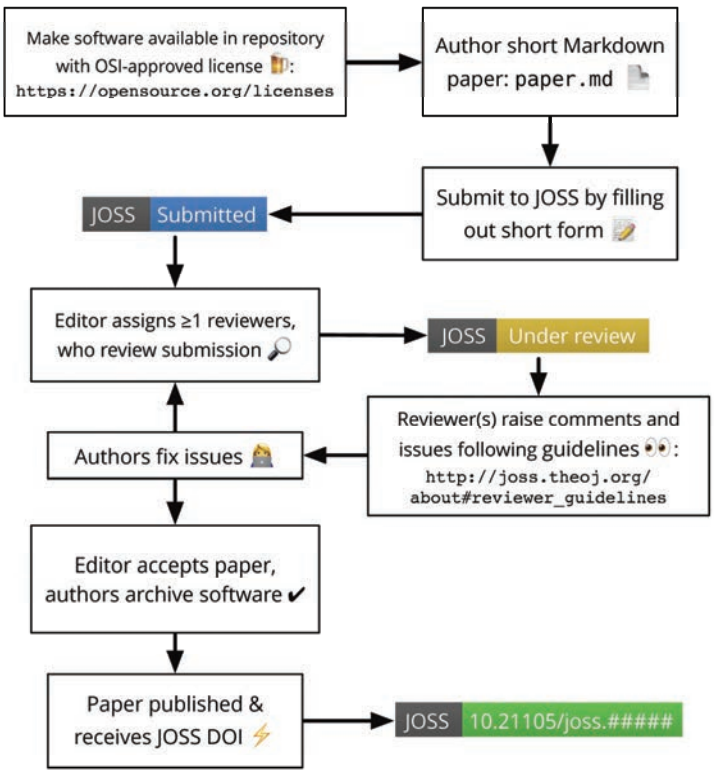


Figure 1. The *JOSS* submission and review flow, including the various status badges that can be embedded on third-party settings such as GitHub README documentation.[8]

STATUS AND FUTURE DIRECTIONS

JOSS received its first submission in May 2016, and as of 22 January 2018, has published 206 papers, with 63 more submitted and being reviewed or otherwise processed. The most-cited articles from *JOSS*'s first 20 months are Daniel Foreman-Mackey's article on corner.py[9] and Conrad Sanderson and Ryan Curtin's article on Armadillo,[10] with 129 and 86 citations, respectively, according to Google Scholar.

The number of submissions and the fact that we have 19 volunteer editors seem to indicate that *JOSS* is satisfying a need. *JOSS* is not the final answer to the problem we ultimately want to solve: authors receiving credit directly for their software. They should not need to write papers about their software to advertise and fit it within the journal system. Nevertheless, our approach is a good intermediate step to highlight the need for recognition and credit for software.

With general awareness of *JOSS* in the community increasing, the *JOSS* editorial team has been contacted by other journals and publishers interested in our approach. These journals generally either want to accept software papers in their journals but desire a deeper review of the software, or want to add software reviews to their editorial process for scientific papers but do not have the expertise in their reviewer pool. Thus, we are considering allowing other journals to request a *JOSS* review as part of their editorial process when software forms a major part of a submission. One publisher interested in such an approach is the American Astronomical Society (AAS), which counts the high-profile *Astrophysical Journal* among its publications.

In addition to potentially collaborating with other journals, we are also in the process of generalizing the *JOSS* infrastructure to allow other journals to be easily created with the same tools. Supported by a small grant from the Alfred P. Sloan Foundation, the core *JOSS* application (https://github.com/openjournals/joss) and the Whedon bot (https://github.com/openjournals/whedon-api) are being adapted to support multiple *JOSS*-like publications. Led by a subset of the *JOSS* editorial team, the *Journal of Open Source Education* will likely be the first new publication to use the generalized infrastructure.

REFERENCES

1. S. Hettrick et al., "UK Research Software Survey 2014," *dataset*, Univ. of Edinburgh on behalf of Software Sustainability Inst., 2015; doi.org/10.7488/ds/253.
2. U. Nangia and D.S. Katz, "Track 1 Paper: Surveying the US National Postdoctoral Association Regarding Software Use and Training in Research," *Proc. Workshop on Sustainable Software for Science: Practice and Experiences* (WSSSPE 5.1), 2017; doi.org/10.6084/m9.figshare.5328442.
3. U. Nangia and D.S. Katz, "Understanding Software in Research: Initial Results from Examining Nature and a Call for Collaboration," *Proc. Workshop on Sustainable Software for Science: Practice and Experiences* (WSSSPE 5.2), 2017; doi.org/10.1109/eScience.2017.78.
4. K.E. Niemeyer, A.M. Smith, and D.S. Katz, "The Challenge and Promise of Software Citation for Credit, Identification, Discovery, and Reuse," *J. Data and Information Quality*, vol. 7, no. 4, 2016, p. 16.
5. A.M. Smith et al., "Journal of Open Source Software (JOSS): Design and First-Year Review," *PeerJ Computer Science*, vol. 4, 2018, p. e7.
6. K. Ram,, N. Ross, and S. Chamberlain, "Lightning Talk: A Model for Peer Review and Onboarding Research Software," *Proc. Fourth Workshop on Sustainable Software for Science: Practice and Experiences* (WSSSPE 4), 2016; ceur-ws.org/Vol-1686/WSSSPE4_paper_13.pdf.
7. J.P. Tennant et al., "A Multi-Disciplinary Perspective on Emergent and Future Innovations in Peer Review [version 2; referees: 2 approved with reservations]," *F1000Research*, vol. 6, no. 1151, 2017; doi.org/10.12688/f1000research.12037.2.
8. K.E. Niemeyer, "JOSS Publication Flowchart," Figshare, 2017; doi.org/10.6084/m9.figshare.5147773.v1.

9. D. Foreman-Mackey, "corner.py: Scatterplot Matrices in Python," *J. Open Source Software*, vol. 1, no. 2, 2016, p. 24.
10. C. Sanderson and R. Curtin, "Armadillo: A Template-based C++ Library for Linear Algebra," *J. Open Source Software*, vol. 1, no. 2, 2016, p. 26.

ABOUT THE AUTHORS

Daniel S. Katz is the assistant director for Scientific Software and Applications at the National Center for Supercomputing Applications, and a research associate professor in the Departments of Computer Science and Electrical and Computer Engineering and the School of Information Sciences at the University of Illinois, Urbana–Champaign. His research interests include applications, algorithms, fault tolerance, programming in parallel and distributed computing, citation and credit mechanisms and practices associated with software and data, organization and community practices for collaboration, and career paths for computing researchers. Katz received a PhD in electrical engineering from Northwestern University. Contact him at d.katz@ieee.org.

Kyle E. Niemeyer is an assistant professor of mechanical engineering in the School of Mechanical, Industrial, and Manufacturing Engineering at Oregon State University. His research interests include the development of advanced numerical methods for modeling of combustion and reactive flows, and computational modeling of multiphysics flows for applications in aerospace, transportation, and energy systems. Niemeyer received a PhD in mechanical engineering from Case Western Reserve University. Contact him at kyle.niemeyer@oregonstate.edu.

Arfon M. Smith is the head of the Data Science Mission Office at the Space Telescope Science Institute. His research interests include academic credit models, crowd-sourcing/citizen science, interstellar medium, open source software, and scholarly publishing. Smith received a PhD in astrochemistry from the University of Nottingham. Contact him at arfon@stsci.edu.

This article originally appeared in
Computing in Science & Engineering, *vol. 20, no. 3, 2018.*

How Wearable Computing Is Shaping Digital Health

Oliver Amft
FAU Erlangen-Nürnberg

Editors:
Oliver Amft
amft@computer.org

Kristof Van Laerhoven
kvl@eti.uni-siegen.de

Wearable computing enables more personalized healthcare through a distributed information sharing model that puts patients and users rather than providers, insurers, and other industry stakeholders at the center. It also fosters the creation of new health knowledge and more effective prevention and treatment techniques by integrating vital-sign data, health-related behavioral data, and environmental-exposure data with clinical and genetic data. Realizing the promise of wearables and digital health, however, will require multiple parallel technological advances.

Wearable computing has introduced new approaches, more efficient processes, and innovative products in entertainment, sports, industrial logistics, and many other areas. However, no other field is anticipating and integrating wearable technology so broadly as healthcare, with interests ranging from well-being and disease prevention to chronic patient care and cross-cutting all medical disciplines.

Wearable computing researchers have foreseen numerous healthcare applications for at least two decades. Within the past several years, however, medical professionals and engineers have started integrating wearable technology in diagnosis and care processes, validating their effectiveness with patients in observational studies[1] and even randomized controlled trials.[2] A leading example is the continuous glucose monitor, which enables diabetic patients to self-monitor blood glucose levels and learn from the measurements about their body functions, leading to advanced self-management and treatment procedures.[3] However, the variety of wearable technologies and applications in healthcare stretches far beyond glucose monitoring. Medical journals now regularly publish articles with wearables at the center of their methodology and some, such as the *Journal of Medical Internet Research*, dedicate space to such technology.

What drives wearable health technology development? Although cost is a primary factor, wearable computing offers more than greater efficiency in clinical processes and the ability to monitor vital parameters—it promises to assume a unique role in maintaining health, improving patient conditions, and extending quality life-years.

 Published by the IEEE Computer Society

THE CHANGING INFORMATION LANDSCAPE

To understand wearable technology's future role in medicine, it is important to reflect on healthcare's ongoing transformation.

As Figure 1 shows in simplified form, patient information has traditionally been stored and processed by healthcare practitioners, hospitals, specialty clinics, insurers, and other industry stakeholders independently. These organizations manage all medical processes—from prevention and diagnosis to intervention, recovery, and chronic disease care. The concentration of more and more data and knowledge in centralized IT systems has invariably led to the systems and staff being overwhelmed with requests and tasks. It is important to note that the well-known concept of *patient-centered care*, wherein each patient's individual values and health outcomes guide clinical and care decisions, is compatible with and widely implemented even in centralized information systems.

The current system's structure can contribute to soaring costs: Across the EU, healthcare spending now accounts for nearly 10 percent of gross domestic product on average.[4]

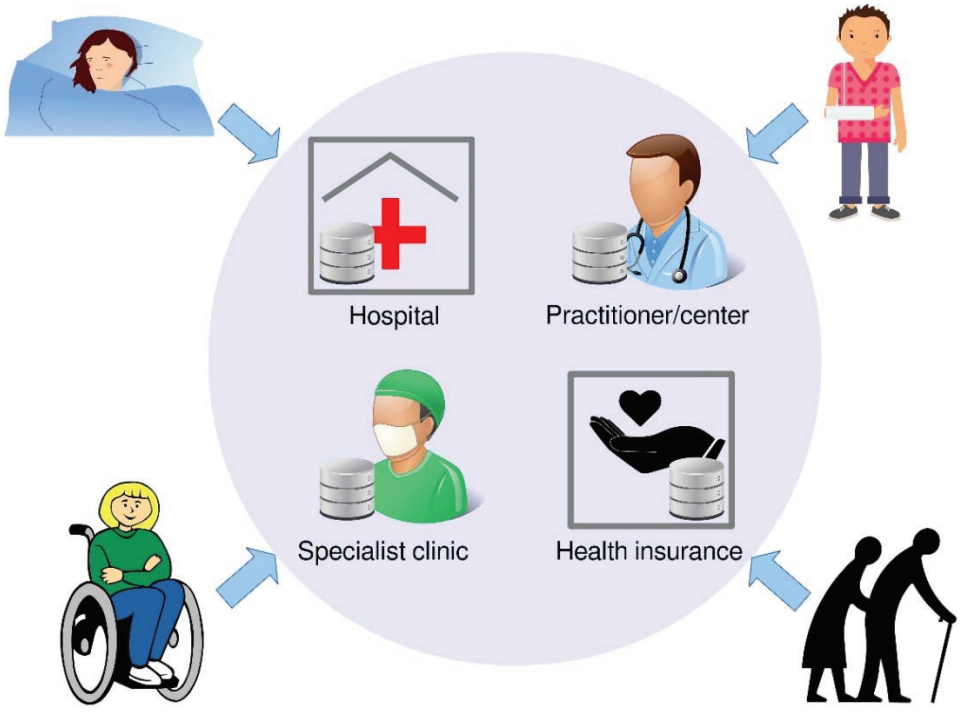


Figure 1. Patient information has traditionally been concentrated in the hands of healthcare providers and insurers.

Also adding to healthcare costs is the unmet need for medical care, in particular for low-income groups, resulting in more patients attending hospital emergency departments. In Germany, for example, about 25 percent of patients who visited an emergency department between 2011 and 2013 might not have required emergency care.[5] In addition, even though treatment effectiveness is continually improving, chronic and noncommunicable diseases are on the rise because of our aging society. Thus, we need concerted efforts and planning for prevention and long-term care with cost-effective health services for years to come.

The advance of wearable technology has motivated healthcare professionals to look outside the office or clinic to help identify health risks, monitor disease progress, and provide treatment or advice. For the first time, relevant data can be acquired independently from a controlled clinical setting. Continuous monitoring over days, months, or even years is feasible. Simultaneously, users are becoming more knowledgeable about health topics and the information collected by

wearable devices, which extends far beyond Quantified Self (lifelogging) nerds. Leveraging Internet resources and social networks, users are self-interpreting their data and sharing and discussing their findings with others. Consequently, new well-being and patient interest groups are emerging as citizen science communities.

As detailed in the previous Wearable Computing department,[6] wearables are moving beyond traditional form factors such as smartphones and smart watches. Sensors in smart on-body accessories, smart cloths, and smart body patches can gather novel types of health data and provide interventions such as on-demand drug release. Smart implants for nonsevere medical conditions are a likely future addition.

As Figure 2 shows, wearables are leading to a new health information model that puts users and patients at the center of a distributed information sharing system. With hospitals and practitioners no longer the sole or even the main data generators, wearables will assume greater importance in forging new insights and developing more efficient processes. Compared to traditional systems, wearables will generate data at higher velocity (as measurements are continuous) and at higher volume (due to wearables' rising utility), and such data will be more varied because of the numerous types of sensors and interactions.

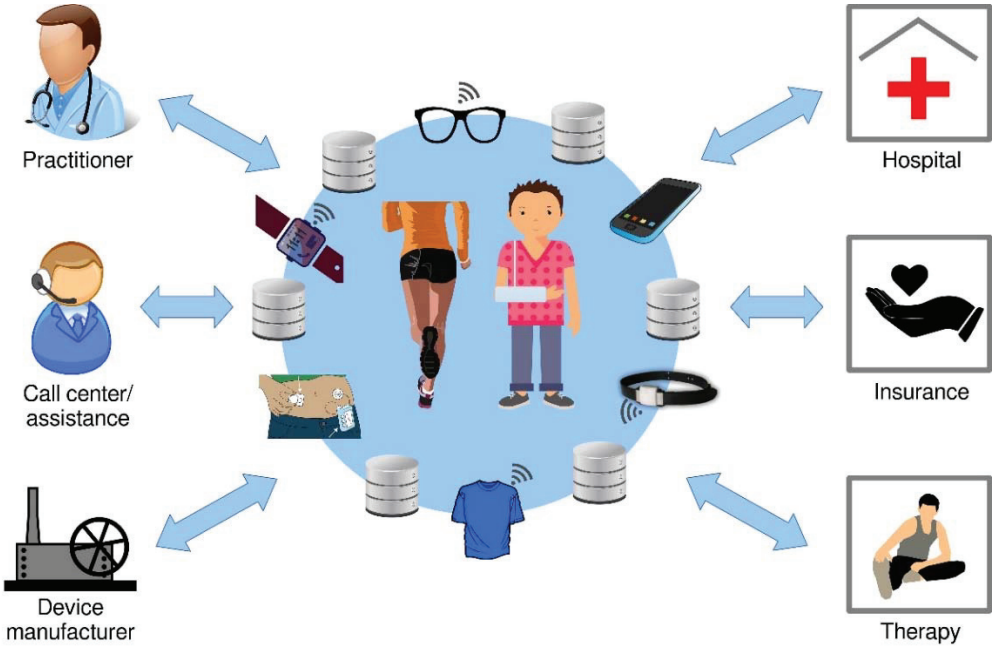


Figure 2. The generation of big data through wearable and mobile health technology is leading to the emergence of a distributed information sharing system in which all industry stakeholders—including providers, promoters, insurers, and device manufacturers—participate in storing, providing, and mining health data.

With data virtualization, healthcare data is no longer centralized in a few places and only accessible through a handful of gatekeepers. Instead, providers, promoters, insurers, and device manufacturers are now storing, providing, and mining user and patient health data.[7]

At least initially independent of the wearables development, many countries have launched efforts to increase collaboration among different health industry stakeholders. In 2015, for example, the Obama administration launched the Precision Medicine Initiative (now All of Us) to facilitate information sharing among patients, researchers, and providers to improve healthcare delivery.[8] Germany recently created a €150 million fund for medical informatics research to improve data management and sharing,[9] primarily in hospital networks.

Clearly, data management, mining, and interpretation require proper access and control mechanisms, as well as federation, curation, and compatibility standards for databases and the integration of electronic medical and patient records. Wearable technology will extend and diversify these challenges. In addition, as patients obtain more control over health data, they also must assume greater responsibility for managing it.

INTEGRATING WEARABLE COMPUTING WITH DIGITAL HEALTH

Digital health refers to the convergence of various information technologies to maximize healthcare effectiveness, including data management (eHealth), mobile devices and apps (mHealth, wireless health), and remote patient management (connected health, telemedicine). Various governments have launched digital health programs through their main health agencies, including the UK's National Health Service[10] and the US Food and Drug Administration.[11] Market studies highlight the many benefits of digital health including higher productivity, better clinical outcomes, and a more satisfying patient experience.[12] Researchers also see digital health as a major driver for personalized medicine.

Wearable computing is shaping digital health in multiple ways.

Preventive Medicine

A central goal of digital health is to prevent disease and identify its onset at the earliest possible stage. Wearables have the potential to identify many early physiological and behavioral markers of chronic diseases. With this in view, the European Innovative Medicines Initiative (www.imi.europa.eu/about-imi) brings together pharmaceutical companies and researchers to investigate how wearable and mobile technology could be used to diagnose autism, diabetes, depression, and Alzheimer's disease sooner. Wearables are already used to promote healthy behavior through self-tracking and gaming approaches. Many other health promotion opportunities are sure to emerge with new wearable technologies.

