OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Gemara

A Governance, Risk, and Compliance Engineering Model for Automated Risk Assessment

Table of Contents

Foreword

Integrating *Governance, Risk, and Compliance* (GRC) into software development pipelines presents a formidable challenge. Traditional, often manual, approaches to GRC are ill-suited for the pace of contemporary development. This has given rise to the discipline of GRC Engineering, which strategically applies engineering principles to GRC processes to make them more efficient and integrated. Its ultimate goal is to achieve automated governance, where compliance tracking is embedded throughout the deployment pipeline, acting as a required quality gate before code reaches production.

The need for a structured approach to this challenge was a key observation during the creation of the CNCF's Automated Governance Maturity Model (AGMM). The authors of that document observed that in a fully automated governance program, maturity can be measured in at least four different areas: "Policy, Evaluation, Enforcement, and Audit." This foundational insight provides a lexicon for describing the core activities of a secure software factory. The Gemara model took shape as industry projects began to apply this lexicon.

The FINOS *Common Cloud Controls* (CCC) and the *OpenSSF's Open Source Project Security* (OSPS) Baseline projects adopted the AGMM's language but identified the need for greater granularity. They added two more conceptual areas to distinguish between high-level, abstract recommendations and technology-specific objectives. This practical expansion from four concepts to six areas that build upon each other formed the genesis of Gemara's layered model.

The Gemara model was created to codify these concepts. As the *GRC Engineering Model for Automated Risk Assessment*, its core purpose is to provide a logical model to describe the categories of compliance activities, how they interact, and how to enable automated interoperability among them.

Introduction

This paper introduces the Gemara model, a structure designed to categorize compliance activities and define their functional interactions. Rather than describing organizational workflows or human processes, this model identifies the activities that are inherent to governance, along with their constituent elements and structural relationships. These activities have long existed in practice, but they lacked a unified engineering architecture with predictable points of exchange. By decomposing these activities into discrete layers, the model facilitates the standardization of documentation and language, and creates a basis for collaborative maintenance of common resources.

1. Scope

The purpose of this model is to provide a common basis for approaching activities and topics related to Governance, Risk, and Compliance (GRC), while leaving room for specific implementation details to grow and mature over time.

2. Definitions

There are multiple industry-accepted uses for many of the terms below. For the purposes of this document, we will use the terms according to these definitions. While we aim to have a single acceptable definition for most terms, this is not always possible and the reader may need to infer the instance based on context.

To assist the reader, defined terms are capitalized throughout the document. A term will be bolded on its first occurrence in the section where it is a primary focus to indicate its definition is available. Subsequent mentions remain capitalized for consistency.

- **Assessment** — (1) the process of determining whether an outcome meets the actor's intent; or (2) an atomic process within an Evaluation used to determine a resource's Compliance with an Assessment Requirement
- **Assessment Requirement** — a tightly scoped, verifiable condition that must be satisfied and confirmed by an evaluator
- **Audit** — a formal, opinionated review of an organization's Policies and posture, conducted at a specific point in time to verify that established requirements are met
- **Behavior Evaluation** — an opinionated observation of simulated or real-world activities
- **Capability** — a feature or function of a system; the primary component comprising an attack surface
- **Catalog** — a structured set of related prose and relevant metadata
- **Continuous Monitoring** — a multi-system process designed to collect Evaluation and operational data on an ongoing basis to better detect non-compliance and malicious action, enable Remediative Enforcement, and observe trends over time
- **Control** — (1) an organization's ability to fully assert desired state on a system, resource, or state; or (2) a mechanism, such as a safeguard or countermeasure, that asserts desired state; or (3) prose describing the Objective and Assessment Requirements associated with a desired state
- **Compliance** — adherence to a Rule or set of Rules
- **Evaluation** — the manual or automated process of forming an opinion on the state of Compliance, guided by a set of Assessment Requirements
- **Enforcement** — an action taken in response to non-compliance findings and their causes
- **Evaluation Finding** — the evidence and opinionated result of an Assessment

