A World of Agents

Guest Editors: Pedro Cuesta-Morales, Zahia Guesoum, JuanCarlos González-Moreno, and Juan Pávón-Mestras

Joint monograph with Novática*

3 Presentation
Agent Technologies at Work – Pedro Cuesta-Morales, Zahia Guesoum, JuanCarlos González-Moreno, and Juan Pávón-Mestras

The authors describe trends and challenges that will need to be addressed over the next ten years in order to progress in the field of Agent Technologies.

11 Open Directions in Agent-Oriented Software Engineering – Franco Zambonelli and Andrea Omicini

This paper identifies key open research directions in the development process of Agent-Oriented Software Engineering (AOSE).

15 Verification and Validation Techniques for Multi-Agent Systems – Rubén Fuentes-Fernández, José J. Gómez-Sanz, and Juan Pávón-Mestras

This article gives an overview of, and presents some of the new methods used in, Multi-Agent Systems verification and validation.

20 Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities of Services – Marco Aiello and Paolo Giorgini

This article gives an overview of, and presents some of the new methods used in, Multi-Agent Systems verification and validation.

27 Developing a Multi-Agent System Using MaSE and JADE – Pedro Cuesta-Morales, Alma-Maria Gómez-Rodríguez, and Francisco J. Rodríguez-Martínez

An example of how to develop a Multi-Agent System using currently available methods and tools is described in this paper.

33 Engineering Multi-Agent Systems as Electronic Institutions – Carles Sierra, Juan A. Rodríguez-Aguilar, Pablo Nobile Blanco-Vigil, Josep-Lluís Arcos-Rosell, and Marc Esteva-Vivancos

This paper introduces an integrated development environment that supports the engineering of a particular type of distributed systems, namely multi-agent systems, 'as electronic institutions'.

40 The Baghera Multiagent Learning Environment: An Educational Community of Artificial and Human Agents – Sylvie Pesty and Carine Webber


This paper presents a Multi-Agent System that applies data mining techniques in a distributed surveillance camera system.

51 An Agent-Based Architecture for Developing Internet-Based Applications – Juan M. Corchado-Rodríguez, Rosalía Laza-Fidalgo, and Luis F. Castillo-Ossa

This paper presents a practical implementation of an agent-based architecture, which has been developed using the methodological framework defined by case-based reasoning systems.

56 Integration and Reuse Based on Web Technology and Aspect Object Model – Yauheni Veryha and Eckhard Kruse

This paper presents an object-oriented integration framework that uses Web technology and Aspect Object Model to integrate applications based on different component models.

61 News & Events: News from CEPIS, ECDL, and EUCIP; QoIS 2004 (Fifth International Workshop on Quality of Future Internet Services)

ENET (The full schedule of events can be found at our website)
Large-Scale Software Development in a Multinational Software Corporation

Jacek Czerniak and Wojciech Spiewak

This paper shows the process of code management. The study was based on the software integrity inspection process performed in R&D departments of the company Lucent Technologies. The authors present processes and principles that rule software code changes and quality control of performed changes.

Keywords: Software Engineering, Code Management, Software Inspection, Software Development Process, Software Quality

1 Introduction

Starting our discussion on code management in a corporation, we provide a number of notes concerning the background of the topic. The Lucent Technologies is a world-renowned company manufacturing telecommunications equipment, including the 5ESS phone switches, named 7R/E in their newest version. Used all around the world, the 5ESS switches are also the telecommunication network base of Polish providers i.e. TPSA, Netia, or El-Net.

The development of the software for the 5ESS switches takes place at Research and Development Departments of Lucent Technologies, which are spread over several countries, including USA, Poland, China and India. The 5ESS, in itself, is a multi-processor system, consisting of over 200 processors. The operating system that manages it is one of Unix releases originating from Bell Labs – formerly a part of AT&T. The software of 5ESS and 7R/E switches has been divided into 50 sub-systems. It is constantly being developed by numerous programmers from about 3000 in 1998 and about 1500 at the moment. To provide a full picture, it is necessary to add that the entire software of the switch includes over 10 million lines of code [1]. From this prospect, it is easy to understand why it is so important for a company to have an efficient code management system.

Experience that has been gained by Lucent Technologies is even more interesting, taking into account that none of Polish software companies employs several thousand developers, what, moreover, are spread over distant countries. Furthermore, the methods applied by LT and the quality achieved thanks to them have been confirmed by the certificates ISO9001 and TL9000.

