Experiences with the Concept of Free Software in a Professional Education Environmet

Francisco José Monaco[1] , Celso Renato Goncalves Providelo[1] ,
Willians Jorge Rodrigues[1] , João Henrique Gião Borges[2]

[1]SEL/EESC, Universidade de São Paulo, Av. Trabalhador Sancarlense, 400, 13566-590, São Carlos - SP, Brasil.

[2]Faculdades COC, Rua Abrahão Issa Hallak, 980, 14096-160, Ribeirão Preto - SP, Brasil

monaco@sel.eesc.sc.usp.br, cprov@iris.sel.eesc.sc.usp.br
willians@igbt.sel.eesc.sc.usp.br, jhgb@coc.com.br

Resumo. *A despeito de exemplos de sucesso como os projetos GNU e Linux, suposições equivocadas no que concerce ao conceito de Software Livre dão lugar a vigorosas discussões entre céticos e entusiastas acerca de temas controversos, urgindo que proponentes demonstrem que o princípio da liberdade, quando aplicado, produz os benefícios propostos. Este artigo reporta experiências no tratamento de tais questões dentro de um ambiente de formação profissional.*

Abstract. *Despite the successful examples of the GNU and the Linux projects, misleading beliefs regarding the concept of Free Software give rise to vigorous arguings among skeptics and enthusiasts on major controversial subjects, urging proponents to demonstrate that the FS theory, when applied, produces the proposed benefits. This paper reports experiences in dealing with those issues within a professional education environment.*

1. Introduction

The GNUsp Project at the University of São Paulo is a volunteer initiative leaded by an open group of students and researchers whose goal is to promote the utilization of free software (FS) both in administrative and academic activities. By maintaining an infrastructure of communication and mutual cooperation , the group offers advisory and technical support to the local community, propagating the knowledge on the benefits of non proprietary technology and encouraging its adoption both in research and in educational projects. During this period, GNUsp has been playing an important role in the process of FS dissemination at the University and its mission has been successfully carried out, not only in its activities within the campus but also in external events, taking part in national and international workshops, expositions and seminars.

However, in spite of the enthusiasm with which many goodwill students naturally embrace the FS philosophy and its principles based on human values and community spirit, not all of them keep committed to its ideals when moving from the academic environment to the work field. Out of the two main reasons the practice in the area has allowed to identify, one concerns the massive pressure of "competitiveness ideology" today endorsed by the professional marketing, which imposes an aggressive posture against adversaries, hence allowing no place for cooperation and friendship issues — this perspective has already been introduced in a previous paper [Rodrigues et al., 2001] on the GNUsp activities. This article, on the other hand, is devoted to the second reason we attribute to the mentioned phenomenon: the misunderstanding of FS essential concepts. In fact, as it can be inferred from the continuous interaction with software users and developers, not only opposers but even many FS proponents do not have a clear view of its pertinent fundamental ideas and how they apply in the practical sphere. As consequence, misleading beliefs about the FS model pervade this field and are often associated with skepticism among students and computer professionals who, although sympathetic to the general FS purposes, found themselves challenged by doubts regarding its practical feasibility, thereby ending up by concluding that the referred philosophy is nothing but a dream.

The experience earned from organizing GNUsp lectures and round tables has helped us to identify several obscure points which contribute to general misunderstanding on FS essentials. This information, along with inputs from collaborators, has been used as feedback to improve our arguments and develop a discourse tuned with our audience — this need has become more evident since the GNUsp Project started to present seminars in regular undergraduate disciplines upon invitation of course coordinators. Thus, while the GNUsp approach in this extent is mainly based on the essentials of the FS philosophy itself, presenting the proposal in a way to conciliate conceptual and technical perspectives, and clarifying how the involved moral principles can match professional aspirations, has proven to be extremely important. This paper shares our results in this line with those involved in similar initiatives, specially in the context of professional education environments.

2. Misconceptions

Despite sound counter-examples offered by regarded successful endeavors such as the GNU and the Linux projects, there is a common misconception among non-practitioners that the FS proponents are found among those who suffer from the "syndrome of the proud mediocre": an ego pathology which affects either the lay amateur who disdain the learned professionals' wisdom or the ingenuous self-made expert who is too proud to admit his background deficiency. Such simplistic belief, promoted either inadvertently by the lay media or maliciously by opposers aiming at rendering FS as a social utopia, is responsible for the association of FS with "backyard job" and has contributed to keep the theme out of the topic list of academic and industrial conferences for so long — indeed, our proposal to include FS in a scientific meeting was once rejected under the justification that it is not but "folklore of the Internet".

