On open source software and the organization of cathedral-building: metaphors and realities

Paul A. David
Stanford University & Oxford Internet Institute
pad@stanford.edu

&

Jean-Michel Dalle
University Pierre et Marie Curie & IMRI-Dauphine
Jean-Michel.Dalle@upmc.fr

First draft: 10 December 2001
Second draft: 22 February 2004
This version: 12 October 2007

ABSTRACT

Eric S. Raymond’s (1999) influential essay The Cathedral and the Bazaar paints contrasting pictures of the orderly, hierarchically controlled programs of planning and execution that built the great cathedrals of Europe, on the one hand, and, on the other, the ferment and seeming chaos of the bazaar. The latter then is offered as a metaphor for the mode through which free/libre and open source software (FLOSS) is developed. Numerous commentators have express dissatisfactions with “the bazaar,” but this note argues that both sides of the metaphorical juxtaposition are misleading: large open source projects in some significant respects resemble the medieval mode of cathedral-building, because the latter was not so orderly and hierarchically structured in their planning and execution, whereas the former appear to devote considerable attention to mechanisms of governance and control.

Acknowledgments

On open source software and the organization of cathedral-building: metaphors and realities

Introduction: The double-edged metaphor

Possibly the best known “economic” work about open-source software today has not been written by an economist but by a programmer and open-source “advocate”: it is of course Eric S. Raymond’s influential *The Cathedral and the Bazaar* (Raymond, 1999).

In the essay that carries that title, Raymond paints contrasting pictures of the orderly, hierarchically controlled programs of planning and execution that built the great cathedrals of Europe, on the one hand, and, on the other, the ferment and seeming chaos of the bazaar. The latter then is offered as a metaphor for the mode through which open source and free software is developed. Essentially, the claim is that free/libre and open source software (hereinafter referred to as FLOSS) rests upon an efficient, agile bazaar-like organization, in contrast with the less slow-moving, top-down organization embraced by the large proprietary software producing corporations.

Numerous alternative metaphors have been offered, to convey the essentials of FLOSS production, implicitly demurring from Raymond’s characterization. In some part the implied critique is based on an incorrect interpretation and reporting of Raymond’s claims in regard to the FLOSS-bazaar as an organizational system. On the other side, curiously, most critics of the bazaar metaphor appear prepared to completely overlook the fact that it is a retail distribution organization, and not at all a mode of production!

But there has been considerably less attention to the other side of the putative contrast. This the image of the cathedral-building as a model of design and implementation of complex structures whose organization is inappropriate, antithetical to the democratic egalitarian values and al fresco creativity of the FLOSS movement, and the source of inefficiencies in software production from which the programmers of who form FLOSS communities have been liberated. There are two parts in this juxtaposition: the view of the place of organization and control in FLOSS activities, and picture of European cathedral building. Our purpose in writing these notes is to point out that Raymond’s essay is misleading about both of them. A closer look at each may serve to reveal more clearly a number of important features of FLOSS as a mode of production.

In the following part, our essay suggests that although there obviously is a world of difference between Chartres and Linux, it may be quite misleading to accept an historically inaccurate image of the way that the Gothic cathedrals came to be, and on that basis suppose that the organizational challenges met by those architectural projects below to a universe wholly different from that in which the large FLOSS projects find themselves. The second part turns to examine the claims made by Raymond’s essay about sources of the alleged superior efficiency of the FLOSS production mode, and argues that more attention (and corresponding credit) should be given to the internal organizational features of the major projects.

---

1 FLOSS is even said to be an “important organizational innovation” (Lerner & Tirole, 2000).
Cathedral-building in reality and as a metaphor of hierarchical organization

There are obtrusive points of contrast concerning the organization of actual cathedral-building processes, and the production conditions of software systems that we are concerned with: the scale of material resources was altogether larger, indeed immense in the former case, especially if considered in relation to the society undertaking these projects. Not surprisingly, the projects extended over much longer time spans, being interrupted by want of funds, or of the enthusiasm of patrons and enthusiasts willing to mobilize the necessary financing.

An obvious enough second point to note is that in the modern OS project the individual or individuals at the core of the project are committed to it on a full time basis, and they seek to mobilize volunteers who typically will participate on a part-time basis -- at best. By contrast, L. F. Salzman's *Building in England Down to 1540* (Oxford, 1st Edition, 1952), classic account of medieval construction activities, and the researches of others' into the extant contracts between medieval cathedral patrons and their master-mason/architects, points out that both those parties at the core of the cathedral project often were less than fully occupied with it. The patrons have other responsibilities, being princes of the state and church; and they are continually worried by the tendency of their master-mason/architects to be diverted by involving themselves in other projects. There are nice examples of contractual arrangements stipulating that the mason has to be in residence at the site of the project, or not go away for more than stipulated intervals, etc.