2 Code Change Management

Access to code is provided by a separate group of servers storing the source code [2]. Each programmer has his or her own account on each server that is needed for his or her work. Code management for 5ESS and 7R/E switches is realised on two levels:

- ECMS (Extended Change Management System) [3] – the layer that initiates changes within the framework of a project,
- SCCS (Source Code Control System) [8] – the layer responsible for the control of the stored files’ versions.

Each project feature realised by a given developer is unequivocally identified by a unique IMR number. This acronym stands for Initial Modification Request. As the IMR number is associated with the project feature, it is easy to presume that we need more detailed identification of changes made in specific files of the code. That is why, on the basis of the IMR number, one generates numbers identifying specific changes, as needed. These are so called MR (modification request) numbers. It is a common programming practice to use one MR number to apply
changes in one file. If such a solution seems slightly complicated, one should bear in mind that the database may be accessed 24 hours a day by several thousand programmers. Thus each of the code integrity fluctuations may be dangerous for the functioning of the system.

Once the changes are introduced to the file based on the MR number, the SCCS layer stores up-to-date information on added, changed or deleted code lines. Such a collection of changes is called a delta [4][5]. For each delta, the ECMS layer stores a record of changes that includes:
• date and time of change,
• string of characters that identifies the developer,
• list of MR numbers, introducing changes.

Creating IMR Number for a Given Project

This phase has already been described above, but the schematic presentation gives a complete and precise view.

Creating IMR Number for a Given Project

This phase has already been described quite well. Thus, summing up, it consists in generating a new number that initiates a change for a feature (IMR number). All the changes connected with a specific feature might thus be unequivocally identified by a given project. Thanks to those numbers, gathering code changes that are to be delivered to a specific customer is quite easy and clear.

Setting Programming Environment to a Suitable Module

As has already been mentioned, the entire software that ensures proper operation of the 5ESS and 7R/E switch is divided into about 50 modules [1] which control the different aspects of its functioning. Among others, they include such subsystems as:
• Call Processing Subsystem,
• Charging Subsystem,
• Traffic Measurement Subsystem.

Thus, the developer sets environmental variables specific for the module, the files of which will be changed by him or her.

Creating MR Number Associated with IMR Number in order to Perform Changes to the Module

A developer who already has the IMR number associated with his or her feature has to generate a list of MR numbers specific for the current module by use of which the changes in files are going to be made. Those numbers allow for unequivocal identification of the change as belonging to a specific project feature and as applied by the engineer. Thanks to this, none of the changes in the code is anonymous, which also efficiently protects the system against sabotage.

Collecting Files for Editing

The files are unavailable to other users while they are being edited. Whereas other developers who would like to make changes to the same file can see by the ECMS layer interface, who is editing the file at the given moment. Thus editing the file for too long exposes the developer to phone calls and emails from annoyed co-workers. For testing purposes, it is also possible to create local copies of files, which are to be modified. However, it is intended exclusively for initial testing of a concept by the programmer. In the official base of the code, the changes are made only by collecting the files for editing using MR number.

Making Changes in Files

This part does not seem to require many comments. It’s an everyday programming practice, maybe apart from that the question of good and bad programming habits is settled by an internal document, so called Coding Process. It includes, among others, code formatting examples that are clear to all developers. There are also suggestions regarding comments and project documentation.

Transferring Files Including Unofficial Changes to the Base

The changes are unofficial as long as they are not officially added to the code. They are written as so called deltas in the SCCS layer. Each user can view the code using the ECMS interface and find out information whether a given code segment is included in the official file and, if not, then what its status is. It is very important because certain changes may overlap and therefore such information are crucial for the developer.

Compilations and Code Testing

This is the phase of preliminary testing, performed by the programmer. Here he or she tests the entire feature or its separate part developed by him or her. The phase of testing the whole project, which consists of several or a dozen or so features carried out by different engineers, takes place later on.

Code Inspection

This phase of software generation is very interesting and requires more discussion. This process is directly responsible for maintenance of code quality, creation of engineers’ good programming habits as well as for standardisation of the form in which the entire code is kept. The details of the code inspection...
process will be discussed in the next paragraph.

Transferring Changes into the Official Files
The actual transfer of changes for testing by the Feature Tester takes place in this phase. This is the person in charge of inspecting the entire project by testing the changes made by all the project participants. In fact, we deal here with a complete inspection of the project. Without a successful outcome of this phase, it is impossible to integrate the code with the official version.

