DOI:10.1145/3180664

As software becomes a larger part of all products, traditional (hardware) manufacturers are becoming, in essence, software companies.

BY TONY GORSCHEK

Evolution Toward Soft(er) Products

SOFTWARE IS A cornerstone of the economy, historically led by companies like Apple, Google, and Microsoft. However, the past decade has seen software become increasingly pervasive, while traditionally hardware-intensive products are increasingly dependent on software, meaning that major global companies like ABB, Ericsson, Scania, and Volvo are likewise becoming soft(er).[10] Where software was bundled with hardware it is now increasingly the main product differentiator.[10] This shift has radical implications, as software delivers notable advantages, including a faster pace of release and improved cost effectiveness in terms of development, ease of update, customization, and distribution. These characteristics of software open a range of possibilities, though software's inherent properties also pose several significant challenges in relation to a company's ability to create value.[10] To investigate them, we conducted in-depth interviews from 2012 to 2016 with 13 senior product managers in 12 global companies.

The first interview in 2012 was followed by confirmation and updates in 2014, 2015, and 2016. Common to all 12 companies is that they are continuously moving their products toward being "softer." The 13 managers work on different software-intensive products, from Intel in embedded and mobile software to ABB in power-automation technologies to telecom (see the table here). The central aim was to identify the challenges emerging as a result of companies making their products increasingly soft, specifically those using software as an innovation driver of their products and services.

All 13 managers were positive about software, seeing it as an enabler, giving their companies the ability to evolve their products more quickly and develop features and customer benefits that were difficult or more expensive before software was part of the product. They generally viewed the ability to develop ideas fast (compared to the relatively slow pace of hardware product development) and release features and products at a pace much more frequent than before. It also means updating current products without shipping, requiring just a "press of a button," as one manager put it. The managers also viewed the ability to customize products as a big competitive advantage, as one manager explained, "We can create a new version in hours something that was almost impossible a month before," as changing the software also changed the product, not

» key insights

- **The benefits of software as part of a product are sometimes offset by the challenges of engineering, evolving, and managing software as part of the product.**
- **Many traditionally hardware-intensive companies transitioning to software-intensive underestimate the organizational, managerial, and engineering changes involved.**
- **Software is flexible and can enhance product offerings, and is also complex and fast changing while involving potential for degradation.**

