These days, ICT environments have dramatically changed in their coverage and complexities, partly owing to the complexity of modern distributed architectures of service cooperation and partly because of mission-critical services and applications, including online shopping and transactions, and autonomous cars.

Consider online shopping. Today, e-commerce is rapidly growing all over the world. Statistics show that in 2014, e-commerce accounted for more than 6 percent of retail sales in the US (www.census.gov/retail/). Market growth in China and India is also remarkable. Several software products are involved in completing a transaction, including webservers, Transport Layer Security (TLS) libraries, and Web browsers. For e-commerce, note that security is the first priority in acquiring customer trust. Nobody wants to visit a site where their credit-card details are easily exposed to others. But are our browsers, communication libraries, and webservers correctly coded against possible attacks? How can we prove it?

As another example, consider an autonomous car, a typical Internet of Things (IoT) application. Its controller area network (CAN) communicates with the sensors attached to the car and makes some decision based on the information from the sensors, GPS, and the Internet. Major automobile vendors, as well as IT vendors such as Google, are hurrying to launch new products in the automobile market. In this case, safety is the first priority in acquiring customer trust. Nobody wants to get into a car that is unsafe and can crash into other cars. To instill user confidence, how can we prove and demonstrate that the complicated controllers in a car are correctly configured?

Let’s examine the problems and potential solutions that help to prove or demonstrate correctness in ICT environments that are becoming more complex and interlinked.

**Complexity and Correctness Concerns**

The growing complexity of control causes a major concern: nobody can guarantee the “correctness” of the control. Consider an e-commerce transaction between a browser and a site. Operating software suites form a multilayer complex. Moreover, some sites might federate with other sites (see Figure 1). Here, we see that sites are distributed over the Internet and that they communicate using some standard protocols. Does each protocol work correctly? If so, does each protocol have a correct implementation? If so, does the federated system work correctly as a whole? In the past, we’ve had some bitter experiences in which flaws were pointed out in protocols that were believed to correctly work, bugs were pointed out in software that was believed to correctly work, and ill-configurations were pointed out in software that worked securely in the correct configuration. Thus, there are good reasons to suspect a system of incorrect operations.

Of course, some control programs operate under different policies. Advanced services are often leveraged using cutting-edge artificial intelligence technologies, which can create innovative solutions. However, these programs generate solutions that cannot be validated for correctness and effectiveness for a long time.

However, we definitely have services and applications that demand some guarantee of correctness. The most practical and trustworthy approach is likely to hire talented programmers and rely on their wizardry skills. However, we also know that even a programming wizard can overlook bugs, which is partly reflected by the Common Vulnerabilities and Exposures (CVE) dictionary (http://cve.mitre.org). So, we need a rigorous approach for guaranteeing program
correctness. Furthermore, we have to take several additional factors into consideration when demonstrating correctness, security, and safety, including human factors and interactions with different environments. Rigorously guaranteeing correctness for such complicated systems is a tough task.

**Methodologies for Verifying Correctness**

Let us examine what has been tried so far for verifying correctness in the fields of software engineering.

*Verification* is the effort to rigorously prove the correctness of given programs. It is focused on the formal description and verification of program behaviors. Early achievements include specification languages such as Z (ISO/IEC 13568:2002) and Vienna Development Method Specification Language (VDM-SL; ISO/IEC 13817-1:1996), and (early) model checkers such as Spin (http://spinroot.com/spin/whatispin.html). Such languages have been partly successful in verifying small programs, small systems, and simple protocols. Recently, Kim Schaffer and Jeffrey Voas presented a discussion on the application of verification to practical security. However, it is believed that these languages’ scalability is too poor to be practical.

*Validation* is a looser but more practical approach. It is characterized as the process to ascertain whether software satisfies a given specification. Software testing is a typical process for validation in which a given software is tested against its specification using a suite of test patterns. Testing levels are layered as unit, functional, integration, interface, and system. Testing methodologies include acceptance, ad hoc, automated, boundary, and path testing. Today, a large part of the software production process is occupied with testing.

Among correctness properties, security occupies a special position. Today, a number of mission-critical services place top priority on security. Security breaches might be caused by insecure (incorrect in terms of security) programs or by insecure operations. In addition to conventional program verification, program validation, and adoption of appropriate security technologies, we need governance over system configuration and operation. For better governance, dedicated methodologies for organizational control, such as the Information Security Management System Standard (ISO 27001), can be adopted.