Integration of Changes with the Official Code
Only at this phase the changes made to the code by the programmers actually become part of the official code.

Summing up, it has to add that all the phases mentioned above are executed very strictly. It is easy to understand this fact, bearing in mind that several thousand programmers are working simultaneously on several million lines of code. From this perspective one can see why it is so important for a company to have and to maintain quality confirmed by certificates mentioned at the beginning of this paper.

3 Code Inspection Process

3.1 Introduction to the Code Inspection
As mentioned in the previous paragraph, each change made to the code goes through called Inspection Process [6]. The author of the particular change is responsible for the organisation of such consulting event. Thus he or she collects the team consisting of the moderator and 2–4 inspectors, depending on the complexity level of the project feature which he or she deals with. The moderator is usually an experienced engineer who has been specially trained for leading code inspection processes.

Generally, the Code Inspection Process has three phases [7]:
• preparation,
• collection,
• repair.

The first one consists in an individual analysis of the code made by the inspectors searching for errors. Then, during the meeting of the inspectors, the moderator and the author, they exchange comments and propose possible changes. Finally, the author makes corrections suggested by the inspectors. Figure 2 presents schematically the phases described above.

3.2 Details of the Code Inspection
To have a complete understanding of the code inspection process, one has to present detailed study of it. Below (see Figure 3), there is a set of phases [7] through which the changes made by the developer must go.

Creating MR Numbers
Creating MR numbers, always when some change is to be done. As explained in the previous paragraph, MR numbers generate special software tools on the basis of IMR numbers. Thus, one can always identify the changes as associated with specific project or its independent part.

Making Changes to the Code Using MR Numbers
Changes are made to the code using MR numbers. This process has been described in detail in the previous section. It is only worth repeating that without MR number no changes to the code are possible.

Preparation of the Package Intended for Inspectors
The author of the changes prepares an inspection package intended for the inspectors. It is usually prepared in printed form and includes the code developed or changed by the author. Analysing those materials, inspectors may also learn how the code before the changes looked like. The package usually includes a short note that explains the type of changes that were made.

Choosing Moderator and Inspectors
The number of inspectors depends on the number of lines of code inserted by the author. The smallest team of inspectors consists of:
• moderator,
• three inspectors,
• secretary,
• author of the changes.

However one of the inspectors may also be the moderator, whereas the author usually also acts as the secretary. The moderator usually is an experienced engineer who has been specially trained for leading code inspection meetings.

Arranging the Meeting of the Moderator and Inspectors
The meeting of the moderator and inspectors is arranged by the author. In exceptional cases, it may be held on-line, by sending an e-mail with comments. This is called a Desk Inspection. However, the recommended form in the company is a face-to-face meeting, which is called a Meeting Inspection.

Preparation Period
Inspectors review the code independently in order to find potential faults. The time needed for this activity is then noted in the inspection protocol. There are certain standards binding in the corporation regarding the minimum time that should be spent for an inspection of a given quantity of code lines.

Meeting of Inspectors and the Author
As has already been mentioned, such a meeting may have two forms, i.e. Desk or Meeting. The entire meeting is chaired by the moderator. At the beginning the author briefly presents the problem which prompted the changes. Then, the participants exchange comments, suggestions and questions. Sometimes inspectors suggest alternative solutions and methods to those used by the author. During this meeting, the author collects all the suggestions and comments.

The Author Makes Changes, if Needed
The author is obliged to check the changes suggested by the inspectors. He or she pro-
ceeds in the same way as while making initial changes, using MR numbers in the way that has already been described.

**Moderator Checks Author’s Changes**

Moderator is obliged to check if the author took into account all the inspectors’ suggestions stated during the meeting.

**Moderator Approves Entire Changes Made by the Author**

Moderator fills in a special protocol of the meeting, including the approval of all the changes made by the author. To the official software version may be introduced only the changes made using MR numbers, for which in the base exists a record from the code inspection process, which has been approved a certified moderator.

Only after passing through all those steps, the code moves to the final testing phase. This is so called the Feature Test. After successfully passing those tests, the code will finally be added to the official version of the software and delivered to the customer. The same approach is applied in the project documentation.

### 3.3 Gains and Losses Resulting from the Code Inspection

#### Code Inspection Costs

There are two types of costs resulting from the introduction of code inspection. The first are of single nature and they may be treated as preliminary costs:
- implementation of procedures,
- training of moderators and inspectors.

