UDK 004:37

PSYCHOLOGY OF RAPID SOFTWARE DEVELOPMENT – 2010

Kondratyev Y.
TTG (independent workgroup)

The article briefs the experience in developing software systems in several areas (physics, trading, telecom) in 1999-2010. The systems are developed by workgroups for private customers and small enterprises. This segment of the market offers young developers to fulfill their individual propensities out of any corporate control. From other side, the article highlights the necessity of preserving and reproducing the CIS IT community, which, traditionally, is on the very high professional level.

Keywords: psychology, RAD, CIS, freelance

Nowadays, it's easy to notice the multitude of technical means and organizational processes, as in the field of information technology (IT), so in all national economy. Free publication of and momentary transferring the formalized knowledge by means of the network exposes it in full or almost full volume to the attention of every specialist and student. Today, in comparison to the end of 20th century, this situation looks already problematic: abundance of forms requires making choices, from basic data collection, familiarizing with it, filtering, to deciding what concrete schemes and tools to learn and use. In the IT, means, offered for each particular purpose, sometimes count to hundreds (algorithms, ERP and ERP-like systems, text editors, compilers), with non-severe and non-monotonic, but very annoying interdependence between quality, terms of use and the cost.

Serious projects take months and years of life. This influences so that practices, employed by developers, form individual structure and character of the professional experience and, at once, intellectual potential of the whole community.

It's not a secret that there is a very high professional level of IT specialists in CIS countries, so the western companies are interested (not always consciously) in preserving and reproducing the community in the vista of decades. Gaining this requires comprehending causes. Here, we benefit from analyzing the development occurring in CIS both geographically and in the aspect of initiative (local customers, beginnings of creatively oriented programmers).

This article is a cross section of the experience in developing and supporting the systems in several areas (physics, trading, telecom) within the period from 1999 till 2010, with about half million code lines total. Particular systems (between 10..200K lines) are developed by workgroups for private customers and small enterprises. This segment of the market attracts many young developers as the opportunity to fulfill their individual propensities without interference from the side of corporate control. Today, in the light of recent crisis phenomena, it is difficult to forecast the future of the market, but it's necessary to note that private orders segment has excellent vistas of growing and employment there will become significant factor influencing the higher educational establishments policy concerning graduation course programs. The reason is simple: many users turn their attention to the possibility of creating the applications for their specific needs with factual gaining all rights on the product when paying for time spent on the development, instead of buying packaged software. The latter, in spite of prevalence and low cost, in the majority of cases loses in functionality (plus 70-95% unnecessary functions minus several missing special functions).

The traditional scheme of education implies that young specialists apply their knowledge (e.g. design patterns, organizational patterns) to solving customer's tasks. Regrettably, factual situation shows unpleasant nuances, for example:

1. Only abstract concepts are universal (OO concepts, algorithms etc.). "Universal" programming languages are such only conditionally, being limited by traditions of applying and technical platforms. Other technical means quickly stale and/or are out of the scope of worker's specialization, thus studying them benefits only indirectly.

2. Many organizational practices are developed by western companies, in the context of large projects, usually remaining incorrectly understood and not digested by specialists in CIS. This benefits also indirectly, because that the real application is possible only for practices, adequate to organization in which you are employed.

***

Rapid development is the answer of private developer to the needs of private user. This implies the following specific aims:

1. To learn forecasting customer's evolution in relation to the system, which is under development. (At the beginning of consultations, even professional customer usually has only some vague idea on what should the new system provide.)

Having generally more fluent intellect plus experience in IT, the developer can model customer's thinking and develop the system aiming at the needs, which are possible to emerge on the second and the next iterations.

This approach is the most advantageous in long-term relations, but demonstrates unpleasant estimates of the time and cost in cases when the customer is not sure in the project's future or consciously (sometimes erroneously!) refuses to evolve it.

2. To learn seeing the system from user's point of view and interest, in the vista of many years.

Good system works during years and decades. The response time of the system and typical sequences of user actions should be gradually optimized (or tend to be optimized) so that the time of performing them by the user was less than the threshold of the nervous system reacting to visual and phonic stimuli.

A negative example: some prevalent schemes of user interface design (including web interfaces) allow the same functional element to appear on the screen and move along the hardly predictable path in response to simple mouse movements and program's internal events.

Another example: self-stimulated program activity and the lack of settings, both introduced on the behalf of typical, "illiterate" user. Time loss by professional users on the manual counteracting those factors, very possibly, exceeds the profit from automating and refusing to teach unprofessional users. Regrettably, this factor acts above concrete programs, computers or communities, and it's hard to estimate it objectively.

