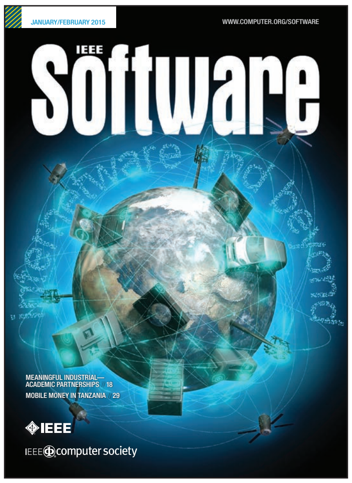
Building the Community of Leading Software Practitioners


JANUARY/FEBRUARY 2015
WWW.COMPUTER.ORG/SOFTWARE
IEEE
Software
MEANINGFUL INDUSTRIAL–ACADEMIC PARTNERSHIPS 18
MOBILE MONEY IN TANZANIA 29
IEEE
IEEE computer society

What Next? Advances in Software-Driven Industries

Christof Ebert, Gerd Hoefner, and Mani V.S.

January/February 2015
Special Issue on *Internetware and Beyond*

IEEE computer society
www.computer.org/software

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

What Next? Advances in Software-Driven Industries

Christof Ebert, Gerd Hoefner, and Mani V.S.

Which software technologies will be relevant in the near future? Where is development of innovative products and solutions heading? Companies and engineers need to flexibly respond to markets and changes and adapt their competences. But which of the many new technologies and hypes deserve focus? Further progress in software technologies, coupled with the growing capability to reliably develop complex systems, will impact every engineer. In this installment of Software Technology, Siemens experts Gerd Hoefner and Mani V.S. and I outline major software trends and offer recommendations for practitioners. I look forward to hearing from both readers and prospective column authors about this column and the technologies you want to know more about. —*Christof Ebert*

COMPLEXITY SCALES. With society's increasing dependence on software, its complexity and scale continue to expand. Furthermore, while becoming more pervasive, software is also becoming more transparent, with increasingly invisible interfaces. Software is transforming entire industries such as healthcare and the automotive industry. Electronics, software, and IT are major drivers of innovation across all industries.

This article provides an overview on trends in software-driven industries. We have spoken with clients from technology companies worldwide in various industries to identify where they are heading and what topics are relevant for 2015 and beyond. We will outline major trends and supplemented these trends with concrete recommendations.

Complexity scales. Simplicity secures.

Products and solutions must meet more and more requirements but also must be designed for cost effectiveness, easy adaptability, and the ability to exploit the advantages of emerging and dominant industry platforms. New competitors are entering markets with new solutions, which in some cases circumvent the dependence on legacy systems. Software-driven systems are characterized by rapidly growing complexity coupled with an unprecedented increase in scale, based on a fast-changing technology landscape.

Software Advances

The Billion-Dollar Startup Club of the *Wall Street Journal* and Dow Jones

0740-7459/15/$31.00 © 2015 IEEE

VentureSource tracks companies that venture capital firms have valued at more than US$1 billion (http://graphics.wsj.com/billion-dollar-club). It's an interesting source to compare technologies and their expected impact on worldwide needs. Currently, the growing list has 60-odd companies.

The Five Dimensions of Software Advancement

Looking at these companies along with our own wealth of industry contacts, we cluster software advancement along five dimensions (see Figure 1):

- *collaboration*—consumer Internet, social network interaction, single-customer segmentation, configurators for products and services, digital money, computer-assisted collaboration tools, and crowdsourcing;
- *comprehension*—semantic search, big-data handling, smart data, data analytics, the data economy, online data validation, and data quality;
- *connectivity*—ubiquitous mobile computing, mobile services, cyber-physical systems, Industry 4.0, machine-to-machine communication, sensor networks, and multisensor fusion;
- *cloud*—applications and services in the cloud, location-based networks, new license models for software and applications, sustainability, and energy efficiency; and
- *convergence*—mobile enterprises, bioinformatics, the Internet of Things, pervasive sensing, and autonomous systems.