On the other hand, the artisans at the construction sites of real cathedrals are attached to their respective trades on a full time basis so long as there is work for them. These were the Fre-Maccons (free-stone masons), a workforce whose size at a site fluctuated within the week, and from week to week on the large projects where hundreds were employed. Salzman's discussion of this implies that this was not a reflection of difficulties of keeping the attention of the workers. Rather it seems to have stemmed from shifts on the demand side, due to interruptions in the availability of materials, financing for wages, or suitable weather conditions.

A further significant point concerning the Fre-Maccons' availability is that the mobility of these skilled artisans in the middle ages was atypical, and not a feature of the organization of construction trade in general. It reflected the small number of building projects in stone, whereas there were enough projects of other kinds to support local (guild) organization for the carpenters, brick-masons and other artisans in the towns and cities. These masons placed their "marks" not only on sculptures, but on stonework in the fabric of the structure. "Free" stone contrasted with building using stone rubble and "clunch"; it referred not only to sculpture, but to stonework for ashlar facing, and mouldings on rib-vaults, etc., which typically would be prepared -- along with sculpted stonework -- by masons in "traceries" (typically, covered temporary buildings where "tracings" of standardize mouldings and facing blocks were made) and not at the immediate building site itself. The itinerant gangs

---

of masons would sometimes be "lodged" in these same (tracerie) buildings -- especially before the large projects such as Chartres. But master-masons occupied a more elevated status.

Salzman also provides good details on the point that ground plans and elevations, which a medieval master mason would know how to prepare, were used chiefly in contracting between the patron and his mason-architect. Such plans rarely survived long, because as soon as the building's foundations had been laid out the ground plan would be "exposed", and similarly, a section of wall would indicate the design of the symmetric elements. The parchment was too valuable to be stored, and so would be wiped clean and re-used. Wooden templates to indicate the stone moulding designs, however, would be preserved for re-use in later projects -- and are shown to have been transferred from France to England, and vice versa where the mason-architect worked on several cathedrals. Is this not "modularity of "codified design elements" – analogous to libraries of use-tested sub-routines and algorithms that software programmers may draw upon to assemble larger systems of code.

A first point to which we should attend is that “the cathedral” is not just one structural form. The major distinction to be drawn here is between the generic "cathedral" and the Gothic cathedral, the latter being the form to which it is reasonable to suppose Eric Raymond and others have referred. von Simson's is well known for having made the case that Gothic cathedral was a 12th century innovative reaction against the Romanesque. The new Gothic style and the construction practices associated with it has been shown to emerge first in the Ile de France, from whence the model was diffused by imitation throughout western Europe. In its integration of great height and illumination it had broken in a discontinuous fashion from the antecedent style of cathedrals. Yet, it was not wholly novel in every respect; it integrated non-Romanesque architectural features that had been employed outside the Ile de France by the Cistercians -- pointed arches, sequences of transverse oblong bays, buttressing arches visible above the roofs of the side aisles).3

Nevertheless, according to von Simson, the new model, could not have emerged by a great patron simply sending a master mason-agent out to visit and copy features of the design of existing great structures, such as the magnificent and extraordinarily large Romanesque cathedral in Toulouse.4 Gothic represented a new conceptualization (which we might liken to a radically new programming language, or the architecture of a new operating system, or a graphical user interface) which sprang from an act of creative imagination. Moreover, the implementation of that novelty required a break from the prior, long-established and empirically tested building conventions that were the province of master mason-architects of the day. Salzman (1952) offers what for many would be an eye-opening survey of how frequently medieval structures collapsed soon after their completion, and how central was the conservatism of knowledge acquired through such experiences."

3 The elaboration of the early Gothic model into three distinct (regional) schools during its diffusion in France, is traced by Jean Bony, French Gothic Architecture of the 12th and 13th Centuries, (Berkeley: University of California Press, 1983), where one will find splendid maps showing the distribution of particular features at different dates (pp. 110 ff.).

