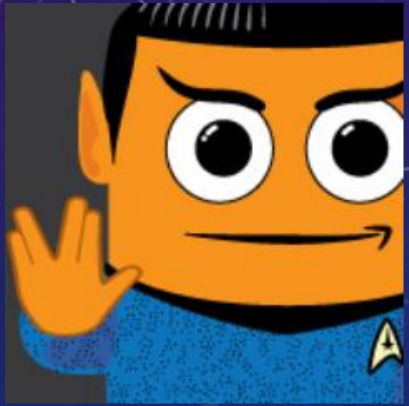
# TAMING THE SBOM CHAOS

Using AI Agents for Audit SBOMs for OSS Compliance

By Oscar Valenzuela B.

## ABOUT ME

- Background in Software Engineering & Electronic Engineering.
- 10 years experience in Telecommunications.
- 20+ Years Experience in Open Source, Compliance Tooling, Community Building.
- Active in Yocto Project, OpenChain, Software Heritage, FSF LA, Xpertians.
- **Principal Open Source Engineer @ Amazon**
  - Leading strategy for the Open Source License Compliance team.
  - Driving initiatives to scale compliance & adopt best practices/standards.
  - Led numerous M&A technical due diligence & compliance projects (from spaceships to IoT).
- **Fun Fact:** Also known as Andrew/Andres!



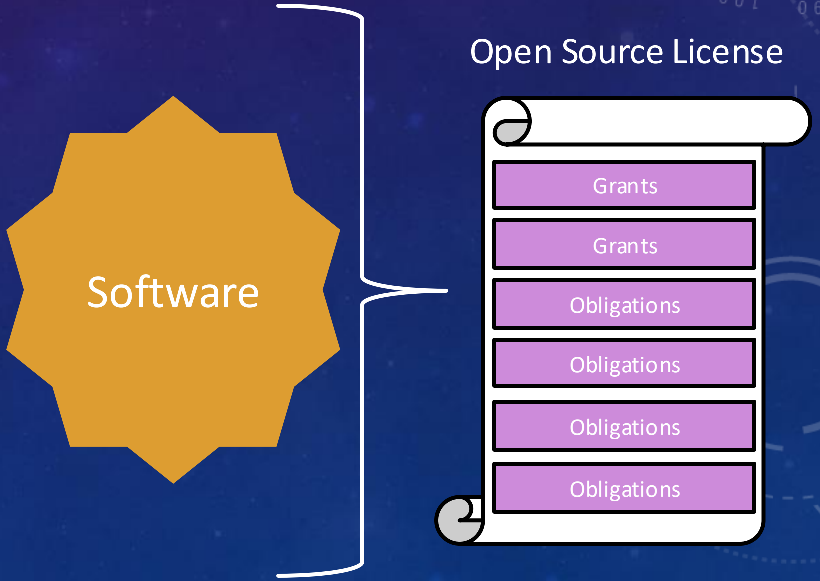
# OPEN SOURCE LICENSE COMPLIANCE

## UNDERSTANDING THE PROBLEM

## LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

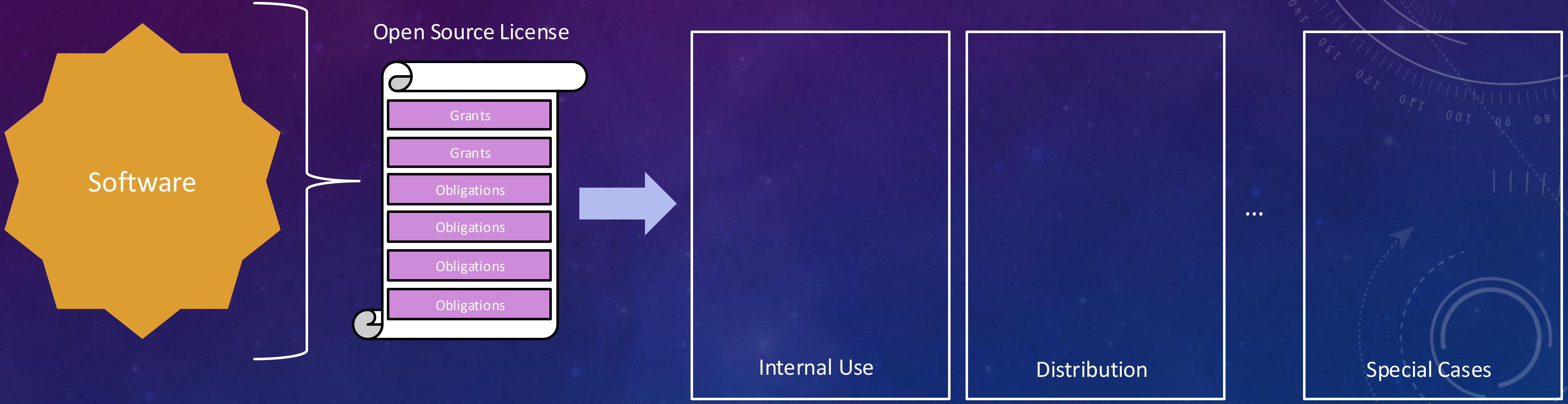
Open Source Licenses are static documents, but conditions are dynamic and depend on the specific use case.

- Factors such as technical implementation, modifications, customer interaction, and internal use versus externalization change the parameters for the risk analysis.
- Dynamic conditions also create challenges in scaling analysis capabilities and managing use case complexity.



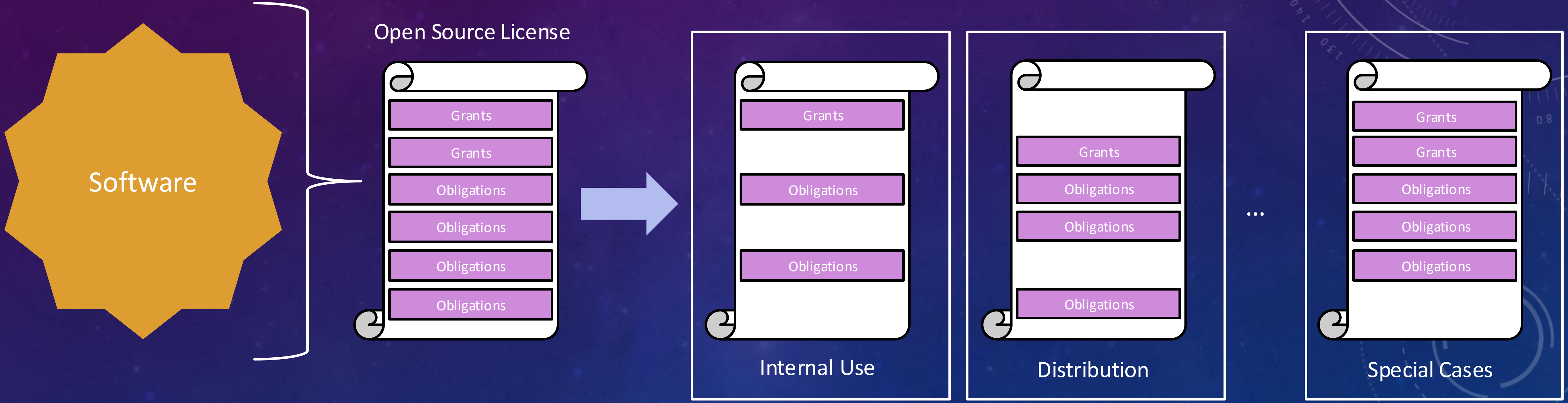
# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