Clinical Support

Capturing complex clinical processes using digital data management models could improve healthcare treatment and decrease errors. Researchers have demonstrated that wearable technology can support this effort by, for example, dynamically providing relevant information in ward rounds or detecting mistakes in operating theaters.[13] In furtherance of this goal, my colleagues and I developed pattern analysis methods to interpret unsupervised behavior and estimate independence in daily activities of poststroke patients measured using body-worn inertial sensors.[14] Support for clinical processes, along with online and remote assessments, is an important growth area for wearable technology.

Monitoring and Intervention

Although researchers have extensively explored the use of wearable technology for outpatient monitoring, only a few medical-grade systems exist. Potential target patients range from those with heart arrhythmia and others at risk of heart failure to fragile walkers with a high risk of falling. Wearable monitoring systems are also an ideal tool for tracking many health-relevant behaviors, activities, and consumption habits as well as physiological responses (such as arousal, stress, and emotions) and cognitive performance. The next crucial development step for wearable technology is intervention—for example, in glucose level management and behavior guidance. More progress in context recognition research is necessary to yield robust wearable intervention systems.

Integrating Wearable and Clinical Big Data

Integrating data from clinical sources and wearable monitors provides the potential to yield new biomedical knowledge. In particular, clinical and wearable data complement each other in coverage (the lab versus anywhere), fidelity (exact diagnostics versus estimates and noisy measurements), and supervision (supervised versus unsupervised/free living). As wearable use increases and more clinical data becomes accessible through new data interoperability standards, researchers will gain new knowledge by combining these two data sources. There will be greater insight into treatments' effects in free living and better diagnostic evidence. Physicians and therapists will be able to adjust both clinical treatments and wearables' monitoring and intervention functions continuously for each patient.

Health Education

Outside the healthcare and research communities, the many existing and potential benefits of wearable health technology are not widely known. Wearable health education initiatives along with virtual coaches are needed to promote and sustain use of this technology among the public, beyond wearable enthusiasts, lifeloggers, and a limited number of well-informed patients.

INTEGRATING BEHAVIORAL, GENOMIC, AND ENVIRONMENTAL EXPOSURE DATA

Perhaps the most profound potential of digital health lies in integrating health-related behavioral data with personal genetic information and environmental-exposure data to identify individual health risks and ideally prevent diseases before they occur.[15] While health risks might originate from a genetic predisposition, environment and behavior are modulating health risks and can introduce additional ones. The health implications of living in an air-polluted area or smoking are two well-known examples.

Christopher Paul Wild denotes the monitoring, measuring, and modeling of a person's lifetime environmental exposure, or exposome, as exposomics,[16] analogous to the sequencing and mapping of the human genome, or genomics. Since 2013, various large EU and US research projects have investigated the exposome from prenatal to adult life, including air, water, and food pollutants. These projects have integrated on-body measurements from available technology, mostly smartphones and a few additional sensors. Rapid sensor and wearable development will likely expand the ability to cover most environmental variables, maximize regional coverage, and provide long-term statistics.

The exposome includes well-known lifestyle factors important to health such as alcohol use, diet, and exercise. However, health is also impacted by other aspects of human behavior such as social contact, daily activities, and mental attitude. Inspired by Darryl Macer and Masakazu Inaba's term behaviorome, which refers to the spectrum of human ideas,[17,18] I refer to all health-related behavior as health-behaviorome.

As Figure 3 shows, wearable computing technology is critical to the digital health vision because it offers the ability to continuously assess an individual's exposome and health-behaviorome. Integrating exposome and health-behaviorome data with genetic data will make it possible to implement personalized risk prevention and treatment strategies.

WEARABLES IN FUTURE MEDICINE

Wearable computing is beginning to assume a prominent role in digital health, but distributed information sharing and health data integration is still largely a vision. To leverage wearables' full potential, researchers must address several technical challenges (some of these were discussed in the previous "Wearable Computing" department[6]). The most critical relate to validity/reliability and long-term compliance.

Navigating the current health devices market is tedious even for the technically savvy and is at least murky for most users and healthcare professionals. Research has debunked the oft-claimed one-device-can-do-it-all paradigm, which presents a challenge for physicians of which devices to prescribe. The solution likely involves

- scalable validation methods and proper medical study results,
- navigation tools to identify adequate (ideally validated) wearables, and
- novel procedures to address the tradeoff between market size and development effort.

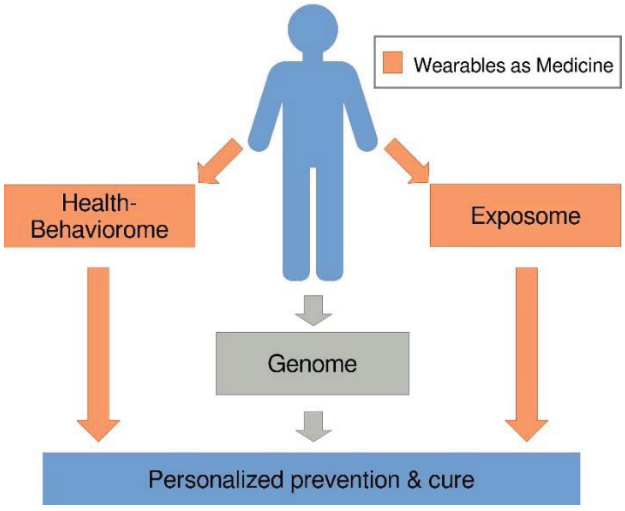


Figure 3. Integrating an individual's exposome and health-behaviorome, obtained through wearables, with genetic data will make it possible to implement personalized risk prevention and treatment strategies.

Digital health benefits from wearables will only be realized through routine use potentially over many years. Yet, many current wearables have demonstrated critical compliance problems. Progress on several frontiers might address this challenge too:

- health information must be continuously delivered to users like the weather and news through automated education services;
- actively dealing with and communicating uncertainty in estimated information may help build users' trust in wearable health devices; and
- innovative device design must align with wearers' needs and preferences, not vice versa.

CONCLUSION

Wearable computing is shaping digital health in several ways. First, wearable computing is fostering a transition in the way health data is created, stored, processed, and managed from the traditional model of centralized IT systems to a new model of distributed information sharing with patients and users rather than providers, insurers, and other industry stakeholders at the center. Second, the integration of vital-sign and other wearable data with clinical data is leading to new health knowledge and techniques. Third, wearables are ideally suited to capture health-related behavioral and environmental exposure data, which can complement genetic data to more effectively prevent and treat chronic diseases and thereby gain quality life-years. Progress must be made on several fronts to address wearable computing's current technical challenges, but the promise of profoundly improved health outcomes and more personalized prevention and care makes such endeavors worthwhile.

This article originally appeared in
IEEE Pervasive Computing, *vol. 17, no. 1, 2018.*

REFERENCES

1. J. Seiter et al., "Daily Life Activity Routine Discovery in Hemiparetic Rehabilitation Patients Using Topic Models," *Methods of Information in Medicine*, vol. 54, no. 3, 2015, pp. 248–255.
2. M. Faurholt-Jepsen et al., "Daily Electronic Self-Monitoring of Subjective and Objective Symptoms in Bipolar Disorder—the MONARCA Trial Protocol (MONitoring, treAtment and pRediCtion of bipolAr disorder episodes): A Randomised Controlled Single-Blind Trial," *BMJ Open*, vol. 3, no. 7, 2013; doi.org/10.1136/bmjopen-2013-003353.
3. D. Rodbard, "Continuous Glucose Monitoring: A Review of Successes, Challenges, and Opportunities," *Diabetes Technology & Therapeutics*, vol. 18, no. S2, 2016, pp. S2-3–S2-13.
4. *Health at a Glance: Europe 2016*, tech_report, OECD Publishing, 2016; www.oecd.org/health/health-at-a-glance-europe-23056088.htm.
5. M.J. van den Berg, T. van Loenen, and G.P. Westert, "Accessible and Continuous Primary Care May Help Reduce Rates of Emergency Department Use: An International Survey in 34 Countries," *Family Practice*, vol. 33, no. 1, 2016, pp. 42–50.
6. O. Amft and K. Van Laerhoven, "What Will We Wear After Smartphones?," *IEEE Pervasive Computing*, vol. 16, no. 4, 2017, pp. 80–85.
7. D. Lupton, "Health Promotion in the Digital Era: A Critical Commentary," *Health Promotion Int'l*, vol. 30, no. 1, 2014, pp. 174–183.
8. F.S. Collins and H. Varmus, "A New Initiative on Precision Medicine," *New England J. Medicine*, vol. 372, no. 9, 2015, pp. 793–795.
9. *Better Therapies Thanks to Medical Informatics*, German Federal Ministry of Education and Research (BMBF), 2017; www.bmbf.de/de/bessere-therapien-dank-medizininformatik-4473.html.
10. "NHS Digital Extends Programme to Widen Participation in Digital Health," *NHS Digital*, 2017; digital.nhs.uk/article/1365/NHS-Digital-extends-programme-to-widen-participation-in-digital-health.
11. *Digital Health*, government report, US Food & Drug Administration (FDA), 2017; www.fda.gov/medicaldevices/digitalhealth.
12. *Digital Health Technology Vision 2017*, tech_report, Accenture, 2017; www.accenture.com/us-en/insight-digital-health-tech-vision-2017.
13. S. Jalaliniya et al., "Touch-less Interaction with Medical Images Using Hand & Foot Gestures," *Proc. 2013 ACM Conf. Pervasive and Ubiquitous Computing Adjunct Publication*, 2013, pp. 1265–1274.
14. A. Derungs et al., "Estimating Physical Ability of Stroke Patients without Specific Tests," *Proc. 2015 ACM Int'l Symp. Wearable Computers*, 2015, pp. 137–140.
15. E.D. Green et al., "Charting a Course for Genomic Medicine from Base Pairs to Bedside," *Nature*, vol. 470, no. 7333, 2011, pp. 201–213.
16. C.P. Wild, "Complementing the Genome with an 'Exposome': The Outstanding Challenge of Environmental Exposure Measurement in Molecular Epidemiology," *Cancer Epidemiology and Prevention Biomarkers*, vol. 14, no. 8, 2005, pp. 1847–1850.
17. D. Macer, "The Behaviorome Mental Map Project," *The Scientist*, vol. 17, no. 8, 2003, pp. 19–20.
18. A. Saniotis, "Reflections on the 'Human Behaviourome': Mind Mapping and Its Futures," *World Futures*, vol. 63, no. 8, 2007, pp. 611–622.

ABOUT THE AUTHOR

Oliver Amft is a professor, leading the Chair of eHealth and mHealth at the Institute of Medical Informatics at FAU Erlangen-Nürnberg. Contact him at amft@computer.org.

IEEE computer society

PURPOSE: The IEEE Computer Society is the world's largest association of computing professionals and is the leading provider of technical information in the field.
MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.
COMPUTER SOCIETY WEBSITE: www.computer.org
OMBUDSMAN: Direct unresolved complaints to ombudsman@computer.org.
CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.
AVAILABLE INFORMATION: To check membership status, report an address change, or obtain more information on any of the following, email Customer Service at help@computer.org or call +1 714 821 8380 (international) or our toll-free number, +1 800 272 6657 (US):

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES
***Computer*:** The flagship publication of the IEEE Computer Society, *Computer*, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.
Periodicals: The society publishes 13 magazines, 19 transactions, and one letters. Refer to membership application or request information as noted above.
Conference Proceedings & Books: Conference Publishing Services publishes more than 275 titles every year.
Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.
Technical Committees: TCs provide professional interaction in more than 30 technical areas and directly influence computer engineering conferences and publications.
Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.
Certifications: The society offers two software developer credentials. For more information, visit www.computer.org/certification.

NEXT BOARD MEETING

7-8 June 2018, Phoenix, AZ, USA

EXECUTIVE COMMITTEE

President: Hironori Kasahara
President-Elect: Cecilia Metra; **Past President:** Jean-Luc Gaudiot; **First VP, Publication:** Gregory T. Byrd; **Second VP, Secretary:** Dennis J. Frailey; **VP, Member & Geographic Activities: Forrest Shull**; **VP, Professional & Educational Activities: Andy Chen**; **VP, Standards Activities:** Jon Rosdahl; **VP, Technical & Conference Activities:** Hausi Muller; **2018-2019 IEEE Division V Director: John Walz**; **2017-2018 IEEE Division VIII Director:** Dejan Milojicic; **2018 IEEE Division VIII Director-Elect:** Elizabeth L. Burd

BOARD OF GOVERNORS

Term Expiring 2018: Ann DeMarle, Sven Dietrich, Fred Douglis, Vladimir Getov, Bruce M. McMillin, Kunio Uchiyama, Stefano Zanero
Term Expiring 2019: Saurabh Bagchi, Leila DeFloriani, David S. Ebert, Jill I. Gostin, William Gropp, Sumi Helal, Avi Mendelson
Term Expiring 2020: Andy Chen, John D. Johnson, Sy-Yen Kuo, David Lomet, Dimitrios Serpanos, Forrest Shull, Hayato Yamana

EXECUTIVE STAFF

Executive Director: Angela R. Burgess
Director, Governance & Associate Executive Director: Anne Marie Kelly
Director, Finance & Accounting: Sunny Hwang
Director, Information Technology & Services: Sumit Kacker
Director, Membership Development: Eric Berkowitz
Director, Products & Services: Evan M. Butterfield

COMPUTER SOCIETY OFFICES

Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036-4928
Phone: +1 202 371 0101 • **Fax:** +1 202 728 9614
Email: hq.ofc@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, Los Alamitos, CA 90720 **Phone:** +1 714 821 8380
Email: help@computer.org

MEMBERSHIP & PUBLICATION ORDERS
Phone: +1 800 272 6657 • **Fax:** +1 714 821 4641 • Email: help@computer.org
Asia/Pacific: Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • **Fax:** +81 3 3408 3553
Email: tokyo.ofc@computer.org

IEEE BOARD OF DIRECTORS

President & CEO: James Jefferies
President-Elect: Jose M.F. Moura
Past President: Karen Bartleson
Secretary: William P. Walsh
Treasurer: Joseph V. Lillie
Director & President, IEEE-USA: Sandra "Candy" Robinson
Director & President, Standards Association: Forrest D. Wright
Director & VP, Educational Activities: Witold M. Kinsner
Director & VP, Membership and Geographic Activities: Martin Bastiaans
Director & VP, Publication Services and Products: Samir M. El-Ghazaly
Director & VP, Technical Activities: Susan "Kathy" Land
Director & Delegate Division V: John W. Walz
Director & Delegate Division VIII: Dejan Milojičić

IEEE

revised 7 Feb. 2018


Wearable Computing

Editors: Oliver Amft ■ FAU Erlangen-Nürnberg ■ amft@computer.org
Kristof Van Laerhoven ■ University of Siegen ■ kvl@eti.uni-siegen.de

What Will We Wear After Smartphones?

Oliver Amft, *FAU Erlangen-Nürnberg*
Kristof Van Laerhoven, *University of Siegen*

In 2009, this department featured an article titled, "From Backpacks to Smartphones," signifying the trend toward smartphone use in many aspects of wearable computing research.[1] For the first time, smartphones were implementing several key properties of wearable computers—not only providing a convenient platform for prototyping on-body systems but indeed blurring the boundary between mobile and wearable computing. Suddenly, it was feasible to implement large-scale behavior analysis studies—for example, understanding dwelling locations, determining user context and frequent conversation partners from acoustics and movement, or conveniently assessing physical activity.[2]

However, when looking at the sessions of wearable computing conferences over the past few years—including the International Symposium on Wearable Computers and the International Conference on Wearable and Implantable Body Sensor Networks—the prominence of smartphones has declined. Although smartphones remain a wearable research tool, often as a body-hub or prototype for data analysis—we're now starting to see more integrated systems in daily accessories, smart textiles, and flexible on-body patches. Moreover, wearable computing is no longer constrained to one or two conferences of enthusiasts, as was the case in the early days. In fact, other research communities, including computer-human interaction, are frequently hosting wearable computing sessions (for example, CHI 2014 hosted a wearable computing exhibit curated by Clint Zeagler and Thad Starner). Exceeding promises of the early days, wearable computing has indeed entered the business world, with presentation tracks and product booths at industry events, including the Consumer Electronics Show, the IDTechEx Show, and the CeBIT Expo.

Does this progress mean that wearable computing research will now pursue minor incremental improvements of methods and solutions? Are we done with major breakthroughs? If so, should investigators move to new strands, such as the Internet of Things (IoT) and machine-based sensor data interpretation? And what should we make of reports regarding the lack of sustained business success of wearables such as smartwatches and activity trackers, with users quitting after 6–18 months of use?[3]

Sustainability might indeed be the grand challenge. In fact, the EU recently funded the Wear Sustain initiative (http://wearsustain.eu), which pushes creative industries to engage with technology industries though calls for bottom-up projects. Yet it is precisely because of this challenge that we believe wearable computing is entering its most exciting phase yet, as it transitions from demonstrations to the creation of sustained markets and industries, which in turn should drive future research and innovation.

REVIEWING MARKET DEVELOPMENT

Over the past two decades, we have witnessed the market's evolution from bulky carry-on electronics to smartphones and now computing-integrated everyday accessories, clothes, and body patches.

The approach of the early days was to realize computing on the body by replicating the input, output, and processing of personal computers. Early examples included the Twiddler (currently in its third version, twiddler.tekgear.com) and the Matias half-keyboard, worn on one forearm and used with a display on the opposite arm (www.matias.ca). The various commercial head-mounted displays (HMDs) of the 1990s and early 2000s provided the interface for carry-on computing units, including the 1990 Xybernaut, the 2002 Panasonic "Brick," and the 2007 Zypad and OQO, all of which were aimed at commercial success.