Nevertheless, it is possible to try another direction: to stimulate involvedness and growth of professionalism of capable users, at the same time barring others from computers as such. With working enterprise systems, this occurs, as a rule, when deploying the new system having qualitative advantages in the workflow and user interface model in comparison to the previous.

3. Intellectual growth of the developer, deepening organizational differences between the talented and the lazy.

In general, this process is similar to the one described above for users of the system, excluding only the IT tradition of appreciating the good developer more than the clever and literate user.

Organizationally, giving more rights to capable specialists would help raising the quality of intellect in the whole IT community, with the single proviso: any offers concerning "rewriting all from scratch" must be considered with help of third party experts.

4. Controlling the attention, concentrating on the main, elaborating the approach to life as whole.

Psychological health of the society does not depend directly from the means of economy. Nonetheless, capabilities, intentionally developed while working in the IT, the human may transfer to any other activity, where the bases are logic, attention and control.

Remark. Today, de facto, the practice of rapid development is, first of all, the subject of conscious personal choice. And only as the second it is the culture, with elements of which the students familiarize while preparing to work in IT. An example: the top rated university student initially regards the enterprise tasks exactly as a regular training, i.e. sees only the aspect of gaining functionality, but not the aspect of shaping the architecture apt of evolving and not the aspect of

project supportability. For him, taking these aspects into account is equal to slowing down, but not speeding up, the development. For experienced company workers, it would be very useful if the higher schools tended to pre-teach their students the correct (practical!) point of view on the above. A curiousity is possible when the academic should pass some relevant development practice within the staff of a company.

Remark 2. Subset of methods, tools, and means of their use the developer should choose according with personal capabilities and character. Striking difference in human capabilities for IT, qualitative and quantitative, is the well known fact. In the context of rapid development, this means that less capable specialists will be to some extent constrained in choice of means on the behalf of the more capable. In unsuccessful cases, this influences negatively on the spirit of collaboration and the atmosphere of work.

***

Approaches to rapid development are many, in different contexts, so it's only possible to give an example (not a template) of their constituents.

1. Individual selection of practices and means for each particular project.

Languages, environments etc. learned before must not limit studying the new means. The developer himself should get rid of the barrier to choosing an mastering the means. For 21st century with its overabundant spectrum of the means, this practice should be regarded as norm.

Potential problems:

1.1. Copyrights on the commercial means, low quality of the free means.

Possibly, the following sounds like heresy, but: stimulate using the unauthorized program copies by the young people in CIS! The best to do for growing really capable is to look into everything from within.

The above problem is purely political, it roots out the potential of talented young people in civilized countries. To be honest, we all know very well that license is the empty conventionality from the point of view of gaining knowledge while exploiting and internally examining the best software products.

1.2. The typical modern customer, as a rule, refuses any means with which he is unfamiliar. To re-educate him is awfully hard until the control over project development strategy passes to the developer. This occurs a) at the very beginning of projects, when choosing the means may be limited by the customer according to his narrow scope b) when assuming the responsibility for projects with relatively large history.

2. Creating the base means, like an alphabet, for constructing the project and its development process. Optimal alphabet causes increasing work tempo and quality of the result.

As positive examples, we can note DSL (domain-specific languages) and code generators. Another example, on the more concrete level – optimizing the storage of Excel books in the SVN by means of saving them in XML-format, allowing to explicitly express deltas between versions.

3. Refusal of prototyping. The good system exactly equals to the single model provided that the internal dimensionality of the model is high enough.

With such an approach, the possibility to show the system to the user appears on the latest stages of the development, but the quality and transparency of the system make it more safe and economic in use, in the sense of cost, work expenses and required volume of attention.

Briefly: until you have an idea, no prototype can help.

4. Stimulating users to master slightly more complex conceptual model of the system, aiming at evolving the culture of IT among the non-specialists.

Remark. The notion on the beauty is the essential as for internal system architecture, so for external aspects. See details below.

Remark 2. CIS countries suffer from total computerizedness of the enterprises with workers using computers inadequately to their functions spectrum. There is obviously too much equipment in comparison with the capability of the nation to master the IT. Many young people thus have bad experience because otherwise they could spend their efforts on something unrelated to computers.

It is known that under such conditions the computer work time is used for personal needs, provoking non-productive conflicts within the triangle workers-sysadmins-management.