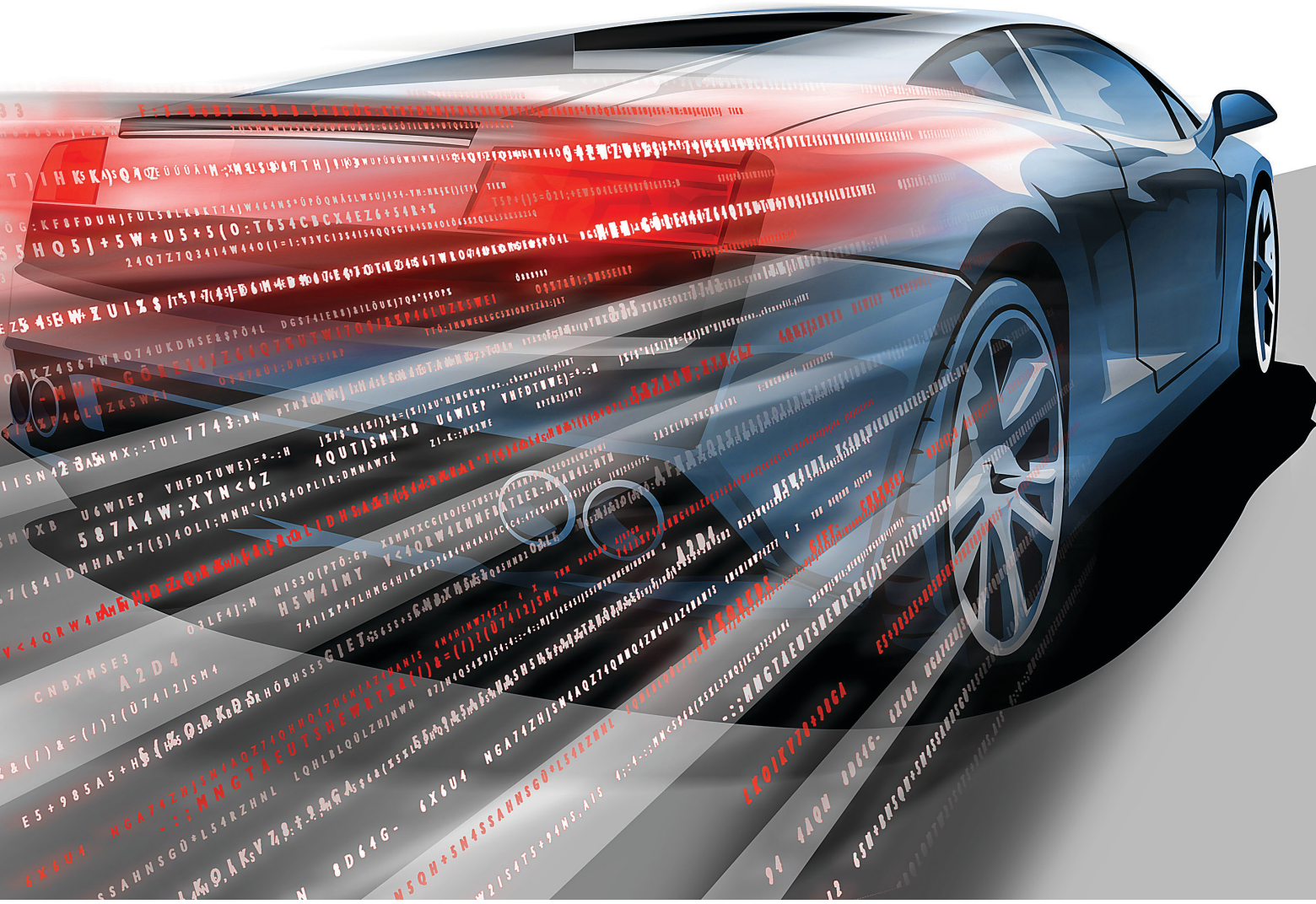

IMAGE COLLAGE BY ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

possible in the former version that was mostly hardware. Overall, they viewed software as part of a product as a revolution in terms of both technology development and business competition.

Challenges

With any revolution, evolution becomes a necessity; that is, by adapting and changing technology, development, and business practices, as identified by the managers and outlined in the table. The main challenges associated with becoming soft(er) are real, based on the gradual change seen over the past decade in each company. More important, the challenges persist, as the managers reported. Moreover, even if research into the state of the art views some problems as "solved," solved is not the case if companies and senior managers continue to perceive the challenges as immediate. That is why we focused on challenges as they are perceived, not on "best practices" from research (see the sidebar "Study Design and Result Analysis").

Software was not new to the companies. Major global companies like ABB, Ericsson, and Scania pioneered the use of software, developing their own programming languages and operating systems in the 1970s and 1980s. However, it was often used as an embedded component or just as support for hardware. In recent decades, the "revolution," as stated by some managers, was in software moving up the food chain, becoming increasingly the main part of a product. Today, however, the tables have largely turned, as software drives innovation, including in process, product, market, and organizational innovation. This fundamental change has put new demands on the companies having to address challenges "as real as the products," as stated by one product manager. This context involves two main types of innovation: product and market, focusing on product and sales/delivery; and process and organizational, implying changes in how products are developed and how the company doing the developing

changes as a result. All such innovation (changes) involve challenges, as explored in the online appendix "Challenges and Related Work" (dl.acm.org/citation.cfm?doid=3180492&picked=formats), associating them with implications and sources for proposed solutions. The challenges, implications, and further reading sometimes overlap, as the 10 challenges (and their potential solutions) overlap. Here, we identify the various challenges in the interest of readability.