Later approaches migrated to sensor-mediated (or exclusive sensor-based) data and information input. During the late 2000s, and driven by the success of smartphones, the wearable computing paradigm led integrated circuit manufacturers to propose various miniaturized, low-power platforms for wearable systems development that either integrated sensors as systems-on-chip or

Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

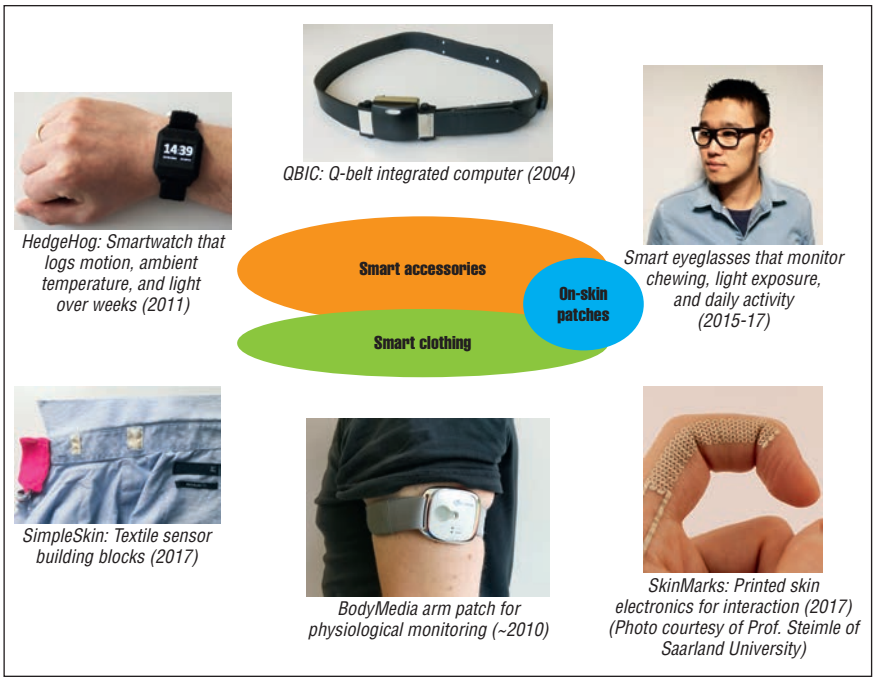


Figure 1. Categories of wearables today and examples illustrating the diversity of systems and applications.

provided interfaces for sensors, displays, and actuators.

Today, many in the wearables industry are trying to build on high-volume, low-cost smartphone components to create secondary markets. A primary example is smartwatches (and other wearables) with integrated inertial measurement units. The InvenSense MPU-92xx family (www.invensense.com), for example, is a frequently used inertial measurement unit. But the InvenSense device does not only provide digital acceleration, gyroscope, and earth-magnetic-field sensor readings in one 3 × 3mm multichip package, it also supports microcontrollers with built-in feature computation, sensor-orientation estimation, and basic pattern-recognition functions.[4]

CATEGORIZING TODAY'S WEARABLES

Today's wearable computing systems can be categorized as follows (see Figure 1):

- smart accessories, including earwear, wristwear, eyewear, shoewear, and beltwear;
- smart clothing; and
- smart on-skin patches, including electronic plasters and tattoos.

The unprecedented market success of activity trackers has fueled the wearables market far beyond the quantified self. For example, Bragi (www.bragi.com) markets wireless earbuds for music playing and communication in addition to sports activity recognition. Smartwatches from Apple and others aim to extend the smartphone to the wrist with sensing, information display, and interaction capabilities. However, aside from smartwatches and wristbands, most commercial "wearable" systems today still appear as a clip-on to clothes or limb straps, rather than being actually integrated in everyday accessories.

As for smart clothing, there is still only a handfull of systems that scale beyond the prototype level. Myant (www.myant.ca) is developing garments with integrated electrodes and sensors for heartbeat and muscle activity monitoring. On the interaction side, Google's Project Jaquart (https://atap.google.com/jacquard) is developing yarns for touch and gesture interfaces.

Similarly, the market for on-skin patches is currently small, and various technical challenges exist regarding cost, durability, bio-compatibility, and low-power operation. Nevertheless, the growth potential of skin patches is promising. Companies such as MC10 (www.mc10inc.com) are prototyping skin patches for physiological monitoring, including important physiological variables: hydration, temperature, and others.

UNDERSTANDING THE CHALLENGES

The application space of wearable systems is highly diversified, as the previous examples indicate. Not only does each sector, from logistics to medical assistance, have its own separate requirements for system implementation and features, but there are many distinct niche applications within the sectors, seeking specific functionality and design.

Application Diversification

The app concept for smartphones is an excellent example of how to deal with diversified user needs: development is focused on small software applications on a common hardware platform. In contrast, wearables are created by combining specific functionality in hardware, software, and design, thus limiting developers in addressing several potential wearable markets with one solution (see Figure 2).

Just in the accessories field, software apps for smartwatches or smart eyeglasses[5] can provide wearers with alternative functions, but are constrained by the form factor, body position, or the device's use model—sports watch or business watch, for example. Google Glass illustrates the *diversification challenge* for an individual wearable device.

Although the preliminary investigation of Glass by many researchers resulted in a variety of applications, its current use is narrow, dominated

WEARABLE COMPUTING

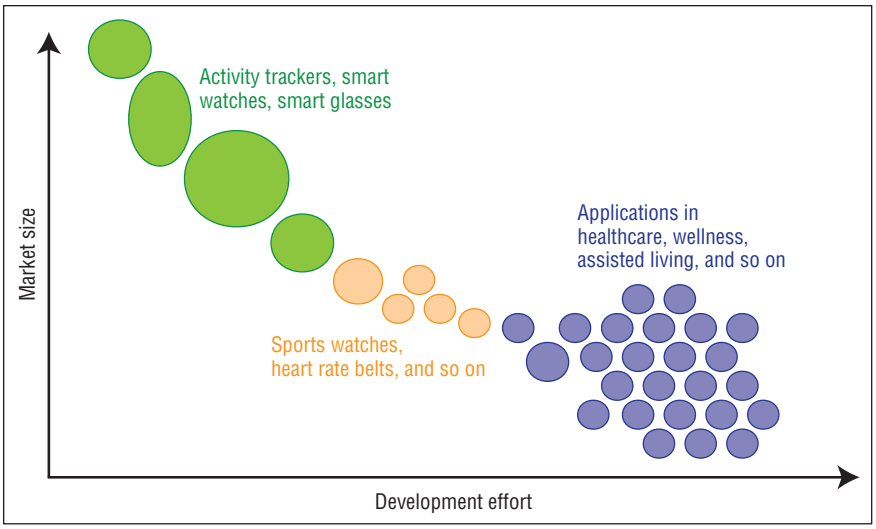


Figure 2. The "market size versus development effort" dilemma in wearable computing. In this qualitative illustration, dots represent examples of wearable computing applications, and their size represents the application bandwidth.

by logistical applications and process-optimization needs in manufacturing.[6] Diversification thus requires considerable effort in terms of individualized development for each application-specific wearable system.

With recent advances in integrating technology into wearables—such as in daily accessories and textiles—using high-volume electronic components is no longer a limiting factor in terms of development effort and product cost. In particular, wearables in healthcare, wellness, and assisted living have relatively small individual market volumes, even though they represent a promising application sector. However, the "market size versus development effort" dilemma cannot be resolved merely using established wearable computing methods. Developers also need robust novel production processes and reliable platforms.

Dealing with Uncertainties

Compared to other IT segments, wearable computing has a low barrier for communicating potential technology benefits to wearers. Because of this low entry barrier, many startup teams are pursuing wearables—from rings to smart glasses and service ideas—successfully using crowdfunding to finance the development and business initiation. Yet many have ultimately ended up as *vaporware*—that is, "good ideas incompletely implemented" (http://downloads.oreilly.com/radar/r1/11-83.pdf). Even those startups and businesses that do eventually succeed often undergo highly critical phases in their development, marked by delays (of months if not years), changes in product concepts, a reduction in features, and user complaints of broken products.

Although development experience in wearable hardware and software is a key to success, teams underestimate the hurdles to realizing reliable products. Here, in particular, embedding electronics into accessories, patches, and so on requires research in reliable processes and the standardization of evaluation methodologies.

IDENTIFYING NEW DRIVERS

Until the early 2000s, research around wearable computing centered on device development, input and output modalities, and context awareness, with the latter representing much of what was covered in ISWC proceedings. Although classic PC-like input and output has practically disappeared, context awareness continues to dominate the research programs, supported by novel sensing approaches, progress in sensor integration, and advances in machine learning. Interaction research is also exploring the use of wearables, further pushing context awareness and demonstrating how wearable computing has matured as a research field such that it's now ready to share its methods and tools. We observe a similar convergence in sensor-based context awareness, which shares methods with fields such as computer vision.

Advances in smart materials over the last few years have just begun to stimulate the wearable computing field in entirely new ways. For example, the skin patches and electronic tattoos that have recently been introduced for physiological monitoring are made of new stretchable, conductive, and skin-compatible materials (see "milestone 5" in the "Beyond Smartphones" sidebar). New smart materials with different elasticity, conductivity, and changeable mechanical properties can provide the basis for new production approaches that could replace the rigid printed circuit boards in wearables, where body contact, form factor, and skin-like properties are key.

Substantial advances in production and process technology are also coming up, which could simplify the development and prototyping of wearable devices. For example, ink-jet printing enables designers to print small-pitch conductive traces for interconnects in flexible electronics. Furthermore, bio-printing, offered by nScrypt (www.nscrypt.com), for example, lets researchers print tissue—possibly with sensing capabilities in the future. Figure 3 shows wearable computing's potential influence on future solutions using a variety of input methods.

OUTLINING FUTURE RESEARCH

So what lies ahead for wearables researchers? We foresee a variety of exciting research challenges and future trends for wearable computing.

BEYOND SMARTPHONES

Wearable computing research recently passed the 20-year mark, having grown from a gathering of like-minded, mostly US-based, researchers into a strong, diverse, and international scientific field.

KEY MILESTONES

Here we provide a summary of key milestones in the development of wearable computing.

1: A Wearable Prediction System
Although the foundations of wearable assistive devices can be seen in early pocket watches of the 16th century, most consider the wearable system by Edward Thorp and Claude Shannon in the 1960s that predicts the outcome of a roulette wheel as a starting point for wearable computers.

2: Digital Hearing Aids
In the 1980s, digital hearing aids entered the market. Although they were imperfect at the time, they addressed a major patient concern and fulfilled even today's definition of wearable computing systems.

3: New Concepts in the 1990s
The modern concept of wearable computing was developed in the early 1990s by Thad Starner and Steve Mann, among others. At a time when PCs were entering offices and homes, Starner and Mann envisioned systems that would always be with the user, integrated into everyday outfits, aware of the physical environment, and thus able to augment human perception.

4: Smartphones
Although various business ideas—including those focused on smart shirts and wrist-worn computers—found niche markets in the late 1990s and early 2000s, it was the advent of smartphones in 2005 that really opened up opportunities for wearable systems. With the emerging mass market, high-volume production of low-power integrated circuits, digital sensors, and systems-on-chip made a variety of wearables viable, including activity trackers and sports watches.

5: Technology Integration
More than a decade after the technological opening through smartphones, we consider the novel technology integration approaches to be the next milestone. Examples include on-body electronic patches,[1] computer-augmented accessories that (more than ever) resemble their traditional counterparts, and novel approaches in smart textiles.[2]

ONGOING RESEARCH

The scientific literature on wearable computing shows a host of application opportunities in addition to many open research challenges. Bruce Thomas's discussion of the "ultimate wearable computer" highlighted the progress from bulky devices to smartphones and diverse application explorations, leading to a multidisciplinary research community of electronics, computer science, art and design, and fashion and materials.[3] Also, one of us (Oliver Amft), along with Attila Reiss, reviewed "real" wearable computers to investigate the persisting challenges for integrating computing, sensors, and interaction in form factors of daily accessories and clothing.[4] The review highlighted the fact that technology integration in real everyday life is essential in many areas to achieving wearer compliance. Currently, there is still a lack of integrated solutions and prototypes, which would make the development path sufficiently repeatable.

Lukasz Piwek and his colleagues reviewed health wearables and described concerns regarding safety, reliability, and security, concluding that primarily users who are already health-aware could benefit.[5] Mary Ellen Berglund and her colleagues observed a recent preference for wrist-worn wearables, warning that the trend could limit creativity in exploring other areas of wearables.[6] And Dan Siewiorek, in a recent retrospective, emphasized ongoing challenges in attention management, wearer comfort, and the powering of wearables.[7]

REFERENCES

1. S. Xu et al., "Soft Microfluidic Assemblies of Sensors, Circuits, and Radios for the Skin," *Science*, vol. 344, no. 6179, 2014, pp. 70–74; https://doi.org/10.1126/science.1250169.
2. J. Cheng et al., "Smart Textiles: From Niche to Mainstream," *IEEE Pervasive Computing*, vol. 12, no. 3, 2013, pp. 81–84; https://doi.org/10.1109/MPRV.2013.55.
3. B.H. Thomas, "Have We Achieved the Ultimate Wearable Computer?" *Proc. IEEE 16th Int'l Symp. Wearable Computers* (ISWC), 2012, pp. 104–107; https://doi.org/10.1109/ISWC.2012.26.
4. A. Reiss and O. Amft, "Design Challenges of Real Wearable Computers," *Fundamentals of Wearable Computers and Augmented Reality*, 2015, pp. 583–618; http://doi.org/10.1201/b18703-27.
5. L. Piwek et al., "The Rise of Consumer Health Wearables: Promises and Barriers," *PLOS Medicine*, vol. 13, no. 2, 2016, e1001953; https://doi.org/10.1371/journal.pmed.1001953.
6. M.E. Berglund, J. Duvall, and L.E. Dunne, "A Survey of the Historical Scope and Current Trends of Wearable Technology Applications," *Proc. 2016 ACM Int'l Symp. Wearable Computers*, 2016, pp. 40–43; https://doi.org/10.1145/2971763.2971796.
7. D. Siewiorek, "Wearable Computing: Retrospectives on the First Decade," *GetMobile: Mobile Computing and Comm.*, vol. 21, no. 1, 2017, pp. 5–10; https://doi.org/10.1145/3103535.3103537.

Resilience and Long-Term Validity

As wearable technology continues to advance on almost every front, it's hard to imagine a single product lasting more than a few months before a better version (in terms of processing, display, or battery life) comes along. The current market is built on the fact that hardware components are cheap and small, making it easy to buy replacements and grow our drawer-collection of "obsolete gadgets."

This is not solely a hardware issue, either. Wearable sensors—and the data they produce—change with every replacement as well, making it hard to design a wearable service that remains in operation beyond several months. We need to develop methods for collecting consistent data with wearables over longer time spans.

WEARABLE COMPUTING

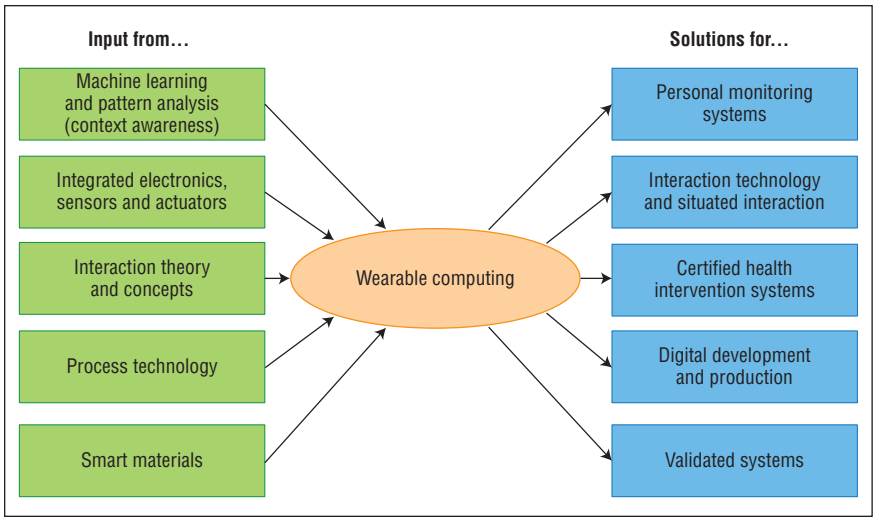


Figure 3. How wearable computing might be influenced by and influence future research.

Benchmarks and Evaluation

Many wearable products are sold as a life-enhancing or health-improving accessory that, when worn, promise a variety of benefits. Good examples include fitness trackers that display caloric expenditure and sleep trackers that display the wearer's different sleep phases during the night, but how accurate are these estimates? It's clear that many devices can easily be fooled and that most wearables on the market today have not undergone thorough (clinical) testing—such as those for sleep estimation.[7]

For each wearable application, from step counters to HMDs, we'll need to find out what systems and algorithms perform best and which evaluation method is the most rigorous for the job. However, we can't foresee all situations, so we must also identify system limits. Wearables need to declare side effects, co-conditions, and limitations, similar to what's done for medications.

Sustainable Software

Many current wearables have been designed from scratch, including home-grown software. The few operating systems for wearables form a very messy landscape that lacks developer guidance. There are many design options complicating the software for wearables, from custom microcontroller code to Arduino sketches and Android Wear Apps, depending on available system resources and applications. Although wireless Internet access even in the smallest devices lets developers run online updates and delegate computation to a cloud, it's likely insufficient to scale wearable software development.

Wearable system software needs (standard) frameworks that fit an increasing pool of diverse devices. Moreover, we need software frameworks for critical applications in certified health intervention systems—wearables for providing time- or measurement-dependent medication, for example.

Wearer Compliance

Whether it's new smart eyeglasses or a body patch, technology integration has been a highly underestimated feature of wearable system development and user compliance, particularly for personal monitoring and intervention applications. So far, few products go beyond the concept of the carry-on gadget to truly integrate computing into practical accessories, garments, and so on.

We need research into integration methods and tools, as well as demonstrations. In the near future, a multitude of widely diverse mobile and wearable systems will reside in the personal space of every user, making generic interoperability the critical factor for wearer compliance. Such interoperability must scale beyond individual applications that share, for example, a text message on different devices.

Market Size vs. Development Effort

Addressing low-volume markets is a key element to making wearables successful in the next decade. There are already research efforts underway to develop wearable technology that lets businesses scale across small applications. For example, the EU-funded SimpleSkin project (simpleskin.org) developed fabric material that lets developers realize different sensor functions, while textile manufacturers could mass-produce the generic fabric in standard processes.[8] "Sensor-ready" garments thus could be produced, with electronics or even software determining the functionality.

