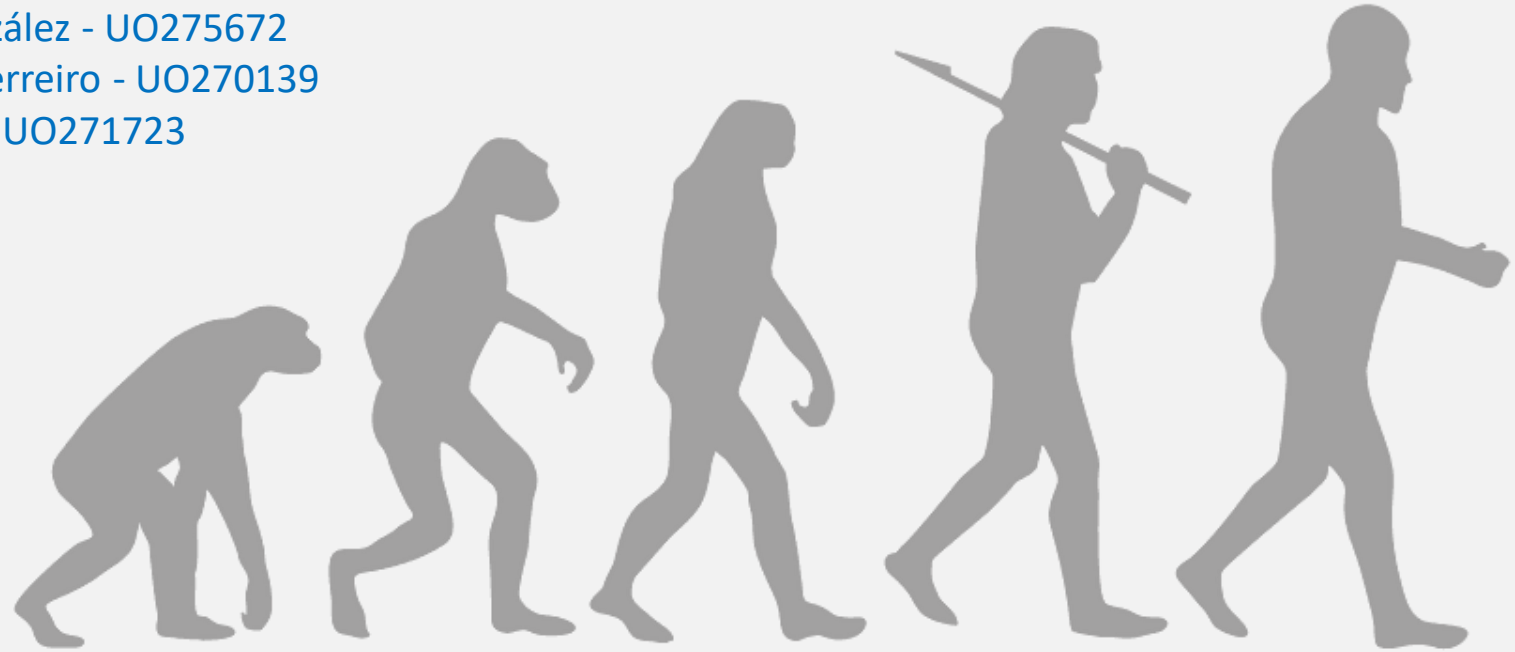
LEHMAN'S LAWS

OR THE LAWS OF SOFTWARE EVOLUTION

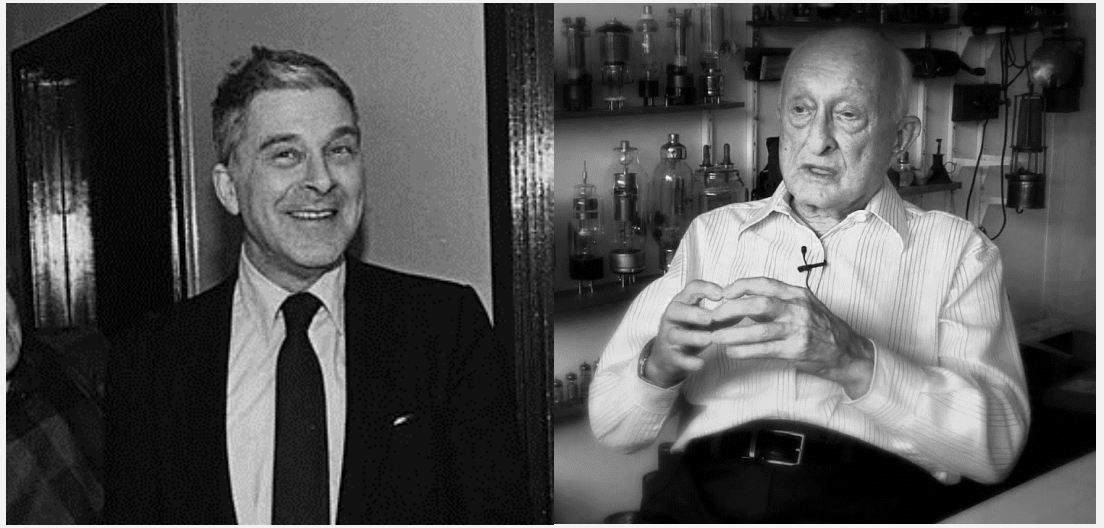
Carolina Barrios González - UO275672
Luis Miguel Alonso Ferreiro - UO270139
Jesús Alonso García - UO271723



CONTENTS

INTRODUCTION

- Laws of software evolution.
- By Manny Lehman and László Bélády in the 70s.
- Comprehend the changes that software experiences along its existence



INTRODUCTION

- 1969: Empirical study while working on IBM.
- Purpose: Improving programming effectiveness of the company.
- New field of research: *software evolution.*
- First 3 laws: 1974 at the Imperial College of London.
- The laws were revisited, edited, and updated until 1996.

INTRODUCTION

- Three main system categories:
  - *S-programs,* which are written according to an exact specification of what that program can do.
  - *P-programs,* which are written to implement certain procedures that completely determine what the program can do.
  - *E-programs,* which are written to perform some real-world activity.

THE LAWS

1: CONTINUING CHANGE

- A program used in a real-world environment must continuously re-adapt to the environmental changes.
- When the modified system is re-introduced, it promotes environmental changes.
- If it is not re-adapted, it turns progressively useless, until it becomes deprecated.

2: INCREASING COMPLEXITY

- As an evolving program changes its structure, it becomes more complex.
- It is necessary to spend time to preserve and simplify the structure.
