

Software Development

Leandro Caniglia

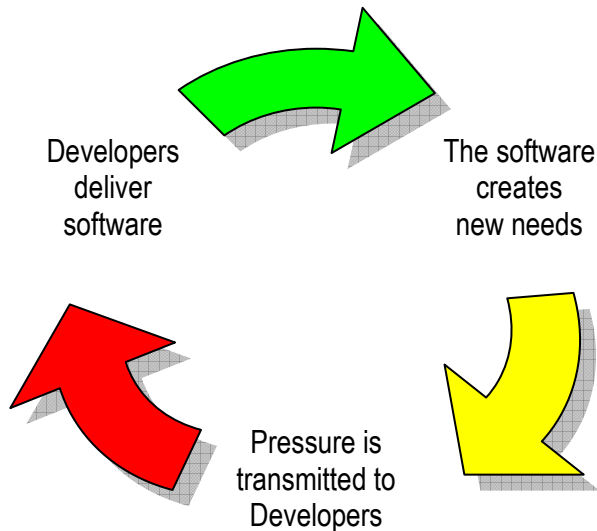
October 24, 2007

Software development is a highly sophisticated discipline whose success requires the concurrence of many best practice patterns. Smalltalk is an ideal environment to integrate all of them homogenously throughout the entire development process. This talk reviews several aspects of software production and shows why smalltalkers are in a privileged position to address its challenges.

Introduction

Computer programming consists of a series of activities aimed to produce an executable model of some domain. These activities include learning, brainstorming, design, coding, tool building, etc. However, *software development* goes far beyond the admittedly fundamental task of programming and failing to identify and address all its complementary requirements may ruin an otherwise successful project.

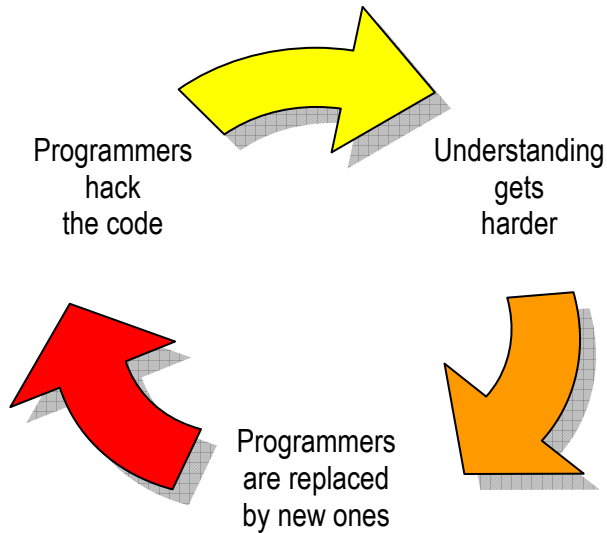
The extensive use of computers has made users to no longer have *desires* on what an application should provide them. Instead of desires, computers have created actual *new needs*. Unless timely fulfilled, unsatisfied new needs may negatively impact jobs, businesses and lives. In turn, needs already satisfied greatly increase the urgency for further features. As a consequence of this boomerang effect the pressure on the development team gets amplified. Incorrectly managed relationships among the parties eventually overheat and the consequences are catastrophic.



Under this fairly familiar scenario the ability to manage pressure becomes unattainable to the dev team. Worse than that, developers are invariably encouraged to produce *quick*

and dirty code under the unrealistic promise that they will have the chance to readdress it in the future. Useless to say, that future never arrives and the moral of the dev team gradually deteriorates to disdain, apathy or plain frustration.

Inappropriately managed, the intellectually rich activity of programming, originally intended to enhance the horizons of human creativity, gets reduced to a tedious practice. Eventually, developers quit or get promoted to positions where programming can be forever avoided. This creates a second vicious circle:



In this presentation we will review some of the actions that may be implemented to turn the vicious circles into virtuous ones, or better than that, avoid them completely from the very beginning.