Moreover, novel smart materials and printing methods are helping designers and researchers not only prototype ideas but also develop digital production processes. Digital production can be used to help realize the design[9] or function[10] of personal products.

Between the hope and hype, wearable computing has always aimed for more personal, intimate computing, and in looking ahead, we foresee many interesting research opportunities. Wearables have transitioned from single, general-purpose computers to a set of devices, each with its own challenges. Consequently, core wearable research must grow beyond the well-established topics in context-awareness, sensor, and interaction

research, providing new methods and tools for truly integrating computing with the body and creating real-world impact.

REFERENCES

1. O. Amft and P. Lukowicz, "From Backpacks to Smartphones: Past, Present and Future of Wearable Computers," *IEEE Pervasive Computing*, vol. 8, no. 3, 2009, pp. 8–13; https://doi.org/10.1109/MPRV.2009.44.
2. Ö. Yürür et al., "Context-Awareness for Mobile Sensing: A Survey and Future Directions," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, 2016, pp. 68–93; https://doi.org/10.1109/COMST.2014.2381246.
3. D. Ledger and D. McCaffrey, "Inside Wearables: How the Science of Human Behavior Change Offers the Secret to Long-Term Engagement," white paper, Jan. 2014; https://blog.endeavour.partners/inside-wearable-how-the-science-of-human-behavior-change-offers-the-secret-to-long-term-engagement-a15b3c7d4cf3.
4. K. Van Laerhoven and P.M. Scholl, "Interrupts Become Features: Using On-Sensor Intelligence for Recognition Tasks," *Embedded Engineering Education*, 2016, pp. 171–185.
5. O. Amft et al., "Making Regular Eyeglasses Smart," *IEEE Pervasive Computing*, vol. 14, no. 3, 2015, pp. 32–43; https://doi.org/10.1109/MPRV.2015.60.
6. T. Shamma, "Google Glass Didn't Disappear. You Can Find It on the Factory Floor," NPR, 18 Mar. 2017; www.npr.org/sections/alltechconsidered/2017/03/18/514299682/google-glass-didnt-disappear-you-can-find-it-on-the-factory-floor.
7. K. Russo, B. Goparaju, and M.T. Bianchi, "Consumer Sleep Monitors: Is There a Baby in the Bathwater?" *Nature and Science of Sleep*, vol. 7, 2015, pp. 147–157; https://doi.org/10.2147/NSS.S94182.
8. J. Cheng et al., "Textile Building Blocks: Toward Simple, Modularized, and Standardized Smart Textile," *Smart Textiles*, 2017, pp. 303–331; https://doi.org/10.1007/978-3-319-50124-6_14.
9. M. Gannon, T. Grossman, and G. Fitzmaurice, "ExoSkin: On-Body Fabrication," *Proc. 2016 CHI Conf. Human Factors in Computing Systems*, 2016, pp. 5996–6007; https://doi.org/10.1145/2858036.2858576.
10. F. Wahl et al., "Personalizing 3D-Printed Smart Eyeglasses to Augment Daily Life," *Computer*, vol. 50, no. 2, 2017, pp. 26–35; https://doi.org/10.1109/MC.2017.44.

Oliver Amft is a full professor for eHealth and mHealth at the Institute of Medical Informatics of the Friedrich-Alexander University of Erlangen-Nürnberg, Germany. Contact him at amft@computer.org.

Kristof Van Laerhoven is a full professor of ubiquitous computing at the University of Siegen, Germany. Contact him at kvl@eti.uni-siegen.de.

This article originally appeared in IEEE Pervasive Computing, *vol. 16, no. 4, 2017.*

Editor: **Christof Ebert**
Vector Consulting Services
christof.ebert@vector.com

Practices and Technologies in Computer Game Software Engineering

Walt Scacchi

Computer games are at the forefront of software engineering. Games software engineering, although often neglected in curricula, poses huge challenges such as time to market, complexity, collaborative development, and performance. Game development ranges from entertainment, with games being more sophisticated and complex than movies, to serious games used for education in universities and industry. Here, Walt Scacchi introduces us to computer game software engineering with technologies and a hands-on case study. I look forward to hearing from both readers and prospective column authors. —*Christof Ebert*

COMPUTER GAMES ARE RICH, complex, and frequently large-scale software applications. They're a significant, interesting, and often compelling domain for innovative research in software engineering (SE) techniques and technologies. Computer games are progressively changing the everyday world in many positive ways. Game developers, whether focusing on entertainment market opportunities or game-based applications in nonentertainment domains such as education, healthcare, defense, or scientific research (that is, serious games), thus share a common interest in how best to engineer game software.

Here, I examine aspects of contemporary computer game SE. To supplement this description, the sidebars present a brief look at game development technologies and a case study in applying computer game SE techniques.

What Game Developers Should Know

There are many different and distinct types of games, game systems, and gameplay, much like there are many dif-

 Published by the IEEE Computer Society

GAME DEVELOPMENT TECHNOLOGIES

Computer games might well be the quintessential domain for computer science and software engineering R&D. Why? Modern multiplayer online games must address core issues in just about every major area of computer science research and education. Such games entail the development, integration, and balancing of software capabilities drawn from many areas. These areas include algorithm design and complexity, AI, computer graphics, computer-supported cooperative work or play, database management systems, human–computer interaction and interface design, OSs and resource or storage management, networking, programming- or scripting-language design and interpretation, and performance monitoring. Few other software application arenas demand such technical mastery and integration skill. Yet game development is expected to rely on such mastery and provide gameplay that most players find satisfying, fun, and engaging.

Computer games are thus an excellent domain for which to research and develop new ways and means for software engineering. Accordingly, there are many kinds of commercial or open source software development kits, engines, services, and approaches for producing, delivering, and evolving computer games of different genres.

Table A provides a small sample of possibilities that serve as a starting point. Interested software professionals and students should also go online and search for the software technologies that best match their interests in, constraints on, and enthusiasm for developing computer games.

TABLE A

Some technologies for computer game software engineering.*

Game SDK or game engine motif	Commercial examples	Game development features	Open source software alternatives	Development or target platforms	Common game genres
HTML5 or web	Construct 2, GameSalad	Rule-based, UI event processing	EaselJS, GDevelop, Kiwi.js, Phaser	Computers or devices with web browsers	2D web-browser-based games
Game-genre-specific	Adventure Game Studio, Minecraft, RPG Maker, SAGE	Genre-based UI, user experience	Freeciv, Minetest, Ren'Py, Quest, Stratagus	Networked PCs	Adventure and role-playing games, real-time strategy games, visual novels
Library, framework, or runtime environment	GameMaker, libGDX, Microsoft XNA	Game programming primitives, open APIs	ANX, Cocos2d, OGRE	PCs	2D or 3D single-user or multiuser games
Game modding (modifying)	Half-Life, Neverwinter Nights, Unreal	Modification or reuse of working games	Doom, Quake, Quake Arena	Networked PCs	Depends on the originating game
Game IDE	CryEngine, Source, Unity, Unreal Engine, UDK	Production quality workflow	Blender, Torque 3D	PCs	Mass-market games, 3D first-person action and shooter games
Cloud-based or MMOG service	Amazon Lumberyard, Facebook, Steam, Twitch	Scalable services and secure e-commerce	OpenSimulator, Worldforge	PCs, consoles, Internet-connected smartphones	eSports, free-to-play games, MMOGs

SDK stands for software development kit; MMOG stands for massively multiplayer online game.

THE *BEAM* GAME

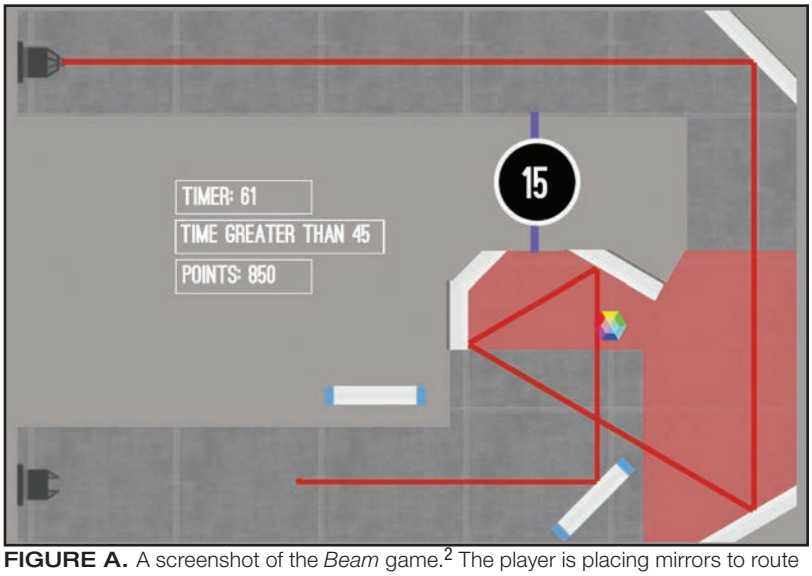


FIGURE A. A screenshot of the *Beam* game.[2] The player is placing mirrors to route an optical beam from the upper-left source to the lower-left target.

Case studies can help elucidate how to apply current practices and technologies in computer game software engineering (SE). In five previous case studies, I focused on software reuse and game repurposing.[1] Here, I focus on a study investigating whether a STEM-literate high-school student who was an avid gamer could learn basic SE concepts and practices.[2] (STEM stands for science, technology, engineering, and math.)

First, I had a student, Mark Yampolsky, identify a new game he would develop and demonstrate that could help his fellow students learn a challenging STEM topic—beam physics. Beam physics is central to modern physics—for example, in the design of simple optics as presented in high-school physics and as the basis for advanced particle accelerators and quantum teleportation devices.

The study proceeded in an agile incremental manner whereby the student identified functional or nonfunctional requirements that could be translated into game mechanics that could be realized using an event-driven, rule-based-system architectural framework. Such a framework is supported, for example, by 2D software development kits, such as Construct 2 and GameSalad (see the sidebar "Game Development Technologies"), that support rapid prototyping of interactive media or games for deployment with web browsers. By starting with an architectural framework and a software development kit, rather than a programming language, the student could focus on identifying play input and display output events and event types (mouse clicks, object drag-and-drop, game start and end, and so on). These events and event types could then trigger reactive rules that would update the gameplay (display) space and points earned (or lost).

Figure A shows a screenshot of the resulting *Beam* game. *Beam*'s multilevel goals entail finding either the shortest path or a path routed to achieve certain outcomes. One outcome could be to minimize where to place and how to orient (rotate) optical devices such as mirrors and lenses to ensure beam routing from the source to the target.

As the student developed *Beam*, issues or tradeoffs surfaced regarding how best to structure and refactor different rule sets. Overall, this version of *Beam* uses seven rule sets entailing more than 180 event-update rules.[2] Developing a new game with a rule-based system such as *Beam*'s presents a classic software engineering problem: refining and evaluating architectural alternatives. The *Beam* case study illustrates how SE concepts can mediate computer game development.

References

1. W. Scacchi, "Repurposing Gameplay Mechanics as a Technique for Developing Game-Based Virtual Worlds," *Computer Games and Software Engineering*, K.M. Cooper and W. Scacchi, eds., CRC Press, 2015, pp. 241–260.
2. M. Yampolsky and W Scacchi, "Learning Game Design and Software Engineering through a Game Prototyping Experience: A Case Study," *Proc. 5th Int'l Workshop Games and Software Eng.* (GAS 16), 2016, pp. 15–21.

ferent and distinct types of software applications, information systems, and systems for business. Understanding how to develop games for a particular platform requires identifying what types (genres) of games are commercially available. Popular genres include action games, first-person shooters, adventure games, role-playing games, fighting games, racing games, simulation games, sports games, strategy and real-time strategy games, music and rhythm games, parlor games (board and card games), puzzles, educational or training games, and massively multiplayer online games (MMOGs).

This suggests that knowledge about one type of game (for example, role-playing games such as *Dungeons & Dragons*) doesn't subsume, contain, or provide the gameplay experience, user interface, gameplay scenarios, or player actions in other types of games. So, being skilled in software development for one type of game (for example, a turn-taking role-playing game) doesn't imply ability or competence in developing software for another type of game (for example, a continuous-play twitch or action game). (Twitch games test players' reaction time.) This is analogous to saying that just because a developer is skilled in payroll and accounting software doesn't mean that the developer is skilled in enterprise database management or e-commerce systems. The differences can be profound, and the developer skills and expertise can be narrowly specialized.

Conversely, common games such as card or board games raise the obvious possibility of developing one game engine that can be shared or reused to support multiple games of a single type—a game product line. For example, checkers and chess are played on an 8 × 8 checkerboard, and player actions in both games are basically the same (picking up a piece and moving it to a square allowed by the game rules), even though the game pieces, rules, and gameplay differ.

So, being skilled in developing a checkers game can suggest having the skill to develop a similar game such as chess, especially if both games can use the same game engine (the game's runtime environment[1]). However, this is likely only when the game engine allows for distinct sets of game rules and the distinct appearance of game pieces. That is, the game engine must be designed for reuse or extension, which isn't always an obvious engineering choice and which increases game engine development's initial cost. Subsequently, developing the software for different kinds of games that are of the same type or use the same game engine requires a higher level of technical skill and competence than designing an individual game of a given type.

Understanding how game software operates on a game platform requires understanding gameplay and player actions. Understanding a game platform entails understanding an embodied game device (for example, the Apple iPhone, Microsoft Xbox One, or Nintendo Game Boy) and the internal software runtime environment that enables its intended operation and data communication. Developers must also understand the game's architectural structure, how the game functions, how the player controls the game device through its interfaces (keyboard, buttons, stylus, and so on) and video and audio output, and how the interfaces and output affect game data transmission and reception in a multiplayer game network.

Requirements Engineering

Understanding how best to elicit and engineer computer game requirements is unsurprisingly a fertile area for computer game SE research and practice, much as it has been for mainstream SE. However, relatively few game development approaches employ SE requirements development methods such as use cases and scenario-based design.

Many game developers in industry have reviewed the informal game postmortems that first appeared in *Game Developer* magazine in the 1990s and currently appear on Gamasutra.com. Austin Grossman's edited collection of 50 or so postmortems revealed common problems in game development projects.[2] These problems cluster around project software and content development scheduling, budget shifts (generally budget cuts), and other nonfunctional requirements that drift or shift in importance during game development projects. None of this should be surprising to experienced SE practitioners or project managers, although it might be new knowledge to SE students and new self-taught game developers.

Similarly, software functional requirements for computer games most often come from the game producers or developers, rather than game players. However, nonfunctional requirements (for example, the game should be fun to play but hard to master, and it should run on mobile devices and the web) dominate computer game development and thus marginalize the systematic engineering of functional game requirements. Nonetheless, the practice of openly publishing and sharing postproject descriptions and hindsight rationalizations might prove valuable as another kind of

empirical SE data for further study, as well as something to teach and practice in SE education project courses (as I discuss later).

Architecture Design

Computer games often represent configurations of multiple software components, libraries, and network services. Consequently, computer game software must have an architecture,[1] and ideally this architecture is explicitly represented and documented as such. The architecture might be proprietary and thus protected by its developers as intellectual property covered by trade secrets and user license agreements. However, there's substantial educational value in having access to such architectural renderings as a means for quickly grasping key system design decisions and the modules that participate in gameplay event processing. This is one reason for interest in games that are open to modding (modifying) or free or open source software extensions.

But other architecture concerns exist. For example, networked multiplayer games employ at least four kinds of software or information architectures:

- the static and dynamic runtime architectures for a game engine,
- the architecture of the game development frameworks or software development kits that embed a game's development architecture together with its game engine,
- the architectural distribution of software functionality and data-processing services for the games, and
- the informational and geographical architecture of the game levels as designed play spaces.

Game system architecture can have different configurations. For instance, for the architectural distribution of software functionality and data-processing services, five system configurations are common:

- a single server for multiple interacting or turn-taking players;
- peer-to-peer networking;
- client-server networking for user clients and play-space data exchange servers;
- distributed, replicated servers for segmented user play sessions through sharding; and
- cloud-based game content delivery, a play event or score database, a game forum and chat or voice services, analytics, and commerce services.

In contrast, the focus on computer games as interactive media often sees little or no software architecture as being relevant to game design. This is especially true in the design of games that assume a single-server architecture or PC game runtime environment, rather than in game systems in which distributed services must be provided and system architecture is critical.[3] My point here is not to focus on the gap between game design and game software (architecture) design as consisting of alternative views but to draw attention to the need for computer game SE to finds ways to span the gap.

Playtesting

Computer games that potentially involve millions of players will consistently and routinely manifest bugs. Again, this is part of the puzzle of any complex SE effort; games are no exception. However, the user experience, and thus user satisfaction, might be key to driving viral social media that help promote game sales and adoption. So, paying close attention to bugs and features in game development and usability might be key to a game development studio's economic viability.

Furthermore, as decades of developer experience with large-scale software applications have shown, most users can't articulate their needs or requirements in advance but can assess whether what's provided meets their needs. So, the development of large-scale, high-cost computer games that take years to produce and person-decades (or person-centuries) of developer effort could change from monolithic product development lifecycles to ones that are much more agile, incremental, and driven by user feedback based on progressively refined or enhanced game version (or prototype) releases.

Early and ongoing game playtesting will likely become a central facet of computer game SE, as will tools and techniques for collecting, analyzing, and visualizing game playtesting data.[4] This is one area in which computer game SE might substantially diverge from early computer game development approaches, much like agile methods often displace the waterfall method. So, computer game developers, much like mainstream software engineers, are moving toward incremental development, rapid release, and user playtesting to drive new product versions.

Reuse and Repurposing

Systematic software reuse could be considered in multiple SE activities (requirements, architecture, design, code, build and release, and test cases) for a single game or game product line. For example, many successful games become franchises

through the production and release of extension packs (that provide new game content or play levels) or sequels (for example, *Quake II* and *Quake III*). Whether or how to employ software-product-line concepts and methods in widespread computer game business models is unclear and underexplored. A new successful computer game product might have been developed and released in ways that sought to minimize software production costs. In that way, the software company could avoid the investment necessary to make the software architecture reusable and extensible and the component modules replaceable or upgradable without discarding much of the software developed up to that point. This means that SE approaches to computer game product lines might be recognized in hindsight as missed opportunities, at least for a given game franchise.

