A model of a mature open source project

London School of Economics

MSc Analysis Design and Management of Information Systems

Dissertation 2004/05

459134

Abstract

Many perspectives have been offered in the academic literature to explain the phenomenon of open source software development. Unfortunately, most studies fail to adequately account for the depth and social complexity of the communities found at the heart of mature open source projects. Through an in-depth interpretive case study of Plone™, a vibrant open source project, as well as an evaluation of the existing literature on open source, a model of a mature open source project is proposed. The model incorporates concepts such as the project's life cycle, community-of-practice, culture, and business- and end-user orientation. Limitations to this model are acknowledged, and avenues for future research to improve theoretical accounts of the social dynamics of open source communities are suggested.

Acknowledgements

The author would like thank the entire Plone community for providing a wonderfully rich and exciting research setting. In particular, the members of the community who responded to the author's survey and interview e-mails are gratefully acknowledged: Alexander "limi" Limi, Alan "runyaga" Runyan, Paul "zopepaul" Everitt, Joel "joelburton" Burton, Stefan "lurker" Holek, Helge "tesdal" Tesdal, Geir "elvix" Bækholt, Alec "alecm" Mitchell, Florian "fschulze" Schulze, Hanno "hannosch" Schlichting, Matt ""HammerToe" Hamilton, Jens "jensens" Klein and Matt "mattl" Lee, as well as two anonymous contributors. The author is also indebted to Beatrice During for her insights on the dynamics and importance of "sprint"-based development.

Alexander Limi is gratefully acknowledged for having helped proof-reading and fact-checking earlier drafts of this paper. The author would also like to thank his supervisor, Jannis Kallinikos, for providing exactly the right amount of "hands-off" guidance.

Table of contents

Introduction

Since the widely-publicised rise of the GNU/Linux operating system, the open source movement has received significant attention in the mainstream press and academia. Spurred by an enthusiastic evangelist literature and tangible, even impressive software artefacts, journalists and academics reported from the frontiers of a new digital revolution, driven by community values and idealism. The idea that high-quality, complex software could be produced by global teams of volunteers seemed baffling at first, and a variety of theories in fields such as economics, sociology and information systems were put forward to explain the motivations and dynamics behind open source software. Unfortunately, most accounts were somewhat lacking.

The concept of open source is much older than GNU/Linux and the ecology of software that has since been written to run on this and related platforms (for a brief timeline, see Markus et. al., 2000). Fundamental to the open source concept is a "hacker1 culture" (Hannemyr, 1998), and the focal point of community – not just a single community, but a complex network of overlapping and interacting communities and sub-communities (Tuomi, 2000). However, it is a common misconceptions in the literature (e.g. Krishnamurthy, 2002) that there is "one" open source community of developers, that operates in one particular way.

This misconception can perhaps be traced back to Raymond's (1999) influential and widely cited essay "The Cathedral and Bazaar". The problem is that this work is part evangelising about open source, part advocating an approach for open source practitioners and part a description of its author's own open source project, *fetchmail*. To those already involved in open source, some implicit limitations of the paper as a description of open source in general are obvious and negligible, but to those not already in the know, it may not be. In particular, studies such as (Krisnamurthy, 2002) claim to refute some of the open source ideals by discovering that "most" open source projects generate relatively little community support and are in fact the work of one or two lone developers. The truth is that it is no more sensible to talk of "one" mode of open source development than it is to talk of "one" approach to commercial software development, or indeed "one" way in which any task-oriented community may be organised. A lone developer may for instance create a very specific piece of software out of personal need, but decide to share his efforts with the world through well-

[1] The word is used here in its original form, to denote a highly skilled software developer who "hacks away" at code. Open source developers detest the use of the word "hacker" to denote a software criminal, preferring instead the word "cracker". See (Hannemyr, 1998) for further details.

established and near-zero cost distribution channels such as sourceforge.net, either in the hope of receiving contributions, or simply because knowing that others use his software is gratifying2.

It could be argued, however, that the more interesting projects are those which *do* manage to create a self-sustaining community of contributors. In these projects, social processes are embedded in culture and artefacts (e.g. the software being built) to create a self-sustaining community that allows for contributors of different backgrounds and motivations to come together to produce a software product (Tuomi, 2005; Markus et. al., 2000; Franck and Jungwirth, 2002). The quality of this software tends to be high, and the distribution of control and responsibility across the community makes the continued health of the project much more likely. This in turn attracts business interests, which, if the community is poised to absorb them, feed back into the community, strengthening it further (Dahlander and Magnusson, 2005).

The two most widely used case studies for open source, Linux and Apache, are indeed such projects. However, perhaps because they are both creating system-level, back-office software, which attracts a certain kind of developers and big-business interest where the social vibrancy and complexity of the community may be less easily visible, most accounts tend to focus on very specific parts of the workings of the projects, such as co-ordination processes (e.g. Mockus et. al., 2002) or individual motivation (e.g. Lerner and Tirole, 2002). Only recently have deeper accounts of the social underpinnings of open source begun to surface (e.g. Tuomi, 2005; Watson et. al., 2005; Lee and Cole, 2003), but even here, an understanding of what exactly the open source community *is* – how it is composed, how it functions – is hard to grasp.

The contribution of this paper, therefore, is to evaluate the main perspectives on open source in the academic literature, and to propose a model that can bring their ideas together, allowing one to holistically understand the composition of a mature open source project. In order to do so, an in-depth case study of a particularly vibrant and successful open source project – Plone[3] – will be presented along with the literature, and used to infer a theoretical model for further empirical testing. Such a model should not only assist researchers in better understanding the interaction between the various aspects of a mature open source project, but also help practitioners of open source reflect upon their own processes.

---

[2] The author has written several such pieces of open source software, for example hprofile, found at http://hprofile.sf.net

[3] Plone's home page can be found at http://plone.org. Most community activity happens through the mailing lists and chatroom. See http://plone.org/contact for more.

Background: About the Plone project

Plone is a content management system (CMS), which allows organisations and individuals[4] to manage their electronic content, such as documents, files and images, through a web interface, and publish selected parts of their content to the internet or an intranet. Additionally, Zope[5], the application server upon which Plone is built, is well-suited for creating other content-centric applications. Because Plone has an attractive, usable and extensible user interface and provides many of the common services such as authentication, search and topology management that are needed in most business applications, Plone has also become a development platform, upon which a multitude of small consultancies build customised solutions for customers around the world[6]. The open source license and spirit of the Plone community ensure that much of the work these companies do for their clients finds its way back into the Plone core or free add-on components, further advancing the quality of the software. To paraphrase Plone co-founder Alexander Limi, collaborating on the base platform in an open source fashion is the only way in which an endeavour such as Plone could ever hope to compete with "behemoth" commercial competitors such as Vignette or Documentum (see *Appendix A*).

However, as expressed by several of the interview subjects of this research, "*Plone* is the community". The technology could be changed radically (and indeed has, and will, over time), but it is the community of Plone developers, consultants, leaders and contributors that truly defines the project. From being founded in 2000 by Alexander Limi and Alan Runyan as a usability layer on top of a the Zope Content Management Framework (CMF), Plone has grown into a mature, user-oriented application with a plethora of add-on products[7] for functionality ranging from visual layout tools to the drawing of phylogenetic trees for biological research. At the same time, community processes have matured significantly, with a near full-time release manager, a lightweight and standardised process for the proposition of new features (PLIPs - Plone Improvement Proposals), and regular social events such as "bug days" – co-ordinated efforts to fix bugs in the software – and "sprints" – real-world session of intense development effort in small groups for several days. Additionally, Plone has an annual conference, and the non-profit, community-run Plone Foundation has recently been established as a legal entity which owns the rights to the core Plone source code

[4] In fact, the author used Plone to manage his research materials for this work.

[5] See http://zope.org for more.

[6] See http://plone.org/about/sites for a (far from comprehensive) list of sites that use Plone.

[7] See http://plone.org/products for an overview.

and has the ability to represent Plone's interests, for example if a company should wish to license Plone under different terms than the standard open source General Public License in exchange for money, or in case of any dispute over ownership or liability.

A full description of the culture, friendships, personalities and social processes that are found among the project's most active members could easily fill the allotted word limit of this paper. The reader is strongly encouraged to read the surveys and interviews found in *Appendix A*, in particular the responses of founders Alexander Limi and Alan Runyan, Joel Burton, president of the Plone Foundation, and Paul Everitt, founder of the Zope Europe Association and Plone Foundation chief executive, to gain a better appreciation of how these influential figures view Plone.

Methodology and research design

Orlikowski and Baroudi (1991) comment on the prominence of positivist research in the field of information systems, and suggest that more interpretive and critical research is needed to build better accounts of organisational IS. Chen and Hirschheim (2004) suggest that the situation has since improved, but that a positivist epistemology still dominates. Certainly, the author's survey of the open source literature reveals a strong bias towards positivistic articles, even among those which attempt to understand the social processes of the community through case study research (e.g. O'Mahoney and Ferraro, 2004; Cummings, 2002; Giuri et. al., 2004). This is perhaps related to the relative youth of this body of literature, but such methods are criticised as being inappropriate for gaining an understanding of the culture and depth of the social worlds of communities (Rosen, 1991), of the *why* and *how* of their functioning (Bensabat et. al. 1987) – the very subject of this paper. Hence, this study adopts an interpretive case study approach (Galliers, 1991).

Klein and Myers (1991), Walsham (1995) and Rosen (1991) all emphasise the importance of explicitly stating the role of the researcher in interpretative research. The author has been a part of the Plone community for nearly two years, and is a core Plone developer, documentation writer and freelance consultant. Through daily participation in the Plone mailing lists and chatroom, the two main communication channels of the project, as well as attendance at one conference and two "sprints", the author has gained a deep understanding of the social processes and composition of the Plone community. At the same time, personal friendships have formed with several of the core members of the community. This could be viewed as a source of bias, but it also permits the author to turn the lens of analysis on himself: How is it that through what on the surface is an essentially task-oriented community, richer social bonds could form? What is the significance of these bonds?

To widen the perspective of the research, combined survey/interview e-mails were sent with six common and three or more subject-specific questions to various figures across the Plone community (see *Appendix A*). Members of the community were also given the chance to comment upon some of the author's descriptions, such as those in the *Background* section above, to ensure that there were no obvious discrepancies. Hopefully, these measures, as well as the author's vigilance as researcher (as prescribed e.g. in Klein and Myers, 1999), will have helped mitigate any bias the author may have introduced to the research[8].

[8] The vigilant reader is invited to search Plone's mailing lists and chatroom logs for the author's full name and online handle "optilude" to assess his involvement. Please see http://plone.org/about/contact

This approach places the research methodology somewhere between the realms of participant observation (Walsham, 1995) and action research (Galliers, 1991), with survey and structured interview methods used as additional support. Although the study will not be presented in a purely ethnographic form, primarily for reasons of space (Rosen, 1991), the focus on culture, and the application of longitudinal observation and the "hermeneutic circle" of interpretation and re-interpretation through the accounts observed and received (Klein and Myers, 1999) mean that the study also draws on organisational ethnography, both in the traditional sense (Rosen, 1991), and in the virtual realm, in which ethnography is not necessarily any simpler or less detailed than in the "real world" (Paccagnella, 1997).