The second category of costs are fixed costs, added to every development. The permanent cost are estimated as follows:
- app. 6% of development time devoted to Inspection,
- usually 10–15% of the development budget.

#### Gains Resulting from the Code Inspection

Earlier detection of defects saves time and money:
- many defects are found before testing, so testing is much quicker,
- total development time is reduced, even if inspection time is included, so inspection improves productivity.

Earlier detection of defects contributes to a better product:
- cleaner design,
- better documentation,
- better code,
- fewer defects,
- defects found earlier result in fewer deadline surprises,
- defects that remain will be easier to fix when found.

Inspectors learn from the inspection process:
- they learn from good code and documentation practice of others,
- they learn from mistakes of others.

In summary, here are some statistics in favour of inspection obtained from software development management tools:
- defects found by inspection: 57.7%,
- cost of finding defects by inspection: 1.58 work hours,
- cost of finding defects without inspection: 8.47 work hours.

### 4 Conclusions

As one can see from the above study, the principles applied during software development at Lucent Technologies are very strict. This process is constantly being improved and dynamically changed by scientists from Bell Laboratories and the University of Maryland. The data used for research on the effectiveness of the software development process are taken from numerous statistical tools used every day by developers. As can be seen, this multinational corporation has worked out its own work method and implements it in all departments, whether they are located in the USA, Europe or Asia. Thanks to the application of the same principles, it is easier to get a final product of high quality and handle migration of employees within the corporation, because apart from the procedures also the language is unified. The official language in the company is English, which significantly facilitates internal communication.

All those principles together are an interesting material for software development analyses. Code inspection process studies are performed not only at Lucent Technologies. For example, IBM company, another giant of the IT market, has worked out interesting material while developing similar methods. Both companies carry out experiments in order to determine an optimal number inspectors in relation to the quantity of inspected code lines. Recently, IT companies have also tested teams working in parallel who check the same code segment. In Poland, with a few exceptions, there are primarily small or medium size IT companies. It also seems to be interesting to analyse whether (and how) the principles presented above can be implemented in Polish companies. Which parts of the processes discussed in this article may be

---

**Figure 3:** Details of the Code Inspection.
applied in small and medium size companies, and which should only be introduced to big ones? It would also be interesting to determine, basing on the studies above, practical rules designed to assure integrity and high quality of the software developed by typical Polish IT companies.

References
Informatics Profession

How will computer professionals earn a living? (And why don’t you teach for free?)

Ricardo Galli-Granada

© Novática 2004

This paper was first published, in its original Spanish version, under the title “¿De qué viviremos los informáticos? (Y tú, ¿por qué no enseñas gratis?)”, by Novática (issue no. 168, March–April 2004, <http://www.ati.es/novatica>). Novática, a founding member of UPPENET, and publisher of UPGRADE on behalf of CEPIS, is the bimonthly journal and magazine of the Spanish CEPIS society ATi (Asociación de Técnicos de Informática, <http://www.ati.es>.)

In this article the author, a university lecturer in Computer Science, takes an unorthodox look at the computer profession and answers the question posed by an opponent of free software / open source “Would you be willing to adopt the same approach to your teaching and teach for free?”

**Keywords:** Computer Profession, Education, Free Software, Open Source, Proprietary Software.

It’s always the same; no sooner do I explain the advantages of development and business model of free software / open source than someone tries to counter my argument by saying: “You only say that because you’re a university lecturer and you have no idea of business... If you think free software is such a good idea, why don’t you teach for free?” [13]

One of the things that seem to be very hard for computer professionals to accept is that there is a good living to be had by selling services, and that we software developers really need to take on board the fact that we are working in a service market and need to focus our efforts towards that market if we want to aspire to the benefits that other professions reap. This requires us to bring our practices in line with those of other professions.

How the other half lives

Whenever I chat with computer professionals, when the conversation turns to the quality of life of lawyers, doctors, accountants, financial advisors, architects, etc. compared with IT developers, they all say, without hesitation, that it is far better than theirs, both in terms of money and social prestige.

“They really live well. They can charge good money for their work, based on set rates which no one tries to beat down. Totally unlike how computer professionals are treated...”