- This maintenance does not add functionality.

3: LARGE PROGRAM EVOLUTION

- Program evolution is a self-regulated process.
- System attributes tend to remain invariant for each system release.
- Making small changes reduces the extent of structural degradation.

4: ORGANIZATIONAL ESTABLILITY

- The rate of development is constant.
- A change of resources has imperceptible effects on the long-term evolution of the system.
- Large development teams are often unproductive.

5: CONSERVATION OF FAMILIARITY

- A system is bound to change over its lifetime.
- The number of changes/new features should be limited.
- This helps with errors/faults and understanding the system.

6: CONTINUING GROWTH

- Every new release should bring something new.
- Closely related with Law 1.

7: DECLINING QUALITY

- The maintenance of a system is critical to its quality.
- New features will make the system more complex.
- More effort is needed to maintain the quality.

8: FEEDBACK SYSTEM

- Developing software is a multi-loop, multi-agent, multi-level feedback.
- As a system grows it becomes much harder to change and maintain.
- User feedback is really valuable.

CONCLUSION

- Lehman's laws were a breakthrough back in the 70s, but we think its time for an update.
- The way we develop software systems has changed a lot, so they need to stay relevant.
- Laws? Not really. More like best practices.

LEHMAN'S LAWS

OR THE LAWS OF SOFTWARE EVOLUTION

Carolina Barrios González - UO275672
Luis Miguel Alonso Ferreiro - UO270139
Jesús Alonso García - UO271723