- **Guidance** — prose intended to help bring about a desired outcome for a topic or generalized scenario, based on knowledge of relevant Vectors
- **Guideline** — atomic element of a Guidance Catalog; often includes explanatory context and recommendations for designing optimal implementations
- **GRC** — (1) the Governance, Risk, and Compliance domain within the cybersecurity field; or (2) a coordinated program dedicated to these elements within a business unit
- **Governance** — strategic oversight of an organization and its activities
- **Intent Evaluation** — an Evaluation ensuring that a resource is prepared in alignment with Policy, such as through proper training, configuration, or code
- **Organization** — any logical grouping of human, physical, virtual, and information resources such as a company, business unit, or team
- **Threat** — a circumstance or event where the concepts of a vector are applied to a Capability in a specific context, resulting in the potential for negative impact
- **Objective** — a unified statement of intent, which may encompass multiple situationally applicable statements or requirements
- **Opinion** — a firmly held approximation of reality formed within the constraints of an evaluator's philosophy, perspective, and capabilities
- **Policy** — a clearly-scoped set of rules based on an organization's Risk Appetite
- **Preventive Enforcement** — any action that interrupts another process which would otherwise cause non-compliance
- **Remediative Enforcement** — corrective action in response to non-compliance in a deployed activity
- **Residual Risk** — the Risk remaining after Risk Mitigation and Enforcement actions have been implemented
- **Risk** — the potential for loss or damage when a Threat is actualized, determined by calculating the impact of an event to an organization and the likelihood of its occurrence
- **Risk Catalog** — a group of related Risks relevant to an organization; used to determine when and how Policies are created for the organization
- **Risk Appetite** — the level of Risk an organization is willing to accept in pursuit of its objectives
- **Risk Assessment** — the process of identifying the potential or actual Risks introduced by a system
- **Risk Mitigation** — the process of developing actions to prevent Threats or reduce their impact on organization objectives
- **Risk Acceptance** — a clearly documented decision to accept an unmitigated Risk as necessary or unavoidable

- **Rule** — an active, enforceable Policy, regulation, or law
- **Sensitive Activity** — a type of action that introduces Risk to an organization
- **Vector** — (1) an opportunity for an attacker to exploit a vulnerability in the system; or (2) a path by which neglect could result in unintentional negative outcomes
- **Vulnerability** — (1) a weakness in a system inherent in or associated with a Capability that can be exploited when used in unintended ways; or (2) a lack of Control or gap in defense, introduced intentionally or unintentionally, which can be leveraged to cause harm

3. Foundational Concepts

3.1. Prior Work

Much prior work exists to guide organizations in the adoption of Compliance Management Systems (CMS), such as I*SO 19600: Compliance management systems — Guidelines and ISO 37301: Compliance management systems — Requirements with Guidance for use.*

Other similar resources abound, such as the recent *Automated Governance Maturity Model* from the CNCF, which allows organizations to measure themselves and set growth goals for their GRC programs. In addition, much work has been done in the standardization space by the National Institute of Standards and Technology (NIST), a U.S. agency, on the **Open Security Controls Assessment Language** (OSCAL), which built upon the scope of automation protocols like **Security Content Automation Protocol** (SCAP).

Still, there is a deficiency in the consistent use of terminology, schemas, and processes — as well as the cross-team interoperability that could reasonably be expected when governing sensitive activities. These divisions have resulted in re-work both within organizations and across industries, with persistent gaps between Policy and reality.

Section 8.1 discusses how organizations might learn from this prior work in comprehensive automated solutions.

3.2. Assessing the State of Risk Assessments

While **Risk Assessment** may appear to be a straightforward topic at first glance, experience has shown that it takes many forms across a variety of scenarios. Some activities look forward, anticipating scenarios where Risk may occur. Other activities look backward, inspecting activities that are active or deployed to determine whether Risk has materialized and how to respond.

In some scenarios, Risk Assessment occurs even in the same lifecycle that a **Sensitive Activity** is being performed, though we might not directly refer to it as such.

These Sensitive Activities may take many forms, from analog scenarios such as the daily operations of a bank teller to the seemingly antithetical software development lifecycle (SDLC). Still, the categorical activities performed to Control outcomes and mitigate Risk are similar.

As we'll see in **Section 4**, Risk Assessments can be categorized into different types based on their inputs and outputs. These categories are cumulative, each calling upon other elements predictably, like layers building upon each other.

At the beginning of the logical flow, Risk is hypothetical and cannot be made concrete without information about the specific organization and activity being performed. These activities are often performed by consortiums or as a matter of course during the creation of a Policy — sometimes causing the processes to be stunted and the outputs to be created with lower quality or insufficient completeness. Because those elements form the foundation, an incomplete understanding can reduce the effectiveness of entire GRC and security programs.

3.3. Establishing a Layered Approach

The strategic value of adopting a layered architectural model for GRC lies in its ability to create a clear separation of concerns, breaking down a complex domain into manageable, interconnected components.

The Gemara model's structure was directly inspired by *ISO 7498: Information Technology — Open Systems Interconnection — Basic Reference Model*. Better known as the OSI Model, it is an authoritative and proven framework that describes distinct layers of network functionality.

By taking a similar approach to GRC, we enable different teams and tools to focus on specific activities, from high level Guidance to low level Enforcement, while ensuring they can interoperate within a cohesive system. Similar to the OSI Model, some tools may operate at multiple layers, and yet they still benefit from the clarity provided by describing their different activities according to the model.

Mirroring the layered approach, the Gemara model defines a contiguous seven-layer architecture. Within this structure, Layer 4 represents a pivot point: implementing Policies in a way that will later be evaluated. This layer captures the sensitive activities, such as software design or infrastructure provisioning that serve as a bridge where requirements and operational reality meet.