But what do all these professions have in common? All of them are based on information which is public and accessible at a very low cost. The laws and jurisprudence used by lawyers are public or easy to access; there are even firms that make money by gathering and collating information in different ways. Doctors simply apply their knowledge of medical science and current protocols which are, of course, fairly widely known and accessible to nearly everyone (pharmaceutical sales reps also form an incredibly effective knowledge dissemination network).

As for accountants and financial advisors much the same applies. Their work is based on an exhaustive knowledge of laws, rules, regulations and the state of the markets, all of which information is also available to anyone who is interested in finding it.

However people continue to hire lawyers, accountants, and architects, and go to the doctor at the first sign of flu. Why? Because those people are experts in their respective fields, they have studied them for a considerable length of time and/or they have a fair amount of experience in them and know how to apply that knowledge in order to solve problems.

In spite of having access to hundreds of Google’s and databases of medicine, law or stock market reports in real time, we will go on using these professionals...

...and why should the computer profession be any different?

Why is it that an architect charges by the hour he puts in and what he most wants is for his buildings to be visited and admired by as many people as possible? Is he mad? Doesn’t that mean that everyone could copy him and take away his source of income? Wouldn’t it be better if he charged a commission for every visitor or copy made of his designs?

If laws are public and we all need to be aware of them, why should we have to spend a fortune on lawyers? It doesn’t make sense. Shouldn’t lawyers work for practically nothing? Or shouldn’t they keep the laws secret to prevent people from muscling in on their territory and forcing their fees down? Imagine, if you will, the following headline: “The bar association asks for public access to laws [1] and to legal information [2] to be restrict-
ed to prevent unlicensed practitioners and unfair competition from people who defend themselves in court thereby causing lawyers to lose money”.

It sounds totally absurd. But it could be argued that it’s a special case as the laws are drawn up by States and paid for by all of their citizens.

So let’s look at another example: “The medical association asks for access to medical formulas and protocols [3] to be restricted because otherwise the medical profession will die out due to low salaries and a lack of patients”. Yes, this would be totally ridiculous and unacceptable in any western society.

In this case it could also be argued that doctors can save lives, or lose them, and therefore it’s not the same situation.

While it can also be argued that computers fly airplanes, manage medical histories, conduct biological and genetic analyses, CAT and ultrasound scans, etc., I’ll assume for the

---

Ricardo Galli-Granada has been a lecturer at the Universidad de las Islas Baleares, Spain, since 1992. He holds a BSc in Computer Engineering and is a Doctor of Computer Science (his doctoral thesis was entitled “Data Consistency Methods for Collaborative 3D Editing”). He has worked in the field of computer graphics and CSCW (Computer Supported Collaborative Work). He has published more than 40 articles in national and international conferences, journals and chapters of books and he has also taken part in and/or led several European R&D projects. He was the co-founder of the service provider Internet Atlas-IAP S.L. and has directed and coordinated the development of digital editions of 6 newspapers in the Balearics and several e-commerce projects. He is currently collaborating on a number of free software development projects. <gallir@uib.es>
moment that the example is not entirely satisfactory and I’ll try to find another one.

How about this one?: “The Spanish Association of Architects asks for the abolition of the obligation to publicly exhibit plans and scale models of public works. They argue that not only does it hinder innovation, but it also endangers the livelihood of architects (and engineers) by encouraging unlicensed practitioners and unfair competition. The association has the support of Frank Gehry, the designer of Bilbao’s Guggenheim Museum and inventor of a new architecture based on curves and space-age metals. Gehry not only supports the association’s request but also asks for access to the entrance to the museum and the surrounding area to be restricted to prevent the indiscriminate plagiarism of his innovative designs by other architects with less training and no ability to innovate. ‘The fact that the museum is open to all and sundry has caused me to lose money. If projects of this nature become commonplace, the architect’s profession will be in serious trouble. I would say that the whole industry, right down to construction civil engineering, is also at risk’, said Frank Gehry about the work he carried out for the Guggenheim museum”.

All right, so the above example is silly and unbelievable. How about this one then?: “The Spanish Association of Computer Engineers asks for web pages’ HTML code to no longer be made accessible to the general public. They argue that not only does it hinder innovation but it also endangers the profession of computer professionals by encouraging unlicensed practitioners and unfair competition. The association has the support of Tim Berners-Lee, the designer of web architecture and the HTML and HTTP standards. Berners-Lee does not only support the association’s request but he also asks for access to web page source code to be restricted to prevent the indiscriminate copying of his innovative designs by other computer professionals with less training and no ability to innovate. ‘The fact that the system is so widely used has caused me to lose money. If systems of this kind become commonplace, the computing profession will be in serious trouble. I would say that the whole IT industry is also at risk’, said Tim Berners-Lee about the work he has done for W3C”.

