Smart Technologies in Software Life Cycle

Zane Bičevska and Jānis Bičevskis

Datorikas Instituts DIVI, A.Kalnina str. 2-7, Riga, Latvia
University of Latvia, Raina blvd. 19, Riga, Latvia
Zane.Bicevska@di.lv, Janis.Bicevskis@lu.lv

Abstract. In software life cycle models traditionally the main attention is being paid to the software development, including requirement gathering (specification), design, implementation and testing. Less research is devoted to the system maintenance and operation despite the fact that these aspects take up the main part of the duration of a successful system. In the paper smart technologies are being analysed – architectural designs and software components which using meta information on system and its usage conditions are able to solve efficiently the problems of maintenance and usage: data quality and performance monitoring, software flexibility and testability, context dependant user interface. The advantages of smart technology usage are pointed out helping to improve software maintenance and operation processes.

Keywords: Smart technologies, Testing, Maintenance, Life Cycle models.

1 Introduction

The driver of nowadays IT industry is the speed and search for new technologies. It has a strong impact on the process of application development: on the one hand, the high competition in the market requests a high quality of products, and on the other hand it restricts the time resources and capacity available for quality assurance. Moreover, not only developers of development platforms and standard products are faced with this dilemma but also the big community of individual and specific IT solutions developers.

In particular developers of individual software are in a never ending loop – the rapidly developing and heterogeneous environment (hardware, infrastructure, software, data quality etc.) impacts the demand of individual IT solutions substantially. Therefore the IT solutions are subject for further changes and it leads to non-homogenous software that is hard to manage and to distribute.

The proposed way is to develop a „smart" software that like human beings would be able to deal with unknown environment and adequately react on unexpected events. Though development of smart software can take additional resources, it will pay off in phases of software maintenance and operation. Authors propose such software development that includes both the base functionality of information system

J. Münch and P. Abrahamsson (Eds.): PROFES 2007, LNCS 4589, pp. 262–272, 2007.

and additional features (services) for a better maintaining of the software. These additional features, like scaffolding built in the construction process of a building, are created in the process of software design and implementing. But unlike the building process the "scaffolding" of an information system is never taken down, it stays in the information system for it's whole life time.

Conditionally, features for smart technology compatible software can be divided into two big groups as follows:

1. External stability, when SW avoids performance incidents caused by impacts of other system components, for instance, SW operation environments, checking, security monitoring, availability monitoring and processing of other external events
2. Internal stability, when SW avoids or at least limits impact of internal faults, for instance, control of SW performance correctness in production (self-tests of core functionality), automatic download of new versions and converting of collected data and others.

The targets set in the *self-adaptive software* researches [1, 2] partially overlap with the statements stated by the smart technologies. Self-adaptive SW researchers are focusing on SW ability to adapt to implementation environment and external conditions thus allowing even code translation to the targeted environment and defining rules for SW reaction on external events. This report sets different targets: trouble-shooting SW exploitation failures by applying automatic indication of possible failures and reporting them to staff. Implementation of this approach is more beneficial and convenient to use in practice.

Hereinafter the term of software being compatible with principles of smart technology will be denoted as STSW.

2 Software Life Cycle Models and STSW

Since costs of hardware and infrastructure necessary for creating and operation of IT solutions are decreasing rapidly, the main part of producing and maintaining costs are those for software. Unfortunately the producing of high quality software up to users' requirements, secure and easy to maintain is still just a dream of software developers. The real problem is the lack of productive communication between customers and developers. The customers being not well-informed about specifics of IT are not able to formulate the system requirements clearly and precisely enough in order to use them in the development of an application. There is no common communication "language" understandable for both specialists and non-IT specialists that could serve as a precise medium for defining the system requirements. The quite popular among IT specialists Unified Modelling Language UML [3], is only very rarely suitable in an environment being not very familiar with IT terminology.