Internal Business Perspective

Challenge 1. Understanding the value propositions for different stakeholders and sharing it within the company, as supported by seven companies and eight managers. Software offers different value propositions for different stakeholders. For example, the electrical meters developed for utility companies are not designed to read only consumption of electricity but also to perform quality measurements in the network, measuring the amount of reactive energy produced there to phase-off energy and more. Such a meter offers many benefits (value) for the utility company (such as improved peak-load management), resulting in efficient grid use and dynamic tariff models. Value for the consumer can also be significant, providing, say, correct and frequent billing and cost savings through better awareness of their own consumption patterns. Governments are yet another stakeholder group concerned with reduced CO_2 emissions, possibly through smart meters by identifying energy-consumption patterns. Software-based meters might also enable new business models.

However, the company's sales force is "used to dealing with straightforward single-value proposition for a traditional meter," as one product manager put it. The introduction of software added new propositions that are largely unknown to the previously highly effective sales force. It has proved to be "almost impossible" for that same sales force to sell the new product to traditional buyers, so needs new training and insight into how the new software and, in this case, smart meters, change the offering and potential of the product line. In addition, as software offers the potential for constantly updating product features, the potential value propositions of the product likewise evolve continuously and at a much faster pace.

Software today is the main competitive advantage, enabling faster and cheaper innovation and product differentiation, especially as hardware is increasingly standardized.

This increasing amount of software in products is not a new phenomenon, as even companies incorporating it into their products do not always take new value propositions into consideration. Companies that are used to selling "boxed products" (such as hardware) find it difficult to understand the new value propositions and corresponding business models for selling business solutions when bundling hardware with software.[10]

One product manager recalled an incident where he introduced a Web service (for an award-winning previously mostly hardware product) used to connect a customer relationship management system to printing and response handling, postage optimization, and channels handling 13 separate projects and their customer relations. However, the sales team, trained to sell hardware-intensive products, was unable to manage pre-sale of the product, as it was difficult to visualize what was actually being sold, ultimately resulting in limited sales performance.

One manager said, "It is important to change the mindset of people." Traditionally, software is seen as the "poor cousin" that "had to be there," as reported by another manager, bundled with the hardware, but without real value by itself. Software today is the main competitive advantage, enabling faster and cheaper innovation and product differentiation, especially as hardware is increasingly standardized.[10,14] Decision-making patterns that take into account different value aspects of a product can alleviate some of the risks associated with missing important aspects of the product (such as the ability to understand its potential).[14] Also, enhancing sales teams by hiring people with experience selling software is sometimes another way to alleviate the limitations in creating and selling new value propositions. Moreover, given the possibilities with software-based products, pre-sales, sales, product management, and R&D need to work much more closely than before to create "solutions." Several managers viewed collaboration as critical, as the nature of a product changes,

but also to compensate for the faster pace of new offerings, as it does not allow for a formal learning process previously seen in the company. Companies shifting their focus toward software-intensive products often consider it enough to hire software engineers for development, largely ignoring the need to simultaneously evolve other organizational units (such as sales, support, and pre-sales).

Challenge 2. Patenting (protecting) software-based innovation, as supported by four companies and four managers. Applying for software-based patents risks being copied by competitors. The format whereby software inventions are disclosed in patents (such as flow charts, line drawings, and technical specifications) allow any programmer to develop software that can perform the same patented ideas.[a] Such technological copying combined with lower start-up costs (no design or production needed) enable software-based innovations to be copied more readily than hardware-based innovations. One manager said, "Anyone with a home computer can copy our ideas while sitting in a basement, not to mention our competitors." The risk of being copied without compensation is further aggravated by the time delay between when a software patent application is filed (becoming public) and when it is approved, possibly 18 months or more.[b] This can mean lost competitive advantage, as copied software can be included in a competitor's product. Moreover, even if the patent is granted at some later date, the incurred fees for the competitor might be small in comparison to the revenue lost by the original inventing company. Several managers said "being first" (or even being seen as being first) to market is sometimes critical, and compensation after the fact will not make up for the lost position.