## Problem 1: Use Cases can trigger different License Obligations



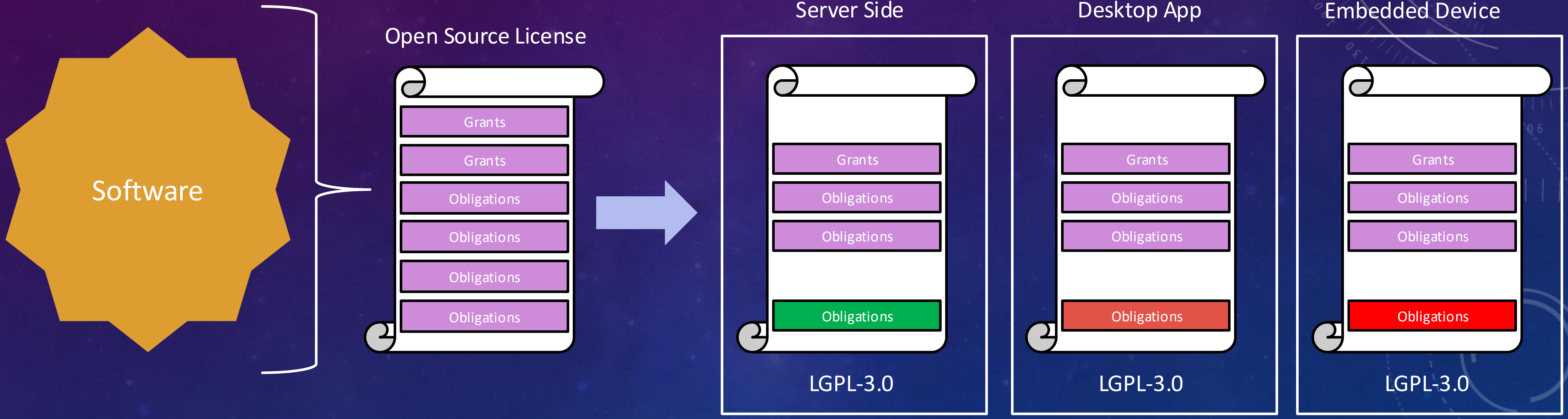
# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

## Problem 1: Use Cases can trigger different License Obligations



# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

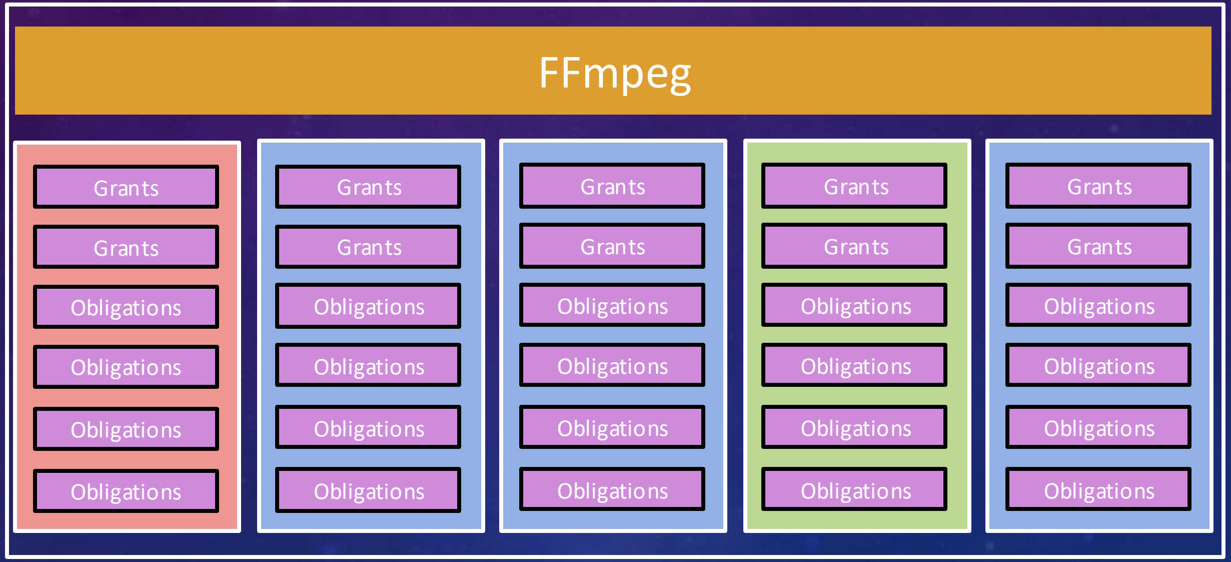
## Problem 1: Use Cases can trigger different License Obligations



# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

## Problem 2: Technical Implementation Affects License Applicability

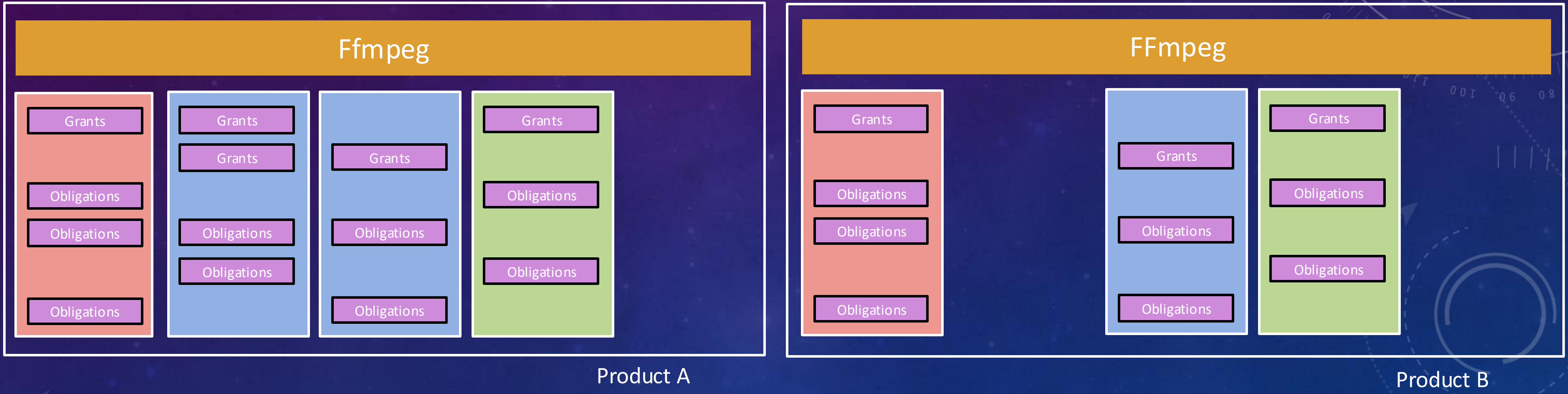
Build configuration can enable/disable Licenses



# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

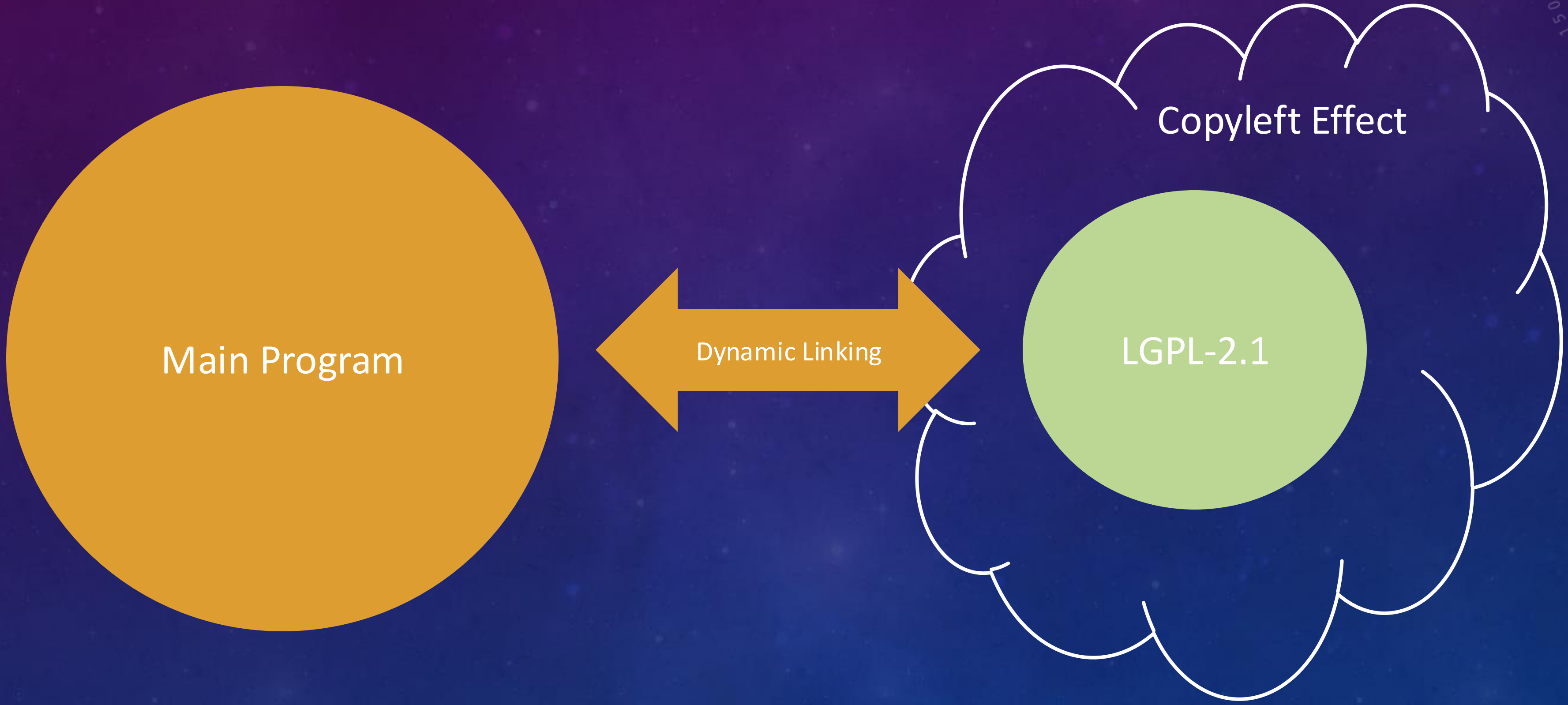
## Problem 2: Technical Implementation Affects License Applicability

