INNOVATIVE INFOTECHNOLOGIES
FOR SCIENCE, BUSINESS AND EDUCATION

SMALL IT UTILITIES FOR E-PROJECTS: OVERVIEW OF PARADIGMAS

Robertas Narkevičius [a]

Department of Information Technologies, Vilnius Business College
Kalvarijų str. 125, LT-08221, Vilnius, Lithuania

Received 2 April 2009, accepted 7 September 2009

Abstract. Mass-usage programs are not able to satisfy the needs of small-medium business companies. Many reasons are involved: too complex, not localized etc. Also comprehension and semantics could be formed depending on region tradition. This article represents the new paradigma: decomposition of big programs into small components allows constructing systems for exact demands. It can be used for new type of search engine, which searches not pages, but program components.

Keywords: Databases; Project management; Programming; Internet; Information technologies; E-learning; ICT4T.
Short title: The usage of small IT utilities.

Introduction

The usage of advanced IT utilities plays an important role in the efficient organization of computer-based system. Empiric experience points out to such problems as: common mass programs including internet based utilities very poorly fits needs of small or medium companies. Customer relationship management system *Vtiger CRM* can be mentioned as typical example. Reasons might be formulated in the following way.
1. Satisfaction exact needs, insufficiency of programs' flexibility. Demands to enter data, print reports and statistic information must fit exact, not abstract, situation, country.
2. If program is flexible, then it is difficult to configure. Time and money spent to change system, which is oriented for some common usage activity in many companies and institutions are higher that creation of new programs. Customers may not understand problem ruling over information. So transition from complex system to simple will completely confuse them.
3. Frequent localization problems.
4. It is easier to involve customer into environment of concepts and set of formal descriptions when projecting from the beginning.
5. The problem is how to find the exact means to fit a person who will work with the data and logics.

The solution to the problem requires a special technique. The main communication method represents "eye to eye" or online scheduled program. When a customer becomes a part of the project then he is not just an observer. Such understanding makes it easier to design business applications and to adapt the applications to changing requirements. Client knows what requirements he can rise for the team of programmers and begins to understand further specifics about the development in the future. Direct communication principle allows to create utilities, which not only fits needs, but involves customers in such comprehensions as data validation, unique records, client - server architecture and limits, which are provided by informational technologies and related things. There is no secret that responsible individuals from various institutions do not think how to use IT abilities the way it should be used and because of this, when they see the complexity of mentioned common usage programs, their proposed ideas and projects will never materialize. The real problem is in finding information and not losing it when programs and versions are changing.

This article represents an attempt to view and solve problems using another, different cut.

1. Entity-relationship model

References about *Unified Modeling Language* (UML) are a standardized general-purpose means of modeling language in the

[a]Corresponding author, email: *robertas@kolegija.lt*

field of software engineering.

Vtiger CRM is an open source CRM application that was formed from *Sugar CRM* with the intention of it becoming a fully open source CRM application with comparable functionality to SugarCRM and Salesforce.com. It offers reporting, a customer portal and an Outlook plug-in in its free edition, whereas those functions are placed in paid versions of other CRM applications. UML includes a set of graphical notation techniques to create visual models of software-intensive systems.

Moodle is a free and open source E-learning software platform also known as a *Course Management System*, *Learning Management System*, or *Virtual Learning Environment*. It has a significant user base with 49,256 registered sites, 28,177,443 users in 2,571,855 courses.

According to notation *You need Developers, not Programmers*, management advice can be worthless or even worse if it is not appropriate for your situation. The right decisions for a big company can be fatal for a small one.

Table 1 represents the classification of IT utilities according to the space of usability, whether people will use it widely. Table 2 represents the classification of IT utilities according to the time of usability.

Common mass programs, including internet based utilities, poorly fits the needs for small and medium projects since it is too difficult to involve them into educational institutions. Typical examples are Vtiger CRM (Customer relationship management) system [1], Moodle learning system [2], Openbiblio library system [3] etc.

Big amount of mass programs could be classified according presented scheme.
1. In satisfying specific needs these programs are not so flexible [4]. Demands to enter data, print reports, and statistic information must fit exact, not abstract, situation, country, etc.
2. If a program is flexible, then it is difficult to configure it.
3. Time and money, spent changing the system into a program which is oriented for exact usage, is more expensive than development of new programs.
4. Many do not understand how to handle information in programmes.
5. Frequent localization problems.
6. It is easier to involve customers into environment of concepts, sets of formal descriptions, when projecting from the beginning, from a simple to a complicated system.
7. A problem is to meet the needs of a person who will work with data.

2. The solution to the problem of IT-utilities.

The solution is to create small programs for institutions using local potentials [5]. Set products for customers encompass programs, which will be completely adapted to exact environment and will solve concrete, specific tasks characteristic for such environment. The main communication method is “face-to-face” or online scheduled program. Then customer becomes a part of the project, when he is not just an observer. Such understanding makes it much easier to design E-applications and to adapt the applications to changing requirements. Client knows what requirements he can rise for the team of programmers and begins to understand further specifics about the development in the future. Direct communication principle allows to create utilities, which not only fits needs, but involves customers in such comprehensions as data validation, unique records, client- server architecture and limits, which are provided by informational technologies and related things. There is no secret that responsible individuals from various institutions do not think how to use IT abilities the way it should be used and because of this, when they see the complexity of mentioned common usage programs, their proposed ideas and projects will never materialize.

Some presented figures will help to understand the main ideas of the article.

Table 1. Classification of IT utilities according to the space of usability. How widely people will use it?

Abb.	Object	Destination
S	Small programs	For one (several) teacher
		For one (several) student group
		For one (several) student
M	Medium programs	For teachers of different subjects in one educational institution
		For students of different courses in one educational institution
L	Large programs	For students and teachers from different departments
		For students and teachers of several specialized schools
XL	Extra large programs	For international usage
		For wide usage on a country level (mass usability)

Table 2. Classification of IT utilities according to the time of usability

Abb.	Object	Destination
S	Small programs	For a part of a lessons or one lesson
M	Medium programs	For many lessons of a subject in one semester
L	Large programs	For many lessons of many subjects in few semesters
XL	Extra large programs	For all types of study processes

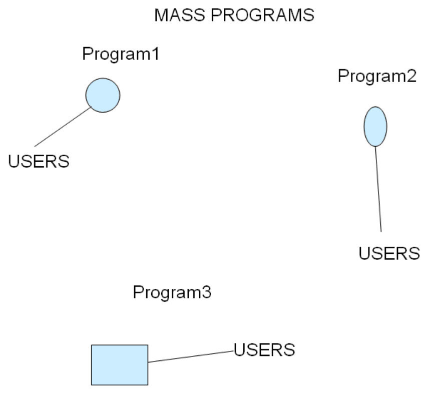

Fig. 1. Three programs belonging to the three user groups. Every program has its own logics of data model.

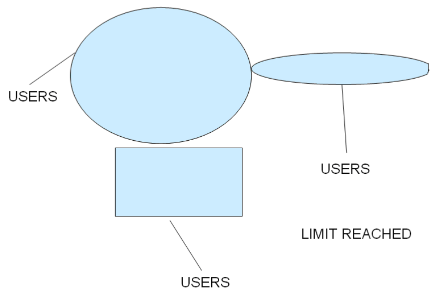

Fig. 2. Due to different concepts is too expensive to make the wide intersection.

Fig. 1 represents three listed programs. For example test program is shown, which has three user groups. Every program has its own logics of data model and, after some time they will be filled with valuable data. Fig. 2 represents the continuation of Fig. 1. After some time they have such data models which could intersect, but because the concepts are too different it is too expensive or impossible to make wide intersection. Every specialist of distributed systems knows how difficult sometimes it is to join them because of the gaps in data integrity.

Fig. 3 represents the continuation of Fig. 2. Intersection of data is impossible. The solution is to waste time and money making standards or remaking systems and it is too expensive. Another issue - errors in data integrity analytics approach which can cause complete failure. There is one more problem: standards will disrupt the birth of new ideas and cases.

Fig. 4 represents the new possibility: another solution can be to cut learning materials, program functionalities in small parts.

Fig. 5 represents the continuation of Fig. 4. Cutted material is unsorted, making chaos - atom environment. In such a case even worse situation will appear. No one will able to filter information so intervention is needed.

Fig. 6 represents continuation of Fig. 5. It can be compared to nowadays torrents or even to artificial intellect. Observation shows an incredibly rapid growth of information in such systems. Why should not we use it in educational purposes?

It is not very difficult to prove that it will involve more people in creation of educational process, because, making "atoms", educational institutions can use their own IT resources. As usual, they can make some small programs, not big. It will be easier for project management.

3. Team-leaders for small projects

What is a team or a person, who helps to materialize projects? Software developer? According to wiki, a software developer is a person or organization concerned with facets of the software development process wider than design and coding, a somewhat broader scope of computer programming or a specialty of project managing including some aspects of software product management. This person may contribute to the overview of the project on the application level rather than component level or individual programming tasks. Software developers are often still guided by lead programmers and also encompass the class of freelance software developers.

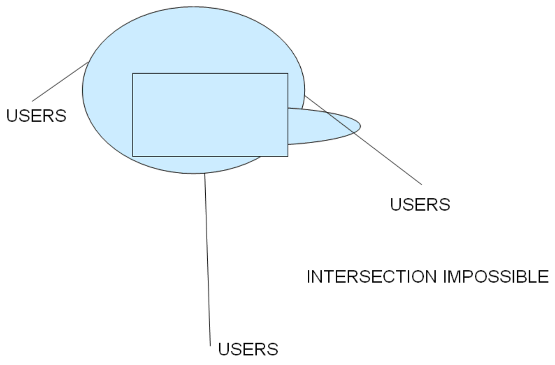

Fig. 3. Continuation of Fig. 2.
Intersection of data is impossible.

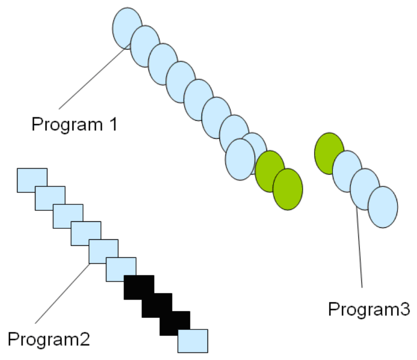

Fig. 4. Another solution.
Cutting of learning materials, program functionalities in small parts....

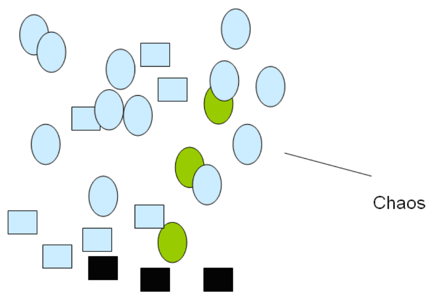

Fig. 5. Chaos - atom environment. In such a case even worse situation will appear. No possibility to filter information. Intervention is needed.

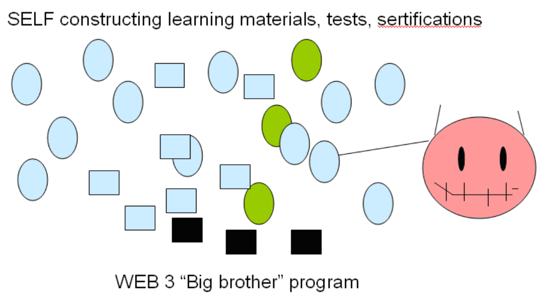
SELF constructing learning materials, tests, sertifications
WEB 3 "Big brother" program

Fig. 6. Incredibly rapid growth of information in such systems.

Management advice [7] can be worthless or worse if not appropriate for your situation. The right decisions for a big company can be fatal for a small one. How to choose a good developer? Who is a developer, communicating, cooperating with small companies? A programmer is a person who can make some sophisticated operations: i) code new features; ii) fix bugs; iii) write specifications; iv) write automated test cases; v) help keeping the automated build system up to date; vi) work out tough problems; vii) prepares documentation; viii) help with testing; ix) of course, can make codes readable.

They will be wanted in small companies. Software developers must be clever and do many things, which are separated in cases of big companies. The software developer is a person or an organization concerned with things of the software development process wider than design and coding, a somewhat broader scope of computer programming or a specialty of project managing including some aspects of software product management. Software developers are often still guided by lead programmers and also encompass the class of freelance software developers. Other names which are often used in the same context are software analysts and software engineers.

In time, differences between system design, software development and programming become more apparent. In current market places there can segregation already be found between programmers and developers; one who actually implements is not one who designs the class structure or hierarchy.

Moreover, developers become systems architects, those who design the multi-leveled architecture or component interactions of a large software system. A *programmer* can be celebrated just for writing a code, but a *developer* could be involved in wider aspects of the software development process as following.

1. Participation in software product definition.
2. Specification.
3. Requirements analysis.
4. Development and refinement of throw-away simulations or prototypes to confirm requirements.
5. Feasibility and cost-benefit analysis, including the choice of application architecture and framework, leading to the budget and schedule for the project.
6. Design.
7. Code implementation.
8. Authoring the documentation needed by users and implementation partners etc.
9. Testing, including defining/supporting acceptance testing and gathering feedback from pre-release testers.
10. Participation in software release and post-release activities, including support for product launch evangelism (e.g. developing demonstrations and/or samples) and competitive analysis for subsequent product build/release cycles.
11. Maintenance.
12. Instead of *programmers* (people that specialize in writing code), what you need is *developers* (people who will contribute in multiple ways to make the product successful).

4. Management specifics for small IT utilities

For small projects, the institution cannot afford having people who think their only responsibility is writing a code. There are many other things to be done, all of which are critical in having a successful product. In cases of big companies they would simply hire more specialists until every job functions are covered. But in institutions with small IT resources versatile people are welcome. In institutions with small IT resources, most IT people are responsible for multiple tasks. It simply is not feasible to have a person who focuses on just one small area. Small companies need versatile people who are willing and capable to step in and do whatever it takes to help the company succeed.

One or few IT administrators handle basically everything. The key concept here is flexibility. Big companies have more specialists. Architects do design. Programmers write codes. Technical Leads manage programmers.

Program Managers are responsible for the specifications and schedule. Product Managers do positioning and message. By the way, these are two very different cultures and unpleasant things may happen when they intersect. Flexibility and boundaries do not mix very well. A person who has been successful in one of these cultures often stumbles badly when is transitioned to the other one. In small companies every developer is first and foremost a programmer. The bulk of their time should be spent writing codes and fixing bugs.

But every developer also needs to be involved in other areas such as: spec documents, configuration management, code reviews, testing, automated tests, documentation, solving tough customer problems. These things are the differences between a programmer and a developer. The developer has a much wider perspective and an ability to see the bigger picture. The programmer writes codes, throws them over the wall to testers and waits for them to log bugs. The developer knows it is better to find fix bugs now, since he just might be the one talking to the customer about it later. The best developer may not be the person who usually gets the really tough problems. A programmer with an amazing talent can be a lousy developer. Coding is a necessary but insufficient part of being a developer. Similarly, less gifted members of one's team can still distinguish themselves as excellent developers through their contributions to the non-coding parts of the product.

Accordingly to Ref. [4], there are several other characteristics for every successful development of the project: i) project is completed on schedule; ii) any changes in the schedule are mutually agreed with the client; iii) project is completed within budget; iv) if the scope of the project changes during the cycle, change orders are initiated and separately billed for; v) future development is clear; vi) code is maintainable.

There are several things that can go wrong on for both small companies and developers side that will hinder the above and cause the project to be less successful.

1. Responsible company person (client) is not responsive or not reachable.
2. A client does not deliver needed material on time.
3. A client constantly changes requirements.
4. Project team fails to meet milestone deadlines.
5. Team leaves project after finishing, no prospects for future development.
6. A client does not understand that he is a part of the project.

Such companies or clients can be classified into two kinds: i) hyperactive - taking care of every small detail of content (according to general rule *a client is very right*; ii) active - taking care of concepts, sense of contents and data logics. It develops more trusted relations than in the previous case.

A passive, which wants to stay out of it and let programmers' team to handle everything (a headache for programmers' team, it is difficult to keep team's motivation). There are several methods one can use to mobilize your client or to communicate with the project manager on a regular basis.

Firstly, a web-based client extranet/project management application should be used. That way the client will have a convenient channel for communicating with the program manager or the rest of the project team and reviewing project progress.

Secondly, for clients that are not computer - friendly, the best solution is regular meetings. A person responsible for those meetings needs to prepare a set of specific questions that you need answering or problems that need to be resolved.

Depending on how big the project is, project manager is needed as a negotiator between developer's and client's company. The project manager carries the most responsibility within the team. He is responsible for setting up a realistic schedule, which should be closely followed in order to assure that a project can be completed in time. A schedule is essentially a collection of major milestones with their deadlines. Milestones are usually associated with deliverables. The completion of certain milestones, like content delivery, is the responsibility of the client. Therefore, the client needs to be accountable for any milestones that are not completed by them on time.

Why is the delay of the project schedule a major problem? General problem arises because the team usually works on several projects at once on both sides [8]. Therefore, time of various team members needs to be scheduled for particular tasks at a particular time so that all the deadlines can be met. By delaying a milestone within a project with a delinquent client, the schedule of the team member who is responsible for the consecutive milestones is completely disorganize.

The so called scope creep is the most financially unpleasant problem that development can experience. Scope creep occurs when the project requested functionality exceeds the initial requirements, which have been quoted in the proposal or project plan. Changes within the project requirements are often justified during the development process. It is essential, however, to initiate a change order for any major extension of the requirements.

A change order is an amendment of the original development contract with the client, which introduces new costs for all the added requirements implementation.

Frequent changes in the requirements can be avoided by: i) writing a project plan based on a thorough survey of the client's needs before initiating the development process; ii) collaborating with the client on a building of website blueprints that addressee site structure, content structure and navigational paths of the website.

If you have an experienced project team and there are no problems with the completion of milestones by a client, but you're still failing to meet project milestones, you must have a scheduling problem. The project manager must consult the project team's calendar before setting up a schedule for a project. It is essential to determine when a particular team member has time to work on a given task and setup the milestone deadlines for new projects according to that.

Client needs: (to place cars rightly, monitor empty places)
⇓
Client‘s understanding about model concepts (data integrity), programming concepts
⇓
Programmer‘s understanding about task environment (logistics structure in company)
⇓
Effective program (system, reachable from internet)

Fig. 7. Abstract distance from client needs to successful program

A well thought schedule will be respected and used by the team as their main guideline for what should be done on a given day. Project management programs enable to setup milestones and assign them to team members, as well as keep track of their completion date/time. Best choice is to use the web based project management programs. Then client will be able to follow the schedule.

Another problem is financial, when educational institutions cannot pay or do not want to pay. On the other hand, they will not waste time adapting complicated systems because of the unknown concepts and appropriate psychological moments. So IT level between large companies, which can maintain their own IT sector to solve various IT tasks and institutions with small-medium IT resources, will increase.

Seeing this motivation, we cannot exclude common solution utilities for many institutions, when some exact, strict standards exist in a level of region, country or a big institution. As examples could be accountant programs.

Conclusions

Small IT utilities can be an efficient way to distribute information in the internet. Method to spread information is needed to be analyzed.

It is possible to search more effective methods to keep and spread information, moving out from standard comprehensions, supported by IT situation, governments and so on.

Acknowledgement

This work was prepared as a part of ICT4T project [9] (2006-2008).

References

1. Vtiger CRM (Customer relationship management) system [http://forums.vtiger.com/], retrieved 2009 09 01.
2. MOODLE [http://download.moodle.org/], retrieved 2009 09 01.
3. Openbiblio library system [http://obiblio.sourceforge.net], retrieved 2009 09 01.
4. You need Developers, not Programmers [http://www.ericsink.com/No_Programmers.html], retrieved 2009 09 01.
5. Optimistic scenarios of Lithuania future. For lith. readers - Kubilius A. Lietuvos ateities scenarijus 2010-2020 m. Optimistinis variantas. – Vilnius, 2003. – P. 19- 21. - [http://politika.osf.lt/Kiti/scenarijai/santraukos/Kubiliaus_santrauka.htm], retrieved 2009 09 01. [http://politika.osf.lt/Kiti/scenarijai/santraukos/Jasiuleviciaus_santrauka.htm], retrieved 2009 09 01.
6. Entity-relationship model [http://en.wikipedia.org/wiki/Entity-relationship_model], retrieved 2009 09 01.
7. Unified Modeling Language [http://en.wikipedia.org/wiki/Unified_Modeling_Language], retrieved 2009 09 01.
8. Baronas R. Duomenų bazių sistemos (in lith.) 2002.
9. EIT project [http://www.english-it.eu/], retrieved 2009 09 04.