To mitigate the risk of having one's ideas copied, companies sometimes keep software-based innovations (such as algorithms) hidden in their code. But hiding innovation is not a sustainable solution according to several managers. Being able to patent software is essential, and a revised patenting process is needed to enable easier filing and quicker decisions. A potential alternative, mentioned by two managers, is to discontinue the patenting of actual software altogether, patenting instead only algorithms. However, this would mean a radical change for most companies, as formerly hardware-intensive companies rely on protection at the core of their business models and cultures. Some technology companies have tried to enhance protection by forming alliances to, say, enable cross-licensing and/or pooling resources collaborative patenting efforts.

a http://www.epo.org/news-issues/issues/software.html

b http://www.uspto.gov/web/offices/pac/mpep/s1120.html

Profiles of interview subjects and their companies.

Designation	Company	Products	Software and type of innovation*
Product manager	Wind River	Simulators (such as for flight control systems, wind-speed simulation, and simulating military systems)	Process, as new customer types emerge
Program manager	Micronic	Control software and software for handling data	Market and product
Global innovation manager		Electromechanical locks	Process, market, organizational, and product
Consultant, senior manager	Anonymous(1)	A range, from electric meters to robots	Process, market, and product
Program manager for innovation and research	Ericsson	Telecom solutions	Process, market, organizational, and product
R&D manager	Scania	Encoders for trucks	Process, market, organizational, and product
System architect and manager	Scania	Application software for trucks	Process, market, organizational, and product
Product manager	Anonymous(2)	Telecom solutions	Process and organizational
Product manager	Anonymous(3)	Telecom products and services	Process, market, and organizational
Product manager	Anonymous(4)	Telecom solutions	Process, market, and organizational
Senior manager	Anonymous(5)	Surveillance solutions	Process, organizational, and product
Product manager	ABB	Automation	Process, market, organizational, and product
Product manager	Anonymous(6)	Mobile applications	Process, market, and product
Product manager	Anonymous(7)	Services	Process and organizational

* The fourth column denotes "innovation type," or how a company categorizes the effect software has on its products, along with the company's internal view.[3] **Innovation types** include process innovation, or implementation of new design and analysis or development methodology that changes how a product is created; **market innovation**, or implementation of new or substantially new marketing strategies and product design or packaging, promotion, or pricing, including creating new market opportunities and implementation of new or significantly modified marketing strategies; **organizational innovation**, or implementation of novel organizational methods pertaining to business practices, team organization, or external relations, including changes in the architecture of production, management structure, governance, financial systems, and/or employee reward systems; and **product innovation**, or creation and introduction of new technology or significantly changed products, including how they differ from existing products.

Challenge 3. When to stop product development and release, as supported by five companies and six managers. Designing hardware involves many physical constraints (such as material availability, manufacturing limitations, and regulatory standards), thus also limiting design options.[10] Included in a product release decision is also the necessity that a company's product designers nail down all product features prior to production. On the other hand, software development and product release have almost the opposite characteristics, including fewer design constraints[10] typically related to system compatibility with other systems and customer requirements. Software requirements and their design are thus left to the imagination and creativity of requirements engineers, designers, and programmers who can spend time on design and its improvement. This situation can pose serious delay in completing and releasing a product, possibly resulting in missed market windows, as confirmed by several managers. Some of the beneficial characteristics of software in this case also pose a risk in organizations not used to applying management decisions to stop feature development, relying instead on the inherent physical inertia of hardware development.

Companies need to ensure a continuous high degree of visibility and communication pertaining to software design decisions, schedule changes, and development progress. Explicit communication, as well as the ability to coordinate multiple development departments bridging hardware and software, is essential but difficult to achieve in practice. Pernståhl et al.[9] identified that the different traditions and timelines, as well as inherent limitations and enablers of software vs. hardware development, involve new coordination and communication activities, not handled by any current prescribed management process or methodology. Release planning, along with continuous delivery, can, however, potentially alleviate some of these issues, as explored in the online appendix.

C*hallenge 4. Size and complexity explosion, as supported by eight companies and eight managers.* Given that it is easy to keep on developing and expanding software (see also Challenge 3), software is vulnerable to "feature creep,"[14] easily exploding in size and complexity ("messy," as pointed out by one manager) and resulting in architectural degradation. Consequently, it becomes difficult to maintain and evolve software code. This makes it challenging for companies developing software-intensive products to do software-based innovation since they lack experience in software-configuration management and control. Moreover, the ever-increasing software legacy acts as core rigidity, posing further development challenges for radical innovation, making fast changes and addition of features more and more difficult and costly as the product evolves.