4. The Model

4.1. Introduction to the Model

A fundamental principle of this architecture is that each layer builds upon the one below it. Higher-level layers leverage the outputs and services of lower layers to perform their functions.

For instance, an Evaluation at Layer 5 is designed to test for conformance with a Policy defined at Layer 3, which is in turn informed by Controls at Layer 2. This hierarchical dependency creates a clear logical flow of information and authority through the Governance lifecycle.

This structure reverses the traditional document-centric view with Policy building upon Control. Since many organizations start with a Policy which includes Controls, the idea that Policy would follow Controls may seem counter-intuitive. However, this example is demonstrative of the model: A Policy is not ready for implementation and Evaluation until the Controls are written.

4.2. The Complete Model

The model is organized into two primary categories of activity: definitions and measurements. These categories will, ideally, each be represented by documents or logs pertaining to the corresponding action.

Activities in the *definition* layers, Layer 1 through Layer 3, should each produce document assets that may be referenced by higher layers or within their own layer. Activities in the *measurement* layers, Layer 5 through Layer 7, should each produce timestamped logs as outputs.

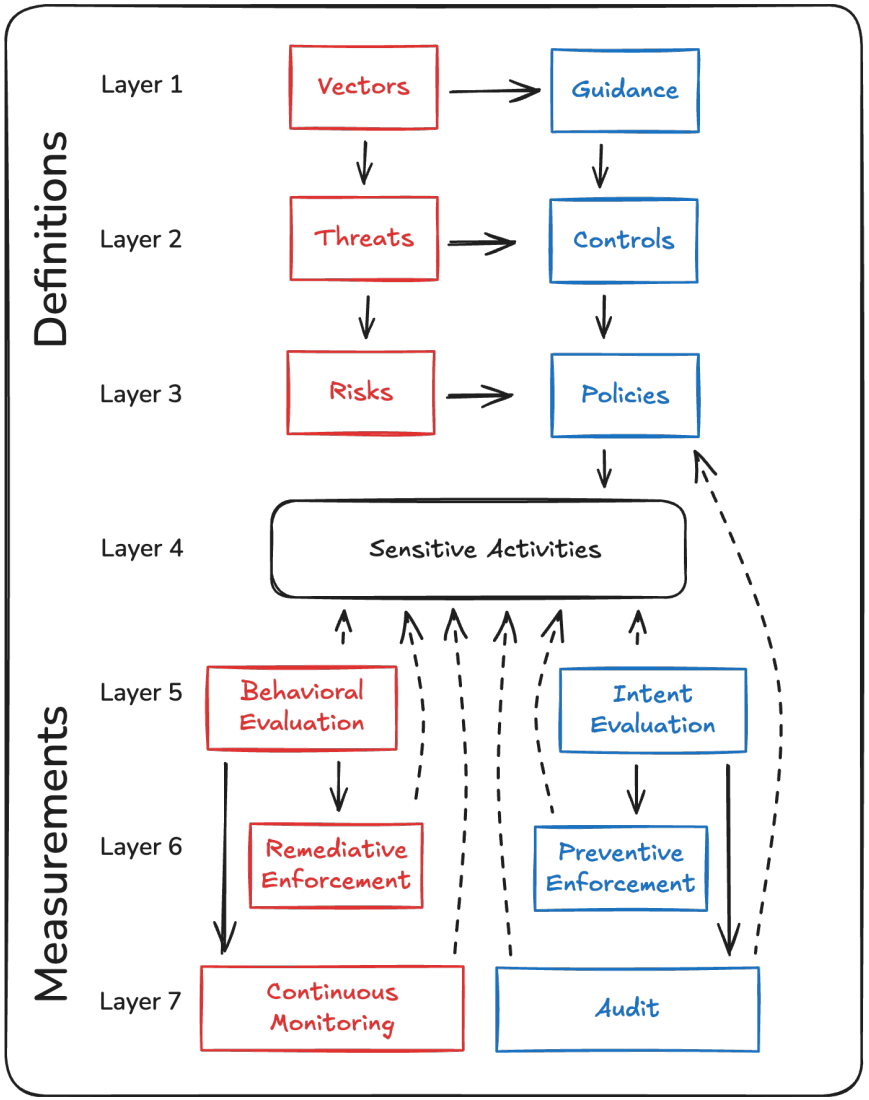
As noted in **Section 3.2**, Layer 4 defines Sensitive Activities connecting the two halves. The first three layers point toward the Sensitive Activities to define what is acceptable and what is unacceptable. Meanwhile, the last three layers look back at the Sensitive Activities or their outcomes to determine whether they comply with defined expectations. In Layer 7, Audit activities determine the quality of results from the lower layers.

FIGURE 1

The model's architectural components and their elements are detailed in the following sections, starting with The Definition Layers **(Section 5)**, followed by The Pivot Point **(Section 6)** examining sensitive activities, and finally The Measurement Layers **(Section 7)**.