Reuse could reduce development costs and improve quality and productivity, as it often does in mainstream SE. Commercial computer game development often relies on third-party software components (for example, game engines) or middleware products (for example, AI libraries for nonplayer characters), which are perhaps its most visible forms of software reuse. Game software development kits, game engines, procedural game content generation tools, and game middleware services are all undergoing active R&D in industry and academia. (For additional details, see the sidebar "Game Development Technologies.")

Game engines are perhaps the best success story for computer game software reuse. However, commercial game development studios and independent game developers sometimes avoid adopting commercially available game engines because they believe that the engines' characteristic patterns or mechanics would overly restrict their game's development patterns or gameplay mechanics. If that happened, players might feel that such games offer a derivative play experience rather than an original one.

Nevertheless, the future of reusable game development techniques might include catalogs of patterns or antipatterns for game requirements, architecture and design patterns for game software product lines, and online repositories of reusable game assets organized by standardized ontologies. You can find other approaches to reuse in free or open source software for computer game development and in game software repurposing. Additional sources are the emerging AI or computational-intelligence methods for automated or semiautomated content generation and level design that appear in *IEEE Transactions on Computational Intelligence and AI in Games*.

Runtime Services and Scalability Infrastructures

Computer games range from small-scale, standalone smartphone apps to large-scale, distributed, real-time MMOGs. They're sometimes played by millions of users, so large-scale, big data approaches to gameplay analytics and data visualization become essential for engineering sustained gameplay and deployment support.[4] Prior knowledge of the development of multiplayer-game software systems and networking services might be essential for SE students learning to develop social or mobile MMOGs.

To engage players and promote the adoption and ongoing use of such large and upward- or downward-scalable applications, computer game SE techniques have significant potential but require further articulation and refinement. Questions are just emerging regarding the integration of game playtesting and user play analytic techniques with large-scale, big data applications. Similarly, the question of how best to design back-end game data management capabilities or remote middleware gameplay services also points to SE challenges for networked-software-systems engineering, as has been recognized within the history of networked-game software development.

> Computer games are a growth-oriented domain at the forefront of software engineering.

The ongoing emphasis on computer games that realize playful, fun, social, or learning experiences across gameplay platforms leads naturally to interdisciplinary approaches to computer game SE. In such approaches, psychologists, sociologists, anthropologists, and economists could provide expertise on defining new gameplay requirements

This article originally appeared in IEEE Software, *vol. 34, no. 1, 2017.*

and experimental designs to assess user experience quality.

Furthermore, the emergence of online fantasy sports, along with eSports (for example, team-versus-team or player-versus-player competitions for prizes or championship rankings) and professional-level tournaments for games such as *Counter-Strike*: *Global Offensive*, *Dota 2*, or *League of Legions*, points to other computer game SE challenges. These challenges include cheat prevention, latency equalization, statistical scoring systems, complex data analytics,[4] play data visualization, and streaming video broadcasts (for example, through MLG.TV or Twitch) that support balanced game systems with performance (monitoring) equalized for professional-level tournaments. Similarly, the emergence of games developed for VR or augmented-reality UIs and user experiences, such as *Pokemon Go*, suggests opportunities for engineering game software that exploits the new devices and sensors available through the UI, along with the engagement affordances these user experiences offer.

Computer Games and SE Education

SE faculty who teach project-oriented SE courses increasingly have sought to better motivate and engage students through game software development projects, as most CS students are literate in computer games and gameplay. The use of game development projects for SE capstone project courses is now widespread.

For educators teaching software engineering education (SEE) project courses, it might be valuable for their students to become engaged with computer game SE through exposure to the history of computer game software development or by reviewing recent advances in computer game SE fundamentals and SEE.[5] For example, C. Shaun Longstreet and Kendra Cooper, Alf Wang and Brian Wu, and others have incorporated contemporary SE practices such as software architecture and model-driven development within game-based SE project courses.[5] In addition, Tao Xie and his colleagues at Microsoft and others have developed game-based software-testing competitions.[5]

Similarly, whether to structure projects as massively open online courses or competitive, interteam game jams also merits consideration. Such competitions can be testbeds for empirical SE (or SEE) studies—for example, when project teams consist of students who take on different development roles, with each team comprising members with comparable roles and experience.

Computer game SE is a growth-oriented domain at the forefront of software engineering. A new generation of software engineers will take on the technical challenges involved in facilitating the development, deployment, and evolution of computer games as complex software systems that support global cultural-media practices. Readers interested in further exploring computer game SE practices and technologies might also find the cited references helpful for learning about game design practices,[3] common approaches and mistakes in game production,[2] game engine and runtime environment architectures,[1] gameplay data analytics and visualization techniques,[4] or recent advances in computer game SE research and education.[5]

References

1. J. Gregory, *Game Engine Architecture*, A K Peters / CRC Press, 2009.
2. A. Grossman, ed., *Postmortems from* Game Developer*: Insights from the Developers of* Unreal Tournament, Black and White, Age of Empires, *and Other Top-Selling Games*, Focal Press, 2003.
3. J. Schell, *The Art of Game Design: A Book of Lenses*, CRC Press, 2008.
4. M. Seif El-Nasr, A. Drachen, and A. Canossa, eds., *Game Analytics: Maximizing the Value of Player Data*, Springer, 2013.
5. K.M. Cooper and W. Scacchi, eds., *Computer Games and Software Engineering*, CRC Press, 2015.

WALT SCACCHI is a senior research scientist and research faculty member at the Institute for Software Research and the director of research at the Institute for Virtual Environments and Computer Games, both at the University of California, Irvine. Contact him at wscacchi@ics.uci.edu.

Editor: **Daniel E. O'Leary,** University of Southern California, oleary@usc.edu

Emerging White-Collar Robotics: The Case of Watson Analytics

Daniel E. O'Leary, *University of Southern California*

Increasingly, systems are being built that might be called "white-collar robots." If you watch TV news, you likely would define a white-collar robot as something that takes jobs from white-collar human workers. That perspective seems unnecessarily limiting. So, let's define a white-collar robot as a set of computer-based capabilities that performs a task or a set of tasks historically done by white-collar workers, but without the need for human intervention. White-collar robots automate particular tasks. A human decides what task a robot will do and then chooses a robot to do that task. As in manufacturing settings, white-collar robots might be part of a process, providing information inputs or outputs to others, whether people or robots. This definition is consistent with robots used for manufacturing and does not require that the robot have "consciousness." Instead, the robot just does things that need to be done. In general, the robot is likely to be able to do some things better than the person, but the person is likely to do some things better than the robot.

What tasks might a white-collar robot do? On the basis of robots in other settings, we would probably expect a white-collar robot to perform certain tasks without the human specifying all of the details every time. For example, we might give data to such a robot and ask it to "analyze it, and I will ask you some questions later—come back to me with some suggestions as to what you think the data says." That description is exactly what is being done by some analytics software. Increasingly, even those with limited statistics or analytics knowledge can use easy and smart analytic software (robots) to analyze data. These robots are part of an effort to "leave no manager behind" in a "data-driven world" simply because those managers don't know much about analytics. One such effort, Watson Analytics (www.ibm.com/analytics/watson-analytics), provides active software that helps any user investigate data. Upload data to the cloud, and Watson Analytics will analyze the data quality, provide initial analysis, and prod you to consider different combinations of variables.

Watson Analytics is distinct from Watson Cognitive, which gained fame from capabilities used on the TV show *Jeopardy!* The latter's capabilities focus on text, natural language, and other related applications. As a result, much of the discussion of Watson's capabilities is bifurcated into two seemingly independent pieces—cognitive and analytics.

Easy and smart analytics may be of great interest to many users. To test the appeal of Watson Analytics, I had roughly 90 master's degree students use the software as part of a class introducing them to information systems and analytics. The students had limited experience with statistics and analytics, but by the end of the class, about 90 percent of them identified Watson Analytics as one of the most interesting, important, and potentially useful topics they studied.

To begin to understand some of the potential strengths and limitations of such software, first see the "Active versus Passive Software" sidebar. I will also review Watson Analytics' capabilities, analyze two sets of data in the form of case studies, and summarize some of the software's strengths and limitations.

Watson Analytics' Capabilities

Watson Analytics requires that the user (or someone helping the user) negotiate a data upload to the cloud. After the data is uploaded, Watson Analytics provides starting points for the user to analyze the data. These starting points are a sequence of questions that

2469-7087/18/$33.00 © 2018 IEEE Published by the IEEE Computer Society June 2018

Active versus Passive Software

Historically, software largely has been passive. For example, in the case of statistical software, people input data and then choose which models to analyze the data. In contrast, Watson Analytics is active software: the software makes recommendations and does analysis without needing the user to specify the models.

Increasingly, software is becoming active, and people are taking "orders" from it. For example, it is not unusual to have direction or map software guide us when we don't know where we are going. Thus, people are beginning to expect software to make those recommendations, particularly if they don't have any particular knowledge, are unsure about the set of issues, or don't want to force themselves to think about the issues that the software can do (for example, they want to relax and not concentrate, or concentrate on something else).

Generally, active software makes assumptions as to where or what the user wants or needs. Active software is likely to make predictions about what would be useful to the user. Active software may be autonomous and function largely without the user or its inputs. Active software is likely to be goal oriented. White-collar robots generally will be active software.

Table 1. What drives satisfaction?

Independent variable sets*	Strength (%)	R-square	ANOVA statistical significance
1. Type of travel	37	0.3551	< 0.0001
2. Age and status	33	0.1884	< 0.0001
3. No. flights per annum and year of first flight	21	0.0904	< 0.0001
4. Price sensitivity and age	18	0.0496	< 0.0001
5. No. loyalty cards and age	17	0.0380	< 0.0001
6. Status	17	0.1421	< 0.0001
7. Age	14	0.0380	< 0.0001
8. Status and type of travel	Model not tested or found in Watson	0.4221	< 0.0001
9. Flights per annum and status	Model not tested or found in Watson	0.1914	< 0.0001

*The dependent variable is satisfaction.

the system has developed on the basis of the data—for example, "What drives X?" "What is a predictive model of Y?" and "What is the trend of Y and Z?"

The data format is important to Watson Analytics. Watson does not work well when there are more than two dimensions to the data, say, with row or nested headings, or both. As an example, column heading by years is appropriate, but column headings that break the data into multiple layers—such as year, month, revenue, and quantity—will not work well.[1] In particular, the system works best with data that can be placed in a classic flat file. However, unlike much statistical software, Watson can analyze variables that are categorical or numeric. Watson Analytics provides an analysis of the data quality that indicates "how ready the data is for analysis."[1] In computing the data quality for each field, Watson Analytics apparently considers several factors, including missing values, outliers, symmetry, skewness, and imbalance. The overall data quality score is an average of the data quality score of every variable in the dataset. Furthermore, the existence of a data quality score emphasizes the importance of the data quality to the user. This quality number gives the user direct information about the data's veracity.

Watson Analytics does not appear to use traditional statistical methods such as correlation and regression analyses, but it does use decision trees to determine which variables appear to influence others and to determine the "strength" of those relationships. I analyze these and other capabilities using two case studies.

Case Study 1: Customer Satisfaction

I analyzed a subset of data used by IBM to illustrate Watson Analytics to different groups.[2] The dataset had 499 randomly chosen observations related to customer satisfaction in air travel. The variables included type of travel, age, status, number of flights per annum, price sensitivity, number of loyalty cards, gender, and arrival delay.

As part of its initial analysis, Watson Analytics analyzed the data and generated 10 questions, including the following:

- What drives satisfaction?
- How are the variables of flight distance and flights per annum related?
- What is the relationship between arrival delay and departure delay?
- What is the predictive model of satisfaction?

Because my interest was in the variable "satisfaction," two of the questions that Watson asked were of direct interest. I first investigated the question, "What drives satisfaction?" Watson Analytics generated seven combinations

Table 2. Predictive model for satisfaction.

Independent variable sets*	R-square for entire sample	Sample size for model
Status, type of travel, and no. loyalty cards	0.4226	156
Status, type of travel, arrival delay, and gender	0.4483	138
Status, arrival delay, and type of travel	0.4337	91
Arrival delay and type of travel	0.3665	114

*The dependent variable is satisfaction.

of one and two variables, listed in Table 1 (that is, variable sets 1 through 7). In addition, I developed nine regression equations (for those seven sets and two additional sets of variables) to compare Watson's strength measure and a statistical measure, R-square. Table 1 summarizes the results. Unfortunately, the system did not find two of the three sets of variables listed with the largest R-square measures (variable sets 8 and 9), although they are combinations of other variables considered by Watson.

For this data, the measures of strength and R-square are statistically significantly correlated at 0.899, which is statistically significant at better than 0.006, indicating a close relationship between those two variables. However, it is clear there are some differences between the two measures for this data, particularly with variable set 5.

All of the variable sets had regression coefficients that were statistically significant at better than 0.05. However, the coefficients on "year of first flight" and "number of loyalty cards" were not statistically significant. In addition, age, number of flights per annum, and price sensitivity had negative coefficients, and thus were negatively related to satisfaction. I would expect users to be interested in knowing whether variables were positively or negatively related to satisfaction, but unfortunately, Watson Analytics did not provide that information.

I also investigated the question, "What is the predictive model of satisfaction?" (see Table 2). As part of building an overall model, Watson Analytics fits different models to different parts of the data. For this data, using decision trees, the system built four models, with corresponding subsamples of 156, 138, 91, and 114 observations. I built regression models for the entire sample on the basis of the four models and included the corresponding R-square for those models. Unlike in Table 1, the number of variables was not limited to one or two. Watson Analytics' measures of strength are not given for each of the models and corresponding subsamples, but only for the entire set of models. In this case, it was 46.1 percent, which is greater than any of the one- and two-variable models in Table 1.

The decision-tree models are based on specific values of the particular variables (for example, number of loyalty cards >2). Thus, using Watson, it is not clear whether the variables are positively or negatively related to satisfaction. Furthermore, in the predictive model, each submodel is for only subsets of the variables. As a result, if actions must be taken for the entire dataset, it is unclear which model to use. However, the prediction model does generate what ultimately seem to be some sets of variables that result in better regression models for the data, and could clearly be used to guide further analysis. Finally, those different models offer different potential "global" strategies.

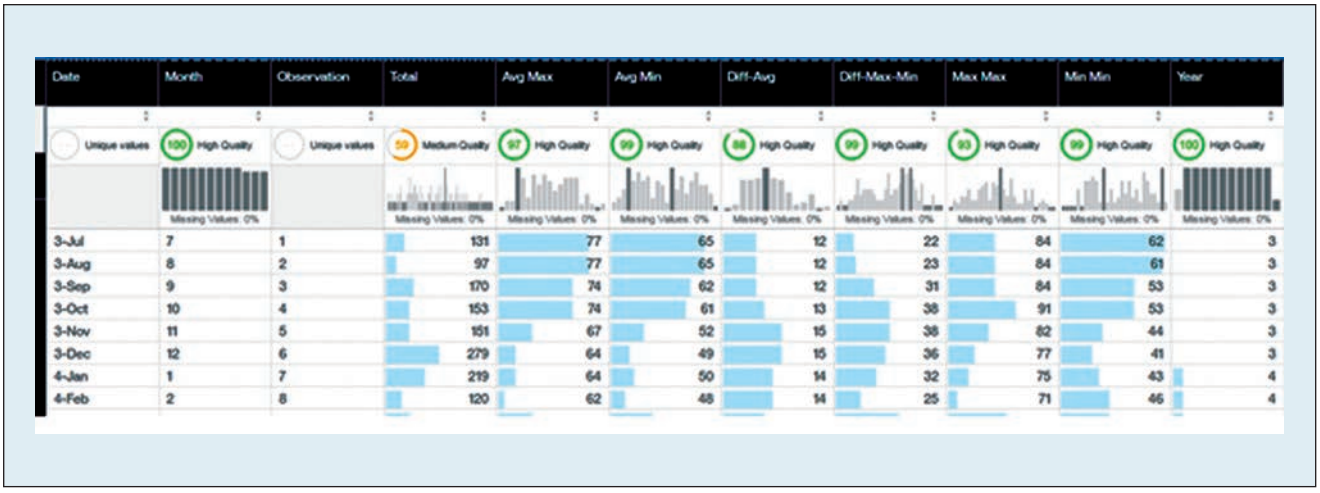
Case Study 2: Water Leak Data

As a second case study, I uploaded data from a study of the impact of temperature on the total number of water leaks in Los Angeles.[3] The data included the observation number, year, month, total number of leaks in the month (total), and several monthly temperature measures: minimum (min), maximum (max), average (avg), difference between the maximum and minimum (diff-max-min), and difference between the monthly average maximum and minimum (diff-avg). Figure 1 summarizes some of the data, along with data quality and reliability measures on each variable. In addition, the system provides visualizations of the data, first in an overview of the data as a whole, and then for individual observations.

Watson Analytics generates a data reliability measure that can help users consider the data's quality before they make assertions about the data. The system notes that one of the data elements is a "unique value" and does not assess it (it is an index value). The data for the number of the month is given a score of 100, whereas "total" is given a score of 59. Although users don't directly receive the reasons for the data's low reliability, they could conceivably use the ratings to massage the data to generate a higher reliability rating, which could be appropriate and could drive different decisions or data. For example, the user could remove the more extreme observations.

After the data was input into Watson Analytics, the system came back with the following (starter) questions:

- What drives diff-avg?
- What is the predictive model for diff-avg?
- How do the variables of diff-avg compare by month?
- How are the values of diff-avg and total related?
- What are the most common values of month?



Date	Month	Observation	Total	Avg Max	Avg Min	Diff-Avg	Diff-Max-Min	Max Max	Min Min	Year
	Unique values	100 High Quality	Unique values	59 Medium Quality	97 High Quality	99 High Quality	88 High Quality	99 High Quality	93 High Quality	99 High Quality	100 High Quality
		Missing Values: 0%		Missing Values: 0%	Missing Values: 0%	Missing Values: 0%	Missing Values: 0%	Missing Values: 0%	Missing Values: 0%	Missing Values: 0%	Missing Values: 0%
3-Jul	7	1	131	77	65	12	22	84	62	3
3-Aug	8	2	97	77	65	12	23	84	61	3
3-Sep	9	3	170	74	62	12	31	84	53	3
3-Oct	10	4	153	74	61	13	38	91	53	3
3-Nov	11	5	151	67	52	15	38	82	44	3
3-Dec	12	6	279	64	49	15	36	77	41	3
4-Jan	1	7	219	64	50	14	32	75	43	4
4-Feb	2	8	120	62	48	14	25	71	46	4

Figure 1. Water leak data quality.

