Abstract

According to Software maintenance consumes 75% of the total IT budget of a company [4].

According to Curtis [4], in 2018, number of software failures had surpassed 100M which consequently represent a significant waste of resources.

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
AV	Autonomous Vehicle
ASIL	Automotive Safety Integrity Level
MDE	Model Driven Engineering
PDF	Probability Distribution Function
RL	Reinforcement Learning
GSN	Goal Structuring Notation
EDA	Exploratory Data Analysis
GAN	Generative Adversarial Network
iCM	Intuitive Certainty Measure

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Benefits of Architecture Recovery

- New members of a development team can benefit from the architecture by having access to the group memory of all the artifacts generated during the development process. Tools such as Hipikat [1] can generate such a memory by analyzing which developers have the most contribution for a component. Leveraging this information, new developers will know which team member to contact in case of any questions.

- Project managers can benefit from the information on which parts of the project is actively changing and how this change is affecting the structure of the project [5].

- By creating a mapping between the changes and author identifiers, project managers can also understand who has worked on an artifact and thus, is more knowledgeable about a problem at hand for the same artifact [3].

- Identify the distance between the requirement and implementation of the system, i.e., as-is-architecture and as-it-should-be architecture [2]
- Architecture Recovery can help learn the structure of a program and how it satisfies the domain needs

Bibliography

[1] Davor Čubranić, Gail C. Murphy, Janice Singer, and Kellogg S. Booth. Hipikat: A project memory for software development. In *IEEE Trans. Softw. Eng.*, volume 31, pages 446–465, jun 2005.

[2] Wolfgang Eixelsberger. Recovery of a Reference Architecture, A case study. Technical report, 1998.

[3] Tudor Grba, Adrian Kuhn, Mauricio Seeberger, and Stéphane Ducasse. How developers drive software evolution. In *Int. Work. Princ. Softw. Evol.*, volume 2005, pages 113–122, 2005.

[4] Herb Krasner. The Cost of Poor Quality Software in the US: A 2018 Report The Cost of Poor Software Quality in the US: A 2018 Report 2 Contents. Technical report, CISQ, 2018.

[5] MM Lehman. Programs, life cycles, and laws of software evolution; Programs, life cycles, and laws of software evolution. Technical report, 1980.