4 The features of that and other great Romanesque cathedrals are discussed by Roger Stalley, Early Medieval Architecture, Oxford,1999.
The innovator-hero of this story was Abbot Suger of St.Denis, who had undertaking the rebuilding of the Abbey of this royal monastery in the Ile de France, for which he had obtained (from the King) the gift of relics that were expected to provide the attraction necessary to make it a great center of pilgrimage. Abbot Suger's design for the rebuilding was meant to give symbolic substance to the idea of the fusion of the political power of the Capetian monarchy with the Church -- in a recreation of the unity of Charlemange's kindgom with Christianity. According to von Simon (p.141):

Precisely because it evoked the mystical archetype of the political order of the French monarchy, the style of St.-Denis was adopted for all the cathedrals of France and became of monumental expression of the Capetian fidea of kingship. It is not surprising, therefore, that in the cathedrals of Paris and Chartres, of Reims and Amiens, the royal theme, evoked not only in the Galleries of Kings but also in the selections of certain Biblical scenes and figures, is completely merged into the Christological one.

There are, consequently, some grounds for seeing cathedral building as an ideological movement, and in that aspect finding a rather distinct, non-organizational parallelism with the ideological impetus imparted to free/open source software communities by the Free Software Movement led by Richard Stallman, and discernable equally in the "revolutionary" overtones of Eric Raymond's reflections on the larger significance of Linux. Although some may find this a tenuous, if not quixotic connection to make, the parallelism seems worth noting when considering the role played in the mobilization of voluntary commitment to contribute to FLOSS projects by sentiments of "dedication" and of "solidarity" in a cause transcending individual interests. Modern scholars working in the social history of architecture historians of architecture – having been persuaded by Otto von Simson's *The Gothic Cathedral*, 3rd Edition (Princeton, 1988) – nowadays are inclined to accord considerable the importance to the ideology associated with Gothic Cathedral in propelling the diffusion of this new architectural form throughout western Europe.5

In response to the taunt that Gothic cathedral-building was an innovative, indeed a revolutionary departure of practice, infused with its own ideology, and not so tightly controlled as one might suppose, proponents of “the bazzar” model of

---

5 The details of that ideology are not of particular significance here, but it is perhaps best to notice that there is scant support for the interpretation sometimes advanced of “catherdral-building” as a pietistic undertaking that stood in opposition to other claims on resources, such as those devoted to campaigns of conquest such as “crusading.” According to von Simson (1988), Abbot Suger's work at St.Denis took place during the years 1127-49, and the last phase of building, involving the complete reconstruction of the nave, remained to be finished after his death. So this project was centered in the century that was marked by “the age of crusades”, and also of great pilgrimages; indeed, the crusades to the Holy Land were represented by the Church as armed pilgrimages that were more meritorious than other pilgrimages – including that to Santiago de Campostela. Suger was a man of his era, gripped by the idea of crusade, and disappointed when, having been made Regent of France, he was unable to accompany his king on the Crusade of 1147. Indeed, after the failure of that crusade, Suger prepared another crusade, with the support of St. Bernard – and when St. Bernard became ill and the promised subsidies for their undertaking failed to materialize, in 1150, he assumed responsibility for this new crusade and diverted his monastery's funds to that venture. He thereby delayed the completion of construction work on the nave of the Abbey. Such details do not fit easily into the view of (Gothic) cathedral-building as an activity that was spiritually antithetical to religious crusades, although it is true that in a world of finite resources all costly projects must compete for budgets.
FLOSS production might retreat to the position that the creativity of the pioneer builders of Gothic cathedrals in France eventually came to be stifled by the rise of corporate organizations in the building industry. But the idea that the rise of "corporations" put an end to creativity in cathedral architecture cannot be interpreted in so simple a way. Firstly, as has been noted, the Fre-Macons who interpreted and implemented the cathedral designs never formed proper guilds. Moreover, "professional architects" never became organized in (corporate) guilds. What does appear to have happened after the mid of the 13th century, at least in France, is the separation of responsibility for economic and financial aspects of cathedral building from the technical and artistic tasks, the latter being assigned to 'professionals' whereas, formerly, master masons would have retained managerial responsibility for both aspects.6

Metaphors and Realities of the Organization of Software Production

Raymond’s “proof” of the superiority of the FLOSS mode essentially relies on the proof of a related claim, stating that FLOSS is a violation to Brook’s Law, which itself states that the more programmers you add, the slower (paradoxically) the development process (Brooks, 1974)7. FLOSS is said to have found a way to a more efficient development methodology, by adding more programmers, as Microsoft itself has recognized (Vallopillil, 1998). Raymond says: “I think Linus’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model.”