FIGURE 2

MODEL RELATIONSHIPS AND THE LOGICAL FLOW



5. The Definition Layers

5.1. Introduction

Risk Assessment activities in Layer 1 through Layer 3 all work together to equip the organization for success when Sensitive Activities are performed.

Beginning with an understanding of the different ways negative outcomes or failures can occur, Layer 1 activities include the documentation of Vectors and the corresponding Guidance that can help prevent those negative outcomes.

Building upon Vectors, Layer 2 defines how Threats are narrow and specific to a particular scenario. Those Threats document the justification for Controls, which provide clear objectives and requirements to guide actors in the mitigation of Threats.

As the capstone to enable robust security implementation, Layer 3 prioritizes the Risks that an organization faces and outlines the Policies that are necessary to mitigate the most pressing opportunities for neglect, mistakes, and malicious activity.

When these three layers are coordinated in a streamlined fashion, they act as design requirements to accelerate and empower implementation activities.

However, the opposite is more often seen in reality: a failure to properly orchestrate definitions as part of the preparation and/or design requirements will inevitably result in Compliance being segmented from security. Instead of using Compliance activities as a strategic part of a larger initiative, Compliance itself becomes the goal.

As **Goodhart's law** teaches us: "When a measure becomes a target, it ceases to be a good measure." If we mandate Compliance for the sake of Compliance, we reduce our own efficacy. For this reason, a deep understanding of the reasoning behind these three layers is essential for ideal security outcomes.

5.2. Layer 1: Vectors & Guidance

The need for generic, high-level Risk Assessment is typically surfaced by factors or requirements far removed from the scope of the activity that is being assessed. Sometimes the need is made clear due to a Rule such as legislation. Other times, this type of activity is demanded as a precursor for Controls in a new technology category that has yet to be fully assessed — such as we have seen with the emergence of artificial intelligence.

When documenting a Vector, it is not necessary to understand the technological intricacies, such as the technologies involved at every step. Instead, the focus is on the opportunity for mistake or malice. These can be documented independently or within a catalog, and may similarly be published as standalone artifacts or alongside related Guidance. An example of Vectors can be found in the **MITRE ATT&CK** framework as *techniques*.

The constituent parts of a Guidance, referred to as Guidelines, do not typically stand on their own, and are most often published as a longstanding Guidance Catalog. Each Guideline often includes explanatory context and recommendations for designing optimal outcomes without foreknowledge of implementation details.

Guidance may be written internally for unique circumstances, but it is often developed by industry groups, government agencies, or international standards bodies. Examples include

the **OWASP Top 10**, **NIST Cybersecurity Framework**, **HIPAA**, **GDPR**, **CRA**, or any of the **PCI** and **ISO** standards.

As noted in **Figure 3**, Vector artifacts can be referenced by both Guidance and Threats to accelerate authoring and increase fidelity. Similarly, Guidance artifacts can be referenced by Controls to demonstrate how a particular Control applies the respective Guideline.

5.3. Layer 2: Threats & Controls

Building upon the high-level assessments that produce Vectors and Guidance, organizations and consortiums both frequently execute Risk Assessments that remain in the hypothetical realm — not yet considering a Risk Catalog, but instead focusing on technology-specific **Threat**-informed **Controls** with Objectives that are supported by clear **Assessment Requirements**. Similar to the activities described in Layer 1, these are hypothetical Risk Assessments which can be fully operationalized later, in the context of the organization's Risk Catalog.

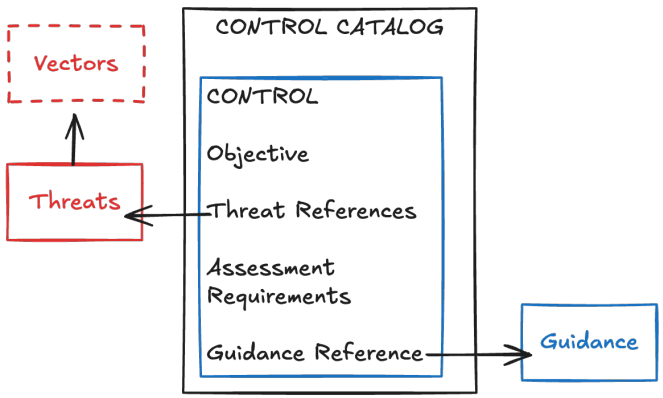
The frequent overloading of Threat and Control terminology in the information technology ecosystem has caused ambiguity. When generic and specific assets are produced, they are both frequently referred to as Controls, causing disparate outcomes and inconsistent quality in Policies. The definitions for Threats and Controls provided in **Section 2** are selected from the many different ways these terms are used.

Threats, ideally, will draw from known vectors and be mapped to a specific factor such as a **Capability**, location, or action, so that it is clear precisely when or where the Threat can be expected to manifest. They may be associated with a specific scenario, business unit, or technology, making them ideal for helping inform organizational Policies. In this way, Controls can be marked for exclusion if the corresponding factor is no longer present.