Among other misleading assumptions that contribute to this scenario is judging, upon examples of remarkable projects not rigidly founded in conventional development methodologies and ruled by an anarchical self-organization, that FS advocates void the utility of software engineering in favor of empiricism and discipline rejection when, conversely, most large long-term FS projects are leaded by professionals from the academy and industry. While it is somewhat difficult to conceive an experienced developer, even in the most convicted FS factions, arguing in favor of poor techniques and chaotic work, this odd belief still remains. In this regard, it is worthy to point out that, since most traditional models and techniques for software development have arisen in the context of proprietary industry needs, it is prudent to understand how they apply in a different context. In fact, given the intrinsic differences between centralized processes controlled by proprietary product manufacturers on one hand, and the enormous diversity of views, needs and convictions present in large world-wide open communities, it is hardly possible to reach unanimous agreement upon too rigid schemes — resulting that we should expect to observe the co-existence of a diversity of methodological approaches adopted by different parties. Likewise, it must be remarked that due to the inherent open nature of FS community, it is desirable to welcome not only software experts but also any goodwill volunteer disposed to take part in the global efforts. Since these naturally tend to comprise the largest fraction within the group, either strict formal models not of broad consensus or too complex techniques which imposes hindrances for the participation of beginners and average-skilled contributors are not likely to become preferential choices. The net result is that, even critical tasks being decided and executed by experienced teams, the whole heterogeneous community may sometimes present an overall aspect of a loosely systematized development practice.

The comprehension of this peculiar feature of the Free Software Movement (FSM) is not trivial, at least for newcomers, seen that it strikingly diverges from the standard conventions taught by the proprietary software experience. As consequence, many just-arrived users are convinced by the arguments of opposers which propose that the transition from the early open cooperative development practice of the 70's to the current approach of proprietary software corresponds to a natural evolution from a primitive and rudimentary activity into a well-structured business branch. Perplexed then by the "paradoxical" success FS examples, some tend to attribute the fact to the technical quality of the piece of software itself, often concluding that it would be even better if it was handled under "modern methodologies".

In accordance with the approach encouraged by the FS community leaders, our reaction face this circumstances has been to keep in mind that the FS conceptual foundations lie upon socio-technological arguments, and that deliberately attempting to dissociate the two complementary aspects to avoid polemics would render the genuine proposal inconsistent. We answer, thus, that the FS movement aim not only the convergence of collaborative efforts on the development of cost-effect quality software, but also an alternative professional attitude more compatible with the human values towards a fairer and more prosperous society. Nonetheless, while this perspective is the basis of our action line, the need to address pragmatic questions raised by either prospective practitioners or undecided users has given rise to vigorous arguings among skeptics and enthusiasts on major controversial subjects, urging us to demonstrate that the FS theory, when applied, produces the the proposed benefits — or in other words, that sticking to good principles is worthwhile in practice. In short, the pertinent observation to point out is that the success reached by distinguished FS projects is not due to casual superiority of a few pieces of software, but rather it is the consequence of a conceptual principle (freedom) whose practical application allows them to evolve fast and consistently, free from constraints imposed by the non-free approach.

3. Freedom on Action

As it happens to other merchandise, the cost of commercial proprietary software tends to increase as a function of its refinement and sophistication. As result, access to quality software and to the last advancements in this field has been restricted to a small portion of privileged users, since this model became dominant on the market. Another large portion of potential users, on the other hand, has been compelled to adopt solutions of inferior quality, usually having to resort to "cheap software", basing their choices more on cost criteria than on technical adequacy. In this regard, the key-concept in the alternative proposal is that in the FS model, rather than a merchandise, software is a work resource — what means that instead of profiting from selling software, professionals are remunerated to develop it (some uses to state it in terms of a shift of the core business from commerce to service). Then, a subtle but crucial feature of FS work model comes into play: the non-restrictive principle. Opposite of what usually happens in the proprietary industry branch, this principle implies that, once the software is ready, it does not belong neither to the one who created it nor to the one who payed for it. In practice, the legal propriety over the final product is actually guaranteed, in the name of either the author or the sponsor (it is often necessary even to avoid the misappropriation of the software). However, the license that the proprietor applies to the software does not restrict the rights of its users to benefit from it according to their own decisions — hence resulting that the software is *free*, in the sense of freedom, as free for the legal owner as it is for anyone else. It is noteworthy, then, that in this scheme all parts are benefited: the author who has exerted his professional functions, the sponsor who has the needed software developed and also the whole society which counts on a new resource available for the common good.