Does this seem likely? Does it seem reasonable?

Personally, I think that the argument is not unlike the one used by many computer professionals who disagree with the idea of free software. Despite the fact that precisely the standards and the freedom of access to HTML source code – and the possibility of copy-and-pasting – have been decisive factors, not only in the growth of the Internet, but also in the standardization of quality levels throughout the website development industry.

Do you see any difference between the web designs and developments of a Silicon Valley company and the same product from a company in Sa Pobla, Mallorca? Would that have happened if HTML code had not been accessible?

Lecturers’ salaries

Moving on to the matter of my salary as a university lecturer, what do a lecturer, an architect and a doctor have in common? The answer is that all the knowledge we lecturers impart (i.e. copy) is public and accessible, as is all science. We merely copy it and impart it (sometimes, admittedly not often, our students actually understand us). That is all we do, but we don’t demand commissions on our students’ future earnings as a result of the knowledge we have just imparted, nor do we impose restrictions on how that knowledge is used, copied or passed on. It becomes the property of each individual student.

So, why do we lecturers get paid? Simply because we sell a service, just like a doctor or a lawyer. There are people who need to study in order to meet certain requirements in order to obtain a professional qualification, and perhaps it’s easier to go to classes and listen to a lecturer and go to laboratories (or do it on the Internet) than to do it privately at home on a self-study basis. You may now be thinking why do we all have to pay university lecturers? But there are also private university and private university lecturers, and sometimes they earn more than the public university lecturers do. And the consumers or customers sometimes pay small fortunes to take engineering, bachelor or masters degrees at private universities.

In short, there is a market and demand, and these give rise to businesses (perhaps public universities are playing their role of ensuring universal access and quality merely by demanding research work from their students.)

There are tens of thousands of tutorials in Word [4] or Excel [5], but there are more and more private centres offering introductory courses to those packages (a question: would those teachers’ lives be very different if the market demanded OpenOffice [6] rather than Microsoft Office?).

In other words, there is a demand for teachers and that’s why we exist. Only because of the demand (teachers do not exist by divine grace – it has been that way for hundreds of years, in spite of the enormous changes there have been in science and education). If the market ceased to demand education, there would be no teachers or they would have to emigrate to other countries where they were needed. As happens in many poor or third world countries. Or as has happened to me, personally: I had to emigrate from Argentina to be able to be a doctor of computer science and a university lecturer (the free market is no guarantee that we will be able to pursue our chosen profession).

There were those who said that the arrival of books, with all that meant in terms of the popularization of education, would cause sages (what we would nowadays call experts or scientists) and the whole education system (for the privileged few) to disappear, but in fact exactly the opposite happened. The dissemination of knowledge, culture and science has not only increased the demand for lecturers, teachers and scientists, but it has also improved their social status.

Never before has there been such a demand for a strong and competitive education system, at least in the leading countries (which not uncoincidentally have the best education and scientific systems…).

But IT is a special case...

So we can agree that the work of architects, doctors… even teachers, and even most scientists who only invent or discover something of little significance in their entire working lives (even those who work in private laboratories) is based on public knowledge and that only some of them, relatively very few, make small innovative contributions to industry as a whole.

In spite of this they are well paid and on average enjoy a fairly good quality of life, but it would appear that computer professionals think that the same business model (as represented by free software) applied to IT is unsustainable.

Most computer professionals complain about the number of hours they work, about the concentration that programming requires (it has been scientifically proven; programming requires a great deal of concentration and is very tiring work), about the pressure they are put under to meet deadlines. Or about how difficult it is for self-employed professionals to find customers who will pay the prices that they ask and are able to recognise the quality and reliability that they provide.

Apart from my academic experience [7], I have also dabbled in the business world [8]: I have been co-founder and partner of a tech company [9], and I have to say that yes, the IT market is a very tough one; I’ve been there. But why the difference with other professions? Are we worse people? Are we more stupid? Are they cleverer?
As software is highly recyclable, replicable and transferable, to make proprietary software marketable as a physical product artificial restrictions are created, such as intellectual property law or software patents that penalise copying without express authorisation.

