SPARKL®

Solution Brief
Embedded Control with the SPARKL® Sequencing Engine

About SPARKL

Bring Machines Together

All enterprises suffer from the black box swamp. Systems that work fine on their own, but won't play nicely with others.

It's hard to describe how a system should work - let alone how or why different systems interact.

SPARKL® is powerful technology for managing the behaviour of distributed systems. The lightning fast, distributed SPARKL Sequencing Engine drives events between machines, applications and things.

It provides Distributed Intelligence for true fog computing, allowing edge devices to interact with or without the cloud.

It introduces Reasoned Provisioning which spins up secure, on-demand infrastructure to meet the need of actual business logic.

Secured by blockchain, SPARKL logs every single event in a clean, connected Audit Trail to solve compliance and regulatory reporting across machines and systems, old and new.

SPARKL designs and develops the SPARKL® Sequencing Engine in London, UK. We work with partners including Cisco and Intel to market the product to innovators and customers worldwide.

SPARKL in Embedded Control

Many manufacturers have a fragmented and insecure IT infrastructure. Undocumented data is divided up in between systems that aren't able to talk to each other properly or work locally. This complexity makes it harder for a business to understand and manage the best use of already scarce resources, such as a capital, operations and staff.

The SPARKL Sequencing Engine is powerful technology, designed to work across embedded and distributed systems alike.

It automates the way machines work together, generating data that is logged in a clean, detailed audit trail. You can prove why, how and where events happened in a business process, enabling real-time alerts, analysis and troubleshooting.

It is implemented using the scalable, high performance environment called Erlang/OTP - a proven platform for telecoms and other high-availability requirements.

SPARKL is designed to solve the large class of problems that arise when applications and systems are used in combination. Although these well-known problems are common to all areas of distributed computing, they have particular impact in embedded systems.

By definition, manually-written programs are black boxes - and they all require custom configuration. For a wide variety of reasons this leads to brittle, unreliable interactions when components are combined to form larger, or composite, systems.

How is it Unique?

The SPARKL configuration abstraction is an entirely new and unified way of expressing the capabilities and intents of collaborating systems, as well as the infrastructure they work on.

It is more expressive, flexible and precise than other well-known abstractions, but it can import such alternatives directly. One such example is the Finite State Machine (FSM), useful in embedded systems; another is the VIRL abstraction used to represent network topologies.

Thus SPARKL delivers high-level benefits during design, development, test, bootstrap and operation - benefits which cannot be achieved by hand-crafted code alone.

How does it Work?

SPARKL includes an execution engine that operates over the configuration abstraction.

No longer does a programmer have to manually code the interactions that form a composite system. Instead, the engine infers the sequence of events from configuration, and drives that sequence to completion. It does this in respect of any initial event that expresses intent.

Because it drives the sequence of events, SPARKL can additionally use the configuration to automatically provision infrastructure artefacts that are required during execution - dynamically and with reason. This leads to new audit capabilities that include all aspects of a composite system.

What is the Net Effect?

The net effect of using SPARKL is that distributed, heterogeneous components automatically combine to form composite systems that are manageable, measurable, resilient and scalable.

The SPARKL abstraction is more expressive and precise than the well-known Finite State Machine, but it can directly import such an abstraction. Take the following, which might represent a Nozzle Guide Vane in an aero engine.