In order for this freedom to be effective, it must allow any user the same rights, what is consistent with the definition of Free Software. While originally brought up in terms or four elementary freedoms [Stallman, 1985], the idea may be also summarized through a trine of fundamental principles which state that a piece of software is free, in this strict sense, if and only if the user is granted the rights of *utilization*, *development* and *distribution*. For this to happen, the software must not subject the user to any sort of artificial constraint either of technical or legal nature, what in turn implies that the user must be allowed the access to any existing information he may eventually need in order to exert his freedom (this include the source code and pertinent documentation). Likewise, the software must be absent of proprietary technologies which might impose restrictions on the referred rights. That assured, FS offers the fair opportunity of access to quality and sophisticated software indistinctly, benefiting mainly small users and independent developers who may count on the same resources at the reach of large companies — mitigating the disadvantages imposed by commercial proprietary software.

Practical effects of the freedom principle go further. For instance, FS is not only cost-effective, but also flexible. Being open-source and absent of restrictive license terms, FS may be modified by the user, who is *free* to fit it to his needs. Thus, in place of general-purpose closed solutions created to satisfy generic needs of a market slice, it offers adaptable systems with extensible functionality suitable for continuous optimization process under the user control. As consequence, another advantage is that the user-driven development is then not subject to the particular conveniences of any technology holder and thus users are *free* from the strategic decisions of third-party suppliers, such as the the common artifice of purposely delaying the release of new versions with bug fixes or improved functionality according to market opportunities. Also, since neither technical nor legal restrictions apply, solution providers are *free* to offer their clients the best option exclusively in function of adequacy and not of license impediments; clients are *free* to integrate their application with other existent, migrate from one technology to another and even change supplier without difficulties. Allowing white-box evaluation, FS can have its quality assessed and its bugs fixed prior to its manifestation, even by the users, who is *free* to do it by himself thereby not having to wait for the main developer's decision.

Working on a publicly accessible collection of resources which is continuously improved through the cooperative efforts of all the community, FS developers are not adversaries competing for conflicting goals, but partners who benefit from the success of each other; so they are *free* to give their best to contribute to the community. Therefore, volunteer spirit is not incompatible with individual aspirations and strategic information need not be hidden but can be shared with benefits for all. Instead of wasting efforts to produce rival versions of already available products, professionals are *free* to efficiently direct time and talent to create new solutions and to optimize existent ones, hence the faster evolution of collaboratively developed software. Finally, having access to a suitable amount of cost-effective quality FS, users are *free* and motivated to bring together professionals — programmers, analysts, engineers etc. — to design and implement customized solutions, reactivating the in-field software development that was dormant since the proprietary closed applications have spread in the 80's. Better software for the user, more opportunities for professionals, fairer perspective for the society. Too god to be true ? No, its just freedom on action.

4. Conclusion

The conclusion, thus, is that the promising scenario is indeed the practical result we get from applying the concept of freedom. That's why, FS proponents believe that justifying the interest on free software merely in virtue of technical convenience of the implied development model is seeking the effects disregarding the causes. Freedom is a principle; good software and friendly relationship are consequences. Defining FS only in terms of methodology does not help people to form a conscious opinion upon which to base their decisions when facing new challenges — like falling back to the resort of non-free software when it eventually suggests individual advantages, or surrendering to the temptation of hiding a very good original idea upon opportunistic interests instead of sharing it with the community. Does such pragmatism help open cooperative work ? Free Software is not only about software; it is an attitude.

References

Rodrigues, W. J., Monaco, F. J., Gonzaga, A., and de Aguiar, M. L. (2001). Free software in superior education : Talking about freedom to future professionals. In *I Workshop sobre Software Livre - WSL'2001*, pages 43–45, Porto Alegre, Brasil.

Stallman, R. (1985). What is free sofware. Web. http://www.gnu.org/philosophy/free-sw.html.