Improving Variability of Applications using Adaptive Object-Models

João Gradim Pereira

Dissertation realized under the supervision of *Ademar Aguiar (Ph.D.)* and *Hugo Ferreira (Ph.D. AbD)*
at *Tecla Colorida, Lda*

1. Motivation

Software systems are usually designed with a specific purpose in mind. They rely on a series of requirements which are often very difficult to capture and maintain, as they have a tendency to evolve faster than the implementation. This is caused mainly by the poor understanding by the stakeholders about their needs and expectations about what a software system should be able to do [**?**]. These situations lead to higher costs in software development, as creating and maintaining software systems is a knowledge intensive task [**?**]. Moreover, most of these system are not static, and have a constant need to evolve in order to adapt themselves to their environment and new business rules, shifting the stakeholders' needs and expectations about these software systems.

In face of these situations, new development methods started to focus more on iterative and incremental approaches, accepting *incompleteness* as part of every software system's development cycle [**?**]. At the same time, many new systems are being developed with an emphasis on flexibility and run-time configuration [**?**].

2. Objectives

At the same time, many new systems are being developed with an emphasis on flexibility and run-time configuration [**?**]. An Adaptive Object-Model (AOM) is an architectural style based on metamodeling and object-oriented design that exposes its domain model to the end-user, and aims to create better mechanisms for the evolution and adaption of software systems to their environments.

The first chalenge is to adapt some of the ideologies and techniques of the AOM architectural style to a Ruby on Rails application (using an MVC architecture), using an application design based on a static schema from a relational database (MySQL), by applying a series of design patterns connected with AOM architectures.

The second goal is to measure how the variability and performance of each component are affected by the application of these design patterns.

3. Approach & Results

4. Conclusion