In terms of proposing a theory, it must be noted that generalising from interpretive case studies is less straightforward than generalising from positivist, statistical methods (Walsham, 1995). However, the case study approach has the potential to generate novel and empirically valid theories (Eisenhart, 1989), and useful generalisations can indeed be made (Walsham, 1995; Klein and Myers, 1999). Three forms of generalisation will be used in this study, according to the taxonomy proposed by Lee and Baskerville (2003). In the next section, theoretical-to-empirical generalisation will be used to determine if the theories proposed in the literature can adequately describe what is observed in the Plone project. In addition, empirical-to-theoretical generalisation will be combined with theoretical-to-theoretical combination of concepts in order to propose a model for a mature open source project, in the final section of this paper.

Perspectives in the open source literature

The romantic view of open source, as put forward by evangelist literature such as (Raymond, 1999), is that a large number of skilled volunteers will come together to create highly efficient, near bug-free code through a process of peer review and decentralised co-operation. Several authors (e.g. Giuri, 2004; Krishnamurthy, 2002) have pointed out that a majority of open source projects fail to attract a large number of contributors, though as noted in the introduction, this is not necessarily problematic. However, examples such as Linux and Apache have demonstrated that complex, high quality software *can* be built in an open source fashion, spawning a series of articles from a software engineering perspective about the specific processes that take place in open source projects and what can be learnt from them (e.g. Mockus et. al., 2002; Crowston et. al. 2004a; Feller and Fitzgerald, 2000). Certainly, the ideals of transparency in process, code sharing (via repositories such as the *Collective*[9]), peer review (via automatic code modification notifications and vigilant automated testing), and open communication (primarily through the project mailing lists and chatroom) are evident in Plone. Such processes are perhaps necessary for any global, virtual and action-oriented community to function (Watson et. al., 2004).

Theories of motivation

Fuggetta (2003) suggests that most of these processes could be applied to proprietary software development as well, but that the open source method may be an effective way of allowing a *combination* of such factors to come together. Tuomi (2005) and Markus et. al. (2000) assert that the ability to support multiple forms of motivation is a particular strength of the open source model. Franck and Jungwirth (2002) theorise that two types of contributors are found in the open source world: those who contribute to "invest" in reputation or learning, and those who "donate" for altruistic or ego-gratification reasons. They suggest that the novelty of open source, in particular the license that ensures the software remains perpetually free (as in speech), is to permit these two types of contributors to come together, and that "donators" are necessary to boot-strap a project in order to make it interesting to "investors".

According to Alexander Limi, Plone attracts a healthy balance of ideologically-driven volunteers and investors – both in reputation, and in the software itself as a platform for further customisation. The interviews in *Appendix A* reveal different motivations among Plone contributors, but perhaps most striking is that the same developers seem to be driven by both "investor" and "donator"

[9] *The Collective* is an open code repository for Plone add-on components: http://svn.plone.org/collective

rationales. For example, Alec Mitchell claims working on Plone is a semiconscious career move as well as personally gratifying, whilst Florian Schulze and Hanno Schlichting list learning, the joy of community participation and personal need for the software as reinforcing motivations.

Motivation, particularly of the "investment" kind, has also been examined in the field of economics to explain why rational, self-motivated individuals would provide open source software as a public good and not free-ride. Lerner and Tirole (2001; 2002) explain the basic economic costs – principally the opportunity cost of time – and benefits – personal use of the software, ego gratification and delayed rewards such as career signalling or peer recognition – of open source development. von Hippel and von Krogh (2003) posit that open source development constitutes a "private-collective" innovation process, where innovation is achieved by users, not producers, and point to intangible personal rewards, such as social benefits, learning or a sense of ownership, as motivation for such innovation. Bitzer and Schröder (2005) provide an economic model of the private provision of an open source public good, demonstrating that while there are incentives to free-ride, contribution offers greater benefits to some actors, and observe that good programmers will tend to attract other good programmers to a given project.

However, relying purely on economic explanations of motivation may be problematic, as their positivistic epistemological underpinnings mean they fail to adequately account for social realities where value systems may be unstable (Tuomi, 2005), and motivation consists of a mixture of rational and socially determined factors (Watson et. al. 2004). The aforementioned blurring of the lines between "investor" and "donator" motivations in the interview subjects could be seen as an indication of this. Additionally, most open source projects, Plone included, have a primarily internal status orientation (Reagle, 2003), which means that the relevance of career signalling incentives is likely to be lower than what is implied in the economic motivation literature. Proving oneself in the Plone community is indeed a way to signal skills, which have lead to career benefits in many cases (including for the author), but outside the sphere of the Plone community, the investment in reputation is likely to yield significantly lower value for all but the most prominent figures in the community.

Project organisation

Based on descriptive and prescriptive research on open source development methodology, several authors have studied projects' collaboration processes and internal organisation (e.g. Crowston and Howison, 2004). The common image of an open source project is that of a strong "core" of

developers, and a larger bug-fixing and testing periphery. Mockus et. al. (2002) hypothesise that a core of 10-15 developers will create about 80% of new functionality, and if the core group grows beyond this, strong code-ownership will be necessary to manage complexity, a view consistent with Kraut and Streeter's (1995) analysis of the importance of informal co-ordination mechanisms in (commercial) software development. This has given rise to "layers of the onion" models such as the one proposed in (Crowston and Howison, 2004):

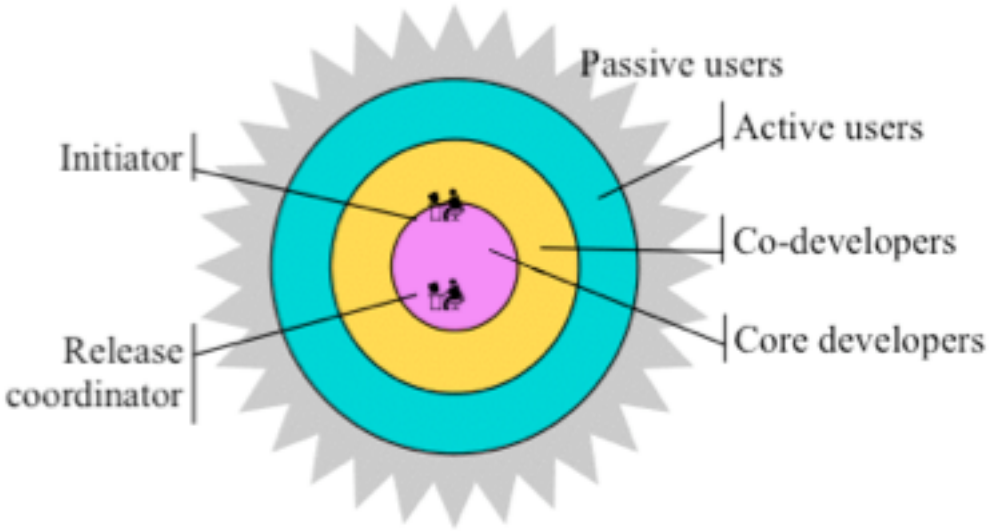

Figure 1: Model of an open source project's organisation. (Crowston and Howison, 2004, pp. 7)

Although the core group at the centre of the model typically receives the most attention, Zhang and Storck (2001) note that *cumulatively*, the peripheral group can have a great impact. Madanmohan and Navelkar (2002) also emphasise that there are many, varying roles in a typical open source project, such as core organisers, experts, problem posers, implementers, integrators, institutionalisers and philosophers.

In Plone, there is indeed a strong core of developers, although the set of "active" developers tends to change over time, as developers' professional and personal circumstances change. For example, the interviews with Florian Schulze and Alec Mitchell reveal how their involvement was sanctioned by personal need as well as "recruitment" in the community, and how they quickly attained "core developer" status by showing commitment. Joel Burton also emphasises the importance of a large community of users "cheering from the sidelines", promoting the project and reporting bugs. However, most bugs are fixed by the core developers, not more peripheral members as suggested e.g. by Mockus et. al. (2002). This may be due to the relative complexity of the software, as well as the fact that most core developers need Plone for their own work, and hence are highly motivated to correct defects quickly.

Decision making in Plone usually occurs through consensus, although figures such as Alexander Limi and Stefan Holek, the release manager, are sometimes seen making dictatorial decisions where necessary. The community appears to accept these, principally because they trust the leadership's judgement, and are always given a fair hearing. There are, however, concerns that such a model would not scale if the core group grew significantly (see Michlmayr, 2004; Mockus et. al., 2002), and the community has begun the process of moving to a team-based structure, where different teams would be given responsibility over different aspects of the software.

The organisation of open source development has also attracted interest from economists who view the production of the software artefact from a transaction-cost economics perspective. Demil and Lecocq (2003) suggest that a new "bazaar" form of governance is evident in open source software production, distinct from the traditional market, hierarchy or clan governance models in the theories of Ronald Coase and his followers. In this model, both incentives and transaction costs are low, but whilst uncertainty is high, it is counter-balanced by a high number of adopters and the relative ease of diffusion of innovation. Co-ordination is therefore based on the features of software, not the price mechanism.

Watson et. al. (2004) extend this theory, but prefer to use the term "community governance". Criticising purely economic rationality as too limited, they propose to add a social dimension to transaction cost economics through the concept of *transaction benefits*. Certain benefits, such as reputation, collegiality, intellectual challenge or self-esteem, are gained from the contribution transaction itself. These benefits, they argue, will determine governance where transaction costs are low and there are no direct profits to be made from the resultant artefact, as they are the *only* source of "profit" for contributors, in line with the delayed benefits argument by Lerner and Tirole (2002). Individual characteristics, cultural and regulatory factors, as well as the relevance of the underlying technology in terms of learning and skills signalling will influence the intensity of transaction benefits. Four modes of community governance are suggested, determined by the intensity of transaction benefits and the responsiveness of the community, of which open source communities fit the "inventive", meritocratic model, focusing on learning, skills transfer and local adaptations.

An evaluation of the transaction costs and benefits of Plone development is beyond the scope of this study, but the concept of transaction benefits could certainly be used to explain contribution in terms of both career signalling (e.g. in the case of Alec Mitchell), and contribution by firms. In addition to the aforementioned social benefits, which frequently extend to business partnerships or new clients, contributing a piece to the Plone puzzle earns a developer a deeper understanding and

a certain degree of ownership over that area of the code. Such knowledge could well prove a marketable skill. Hence, the value actors place on such these will at least partially determine who contributes what, when, although social pressures and other factors will play a part as well.

Communication

Although Plone members do meet in person quite regularly as part of "sprints", conferences or through personal friendships, the majority of communication happens online. Much of the literature on computer-mediated communication and knowledge management suggests that text-based communication is "lean", and thus poorly suited for communicating tacit knowledge and social cues (see e.g. Kraut and Streeter, 1995), making virtual communities more action-oriented and less intimate (Yamauchi et. al., 2000). However, Walther (1995) suggests that over time, asynchronous communication helps equalise people's ability to express themselves and socialise as they are better able to prepare their statements.

As open source developers tend to perform "pooled interdependent tasks" (Watson et. al. 2004), they also require tools and techniques to aid group awareness (Gutwin, et. al., 2004). A small core of developers is the easiest way to achieve this (Gutwin, et. al., 2004; Kraut and Streeter, 1995), but technical solutions such as asynchronous mailing lists, real-time chat, openly accessible source code repositories and automatically generated modification notification emails also help Plone developers stay aware of each other's activities. Gutwin et. al. (2004) emphasise the importance of making sure developers have time to read mailing lists and stay on top of other developments. In Plone, reading "the list" and chatting in "the channel" seem part of most core developers' daily life. However, on a deeper level, personal and professional relationships are just as important. Alexander Limi is "the UI guy", and when there is a UI issue to be resolved, the developers will ask him, or at least look at the work he has been doing. Developers will also call each other, sometimes across the Atlantic, to resolve urgent issues.