In proprietary software we invent the wheel over and over again and it doesn’t always come out round since it can rarely be compared with competitors’ products.

There is a serious contradiction in that, as they are secret, a large proportion of proprietary programmes may be in violation of the very intellectual property laws or patents that allow them to be marketed.

As it is very easy to change computing standards, at least much easier than it is for a power supplier to change from 220V and 50Hz, or for Canal+ to change to PAL, it is much easier to create captive markets (customer lock-in) and thereby either do away with entry barriers or raise them even higher.

The intrinsic properties of infinite replicability (zero marginal cost) and software transfer make it easy for monopolies and non-competitive markets to be established, ones with non competitive pricing, that is, prices that bear no relation to the production costs of the software (you only have to look at Microsoft’s earnings, despite participating in markets which generate huge losses, such as the XBox).

Monopolies take advantage of the above-mentioned problems and of the network effect [10] generated by the sheer number of users their systems have, which allows them to define their own standards (or the de commoditization of protocols [11]) which in turn extends the life of the monopoly.

As the major markets are controlled by the monopolies and due to the replicability of software, very few programmers are actually employed. You only have to consider that Microsoft, one of the largest companies, with over 90% monopolies covering two very important areas, more than 400 million Windows licences sold, which is where the biggest profits come from, with 200 millionaires among its staff... employ a mere 10,000 programmers, approximately 0.4% of all the programmers currently employed in the USA (some 2.5 million in total) and 0.1% of the estimated total number of programmers worldwide.

If it were not for free software, or open code programmes, only the smallest minority of technicians would have access to the knowledge of the basic techniques used in computing technology (operating systems, databases, text processors, viewers, calculating systems...).

As the tech industry is mainly centred around very few companies which concentrate all their programmers on American campuses, the geographical concentration is very high and consequently the regionalization of profits in these new industries is practically non-existent. Basic developments can only be made at the top of the technological pyramid (SQL, personalization of ERPs and CRMs, spreadsheets, web pages, applications based on third party middleware, etc.) which makes regional SMEs think more than twice before embarking on RDI projects.

As there is no motivation for companies to invest in research or risk money on innovation, the technological gap with the countries in which the major IT industries are concentrated (principally the USA and some Scandinavian countries) is growing wider by the day.

### How do computer professionals survive in such a hostile market?

How can they:

- Sick to the stomach when you have managed to persuade someone to pay you 6,000 euros for developing a complete management programme, only to discover that some consultancy firm with an impressive English sounding name has billed the same for just twenty hours of work involving the drafting of a report advising the use of SAP or Oracle, which was done by someone on work experience.

- Finally, when you have found a good customer who wants you to install 100 computers, 50% of the software budget goes to a couple of American companies whose sole contribution has been to send you a box of CDs, a plastic card which says “Certified Engineer” and four manuals in PDF describing the programming and basic set up of the system.

- And for the coup de grace, the freelance designer of the web pages you made for the company as a freebie sends you a bill for practically half of what’s left of the money.

Meanwhile, computer professionals earning a pittance are praying that things won’t get any worse.

### But we shouldn’t lose sight of the basics...

... how do the customers fare? Well, even worse,

- with systems which fail for want of patches or maintenance,
- a constant renewal of licences, either software of hardware to keep systems up to date,
- tied to a supplier which prevents them from changing even when there are better technical and economic offers on the market,
- total ignorance of what they are being sold or what is being installed, which prevents them from ensuring that they have technological independence and are able to recover data to migrate to rival systems,
- complete renewal of systems every so many years due to the disappearance of the company that developed them or maintained the previous technology,
- etc. etc.

All the abovementioned reasons are why the prestige and ‘social status’ enjoyed by computer professionals is currently at their lowest ebb.
So, what now?

The scenario is depressing, but free software / open source may provide the solution to many of these problems. Especially if we accept that our profession is just that, a (liberal?) service profession, just like lawyers or architects.

And maybe it’s better to dream of being a comfortable middle class professional, like the dozens of lawyers or doctors that we all know, than dream of being the one and only Bill Gates that we see on the telly.

In another article I intend to go into some more issues, such as the reason behind the inevitability of the existence of free software, the other, non-monetary motivations of many FS developers, etc... (but this article is hefty enough for now :-). 

Translation by Steve Turpin

Note: Further information on the subject can be found, in Spanish, at <http://bulma.net/body.phtml?nIdNoticia=1961>.

References (all of them in Spanish)