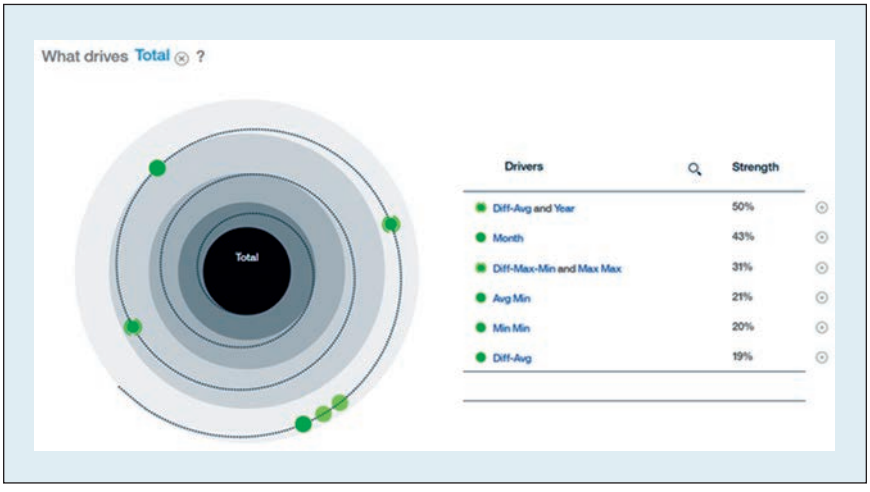


Figure 2. Initial Watson analysis of "total" (the number of leaks).

Unlike in the first case study, in which the system found our particular variable of interest (satisfaction), in this case, limited attention is given to "total" (that is, the number of water leaks) and the relationships with temperature. Instead, these questions show primary interest in diff-avg, rather than our concern for the total. Some of Watson Analytics' questions would probably rarely be asked by a human investigator (for example, "What are the most common values of month?"). This illustrates that Watson Analytics apparently does not employ any semantic understanding of the variables. As a result, its findings do not consider the semantic reasonableness of the relationships generated. This can result in apparent surprise findings, both interesting and uninteresting.

If the system does not generate the questions that the user wants answered, the user can generate new questions. Thus, I asked the system, "What drives total?" and "What is the predictive model of total?" In response to the first question, the system created an interesting visualization of a spiral graph (see Figure 2). As in the first case study, the system provided multiple sets of one and two variables in response to the question. The variables are listed by a measure of "strength."

Table 3 summarizes the Watson strength measures and the R-square values from the regression models built with the corresponding set of variables. Unlike in the first case study, strength and R-square are not statistically significantly correlated for this data.

I also analyzed the findings associated with the question, "What is a predictive model of total?" Watson Analytics developed a decision-tree model with a predictive strength of 10.9 percent that employed only a single variable diff-avg. In previous work,[3] I built regression models with the month and year as control variables. Given those two control variables, the best three-variable model was with diff-average, a model with an R-square of 0.3078, and not found by Watson. Whether or not the control variable approach was appropriate, it does not seem that Watson automatically recognizes the notion of a control variable. In addition, I also found an overall best model of year, month, min-min, max-max, and diff-avg, with an R-square of 0.3513, which was different than Watson's predictive model.[3]

Unfortunately, Watson Analytics does not provide information as to whether the variables are positively or negatively related to the number of leaks as part of the prediction.

This article originally appeared in IEEE Intelligent Systems, *vol. 32, no. 2, 2017.*

The importance of the relationship between the variables is illustrated by the potential concern of the original analysis, in which the city of Los Angeles claimed that the number of leaks was decreasing over time. This would result in a negative coefficient on the year variable, and users would be interested in that finding, because it indicates that the number of leaks is decreasing over time, as the Department of Water and Power suggested.

Case Studies' Strengths and Limitations

In this short investigation, my analysis has focused on two questions asked by Watson Analytics: "What drives X?" and "What is the predictive model of X?"

This analysis revealed several strengths of Watson Analytics that probably apply to other similar analytics software as well. First, such software can provide analytical analysis to virtually anyone who can upload data to the cloud. Second, the system clearly can generate interesting questions. In some cases, analysis seems likely to bring some surprising relationships to the user's attention. Third, the system considers the data quality, a variable that is often ignored. Fourth, increased use of analytics through systems such as Watson Analytics is likely to impact a manager's need to understand such analysis, and could lead to more detailed statistical analysis of a broader range of issues. Fifth, Watson is particularly effective in those settings in which some discrete variables have a threshold value(s) that needs to be managed.

However, there are also limitations. Watson Analytics likely frames the data for the user, which could lead the user to form initial conclusions that are not appropriate under greater scrutiny. Furthermore, Watson Analytics does not focus on how independent variables drive the dependent variable, either positively or negatively. As a result, although users might understand which variables predict or drive, they may not know in which direction this occurs.

Other limitations include the apparent unavailability of classical statistical tests, an inability to designate control variables, and an inability to choose particular statistical methods. In addition, the modeling approach limits the number of variables (or at least the user's perspective) used in estimation to one or two variables. Although this approach does make it easy to understand the resulting prediction as to what drives a variable, it potentially ignores larger sets of variables. Furthermore, Watson prediction models could be developed on the basis of multiple variables or not, as both case studies showed. From a statistical perspective, Watson Analytics does not seem to account for multicollinearity or endogeneity.

Finally, Watson Analytics does not seem to employ semantic understanding. Perhaps the user could be asked to provide information about the variables to facilitate the analysis. Alternatively, the system could use natural language and try to understand what the variables represent, providing a glimpse at a potential theoretical structure. For example, a person seeing a variable labeled "month" or "year" would likely infer knowledge about that data. As another example, in an analysis of another dataset, Watson Analytics found that "total time" was related to the pair "start time" and "finish time," providing limited insight to the users.

Table 3. What drives the total number of leaks?

Independent variable sets*	Strength (%)	R-square	ANOVA statistical significance
Diff-avg and year	50	0.3007	< 0.0001
Month	43	0.0115	< 0.0001
Diff-max-min and max-max	31	0.1308	< 0.0001
Avg-min	21	0.1089	< 0.0001
Min-min	20	0.1307	< 0.0001
Diff-avg	19	0.1623	< 0.0001

*The dependent variable is "total."

In the movie *The Incredibles*, the villain Syndrome hired Mr. Incredible (a superhero) to unknowingly design an almost unbeatable foe. As Mr. Incredible found limitations in Syndrome's robot creations, Syndrome would eliminate the limitations and come up with a more formidable and, ultimately, a seemingly indestructible robot. I expect that Watson Analytics (and other smart and easy analytics) will take a similar evolution as those robots, evolving from its current incarnation to increasingly more capable versions as people find strengths and limitations associated with the software. I expect that Watson Analytics will begin to embed additional analytics and statistics knowledge. Furthermore, I expect some users will begin to integrate semantic knowledge and domain expertise to further build the capabilities of Watson Analytics to provide users with a domain-guided analytics experience. ■

References

1. *Introduction to IBM Watson Analytics Data Loading and Data Quality*, IBM, Mar. 2016.
2. "IBM Watson Analytics Workshop Reference Guide," IBM, May 2016.
3. D.E. O'Leary, "Modeling Los Angeles Water Leaks Using Different Measures of Temperature," *Advances in Business and Management Forecasting*, K.D. Lawrence and R.K. Klimberg, eds., vol. 11, 2016, pp. 187–207.

Daniel E. O'Leary is a professor in the Marshall School of Business at the University of Southern California. Contact him at oleary@usc.edu.

Editor: **Christof Ebert**
Vector Consulting Services
christof.ebert@vector.com

Different Databases for Different Strokes

Gregory Vial

From the Editor

Databases are one of the oldest technologies we use in computer systems. Yet database management system (DBMS) technology isn't at a standstill—quite the opposite. DBMSs offer modernized relational schemes designed for transactional applications as well as nonrelational systems and mere streaming-oriented technologies for fast analytics. Users have a wide choice of DBMS technologies to best suit their online, social, mobile, analytics, embedded, real-time, and cloud application needs. In this installment of Software Technology, Gregory Vial provides an overview of current DBMS technologies and suppliers, along with a case study of an Internet application. I look forward to hearing from both readers and prospective authors about advances in software technologies. —*Christof Ebert*

WHEN WE TALK about databases, we often mean relational databases (RDBs). To this day, the RDB is often the de facto choice for the persistence of application data. At the same time, the past 10 years have seen the emergence of a variety of alternative forms of databases (for example, NoSQL) that were born out of extreme use cases for Internet giants such as Yahoo, Google, and Facebook. In a context in which social media, mobile computing, and the Internet of Things are becoming ubiquitous, developers are increasingly looking for a different way to store and access data, as are database management system (DBMS) vendors, cloud providers, the open source community, and database researchers.[1]

Here, I look at the state of database technology and offer guidelines to help select the best technology for your purpose.

Relational Databases

The RDB's continued popularity is thanks to several characteristics. For example, it offers a convenient way to store and access data based on a predefined structure that can be normalized to optimize storage and performance, a feature that's both intuitive and convenient for software developers. In the context of transactional systems, RDBs are especially useful because data integrity is guaranteed if the RDBMS implements ACID transactions. (ACID stands for atomicity, consistency, isolation, and durability.)

Although RDBs were first envisioned in the context of transactional systems, they've also been successful for analytical applications. Specifically, they're typically modeled after the dimensional approach proposed by Ralph Kimball[2] and often seen in data warehouses and data marts. Even when online analytical processing (OLAP) requires multidimensional databases and the building of cubes, it most likely will depend on a data warehouse or data mart built in an RDB.

Finally, SQL has proven effective for defining and manipulating relational data models. Although differences do exist across RDBMSs, the ANSI SQL standard and the relative simplicity of the language's basic semantics have contributed to

 Published by the IEEE Computer Society

the development and maintenance of a large pool of human and technical resources to support RDB development.

RDBs, of course, have pitfalls (although these issues generally stem from how we use them rather than their underlying principles). In the realm of transactional systems, the most frequently mentioned pitfall is *impedance mismatch*,[3] wherein the software's data structures (for example, an object-oriented model) aren't always completely transferable to their relational counterparts. So, compromises are often necessary, even if an object-relational mapping engine (for example, Java's Hibernate) manages the translation of an object model into a relational data structure.

In analytical applications, the row-based storage architecture that many RDBMSs use to manage transactional workloads has also received criticism. Specifically, it has been found to be ill-suited for common analytical tasks such as large-scale computations or data compression.

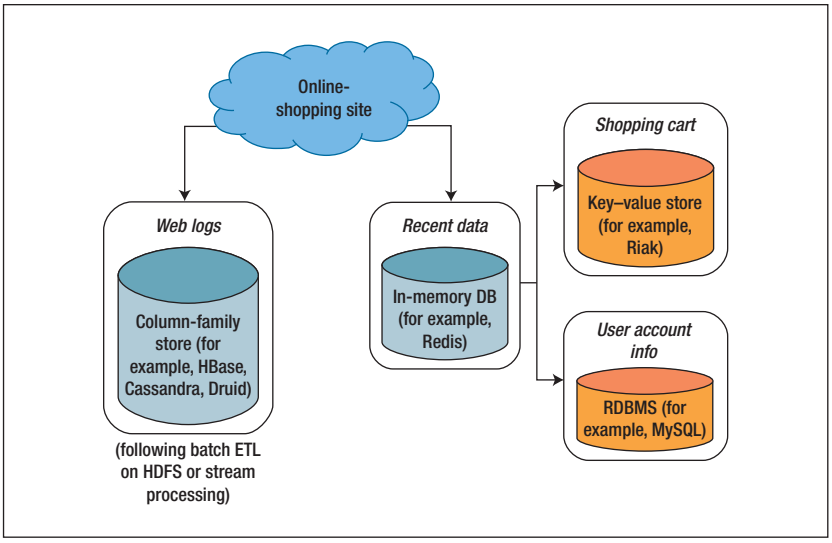


FIGURE 1. Using different database management systems (DBMSs) for a single application. The nature of the data and its purpose can dictate the need for different DBMSs, even in the context of a single application such as an online-shopping site. ETL = extract, transform, and load; HDFS = Hadoop Distributed File System; and RDBMS = relational database management system.

Enter NoSQL

Over the past 10 years, NoSQL and other forms of nonrelational databases have emerged as useful alternatives to RDBs, to avoid their perceived inflexibility and performance overhead. In particular, these alternatives are used heavily in contexts involving the need to ingest, manage, or analyze large quantities of data that don't necessarily follow the way of object-oriented programming (for example, exploratory analytics). This enables us to sidestep impedance mismatch.

Driven by the explosion of open source software, these new categories of DBMS are continuing to evolve rapidly. So, companies can now use different types of DBMSs for different needs (see Figure 1), employing a combination of products that might rely on a common underlying infrastructure such as a Hadoop cluster.

For example, consider web logs (not to be confused with weblogs or blogs). In the past, they were simply text files generated by a webserver and used primarily for monitoring and debugging. Now, they're analyzed to understand how users interact with a website and to optimize conversion rates (for example, to increase the probability that a user will place an item in a shopping cart and finalize his or her transaction).

Although you can break down web logs into rough pieces (for example, a user agent or protocol), their analysis often requires manipulating their raw contents in several ways. Forcing an a priori structure in an RDB to store web logs can limit the potential for analysis and can hinder performance if the logs must be broken down and ingested into the RDB in real time. Instead, companies can store web logs in text files in an HDFS (Hadoop Distributed File System) store and ingest them into a column-oriented DBMS such as Druid, HBase, or Cassandra for analytical queries. (Figure 2 provides an overview of the differences between row and column storage.)

In the NoSQL paradigm, the DBMS provides generic structures to facilitate the rapid insertion, access, and analysis of data at scale. The key, however, is that a NoSQL DBMS doesn't seek to handle all potential use cases. Instead, different NoSQL DBMSs are used for different purposes. This distinction, which follows the Unix philosophy, is crucial because it lets NoSQL perform much better than an RDB when the use case fits the tool at hand. So, the number of DBMSs on

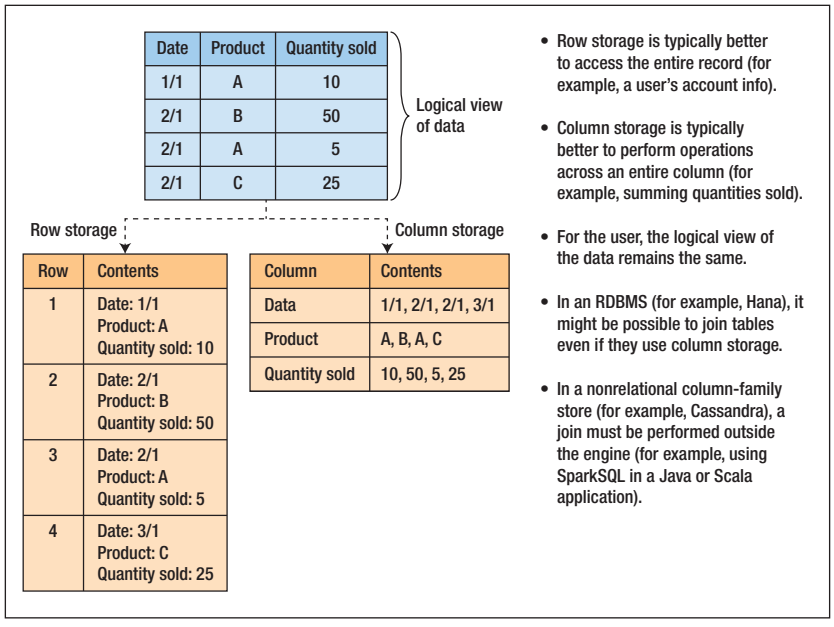


FIGURE 2. Differences between row storage and column storage. Row storage provides high performance for the retrieval of specific records in whole (for example, looking up a customer by ID), as is often needed in a transactional system. However, column storage is better suited to perform aggregations across records (for example, to sum up purchases across customers) in the context of analytical queries.

the market, both proprietary and open source, has grown exponentially. In the NoSQL paradigm, we now find several dozen DBMSs, including key–value stores, document databases, column-family stores, or graph databases.[4]

Interestingly, many NoSQL DBMSs started out with generic APIs (for example, put, get, and remove) to perform basic operations. These DBMSs remained consistent with the idea that the absence of a predefined data structure puts the onus on the developer to ensure that the data is consistent with the application's requirements. Yet over time, several of those systems have also implemented APIs that let developers interact with the underlying data through SQL-like commands (for example, Apache Phoenix and Hive) or that add some support for transactions. Paradoxically, SQL and the relational approach to managing data are so ingrained in our way of thinking about data structures that we often revert to them, even when we're using a nonrelational DB.

ACID Transactions, the CAP Theorem, and PACELC

Early skeptics criticized (often justifiably) NoSQL DBMSs' inability to enforce ACID transactions because they focused on low latency through eventual consistency. The reality is that most NoSQL DBMSs provide ACID transactions, but to greatly varying extents. For example, MongoDB supports ACID transactions only in a single document. Whether this is an issue depends heavily on the use case for the DBMS. Perhaps the more interesting question relates to the distributed nature of NoSQL DBMSs and how they manage the consistency, availability, and partition-tolerance properties of Eric Brewer's CAP theorem.

By design, tradeoffs must be made regarding the CAP properties. Again, the important part is knowing the use case to find the best fit, rather than looking for a universal solution. For instance, eventual consistency across geographical partitions might be satisfactory in an online-shopping scenario but not for a high-speed-trading company.

Seeking to clear up some of the confusion behind CAP, Daniel Abadi proposed a more granular approach to understanding partitioning.[5] To that end, he coined PACELC, an acronym that helps us understand how a distributed DBMS works:

> *If there is a partition (P), how does the system trade off between availability and consistency (A and C); else (E), when the system is running as normal in the absence of partition, how does the system trade off between latency (L) and consistency (C)?*

Blurring the Lines between RDBMS and NoSQL

If comparing RDBMSs and NoSQL DBMSs used to be relatively simple, the trend toward hybrid, ACID-compliant distributed DBMSs such as NewSQL systems renders this task more challenging every day as functionalities evolve (see the sidebar in this article and the table at extras.computer.org/extra/mso2018020080s1.pdf).

