ON THE MODELS OF DEVELOPMENT AND DISTRIBUTION OF SOFTWARE

Neli Maneva, Krassimir Manev

Abstract: The notion model of development and distribution of software (MDDS) is introduced and its role for the efficiency of the software products is stressed. Two classical MDDS are presented and some attempts to adapt them to the contemporary trends in web-based software design are described. Advantages and shortcomings of the obtained models are outlined. In conclusion the desired features of a better MDDS for web-based solutions are given.

Keywords: models of development and distribution of software, efficiency, web-based solutions.

ACM Classification Keywords: D.2.7 Distribution, Maintenance, and Enhancement; D.2.9 Management.

Introduction

It is a fact that the software industry is one of the most successful and the volume of sales of software is growing each year. According to a study of *Forester Research* [CIO-3, 2007] the volume of sales of corporative software only for 2006 is up to 50 billion USD. The efficiency of the software solutions and especially of the business software solutions is one of the most important challenges for the software industry. Some statistics show (see for example [CIO-11,2006]) that about 50% of the installed Enterprise Resource Planning systems (the leading class of corporative software products) in companies from USA and Western Europe are inefficient. Such negative trends are crucial and have to be analyzed in order to identify the reasons for them. In this paper we will try to introduce a notion that will permit us to analyze the problem.

Let us denote with the notion *model of development and distribution of software* (MDDS) the system of principles, rules, and especially ***relations*** established between a developer and the potential users during the whole lifecycle of a software product. It will happen that, beside the other usual characteristics of a specific software product – its functionality, usability, easy maintenance, etc. – the used MDDS could be crucial for the efficiency of the product. This is especially valid for software developed for small and medium companies with limited resources.

In second section two classical MDDS – *On Purpose* and *On Demand* – are discussed. An attempt is made to list some important characteristics of these MDDS and to stress their role for the efficiency of the software product. Third section outlines the requirements to the MDDS that are generated by the development of the contemporary web-based software solutions and how these requirements could be satisfied on model level. Some conclusions and recommendations for features of an MDDS adequate to the development of web-based software solutions are given in forth section.

Model of development and distribution of software

Let us consider a simplified *development and distribution of software process* (DDSP) with only two kinds of participants – *developers* and *users*. Since the beginning of the software production two classical MDDS were applied – *On Purpose* and *On Demand*.

On Purpose model. When a developer identifies a task/need that is concerning many users and that could be solved/satisfied by a software product then he creates such product and put it on the market. The potential users find on the market one or more such products, chose one of them and use it to solve their tasks or satisfy their needs. We will call such model of development and distribution of software *On Purpose* (OP) and the corresponding software products – *OP-products*. For example, as OP-products essentially operating systems, software development environments, DBMS and other *system software* are developed and distributed.

Being dedicated to serve many potential users OP-products have relatively low price. Due to the same reason OP- products are trying to cover as many as possible aspects of the solved problem or satisfied need and sometime they include too much functionalities. Any way, it is possible that functionality, specific for some user, is missing in an OP-product. Terms of delivery of these products are very short because they are complete and ready for distribution.

On Demand model. When a user identifies a task/need that is concerning him and on the market there is no software product able to solve/satisfy task/need then the user finds developers that are able to create such product, then chose one of them and assign the development of the product to him. We will call such model of development and distribution of software *On Demand* (OD) and the corresponding software products – *OD-products*. For example, as OD-products are developed and distributed different kind of *applications* for end users – information and resources management systems, etc.

Terms of delivery of such products are usually long and sometime unpredictable. Being dedicated to serve a single user the OD-product have relatively high price. Due to the same reason OD-products cover almost precisely the needs of the user. Anyway, it is still possible that, during the process of usage, some specific for the user functionality is happening to miss in the OD-product.

In both cases it is coming a moment when the user of OP-product has to buy a new version and, respectively, the user of OD-product has to demand redesign of the product. Having in mind the trend of shortening the life cycle of the software products, this is leading to additional expenses for users. The developers could try to escape the stressed negatives by some build-in components for *tuning* of the product. Tuning could be very easy – giving values to set of parameters of the product, as well as very difficult – writing a code in some (inner for the product or standard) programming language. Successful tuning of a software product will depend too much on the *qualification* of the user.