"Configuration management" is well established as an engineering practice and can enable more control for a development organization. However, good configuration-management principles[14] can be adopted without becoming rigidly time-consuming, keeping it lean but under control. The point is that the growing software legacy must be managed properly, and explicit decisions taken when to build on, or scrap, legacy. Also, the build-up of legacy requires maintenance of said legacy, while not incurring avoidable technical debt. Overall architectural and product offering choices can also be used as a tool to alleviate complexity, when, say, a product line is introduced as a way to control and maximize potential reuse and, more important, control product variants.

Challenge 5. Critical success factors, or knowing what to develop and for whom, as supported by nine companies and 10 managers. "Since it is easy and relatively quick to develop software, it is challenging to scope and budget software development," as one product manager explained. Moreover, in the case of innovation, the inherent lack of clarity about what to build and the risks involved add further to the complexity. This challenge arises as companies are constantly searching for innovative solutions that can be developed quickly. However, due to their orientation toward hardware development, they lack awareness and training in methods and techniques that might identify the needs of their customers, gauging scope and thus planning product development. In addition, the software code itself is difficult to estimate; software as a product component makes the entire offering more "unpredictable," as one manager put it. Long-term product maintenance and evolution of the product also change when software is introduced, and the decisions taken during development of new features due mainly to software do not account for the long-term maintenance of the software.

In 2014, Porter and Heppelmann[10] reported critical success factors and determinants for developing software-intensive products and services, using, say, early-concept exploration and feasibility assessment and root-cause analysis of customer needs that could be helpful in addressing these challenges. However, few researchers focus on combined hardware-software products. Value estimation, along with practices for scoping and market analysis for selection decisions could be used to address these points.

Learning Perspective

Challenge 6. Tacit knowledge and coordination between software and hardware engineering, as supported by eight companies and eight managers. Engineers have significant tacit knowledge relating to software design, development, and marketing insight. "Specialization and separation of concerns dominate the organizations," as one manager explained it. The same mechanisms that enable specialization also limit coordination and understanding how each task and team contributes to the product as a whole. This is especially challenging in large, complex products like those being developed in the automotive industry. As knowledge is tacit and not communicated, lack of communication can result in problems in terms of misunderstanding and serious system integration conflicts but, more important, also limits the ability to develop new products.[10]

Managers tend to focus on "just my thing" and the principle that if "not developed here" it "does not belong to us," as several managers reported. This behavior is sometimes seen in pure software companies but is aggravated in companies that develop both hardware and software, as the respective teams may be isolated from one another.[10] Failure to take ownership, along

with poor communication, results in hardware teams taking design decisions independently, without consulting software teams, and vice versa. As explained by one product manager, "When the software teams see the hardware, it does not meet their expectations, and, as a result, they suggest modifications which are not welcomed by the hardware teams." Such communication gaps and resistance to form common solutions inevitably cause delays in product design and development.

Challenge 7. Lack of competence in software engineering, as supported by five companies and five managers. Many companies developing hardware-intensive products are not used to the operations, sales, delivery, and development of software. They thus generally lack required expertise and competence. To limit costs, they prefer to either outsource design and development or hire external consultants, a trend that is dangerous, as potential new ideas and products can easily spread to competitors, as mentioned by several managers. A more important aspect of this challenge is that the knowledge and capacity to create software and software-based innovation resides outside the development company. Moreover, doing software development with consultants in-house does not enable companies to evolve themselves and "...putting off the problem to the future when it is even bigger," as one manager explained.

Addressing this point, several managers suggested establishing a dedicated software R&D unit in-house and hiring engineers for software development, helping retain the core knowledge of software-based innovations. However, keeping this knowledge also incurs extra cost, as well as separation between software- and hardware-development teams, potentially complicating coordination (see also challenge 6).