5. The developer must see into the principle, and, to some extent, into the structure of the 3rd party tools. Ignoring this principle leads to emergency-prone system, requiring increased attention from all sides and serving as a bad example for young developers dealing with the support.

An example: analysis of the 90K lines project, the system for wholesale trade accounting.

The system belongs to the third generation in the line of resource management and accounting systems/practices, developed in one of Dnepropetrovsk manufactures. It includes means for direct reflection of primary operations (about 10 types of operations plus causal links), object DBMS, and hybrid analytical model, based on models used by two companies of customer's concern.

Remark. The first generation was just a workflow + Excel books, the second was the partially optimized workflow + accounting model + books, redesigned to support the model.

«Methodology and value reference points of the xda project

(a posteriori summary)

4.5.2008

[*Method/task – target object*]

WYSIWYG – user interface

Implicitly, the low culture of our users forces developing systems with interface as close as possible to the framework of visual forms and action sequences they use on the daily basis. The user does not want to learn any new tools and concepts. The best for us in this case is to integrate the program into user's environment to the maximal technically possible extent (in part., this leads to integrating with Excel), and show him the means of controlling the necessary functions in the form, which is as close as possible (at the limit – identical) to his views on functions, imaginative associations (if any) and priorities. The user expresses this very simply: "the simpler – the better", but fails to concretize what does this mean.

Evolutionary automation – required functionality

The high-level task of the accounting automation lies in transferring into technical form those intellectual processes of users, which are already templatized and thus require mechanical work instead of solving informal tasks on the higher level. The user will only have to set the initial conditions (e.g. input the primary data), start the system, and get the results in the form, which can be easily understood and used by people who are not familiar with the system. Evolution here means that any repeated action we make subconscious, the user only initiates the process. Using the program in its normal mode of operation, the users get onto another level (composition) of tasks and problems, master it, and after, a new turn of evolution becomes necessary. N.B. This describes the well-known ideal model, omitting difficulties, connected with template processes, which users are not aware of or do not wish to expose because of political reasons.

As fast as possible – all workflows

To develop the program, to enter data, to get analytical results – ASAP. Cross section 1. There is a subtlety: assigning priorities to processes under optimization ferociously conflicts when we look from different sides (ideal, real customer; developer). Direct modeling the system of priorities of the customer would take 20 years instead of two. Further, modeling priorities from the business point of view requires radical reengineering of that business itself, which is absolutely impossible in our case. Simply, no good compromise at all. So, in our case we worked on the behalf of ourselves, experimenting with practices of design and programming etc. in such a manner that allows to rise ASAP the professional level of the developers. The project contains many concepts, which are well known and widely represented on the software market (e.g. containers, OLAP, OODB, packaged accounting systems). Provided with the due support from customer, we probably would get the result yet faster. Cross section 2. "Faster" in application to development means automating code creation on the basis of high-level specifications, automating project build and function testing. Cross section 3. During the design, we found and used the "matrix approach",

loosing some principles of OO design. For example, many structures are not encapsulated, for the following reasons: speeding up the development, technical complexity of implementing it in full in the existing context, excluding the loss in system performance. All excluded OO rules have been moved to function and structure specifications. It is implied that the new developer familiarizes himself with all specifications to master several main structures. We've used the approach with a few complex structures instead of many simple ones. Our structures must be comprehended, but, provided it has succeeded, upgrading or evolving the system will also succeed. The system consists of the shortest logical links. This constitutes the matrix approach – tending to shape the multi-dimensional and multi-sense entity for shortening routes.

Beautiful – external and internal look of the program and information it produces

This is just a postulate, necessity of architecture through the whole question. The art of architecture = beauty + functionality + technologies. As the kind of art, architecture is the part of spiritual culture of humanity. Aiming at creating something having all three qualities – is our attempt to raise both our personal culture and culture of the users. From my personal point of view, this motive must be present in any system ever created. Regrettably, the common American model of thinking ignores beauty and it's hard to get rid of feeling that to **satisfy** the customer, we could proceed much faster if threw away the beauty (and, as consequence, the architecture), limiting ourselves with the pair "functionality + technologies". N.B. Initially, the above definition belongs to the classical (constructional) architecture, but may be easily applied to any kind of engineering, software development, social systems, within which we live our life. N.B. 2. In the relatively poor information aesthetics of the customer we noticed a good element, and it had been taken into account everywhere where possible: compact data views. "All that is really essential always fits onto a single page".

