1 - Introduction to AOP

Let's start by look at a software example, if an architect is designing a robotics application, the core concerns are the motion management and path computation. The concerns that are common to many of the core modules involve features such as logging, remote management, and path optimization.

These system-wide concerns that span multiple modules are called **crosscutting** concerns. Aspect-oriented programming (AOP) manages these crosscutting concerns. While object-oriented programming (OOP) is the most common methodology employed today to manage core concerns, it is not sufficient for many crosscutting concerns. A typical OOP implementation creates a coupling between the core and crosscutting concerns that is undesirable, since the addition of new crosscutting features and even certain modifications to the existing crosscutting functionality require modifying the relevant core modules.

AOP is a new methodology that provides separation of crosscutting concerns by introducing a new unit of modularization—an aspect—that crosscuts other modules. With AOP you implement crosscutting concerns in aspects instead of fusing them in the core modules. An aspect weaver, which is a compiler-like entity, composes the final system by combining the core and crosscutting modules through a process called **weaving**. The result is that AOP modularizes the crosscutting concerns in a clear-cut fashion, yielding a system architecture that is easier to design, implement, and maintain.