Build configuration can enable/disable Licenses



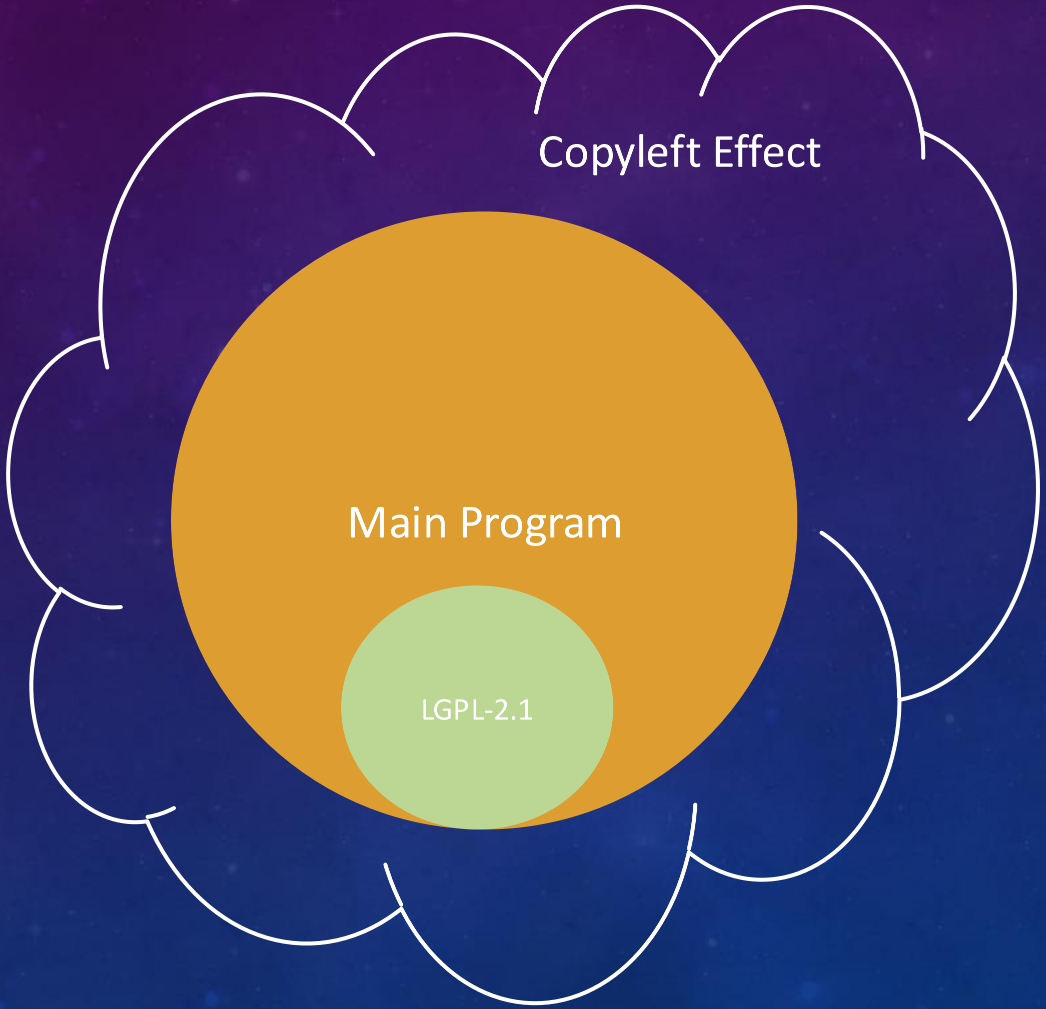
## LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

Problem 3: Technical Implementation affects the Scope of the License



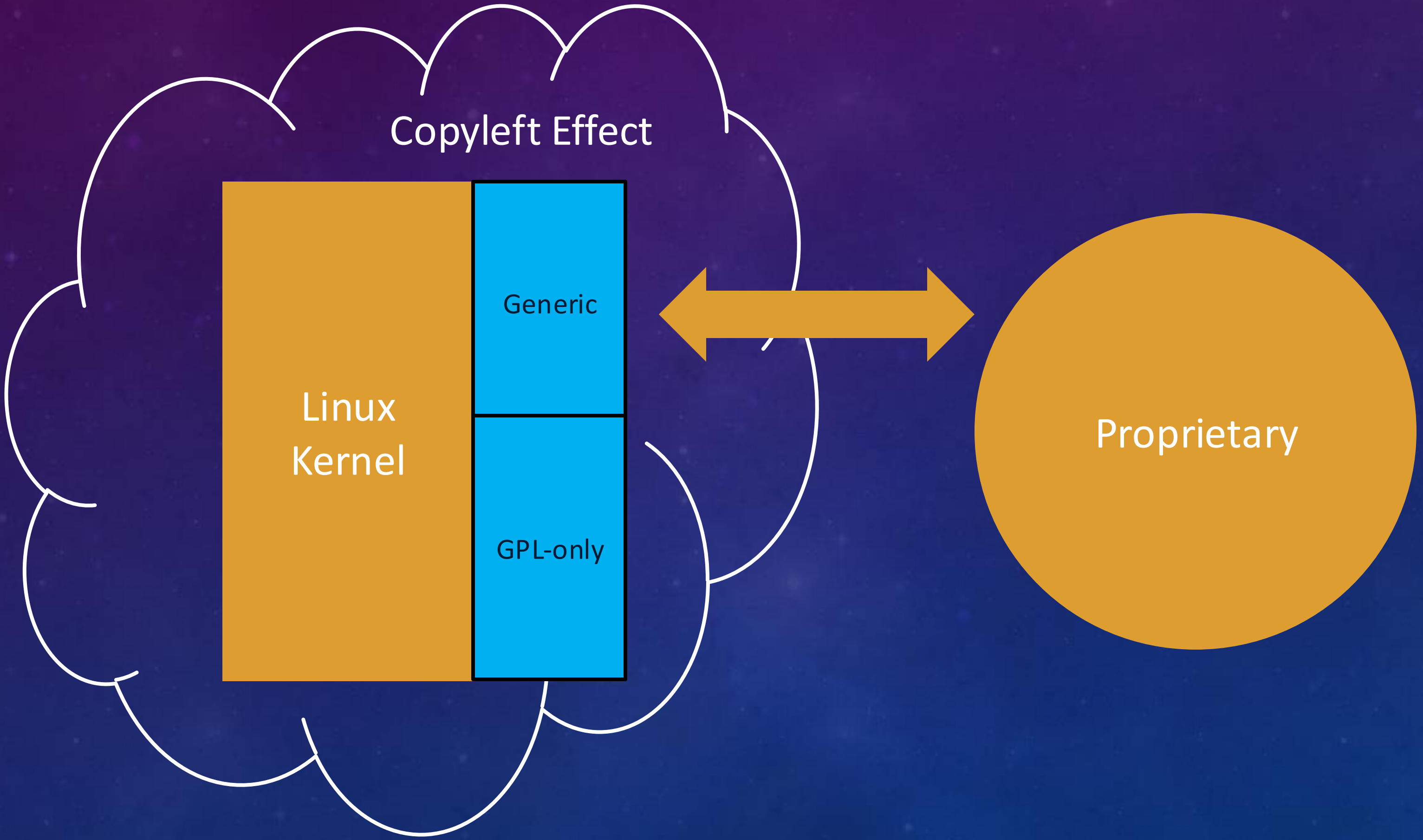
# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