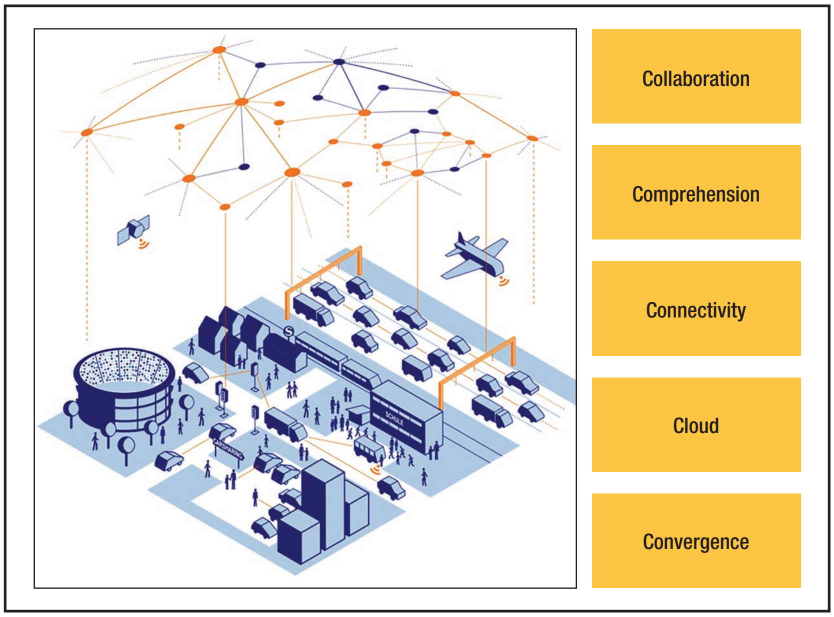


FIGURE 1. Software is advancing along five dimensions—collaboration, comprehension, connectivity, cloud, and convergence. These dimensions, coupled with the underlying complexity and scale, demand new software solutions based on new computing paradigms and infrastructure.

These dimensions, coupled with the underlying complexity and scale, demand new software solutions based on new computing paradigms and infrastructure. Examples include IT architectures that facilitate seamless connectivity, robust infrastructures for cyber-physical systems in safety-critical environments, and data analytics to predict choices and behaviors to improve the overall customer experience. Such software-driven solutions can create nontraditional market entry points and consequently entirely new mechanisms to address a single customer with time- and location-specific services.

Security, Robustness, and Usability

Such solutions require new technologies that will not only create numerous opportunities but also introduce complexity. So, these solutions introduce new challenges—for instance, regarding information security, robustness, and usability.

Security and robustness have a tremendous impact on business decisions. The more we share and network, the more we're exposed to attacks of all kinds. The exploding need for secure software and protection schemes for our business processes, end-to-end, indicate this impact. Imagine automotive suppliers working on multisensor fusion connected to GPS and vehicle-to-vehicle communication to predict critical situations and foresee appropriate measures in situations in which even the driver might not be aware of what will happen. Other examples are service companies that leverage their sales channels to flexibly provide related services such as door-to-door transportation, or firms that offer a single service card for identification, payment, and access to services of various providers, both physical and in the cloud.

Complexity and scale demand a focus on usability. We already face situations in which inadequately trained