Challenge 8. A rigid state of mind and ability to rethink the product while software becomes a non-trivial component, as supported by nine companies and nine managers. Although traditional knowledge exists, knowledge and expertise pertaining to software development is often lacking.[10] The head of development is often a (former) hardware engineer who seldom has inherent knowledge about software development beyond passing experience. Consequently, management lacks the competence and expertise to understand and solve software-related issues, as mentioned by several managers.

Failure to take ownership, along with poor communication, results in hardware teams taking design decisions independently, without consulting software teams, and vice versa.

Consider software quality, as mentioned by one interview subject, when managers with limited software experience see hardware validation as a complex and precise task, but software, due to its flexible and updatable nature is seen as easily "fixable," even post-release, as stated by one manager. This can have severe consequences, as software is increasingly critical to the main product offering, but such insight is often lacking. Despite genuine ambition to perform rigorous validation, experience and competence to achieve good-enough quality might still be lacking.

While it is possible to use techniques in hardware development for software development, as demonstrated in Wnuk et al.,[15] caution is still needed, as some solutions might not fit within the software-development context, not to mention that product change close to or even up to release represents a challenge for an entire company.

Customer Perspective

Challenge 9. Difficulty estimating perceived value of software-based innovation, as supported by 10 companies and 11 managers. As difficult as it is to estimate the value of the software-based aspects of a traditionally hardware-focused product, putting a price on it is even more difficult. The software-engineering challenge is relevant because, unlike physical products, software is intangible and flexible. Customers do not always "see or feel" a software-based component. One product managers explained it like this: "...in one instance, a car-manufacturing company offered an upgrade feature in its cars at an additional price, through which new features could be added to the car; however, the customers were not ready to pay extra, arguing they already paid an arm and a leg when buying the car, and such services should be part of the initial price of the car." Since the customer could not tangibly see the upgrade feature, its perceived value was not recognized as a benefit.

A strong case needs to be made for software-based innovation that

is shared with marketing and sales people before it is developed. Several managers suggested actively involving customers in the early-idea-generation-and-refinement process. Showcasing ideas to customers, a company can generate early feedback on potential value as perceived by those customers. This input can help devise, identify, and plan for different value propositions for different customers, including, say, whether or not to develop a feature if customers are clearly unwilling to pay for it or if the features are not sellable by the company in question. Several methods support the estimation of software value, though being able to separate software's relative value remains a challenge.

Ecosystem

Challenge 10. Changes in the internal and external ecosystem when software is introduced, as supported by four companies and four managers. The increased size and role of software in traditional hardware-intensive products and services changes the ecosystem and consequently the roles and the players in the marketplace. Consider again electricity meters. Electricity meters are traditionally the core product, and all connecting products are of a supporting nature, supporting the main product, and nothing more. When communication systems for electricity meters became part of the product, the companies in the electric-meter-product ecosystem began exploring metering systems in light of communication (such as what data to store and how to store it). The focus thus shifted from meters as core product to metering systems as core, giving rise to new competitors, including IT companies, entering the marketplace. If a company cannot cope with such competitive change, there is greater risk it will be reduced to mere component supplier, with profit margins decreasing over time, as mentioned by several managers. This challenge calls for companies to forecast changes in the ecosystem and proactively plan to address them so as not to lose market share. This implies the development company's internal ecosystem needs to be as flexible (changeable) as the external ecosystem[10] (see the online appendix).

Conclusion

Software is a fundamental component in the final product offering in the 12 companies studied and thus constitutes a significant aspect of their ability to create new products, posing challenges, as identified by the managers interviewed. The feeling among them is they have persisted over the past decade and continue to pose limitations on the potential possible today through software as part of a product offering. While some challenges have been discussed and researched (see the online appendix), further research is needed. We also found many industry partners view themselves as isolated, thinking they are the only ones confronting these challenges or at least falling behind on the learning curve. Our experience, supported by the study, shows this to not be the case. Many companies in the study face such challenges. One manager said, "We are looking for solutions and good ways to follow, but the consultants, even the expensive experts, seem to only be able to give us general advice...not much practical help. We even looked to science ourselves, but the information there is all over the place, and it is hard to see what works..."