## Problem 3: Technical Implementation affects the Scope of the License



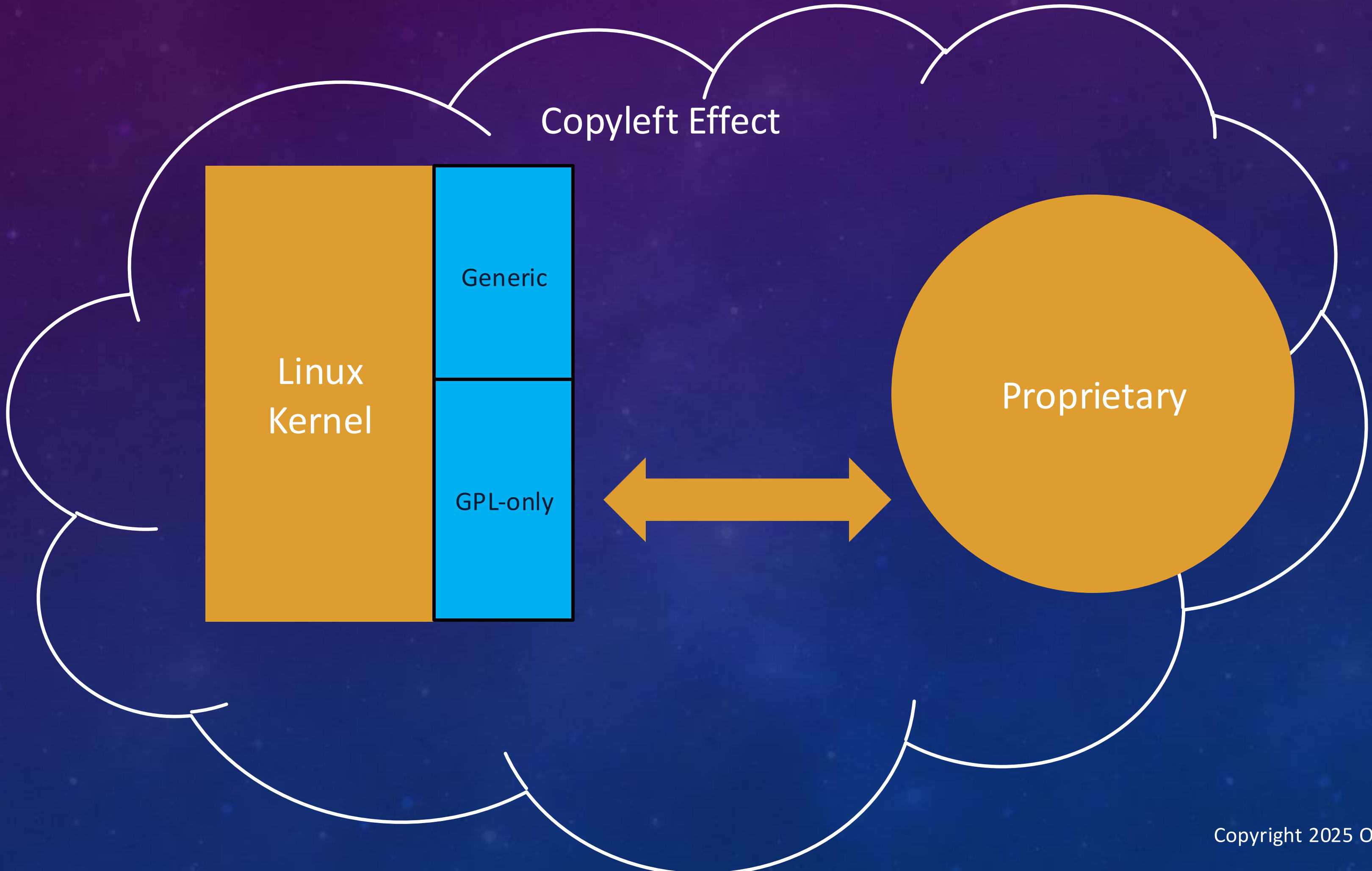
## LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

Problem 3: Technical Implementation affects the Scope of the License



## LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

### Problem 3: Technical Implementation affects the Scope of the License



# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

## Problem 4: Programming Language affects the Scope of the License

Risk Analysis: $\underbrace{\text{Python Package}}_{\text{Domain}}$ under $\underbrace{\text{LGPL-3.0}}_{\text{License}}$

| | Ingestion | Consumption | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Internal Use** | | **Distribution (Externalization)** | | | |
| Obligations | Ingestion (Import) | Dev/Testing tools | Server Side | Linux OS | Customer Desktop App | Customer Mobile App | Consumer Devices |
| Source Compliance | No | No | No | Yes | Yes | Yes | Yes |
| Legal Notices | No | No | No | Yes | Yes | Yes | Yes |
| Copyleft Risks | No | No | No | No | No | Maybe | Yes |
| Patent Risks | No | No | No | No | No | Maybe | Yes |
| Void DRM Protection | No | No | No | No | No | Maybe | Yes |

Example Risk Analysis

# LICENSE COMPLIANCE: BEYOND STATIC RISK ANALYSIS

## Problem 4: Programming Language affects the Scope of the License

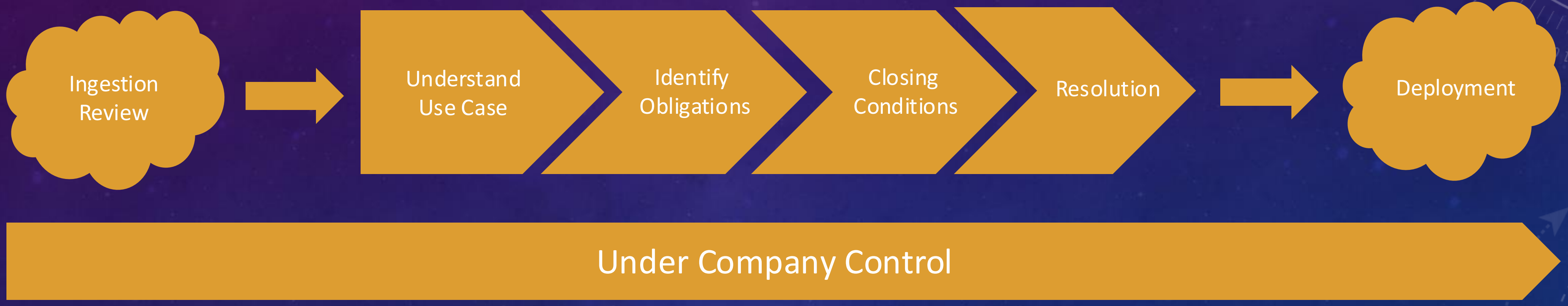
$\text{Risk Analysis: }\underbrace{\text{Rust Package under}}_{\text{Domain}}\ \underbrace{\text{LGPL-3.0}}_{\text{License}}$

| | Ingestion | Consumption | | | | | |
|---|---|---|---|---|---|---|---|
| | | Internal Use | | Distribution (Externalization) | | | |
| Obligations | Ingestion (Import) | Dev/Testing tools | Server Side | Linux OS | Customer Desktop App | Customer Mobile App | Consumer Devices |
| Source Compliance | No | No | No | Yes | Yes | Yes | Yes |
| Legal Notices | No | No | No | Yes | Yes | Yes | Yes |
| Copyleft Risks | Maybe | Maybe | No | Maybe | Yes | Yes | Yes |
| Patent Risks | Maybe | Maybe | No | Maybe | Yes | Yes | Yes |
| Void DRM Protection | No | No | No | No | No | Maybe | Yes |