Business interests

By its nature, Plone attracts a fair amount of business interests, and Alexander Limi estimates there is about a 50-50 split between commercially-backed and volunteer contributions, which reinforce and challenge each other, driving the software forward. Having professional software developers at the core may also be a determinant of success (Healy and Schussman, 2003).

Most Plone companies believe in contributing to the core project whenever possible (see interviews with Joel Burton, Alexander Limi, Helge Tesdal, Geir Bækholt, Alan Runyan and Jens Klein), with the understanding that growing Plone and the Plone user base is better than trying to grab market share from each other (Joel Burton). History shows that major new pieces of functionality usually only get completed when sponsored, because creating sufficiently generalised and scalable solutions typically requires a more focused effort than developers can muster in their spare time. When sponsorship is not an option, "sprints" are organised to complete important missing pieces.

Bonaccorsi and Rossi (2003b) suggest that commercial ventures are the best way of completing "non-sexy" work in open source projects. A certain degree of social pressure and sense of responsibility also motivates volunteers to complete "non-sexy" work at times. Between larger development efforts, firms and volunteers contribute smaller components and bug fixes on a regular basis.

Several authors (e.g. Karels, 2003; Markus et. al., 2000; Bonaccorsi and Rossi, 2003a) describe common strategies firms employ to profit from open source. Primarily, these rest on Raymond's (1999) suggestion that the software industry is a service industry under the illusion of being a manufacturing one, and include such options as dual open/commercial licensing, or selling support, add-on components or consultancy. Plone solution providers usually operate either by selling add-on components (e.g. Enfold Systems, owned by Alan Runyan), or, more commonly, by building bespoke solutions on top of Plone (e.g. Plone Solutions, owned by Alexander Limi, Helge Tesdal and Geir Bækholt, see *Appendix A*). The Plone base platform enables these firms to compete against larger, closed-source rivals (see Alexander Limi and Paul Everitt in *Appendix A*). Indeed, open source has been identified as a means for small firms to afford innovation (Bonaccorsi and Rossi, 2003a). However, providing a community-backed solution also enables firms to lower the costs of maintenance and quality assurance. In the words of Tres Seaver[10], "every [proprietary] line of code in production is a liability". Without community support, the provider is solely responsible for ensuring that the entire software stack continues to work.

The relationships between "altruistic individuals" and "selfish firms" may be strenuous at times (Bonaccorsi and Rossi, 2003a). Dahlander and Magnusson (2005) classify firm-community relationships as symbiotic, commensalistic or parasitic, and Dahlander (2004) suggests firms may be "friends" or "foes" in the eyes of the community. Because Plone has had a business focus from the

[10] Tres Seaver is the father of CMF, a framework on which Plone is built. This statement was made at a panel discussion at the EuroPython 2005 conference in Göteborg, Sweden.

beginning (Alexander Limi, Alan Runyan), there are many Plone firms which operate on a symbiotic level, where firm and community gains coincide and a significant amount of time is spent managing the firm/community relationship. Other commercial interests are of a commensalistic nature, where the firm gains and the community is indifferent. Examples include both less community-oriented solution providers and larger organisations using Plone with in-house development expertise (see interviews with Matt Lee of the NHS and "Mr. Y"). There is little evidence of the type of covert code protection mechanisms described in (Henkel, 2004) to "get around" the open source license, or other parasitic behaviour. Interestingly, Alexander Limi in Europe, claims the license can be a help because an open and widely *understood* code base offers protection against lock-in (Fuggetta, 2003; Tuomi, 2005), whilst Alan Runyan, who represents a US perspective, often sees the license as an obstacle to selling Plone, because it introduces ambiguity in ownership and liability.

However, Plone's business model runs deeper than what is found in the literature, where presumptions of a single or dominant commercial interest and a firm/community dichotomy are usually implicit. In the words of Paul Everitt (see *Appendix A)*, founder of the Zope Europe Association[11] which aims to bring smaller solution providers together in order to be able to deliver solutions larger than any one of them could handle on their own:

> *15 years ago, people accepted at face value that only a multi-billion dollar company could make an operating system. The idea that disconnected volunteers worldwide could beat publicly-traded "grown-ups" would have seemed insanely absurd.*
>
> *Sometimes the conventional wisdom is utterly clueless. Sometimes, perseverance overcomes skepticism (sic), and an alternate history emerges.*
>
> *I see the same thing on the business side. People think that open source *business* requires expensive, cathedral "grown-ups" for integrators, to make that open source stuff "controlled" and safe. Might it be possible, though, to organize (sic) a market using the same principles and benefits from open source software? Could you provide scale, redundancy, choice, fairness, and invigoration in a business model by removing the cathedral?*

This model would probably only work if the developers and the community felt ownership of the product itself (West and O'Mahoney, 2005). Indeed, tasks such as ensuring a stimulating environment are tricky for commercial vendors to get right (Dahlander and Magnusson, 2005), and are probably best left to the community to self-organise (Markus et. al., 2000). Plone companies such as those of the founders are usually seen taking great care not to place themselves "above" the rest of the community in terms of ownership or control. The creation of the non-profit Plone

[11] See the front page of http://zope-europe.org for the ZEA manifesto.

Foundation, which role is to "protect and promote" Plone (see interviews with Joel Burton and Paul Everitt for details about the role of the Foundation) has also helped objectify and disambiguate issues of ownership and intellectual property rights.

Literally days before the deadline of this research, a prime illustration of the Plone business model was presented – the *Goldegg*[12] initiative. Through customer funding, certain project leaders will be paid to help bring Plone to the next-generation Zope platform, an undertaking which has sparked concern in the past as it is both necessary in the long run, and will require a huge effort. The Goldegg primer is careful to emphasise that control is still in the hands of the developers, but suggests that offering funding for Goldegg is an important way for Plone companies to promote themselves and show commitment to the next generation architecture, much in the same way code contribution affects the social capital of individuals. Companies which rely on Plone for their custom solutions are thus able to collaborate on investing in the platform, much like their developers are able to collaborate on producing the actual code.

The potential for free-riding is just as evident here as they are when considering developers' motivation to contribute, but the problem is also mitigated by the same mechanisms: The transaction benefits of participation as well as individual firms' needs may be sufficient for many to justify the investment. Further, as the same social webs that underpin the Plone community permeate those organisations most likely to benefit from such investment, firms are more likely to contribute in the understanding that other key players will too.

Social structures

Hence, what brings the accounts of motivation, organisation, collaboration and commercialisation together, are the underpinning social structures of the project itself. These can be viewed as a means of differentiation (Crowston and Howison, 2004) – there are dozens of open source content management systems; why should developers choose to rally behind Plone? Technical strength is of course a determinant, but in reality, it is rarely the only one, as a novel architecture can usually be replicated by other projects given sufficient resources. Embedded, self-reinforcing social processes are also the only way in which the continued viability of a project can be assured, even if its leaders or more active developers withdraw (Tuomi, 2005). When co-founder and architectural lead Alan Runyan begun to focus on his company and on value-added third party products, a gap in leadership appeared, but Plone did not suffer a crisis, because the community itself was capable of

[12] Goldegg is named after the castle of core developer and Austrian prince Phil Auersperg. See http://www.goldeggstack.org/about/primer for full details on this initiative.

making the necessary decisions (Alan Runyan) and re-focus its efforts (Alexander Limi). However, despite the recognition of the importance of social structures and processes, several authors have complained about a lack of models for understanding them (O'Mahoney and Ferraro, 2004; Watson et. al., 2004), and in particular their functioning at the micro-level (Lakhani and von Hippel, 2002).

Markus et. al. (2000) assert that for any "virtual organisation" such as an open source project to work, self-governance is key. Without it, fairness would be eroded and volunteers would leave. The management of membership, rules and institutions, monitoring and sanctioning of members' behaviour, and shared culture, norms and beliefs are vital components of self-governance, which in turn is a determinant of the strength of the social structures in place (ibid.). In order to build a community, various activities must take place, such as tending the collaboration tools, recruiting members, managing social dynamics and resolving disputes. These activities are undertaken both by "regular" project members and project leaders, although leaders tend to pursue them more actively (Butler et. al., 2002).

In fact, project leadership is of an ephemeral nature in open source communities. Although project founders typically will occupy a leadership position, this role is essentially granted only by the acceptance of community members. In projects where faith in the leadership is reduced, "forking" – a split of the code into alternative paths of development – may take place. Indeed, the possibility of "forking" can be viewed as an integral part of the social processes behind open source – essentially, it is a safeguard of individual freedom and choice (Reagle, 2003). Leadership may also be localised to specific modules or sub-projects. For example, Jens Klein (see *Appendix A)* is the current maintainer of Archetypes, a library related to, but not part of, Plone, as its creator is no longer actively involved in the project[13]. Several theories of how project members come to occupy positions of leadership have been proposed, such as the importance of occupying a structurally advantaged position (O'Mahoney and Ferraro, 2004) or spanning technical and social boundaries (Fleming and Waguespack, 2005), as well as the need for people skills and diversified technical skills (Giur et. al., 2004; Butler et. al., 2002).

However, leadership in open source is frequently a synonym for trust, with developers being asked and trusted to make decisions in areas where they have shown commitment and expertise. Trust can be based on recurring social interaction (Ardichvili et. al., 2003; Kimble et. al, 2000) as well as

[13] The lack of clear leadership in Archetypes is a cause of concern for many. There are indications that this could change, with recent discussion about a re-design of Archetypes by its original author, Benjamin Saller. In this case, it seems understood that Benjamin would take over leadership of this process, purely on technical merit.

action and contribution (ibid.). More generally, trust is linked to social capital, which can be used as a unit of analysis of social ties. Social capital is "a stock of active connections among people: the trust, mutual understanding, and shared values and behaviours that bind people as members of human networks and communities." (Daniel et. al., 2002, pp 1) Contributing and participating in a project is a way of gaining visibility and building social capital, which can be intrinsically rewarding as well as result in (delayed) reciprocal gains (Ginsburg and Weisband, 2002).

Zeitlyn (2003) uses the related concept of gift giving to emphasise this point further: Giving a gift – of code – implies an expectation of reciprocity. Thus, it is better to be owed a gift than to owe one. Giving a gift builds status and *symbolic* capital (ibid.). However, to be able to *accept* the gift in the first place, it is important that the giver and the receiver have a compatible mind-set, and shared values and mental models. That is, they must feel part of the same community.

Expanding the concepts of gift-giving and symbolic capital, Zeitlyn (2003) proposes that open source communities can be understood through a metaphor of kinship, where reciprocity is expected to balance out over time and members of the "family" do not question each other's commitment. Observing interaction in the Plone community, collegiality and kinship are readily evident among the core developers. When a familiar "face" signs on to the chatroom, other core developers will typically offer a friendly greeting, and core members ask each other for favours frequently. The kinship metaphor also applies to the treatment of "strangers". This is evident both in "playing host" to those showing an initial interest on the chatroom or mailing list, as well as in the interaction with "outsiders". Consider, for example, the difference in tone between the Plone community and the "outsider" defending DotNetNuke, a competing product, in the post asking for comparisons between these, in *Appendix B*.

