A Systems Architecture for IoT Connected-edge Runtime Configuration

Ebrahim Y. Said
Faculty of Engineering & the Built Environment
University of the Witwatersrand

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Johannesburg 2022

Abstract

This thesis presents a systems architecture method that applies object, process, and state as primary attributes for modelling smart devices in an IoT application ecosystem. This systems architecture method is grounded in pure systems thinking that enables a significant focus on systems purpose using a state-of-the-art modelling notation while supporting complexity reduction in systems architecture model representation.

In current research, the Internet of Things (IoT) body of work mainly applies functional and domain coverage related to the representation of its architecture. This abstraction aims to hide enabling technologies while simultaneously trying to handle system complexity. Increases in complexity due to the evolution of IoT edge systems have seen requisite increases in technological needs advancement. IoT edge systems enable the physical extensions of the internet that exists today. This cyber-physical evolution goes beyond the services that individual embedded devices expose. It underpins the IoT by creating connected edge systems that integrate large-scale digital and physical processes. The variation in heterogeneity predominantly causes increases in complexity in these systems. A systems approach can reduce the complexity in stating a systems architecture for the IoT Connected-edge (IoT-Ce). This approach stems from the belief that low-level IoT-Ce subsystem characteristics that influence the state of the system can abstract to a conceptual IoT system architecture to increase the accuracy of model dependability through increased state awareness.

This thesis presents two contributions. Firstly, the study proposes a standards-based systems architecture method that outlines the systems, subsystems, and components employed at runtime in the IoT-Ce. The Object Process Methodology (OPM) ISO-19450 guides the modelling of the systems architecture. The **IoT-Ce Systems Architecture for Runtime Configuration (ISARC)** suggests a hierarchical structure for a multi-layered connected heterogeneous IoT edge system. The study of the runtime properties of systems can provide a better understanding of how to model systems that are dependable and scalable. Additionally, these systems have the essential properties to find application in large-scale

distributed systems architecture. The primary contribution is the suggestion that identifying well-defined system, subsystem, and component boundaries using a known approach supports complexity reduction in IoT-Ce models. The proposed approach provides the method for testing variations of configuration for the IoT-Ce.

Secondly, the thesis provides a method to validate the ISARC using OPM and Design Structure Matrices (DSM). The method for modelling configuration decisions for the IoT-Ce uses OPM as the basis for decision parameter extraction and DSM to represent the decision structure. A test of conceptual architecture variations of the ISARC uses a novel concrete control design based on instantaneous and accumulated errors. This contribution's primary benefit extends the systems architecture by proving that the ISARC can support ongoing runtime configuration decisions using a series of Python simulated data sets. Finally, these data sets compare a historical data set that mimics a hierarchical systems architecture to argue the applicability of parameter-based architecture decisions for the IoT-Ce.

The ISARC aims to position an approach to simplify system complexities by suggesting the function and form of the IoT-Ce. The effort to reduce complexity in design is in line with an existing body of research in model-based systems engineering using the OPM ISO-19450 standard.

Keywords IoT System Architecture, OPM ISO-19450, IoT Connected Edge, Runtime Configuration, Design Structure Matrix, Architecture Decisions, PI Control, Python