Software Engineering for Compliance

Uwe Zdun, University of Vienna
Ayse Bener, Ryerson University
Erlinda L. Olalia-Carin, KPMG Canada

THE TERM COMPLIANCE addresses the external regulations, internal policies, standards, and governance to which an organization must adhere. In general, compliance in the context of information systems means ensuring that an organization’s software and systems comply with multiple laws, regulations, and business policies. Compliance imposes certain IT controls that focus on information creation and retention, as well as on its protection, integrity, and availability. This is a major issue in many organizations because non-compliance might lead to severe financial penalties and reputational risks.

Software engineering for compliance is needed in response to regulations such as the BASEL III, a global regulatory standard on bank capital adequacy, stress testing, and market liquidity risk agreed upon by the members of the Basel Committee on Banking Supervision in 2010; the International Accounting Standards Board’s International Financial Reporting Standards; the EU’s Markets in Financial Instruments Directive (MiFID; concerns competition and consumer protection in investment services), the French financial security law (Loi de Sécurité Financière, or LSF; concerns legal provisions relating to corporate governance), the US Title 21 CFR Part 11 (concerns privacy issues in electronic record-keeping), the US Health Insurance Portability and Accountability Act (HIPAA; concerns among other things security and privacy of health data), the Dutch Tabaksblat Code (concerns corporate governance), and the US Sarbanes-Oxley Act (more commonly known as SOX; set new or enhanced standards for all US public company boards, management, and public accounting firms), to name just a few. One of the more recent regulations facing the business community is the Dodd-Frank Act, which brings significant changes to financial regulations in the US and is expected to have implications for data reporting that will require significant IT investment.

Compliance cannot be implemented and enacted solely by business or compliance experts or IT experts—rather, it involves an enterprise-wide scope. Regulations are typically specified in highly abstract legal writing that requires business or compliance experts to interpret and translate into concrete requirements. IT experts such as software architects and developers must work with business to ensure that their software and systems meet these requirements.
The process of implementing compliance measures must be documented and periodically reported to executive boards, auditors, and sometimes regulators. Unfortunately, however, each stakeholder group has different interests, knowledge, and expertise, and the work is often performed at very different abstraction levels.

This special issue of IEEE Software explores the challenges in developing compliant software systems. Typically, organizations face conflicting objectives, with compliance policies possibly hindering innovation, slowing down the product development process, or making the whole process most costly. The goal of software engineering for compliance is to bridge the gap between the software engineering community and the compliance community. The articles in this special issue explain the nature and extent of this domain from different viewpoints, the technical challenges it poses, novel software engineering methods for supporting compliance, and the current state of the art.

**Challenges for Achieving Compliance**

To be compliant means that an organization satisfies the requirements of the various regulations imposed on it. Lawmakers do not explain how exactly compliance is to be accomplished—rather, they just stipulate what is to be accomplished. The end result is that each organization must determine for itself what policies, procedures, and controls to implement to ensure compliance, from internal policies, standards, certifications, and licensing compliance to IT governance.

In many cases, compliance today is reached on a per-case basis—that is, many companies use ad hoc, hand-crafted solutions instead of clear software engineering and architecture concepts. This poses several challenges. In particular, the compliance solutions employed in many systems today are often

- hard to maintain because ad hoc, hand-crafted solutions don’t usually follow a clear architectural concept throughout the system architecture;
- hard to evolve or change because ad hoc, hand-crafted solutions usually lead to tangled code spread out over the system with many difficult-to-resolve dependencies on other components;
- hard to reuse because ad hoc, hand-crafted solutions often involve special-purpose code added into the systems at several places; and
- hard to understand because tangled code added in several places offers no adequate separation of concerns.

Furthermore, it is difficult for such solutions to guarantee compliance to a given set of rules and regulations or to keep up with constant changes in regulations and laws.