Controls should build upon established Guidance, and contain Assessment Requirements that can be directly implemented by evaluators. These are typically developed by an organization for its internal use, or for general purpose by industry groups, government agencies, or international standards bodies. Examples include **CIS Benchmarks**, **FINOS CCC**, and the **OSPS Baseline**.

While Controls are traditionally delivered in topical batches as Control Catalogs, there is an emerging trend toward composability, which defines reusable agnostic Controls for information systems. These are almost like Guidance in their generic prose, but with Assessment Requirements that are detailed enough to serve as Layer 2 assets. An example of this can be seen in the **FINOS Common Cloud Controls** (FINOS CCC) "Core Catalog," which contains both Threat and Control definitions that are later imported into technology-specific catalogs.

FIGURE 3



The recommended process to create a Control Catalog for an information system is to first assess the technology's capabilities, then identify Threats to those Capabilities, and finally develop a set of Controls to mitigate those Threats. However, it is not uncommon for an organization to write Controls first based on the author's experience and then later backfill Threats to solidify a justification or rationale for those Controls.

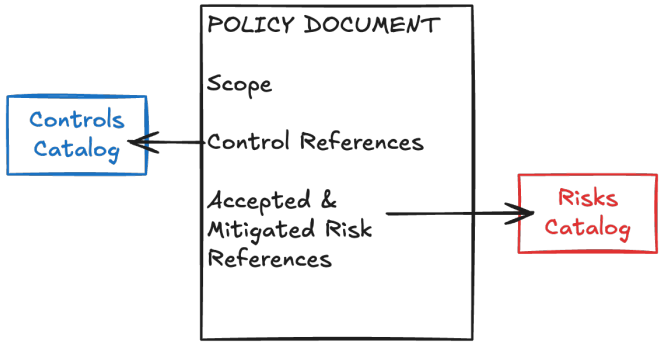
5.4. Layer 3: Risks & Policy

The respective authority of any Control or Guidance is strictly dependent on when and how it is referenced by an organization. This is the phase where Risk Assessments exit the conceptual phase and meet reality. When done well, this ensures that Controls are appropriately selected based on the most relevant details.

Precisely defining **Risk** requires a firm understanding of an organization's landscape: everything from technical to geopolitical details. It is typically informed by a Threat Assessment, to calculate the situational likelihood of a negative outcome, which is then compared to the impact it would have on the organization in the context of a particular Sensitive Activity. This may include elements such as the technologies used, the staff involved, and the value of the resources being handled.

Not all Risk is handled the same, however. Risk Appetite is the level of Risk an organization is willing to accept in pursuit of its objectives. Risk Appetite should be clearly represented in a Risk Catalog, which is used to determine when and how rules are created for the organization. A **Policy** is a clearly-scoped set of rules based on an organization's Risk Appetite. It provides Governance rules that, while based on best practices and industry standards, are tailored to an organization. Because Policies inevitably introduce some level of Risk Acceptance, they cannot be properly developed without consideration for organization-specific Risk Appetite.

FIGURE 4



A complete Policy document will be time-bound, contain references to Threat-informed Controls with Assessment Requirements, and include a clear plan for rolling out the Policy to impacted parties.

Policy documents may be referenced by other Policy documents, creating a functional inheritance model. These documents can be distributed to relevant parties as design requirements and used as a starting point for Layer 5 Assessments. They are significantly more likely to succeed in practice when Risks are included with the distribution of a Policy, where they are mapped to specific Threats and subsequent Controls.

If created during planning, a Policy document can serve as a functional design requirement to ensure that security is baked into the sensitive activity instead of becoming an obstacle later.

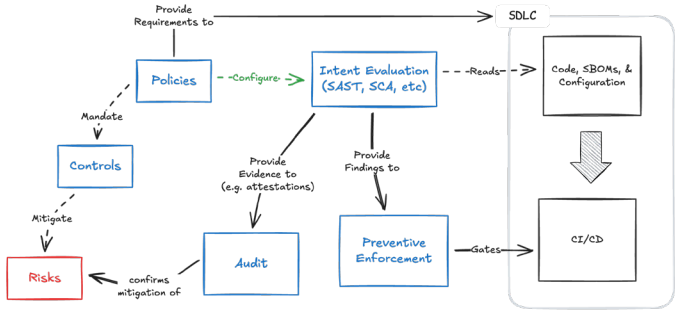
6. The Pivot Point: Sensitive Activities

Sensitive Activities are any type of action that might introduce Risk to the organization, creating the need for Governance. They are at the heart of the Gemara model, and the entire reason GRC has existed in myriad forms throughout history.

Sensitive Activities may take any number of forms: the building code regulating how a bank branch is built, the rules a teller needs to follow to stay within the law, the processes a development team must follow when creating a new mobile app, and the requirements placed on the software code itself.