Contrary to the romantic view of open source development as taking place only in cyberspace, Plone developers meet in real life regularly. Face-to-face meetings are important for building trust in virtual communities (O'Mahoney and Ferraro, 2004; Ardichvili et. al., 2003; Kimble et. al., 2000), and socialisation is a way for teams to build the shared mental models necessary for effective collaboration (Crowston et. al., 2004a).

More importantly, socialisation in the real world strengthens ties in cyberspace. According to Alexander Limi (see *Appendix A*):

> *Another thing that separates the Plone community from a lot of other communities is the amount of real face-to-face communication we have. We organize (sic) workshops, conferences, informal gatherings - and*

> *most people have a lot of close friends in the Plone community. One of the strengths of this approach is that we can have really heated discussions about things related to Plone - but people are very seldomly (sic) offended, since they most likely have met the person on the other end in person - and know that he's a human being, that he's a nice guy. There is a good separation between what is being discussed and the persons involved, and this makes for a healthy community. We can be the best of friends in real life, and still argue agitatedly about a particular part of implementation detail - without offending anyone.*

The author got to experience this first hand when he suggested to Stefan Holek, the release manager, on a public mailing list that perhaps the next version of Plone should be delayed to incorporate some user interface tweaks, and was met with:

> *F**k[14] off Martin ;-)*

No offence was taking from this, nor was any intended. However, it is unlikely such a response would have been elicited had the author not met Mr. Holek in person a few days earlier and got to experience his particular brand of humour. In retrospect, this experience markedly improved the social ties and commitments the author felt to the project (see *Appendix B* for the full excerpt). In the words of Hanno Schlichting (see *Appendix A*):

> *I would call myself part of the community but definitly (sic) not of the core group. Possibly you have to drink some beers in real live (sic) with other members to get that status ;)*

Learning, knowledge and communities of practice

At a deeper level, open source development can be viewed from a knowledge and learning perspective. Certainly, learning is a common motivation for contributing, and learning through "apprenticeships" takes place whenever developers of different skills collaborate on the same piece of software. For Plone, this is particularly evident at "sprints" and "boot camps", where new core developers are often recruited and shown the ropes. Through interactions in cyberspace, open source development is driven forward by knowledge sharing and experimentation, through processes of destruction, reflection-in-action and discourse (Hemetsberger and Reinhart, 2004). Knowledge constructed in this manner is intrinsically linked to the community and the identity contributors find there. Thus, meaning, as defined by the community and its culture, becomes embedded in the software artefact, recursively influencing future development and knowledge sharing (Tuomi, 2000).

Lee and Cole (2003) use the principle of Hegelian dialectics to examine a community-based model of knowledge creation. Through the discourse of criticism and code review, variety is created, which ultimately evolves *both* the software artefact and the community itself. This process is governed by

[14] In fact there were no stars in "F**k" in the original post

the *norms* and the *forms* of the community. The norms include quality assurance processes, the legal environment created through the license, and the use of culture as a social control mechanism. The forms are processes of criticism and error correction, experimental variation, public discussion, and a two-tier task structure of a small core, where most critique occurs, and a larger periphery, from which help may be solicited. In Plone, the discourse of criticism takes place principally on the mailing lists. Developers read the check-in messages generated whenever the software is modified, and where necessary will post to the mailing list offering critique (see *Appendix B* for an example). Proposed features are discussed on the lists and through the "PLIP" process. Being seen as offering constructive criticism is indeed an important part of reputation building (ibid.).

Lee and Cole (ibid.) suggest that the concept of communities of practice, as pioneered by Leave and Wenger, is a promising theoretical foundation for understanding community-based knowledge creation, and that whilst the original theories used the firm as the unit of analysis, there is no *a priori* reason why these principles could not be applied to open source communities as well. Markus et. al. (2000) also state that a large community of practice is a pre-requisite for a virtual organisation to work.

Tuomi (2000) offers an examination of the evolution of the community-of-practice concept, from Schön's assertion that professional practice is fundamentally reflective and occurs in communities of practitioners, to Leave and Wenger's formulation of *legitimate peripheral participation*, through which knowledge and reputation is gradually gained by peripheral members of the community via social interaction and participation, or learning-by-doing. Thus, learning is fundamentally about becoming an accepted member of the community: without some Plone skills, a participant will not be a true "Plonista". Identity, membership and expertise are therefore inseparable (ibid.). Brown and Duguid (1991) emphasise that learning in a community of practice is fundamental to innovation. Learning occurs through the processes of narration, collaboration and social construction via bricolage – making do with what is at hand. Certainly, all these processes are readily apparent in Plone's innovation process.

Wenger (2000) emphasises that successful organisations must organise themselves as social learning systems, by giving primacy to informal learning mechanisms, meaningful participation and receptiveness to complexity. These concepts lie at the very core of Plone: Informal learning occurs daily through the mailing lists and chatroom, participation is greatly valued and encouraged (see *Appendix B*) and the project's decentralised nature makes it well suited for dealing with complexity,

with the ability to re-factor different parts of the software into add-on modules or lower level frameworks where necessary.

Wenger (2000) further adds that organisations must consider three modes of belonging: *engagement*, e.g. in programming; *imagination* – the construction of an image of the community; and *alignment* of local activities to global goals, through the aforementioned co-ordination processes. Communities that define competence are built through events (such as "sprints" and conferences), leadership, statements of membership, learning projects, and artefacts (such as the software itself). At the boundary of the community and its environment, innovation occurs through discourse and brokering (c.f. Fleming and Waguespack, 2005). Finally, allowing community members to create an identity at the community's "home base", e.g. through their chatroom nicknames and mailing list signatures, can build enduring social relationships and trust.

Several authors have extended the concept of communities of practice to virtual communities (for an overview, see Johnson, 2001). Issues particular to virtual environments include the use of principally text-based communication, which can make it more difficult to communicate tacit knowledge (Hemetsberger and Reinhardt, 2004), but also help neutralise differences between introverts and extroverts. Additionally, there may be problems of attrition and unnoticed withdrawal (Michlmayr, 2004). The importance is trust is frequently emphasised, with real-world meetings offered as a way to build deeper social relationships (Kimble et. al., 2000). Kimble et. al. (2001) find that shared artefacts (e.g. the source code) and "war stories" are important means of building deeper social bonds in virtual environments. Such stories are frequently shared when Plone developers communicate in private, both on and off-line, such as at a recent conference, where over drinks and barbecue, developers shared stories about the "bad old days of bobo[15]".

Although virtual communities of practice have been enthusiastically received in the knowledge management literature, Schwen and Hara (2003) caution against romanticism them, pointing out that whilst virtual organisations may be designed, communities of practice are emergent by nature (c.f. Johnson, 2001). Virtual communities of practice are most useful for supporting knowledge-in-action by motivated members. This condition does appear to apply to Plone, however. Lueg (2000) also points out that in examining virtual communities of practice, it is important to consider *where* action is taking place. The actual "learning and doing" takes place in the real world, by real people. Kimble et. al. (2000) thus propose that the "complex geography" of the virtual community of practice is in fact very important. For example, Plone members in the same time zone are more

[15] An unfortunate earlier name for Zope, the application server on which Plone is built

likely to collaborate with each other, and those who live closer together will more likely attend the same conferences and "sprints", thus having greater opportunity to form closer social bonds.

Defining success

Taking a step back, Plone has been described as a successful and mature open source project, but of course this entails a value judgement. An understanding of how Plone fits into the various perspectives on open source processes and motivation as developed in the preceding sections allows the rich, multi-faceted operation of the project to be examined. However, some more objective criteria for calling a project a "success" or "mature" would be useful.

Naturally, most core developers believe Plone is a success, although the interviews and discourse in the community reveal that there are a multitude of technical and organisational issues which need to be resolved. In the words of Alexander Limi, "I believe we're just getting started [as a mature project]."[16] Perhaps just as important as having high-quality technical and non-technical system components, though, is the *recognition* that there are sub-optimal components and a commitment in the community to improve them.

Bonaccorsi and Rossi (2003b) list the three main problems in open source development as motivation, co-ordination and diffusion. Overcoming these could be seen as a measure of maturity and success. The preceding sections demonstrate how Plone copes with each one. In more concrete terms, O'Mahoney and West (2005) list some characteristics of a mature project: having made several releases – the imminent 2.1 release is, in the words of Alexander Limi in a recent presentation, "the first version of Plone that doesn't suck"; having adopted governance mechanisms that allow representation in commercial and legal settings – this is the role of the Plone Foundation; having developed an ecology of institutions to support developments – the various Plone consultancies, the Foundation and the Zope Europe Association are all examples of such institutions; and having an accepted governance structure – the semi-formal role of release manager is designated by the community, whilst governance in other areas is based on merit and knowledge. There are also accepted processes in place for proposing feature enhancements or architectural changes.

Plone does not have formal elections (except to the Foundation), but conflicts of control or authority are extremely rare. In fact, another measure of success and maturity may be a lack of "religious wars" (Fuggetta, 2003) and immature behaviour on the open mailing lists. The excerpt in *Appendix B*

[16] Said in a private chat message to the author, August 27th, 2005

of the thread asking for comparisons between Plone and DotNetNuke, a competing open source product, demonstrates this. The response of the Plone community is surprisingly calm, the only sign of agitation – and a marked difference in tone – being displayed by the "outsider" defending DotNetNuke. In the author's experience, many, if not most, open source projects are prone to a certain degree of "flaming" taking place on the lists on a regular basis, as contributors' personalities clash. The "friendliness" of the community is frequently cited as a reason for people to stick with Plone (see *Appendix A, questions 3 and 6)*.

More narrowly, project *success* could be defined in various terms, such as the size of the development community, project activity, bug fixing time or number of downloads (Crowston et. al., 2004b), or more qualitatively in terms of code and documentation quality, user ratings, user interest or developer satisfaction (Crowston et. al., 2003). Statistical examination of any of these measures is beyond the scope of this study, but qualitatively, it is relatively easy to assert that the development community is healthy and growing (see *Appendix A, questions 4-6)*, that the project is quite active, that most bugs are fixed relatively quickly, and that Plone is downloaded frequently. Quality and satisfaction is course harder to estimate, though the consensus seems to be that the code quality is generally perceived as good (especially as of Plone 2.1), but also that there is scope for improvement in many technical and some organisational areas (see *Appendix A, question 4, e.g. for Stefan Holek, Florian Schulze, Hanno Schlichting or "Mr. Y")*. Documentation quality is sometimes criticised, and there are efforts underway to improve documentation at the time of writing.