Example Risk Analysis

## LICENSE COMPLIANCE: INTERNAL VS. DISTRIBUTION

Problem 5: Responsibility for OSS Obligations Shifts

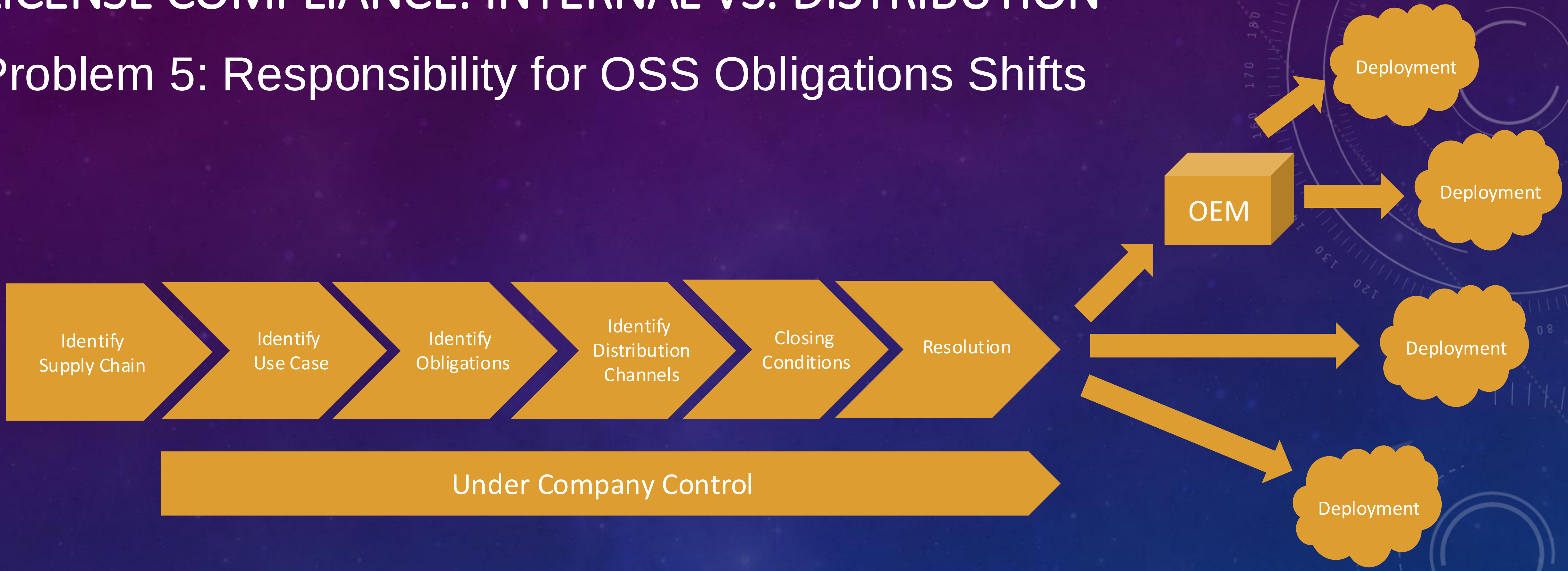


Risk Management:

- Package Curation (scanned pre-import).
- Source Code available in the Code Vault.

# LICENSE COMPLIANCE: INTERNAL VS. DISTRIBUTION

## Problem 5: Responsibility for OSS Obligations Shifts

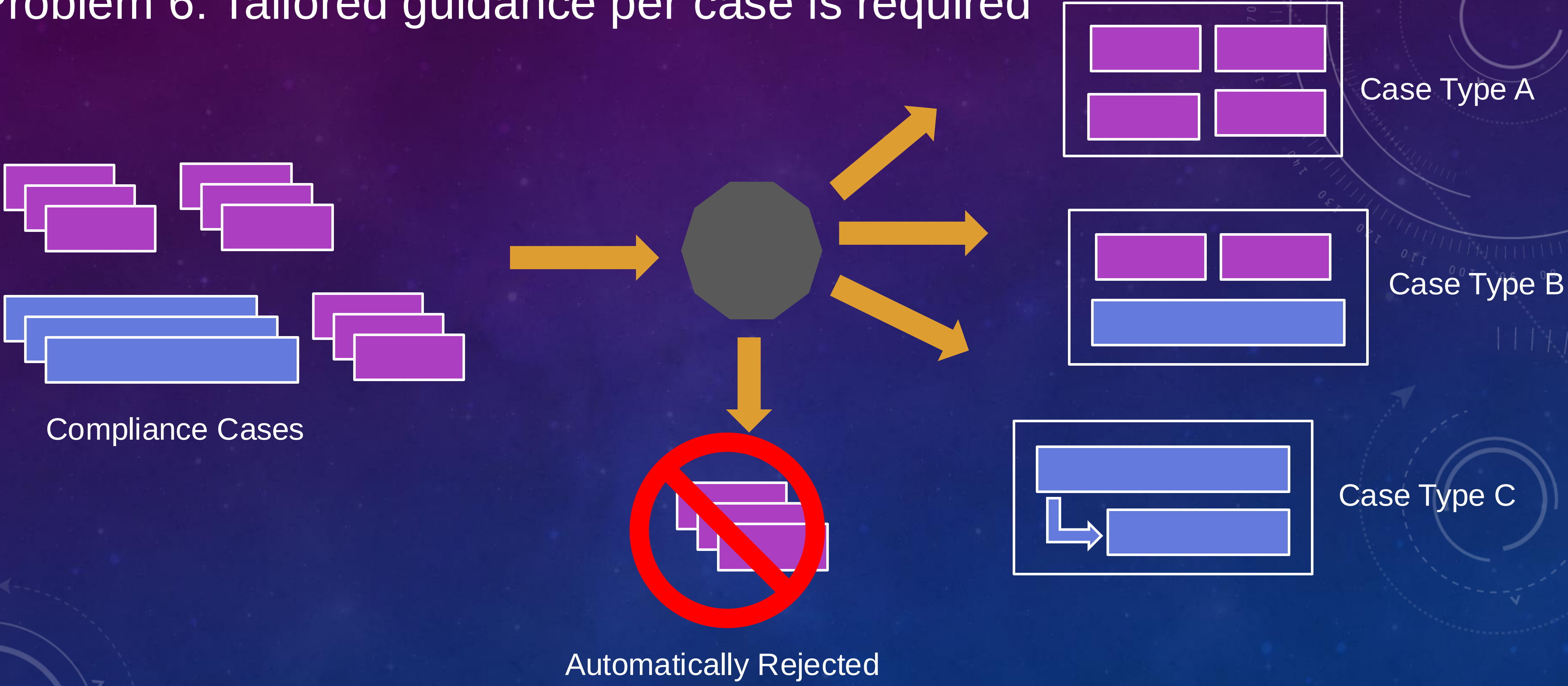


Risks Management:

- A defined package curation process during ingestion is absent
- Source Code is not always available as software is built by partners.
- The software is used by Company, OEM or partners.

## NUANCE AND INTERPRETATION ARE REQUIRED

Problem 6: Tailored guidance per case is required



## SOFTWARE COMPOSITION ANALYSIS

TOOLS ARE NOT MAGIC WANDS

## TOOLS CANNOT ADDRESS ALL SCENARIOS

Tools expect binary input, but OSS compliance is rarely black and white:

- Nuance and interpretation are frequently required.
- Tooling “over-trust” means accepting an Opinionated Output.
- Tools struggle with complex technical scenarios.
- Commercial tools with proprietary/locked Compliance Databases prevent contributions or corrections, acting as a Black Box.
- Open Source Tools tend to suffer from Swiss Army Knife Syndrome.

## SOFTWARE COMPOSITION ANALYSIS

Dynamic vs. Static Analysis can produce different results.