FIGURE 5

APPLYING THE MODEL TO SUPPLY CHAIN SECURITY



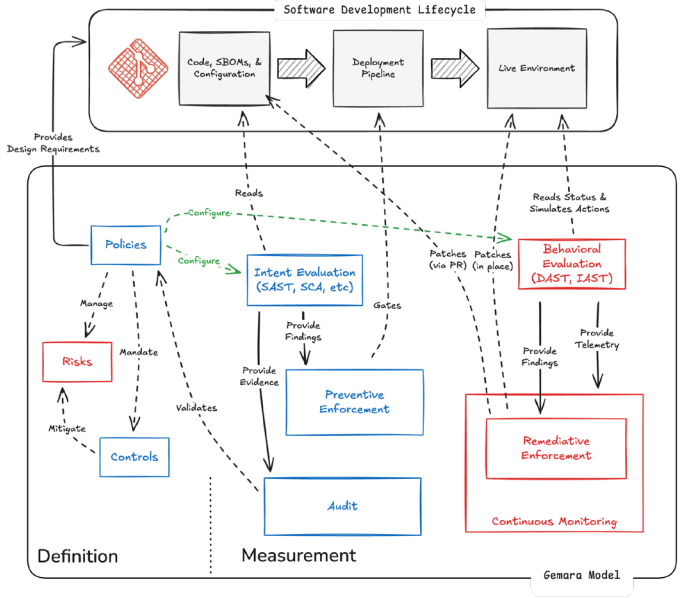
The model helps provide essential context to work that is already common. **Figure 5** uses the model to help demonstrate the different categories of work that are necessary to ensure a stable and secure software supply chain. Policies are used to inform both development workflows and Evaluation tools such as software composition analysis (SCA). For full maturity, the SCA Evaluation Findings should then be integrated into an Enforcement mechanism as well as Compliance Audits.

While some tools provide multiple elements of this process, leaders can reference the model to ensure that all the essential activities are being performed.

We see a fuller example in **Figure 6**, continuing the process into Evaluation of live runtime environments. When fully integrated, Evaluation Findings operate as the last stage of the software development lifecycle — similar to quality assurance, user acceptance testing, and product feedback.

FIGURE 6

APPLYING THE MODEL TO THE COMPLETE SOFTWARE DEVELOPMENT LIFECYCLE



7. The Measurement Layers

7.1. Introduction

Complementing the preparation done at lower layers to ensure sensitive activities are planned and executed securely, the final three layers look back on outcomes to ensure adherence to the organization's Policies.

Beginning with an inspection of the intended and actual outcomes, activities within Layer 5 can be described as either Intent Evaluations and Behavioral Evaluations.

Building on Evaluation Findings, Layer 6 describes Preventive Enforcement activities that serve as guardrails, blocking non-compliant designs before they go live, and Remediative Enforcement activities which produce corrections after negative outcomes are detected in a real world scenario.

Finally, Layer 7 describes activities which serve to Audit the effectiveness of the organization's Policies, Evaluation and Enforcement activities, and orchestrate Continuous Monitoring to ensure that sensitive activities remain compliant indefinitely.

An old leadership adage states that "a unit only does well that which its commander inspects well." Activities categorized within Layers 5, 6, and 7 act as that inspection which equips our organizations to excel.

7.2. Layer 5: Intent & Behavioral Evaluation

As the Risk Assessment activities in Layer 3 produce Policies, the activities in Layer 5 produce opinions through evaluation of Policy Compliance. These **Evaluations** consist of structured inspection in the form of either **Intent Evaluation** or **Behavior Evaluation**. These two types of Evaluation do not need to happen in parallel, but they are most informative for Enforcement decisions when the results are compiled.

Evaluations typically include multiple Assessments with independent failure modes. The best way to ensure compliance with organizational Policy — and assess unmitigated Risk — is to clearly document the relationship between each Assessment and an Assessment Requirement that is defined for a particular Control.

Examples of Intent Evaluation include both human-to-human interviews and automated or manual examinations of configurations. In information systems, many traditional Audit activities rely on manual Intent Evaluation such as dashboard screenshots, and modern trends are increasingly shifting toward automated configuration scanning and code analysis.

Analog Intent Evaluations may look at a team's procedures, training records, hardware, or facilities. A digital equivalent would include software composition analysis or cloud resource configuration scanning.

Behavior Evaluation may take the form of any action which observes or simulates user behavior to ensure that the expected outcomes are achieved. While simulation of bad behavior is useful for identifying security gaps or non-compliance with Policy, regular simulation of good behavior is also useful for ensuring that the system always operates as expected. In an analog environment this could manifest as secret shoppers, while evaluating an information system might include penetration testing.