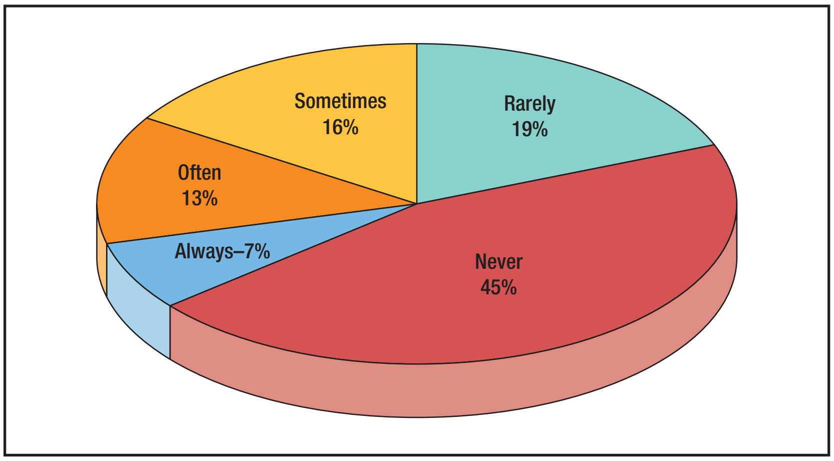


FIGURE 2. Only a small percentage of features actually create value in a software-driven system.

users are forced to operate systems they don't understand sufficiently to meaningfully assess risks and stay in control across normal day-to-day scenarios. Insufficient usability is a major source of critical failures caused by humans in healthcare, transportation, and production plants.

Creating Value

Software and IT move on a fast highway. Global software development requires managing software projects that span geographic and organizational boundaries, which adds to the challenge of developing software. But we see many companies and endeavors that failed because they overemphasized technology and didn't sufficiently implement a sound business strategy.[1]

Consider Netscape. For many of us, Netscape was our first experience of the Internet. In 1995 it had a market share of 80 percent. But by 1997, it slowed down and lost market share, and in 2003 it went bankrupt. What went wrong? One manager put it simply, admitting that instead of using sound processes, they just dumped features and software technologies into their products.[1] Insufficient product management still hampers companies. Research has shown that roughly half of delivered features don't create value (see Figure 2).[1,2]

The Complexity Trap

While working with clients on product strategy and requirements engineering, one of the first questions we ask is, what value will a potential feature add to a product? The vast majority of responses we receive can be reduced to "We don't know; the spec says this feature is required." Although this might be true, it's important to recognize that this will lead to an unsustainable increase in complexity and cost. This is what we call the *complexity trap*.

As complexity expands, it must be balanced. Companies that make the wrong decisions during a period of fast technology evolution and change will fall back or fail. Most selections involving human choices follow a "long tail" or $1/f$ distribution. The Pareto principle states that, for many events, roughly 80 percent of the effects come from 20 percent of the causes. Today, economists have put more energy into analyzing the long-tail distribution; they've found that, for instance, going from 20 to 100 features in a system adds only 10 percent of value.[3] Steve Jobs was one of the few taking concrete lessons from this principle, demanding simplicity in his products. Yet managers are unsure whether the measures taken are sustainable and how to manage the increasing innovation pressure. Specifically, they expect technology and IT proposals to both reduce costs and set the right priorities for technology innovation and new products and solutions.

Impacts for the Leading Practitioner

Simplicity secures. Too often software engineers are overly fascinated by new technologies. In addition, their companies obviously demand rapid transition to new products and services, to create sales opportunities and growth. However, in the interest of balance, you need to combine evolving software technologies with sound engineering and management practices.

In talking with clients in many different projects and industries worldwide, we identified four major levers to avoid an overly narrow focus on technology:

- Connect architecture and functionality.
- Master the entire life cycle.
- Strengthen globally distributed teams.
- Streamline development.

We now examine these levers and offer recommendations for practitioners. Most of the recommendations are further detailed in *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing.*[2]

Connect Architecture and Functionality

Software and IT systems need a close connection of architecture and functionality. Any system's requirements and architecture are highly interdependent. Projects without early architecture evaluation will likely overlook critical requirements, especially nonfunctional ones. In addition, these projects are more likely to fail in prioritizing requirements. They typically face budget overruns and rework due to inappropriate and late design changes.

The following example illustrates this disconnect. A project team introduced a service-driven approach but did not adequately align the architecture and requirements. The team developed and modeled the requirements without looking to solution models and architecture constraints. They modeled business processes with a top-class modeling tool but didn't map the processes to the requirements. The intended service-oriented architecture copied previous technology-driven silos because this was the language everybody understood. The result was a mixture of individual components that didn't perform according to business expectations. Getting the project back on track required restarting the entire requirements-modeling and architecture concept—at a high cost of rework.