Reuse – general-purpose software components

Here, nothing to comment, this thesis is obvious for any professionals. We have had pleasure to yield several high-quality general-purpose libraries as part of the output of the development process.»

An example: practical research. Experimental news aggregator.

We often forget that any project starts from the moment when the customer becomes aware of his own need in a software system. Next, the search-choosing-filtering phase starts. And only next is the development itself.

The value of search-choosing phase is well seen on the freelance market, where any person may look at hundreds of projects, waiting for their performers during weeks, sometimes months. Moreover, researching the freelance segment as the whole exposes wonderful facts:

1. Almost all popular technical platforms (GAF, RAC, Elance, Guru) are short-circuited today, being oriented to the direct use by the end user. But, their essence is the same, regardless of forms of information representation and some policy details. This is just an offering a placement for commercial offers concerning the development and services, and ensuring the financial connection between customers and providers.
2. Typical customer of the freelance segment seeks cheap lonely programmers, thus neglecting higher quality services offered by workgroups and companies.
3. Typical provider spends extra attention while searching and analyzing jobs, because platforms are not ready for integrating and individual filtering the news. Also, platform developers fail to see the world with freelancer's eyes and build the really convenient web interface. This fact corresponds to the above theses of this article.

Thematic structure of the jobs stream is interesting for forecasting the needs of IT in skills and skill sets of different kinds on the provider side. In the scope of this and other interests, currently, there is an experimental project, news aggregator, with the add-on specialized for collecting and analyzing freelance job news.

The news contain both tabular and text data. The latter is in free form, so any quantitative analysis requires employing algorithms of classification and parameterization.

Several features of the aggregator:

1. Integrating job news flows from several platforms.

2. Tracking the history of the job, estimating the current actuality, estimating the current relevance (criteria – info about customer, info about job, results of weighing the text based on regular expressions).

3. Filtering the news individually for each user of the system.

The nearest target is boosting the individual selectivity of the system by means of machine learning mechanism. Today, the algorithms classifying the large quantities of news items are under development. The idea lies in combining three methods (hierarchical clustering, MDS (multidimensional scaling), and SOM (self-organizing map)) for selecting the representative items, their quantitative estimation, labeling and positioning the rest of items relatively to the identified classes.

The direct economical effect from the existing functionality both alone and with planned additions – the great (more then 10 times) speeding up the reaction of high-level specialists to the relevant queries of the customers, regardless of any particular platform. The time of the reaction is not currently optimized by the platforms. There are several causes:

1. A good specialist cannot compete with "mayflies" who are engaged in copypasting much more then interested in finding jobs they are fit for. Good specialists are always busy with projects and have no time to look at the raw news lists.

The aggregator eliminates this cause.

2. A good, personalized offer of service requires time for familiarizing with the theme and cannot be replicated. This complexity has principal nature and cannot be avoided technically.

3. Both filtering out the irrelevant applications to the job posts and limiting the activity of low-end workers require non-trivial efforts from the side of platform developers, again implying use of artificial intelligence methods. As negotiations show, the developers are not only not ready for such efforts, but even cannot imagine some natural ways for optimizing the connection of customer's needs with provider's potential. Regarding the latter, a single exception is the oDesk platform, offering the free XML-interface for getting news data in the form of object model.

Resume

Rapid software development is the today's reality for highly intelligent IT-specialists. As a rule, high tempos and quality of the development together are reachable only beyond the hard organizational schemes, because that the latter are themselves subject of the development in each particular project. Talented developers should be encouraged to express their creativity in any form. In the context of higher schools this may imply forming elite groups based on students' abilities.

Aesthetic education and self-education is the practice not less significant in the IT than in constructional architecture.

Students should be made legally irresponsible for any kind of use of the commercial software, because that that use is the key factor of intuitive self-teaching of the talented.

Higher schools should turn their attention to freelance market as a perspective for self-employment of many today's students, and study psychological nuances of the private relations in this segment, aiming at adequate training the young specialists. Concerning the practice, it is necessary to work directly on that market. It must be noted that, for minimizing unproductive time expenses, certain toolset is necessary (the aggregator, described above, may serve as an example).

BIBLIOGRAPHY

1. Frederick P. Brooks. The Mythical Man-Month: Essays on Software Engineering. 1975, 1995.
2. Константин Берлинский. Набор серебряных пуль. Справочник удачных проектных решений при разработке ПО. 2004.
3. Edward Sullivan. Under Pressure and On Time. 2001.

Рецензент: Булат А.В.