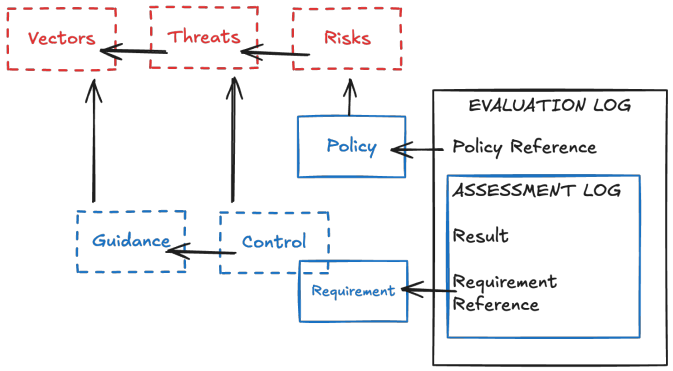
The **Opinions** of an evaluator are firmly held approximations of reality formed within the constraints of its philosophy, perspective, and capabilities (or those of its creator). This is a *post hoc* Risk Assessment that establishes an understanding of what Risks have been introduced or proposed.

The process involves comparing the Evaluation Findings to the organization's expectations, which are ideally captured in Policies through Controls and Assessment Requirements. While evaluators may be provided by vendors and industry groups alike, robust Evaluation should be informed by specific Policies in order to custom-tailor the Assessment to the needs of the Compliance program.

As seen in **Figure 7**, proper maintenance of relationships, also known as "mappings," between each artifact allows Evaluations to provide multiple opportunities to demonstrate a system's state of Compliance with various relevant artifacts.

FIGURE 7

MAPPING EVALUATIONS LOGS TO DEFINITION ARTIFACTS



7.3. Layer 6: Preventive & Remediative Enforcement

When an Evaluation logs a non-compliance finding, the next step is to respond with corrective action. This is often done within the same tool or process that identified the Evaluation Finding, which will then alert, interrupt, or remediate. The latter two are considered Enforcement actions, while ongoing activities such as alerting are part of Continuous Monitoring.

While this activity is often viewed as purely objective, we highlight this as a continuation of Risk Assessment because the enforcement layer involves forming an Opinion on the best course of action to take in the event of non-compliance findings. Choosing to prevent or remediate, as well as when and how to do each, requires an understanding of the Risk associated with each possible course of action.

Preventive Enforcement involves any action that interrupts a process to prevent non-compliance. This is primarily in response to malformed intent, such as a bad configuration, but it may also include more complex behavioral Evaluation processes that are executed prior to the deployment of the sensitive activity.

In software development lifecycles, this may take the form of a deployment gate, equivalent to a perimeter gate on a physical property that controls admission. This ensures that all actions that are intentionally taken are vetted according to Policy.

Remediative Enforcement describes corrective action in response to non-compliance in a deployed activity. This may include isolating, replacing, retraining, or re-deploying the noncompliant resource with a compliant configuration. This typically requires some manner of observability to detect target state, and communication with the latest applicable Policies to ensure that any changes to the environment — or policy-aligned remediation guided by defined operational thresholds.

Due to the volatile nature of many remediation activities, and the potential for end-user confusion, proper logging and alerting should always be put in place alongside remediation tools.

Extending the point made in **Figure 7**, Enforcement activities may find heightened success when properly mapped to Evaluation results. This allows every person or system involved in the change to understand the complete justification, instead of the Enforcement being applied arbitrarily.

7.4. Layer 7: Audit & Continuous Monitoring

Perhaps the most famous and most feared part of Compliance is the process of demonstrating it to those who weren't involved with the sensitive activity.

An **Audit** is a formal, opinionated review of an organization's Policies and posture, assessing the quality of past Risk Assessments as well as **Residual Risk**.

Because an Audit is an approximation of Compliance posture at a point in time, its outcome is a firm assertion based on observations of available evidence or gathered facts. Audits may have a variety of scopes, and may involve forming opinions on the Guidance an organization follows, the Controls it writes, the Policies it implements, its Evaluation methods, and its Enforcement status. The opinion is typically informed by a combination of geopolitical regulatory requirements and known best practices for the industry and resources involved.

Audits are typically conducted in batches, and look for point-in-time evidence collected during an Evaluation activity. Traditionally, Audits have favored manual Intent Evaluations due to their simplicity and predictability, but cybersecurity Audits are increasingly integrating automated tools for both intent and behavioral Evaluation.

While Audit activities often focus on historical inspection, **Continuous Monitoring** aggregates results from Evaluations and operational metrics in real-time. This process identifies Threats and Control failures as they occur to support Remediative Enforcement. This is also known as "CCM," continuous Compliance monitoring.

Mature CCM operations establish a persistent, Policy-driven process that harnesses multiple systems to ensure maximum visibility of all deployed assets at all times. A physical equivalent might be constant security patrols, live video surveillance, and active perimeter guards.

8. The Need for Machine-Optimized Documentation Standards

8.1. Going Beyond Machine-Readable

As mentioned in **Section 3.1**, the *GRC Engineering Model for Automated Risk Assessment* is the product of many previous iterations and learnings.

Immense value has come from the work on OSCAL, where a large body of machine-readable documents have been crafted both publicly and privately by highly regulated organizations. The work done in the OSCAL space cannot be overlooked or discounted, as it has provided a great benefit to countless firms.