Bringing it together

A number of perspectives on what open source "is" and how open source projects function have been offered in the literature. However, wearing an open source practitioner's hat, one is often left with the sentiment, "yes, but...". To guide his initial research, the author produced the following mind map of some of the different aspects of Plone that come together to shape the community:

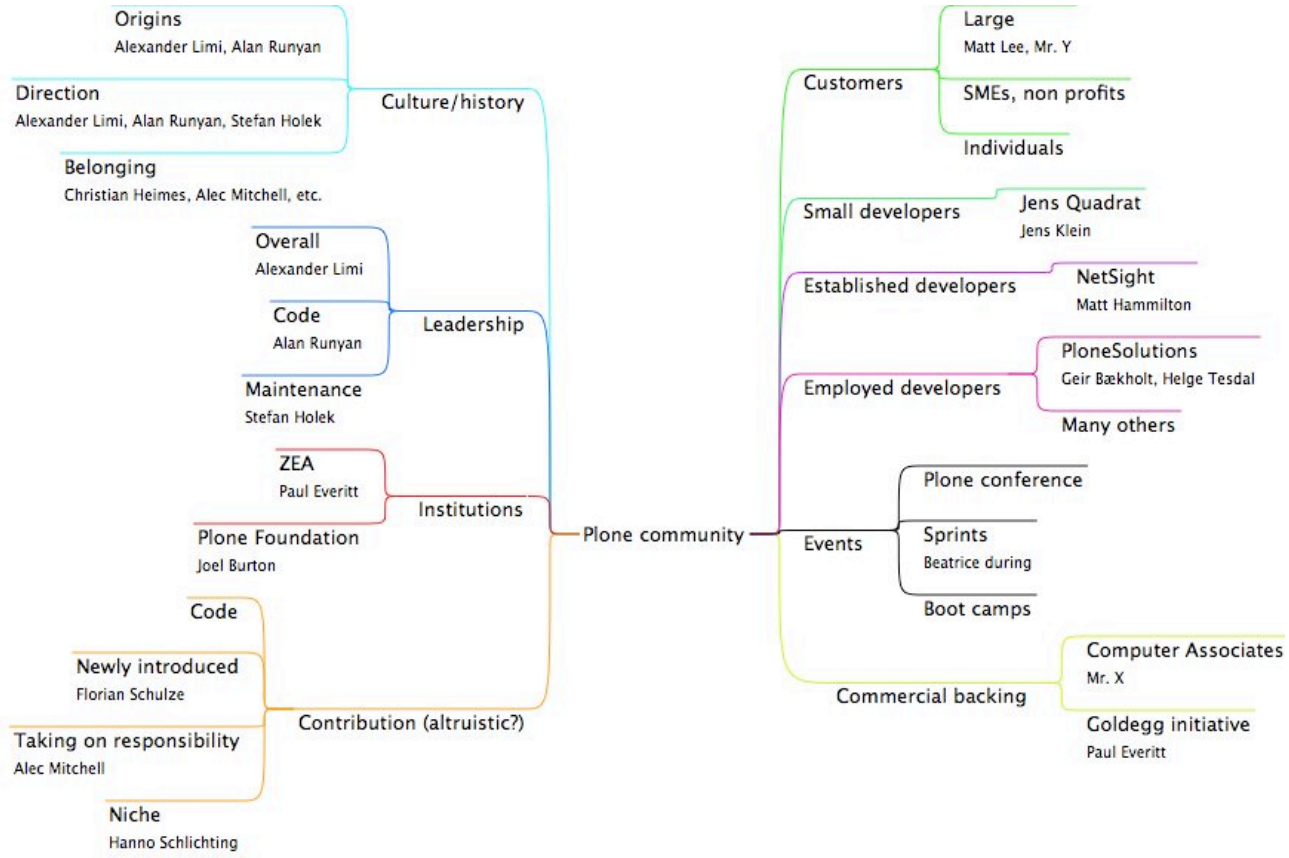

Figure 2: Elements of the Plone community

Undoubtedly, this could be expanded to many other dimensions. Do the accounts of open source in the literature truly allow for this complexity to be understood?

Elements of a mature open source project

In trying to answer this question, it is useful to reiterate the most important elements of a mature open source community, as described in the literature and as interpreted through the analysis of Plone in the preceding section. These elements will then be recombined to propose a theoretical model of such a community.

Project organisation and governance

The most common understanding of the organisation of an open source project is that of a strong core of developers and a larger periphery of users, bug fixers and bug reporters (Crowston and Howison, 2004). A two-tier task structure emerges (Lee and Cole, 2003), with some tasks delegated to the periphery, although the boundary between core and periphery is seldom precisely defined. Further, the composition of the core itself can be just as socially complex as that of the rest of the community, and can be subject to changes in leadership and participation, as well as the dynamics of evolving personal relationships.

Decision-making happens among the core, most active developers, and an accepted, meritocratic governance structure tends to emerge, rather than be imposed, based on trust and social capital. This in turn is based on complex social ties, which can be understood through metaphors of kinship and camaraderie (Zeitlyn, 2003). The governance is democratic in the sense that where trust is eroded and the perception of community self-governance is lost, developers will "vote with their feet" (Franck and Jungwirth, 2002) and leave or "fork" the project.

Additionally, mature projects frequently spawn more formal institutions to protect intellectual property, and represent and promote the project (West and O'Mahoney, 2005), such as the Plone Foundation. Other institutions that may be formed include business facilitation organisations like the Zope Europe Associations, or institutions representing key groups of stake-holders. However, in practice, ultimate control will tend to remain with the community and the core developers.

Culture and dynamics

The structure of the project is embedded in its culture and value system, emerging from the language, history, humour, norms and artefacts shared by the project's members (Markus et. al., 2000). Discourse shapes the beliefs of the participants, who in turn enact these beliefs upon the software, and upon the community itself (Lee and Cole, 2003). Knowledge is embedded in the software artefact, and disseminated to project members through informal learning systems as well as documentation. Culture and social webs also fuel participants' motivation to contribute (Tuomi, 2005).

Community cohesion is rooted in shared cultural beliefs. Fragmentation of stake-holder groups and differences in priorities and vision can be beneficial if a project's culture makes it receptive to dialectical processes of critique and experimental variation (Lee and Cole, 2003). Project members must be able to throw out an implementation when a better one is proposed, even if significant

effort went into the first attempt. Conversely, where egos are easily bruised and a willingness to resolve dispute for the good of the project is not implicit in the project's culture, "flame wars" will erupt, and more pragmatic developers may shun the project.

A dimension of culture which warrants particular examination is that of ideology. A number of open source projects are driven to a greater extent by strong ideological beliefs in the "fairness" of the open source model, and are suspicious of non-free software. Other projects are more pragmatic in accepting outside interests, but the fundamental ideology of the open source movement is by definition an important part of any open source project (Franck and Jungwirth, 2002).

Community of practice

Bringing structure and dynamics together, the concept of communities of practice holds promise in developing a deeper understanding of the functioning of an open source project (Lee and Cole, 2003). By examining modes of belonging, the community's definition of competence, the facilities for learning-in-action and legitimate peripheral participation, as well as the community's boundary and mechanisms for defining identity, a researcher can better understand how the community itself is composed (Wenger, 2000). In particular, it is important to identify if and how the community is designed as an informal learning system, allowing new participants to gain legitimacy and identity by acquiring expertise, as this is the primary means by which the community grows and evolves. Thus, the reciprocal relationship between identity, legitimacy and expertise is particularly relevant to open source projects (Kimble et. al., 2001).

In analysing a community of practice, one should also take into account the possibility that it, and hence the identities of individual practitioners, may span across formal organisations, such as firms or institutions connected to the project (Brown and Duguid, 1991). Sometimes, a dichotomy of identity inside and outside the project may be beneficial in bringing in new perspectives at the boundary of the community of practice. At other times, conflicts of interests may occur.

User and business relationships

The lines between consumer and producer are usually blurred in the open source innovation process, as developers are almost always also users of the software (von Hippel and von Krogh, 2003). However, for a project to be successful, it must also attract outside interest – "real" users, who are not programmers. In the words of Paul Everitt:

> *It is soooo (sic) hard to get a vortex created. Meaning, enough mass, buzz, governance, and momentum to separate yourself from the 400 open source CMS projects and actually have a shot at closing mainstream sales.*

When "real" users begin to show an interest, it creates a potential for business interests, too. Businesses may follow a variety of strategies, but for the purposes of understanding the community, they can be separated into two categories: Those which passively or parasitically leverage the software as consumers, integrators or service providers, and those which become an active part of the community itself, integrating their own communities of practice with that of the open source project (Dahlander and Magnusson, 2005).

Life-cycle

Finally, open source projects and products follow life cycles just like their commercial counter-parts. A consideration of whether a project is in the starting phases, trying to gain momentum, in a growing or mature stage, or in decline, will afford the researcher some ability to predict the size, cohesion and vibrancy of a community. Naturally, many projects never reach a mature stage, because the social processes and cultural factors described in the preceding sections are never initiated.

The model

Integrating these factors, a model of a mature open source project is proposed. The following diagrams shows the most important elements to be considered

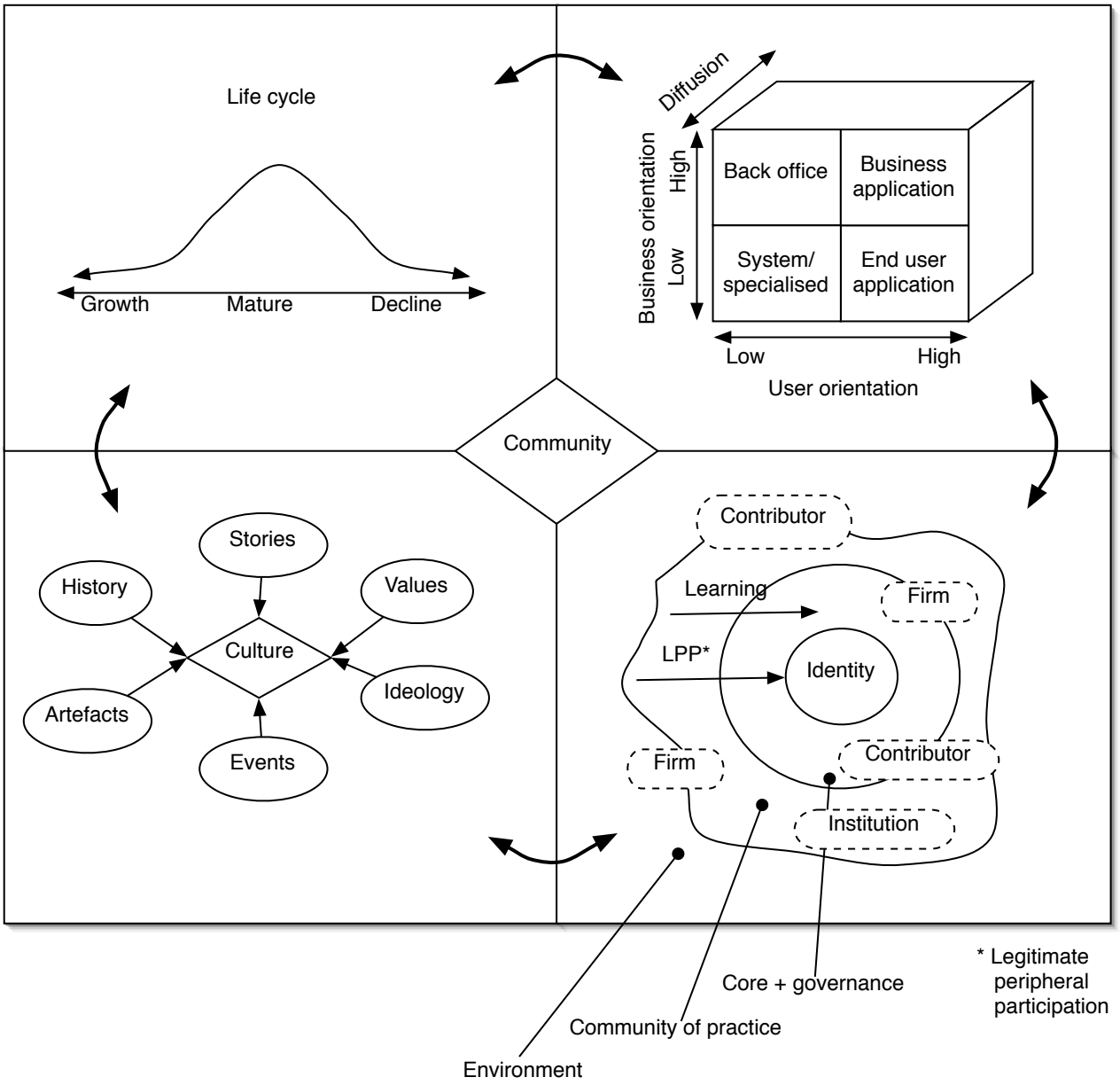

Figure 3: Model of a mature open source project[17]