For managers and other practitioners, the study's main takeaway is that you are not alone. For researchers there is a need to come up with actual solutions that are tested in practice and offer scalable help. Many companies developing software-intensive products are still learning how to be "soft," and some related challenges are not solved in practice or at the very least were not perceived as solved by the 12 companies in the study.

One issue is how to separate out the relative value of software in a complex product offering. The online appendix suggests that "value" is subject to research, but separating the relative value of software is not easy. Another issue is how to get the technological, knowledge-based, mind-set-based transition to include the benefits (and drawbacks) software promises. Most managers in the study realize there is a need for specialized software-engineering competence to tackle many of the challenges but find "solutions" to be lacking. This may be due to gaps in research or in industrial transfer of viable solutions or a combination of both. In any case, the challenges persist, though some managers might disagree. Our intention was not to map challenges to solutions but rather to present 13 current views from 12 different companies that are, or have recently, undergone a transition toward being more soft, and these are their stories. C

References

1. Athey, T. Leadership challenges for the future [of the software industry]. *IEEE Software 15*, 3 (Mar. 1998), 72–77.
2. Broy, M., Kruger, H., Pretschner, A., and Salzmann, C. Engineering automotive software. *Proceedings of the IEEE 95*, 2 (Feb. 2007), 356–373.
3. Edison, N. and Torkar, R. Towards innovation measurement in the software industry. *Journal of Systems Software 86*, 5 (2013), 1390–1407.
4. Khurum, M., Gorschek, T., and Wilson, M. The software value map: An exhaustive collection of value aspects for the development of software-intensive products. *Journal of Software: Evolution and Process 25*, 7 (July 2013), 711–741.
5. Khurum, M., Fricker, S., and Gorschek, T. The contextual nature of innovation: An empirical investigation of three software intensive products. *Information and Software Technology 57* (Jan. 2015), 595–613.
6. Lane, J.A., Boehm, B., Bolas, M., Madni, A., and Turner, R. Critical success factors for rapid, innovative solutions. *New Modeling Concepts for Today's Software Processes, Lecture Notes on Computer Science 6195*, 2010, 52–61.
7. Leon, A. *Software Configuration Management Handbook, Third Edition.* Artech House, Norwood, MA, 2015.
8. Leonard-Barton, D. Core capabilities and core rigidities: A paradox in managing new product development. In *Managing Knowledge Assets, Creativity and Innovation*. World Scientific, 2017, 11–27.
9. Pernstål, J., Magazinius, A., and Gorschek, T. A study investigating challenges in the interface between product development and manufacturing in the development of software-intensive automotive systems. *International Journal of Software Engineering and Knowledge Engineering 22*, 7 (Nov. 2012), 965–1004.
10. Porter, M. and Heppelmann, J. How smart, connected products are transforming competition. *Harvard Business Review 92*, 11 (Nov. 2014), 64–88.
11. Robson, C., McCartan, K., Robson, C., and McCartan, K., *Real World Research, Fourth Edition.* Wiley, West Sussex, U.K., 2016.
12. Saldana, J., *The Coding Manual for Qualitative Researchers.* SAGE Publications, Inc., Thousand Oaks, CA, 2012.
13. Schief, M. and Buxmann, P. Business models in the software industry. In *Proceedings of the 45th Hawaii International Conference on System Sciences* (Maui, HI, Jan. 4–7). IEEE Computer Society Press, 2012, 3328–3337.
14. Ur, S. and Ziv, A. Cross-fertilization between hardware verification and software testing. In *Proceedings of the Sixth International Conference on Software Engineering and Applications*. (Cambridge, MA, Nov. 4–6). The International Association of Science and Technology for Development, Calgary, AB, Canada, 2002.
15. Wnuk, K., Gorschek, T., and Zahda, S. Obsolete software requirements. *Information and Software Technology 55*, 6 (June 2013), 921–940.

Tony Gorschek (tony.gorschek@bth.se) is a professor of software engineering in the Software Engineering Research Laboratory Sweden at Blekinge Institute of Technology, Karlskrona, Sweden; http://www.gorschek.com/doc/start.html