However, there is something to be said about the benefits of optimizing for machines when crafting machine-readable GRC documents. Previous efforts to craft relevant schemas have fallen short in two key areas.

The first, and most obvious, shortcoming comes from tools that misappropriate the term "Policy" to describe security tooling configurations. As useful as these resources may be, far too many organizations waste the potential of highly-trained GRC and security professionals by making them responsible for managing tooling configurations under the pretense of writing Policy-as-Code.

The second shortcoming is more difficult to spot, and it comes in the form of a specification which allows for customized fields.

Although it may sound innocent or even necessary, the addition of customization of GRC document schemas results in the need for custom-tailored tools that have foreknowledge of those fields. The tradeoff for this flexibility is a loss of structural clarity needed to build widely adopted tooling around the data. While a schema should be flexible enough to support a variety of uses, it should also be opinionated enough to allow an automation ecosystem to rise up and thrive around it.

By drawing on the positive lessons and addressing known issues, we can enable organizations to engineer highly efficient GRC ecosystems that are robust enough to address complex problems, but comprehensible enough for any professional to quickly get up to speed, all while maintaining the ability to map to other formats, such as OSCAL.

Achieving an opinionated, standardized schema for each activity type will allow for rapid industry-wide acceleration of automated Risk Assessments.

8.2. Improvements for Humans; Improvements for AI

An organization that adopts an optimized GRC Engineering strategy can gain benefits for both humans and artificial intelligence.

Communication and handoffs are a well-known obstacle for highly regulated firms. Field observations consistently reveal challenges distributing Policies to impacted parties, managing changes, accidental rework, and even months spent on unnecessary work due to misunderstandings.

Firms that standardize document schemas can develop the necessary tools to get documents to the right place at the right time. The database and APIs necessary to build these communication optimizations provide a secondary benefit: they act as the foundation for Model Context Protocol (MCP).

When a properly designed system such as MCP or AI-skills is overlaid on a machine-optimized GRC database, AI capabilities benefit from replacing vague, unstructured prose with unique technical identifiers and explicit resource mappings. This structural clarity allows the model to act upon requirements, minimizing the need for probabilistic inference of intent and leading to more deterministic outcomes.

When partnered with an Agent2Agent (A2A) enabled system that allows GRC specialized systems to provide precisely contextualized information to engineering specialized systems in a near deterministic manner, it has been observed that AI systems will require dramatically fewer resources, security tools increase in accuracy, less time is spent on false positives or negatives, and more time is spent delivering real value to the organization.

9. Conclusion

This model provides a clear, actionable, and extensible framework for engineering modern Governance, Risk, and Compliance programs. By establishing a common vocabulary and a logical, layered architecture, it demystifies the complex interactions between high-level Guidance, technical Controls, organizational Policy, and automated Enforcement.

The Gemara Project has been formed within the Linux Foundation and is stewarded by the Open Source Security Foundation (OpenSSF). The project is responsible for maintaining machine-optimized document schemas for all of the activity types listed in this model, as well as a lexicon of helpful terms and software development kits (SDKs). The project's governance structure ensures that maintenance is fully distributed among multiple organizations, with founding maintainers from Sonatype, Red Hat, and CVS Health.

The community-maintained schemas are open to extension or, in extreme cases, modification to best support the community at large. The similarly open source SDKs are designed to assist in reading and transforming Gemara-compatible documents. These resources ensure that any organization can rapidly bootstrap an internal GRC Engineering solution, and anyone looking to build solutions to accelerate the ecosystem can get started in minutes.

The Gemara Project offers a structured path toward Automated Governance by bridging the divide between high-level industry frameworks and technology-specific safeguards. By providing a common logical basis for GRC Engineering, the project moves beyond abstract concepts to offer a practical, machine-optimized foundation for the modern secure software factory.

10. Authors & Acknowledgments

As mentioned throughout this document, the text herein draws from a myriad of lessons and sources in a way that defies any attempt to establish holistic attribution. Below is a list of authors who compiled the learnings into this model and key contributors who influenced and provided guidance to influence its final shape.

Authors: Eddie Knight, Jennifer Power

Contributors: Hannah Braswell, Travis Truman, Brandt Keller, Damien Burks, Sonali Mendis, Michael Lysaght, Robert Griffiths, Rob Moffat, Naseer Mohammad, Stevie Shellis, Jared Lambert, Andrew Martin, Sonu Preetam, Stephanie Harris, Marcus Burghardt, Sally Cooper, Manju Mayachar

Notable Influences: Ian Miell, Michaela Iorga, Tara Houlden, Ben Cotton, Christopher Robinson, Evan Anderson, Maxime Coquerel, David Wheeler, Kamran Kazmi, Ian Tivey, Dana Wang, Jeff Diecks

OpenSSF

OPEN SOURCE SECURITY FOUNDATION

Thank you! Join us.

gemara.openssf.org

openssf.org