Static Software Composition Analysis (Static SCA)

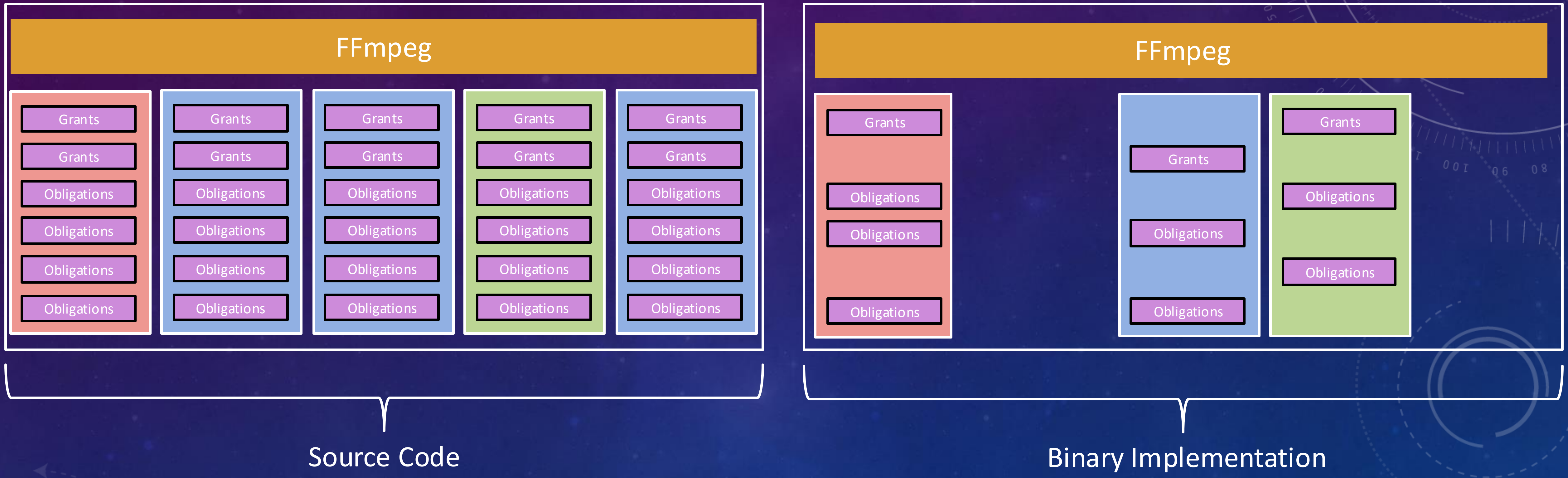
- Analyzes the software's source code, binaries, or other static artifacts (e.g., package manifests, dependency files) without executing the program.
- It cannot detect vulnerabilities or dependencies that are dynamically loaded or resolved at runtime.

Dynamic Software Composition Analysis (Dynamic SCA)

- Analyzes the software during execution, tracking the actual dependencies and components being loaded, called, or used at runtime.
- Requires a working runtime environment, which may not always reflect real-world usage or edge cases.

## SOFTWARE COMPOSITION ANALYSIS

Dynamic vs. Static Analysis can produce different results.



## SOFTWARE COMPOSITION ANALYSIS

### Binary Static Analysis – Methodology Matters for results quality.

| Methodology | Details |
|---|---|
| Fuzzy Hashing | Use a preexistent dataset of hashes to determine proximity between binary artifacts. |
| Symbols Extraction | Utilize symbols like literal strings, export symbols, and other signatures to identify binaries. |
| Package Metadata | Extract package identification evidence from package metadata or other official sources. |
| Compression redundancy | Identify overlaps (code cloning) when compressing two software artifacts. |
| Binary Deltas | Identify diff between two artifacts, inferring identification when delta is close to 0. |

Most SCA tools implement Package Metadata and Symbol Extraction. However, their output typically requires manual confirmation. This is because package metadata only works when packages embed that information, and symbols are only present if debug information is available and hasn't been scrambled for security reasons.

## AGENTIC AI

A TOOL THAT LEARNS TO USE TOOLS

## THE “AGENTIC APPROACH”

### Let’s define the problem and build a goal

- Open Source License Compliance is dynamically complex, requires technical experience (learn technology) and understand the use case in detail.
- The traditional approach is always difficult to scale and require tailored guidance.
- Tools are not an automatic solution, not all tools produce the same data (format and quality), most Knowledge DB are opinionated, which doesn’t mean they are correct.

## THE “AGENTIC APPROACH”

Let’s define the problem and build a goal

- We can teach AI to do the heavy lifting of OSLC (code scanning, data enrichment, verify external sources, identify license texts, compile risk analysis reports, generate compliance artifacts, etc.)
- We can use the information generated to build a conclusion.
- AI will not replace the need for an Open Source Compliance Engineer to make decisions, but it will help handle much of the work—potentially most of it, if the tasks are limited to scanning and data gathering.

## AGENTIC AI

THE BASICS

## WHAT’S THE DIFFERENCE BETWEEN LLM AND AGENTIC AI?

- Provides Knowledge
- Understand & Generate Language
- Answering questions, summarizing, content creation

- Applies Knowledge to Act
- Act Autonomously & Achieve Goals
- Planning, using tools (incl. LLMs), learning, adapting

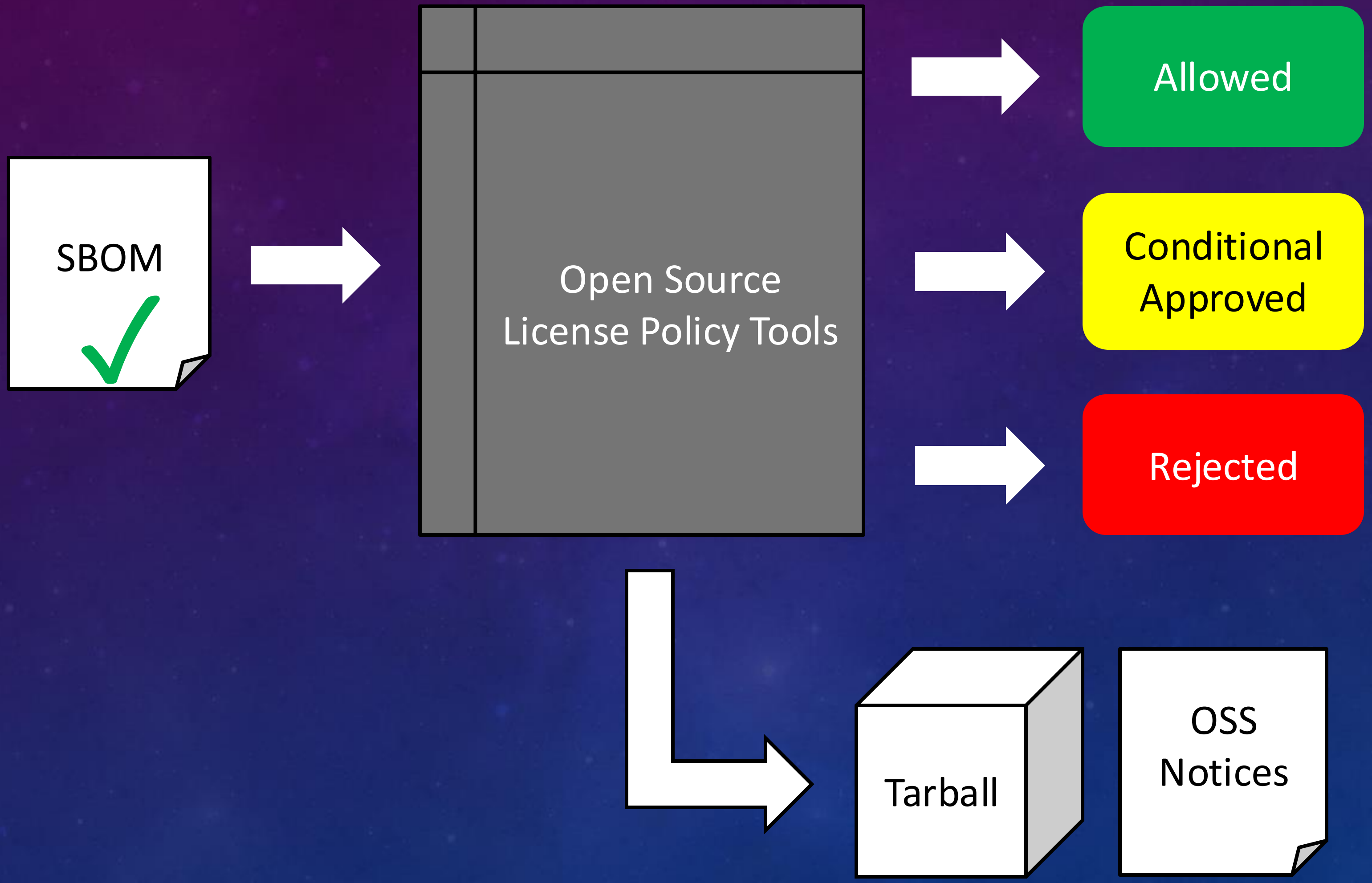
# TYPES OF AGENTIC AI

- **Simple Reflex Agents:** These agents react to specific inputs with predefined actions, lacking the ability to learn or adapt to new situations.
- **Goal-Based Agents:** These agents are equipped with specific goals and use strategies or plans to achieve desired outcomes, employing search and planning algorithms to determine the best course of action.
- **Utility-Based Agents:** These agents use complex reasoning algorithms to help users maximize desired outcomes by comparing different scenarios and their respective utility values, choosing the option that provides the most rewards.
- **Learning Agents:** These agents can improve their behavior over time by interacting with their environment and learning from their experiences, using various learning mechanisms to optimize their performance.
- **Hierarchical Agents:** These agents engage and coordinate multiple agents to pursue a common goal, useful in fail-safe environments or regulated industries where oversight of workflows is crucial.
- **Multi-agent systems:** These systems are designed to act as autonomous agents capable of performing complex tasks, making decisions, and interacting with their environments independently