A community of practice is presented as the canvas on which the community is painted. A two-tier task structure emerges from loose separation between core developers and peripheral participants, whilst informal learning systems are in place to aid peripheral contributors in gaining expertise and identity as part of the community. Knowledge is embedded in the software artefact, and processes of reflection-in-action, critique and experimental variation support the continual evolution of the

[17] The author would like to thank Alexander Limi for assisting with the visual layout of this diagram

software and the community. Individuals and business interests intersect the community of practice, exhibiting different degrees of alignment with the community itself. Complex social processes and culture are supported by an emergent governance structure, underpinning the functioning of the project at an operational and decision-making level, whilst ideology and business/end-user orientation define the project at a strategic level. Finally, the position of the project in its life-cycle is an aggregate measure of community cohesion, strength and vibrancy, and ultimately the viability of the project.

As for Plone, it is a project at the beginning stages of maturity. Governance is mostly well-defined, though steps have been taken to improve the scalability of the governance model for the future as there are areas of the project where it is a cause for concern. Informal learning systems are a key part of Plone's culture, with new developers being groomed by more experienced ones on a regular basis, and more formal "boot camps", "sprints" and other events being organised regularly. As "host", the community is welcoming to new participants, but at the core, identity is defined through rich off-line and online social relationships. Community cohesion is high, with few public disputes. A variety of stake-holders and interests are represented at all levels of the project. The project is strongly oriented towards business end users, and business interests are tightly aligned with the project, as most core developers earn at least part of their living from Plone consultancy.

Conclusion

As with any model inferred from interpretive case studies, the model proposed in this paper needs further empirical testing. The specific elements of the model are deliberately vaguely defined, precisely because every community is different. In other cases, other unique aspects may be crucial in defining the community. Perhaps more important than the specifics, however, is the demonstration that mature open source communities are not so different from communities in the real world: they are socially complex, playing host to rich interactions between real human beings. Any account of motivation, organisation, governance, business models or other aspects of the open source movement which reduces participants to purely technically or economically rational actors inevitably betrays this complexity, ultimately limiting our ability to understand these communities. For researchers, this is an interesting problematic in itself. For businesses, such an understanding may be vital in attempting to deal with an open source project. For open source participants, self-reflection upon these issues may help evaluate their own role in their communities. For open source leaders, an appreciation of how the projects they are shepherding function is vital in community-building.

That is not to say that the proposed model will give all the answers. In addition to the aforementioned problems of generalisability and completeness, the model does not offer any explanation as to how an open source project in the early stages of its life cycle begins to accumulate the kinds of processes and structures proposed. Nor does it adequately address evolution of the aspects presented over time.

In terms of future research, there are a multitude of avenues that could be explored. The overarching aim, however, should be to deepen the understanding of how open source communities are composed and operate through in-depth, interpretive research, to counter-balance the current proliferation of positivistic and overly rational accounts.

The communities-of-practice literature has already been shown to be highly relevant to open source communities. Lee and Cole (2003) apply this concept to understanding the evolution of Linux, linking it to knowledge creation through discourse and dialectics. Analysing an open source project's community of practice, for example through a longitudinal case study, would likely yield further insights into its social processes and underpinning structures.

Additionally, the descriptions of the community as a system with a boundary, interacting with other communities and being composed of distinct but interacting sub-groups, has alluded to the

possibility of using social systems theory as a vehicle for further exploration. In particular, systems theory holds promise for building a holistic account of an open source community, as well as explaining its evolution and life-cycle.

Further, the dynamics and evolution of open source can be examined using the language of actor-network theory. For example, Tuomi (2000) uses the concepts of translation and black-boxing to show how social structures become embedded in software modules in Linux. The ability of actor-network theory to "zoom" in and out between levels of complexity in analysis may also help gain an appreciation of the ecology of groups and sub-groups that form inside open source projects.

Finally, Giddens' theory of structuration as applied by Orlikowski (1992; 2000) to technology, could yield deeper theoretical insight into how the social structures of the community are embedded in and influenced by the open source software artefact itself. Orlikowski's theory rests on the presumption of infinite malleability of the "technology-in-practice". This proposition has been criticised in general, but for a certain group of user-developers in open source, the ability to modify the source code may at least increase the realistic malleability of the artefact.

These suggestions are intentionally superficial. More research is needed even to understand how these meta-theories could be useful in studying open source. The more important point however, is this: Open source is a well-established, complex and intriguing phenomenon. The IS research community would do well to apply the same rigour and depth it has shown in analysing information systems in traditional organisational settings, to developing a deeper understanding of open source communities.

References

Ardichvili, A. et. al. (2003). Motivation and Barriers to Participation In Virtual Knowledge-Sharing Communities of Practice. *Journal of Knowledge Management*, 7 (1), 64-77.

Bensbasat, I. et. al. (1987). The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, September.

Bitzer, J. and Schröder, J. H. (2005). Bug-fixing and Code-writing: The Private Provision of Open Source Software. *Information Economics and Policy*, 17 (3), 389-406.

Bonaccorsi, A. and Ross, C. (2003a). Altruistic Individuals, Selfish Firms? The Structure of Motivation in Open Source Software. Available from http://opensource.mit.edu/papers/bnaccorsirossimotivationshort.pdf (verified August 31st 2005).

Bonaccorsi, A. and Rossi, C. (2003b). Why Open Source Software Can Succeed. *Research Policy*, 32, 1243-1258.

Brown, J. S. and Duguid, P. (1991). Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organization Science*, 2, 40-57.

Butler, B. et. al. (2002). Community Effort in Online Groups: Who Does the Work and Why? In Atwater, L. and Weisband, S., *Leadership at a Distance*.

Chen W. and Hirschheim, R. (2004). A Paradigmatic and Methodological Examination of Information Systems Research from 1991 to 2001. *Information Systems Journal*, 14, 197-235.

Croston, K. et. al. (2004b). Towards a Portfolio of FLOSS Project Success Measures. *ICSE Open Source Workshop*.

Crowston, K. and Howison, J. (2004). The Social Structure of Free and Open Source Software Development. *First Monday*, 10 (2).

Crowston, K. et. al. (2003). Defining Open Source Software Project Success. *ICIS* 24.

Crowston, K. et. al. (2004a). Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development. *Proceedings of ACM WISER*.

Cummings, J. N. et. al. (2002). The Quality of Online Social Relationships. *Communications of the ACM*, 45 (7), 103-108.

Dahlander, L. (2004). Approprating the Commons: Firms in Open Source Software. Available from http://opensource.mit.edu/papers/dahlander2.pdf (verified August 31st 2005).

Dahlander, L. and Magnusson, M. G. (2005). Relationships Between Open Source Software Companies and Communities: Observations from Nordic Firms. *Research Policy*, 34, 481-493.

Daniel, B. et. al. (2002). A Process Model for Building Social Capital in Virtual Learning Communities. *International Conference on Computers in Education*.

Demil, B. and Lecocq, X. (2003). Neither Market nor Hierarchy or Network: The Emerging Bazaar Governance. Available from http://opensource.mit.edu/papers/demillecocq.pdf (verified August 31st 2005).

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review*, 14 (4), 532-550.

Feller, J. and Fitzgerald, B. (2000). A Framework Analysis of the Open Source Software Development Paradigm. *ICIS*, 58-69.

Fleming, L. and Waguespack, D. (2005). Penguins, Camels, and Other Birds of a Feather: Brokerage, Boundary Spanning, and Leadership in Open Innovation Communities. Available from http://ssrn.com/abstract=710641 (verified August 31st 2005)

Franck, E. and Jungwirth, C. (2002). Reconciling Investors and Donators - The Governance Structure of Open Source, Working paper. http://opensource.mit.edu/papers/jungwirth.pdf (verified August 31st 2005).

Fugetta, A. (2003). Open Source Software - An Evaluation. *The Journal of Systems and Software*, 66, 77-90.

Galliers, R. D. (1991). Choosing Appropriate Information Systems Research Approaches: A Revised Taxonomy. In, Nissen, H. E. et. al. *Information Systems Research: Contemporary Approaches and Emergent Traditions*. Elsevier Science Publishers B.V, North-Holland.

Ginsburg, M. and Weisband, S. (2002). Social Capital and Volunteerism in Virtual Communities: The Case of the Internet Chess Club. *HICSS-35*.

Giuri et. al. (2004). Skills and Openness of OSS Projects: Implications for Performance. Available from http://opensource.mit.edu/papers/giuri_etal.pdf (verified August 31st 2005).

Gutwin et. al. (2004). Group Awareness in Distrubted Software Development. *Proceedings of the ACM CSCW*.

Hannemyr, G. (1998). The Art and Craft of Hacking. *Scandinavian Journal of Information Systems*, 10 (1&2).

Hemetsberger, A. and Reinhardt, C. (2004). Sharing and Creating Knowledge in Open-Source Communities: The case of KDE. Fifth European Conference on Organizational Knowledge, Learning and Capabilities. Available from http://opensource.mit.edu/papers/hemreinh.pdf (verified August 31st 2005).

Henkel, J. (2004). Patterns of Free Revealing - Balancing Code Sharing and Protection in Commercial Open Source Development. Available from http://opensource.mit.edu/papers/henkel2.pdf (verified August 31st 2005).

Johnon, C. M. (2001). A Survey of Current Research on Online Communities of Practice. *Internet and Higher Education*, 4, 45-60.

K Healy, A Schussman (2003). The Ecology of Open-Source Software Development. Unpublished manuscript. http://opensource.mit.edu/papers/healyschussman.pdf (verified August 31st 2005).

von Hippel, E. and von Krogh, G. (2003). Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science. *Organization Science*, 14 (2), 209-223.

Karels, M. J. (2003). Commercializing Open Source Software. *ACM Queue*, Jul./Aug.

Kimble, C. et. al. (2000). Effective Virtual Teams Through Communities of Practice. Available from http://www-users.cs.york.ac.uk/~kimble/teaching/hi-2/wp0009.pdf (verified August 31st 2005).

Kimble, C., et. al. (2001). Communities of Practice: Going Virtual. In Malhotra, Y. (ed), *Knowledge Management and Business Innovation*, pp. 220-234. Hershey, PA: Idea Group.

Klein, H. K. and Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. *MIS Quarterly*, 23 (1), 67-94.

Kraut, R. E. and Streeter, L. A. (1995). Coordination in Software Development. *Communications of the ACM*, 38 (3), 69-81.