If we broaden the scope a bit, we see that many concerns in today’s software systems—some of which have a significant business value or impact—are very similar to compliance concerns stemming from regulations:

- **Internal business policies.** The business side might require specific internal policies to be fulfilled throughout an organization’s systems. For instance, a financial institution might set a policy that business transactions over a certain limit require segregation of duties (having more than one person required to complete a task; often called the four-eyes principle), so if one trader carries out a transaction, another trader approves it before it is processed.
- **Internal IT policies and good practices.** The software engineering field often follows its own IT policies and good practices, such as those found in coding and documentation guidelines, as well as in guidelines for change management, configuration management, and deployment management. Note that some of these kinds of guidelines are not only prescribed by internal policies but also appear in the IT standards that enable the implementation of certain regulations.
- **Intellectual property and licenses.** Many systems must respect individual licenses and terms of use for specific software, services, or content. For instance, a service could include both royalty-based operations and freely available operations, but only for noncommercial use.
- **Security policies.** A business might have specific security requirements that are pervasive throughout the whole system. Quite often, the business requirements for security overlap with regulatory requirements such as those in the payment card industry.

**Compliance cannot be implemented and enacted solely by business or compliance experts or IT experts.**
QoS policies. A business might require compliance of the runtime infrastructure to a certain quality of service (QoS), such as a specific performance window, a maximum latency, a particular mean downtime, or a certain throughput. This can be due to the necessity to fulfill contractual agreements such as service-level agreements (SLAs).

Clearly, important business requirements such as existing contracts or important internal business policies drive these compliance concerns as do regulations. However, most systems are ill-equipped to handle compliance requirements, whether they are driven internally through internal business policies or externally, through laws and contractual agreements.

Existing Software Engineering Solutions

Commercial governance, risk, and compliance tools offer some help in addressing software engineering challenges. These tools help define governance measures and controls, as well as the documentation and auditing required for reducing, mitigating, or eliminating the risk of violating obligatory regulations and policies. However, they offer little or no support for automating compliance fulfillment—that is, we can’t solve any of the software engineering and architecture challenges listed earlier entirely through such tools.

Researchers have proposed several software engineering solutions that go beyond tools. Stefan Sackmann and his colleagues categorize the existing solutions broadly into two main approaches: “compliance by design,” that is, implementation of compliance by designing it into a system, and “compliance by detection,” that is, implementing compliance by observing a system to ensure that its execution is compliant.

Many of the existing compliance solutions and research prototypes today address only one specific stage in the software development process and one specific kind of compliance artifact or dimension, from concrete case studies to tools-based software engineering solutions, covering a range of compliance concerns related to privacy, healthcare, business processes, and licenses.

In their article “Capturing Compliance Requirements: A Pattern-Based Approach,” Oktay Turetken and colleagues discuss an approach for capturing and managing regulatory compliance concerns and verifying business processes against them. The authors also present tool support and software engineering concepts for addressing regulatory compliance concerns.

In “Designing and Implementing a Hospital Quality Assurance Program,” Louise Reid and colleagues describe a concrete case from a live clinical environment, an area in which regulations and compliance play a very important role. Specifically, the authors describe the Hospital Quality Assurance Program (H-QAP) designed for establishing compliance with evidence-based best practice in the management of patients and software systems.

The article “A Framework for Managing Privacy-Enhancing Technology” by David Pelkola discusses the introduction of new technologies and how it has a major impact on the privacy practices within organizations. The author primarily proposes organizational measures to address the problem, hence this article shows a problem and solution from a typical viewpoint of the compliance community for a specific compliance concern: privacy.

Often, compliance is reached by mapping regulations or other compliance requirements to standards before they’re implemented in systems. In the article “Arguing Conformance,” Patrick Graydon and colleagues address the specific issue of how to argue about standards conformance. Although

However, most systems are ill-equipped to handle compliance requirements.
Compliance sources generally prescribe business practices for a wide range of compliance domains such as risk management, privacy, security, QoS, intellectual property, or licensing. No one-size-fits-all model can accommodate the divergence of compliance sources in an organization’s software and systems. Instead, compliance concerns today are implemented on a per-case basis using ad hoc, hard-coded solutions, which makes the resulting solutions hard to maintain, evolve, reuse, and understand. Ultimately, compliance should be defined internally and allowed to constantly evolve as business and technology drivers change. Our hope is that the articles we’ve selected for this special issue help further bridge the gap between the compliance and the software engineering communities by explaining compliance-related challenges and possible software engineering solutions for these challenges from different viewpoints.

References