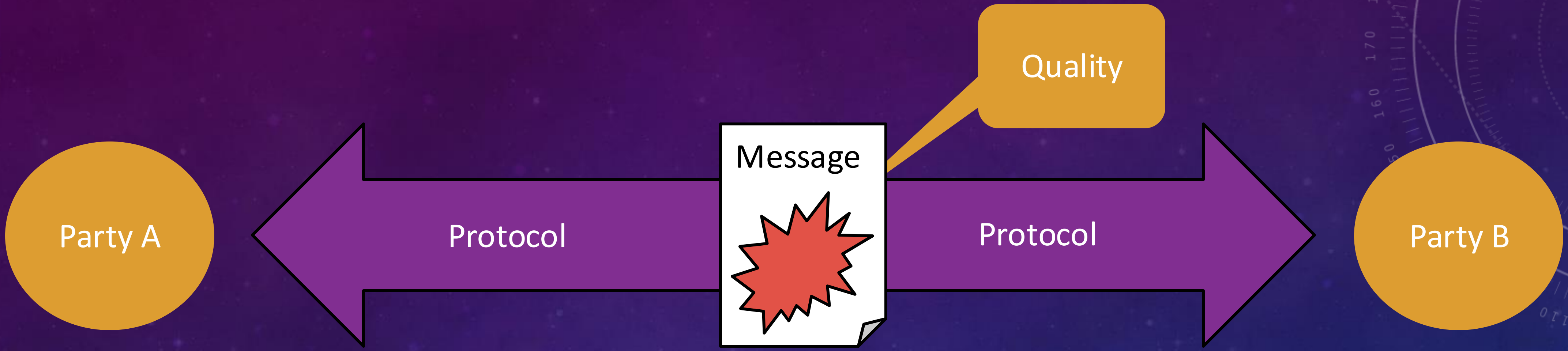
## AGENTIC AI

RESOLVING A PRACTICAL PROBLEM: SBOMS

## SBOMS IN AN IDEAL WORLD



## COMMUNICATION PROTOCOL VS. MESSAGE QUALITY



SBOMs define the structure, content, and interpretation between parties.

SBOMs facilitate data sharing, they do not guarantee the quality or accuracy of the information conveyed.

Quality of the SBOM depends on the capacity of the tool we choose.

## PRACTICAL EXAMPLE: USING AGENTIC AI FOR SBOM ANALYSIS

Problem: SBOM Analysis for OSS Compliance is practically impossible to automate.

Reasons:

- File Format:
  - Multiple formats (SPDX and CycloneDX) and versions, which tooling doesn't fully implement.
  - Commercial tools don’t produce compatible SBOMs (valid JSON schema files, but the information is incompatible).

- Lack of Data:
  - No assertions. No hashes, incorrect hashes, or manually edited hash strings. Some SBOMs will include license data, while others don’t, as there’s no formal requirement to include such data.
  - There’s no definition for data quality on SBOMs.

Problems in handling SBOMs lead to difficulties in drawing conclusions.

## CHOOSING THE RIGHT STRATEGY

### Simple Reflex Agent

- Process SBOM files and parse package information.

### Utility-Based Agent

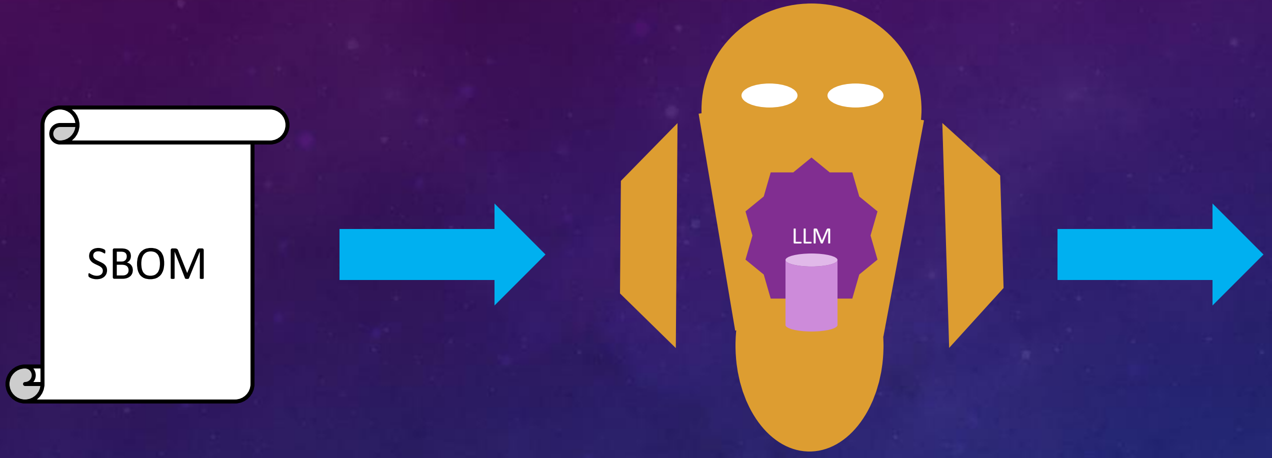
- Identify data gaps autonomously and use tools to enrich data.

### Goal-Based Agents

- Use information from other agents to build a summary, produce compliance artifacts, and generate a conclusion.

## LET'S BUILD OUR AGENTIC AI PROJECT

PROBLEM 1: Parsing information from an SBOM.