Consider columnar storage. Although true nonrelational column-family stores exist (for example, Cassandra), several RDBMSs such as SAP Hana, Microsoft SQL Server,

and Oracle also implement variants of column stores in relational structures. Similarly, RDBMSs often provide in-memory structures for fast, low-latency transactional workloads. Current trends from vendors point to an overarching attempt to provide unified database solutions that cater to both transactional and analytical workloads (sometimes called *translytical* databases), as opposed to specialized, fragmented products such as those in Figure 1 (see Figure 3).

To further complicate things, cloud providers offer a variety of DBMSs, some of which are customized versions of existing systems. For example, Amazon RDS (Relational Database Service) can be provisioned with one of six database engines. At a high level, cloud providers such as Google, AWS (Amazon Web Services), and Azure offer three main database products:

- a NoSQL database optimized for fast performance, with limited-to-no support for ACID transactions;
- an RDB designed for simplified replication and administration, with full support for ACID transactions; and
- a data warehouse optimized for batch loading and analytical workloads.

The other advantage is that cloud providers are committed to these products because they're these product's biggest users. For example, DynamoDB, Amazon's NoSQL store, was born out of the need to manage the company's ever-expanding catalog of products. However, costing these solutions might not always be easy (costs involve a combination of computation time, storage capacity, data transfer, and fault tolerance based on predefined profiles).

Further Considerations

For database and software developers alike, these are truly exciting times. The ability to pick the DBMS that best fits your use case is empowering (although it can also be scary). The widespread availability of DBMSs that are not just relational and designed for transactional applications has greatly helped organizations respond to the implications of SMAC (social, mobile, analytics, and cloud) technologies.

In scenarios that also require real-time analysis, you can use stream-processing engines such as Apache Flink, Kafka Streams, or Samza to quickly parse and process incoming streams of data (for example, for real-time fraud detection) before they're persisted in a database.

Setting aside questions regarding constraints that are specific to an organization's current situation (for example, financial and human constraints or the existing code base), there are six simple, fundamental questions you should ask when selecting a DBMS.

First, is the load you're handling transactional, analytical, or both? If you truly need both, recent DBMSs such as SAP Hana or VoltDB might be a good fit. However, if your analytical workloads are infrequent or use a batch load logic, your requirements might be less stringent, allowing you to use an HDFS store with Apache Kylin, for example.

Second, what type of data model or models does your application need? Is it relational or NoSQL

SPANNER

Spanner is a distributed, ACID-compliant relational database (RDB) developed by Google and available on its cloud platform.[6] (ACID stands for atomicity, consistency, isolation, and durability.) Google originally designed Spanner for internal use to host mission-critical RDBs that ran on custom MySQL cluster installations. At the time, some of the most challenging issues faced by the teams that managed the MySQL setup involved manual maintenance tasks such as data distribution or redistribution across nodes and datacenters, as well as failovers. This was because the architecture was based on a master–slave replication model.

To support the high reliability that Google's systems require, Spanner servers use GPS and atomic clocks to assign commit time stamps that are externally consistent (and thus reliable in and across datacenters). Spanner automatically performs load balancing and data distribution or redistribution, eliminating the need for additional maintenance tasks. Connectivity across datacenters is powered by Google's redundant hardware and software infrastructure, minimizing the potential for network partitions.[7] Although older versions of data can be queried, garbage-collection policies can also be configured to retain only the most relevant data, on the basis of an application's needs.

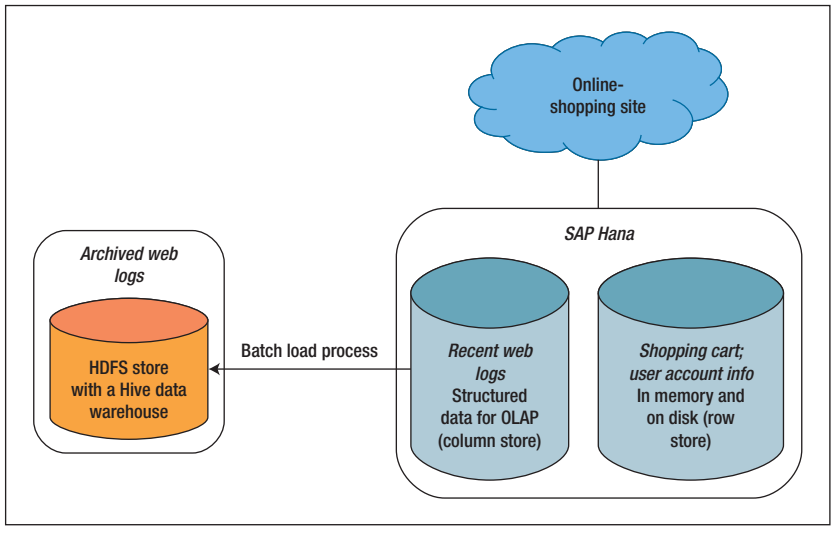


FIGURE 3. Managing transactional and analytical workloads using NewSQL systems. At the application level, it's possible to build views for analytics that effectively combine data from SAP Hana's column store with the Hive data warehouse. (The HDFS store is cheaper than SAP Hana on the basis of the price per Gbyte of data and is better suited for archiving data.) OLAP = online analytical processing.

(key–value, column-family, document, or graph)? Determining this will help you determine, for instance, how to combine multiple data structures in or outside the DBMS engine (for example, to perform joins).

Third, what are your scaling and performance requirements? Determining this will help you determine, among other things, in-memory database requirements, horizontal scaling, and geographical-distribution requirements to optimize storage, throughput, and latency.

Fourth, what type of transaction support do you require? Some engines might not fully support ACID transactions the way we understand them in an RDBMS, but the level of transaction support provided by a nonrelational DBMS might still be sufficient for your use case.

Fifth, how will the DBMS manage partitions? Using PACELC can help you compare DBMSs if you need to scale horizontally.

Finally, will you set up, configure, and maintain the hardware and software infrastructure yourself? If this proves difficult, a cloud service offering might be an attractive solution, at least in the short run.

Beyond the hype associated with NoSQL, big data, and Internet-scale computing, it's important to remain grounded within your context. If your system runs on an RDBMS for which you have acquired extensive knowledge over the years and that performs satisfactorily, there's no pressing need to change. The RDBMS remains the most widely used DBMS scheme, although new technologies are experiencing increased use. You can, however, start experimenting with new features and technologies to learn how they fit your use case (cloud instances are an easy, relatively cheap way to develop proofs of concept or test new features of your preferred DBMS).

For example, if you haven't tried them yet, sample long-running analytical queries and compare their execution times on column stores. Performing this test isn't particularly difficult and can help you see, beyond the vendors' claims, this feature's potential benefits. So, the question isn't just about whether a DBMS can handle your workload and scale with your growing needs. The discussion also revolves around how the DBMS will fit and be managed within your existing hardware and software infrastructure (including its potential dependencies). Unlike most other companies, Google, Facebook, and Netflix actively use and develop these DBMSs. The rest of us just want a DBMS that works and performs as expected. Defining those expectations is perhaps the most difficult and important part.

References

1. D. Abadi et al., "The Beckman Report on Database Research," *Comm. ACM*, vol. 59, no. 2, 2016, pp. 92–99.
2. R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, John Wiley & Sons, 2011.
3. C. Ireland and D. Bowers, "Exposing the Myth: Object-Relational Impedance Mismatch Is a Wicked Problem," *Proc. 7th Int'l Conf. Advances in Databases, Knowledge, and Data Applications* (DBKDA 15), 2015, pp. 21–26.
4. P.J. Sadalage and M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Pearson Education, 2012.
5. D. Abadi, "Consistency Tradeoffs in Modern Distributed Database

System Design: CAP Is Only Part of the Story," *Computer*, vol. 45, no. 2, 2012, pp. 37–42.
6. J.C. Corbett et al., "Spanner: Google's Globally Distributed Database," *ACM Trans. Computer Systems*, vol. 31, no. 3, 2013, article 8.
7. E. Brewer, *Spanner, TrueTime and the CAP Theorem*, tech. report, Google, 2017; research.google.com/pubs/pub45855.html.

ABOUT THE AUTHOR

GREGORY VIAL is an assistant professor in HEC Montréal's Department of Information Technologies. Contact him at gregory.vial@hec.ca.

This article originally appeared in IEEE Software, *vol. 35, no. 2, 2018.*

The Role of a Customer Data Platform

Seth Earley
Earley Information Science

Editor:
Seth Earley, Earley Information Science; seth@earley.com

At the end of the day, everything boils down to providing value to our customers. The more we know about our customers, the better we can provide that value and stand out from our competitors. Although this objective sounds simple, many challenges arise along the way. The entire customer experience relies on data, so creating a positive and compelling experience depends on the quality and types of available data to understand the needs of the customer. The challenges to getting the right data fall into four broad categories related to systems that produce and consume customer data: the ways that data is modeled, the data itself, gaining insights from the data, and acting on the data.

Customer data platforms (CDPs) aggregate data from many different sources to provide a 360-degree view of the customer. These platforms are designed to be managed and used directly by marketers, and they eliminate the need to access multiple systems to create customer profiles, develop marketing campaigns, test the effectiveness of marketing strategies, and predict customer behavior.

WHAT IS A CDP?

According to the Customer Data Platform Institute (CDPI), a CDP is "a marketer-managed system that builds a unified, persistent customer database that is accessible to other systems." The CDP acts as a centralized clearinghouse and repository for all sorts of data from various internal and external systems. Consider any place where a customer interaction is recorded, tracked, or managed. Past purchases constitute a big category of customer behavior, of course. But so do social media interactions and website visits, even when nothing is actually purchased. Collectively, this data produces signals that can be thought of as "electronic body language."

Some data is reasonably straightforward (such as name, address, and demographic details). However, some information requires processing and interpretation. Clickstream data, for example, tracks part of the customer journey and can be very informative, but understanding what it means requires effort and human intervention. Data about website behavior can be stored in a CDP, but the dataset is large and has numerous components that are time- and context-dependent. Before the data can be meaningfully acted upon, it must be analyzed and interpreted.

Ability to Summarize Data or Surface Trends

A CDP can be used to summarize certain types of data to present a trend or characteristic that can then be exploited. Rather than require a marketer to sift through hundreds of log lines from a

 Published by the IEEE Computer Society

website visit, certain events or behaviors can be segmented and acted upon. These could include offers, promotions, content that similar users find valuable, or other relevant signals derived from large datasets.

Integration of Varying Data Formats and Structures

Another valuable function of CDPs is their ability to accommodate different data types and formats that might have varying structures and naming conventions. Data might come into the CDP through a live feed via an API or web service layer, or might be input on a batch basis through a file transfer. Formats can be structured by transaction such as accounting and purchase data or unstructured content such as chat logs or Facebook conversations, tweets, and even images from Instagram. This varied data can then be put into a consistent format (through rules retained in the CDP) that can be more easily interpreted by a marketer or acted upon by other systems without repeated manual running of extraction, translation, and load scripts.

Ability to Cleanse and Process Data

Data might have redundant records and content; for example, it might be missing details (fields or values) or contain incorrect data that has to be reconciled with another system. When new data is available, only certain values can be updated. Other data sources can be used to enrich the data or append missing information. Rules for cleansing, enriching, appending, and correcting data can reduce the cost and complexity of data hygiene by automating remediation.

Exposing Data for Use by Other Systems

A major function of the CDP is to act as a centralized location so other systems can access and act upon customer data. The CDP becomes a broker or orchestration layer that can take the outputs from one customer-facing application, process it, convert the format, and export it or make it available via an API for a personalization engine. The personalization engine might then send data back to the CDP, which can use the results to inform or drive another process.

A major function of the CDP is to act as a centralized location so other systems can access and act upon customer data.

Doesn't My Marketing Automation System Do This?

A CDP can provide some of the functionality of other marketing systems and customer engagement platforms, but it is fundamentally different in design and function. Marketing automation systems can integrate with other tools but usually in a limited fashion to suit a narrow set of use cases. CDP tools are designed from the ground up to talk to other systems. They also retain details from other systems that the engagement or automation tool does not. This is valuable for trend analysis, predictive analytics, and recommendations that can leverage large amounts of historical data.

The various functions of a CDP, from data ingestion and normalization to identification of relevant signals and output to downstream systems, are illustrated in Figure 1.

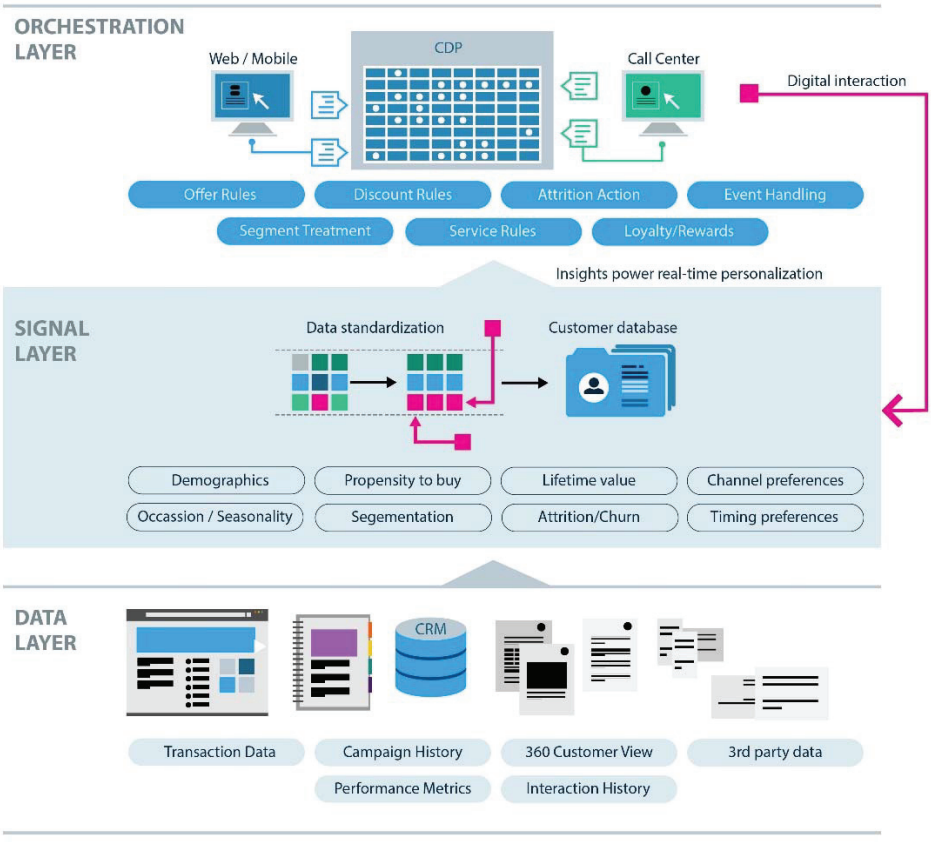


Figure 1. Functions of a customer data platform (CDP).

CUSTOMER DATA MODEL CHALLENGES

A customer data model identifies the factors that a marketer believes to be important in understanding and predicting a customer's behavior. Without a model, there is no way to systematically segment customers into groups and test the effectiveness of different marketing strategies. A data model captures a variety of data, from unambiguous, clearly defined attributes like name, address, and demographic details to data that can be derived and inferred through interactions and by processing data produced by other systems.

The data model contains attributes that might be created when an event occurs—for example, when a customer's purchases exceed a certain threshold. Or an attribute can be based on who the customer is and where they live. Segmentation models can be based on a combination of explicitly defined data points, such as purchase history, user-declared values (for example, an expressed preference), externally referenced information (subscription or membership information), and attributes and values that are inferred by comparison to large numbers of customers with similar characteristics. Some techniques find hidden or latent attributes, or create relationships based on numerous subtle data signals.

Data models can be mathematical, rules-based, visual, or based on a list of the relevant factors that are believed to produce certain behavior; for example, all people under the age of 40 will prefer slim-cut jeans and those over the age of 40 will prefer a looser fit. (This is not a valid general statement, but it could form the basis of a hypothesis that could then be tested.)

A data model represents the customer and the collective insights and understanding of their real-world needs.

Creating a data model is a valuable exercise for non-technical specialists because it allows marketers to use language to describe what they know, believe, or can infer about their customer. These characteristics are converted to a structure that the system leverages (either capturing the details or defining rules and algorithms for inferring them). Data models can also inform marketers about the types of metrics they should track, and how well their strategy and specific campaigns are performing relative to a particular customer segment or characteristic.

Customer data is usually collected from a large variety of systems that come from different vendors or, if homegrown, are created by different groups. Therefore, they will have varying formats, architectures, and naming conventions. As a result, customer data models could be inconsistent, which makes it challenging to create a unified model that incorporates meaningful data in actionable formats. For example, one system might define one customer at the individual level and another at the household level. If one system totals all of the purchases for multiple members and another tracks individual purchases, the analysis of sales per customer will produce different results.

Despite variations in the data, the model must contain enough detail and the correct attributes to support advanced functionality such as effectively predicting purchase patterns or recommending an appropriate product that meets the precise needs of the customer. A customer data model is analogous to a content data model (typically called a content model). Customer profiles contain attributes that are used by other systems to improve their outputs. For example, for personalization to work correctly, the model needs to provide signals to customer engagement systems that tell those systems how to differentiate the customer's experience—what content to present, what products or solutions to offer, and the overall experience that will move them forward in their journey. What is it about the customer that can be captured as metadata (or attributes) in the customer data model and represented in the details of their profile that will drive a unique interaction? It might be the customer's age, or whether they were active on social media, or whether they had children. The CDP stores data about the customer that can be leveraged by various downstream systems to predict and influence the customer's behavior.

These signals can come from many sources. Some are based on explicit attributes such as demographics, content preferences, and account information (see Figure 2). Others come from subjective or behavioral attributes (see Figure 3). These might be interest profiles, past purchase behaviors, social media patterns, loyalty scores, and real-time website behaviors.