Krishnamurthy, S. (2002). Cave or Community: An Empirical Examination of 100 Mature Open Source Projects. *First Monday*, 7 (2).

Krishnamurthy (2003). A Managerial Overview of Open Source Software. *Business Horizons*, Sep.-Oct. 2003, 47-56.

von Krogh, G. et. al. (2003). Community, Joining and Specialization in Open Source Software innovation: A Case Study. *Research Policy*, 32, 1217-1241.

Lakhani, K. R. and von Hippel, E. (2002). How Open Source Software Works: "Free" User-to-User Assistance. *Research Policy*, 32, 923-943.

Lee, A. S. and Baskerville, R. L. (2003). Generalizing Generalizability in Information Systems Research. *Information Systems Research*, 14 (3), 221-243.

Lee, G. K. and Cole, R. E. (2003). From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development. *Organization Science*, 14 (6), 633-649.

Lerne, J. and Tirole, J. (2001). The Open Source Movement: Key Research Questions. *European Economic Review*, 45, 819-826.

Lerner, J. and Tirole, J. (2002). Some Simple Economics of Open Source. *Journal of Industrial Economics*, 46 (2), 125-156.

Lueg, C. (2000).Where is the Action in Virtual Communities of Practice? Presentation at the Workshop Communication and Cooperation in Knowledge Communities, at the German Computer-Supported Cooperative Work Conference (D-CSCW), Munich, Germany. Available from http://www-staff.it.uts.edu.au/~lueg/papers/commdcscw00.pdf (verified August 31st 2005).

Madanmohan, T.R. and Navelkar, S. (2002). Roles and Knowledge Management in Online Technology Communities: An Ethnographiy Study. Available from http://opensource.mit.edu/papers/madanmohan2.pdf (verified August 31st 2005).

Markus, M. L. et. al. (2000). What Makes a Virtual Organization Work? *Sloan Management Review*, Fall 2000, 13-26.

Michmayr, M. (2004). Managing Volunteer Activity in Free Software Projects. *Proceedings of USENIX Conference*.

Mocks et. al. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, 11 (3), 309-346.

O'Mahony, S. and Ferraro, F. (2004). Hacking Alone? The Effects of Online and Offline Participation on Open Source Community Leadership. Available from http://opensource.mit.edu/papers/omahonyferraro2.pdf (verified August 31st 2005)

Orlikowski, W. J. (1992). The Duality of Technology: Rethinking the Concept of Technology in Organisations. *Organization Science*, 3 (3), 398-427.

Orlikowski, W. J. (2000). Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations. *Organization Science*, 11 (4), 404-428.

Orlikowski, W. J. and Baroudi, J. J. (1991). Studying Information Technology in Organizations: Research Approaches and Assumptions. *Information Systems Research*, 2 (1), 1-28.

Paccagnella, L. (1997). Getting the Seats of YourPants Dirty: Strategies for Ethnographic Research on Virtual Communities. *Journal of Computer-Mediated Communication*, 3 (1).

Raymond, E. S. (1999). The Cathedral and the Bazaar. Available from http://www.catb.org/~esr/writings/cathedral-bazaar/ (verified August 31st. 2005).

Reagle, J. M. (2003). Socialization in Open Technical Communities. Available from http://reagle.org/joseph/2003/socialization/voluntary.html (verified August 31st 2005).

Rosen, M. (1991). Coming to Terms with the Field: Understanding and Doing Organizational Ethnography. *Journal of Management Studies*, 28 (1), 1-24.

Schwen, T. M. and Hara, N. (2003). Community of Practice: A Metaphor for Online Design? *The Information Society*, 19, 257-270.

Stewart, D. (2004). Status Inertia: The Speed Imperative in the Attainment of Community Status. Available from http://opensource.mit.edu/papers/stewart1.pdf (verified August 31st. 2005).

Tuomi, I. (2000). Internet, Innovation and Open Source: Actors in the Network. Available from http://opensource.mit.edu/papers/Ilkka%20Tuomi%20-%20Actors%20in%20the%20Network.pdf (verified August 31st 2005).

Tuomi, I. (2005), The Future of Open Source. In: Wynants, M. & J. Cornelis (eds.) *How Open is the Future?*, VUB Brussels University Press, pp. 429-59.

Walsham, G. (1995). Interpretive Case Studies In IS Research: Nature and Method. *European Journal of Information Systems*, 4, 74, 81.

Walther, J. B. (1995). Relational Aspects of Computer-mediated Communication: Experimental Observations over Time. *Organization Science*, 6 (2), 196-203.

Watson, R. T. et. al. (2004). Governance and Global Communities. *Journal of International Management*, 11 (2), 125-142.

Wenger, E. (2000). Communities of Practice and Social Learning Systems. *Organization*, 7 (2), 225-256.

West, J. and O'Mahony, S. (2005). Contrasting Community Building in Sponsored and Community Founded Open Source Projects. *Proceedings of the 38th Annual Hawai'i International Conference on System Sciences*.

Yamauchi, Y. et. al. (2000). Collaboration with Lean Media: How Open-Soruce Software Suceeds. *Proceedings of ACM CSCW*.

Zeityln, D. (2003). Gift Economies in the Development of Open Source Software: Anthropological Reflections. *Research Policy*, 32, 1287-1291.

Zhang, W. and Stock, J. (2001). Peripheral Members in Online Communities. *Proceedings of the Americas Conference on Information Systems*.

Appendix A: Survey and interview responses

As part of this research, a combined survey/interview e-mail was sent to the several members of the Plone community, containing six general and three or more subject-specific questions. In addition, one email was sent to Beatrice During of the PyPy project with questions about "sprint"-based development, following a talk she gave on the subject at the EuroPython 2005 conference.

The full, unedited responses received are included below. The respondents are:

- *Alexander Limi*, co-founder, UI leader and chief architect of Plone Solutions A/S
- *Alan Runyan*, co-founder and architectural leader, owner of Enfold Systems LLC
- *Paul Everitt*, chief executive of the Plone Foundation and founder of the Zope Europe Association
- *Joel Burton*, president of the Plone Foundation
- *Helge Tesdal*, chief executive of Plone Solutions A/S
- *Geir Bækholt,* founding partner of Plone Solutions A/S
- *Stefan H. Holek*, current Plone release manager, employee of Plone Solutions A/S
- *Alec Mitchell*, independent IT consultant and core Plone contributor
- *Florian Schulze*, student, independent IT consultant and core Plone contributor
- *Hanno C. Schlichting*, IT consultant and core Plone contributor
- *Jens Klein*, IT consultant and Archetypes release manager
- *Matt Hamilton*, Technical Director of Netsight Internet Solutions Ltd.
- *Matt Lee*, Senior Developer at NHS Connecting for Health
- "*Mr. X*" (anonymous) of a large organisation using Plone as part of a product offering
- *"Mr. Y"* (anonymous) of a large organisation using Plone as part of an internal initiative
- *Beatrice During*, of the PyPy project, researching "sprint"-based development

Alexander Limi

Plone co-founder, UI leader and chief architect of Plone Solutions A/S

A. About you

1. Briefly, for the record, who are you?

My name is Alexander Limi, I work as the Chief Architect and Interaction Designer at Plone Solutions in Norway. I am 30 years old, and have an educational background in psychology and computer science.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become involved in Plone?

I started the project back in 2000 with Alan Runyan from Houston, Texas. My current role in the project is user interface design, project management and community communication and coordination. In parallel, I'm part of Plone Solutions, a company that I own along with a few friends and fellow Plone developers. I do Plone-related work full-time, although I am occasionally part of projects where I do pure user interface and interaction design.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

Plone is built on a very agile stack of software, and we can adapt very quickly to meet new challenges or implement new ideas. Plone is very light on process, and getting something implemented and being a part of the core framework is easy as long as people agree that it makes sense to have such a component as part of the system.

A major part is also that we have strong multilingual support. At this time, Plone has more than 50 translations, and has a unique and powerful way of handling internationalization and localization as well as maintaining content in multiple languages. This gives us the upper hand in a market like Europe, where multiple language support is pretty much mandatory - but it is also starting to matter in markets that have traditionally been monolingual, like the US. The rise of Spanish as a second language has a lot of companies seriously looking at software that can handle more than one language.

Interestingly, I think the strongest part of Plone as a technology is that most of it's value lies in Plone's vision, goals and approach to solving challenges. The most positive aspect of Plone is actually that the technology is less important - it's not technology for technology's sake. If Plone were to switch out all of its underlying infrastructure with something else - it would still be Plone, it would still be a strong community with a strong vision and direction. The technology was chosen because it was what solved our problems in the best possible way when we started - but we are agile enough as a community to adjust the underlying infrastructure to what we need to solve our problems. That's the most positive thing about Plone and its community - its versatility and agility. I firmly believe that this attitude affects both the software and the people involved.

4. What are the greatest problems with, or threats to, Plone?

As with most open source projects, documentation is a major hurdle. Introducing new developers to Plone could be a much more streamlined process, but it's also made more complex by the

comparatively big stack of software involved. Web development is a complex undertaking, and content management using web applications even more so.

As for threats - the biggest threat at the moment isn't a technical one, it's more of a marketing and mindshare battle. There's enough systems out there that being heard is a challenge. Fortunately, we have come out among the most visible and well-regarded systems out there, and people are paying attention when we do a release. It's been interesting to see how Plone has started showing up in random articles that are not Plone-centric, but Plone is being used as an example of X or Y, or being mentioned as "one of the available systems in this business area". We still have a lot of ground left to cover in this area, though.

I firmly believe that the next big challenge for Plone is in the marketing/mindshare/storytelling area, and this is where I personally spend a lot of my time and energy these days.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

The Plone community is at a stage where it has reached critical mass. It means that the project does not stagnate when some of its leaders are less involved in a period of time, and that there are enough companies basing their entire business model on Plone to keep the project alive, no matter what happens to the people or companies involved.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

The community influences my dealings a lot. Constant feedback, filtering and monitoring of the Plone users and developers is crucial to bring Plone forward and to solve the problems and challenges people are facing in the real world. The community *is* Plone.

C. Particularly about your role

7. Plone has always been open source. In the beginning, why was this decision taken?

It's a combination of idealism and business sense. We all believe that software is a commodity, a liability if you want - it a means to an end, and is worth nothing if people can't contribute. It's about building and refining a framework to solve your challenges, and you want people to help out with solving those problems. So much valuable time and energy is wasted when a company goes bankrupt and their software withers away because it is closed and proprietary. The real value comes from the refinement, constant testing and improvement that goes into a product, which again leads to solving problems for your customers. That's where the important part of the value chain is. Software is no good if it causes more problems than it solves, and proprietary software all too often falls in the former category - especially in this particular class of software.

Business-wise, it was also to the only thing that made sense. There's no way a small group people can take on behemoths like Vignette, Interwoven or Documentum on their own - so you use open source as a competitive advantage. Open systems are the only long-term viable solution for something that is as central for your company as the content management system you use.

The content management market is dysfunctional by its very nature - why would you pay $X million to lock yourself to one vendor, one provider of consultants and training? When you get a system from a vendor with a proprietary product, you are still only halfway there - you need to integrate with your existing systems, your workflow and your business processes. Why should you pay for the privilege of only being able to use one company to do this? What happens if your

vendor changes business focus or goes bankrupt? Can you still keep your system running, can you still extend and adapt the system to the constantly changing needs of your business?

