Contribution ID: **2063** Type: **Oral**

The Evolution of the controls-kt Framework: From an Imperative Approach to a Declarative Composite Architecture

Wednesday 29 October 2025 14:45 (15 minutes)

In modern scientific experiments, equipment control systems face constantly growing requirements for reliability, scalability, and ease of maintenance. The controls-kt Kotlin framework, developed at the Scientific Programming Centre MIPT, was a successful first step in this direction, offering a modern reactive model based on Kotlin Multiplatform and coroutines for creating asynchronous data acquisition systems. It allowed for the flexible description of devices and their interactions, solving numerous applied problems.

However, as the complexity and distributed nature of these systems grew, the limitations of the initial imperative approach became apparent. The programmatic assembly of complex device hierarchies with diverse protocols and the manual binding of their states led to increased code complexity. The absence of a formal lifecycle model and declarative capabilities to describe it made it difficult to build fault-tolerant and maintainable systems in a laboratory setting. Furthermore, modeling and simulation existed in a separate paradigm, complicating the transition from prototypes to real hardware.

These challenges prompted a complete architectural redesign, resulting in controls-composite-kt. This is not merely an update, but an evolutionary shift to a new, declarative philosophy. Instead of writing code on how to assemble and run a device, one now creates its "blueprint"—a complete, self-contained, and serializable declaration of what a device is, including its structure, public API, internal logic, and dependencies. A dedicated runtime environment (the new runtime module) then takes on the responsibility of bringing this blueprint to life.

This approach has yielded key advantages. The device lifecycle is now described as a formal and predictable Finite State Machine (FSM) based on the KStateMachine library, which eliminates ambiguity and enhances reliability. Each property value is now intrinsically linked to its acquisition timestamp and a quality assessment, which is crucial for preventing operations with stale or invalid data. The composition of complex systems now resembles assembling a kit, where child components and the bindings between their properties are simply declared in the parent's blueprint.

For complex, multi-step operations requiring atomicity, a "plans" mechanism has been introduced. A plan represents a transaction with the capability for compensating actions (the Saga pattern), which guarantees system integrity even if a failure occurs in one of the steps. A critical consequence of this new approach is the unification of the real world and simulation: a simulation in the new framework is the very same device blueprint but with a "virtual" driver instead of a real one, ensuring a seamless transition from modeling to hardware interaction.

This presentation will cover the process of this architectural evolution. The key design decisions underlying controls-composite-kt will be analyzed in detail, and practical examples will demonstrate how the new declarative DSL simplifies the description of complex systems compared to its predecessor's imperative approach. Special attention will be given to the advantages of the new models in the context of building reliable, observable, and easily maintainable distributed control systems.

Author: KOLPAKOV, Maxim (MIPT)

Presenter: KOLPAKOV, Maxim (MIPT)

Session Classification: Information Technology

Track Classification: Information Technology