In the context of Internet services, a new direction for establishing correctness has emerged: trust. In addition to long-term efforts in validating service correctness and security, experts have proposed the notion of a “web” of trust, meaning that we evaluate services via public scrutiny together with a priori anchors of trust. Note that this methodology never provides us with mathematical correctness. However, we know that this trust solution shows practical effectiveness. Instead of formal mathematical proof of correctness, our attention is now focused on practical correctness assessed by users.

Note also that certification is a typical embodiment of trust. Instead of directly validating a service’s correctness, we rely on a given authority for the correctness guarantee. This reduces social costs and provides service scalability. Public-key infrastructure (PKI), WebTrust, and the infrastructure of certification for TLS/SSL certificates are among the most significant deployments of the web of trust (www.webtrust.org). PKI is now recognized as one of the most successful trust deployments on the Internet. Without PKI, we wouldn’t have seen e-commerce flourish so successfully today.

What is a suitable approach for ensuring practical correctness in the real world? Table 1 summarizes the aforementioned methodologies for checking correctness and their key features. To guarantee correctness, the approach could be rigorous (mathematical) or practical. Verification is grounded in mathematics, while trust is grounded in evaluation or
From the editors

Table 1. Correctness methodologies.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Principle</th>
<th>Rigor</th>
<th>Scalability</th>
<th>Techniques/approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification</td>
<td>Mathematics</td>
<td>High</td>
<td>Poor</td>
<td>Automated reasoning; model checking</td>
</tr>
<tr>
<td>Validation</td>
<td>Test suites</td>
<td>Moderate</td>
<td>Middle</td>
<td>Testing</td>
</tr>
<tr>
<td>Governance of operation</td>
<td>Organizational control</td>
<td>Low</td>
<td>Good</td>
<td>Information Security Management System (ISMS) standard</td>
</tr>
<tr>
<td>Trust</td>
<td>Public evaluation; certification by recognized authority</td>
<td>Low</td>
<td>Good</td>
<td>Certification</td>
</tr>
</tbody>
</table>

society’s expectations. Although operation governance does not guarantee correctness, we know that it will work for well-organized institutions. Note that there is a strong requirement for guaranteeing correctness for practical systems and services. For validation, we cannot have exhaustive testing against a given specification in general, but deliberate selection of test suites would effectively work even for complicated programs.

Recent Advances

With growing computing power and evolving theories related to correctness, control, and governance, we are now (partially) ready for guaranteeing the correctness of practical systems. I now highlight further advances that have influenced our society.

Verification

Continuous efforts have been made to apply verification methodologies to more complicated programs. For example, CompCert (http://compcert.inria.fr) aims at formal verification of usable compilers. Note that compilers are classified as one of the most complicated systems.

In addition, protocol verification has received significant interest. For instance, we can observe successful results at verifying complicated protocols used in federated services on the Internet—such as the Security Assertion Markup Language (SAML) and OAuth-2—using modern model checkers such as the SAT-Based Model-Checker for Security Protocols and Security-Sensitive Applications (SATMC; www.ai-lab.it/satmc/). These checkers can also reveal protocol flaws, if any, which leads to re-specifying secure configurations. For instance, SATMC actually found an insecure configuration of SAML.²

Today, we can say that protocol verification is completely practical. Nevertheless, it’s a problem that will never be fully resolved. Particularly, to effectively work, most modern network protocols are supported by browser scripts and plugins. Verifying these protocols includes browser behavior, which will substantially raise the complexity of verification.

Furthermore, the verification for a kind of implementation is within our scope—that is, by verifying “reference implementations.” Here, program verification is a target, in addition to protocol verification. As reported in CVE (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224), one of OpenSSL’s vulnerabilities (https://www.openssl.org/news/vulnerabilities.html) was identified using the Coq proof assistant (https://coq.inria.fr/) while OpenSSL implementation was being checked against the formal description of SSL.

Validation and Testing

For validation, automated testing has been a major focus because manually providing an effective suite of test patterns is almost impossible due to the growth of complexity in software and environments. Automated testing schemes available as software packages or cloud services include Randoop (http://randoop.github.io/randoop), JWalk (http://staff-www.dcs.shef.ac.uk/people/A.Simons/jwalk/), and Evosuite(www.evosuite.org). To improve coverage, it has been made clear that automated random testing must be guided with static analysis and dynamic feedback. Randoop is feedback-directed random testing, whereas GRT³ is guided by program analysis. These testing schemes are proving to be practical at finding bugs in real products.