The essential property here is that “debugging is parallelizable”, a property which economists should rely to a neglected property of software, i.e. its very low costs – and even risks – of testing and prototyping. This comes from the fact that software is easy and almost costless to reproduce, but also from the fact that there is no risk that a computer explodes when it is tested. Involving as many users as possible in software development according to an open-source model would then allow them not only to detect problems, but also to provide solutions, as long as there exist appropriate incentives.8

But is this development model really a bazaar-like model, as it seems? On the contrary:

First of all, bug corrections have simply to be integrated in newer versions of the software. Who’s in charge? More, the best solutions have to be selected, and more generally suggestions have to be sorted out because many of them are not relevant. Somebody – a person, as for Linux, or a committee with appropriate decision rules, as for Apache - has to be in charge of the regular release of newer versions: Alan Cox does this job for the ‘stable’ version of Linux, while Linus Torvalds does it for the development version. This handling of versions, and even this very split between stable and development versions indeed already denotes a very efficient and well-though organization.

---

6 On the traditional responsibilities of the master-mason, see Salzman (1952), especially regarding the situation in England. On the subsequent separation of responsibilities, see P. Colombier, Les Chantiers des cathédrales (Paris, 1954, p. 61).

7 Brooks’ work was also extremely influential and Brooks himself was even awarded the Turing prize for his leadership role in the IMB 360 project, a computer science equivalent for the Nobel prize.

8 Much more on incentives could be added here.
This is all the more necessary as the more important the suggested improvement, the more important the need for some leadership and coordination. Raymond’s reference to additions as ‘bugs of omission” does not make selection and handling of versions disappear, nor is it sufficient to note that Linus Torvalds is a “great leader”. As it goes, FLOSS needs a “great leader” in any case, to integrate new developments and to further reduce coordination costs so that it development becomes “parallelizable”.

This (hopefully) charismatic leader is also responsible for recruitment\(^9\). Raymond says: “Linus (Torvalds) was treating his users as co-developers in the most effective FLOSSible way”. To put it slightly differently, FLOSS project leaders recruit co-developers, and this is one of the key features of FLOSS projects.

Actually, not all users are co-developers, and there always exists a core team, at least because all FLOSS projects clearly need some kind of a kick-off. As a matter of fact, major confusions can be avoided here if we distinguish between kernel and obscure developers (Dalle & Jullien, 2001), i.e. between co-developers and users, in Raymond’s terminology. The contributions of a limited number of kernel developers even account for the most important part of FLOSS contributions (Ghosh, 2000).

And these co-developers are indeed part of an organization. Indeed, all FLOSS projects, and their success it we refer back to the initial failure of the Mozilla project, rely on FLOSS being highly modular, thanks notably to recent software technologies. Modularity allows to identify one sub-leader of each sub-project, reporting to the leader above him, and so on… FLOSS organizations finally tend to be extremely hierarchical.

The fact that debugging is parallelizable relies on an appropriate organizational structure. Leadership is absolutely critical, as is recruitment and modularity. The fact that coordination costs are reduced by an appropriate organization does not mean that there is no organization at all. In this respect, Brooks’ Law is not completely violated by FLOSS. Brooks’ Law is simply proven to be valid in a particular organizational context, namely the one of an existing firm.

FLOSS is not at all a bazaar, and ‘bazaar’ is even an extremely misleading word. FLOSS as a development methodology relies on an appropriate and efficient organization. FLOSS advocates tend to be libertarians, and come often close to ultra-liberal positions, neglecting organization and even here, central planning. As a matter of fact, FLOSS relies on a “benevolent dictator”, as they say, and also, we should say, on a sort of a nomenclature. Calling after self-organization (Kuwabara, 2000) is here also misleading, as it certainly does not explain the organization of FLOSS. Certainly, aggregation phenomena have been shown to display richer forms than should be expected, endogenous orders (see e.g. Dalle, 1995, 1997). But here it is much more about a visible hand which it hidden.

But reference to self organization leads to a very interesting question here, which is how organisations emerge out of the interplay of individuals. As a matter of fact, the OS methodology is so efficient because it allows it to build very efficient cathedrals. In a way FLOSS, might be seen as all about organization, since FLOSS communities build pFLOSSibly not Gothic, but Roman Cathedrals. And this is all the more important as results about the competition between open-source and

---

\(^9\) Appropriate leadership also reduces the risks of code-forking, while insufficient leadership might have been the cause of BSD forking and limited success (Lerner & Tirole, 2000).
proprietary software (Dalle & Jullien, 2000, 2001) show that organization is the essential feature which account for FLOSS success.

Indeed, FLOSS communities not only build hierarchies, they are beginning to establish Foundations to deal with copyright assignment issues, assist in the enforcement of licensing terms, manage relationships with sponsoring business firms. Moreover, they also help build ancillary firms – as in the case of the Apache community’s relationship with RedHat – which may prove to be a key condition in the long-term viability and performance of this mode of software production, because it creates a feedback connection with a broader and less expert population of end-users of the products.