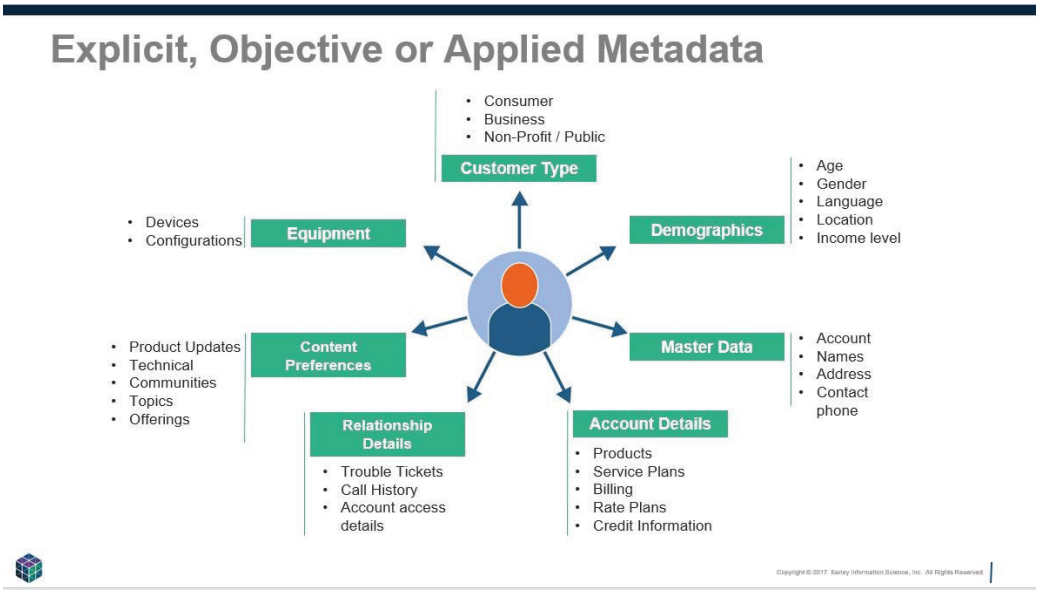


Figure 2. Customers can be described with explicit metadata from a variety of source systems.

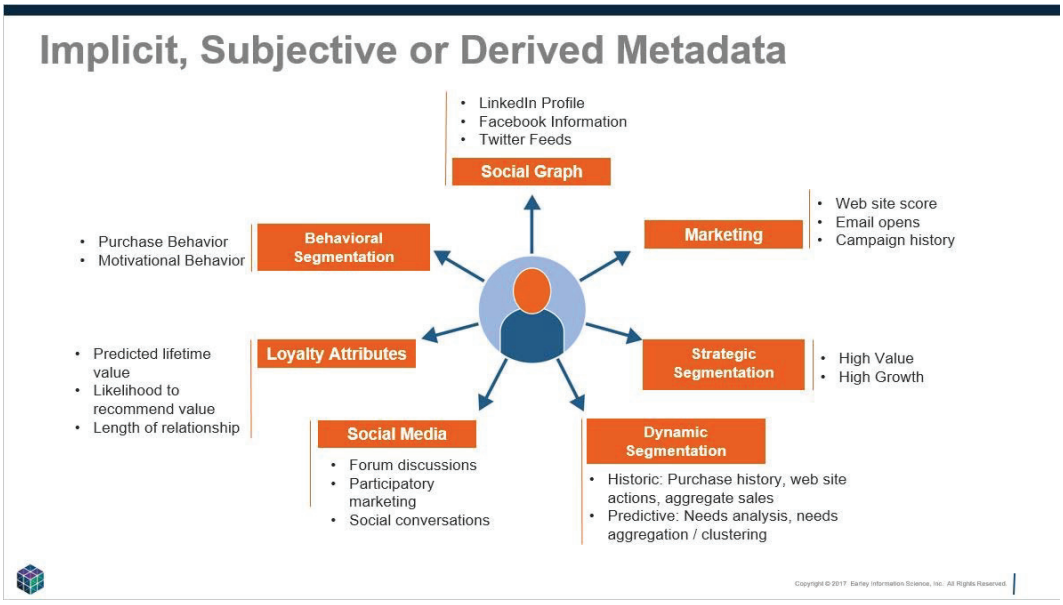


Figure 3. Implicit metadata about a customer is based on judgment and/or derived from other data sources.

ATTRIBUTES, METADATA, AND SIGNALS

Attributes are descriptors that define all the things we know about the customer and describe their characteristics, needs, and behaviors. They determine how systems and tools provide an experience that helps customers meet their objectives and solve their problems. For systems and tools to meet user needs, these attributes have to be represented as metadata in the customer experience applications and in back-end systems that contain transaction and interaction histories.

Metadata supports the machinery of back-end operational and front-end customer-facing technology. Metadata describes the "is-ness" and "about-ness" of a customer. Are they consumers or are they businesses? The metadata field of "customer type" contains the attribute of consumer.

A signal is the data produced by an action of the customer when interacting with an application. The applications can fall into multiple categories of functionality including those that generate explicit, objective, or applied metadata (see Figure 2). This metadata and associated data is typically sourced from customer relationship management systems, enterprise resource planning, e-commerce, sales automation, order management, and external data sources like credit bureaus, data aggregators, and credit card processors (some of which can also fall into the derived metadata category). Examples of this type of metadata include:

- customer type (consumer, business, or nonprofit),
- demographic (age, gender, language, location, income level),
- master customer data (account, name, address, contact phone, email),
- account details (products, service plans, billing, rate plans, credit information),
- relationship details (trouble tickets, call history, account access details),
- content preferences (product updates, technical, communities, topics, offerings), and
- equipment (devices and configurations).

Applications that generate implicit, subjective, or derived metadata (see Figure 3) are sourced by analyzing outputs from diverse customer experiences and operational and social media applications, and by applying conventional analytics and machine-learning algorithms to create

new ways of understanding and describing the customer. Examples of this type of metadata include:

- social graph (LinkedIn, Facebook, Twitter, Instagram),
- marketing applications (website score, email opens, campaign history),
- strategic segmentation (high value, high growth),
- social media (forum discussions, participatory marketing, social conversations),
- loyalty attributes (predicted lifetime value, likelihood to recommend value, length of relationship), and
- behavioral segmentation (purchase and motivational behavior).

The overall information architecture (for customer data models and content models) needs to be aligned so that specific pieces of information can be surfaced to the user depending on the real-time signals from customers' digital body language. This step requires human understanding—which pieces of information most contribute to a solid model for behavior. While some of these features can be defined in advance, many are based on upstream and downstream system architecture and algorithms.

MINING THE DATASTREAM

What does customer data reveal about what the customer needs? Many different data sources are available. For instance, social media data might contain information about a preference. How is that captured? Clickstream tells us something about how customers are consuming content and traversing the website: whether they click through an offer, whether they respond to a promotion, or whether they are able to complete their purchase. The data tells a story—the question is how to understand that story.

Every tool and technology in the digital engagement ecosystem produces datastreams that need to be interpreted to be acted upon. The challenge lies in identifying what data is important, understanding what it is saying, and determining what to do with it. These questions must be asked over and over again to keep the focus on the purpose of collecting and analyzing data, which might change over time.

Metrics fall into several classes, including search, behavior, utilization, content, and response. Each category of metric can have dozens of details and reports. The goal of understanding these metrics is to drive an action to optimize or improve an outcome. When users browse to a certain point and then leave the site, they were unable to complete their task. What can be changed to impact this behavior? Experiments need to be designed to find the best combination of user outcomes. These are the insights that lead an organization to change its strategies for campaigns and offers. The CDP not only allows ready access to all the metrics associated with customer behavior, but also provides the ability to execute an appropriate response based on the data, sometimes by accessing and triggering actions in external systems.

DERIVING CUSTOMER INSIGHTS

A good sales associate knows both their customer and their products and solutions, and makes recommendations that meet the customer's needs. In the digital world, organizations need a way to capture and act on the insights that come from a digital representation of this knowledge. This means first interpreting the signals, which arrive in a different form from those a sales associate receives. Marketers and merchandisers have been dealing with the issue of signals for years, but in a different context—the signals were in the form of customer feedback, market intelligence, and sales trends captured over a period of weeks or months and across broad segments of customers rather than at the individual level.

The new pieces of the puzzle are the scale of the challenge, the velocity of commerce, and the number of variables that need to be interpreted as well as the granularity of responsiveness to customer needs. It is now possible to act on detailed understanding of customer needs based on signals from data; however, it is not possible to manually design and create the combinations of

products, services, and solutions that meet each customer's unique needs. In the physical world, this is what a great salesperson does—they know the customer and offer solutions based on that knowledge. Digital technology is the stand-in for the best salesperson in an organization.

The process begins with a hypothesis about what offers the customer will respond to and what components of that offer can be recombined. It could be as simple as shopping basket analysis, where purchase history combinations are mined and presented to customers exhibiting similar signal patterns. Other sources of variables can be mined from human experts, product engineering documentation, maintenance manuals, and support call chat logs.

Personalizing search results can be based on customer data models that are related to product data and content models mapped through an ontology. The ontology tells the search engine how to interpret a search query based on interests, industry, or other derived profile data. For example, "mold stripping" means one thing to a manufacturing engineer and another thing to a construction manager. In the first case, the task is related to an injection mold maintenance procedure. In the second, it is related to an approach for renovating a flooded building. The ontology can disambiguate a search query by associating the context of the term with the industry or task of the user. This requires a customer data model that captures information about the user's world at a more granular level.

Marketing organizations need to understand users and their tasks and journeys at a level of specificity that allows them to anticipate needs and determine what offer, product combination, solution, advice, or content will move the customer closer to their objective. Marketers have to ask, "What does it mean to personalize my customer's experience? How can I differentiate between one type of customer and another, and what does that mean for how they interact with us? What content and information can be served that will be different? Why is it important?"

Digital technology is the stand-in for the best salesperson in an organization.

An analysis might reveal that for a specific segment of customers, a particular combination of products leads to increased conversions under certain circumstances. The hypothesis is that extending the product breadth with new audiences will also lead to increased revenue. To test this, the CDP needs to integrate with downstream systems that orchestrate the user experience. These systems include content management systems, e-commerce applications, product information systems, and the CDP.

Once the insights are gained, the challenge becomes one of converting knowledge into action.

ACTING ON INSIGHTS FROM CUSTOMER DATA

The goal of understanding the customer is to take action based on that understanding. How does the organization interpret the interactions, preferences, experiences, and all the signals that stream from every customer relationship? What do you do with the data? This is the "what next" question that has to be continually asked and answered. What do we do when we know our customers' needs? The obvious answer is that we try to meet those needs as cost-effectively as we can. The question is always, "How do we act on the information?"

Human intervention is critical at this stage. Acting on insights requires a human to interpret the data and recognize a pattern, the ability to test hypotheses about actions that will create the desired behavior, and the machinery to operationalize the confirmed (or fine-tuned) hypothesis. The end result is dynamic functionality derived from a combination of human judgment, expertise, and creativity. However, this outcome cannot be achieved unless the foundational information, data, and content architecture is in place for the digital machinery.

Because data comes from different processes and applications, there are likely varying constraints on the data—from usage and permissions to downstream enrichment. In many cases, owners of data in one application might not be aware of the downstream impact of their decisions about customer data. They also might not be impacted by quality issues that would be

critical to another use case at the downstream orchestration layer. In that case, there would be no incentive to fix the data and no business case to invest in remediation. These complexities make CDPs an enterprise priority that requires executive support, sponsorship, and funding that considers the entire customer data ecosystem.

CONCLUSION

CDPs are increasingly essential to integrated digital marketing programs. Deploying these technologies reveals gaps and challenges throughout the entire enterprise and the digital supply chain serving the customer. If the customer cannot be fully understood from every point of view of the enterprise, it is not possible to serve them optimally. These gaps and challenges can only be remediated with board- and C-level resources and attention. If you are not serving your customers optimally, they will go to your competitor.

Successful use of customer data requires the development of a robust model, judicious selection of data, careful interpretation of analytics, and the ability to act on the results. Each of these steps poses its own challenges. By providing access to data from numerous systems in one database and supporting the systems that can produce an appropriate customer experience, the CDP overcomes the limitations imposed by fragmented point solutions and presents a holistic approach to customer interactions.

ABOUT THE AUTHOR

Seth Earley is CEO of Earley Information Science (www.earley.com). Contact him at seth@earley.com.

This article originally appeared in IT Professional, *vol. 20, no. 1, 2018.*

UC RIVERSIDE UNIVERSITY OF CALIFORNIA | Marlan and Rosemary Bourns College of Engineering

The Electrical and Computer Engineering (ECE) department at University of California, Riverside invites applications for a faculty position (open rank) in Computer Engineering. The position may be filled starting the winter quarter of 2019, but the search will continue, with a possible later starting date, until the position is filled.

Applicants are invited in all major areas of Computer Engineering such as embedded and real-time systems, cyber-physical systems, connected and automated systems, computer architecture, design automation, and VLSI design. For appointments at the Assistant Professor (JPF00905) level, we seek candidates that demonstrate potential for exceptional research and teaching. For appointments at the Associate or Full Professor (JPF00906) level, we will consider candidates with an exceptional track record in research, teaching and graduate student supervision. The position requires a Ph.D. in Computer Engineering, Electrical Engineering or Computer Science at the start of employment.

The Department of Electrical and Computer Engineering has over 30 faculty members. Over half of our senior faculty members are fellows of IEEE, AAAS or other professional societies, and 12 junior faculty members have been awarded Young Investigator/CAREER awards. The ECE department is focused on research and graduate education, with over 150 graduate students and over $7 million USD in new grants annually. Our faculty are active in areas including computer engineering, robotics and controls, computer vision and machine learning, intelligent systems, smart grids and energy, nano materials and devices and signal processing,. The ECE department was ranked 24th overall in the scholarship metric in the latest National Research Council (NRC) rankings. The position also includes appointment in the Computer Engineering Program, which is jointly administered with the Computer Science and Engineering Department. More information on the department is available at http://www.ece.ucr.edu

Salary level will be competitive and commensurate with qualifications and experience. Advancement through the faculty ranks at the University of California is through a series of structured, merit-based evaluations, occurring every 2-3 years, each of which includes substantial peer input.

Full consideration will be given to applications received by August 31, 2018. We will continue to consider applications until the position is filled. To apply, please submit your application materials (CV, cover letter, statement of research, statement of teaching, statement of contributions to diversity, and contact information for 3 references for Associate/Full Professors level through the web link at https://aprecruit.ucr.edu/apply/JPF00906. For Assistant Professors use the web link at https://aprecruit.ucr.edu/apply/JPF00905. For inquiries and questions, please contact us at cen-search@ece.ucr.edu at http://www.engr.ucr.edu/about/employment.html.

UCR is a world-class research university with an exceptionally diverse undergraduate student body. Its mission is explicitly linked to providing routes to educational success for underrepresented and first-generation college students. A commitment to this mission is a preferred qualification.

The University of California is an Equal Opportunity/Affirmative Action Employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, age, disability protected veteran status, or any other characteristic protected by law.

UC RIVERSIDE UNIVERSITY OF CALIFORNIA | Marlan and Rosemary Bourns College of Engineering

The Department of Electrical and Computer Engineering (ECE) at the University of California, Riverside invites applications for an Assistant Professor of Teaching (Lecturer with Potential Security of Employment LPSOE) beginning the 2018/19 academic year. The appointment requires evidence of a long-standing record of exceptional undergraduate teaching, professional achievement, and service to the academic community. A LPSOE/LSOE is a member of the academic senate and will be expected to teach undergraduate and graduate courses, and provide university and public services related to the undergraduate program including curriculum development, student advising, ABET accreditation, recruitment outreach etc. Applicants with strong teaching experience in robotics, circuits and systems, computer engineering, control, signal processing and computer vision are encouraged to apply.

Preferred qualifications include:

- A minimum of 5 years (LPSOE) or 10 years (LSOE) of experience as an undergraduate course instructor in an ABET-accredited Electrical or Computer Engineering program.
- A demonstrated record of high-quality and innovative teaching in a wide variety of undergraduate ECE courses in at least two of the areas of robotics, circuits and systems, computer engineering, control, signal processing and computer vision.
- A demonstrated record of teaching in capstone senior design courses, as well as successful engagement with industrial sponsors to generate support for senior design projects.
- A demonstrated record of development of successful technical elective courses that address important emerging topics in the field of Electrical and Computer Engineering.
- Experience in advising undergraduate students in course selection and career development strongly preferred.
- Prior work experience in industry as a practicing engineer. Further information regarding the ECE Undergraduate Program can be found at the following site http://www.ece.ucr.edu/.

A Ph.D. in Electrical or Computer Engineering is required at the time of employment. Salary will be competitive and commensurate with qualifications and experience. Full consideration will be given to applications received by August 31, 2018. The search will continue until the position is filled. To apply, please register through the weblink at http://www.engr.ucr.edu/facultysearch/. Inquiries should be directed to ecesearch@engr.ucr.edu.

UCR is a world-class research university with an exceptionally diverse undergraduate student body. Its mission is explicitly linked to providing routes to educational success for underrepresented and first-generation college students. A commitment to this mission is a preferred qualification.

The University of California, Riverside is an Equal Opportunity/Affirmative Action Employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, age, disability, protected veteran status, or any other characteristic protected by law.

IEEE computer society

SHARE THE GIFT OF KNOWLEDGE:

Give Your Favorite Student a Membership to the IEEE Computer Society!

With an **IEEE Computer Society Membership**, your student will be able to build their network, learn new skills, and access the best minds in computer science before they're even out of school. Your gift includes thousands of key resources that will quickly transition them from classroom to conference room, such as:

- **A subscription to *Computer* magazine** (12 issues per year)
- **A subscription to *ComputingEdge*** (12 issues per year)
- **Local chapter membership**
- **Full access to the Computer Society Digital Library**
- **Eligible for 3 student scholarships where we give away US$40,000 yearly**
- **Skillsoft:** Learn new skills anytime with access to 3,000 online courses, 11,000 training videos, and 6,500 technical books.
- **Books24x7:** On-demand access to 15,000 technical and business resources.
- **Unlimited access to computer.org and myCS**
- **Conference discounts**
- **Members-only webinars**
- **Deep member discounts** on programs, products, and services

Give Your Gift at: www.computer.org/2018gift