Modeling functions and architecture will help avoid such failures. Functions, architecture, implementation, and dependencies must be modeled and connected suitably. It's essential to evaluate nonfunctional requirements such as safety, security, performance, reusability, maintainability, and system cost—and their impact on architecture—at an early stage. Product line development with efficient variant management has a significant economic impact. The recent evolution toward modular concepts is a positive step. Working at a higher abstraction level and automating activities can improve productivity and quality. The trend is "from field to system to lab to math." Development is getting increasingly virtualized, so that the code runs on multiple platforms.

The greatest obstacle is the learning curve, thus not achieving consistency across organization units. Developers are tasked with optimizing the code without getting the time or training to understand its function. Systems engineering remains in the background and is isolated from application development. So, roadmaps are created for only the subsystems, and new features and variants introduce overwhelming complexity. The business case is clear: Consistent modeling of the product's critical software components significantly reduces the defect rate and development costs. We typically find a share of 30 to 50 percent as adequate; that is, rarely used uncritical parts should not be modeled.

Recommendations: First, establish a strong focus on systems engineering and modeling of functions and architecture. Define quality requirements and measures early and consistently from a system point of view. Break the system down to its components and functions and analyze the impact of requirements on architecture. Integrate suppliers and customers into your overall quality and life-cycle concept. Reduce the nonconformance cost through integrated modeling, early defect detection, and reuse. Ensure consistency through traceability, automated consistency checking, and automated code generation from models of functionality.

Move stepwise toward model-driven development, and focus on critical components and consistency of requirements throughout architecture, design, and test cases. Measure the migration and its effects. In each project, try to improve by 10 to 20 percent in areas on which you want to focus—for example, 20 percent fewer cost variations or 10 percent less cost in the test phase.

Master the Entire Life Cycle

Rising cost pressure is forcing companies and their suppliers to jointly and consistently master product development. Product life-cycle management and application life-cycle management (PLM and ALM) are the primary mechanisms for integrating engineering processes, tools, and people across the domains of system, software, hardware, and mechanical engineering. Unfortunately PLM and ALM often aren't well introduced. Companies believe that with a tool and the necessary IT interfaces, all issues will be resolved. This isn't the case; the high percentage of abandoned PLM and ALM projects indicates the criticality of professional change management.

The following example illustrates the significance. A supplier introduced model-driven engineering (MDE) based on a modern tool environment that enabled seamless collaboration across development centers and with partners and customers. Cost-effectiveness was evident up front because the system was going to provide faster data access, while the improved change and configuration control was expected to produce fewer defects and reduce budget overruns.

The engineers used the tool for modeling during design, but not during the test, and without much modeling methodology. Soon, models became inconsistent and were

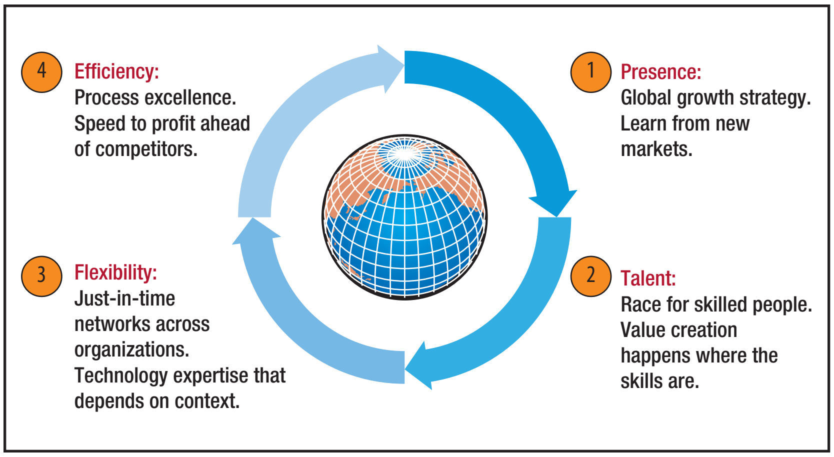

4 Efficiency:
Process excellence. Speed to profit ahead of competitors.
1 Presence:
Global growth strategy. Learn from new markets.
3 Flexibility:
Just-in-time networks across organizations. Technology expertise that depends on context.
2 Talent:
Race for skilled people. Value creation happens where the skills are.

