Linkages among the Laws of Software Evolution throughout Object-Oriented Software Development Processes: Experimental Studies

Marco Antônio Pereira Araújo, Guilherme Horta Travassos
Federal University of Rio de Janeiro/COPPE
{maraujo, ght}@cos.ufrj.br

1. Introduction

Software systems are growing quickly in needs of functionality, complexity, size and structure. Therefore, it becomes necessary to advance the ways where the software suffers changes, so it can be modified more easily to accommodate these needs. However, anticipating the changes is not an easy task, once there are a lot of reasons why the systems change. The Laws of Software Evolution (LSE) [4] describe how a system behaves throughout their successive versions regarding its decaying. Most of the experimental studies supporting these LSE are regarding legacy systems' source code. Besides, as pointed by Lehman, it is necessary to evolve the studies regarding object-oriented systems [5] and other software development process abstraction levels [6]. We believe the LSE can also be observed when dealing with the other software development process phases. Therefore, we presented [1, 2, 3] a set of hypotheses to verify whether the LSE can also be supported by the different phases of a software development process, using the object-oriented paradigm. We mapped each one of these hypotheses considering the software characteristics’ trends associated with the corresponding metrics for each phase. Thus, we intend to experimentally study these trends in order to have a better understanding of how systems can evolve, considering the linkages among LSE.

2. Linkages among the LSE

The LSE are not independent [9, 10]. Thus, the presented hypotheses, defined by software characteristics’ trends [1, 2, 3] (table 1), were discussed in order to identify the linkages among the LSE (figure 1). This discussion is particularly relevant because this work is concerned with the identification of the software decay process rather to study the individual behavior of LSE.

Therefore, this paper aims to describe a framework, based on the LSE, for experimental studies on OO software decay. The purpose is the elaboration of a conceptual structure to support the definition of experimental studies in different OO software development process phases. The development phases regard Requirements Specification, High and Low Level Design and Coding are usually present in most of the OO software development processes.

Figure 1. Linkages among the LSE

However, it’s important to describe how LSE influence each other. Table 1 describes the logical formulation for the hypotheses related to each LSE. These established hypotheses are the definition bases of the LSE main cause, what logically defines the LSE itself. However, the linkages among the LSE (figure 1) indicates that some of them have straight influence in others; that is, if the hypothesis presented for a specific LSE is true, we should also verify their impact in the LSE that could be directly affected.

In Table 1, we characterize as Size the amount of artifacts produced in each phase of a software development process. As Periodicity, the interval of time elapsed between each produced version of an artifact. As Complexity, the elements that can measure the structural complexity of an artifact. As Modularity, the coupling and cohesion characteristics between artifacts. As Reliability, the amount of identified defects by artifact in each version of it. As Efficiency, the amount of people and allocated
resources, spent time and average productivity of the team, by each artifact. Finally, as Maintainability, the efficiency in the identification and removal of defects in each version of an artifact.

Table 1. The Hypotheses for the LSE

<table>
<thead>
<tr>
<th>LSE</th>
<th>Hypotheses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuing Change</td>
<td>(Size doesn’t decrease A Periodicity doesn’t increase)</td>
</tr>
<tr>
<td>Increasing Complexity</td>
<td>(Size increase V Complexity increase V Modularity decrease V Maintainability decrease)</td>
</tr>
<tr>
<td>Self Regulation</td>
<td>(Size doesn’t increase A Reliability doesn’t decrease A Efficiency doesn’t decrease)</td>
</tr>
<tr>
<td>Conservation of Organizational Stability</td>
<td>(Size constant A Efficiency constant)</td>
</tr>
<tr>
<td>Conservation of Familiarity</td>
<td>(Size constant A Complexity constant)</td>
</tr>
<tr>
<td>Continuing Growth</td>
<td>(Size increase A Periodicity doesn’t increase)</td>
</tr>
<tr>
<td>Declining Quality</td>
<td>(Complexity increase V Modularity decrease V Reliability decrease V Maintainability decrease)</td>
</tr>
<tr>
<td>Feedback System</td>
<td>(Collection of relative measures to Size, Periodicity, Complexity, Modularity, Reliability, Efficiency, Maintainability)</td>
</tr>
</tbody>
</table>

Based on these studies, we believe that these linkages are applicable not only inside each phase of the software development process, but also through these phases in a software development process. Therefore, we propose a hypothesis based on the fact that the software characteristics’ trends, which logically determine the LSE behavior, should be investigated through the software development phases throughout their successive versions, considering the linkages among the LSE, in order to identify the software decay causes.

3. On Going Works

We hope to experimentally evaluate the applicability of the LSE in the context of object-oriented software development processes through a framework that allows us to study the decay causes and their consequences in the OO software development process. Having the ability of accomplishing some experimental studies can give us insights regarding the feasibility of building software decay models using simulation techniques based on system dynamics models [8]. It will allow us to simulate the behavior of the established hypotheses and compare our results with those described in [12].

The next activities include performing some experimental studies, for available information systems, to evaluate the established hypotheses, according to the experimentation process defined in [7]. For these studies, an environment that offers tool support for experimentation studies is being defined and built [11].

As future works, we need to elaborate experimental studies to evaluate the linkage among the characteristics throughout the software development process.

4. Final Considerations

This work has been developed into the context of a doctorate thesis, associated to the Experimental Software Engineering Team, from the Systems Engineering and Computer Science Department - COPPE/UFRJ (http://www.cos.ufrj.br/~ese).

Acknowledgments

We would like to thank Prof. Barbara Kitchenham, Prof. Shari Lawrence Pfleeger and Prof. Manny Lehman for the valuable comments and technical material provided.

References