Furthermore, the coverage of automated testing has been extended. For example, in addition to conventional stand-alone software products, automated testing schemes for GUI and Web applications are now available. We know that there is a strong demand for software testing of distributed environments, which has been a challenge because of communication and related problems. Centralization has been proposed as one approach to solving these kinds of problems.⁴ By using centralization techniques, software testing can be applied to a set of programs in a distributed environment in general. Centralization can also extend to the software testing of product lines,⁵ which is significant for testing and
managing multiple versions of a product line. These kinds of extensions are very important when we apply software testing to real-world software.

**Trust and Certification**

Building trusted infrastructures in the Internet by governance and trust, and thus guaranteeing correctness within the platform, has also shown remarkable progress. The US is leveraging Internet services via the online identities of companies such as Google and Facebook and initiatives such as InCommon (www.incommon.org). The National Strategy for Trusted Identities in Cyberspace (NSTIC; https://www.nist.gov/itl/nstic), launched in 2011, aims at building the infrastructure of trusted online identities. The Trust Framework Provider Adoption Process (TFPAP) from Federal Identity, Credential, and Access Management (FICAM) is in operation as a certification process of online identities used for federal services. Here, a trust framework for identities and services is defined as its operating body. In a given trust framework, levels of identity assurance and their certification criteria are defined and published. Relying parties (service providers) accept the identities certified in the trust framework as the trusted identities with some assurance level.

Kantara (https://kantarainitiative.org) is a typical trust framework certification body in TFPAP, in which some academic access federations in the EU and Japan are also enrolled. The certification criteria consist of three components: governance criteria, technical criteria on authentication and security, and privacy criteria. Note that the graded correctness of identities (levels of identity assurance) is assessed on the basis of governance of operations and technologies for federated access control. Figure 2 illustrates the correctness schemes behind TFPAP. We can see that real-world trust is also supported by verification and validation technologies in addition to governance and certification control.

FICAM certification intended for use in federal online services will expand to general Internet services. Particularly, establishing trust for IoT environments is very important. In addition to control of enormous devices, we need control over wireless communications. Requiring rigorous security and privacy in the IoT environment is an almost impossible task. Instead, the trust solution will effectively work. We can control IoT by graded control criteria over devices and wireless communications. Requiring rigorous security and privacy in the given IoT environment is almost impossible task. Instead, the trust solution will effectively work. We can control IoT by graded control criteria over devices and wireless communications according to their security strength. The control criteria might be loose if we do not require rigorous security in the given IoT environment, and must be strict if the IoT environment supports mission-critical services. Furthermore, the IoT must be given a trust anchor. One study considers mobile phone carriers to be the trust anchor of the IoT, which indicates that ICT environments including the IoT must be grounded on social infrastructures.

So far, we have discussed the establishment of trust by certification, which has been proved effective. However, the cost of its management and operation has also proved high. Therefore, current focus is on another aspect of trust: reputation. Reputation-based trust is established as a collection of scrutiny by individual participants on the Internet. Because it does not assume any authority, we cannot fully rely on such evaluations. However, observing the success of Bitcoin (https://bitcoin.org), it might effectively work if an appropriate scheme for collecting scrutiny can be established. Actually, the trust judgment of Bitcoin relies on Byzantine agreement in distributed environments.

I have examined several approaches and technologies for proving or ensuring...
correctness. In this complicated real world, we need practical guarantees of program correctness to put our lives in the hands of ICT systems. Although we must admit that related technologies need further improvement, we can consider applications of correctness technologies to be more complicated systems. Here, correctness is not just “true” or “false,” but requires graded values based on risk analysis. Even if we cannot rigorously prove program correctness using verification and validation techniques due to a program’s complexities, we might improve the degree of correctness through a formal statement from an authority, or by social evaluation.

To tackle the complexities of mission-critical services and systems, we must combine the techniques of correctness, enhance its definition, and leverage these techniques to tackle the correctness problem. We know that Internet trust solutions are built on the combination of governance of operations and security technologies. This is the practical correctness that we consider to be a part of social infrastructure. Here, social scrutiny and risk analysis must be discussed together with verification and validation technologies. For those of us involved in ICT in academia or industry, this practicality is essentially significant.

References

Hiroyuki Sato is an associate professor in the Information Technology Center of the University of Tokyo. His interests include digital identity technologies, information security, and programming language design and implementation. Sato is accredited as a Level of Assurance 1 (LoA 1) assessor by the Kantara initiative (https://kantarainitiative.org), established to accelerate the adoption of digital identity relationship solutions to support and simplify our connected world. He received a PhD in information science from the University of Tokyo. Contact him at schuko@satolab.itc.u-tokyo.ac.jp.