FIGURE 3. The way ahead: four drivers fuel future globalization.

discarded in further product evolution. What had happened? The tool was designed to support the development and was integrated into the company-wide product data management system. However, not only developers but also product managers and project leaders couldn't work with the tool and created parallel systems for their documents, which they exchanged using traditional methods. The solution would have been effective if, before the tool was introduced, it had been made clear which processes had to be supported along the life cycle with which methodology, and how these processes had to be *first* improved and then automated.

Recommendations: First improve the process and then the tools, on the basis of concrete improvement objectives that are set, measured, and used to correct deviations. Ensure consistency of features and products with a strong systems-engineering approach. Specifically, in distributed collaborative environments, we see huge benefits from a single repository for consistent requirements, specifications, and models across all versions. Use tools to appropriately model the different abstraction levels, from functions to logic and from architecture to implementation. Evaluate tools on the basis of your own requirements under realistic conditions. Support the interfaces to the various components and processes through traceability, automatic consistency checks, and test automation.

Manage the transformation across the entire organization. Pilot the changes in process and tools, coach and train engineers, and recognize the power users who will set the pace. Introduce model-based development intelligently; step by step, focus on critical components, continuity of requirements throughout code and test cases, and improving processes in parallel. Support developers and ensure continuous improvement.

Strengthen Globally Distributed Teams

The pressure to continuously innovate and reduce cost, the lack of the right skills, and the need for global presence will further boost distributed development. Figure 3 shows the four major drivers of global software engineering:

- *presence*—in local markets, both for visibility and access but also for learning;
- *talent*—to succeed in the race for the best engineers;
- *flexibility*—with just-in-time networks and emerging ecosystems and crowdsourcing; and
- *efficiency*—to balance the complexity and simplicity in reducing overheads and being more agile.

Distributed development teams require new forms of collaboration for teams, projects, and people. The diverse network of components, applications, devices, and users is demanding completely new ways of working. However, communication difficulties, cultural differences, and management overhead lead to numerous challenges. Eighty percent of companies that outsource their development or maintenance have problems. Twenty percent of sourcing contracts are canceled within the first year. Fifty percent of the contracts don't achieve the intended objectives and are then terminated.[2]

So, what's needed is *smartshoring*, which replaces traditional labor-cost-based location decisions with a systematic improvement of business processes in a distributed context. The benefits are tangible; the most often reported ones are multisite collaboration, clean-variant management, and transparent workflows. Merely considering labor costs must be replaced by a holistic strategy taking into account onsite presence and customer proximity as well as reducing friction losses.

Recommendations: Prepare distributed teams and smartshoring as a competence and business process before going operational. This involves risk management, vendor selection, optimizing your own processes, and transparent controlling. Use computer-assisted collaborative-engineering tools to facilitate seamless work across distributed teams. Set clear objectives for improvement, and measure and monitor progress against the agreed targets. Ensure a high level of discipline in distributed teams by transparently controlling both projects and results. Use tailored methods and tools for matching your own constraints and needs. Improve competences for distributed development and soft skills. For instance, use different communication channels rather than sending only email. Benefit from smartshoring and global engineering industry best practices by visiting the IEEE International Conference on Global Software Engineering (ICGSE; see the sidebar).

ICGSE

The annual International Conference on Global Software Engineering (ICGSE) brings together worldwide industry and research leaders in distributed software development. With participants from more than 20 countries and one-third of the papers from industry, it's the preeminent forum on global software development. ICGSE 2015 will take place from 13 to 16 July in Ciudad Real, Spain. Join the conference and learn how to succeed with distributed software projects. Meet the authors of this article, and learn from their keynote speeches how to make distributed teams more effective, the benefits of smartshoring, and how to cope with the challenges, such as heterogeneous methods and tools. More information is at www.icgse.org.