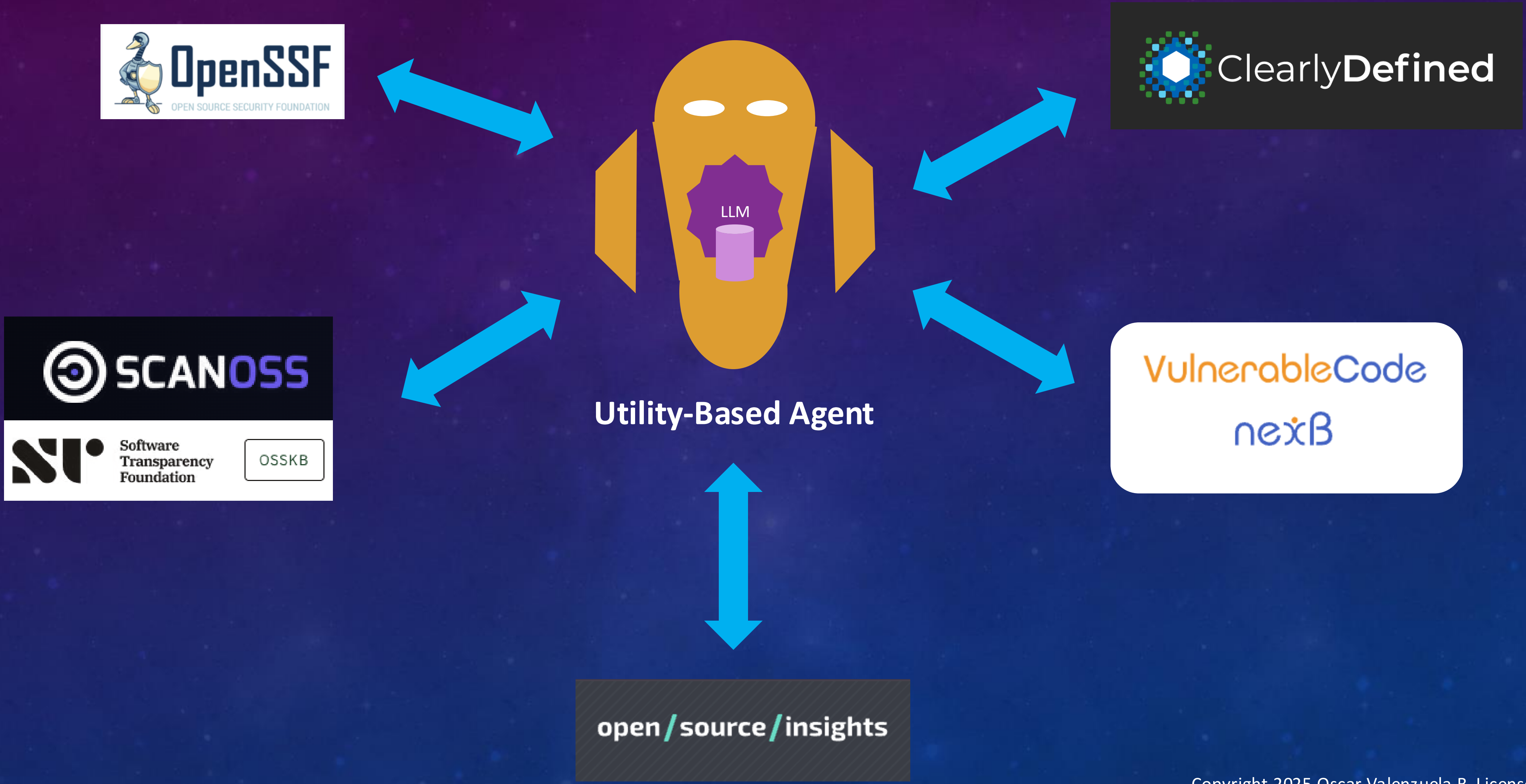


Simple Reflex Agent

| Package URL |
|---|
| pkg:pypi/django@1.11.1 |
| pkg:npm/foobar@12.3.1 |
| pkg:gem/ruby-advisory-db-check@0.12.4 |

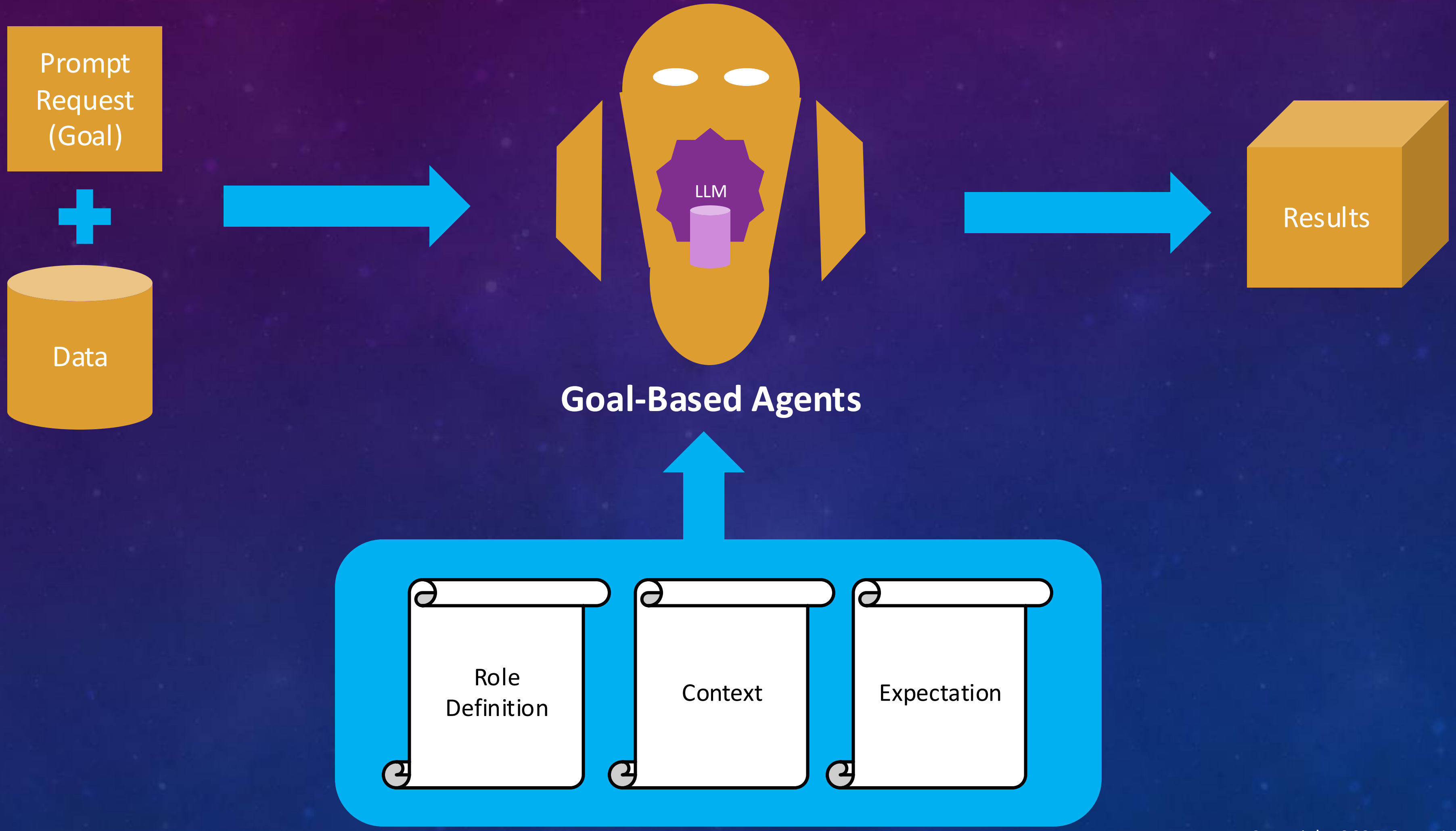
## LET'S BUILD OUR AGENTIC AI PROJECT

PROBLEM 2: Detecting data GAPs and collecting data from trusted sources.



# LET'S BUILD OUR AGENTIC AI PROJECT

PROBLEM 3: Review complementary information to address a GOAL.



## LET’S BUILD OUR AGENTIC AI PROJECT

What do you need?

- A place where agents will live (the workflow).
- An LLM to support operations.
- An input/output for the workflow.
- “Tools” to enrich data (APIs, tooling, etc.).
- An SBOM (to test the workflow).
- Coffee.

## THE FUTURE OF OSS COMPLIANCE: DATA FEEDS OF KNOWLEDGE

- The growth of Agentic AI and the 'no-code' movement will continue to drive automation. This means you won't necessarily need deep coding skills to implement Agentic AI.

- We'll likely see data feed providers become increasingly central to this technological shift. While traditional tools often aim for capability parity “Swiss Army knife Syndrome” the focus will shift towards entities specializing in providing the high-quality data that AI Agents require.

- AI won't replace Open Source Engineers – human oversight and final decision-making remain essential. However, roles heavily focused on routine operational tasks, like simply running scanners and reading raw reports, will need to evolve, adapting to leverage these powerful new AI capabilities.

## ORGANIZATIONS THAT ARE PAVING THE WAY

SCANOSS
Software Transparency Foundation
OSSKB

- Custom KDB for Snippet Detection.
- Cryptographic Algorithms Open Dataset.
- PURL to CPE Relationship mapping project.
- Dataset Geo Provenance (ex: Export Control).

VulnerableCode
nexB

- A free and open vulnerabilities database.
- Open database of all the licenses.
- Scanning as a service (ScanCode.io)

Software Heritage

- A free and open vulnerabilities database.
- Open database of all the licenses.
- Scanning as a service (ScanCode.io)

## ORGANIZATIONS THAT ARE PAVING THE WAY

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

- OpenSSF Scorecard: Project Health
- Vulnerability information

ClearlyDefined

- License Metadata
- Source Code Location
- Copyright and Author information.

open / source / insights

- License Metadata
- Source Code Location
- Copyright and Author information.

## THANK YOU!