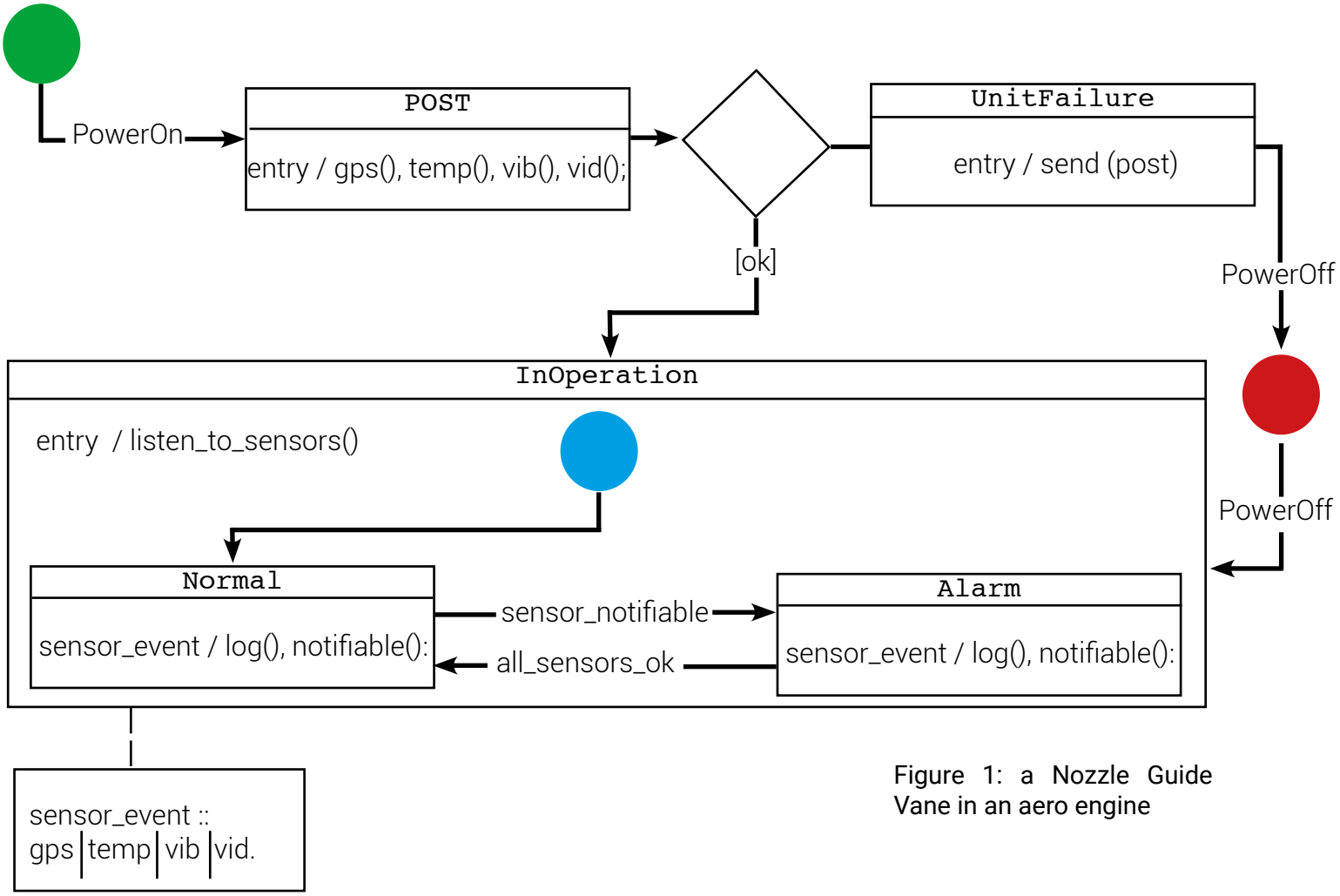

Figure 1: a Nozzle Guide Vane in an aero engine

Example - Nozzle Guide Vane

By importing an XML representation of this FSM directly, the SPARKL Sequencing Engine is able to continuously represent the state of this component, including events such as sensor inputs.

Why bother? Well, it may be that in the state `Alarm`, sensor data is relayed to ground; whereas in state `Normal` it is merely logged for retrieval on landing or at maintenance intervals.

SPARKL keeps these configuration abstractions in a hierarchical tree. A higher-level composite FSM might be called Engine, comprising individual guide vanes, fuel valves and so forth. A yet higher level composite might be called Aeroplane, comprising engines, hydraulic systems and so forth.

Thus, SPARKL provides the means by which embedded systems and their associated physical components can be modelled, aggregated and controlled in a transparent, repeatable and reliable way.

Find Your Use Case

Mark Dawber
Head of Business Development
mark@sparkl.com

sparkl.com
@sparkl

See SPARKL tutorials and demos at
sparkl.com/docs/web