Participate in our technologies 2015 survey and have the chance to win a free copy of our IEEE book *Global Software and IT* with many practical case studies. Directly go to the survey: www.vector.com/trends-survey.

Streamline Development

In uncertain economic times, technologies and processes that make a company more efficient and powerful gain attention. Software development managers must evaluate costs and productivity and implement goal-oriented improvements. In our experience, you can reduce cost and cycle times by continually optimizing development processes. The application of lean-software-development principles and agile practices can effectively streamline interfaces and reduce rework and inefficiencies. Agile practices such as Scrum or Extreme Programming are increasingly applied but often seem a mere slogan instead of a sustainable way to work.

Our experience shows that far too often, teams reinvent the wheel. Earned value, value stream mapping, and Scrum are proven techniques that shouldn't be developed in-house with a lot of energy and cost. Our clients at Vector claim that nearly 90 percent of companies want to improve their efficiency in 2015 but that only a third of them are satisfied with their previous results. Unprofessional change management is a common reason for failing efficiency projects.

Recommendations: Check your project performance: what creates pressure in projects, what demotivates your teams, and where do work products need rework? Streamline workflows and related tools stepwise, with an overarching strategy, incremental goals, and a future-oriented IT architecture. Set concrete improvement targets every quarter. Train employees in lean principles. Have each team develop its own action plan for reducing waste, rework, and interface conflicts—with reference to your company-wide efficiency targets. Evaluate your performance—for example, by revenue per developer, lead time, fault detection rates, and cost drivers. Ensure that agile practices don't lead to arbitrariness. Apply professional change management.

Complexity Scales, Simplicity Secures

2015 calls for more efficiency and competitiveness, because the business climate is more volatile. We talked to our clients from various industries to identify where they are and what topics will be important for product development in 2015 and beyond. The response: companies will continue to invest in growth through innovation by developing new products and solutions because this determines their market position. At the same time, they're aware of the volatile market situation and want their development teams across the world to be as lean and efficient as possible.

These days, companies must be doubly innovative—in both technology and efficiency. The ability to successfully implement innovations in a short time is the all-important competitive factor. Innovations are not only new products and optimized processes but also entirely new basic technologies, such as we see in electric vehicles, communication networks, or even intelligent energy use.

Insufficient use of road maps, unmanaged complexity, and cost-cutting in the wrong areas lead to a situation in which a significant proportion of R&D expenditure doesn't lead to successful innovation. Customers in various industries complain about overly long cycle times from idea to market. Techniques such as lean innovation can address 20 to 40 percent of customers' total cost structure. This is an enormous savings potential that should be effectively and sustainably captured.

What determines a product's success isn't the number of features; it's the few that differentiate it from others. Complexity scales but must be mastered with product strategy, sound engineering processes, and technology management to achieve the necessary simplicity that secures your growth and sustains your markets. SW

References

1. C. Ebert and S. Brinkkemper, "Software Product Management—an Industry Evaluation," *J. Systems and Software*, Sept. 2014, pp. 10–18.
2. C. Ebert, *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing*, John Wiley & Sons, 2012.
3. T. Hey, S. Tansley, and K. Tolle, *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Microsoft Research; http://research.microsoft.com/en-us/collaboration/fourthparadigm/contents.aspx.

CHRISTOF EBERT is the managing director of Vector Consulting Services. He serves on the *IEEE Software* editorial board. Contact him at christof.ebert@vector.com.

GERD HOEFNER is the managing director and CEO of Siemens Technology and Services in Bangalore. He's also the head of Siemens' Corporate Development Center. Contact him at gerd.hoefner@siemens.com.

MANI V.S. works on the marketing team of Siemens Technology and Services in Bangalore. Contact him at vs.mani@siemens.com.