These two classical models are extreme cases of MDDS and it is possible to define different middle cases – more close to one or the other of the extreme cases. But they are demonstrating how important is to establish mutually beneficial relationship between developers and users. For the users it is very important to obtain the product that meets its requirement – especially the functionality and performance, under objectively given constrains – price, term of delivery, level of complexity of usage and tuning etc. Developers, from the other side, have to be able to organize efficient development process in order to make profit by supplying products that meet requirements of users. Contemporary MDDS has to be able to help for establishing such mutually beneficial relationship.

MDDS for web-based software solutions

The age of web-based software solutions raises some new objectives to the MDDS making the both classical models inappropriate. Instead of a single large task/need to be solved/satisfied by the product, there is a diversity of smaller tasks/needs dynamically changed over the time. For each user there is a subset, also dynamically changed, of such smaller tasks/needs. So it is no more necessary that a single software product to solve all such tasks or satisfy all such needs. It is quite more efficient if the product is able to supply the corresponding solution as *content* or to satisfy corresponding need as a *service* through the (local or global) net. Something more, web application users ask for services and contents *highly tailored* to their particular contexts of usage. It is no more possible for a single software product to meet all requirements of the user.

The contemporary software *solution* is rather a set of interacting *applications* (some of them web-based, some of them working locally). Each application could be purchased as OP-product or as OD-product. It is not so difficult to identify some typical applications that will be necessary for any web-based software solution (i.e. web-server and web-client, DB-server etc.). Such applications are developed and distributed as OP-products and are available on the market. The other applications in the solution have to reflect the particularities (business logic) of the user and it is more natural to be developed as OD-products. The challenge of building integrated web-solutions is to ensure the interoperability of the applications.

The answer of big developers to this challenge is to organize development of as many as possible of the necessary applications around some general concept or approach, called *technology* (i.e. Java-technology of SUN, or .NET-technology of Microsoft). They also have to provide some tools (programming languages, IDE, API, etc.) to support integration of the solution. There is no standard definition what a “technology” really is and what has to comprise, so the products of different developers could be quite different by implemented functionality. Small developers who accept the technology (*technology-partners*) could develop some applications that are missing and even to develop better versions of some existing applications. We will call such model *Technology-based MDDS*.

Technology-based model. The user is choosing one of the possible technologies, buy the necessary applications that fit the technology and that are available (OP-products) on the market. Then the user searches a developer which is able to create the missing parts of the solution as OD-products and to integrate them in the solution. In most of the cases such developers are the corresponding technology-partners.

Technology-based MDDS has some positive aspects – the technology is usually supported by a stable company and large set of partners who are ameliorating the technology, applications and instrumental tools. But some negative aspects could be outlined, too. The most negative aspect of such model is the *dependence* of the user on the technology. For example, any attempt of the user to extend the solution with new functionality has to be implemented "within" the technology. Issuing of new generation of elements of the technology could lead to necessity of total upgrading of all bought to the moment elements. And more, when an element of the technology is of low quality, comparing with the concurrent products with same purpose, it is very difficult to eliminate this element from the solution and to replace it with a better one.

Minimalistic model. The Technology-based model is appropriate for big users that have resources to keep their solutions up to date with the technology. Small users usually prefer more cheap solutions. For example, the famous combination Appache web-server + PHP + MySQL DB-server is applied in many web-solutions of small users. Such solutions are very cheap because the three products are free and all expenses of the user are for integrating the solution. We could accept such form of development of web-solutions as a model and to call it *Minimalistic*. It is clear that the Minimalistic model has much more negatives then positive aspects – for example, minimal support from the developers of the free applications, missing of enough instrumental tools, very low level of performance and security etc. Instead of the dependence of the technology small users are caught by the dependence of the limited resources.

The Minimalistic model has its place on the market of web-based software solutions. It is acceptable for some non-commercial solutions – private sites, sites of non-profit societies etc. where the goal is to inform interested people. But it is not appropriate for business companies, where on the software solutions lay much more responsibilities than just to provide necessary information.

Since 2000/2001 an idea is circulating, know as Software as a Service (SaaS) associated with the paper of Tim O'Reilly [O'Reilly, 2004]. It originates from the popular conception "open source" and in principle is an apology, on an abstract level, of a dream for total standardization and interoperability of all software products. The goal is to give to the users freedom to choose any component of their solutions on the base of needs and available resources, i.e. to reduce as much as possible the dependence of the user on technologies and products.

