Scheduling and Running Software Test-cases Based on AI

Seth Kofi Agbavitor | B M Nafis Fuad

Master’s Thesis in Computer Science, Spring 2025

Supervisor: Morten Mossige

Universitetet i Stavanger

Introduction

Software reliability is very important, especially in robotics, where failures can lead to costly downtime and safety risks. The execution of traditional software tests often follows fixed orders, which limits efficiency and delay fault detection. This project explores intelligent test case scheduling using historical test results and real-time data to prioritize test cases with a higher likelihood of failure. By combining reinforcement learning for dynamic test case prioritization and constraint-based scheduling, the system improves fault detection, reduces testing overhead, and enhances cloud resource utilization.

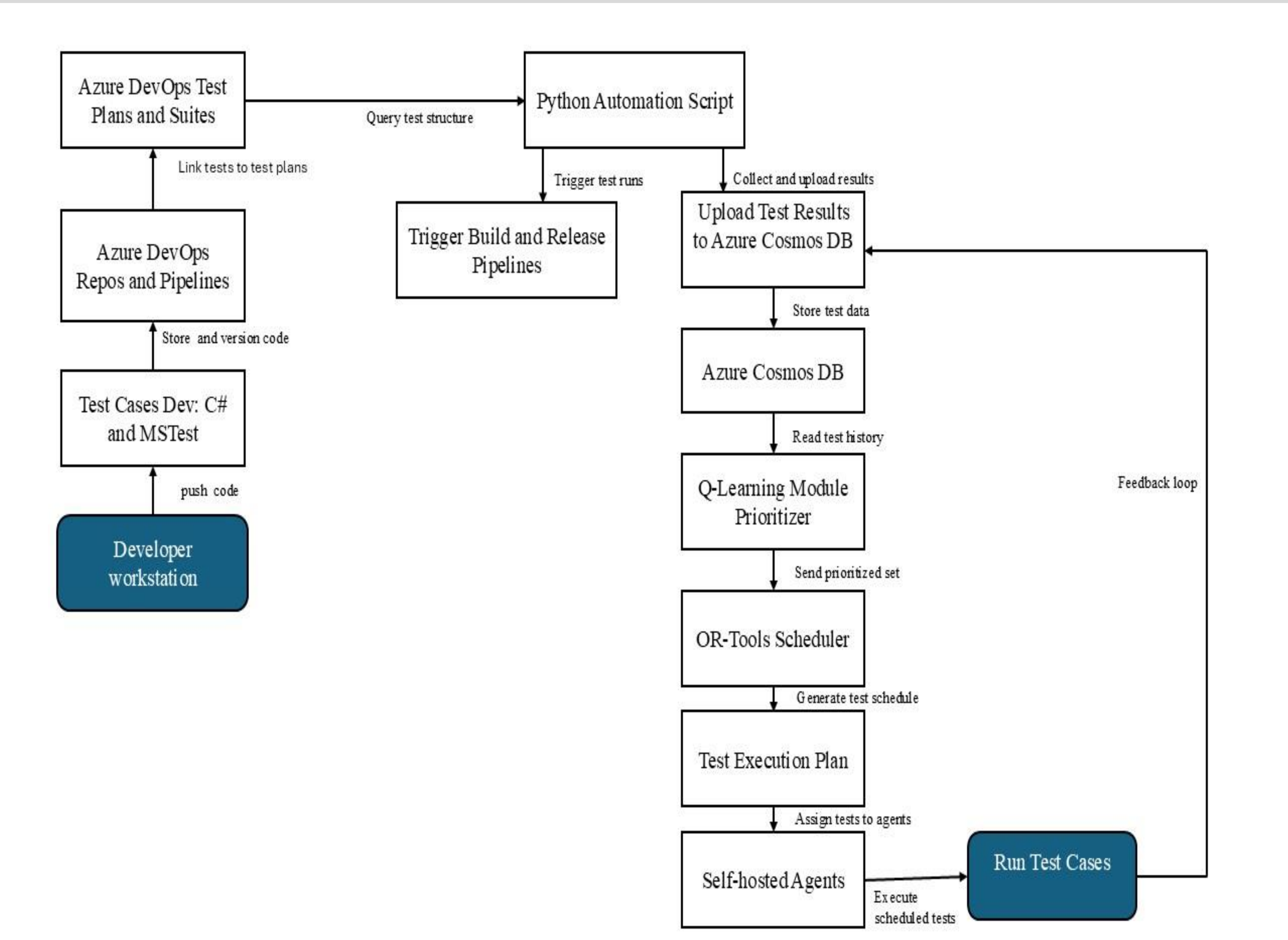
Problem Statements

In large-scale software testing environments, running all test cases in every CI/CD cycle is time-consuming and inefficient. This thesis addresses the problem of intelligently selecting and scheduling test cases using Q-learning, a reinforcement learning technique. The goal is to maximize fault detection and test effectiveness while minimizing execution time and redundant test runs.

Objectives

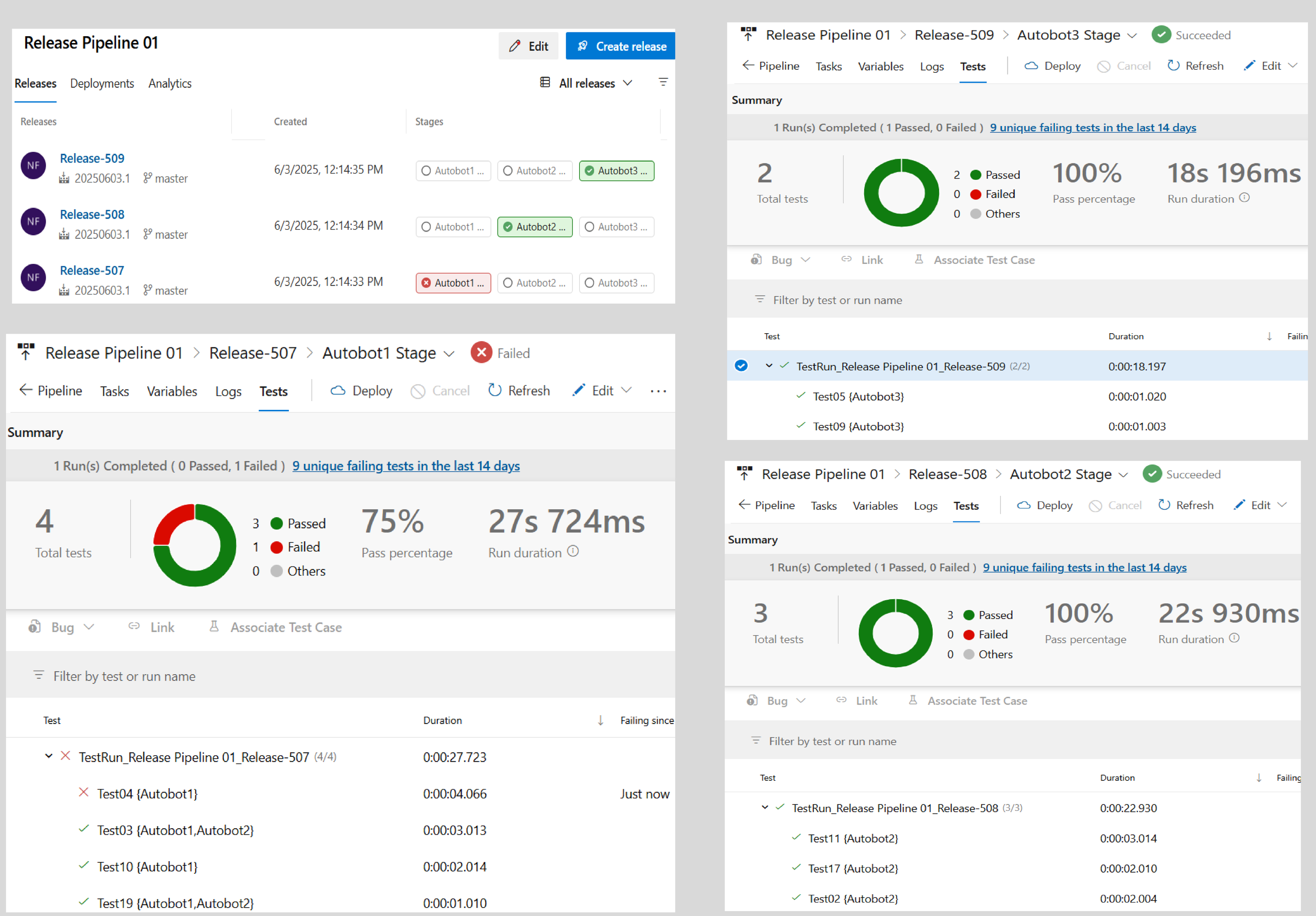
- Develop a software component that reads and analyzes automated test results stored in Azure Cosmos DB.
- Design an AI-based scheduling system that dynamically prioritizes test cases using historical and real-time data.
- Apply optimization techniques, including Google OR-Tools, to improve the efficiency of test case scheduling.
- Integrate the solution into the ABB internal test infrastructure to improve test execution and resource utilization.

Methodology



System architecture showing the workflow from test development to scheduling and execution.

Azure DevOps Test Logs



Results

📊 Test Cases based on Q-Values:

	testCaseId	testCaseTitle	Q_value	failRate	avgTime	recency
3	6	Test04 {Autobot1}	6.918472	1.000000	4.041838	118.754501
19	22	Test20 {Autobot1,Autobot2,Autobot3}	5.860982	0.333333	4.009775	191.343900
14	17	Test15 {Autobot2,Autobot3}	5.860982	0.166667	5.009961	191.326909
12	15	Test13 {Autobot3}	4.752429	0.000000	5.009139	191.276049
2	5	Test03 {Autobot1,Autobot2}	4.219534	0.186275	3.016539	191.344023
6	9	Test07 {Autobot2,Autobot3}	4.182319	0.571429	3.014543	118.720152
10	13	Test11 {Autobot2}	4.182319	0.504762	3.010552	118.720137
11	14	Test12 {Autobot1,Autobot2}	4.158467	0.009804	4.009010	191.343954
13	16	Test14 {Autobot1,Autobot3}	4.158467	0.107843	3.009324	191.343941
7	10	Test08 {Autobot1,Autobot2,Autobot3}	4.041481	0.000000	4.009275	191.343981
17	20	Test18 {Autobot3}	3.326273	0.000000	3.008426	191.276036
9	12	Test10 {Autobot1}	3.105214	0.419048	2.010000	118.754475
1	4	Test02 {Autobot2}	3.105214	0.523810	2.023733	118.720167
16	19	Test17 {Autobot2}	2.987605	0.078431	2.008284	191.326896
15	18	Test16 {Autobot1}	2.345459	0.342857	2.009876	118.754461
18	21	Test19 {Autobot1,Autobot2}	2.182813	0.058824	1.009431	191.343914
5	8	Test06 {Autobot1,Autobot3}	2.073853	0.285714	2.010010	118.754488
8	11	Test09 {Autobot3}	1.995541	0.000000	1.007743	191.276062
4	7	Test05 {Autobot3}	1.995541	0.000000	1.013703	191.276148
0	3	Test01 {Autobot1}	0.280649	0.057143	1.019029	118.754554

⭐ Total Q Value: 71.7336

Prioritized Test cases based on Q-Values

📌 Selected 9 out of 20 test cases.

	testCaseId	testCaseTitle	Q_value	avgTime
0	6	Test04 {Autobot1}	6.918472	4.041838
4	5	Test03 {Autobot1,Autobot2}	4.219534	3.016539
6	13	Test11 {Autobot2}	4.182319	3.010552
11	12	Test10 {Autobot1}	3.105214	2.010000
12	4	Test02 {Autobot2}	3.105214	2.023733
13	19	Test17 {Autobot2}	2.987605	2.008284
15	21	Test19 {Autobot1,Autobot2}	2.182813	1.009431
17	11	Test09 {Autobot3}	1.995541	1.007743
18	7	Test05 {Autobot3}	1.995541	1.013703

⏱ Total Available Time: 20 sec
⏱ Total Scheduled Time: 19.14 sec
⭐ Total Q Value: 30.6923

Scheduled Tests using Google OR

Conclusion

This project demonstrates that intelligent test case scheduling using Q-learning and constraint-based optimization can significantly improve fault detection and testing efficiency. By leveraging historical and real-time data, the system reduces execution time, minimizes redundant tests, and enhances resource utilization. The successful integration into ABB’s test infrastructure will highlight its practical value in large-scale, safety-critical environments like robotics.

References

[1] H. Spieker, A. Gotlieb, D. Marijan, and M. Mossige, “Reinforcement Learningfor Automatic Test Case Prioritization and Selection in Continuous Integration,”in Proceedings of the 26th ACM SIGSOFT International Symposium on SoftwareTesting and Analysis (ISSTA). ACM, 2017, pp. 12–22.

[2] . C. Jorgensen, Software Testing: A Craftsman’s Approach, Fourth Edition. CRCPress, Dec. 2018, google-Books-ID: TvHcDwAAQBAJ.