# Discrete-Event Control Applications in Cloud System Modelling via P Programming Language

## Introduction

Cloud Computing is a vastly impactful transformation and continues to accelerate as the global market is forecasted to surpass $1 trillion in 2024 per the [International Data Corporation(IDC)](https://www.businesswire.com/news/home/20201015005069/en/Cloud-Adoption-and-Opportunities-Will-Continue-to-Expand-Leading-to-a-1-Trillion-Market-in-2024-According-to-IDC). As the dependency of many vital sectors to a handful of public cloud providers extends rapidly, the responsibility of the engineering teams that develop those public cloud offerings intensifies. All the advancements (scalability, flexibility, performance, etc.) of cloud computing relies on these engineering teams to solve complex technical problems and manage sophisticated backend systems. Thus, it is highly important to keep the balance between improving the public cloud services and improving the tools that are used by the engineering teams. There is a positive-feedback loop in these two processes, as the cloud engineering teams also benefit from their own products and services as they are developing future projects. Yet, they also need more adequate tools to understand and simulate the depths and breath of these highly complicated distributed systems, hence abstracting these complexities from the end-customer is a major win for cloud offerings. One really exciting tool that has gained steam in the last decade is the P Programming Language, *“a state machine based programming language for formally modeling and specifying complex distributed systems”*([P Github page](https://github.com/p-org/P)). As a popular tool that is used by engineering teams of public cloud providers for many different tasks such as AWS and Microsoft Azure, P is still actively being developed as an open-source project. This research project will explore the applicability and usefulness of Discrete Event Systems(DES) type of control theory practices on cloud systems leveraging P programming language. Along the way, it will attempt to create a “plug-in” DES module or template framework for the P programming language.

## Problem Statement

The benefits of using formal methods in the design and testing of distributed systems has been long acknowledged by the cloud computing community. A great example is the P programming language, [providing ways](https://www.youtube.com/watch?v=FdXZXnkMDxs) to: (i) create a “formal model” of the system, (ii) establish “formal specifications” to its behavior, and (iii) testing possible “inputs and faults”. However, in practice, the main advantage of the formal methods are only guaranteed at the “testing of inputs and faults” portion thanks to the built-in “*black-box*” testing backend. The “formal” modeling and specification rests in the developers’ hands to take the full advantage of proven mathematical techniques at the design level. As it is unrealistic to expect every full stack developer to have a deep math background, there is a need to better understand how to encourage the use of formal methods at these two stages more. More specifically, the following research questions need to be addressed:

* What are the formal methods, if any, that fits into the idea of “modeling” and “specification” practices of the P?
* How to introduce these formal methods to the P programming language for easy adaptation by engineers who are not familiar with them in the first place?
* Does introducing these formal methods at the “modeling” and “specification” portions of P result in development speed or testing efficiency?

## Objectives

As an open-source tool that emphasizes the effectiveness of formal-methods and modular design, P programming language should enable different contributors to add new “formal-method” applications to its ecosystem and establish a framework where these contributions can easily be used by others as well. The long term goal of this research is to create a *blueprint* for developing these “*formal-method plug-ins*” for the “modeling” and “specifications” aspects of the P programming language. A “*plug-in*” is defined herein as a pre-constructed bundle of [state-machines, events, and functions] that can be integrated to a P project similar to a Python package or Java class or npm module. These plug-ins would represent different formal-methods that the contributors want to apply to their P projects. Particularly, the study has the following sub-objectives:

* To identify and compare different potential forms of “plug-in” implementations in P
* To create an example “formal-method plug-in” using Discrete-Event Systems(DES) control concepts that can be used in either “modeling” or “specification” portions of P, and test its effects
* To discuss potential ways of sharing these “formal-method plug-in” in different P projects to plant the seeds of a collaborative formal-method ecosystem for distributed system designs

The result of this study would be beneficial for both “formal-method” researchers to find a practical application play-ground and industry cloud engineers for gaining more tools for designing more resilient distributed systems.

## Preliminary Literature Review

There have been two separate preliminary literature reviews conducted for this proposal. The first one is about the applications of DES and cloud computing. Ullah[[6]](#_nz8el819ozl3) provides a survey of different control theory approaches studied for managing elasticity, a major advantage of modern cloud systems. Cañizares[[1]](#_nz8el819ozl3) introduces a GUI application for using DES methods on designing and simulating cloud systems, which has significant overlap with P’s mission yet fundamentally differs by the design flexibility that a programming language provides over pre-defined GUI commands. Finally, Cassandras and Lafortune[[2]](#_nz8el819ozl3) use database consistency as an example scenario for their display of different DES methods, which is easily transferable to distributed system context. In fact, there is a “*Two-Phase Commit”* tutorial in the [P Github page](https://github.com/p-org/P/tree/master/Tutorial/2_TwoPhaseCommit).

The second review is about the P programming language, since a basic understanding of its workings is critical for using it as a verification tool in this study. Desai explains the mechanism of the programming language[[3]](#_nz8el819ozl3) and the mathematical concepts behind it[[4]](#_nz8el819ozl3) in two different papers. Kushwah[[5]](#_nz8el819ozl3) presents a domain-specific application of the P language. Finally, Desai [presents P in 2023 re:Invent conference](https://www.youtube.com/watch?v=FdXZXnkMDxs) in a customer-friendly non-math heavy manner.

## Methodology

The primary research methods will be similar to the ["Compositional programming and testing of dynamic distributed systems."](https://ankushdesai.github.io/assets/papers/modp.pdf) paper, expanding *ModP’s* semantics and programming constructs in order to fit the concept of “formal-method plug-ins” to the bigger picture.

Once the theoretical placement is set, a big portion of the study time will go into actually implementing some basic DES concepts in a P project through coding. These concepts will be already established formal-methods such as *unary/compositional operations and basic observer/diagnoser automata*. These implementations will be tested on predetermined performance metrics such as CPU usage and time. Optionally, there can be an attempt to predict development speed improvements and how these formal-methods increase an engineer’s understanding of a distributed system at the design step, which is a big emphasis of the purpose of the P programming language.

Finally, there would be a discussion section on how these “plug-ins” can be packaged for code-sharing across P projects and a potential demonstration of such a use case.

## References

[1] [Cañizares, Pablo C., et al. "Simcan2Cloud: a discrete-event-based simulator for modelling and simulating cloud computing infrastructures." *Journal of Cloud Computing* 12.1 (2023): 133.](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w)

[2] [Cassandras, Christos G., and Stéphane Lafortune, eds. *Introduction to discrete event systems*. Boston, MA: Springer US, 2008.](https://link.springer.com/chapter/10.1007/978-0-387-68612-7_11)

[3] [Desai, Ankush, et al. "P: safe asynchronous event-driven programming." *ACM SIGPLAN Notices* 48.6 (2013): 321-332.](https://people.csail.mit.edu/zufferey/files/2013_P_safe_asynchronous_event-driven_programming.pdf)

[4] [Desai, Ankush, et al. "Compositional programming and testing of dynamic distributed systems." *Proceedings of the ACM on Programming Languages* 2.OOPSLA (2018): 1-30.](https://ankushdesai.github.io/assets/papers/modp.pdf)

[5] [Kushwah, Shivendra, et al. "PSec: programming secure distributed systems using enclaves." *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 2021.](https://ankushdesai.github.io/assets/papers/psec.pdf)

[6] [Ullah, Amjad, et al. "A control theoretical view of cloud elasticity: taxonomy, survey and challenges." *Cluster Computing* 21 (2018): 1735-1764.](https://dl.acm.org/doi/abs/10.1007/s10586-018-2807-6)