SaaS model. Recently there are some attempts to turn the idea into realistic MDDS – *Software as a service model*. Unfortunately, as it is shown in the description of SaaS, given in Wikipedia, the attractive idea was transformed in non-attractive model. The essence of the model is that the developer (of software products) in simplified bipolar DDSP is replaced by a *provider of services*. If the user need some service (for example, to keep data and search them) than the provider is organizing (on his own servers) the corresponding DB and give to the user the necessary web-based interface in order to input, update and search the necessary data. If the user would like to publish a site in Internet then the provider is developing the site and is hosting it. Maintenance of the content of the site is made also by provider on the base of data supplied by the user.

Positive element of the SaaS model is the possibility the user to solve a task or to satisfy a need without buying the corresponding software product but use it as a service. For tasks or needs that are arising rarely such form could be the most efficient. The negatives of SaaS model, formulated as it is mentioned above, are obvious. It is clear what will happen if the user is loosing for some period of time the Internet connection with provider or provider announce that he lost some of the user's data. The dependency of the technology or of the limited resources is replaced by the dependence of provider.

Conclusion

As a result of this overview we have to conclude that especially for the medium and small users of business web-oriented software solution there is no established and appropriate model for development and distribution of software. This leads to big inefficiency of the implemented solutions – for relatively large expenses made by the users they do not obtain the products that are able to solve their tasks or to satisfy their needs in an adequate way.

It is not easy to establish and reason a new MDDS that is adequate of the needs of the medium and small clients but it is worth to invest some efforts in it. Taking into account the discussion above, we could define the following features of such model:

- It has to guarantee the independence of the user from the technologies, i.e. the user have to be free to chose for each component of the solution the existing technology that is the best for this component;
- It has to guarantee that the user will obtain a service with a quality that is relevant to the paid cost;
- It will be very good if the model has the tolerance for the qualification of the user and to allow, etc.

In order to prove feasibility of such model a lot of programming and experimental work should be done. Each feature of the model has to be examined on sample applications. For example, implementing of efficient and fast interoperability of technologically non-homogenous components is a great challenge.

Bibliography

[CIO-11,2006] Editorial, CIO, No. 11, 2006, Internet version. http://www.cio.bg/?call=USE~home;&page=paper&n=1148.

[CIO-3,2007] Editorial, CIO, No. 3, 2007, Internet version. http://www.cio.bg/?call=USE~home;&page=paper&n=1391&pn=2

[O'Reilly,2004] http://tim.oreilly.com/articles/paradigmshift_0504.html

Authors' Information

Neli Maneva – Senior Research Fellow; Institute of Mathematics and Informatics, BAS, Acad. G.Bonthev St., bl.8, Sofia-1113, Bulgaria; e-mail: neman@gbg.bg

Krassimir Manev – Associated Professor, Faulty of Mathematics and Informatics, Sofia University, 5 J. Bourchier str, Sofia-1164, Bulgaria; e-mail: manev@fmi.uni-sofia.bg

THE LOCALIZATION PROCEDURES OF THE VECTOR OF WEIGHTING COEFFICIENTS FOR FUZZY MODELS OF CHOICE

Elena Prisyazhnyuk

Abstract: The author analyzes the localization procedures of the vector of weighting coefficients which are based on presenting the function of value by additive reduction adapted to fuzzy models of choice

Keywords: fuzzy sets, coefficients of value.

Introduction

Different kinds of uncertainty are to some extent characteristics of practically any situation of making decision in which the expert information is used. The result of the research [Ларичев, 2002; Борисов, 1989] show that the main difficulty is caused by the necessity to appraise numerical values of the objects (variants, criteria) or to give numerical evaluation on the ratio scale between them. It is known that verbal definitions usage allow to define the preferences of a person who makes decisions (PMD) more steadily. This approach seemed to be even more justified because in prevailing number of the cases it is enough to have the approximated characteristic of the data set and the expert info usage does not demand high accuracy.

The experts' evaluation in fuzzy models of choice is described by the functions of belonging to the fuzzy set. The functions of belonging can be interpreted differently: as "subjective probability", expert's confidence degree in object's belonging to the concept described by fuzzy set, the opportunity of its interpretation by this concept and so on [Борисов, 1989]. The choice is characterized by preference relation R the meaning of which in fuzzy models consists of the fact that it can point out for every two objects: