DEVELOPING A CONTROL SYSTEM FROM A DIVAN BED

G. Milcinski*, M. Centrih, J. Dovc, M. Plesko, G. Tkacik, Cosylab, Ljubljana, Slovenia

Abstract

A dream thought to be made possible by Internet – to develop at home, to manage projects from anywhere in the world - should become a reality with high speed connections and new "productivity" tools. Did it work for you or do you find yourself clogged down with hundreds of unread e-mails and the ones read sorted (seemingly not enough) into several tens of folders on the basis of severity, projects or people? Our experience shows that e-mail collaboration is ineffective with a group of five people or more: tasks are left undone and messages are lost. Two solutions are possible: either to use a commercial package for project management, which always has "that crucial feature missing", or to adapt an open source application for your needs. We opted for the later and modified Request Tracker [1]. The article is about this tool and our adaptations, which integrate it with automated build procedures, web page generation and other e-management tools all backed by an SQL database. We present a sample project flow from receiving an order to delivering a product. The system works with a group of students and their professional results satisfy the most demanding customers coming from Europe, USA and Japan; therefore we believe it to work for a group of responsible programmers as well.

1 INTRODUCTION

There are two possibilities to make a living with programming – selling software licenses or getting paid for individual projects. For the first option a company of three is enough (director, secretary and a plant), but the second one demands a cooperation of many people – at least a (usually very demanding) customer and programmers themselves; nine times out of ten a presence of a project manager is unavoidable but sometimes there is also necessity of customer relations manager (CRM), middlemen and other people. All of them must know exactly what they want and have to do. They have to know also, what the other members are doing. Here we come to the first decision – who needs to know what? To keep the customer satisfied, he has to have detailed insight in a progress of the project, but some details have to be hidden from him (there is no need for his headaches all the times something goes wrong); the programmers would also drown themselves while reading a bunch of emails. A project manager probably knows most but definitely not everything. So how can we achieve sufficient flow of information between project members? More often than not there is a network of dependencies in the project, as one part is a precondition for the other. Surprisingly, many programming companies (especially the smaller ones) still solve the problem by simply placing all programmers in one room. Anytime you need someone, you can call him instantly. Is this really productive? These interruptions often distract the people from their normal duties. Moreover, although the communication is as direct as it can be, it does have its drawbacks: the knowledge is transmitted but not shared with other people; solutions are found but also reinvented time and time again. And on top of everything, small talk about your neighbour's wife soon dominates the discussion.

One of our programmers said once: "it is funny to work at home – you work longer, but you are still spending more time with your family." Working at home appeared to be effective enough that a small group of students of physics, computer science and electronics students developed a complete control system for the ANKA accelerator in Germany. The atmosphere was apparently sufficiently stimulating – the majority of this group formed the company Cosylab [2], which is now cooperating in international large-scale projects.

It is obvious, that this type of working demands professional methods and project management tools. This article is therefore a fairy tale about a divan bed, a plant, the net and a person behind a computer.

*grega.milcinski@cosylab.com

2 A FAIRY TALE (OR A TRUE STORY?)

Behind nine mountains and somewhat less than nine oceans, but certainly not such a long time ago, a contract of cooperation with some distant company of non-English speaking country has been signed. The project management began.

2.1 Mailing Lists

Two mailing lists have been activated. The first one is a so-called external mailing list, the participants of which were the customer, CRM, project manager, interpreter and all important project members. It is used to keep customer up-to-date on the project progress and to consult with him on project details. The other mailing list is internal – recipients are project manager, all project member and also the interpreter. A content of this list is more tactical - discussions about how to handle this or that specific problem. A mailing list with exactly defined senders and receivers and with archived conversations is much better than a simple e-mail. By our experiences two mailing lists per customer are enough. In case of one mailing list per project (as opposed to one per customer, which may have multiple ongoing projects) confusion would arise, since it would be unclear as to which mailing list to use for a given problem – especially because certain issues span multiple projects anyway. If a need appears (for bigger projects), it is anyway trivial to establish a special list.

2.2 People, time and responsibility

A bunch of e-mails between CRM and the customer quickly result in an agreement for the first project. As a counter argument to the "ideal software development cycle", we claim that often customers actually don't know how to effectively formulate all their wishes and requests – here is where we can help by drawing upon our long-term experience with the needs of physics community. At that moment at the latest a project manager is assigned to the project, and one of his/her first tasks is to estimate the time needed to get the work done. After he estimates the time needed for the work done, an offer is created. Every offer is also archived in the SQL database, where it is available for further use.

2.3 Request Tracker

After offer is accepted, the game begins. Under supervision of the project manager a project queue in Request Tracker (RT) is generated, project members are selected and proper rights assigned to each of them and also to the customer (each customer can see tickets only for his projects). For easier ticket submission and further replies a special e-mail address is created. In such a way, a client can send a feature request simply by e-mail. After the ticket is created it is automatically assigned to the project manager (if not specified otherwise) who then reassign it to someone else and also specifies priority, a time needed for the ticket resolution, a date and some other properties (which can be defined for each project separately).

2.4 To-Do Lists

Two of RT's features are linking and merging tickets. You can set ticket's parents, children or simply merge two tickets into one. We used this system to integrate our To-Do lists into RT system.

2.5 Ticket's Life-cycle

What is happening with the Ticket after its creation? RT informs the owner and the project manager. Both of them (and also ticket creator and some other authorised project members) can later reply to the ticket (using web form or e-mail) to report a progress being done, to answer to the ticket creator or something else. RT then kindly sends report to all ticket watchers (except the one, who is writing a report) and another great thing is that each of the recipients gets different mail, if there is a reason for it (we do not want people to get bombed with too many information). An important option is the ability of programmers to specify the time needed for task completion and the time actually spent working. In that way project manager gets overview about progress and developer's available time.

2.6 Activity Log

RT shows all their beauty and applicability as Activity Log. By properly selecting the search criteria, we can find out who worked on which project and for how long. It is also possible to make totals of time spent on a project, or of the time spent by one programmer on all projects and so on. We can also ascertain how long did it take to finish a project and how much money do we owe to individual programmer (students are paid per hour for example).

2.7 Concurrent Versioning System (CVS)

During the implementation cycle code has to be continuously synchronised with CVS [3]. We are using Eclipse [4] and other CVS specialised tools like TortoiseCVS [5] to archive data and to get insight into previous versions. We keep no binary, compiled or auto-generated files in CVS – just source code and documentation in XML format. Backups of the complete repository are being made once a day so as to avoid any nasty surprises.

2.8 Bug Tracking Tool

Not making any bugs means not working at all. That is why all projects have to be tested – first using automatic test procedures, so called modular tests, and after that also by hand – manual testing cannot be avoided, especially for GUI panels. All bugs are written into database, which is just one of the RT's queues. Our clients use the same interface if they discover any error.

2.9 Documentation

Our definition of a finished task is a program that passes all accompanying modular tests and is equipped with javadoc and the user's manual (that is a minimal documentation requirement). Documentation is written in XML format and it is transformed with a help of XSLT transformation and ANT scripts into HTML and PDF format. We defined our own XML tags to make writing easier (we all know that writing of the documentation is the hardest part of programming) and to achieve professional appearance. All documents and their abstracts are listed in the SQL database, which is automatically updated from the CVS repository. The beauty of XML documents is also that resources are not duplicated. In contrast to the binary files this fact keeps the CVS repository's total size smaller.

2.10 Night Builds With ANT

Every night (or as necessary) ANT [6] scripts are run. In the process all specified projects (configuration is done using SQL database) are compiled, programs are executed and tested with modular tests, javadoc and other documentation is generated. On successful builds a project web page is automatically generated. The build procedure creates a web-accessible log file and stores the build result into the SQL database. An important achievement is that there is only one, generic, ANT build script for all projects and customers. All project-specific details (like project dependencies, external jars needed for the project to compile, digital keys for jar signing etc are stored in SQL. There are always at least two versions of builds existing – the newest build and the last successive one before that.

2.11 Project Web Pages

These are automatically generated and provide access to the newest versions of programs and their data, such as information about developers, versions, dependencies to the other programs (other projects or some 3rd party software) at all times. Beside that there are also direct links to project documentation and javadoc. Please note that these, so called project pages, are created for the convenience of developers and testers and not for the customer (unless he is cooperating in developing – a Visual DCT [7] is such project) – nightly builds are not official releases but rather intermediate versions. Because they contain technical details, such pages would do more confusion than good for the external user.

2.12 Customer Web Pages

In contrast to the above-mentioned technical project pages, customer project pages are generated under supervision of the project manager after some integral part of the project has been finished. Just some clicking and PHP scripts are enough to create a page, where our customers can find information and news about the project, the latest official release (and some old ones if there is need for it), javadocs, documents in HTML and PDF format, installation executables etc. There are also links to related articles and presentations if some user would want to know some more about the technology deployed. With a system of permissions each user can only see projects related to him/her and some public related project (for example Visual DCT), which are developed for common good and are being funded on a per-feature request basis by different institutes.

2.13 Internal Pages

To integrate all of the aforementioned tools into a quickly accessible project management package, we are using internal pages – where information about active projects, project managers and members, customers, etc. can be found. There are also personal data of all our employees – address, e-mail and mobile phone number are indispensable in our style of working.

3 FUTURE DIRECTIONS

3.1 RT Analyser

RT has quite clear program architecture and very well defined database structure but it is written in Perl. Therefore we are developing a modular tool (in Java) which accesses directly to RT's database, creates tickets, searches and does some other fun.

With RT Analyser a project manager could simply (easy as writing in outline mode in MS Word) define tasks for a part of the project and the program would automatically generate To-Do list – RT tickets with the proper parameters set, above all the others with link to the Master To-Do ticket for specific To-Do list. You could also monitor the progress – there would be a list of tasks, where resolved would be in green colour, opened in yellow and the untouched in red.

4 CONCLUSION

The answer to the question, which could appear while reading this paper – "What is a role of the plant in this story?" – is: "Actually none; well maybe to create a cover for the director, who is spending time on Hawaii with his notebook and supervising the work of his employees simply by using RT, RT Analyser and browsing the web."

The elegance of such a system is that the description (and result) of almost every action is recorded. All information is well structured and therefore easily accessible – either via Web, our tools or e-mail. RT's possibility to process e-mails is maybe the most important of its features – most people feel most at home with e-mail service. Sometimes it is also easier to write an e-mail than to visit some web page, login and fill all necessary fields. This solved our problem with engineers that had preferred to write an error report into a logbook and stuck a screenshot along it, rather than submitting a bug into Bugzilla. There is no need to say that the book was in this other, distant country.

Is there no need for company offices, apart from the marble and glass that makes the owner feel rich? Actually, there is. Some meetings in person are still necessary – to end the long-lasting debates that keep persisting or reappearing on the mailing lists, to confront the people and to confirm important decisions. Usually one meeting in a week is enough. There is one time also, when a bunch of people sitting together is stimulating – when a project is reaching its deadline. In such times our room is full of people testing, compiling, repairing code, generating and helping each other. A so-called last minute panic is at least twice as effective as normal work. Maybe there is a concept alternative to ours – to be always in a state of last minute panic, but most probably people would get used to it and this effect would disappear. Therefore we replicate it only when it is really needed.

Another potential problem of working at home is and a lack of working habits. People have to be really serious, motivated and devoted to the company to adapt to this type of working. But this is already another story [7].

5 REFERENCES

[1] http://www.bestpractical.com/rt/
[2] Cosylab, http://www.cosylab.com
[3] http://www.cvshome.org
[4] http://www.eclipse.org
[5] http://www.tortoisecvs.org
[6] http://jakarta.apache.org/ant
[7] http://visualdct.cosylab.com
[8] J. Dovc et al, "A Guerilla Approach to Control System Development," ICALEPCS '01, San Jose, November 2001.