Thus, whatever their private, profit-making goals might be, the activities of these ancillary firms serve a number of important functions that contribute to the “viability” of FLOSS as a mode of production. They may correct “FLOSS failures” in providing user-friendly programs for in-expert users, they promote the adoption of FLOSS system by undertaking commercial distribution (e.g., the role of Red Hat in relationship to the Apache community), and, more generally, they provided direct economic incentives for participating in FLOSS projects by creating dual employment opportunities in “VA-FLOSS” activities—selling value added open source software products and services.

The proliferation of forms of licensing beyond the basic GNU General Public License is itself a reflection of the efflorescence of what some might regard as symbiotic commercial opportunities – even if others would, in the spirit of the Free Software movement, regard them as “parasitic” on the body of free and open software.

What is possibly the most amazing achieve of FLOSS and FLOSS communities is their ability to build efficient organizations.

Why is it so then? Let us first suggest three tentative explanations.

First, the architecture of organization mimics the one of software, as Raymond himself mentions. One leader per module, for instance. The structure of the organization and the structure of the code would be similar. Indeed, software designers are called architects, and Bill Gates is now Chief Software Architect for Microsoft: great software designers are experienced at designing and building efficient cathedrals. Being both great architects, Linus Torvalds and Bill Gates simply do not build the same cathedrals.

Second, the Internet allows for reduced transaction costs: according to Coasian economics, they allow for a less integrated organization than a firm, which happens to be more efficient since looser but more efficient relations become feasible.10

These two explanations can be combined, since FLOSS allows to build an organization from scratch: then you can built the organization exactly according to the code that is itself being built. You do not have to deal with an existing organization and long-term contracts with employees, and emphasis is often put also on the ability of FLOSS to self-select the best programmers. Simply, you get rid of switching costs (David and Bunn 1988; David and Greenstein 1990). You build a new organization for each new project, as is already the case in the gaming and media industry, where bigger firms and studios rely on the creativity of smaller

10 NOTE: This point will perhaps bear further elaboration in a later draft.
firms created for each particular game or film. FLOSS is therefore indeed close to pure flexibility here, notably labor flexibility: a model that FLOSS advocates might possibly not be willing to plead for in other contexts.

This is especially important because, contrary to what is often said, almost nothing before now would have prevented a similar development model to be implemented and indeed replicated in a pure market environment. It would be close to the gaming and movies industries, and slightly different because of the absence of IPR: we would have distributors which would rather be service firms. As a matter of fact, this is already an on-going process.

A counter-argument may be offered against the proposition just advanced, however: the limitations of creativity within modern corporate culture. In other words, it is claimed that FLOSS attracts the most creative programmers, who would not offer their talents as employees in commercial organizations. This is certainly a key controversial issue. We should simply note here that there has been great software success due to business firms, and to the very creative guys that they had hired. And if they are able to reach de facto monopoly positions (David, 1987), firms are also able to pay their employees more.

Anyway, another explanation might potentially much more interesting. FLOSS organizations are continuously improving their efficiency because FLOSS communities generally rely on open knowledge conventions (see Dasgupta & David, 1994; David & Foray, 1995), once again supported by the Internet and related technologies. Every feature of FLOSS is indeed discussed in forums and sites, such as slashdot.org: licences, the role and strategy of business firms, be they ancillary or potential predators, the very organization of FLOSS projects, the role of leadership, decisions rules, the identity of leaders and so on. Everything, not only software.

FLOSS communities are better at problem solving because they rely on open knowledge, not only for software itself, but also for every relevant feature, and notably for organizational ones. Learning and problem-solving is therefore considerably enhanced. This last feature would clearly be extremely difficult to replicate in an other environment, and specially by business firms.

Because of this open-knowledge-based organizational building, and also to take further advantage of it, FLOSS communities would probably want to attract also non-computer people to speed up their learning curves and ability to problem solving. This would certainly involve economists, legal scientists, and generally business people. We do not feel that enough of these people are interested in this subject for now, although their number is certainly growing.

Possibly also, FLOSS communities are not necessarily “open” enough to such people. They might sometimes be feeling like they will find the best solution anyway. It might indeed be so in many respects, but possibly not for all of them. And, even if it was true, it might also be often too late, as FLOSS is most of the time competing against proprietary software: and, once again, it is critical for FLOSS communities to be able to improve their organization quickly enough (Dalle & Jullien, 2001). In this respect, FLOSS communities could even consider increasing their openness!