Were the specific business models of companies like Plone Solutions, who build their solutions on Plone, explicitly part of this decision?

It was more the other way around - Plone Solutions was founded on a business model that is compatible with the open source approach to software, and founded on the same core values. That being said, one of the truly great things about the Plone community is that it's very business-focused. Most of the serious Plone developers come from companies that do Plone full-time, and has it as their main or even their only source of income. Conversely, there is also a strong faction of the developers rooted in academia and students are an important part of the Plone ecosystem. They often have interesting ideas, and time to implement them.

I believe that it's this combination of business and idealism that makes the Plone community a success - they are both pulling in different directions, but since it's possible to satisfy the requirements of both groups if you do the necessary communication and coordination, you get a very flexible and powerful tool that can handle a variety of challenges.

How much of a help or hindrance is Plone's license when you sell Plone-based solutions?

For us personally, none at all. It depends on what market you are in - some markets (like the European one) are very receptive to open source, other markets (like the US corporate one) are more suspicious, and require more work to be convinced. Of course, these are generalizations, it's usually more per business sector. The aerospace industry can be very critical to open source, whereas the government can be very receptive, for example. We are in the lucky position to get much more job offers than we want to take on, and can cherry-pick interesting and rewarding customers instead of compromising our ideals to make a living.

8. Many would consider the team of "limi & runyaga" Plone's spiritual leaders, if you will. Yet, while you have been very active and vocal on the mailing lists and chat room, Alan has been quite silent, both on the lists and in the codebase in the time leading up to the 2.1 release (e.g. since the end of April he has only one check-in to Plone). Do you agree with this assessment? Do you feel that there is a leadership vacuum on the technical side, or is the community sufficiently strong to manage by itself?

A vacuum can be good in the sense that you get time to focus on one part of the equation without interference from other variables. Specifically, the upcoming release has seen massive and important updates to the user experience you get when you sit down and start using Plone, and it has also given us time to refine and rewrite core components instead of constantly inventing new infrastruture - which has been very healthy for Plone as a product.

Of course, you can't be in that state forever, and that there is an articulate vision on both the technical and the user experience side is equally important. The community recognizes this, and is able to self-repair to a certain extent - another part of reaching critical mass. If the technical side started lagging far behind, it would be raised in the community, and we would pick somebody to shepherd the technical decisions. Fortunately we haven't had to do this in its extreme consequence, but you can easily see the self-reparing mechanisms of the community at work.

I expect Plone to enter a phase of more plumbing and infrastructure work in the near future - the requirements for the next-generation user interface are in a way a driving force for the infrastructure improvements. The current phase has fixed a lot of fundamental issues in the user

interface part, and we are ready to take on the next level of functionality and work closer with getting the user experiences to match the powerful functionality that is being added.

Interestingly, even though the plumbing and infrastructure takes a lot of resources, we're in the luxurious position of having a lot of unrealized potential that can be unlocked with some minor refactoring of the underlying infrastructure and some user interface work. The important thing about user interfaces is to refine, refine, refine. There is always a lot of things that can be better, and most of the time evolution is preferrable to revolution - even though the latter may be more tempting at times.

Going back to the original question, I definitely feel that the community has reached a point where it is self-organizing if needed. There is enough money and business on the line for it to make sense for companies to start investing money in the development process. Of course, people *expect* the original team leaders to provide the vision and the guidance, but they are no longer required to do all the work - which was the case in the past.

I mainly see our roles at the moment as providing vision and organizing the community to pull in a particular direction. It's hard to let go of the actual implementation part, because sometimes the vision of what is needed is so detailed and so strong that you are tempted to "do it properly" yourself. Being able to delegate the work to other is part of growing up as a community, and I strongly believe that Plone is managing that transition in a good way. It has been painful, but very rewarding to go though this process.

9. How consciously and actively do you "shepherd" to Plone community, answering mailing list posts, resolving disputes and so on? How important do you feel that such activity is, and who is responsible for it?

I personally take a very active role in this area - but it is by choice. I like being on top of what is happening, I enjoy motivating people to deliver components that excel in what they do - and I consider the community our most important asset. There's such an amazing amount of talented people out there doing incredible things with Plone - which was started by a tiny group of people unhappy with the status quo.

I believe that it is important that I do this work - but it's also important that I know when to step down and let that responsibility reside with others. Sometimes it's hard to stay silent when you have strong opinions in a particular matter that is really outside your area of expertise, and sometimes it's difficult to realize that you - as an individual - do not scale to being involved on all levels. I guess it's part of your "baby" growing up - being able to let go, let it figure things out by itself, as a community, a collective unit.

Another thing that separates the Plone community from a lot of other communities is the amount of real face-to-face communication we have. We organize workshops, conferences, informal gatherings - and most people have a lot of close friends in the Plone community. One of the strengths of this approach is that we can have really heated discussions about things related to Plone - but people are very seldomly offended, since they most likely have met the person on the other end in person - and know that he's a human being, that he's a nice guy. There is a good separation between what is being discussed and the persons involved, and this makes for a healthy community. We can be the best of friends in real life, and still argue agitatedly about a particular part of implementation detail - without offending anyone.

10. What is your impression of the composition of the Plone community? How much of it is commercial-driven and how much is volunteer-driven? Is one aspect more important than another, and is the relationship between firm interests and volunteer interests always harmonious?

It's pretty evenly split. There are strong commercial interests in Plone, but there is an equally strong force that is volunteer work. As I earlier mentioned I believe this combination is what makes up the Plone community, and is the reason why Plone is succeeding where other projects have failed. We never go too far in one or the other direction, we are specialized, but in a million directions at the same time. We have people that have an incredible amount of knowledge in a particular specialized area, and we have people that are generalists and architects. It's a very healthy composition, and we have very constructive dialogue in the community.

11. Many "archetypical" open source projects have a fair degree of idealism at their core, for example the belief that all software should be free and sharable. Is Plone influenced by such ideology at all, or is Plone's philosophy more pragmatic?

Again, we are back to the healthy split in the community. There's a fair share of idealism, and a fair share of business sense involved. This makes for an interesting approach to these issues, and we make sure that both sides are heard when making decisions. Some people depend on Plone for their livelihood, others are just in it for the fun. In sum, I believe that we are part of the newer, more mature and business-compatible open source. We are not afraid of doing business and making money off our product - but at the same time we retain a set of core values that might lose us a contract or two along the way, but we are able to go to bed and wake up feeling good about ourselves and what we do in our day-to-day work. And that's what it's all about in the end. Control of your own destiny, a fulfilling way to work, and lots of fun and good people.

Alan Runyan

Plone co-founder and architectural leader, owner of Enfold Systems LLC

A. About you

1. Briefly, for the record, who are you?

Alan Runyan, cofounder of the open source Plone CMS project. I have been working with Python and Zope for approx. 5 years.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become involved in Plone?

I started the project with alexander limi. I attempt to move the development forward by donating our companies resources to key aspects of the process. Our company focuses specifically on Windows and DAV supports of the Plone stack.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

Localization. Usability. Python/Easy to mold/change. Community.

4. What are the greatest problems with, or threats to, Plone?

Plone's greatest problem is focus. It needs to stay focused on being 1 thing insead of all things. The biggest threat to Plone is disenchanting the community unhappy

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

I believe the community is large. But I no longer have time to develop the community. What does the traffic stats look like? could you tell me? I believe its fairly strong. Its not too fragmented if so it would be becasue its built on Zope (which has a mailing list) and the CMF (which also has a mailing list) but I think ppl using Plone come to the plone mailing list and is contains Q&A's for all three, Plone/CMF/Zop

I would say the community is strong but UNFOCUSED. There are no dedicated community herders today.

We need people who listen and urge the community to move in constructive ways -- not just empowering them but listening to them - actively.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

absolutely. the community is what made me stick with plone. I have personal relationships in the community and even when i feel unhappy with the project -- the community members are there to reinforce my commitment. I believe this is very important attribute behind cults. And is something we try to exploit in the dealing with people in the Plone project. *wink*

C. Particularly about your role

7. Plone has always been open source. In the beginning, why was this decision taken? Were the specific business models of companies like Enfold who build their solutions on Plone explicitly part of this decision? How much of a help or hindrance is Plone's license when you sell Plone-based solutions?

Plone's license (GPL) - I was always against. I have always been PRO-BSD. Because that is what Python itself is. plone was open source because we wanted to gain as much market share as possible. both alex and I were service-consultants. now enfold has moved to products/value-add on top of Plone which has been interesting. our business model is very interesting and has taken a *very* long time to make cohesive. specifically our model focuses on windows (where zope was a second class citizen with regards to the linux configuration of Zope -- we have resolved these issues) server and desktop.

GPL is a problem for some commercial engagements. Its complicated. I hold a much different view of the GPL than most other people in the plone community. I have also spent many hours involved in legal meetings -- I know way too much about law with regards to copyright. which is obtuse in the first place. Having had the misfortune of spending my time -- I do understand why larger organizations stay away from GPL (usually their council plays it safe and says AVOID GPL AT ALL COST). Why? Because thats the guaranteed way to stay out of court. And thats what lawyers do - and do well.

8. Many would consider the team of "limi & runyaga" Plone's spiritual leaders, if you will. Yet, while limi has been very active and vocal on the mailing lists and chat room, you have been quite silent, both on the lists and in the codebase in the time leading up to the 2.1 release (e.g. since the end of April you have only one checkin to CMFPlone). Do you agree with this assessment? Do you feel that you have a leadership role in the community, and that you are somehow responsible for Plone's future direction? What role do you see yourself and Enfold Systems as having in the Plone community?

Yes I do feel responsible for moving Plone. And the last TWO years I have been very passive. For a variety of reasons. One was a large Zope 3 application (which is now open source) we developed as RUNYAGA, LLC. and then the start of ENFOLD SYSTEMS, LLC. w/ Andy McKay. Where we made Zope 2.8.x run on Windows *very* well. Then we developed quite an impressive way of Windows commercial software: Enfold Server, Proxy and Desktop. And some other products that are closed. But what about the *community*?

- We have developed quite a few community oriented aspects. But It is true that development doesnt matter if you have no visibility. Its the exact same phenomenon as celebrity. Celebrities must be seen out in public at parties or they are not doing their duty.

- I have worked on some of the most inane aspects of the Plone community. For instance the US trademark and the Intellectual Property committee's. This stuff just sucks the life out of you. Its boring. Its not what I enjoy but its necessary.

- I also do have a social life *wink*. Not saying Limi does not. But I no longer can hack all night like I used to. I'm married to my wonderful wife, Alma (who does some design for Plone in her spare time) and just bought a house. Have two dogs and am getting back into cooking! And starting up exercise and qi gong.

- In the past I just hacked. In fact for the first 6 months of Plone's life -- alma supported me. and I had money to live on. My living expenses were about 1k/month. It was *great*. I swam during the day and hacked pre/post swim and at night when alma was sleeping. Those days have changed.

- Community has grown significantly and many of the new guys I dont really know. I met them at the NOLA conference and they seemed to be the same caliber as the original Plone community -- personable, friendly, smart, and into music. Great people doing amazing things. On Plone.

But Enfold Systems has done some amazing work:

- Here is a link to the talks we have done recently: http://www.enfoldsystems.com/About/Talks

- We have a much nicer installation for Plone: http://www.enfoldsystems.com/downloads.html

- We will shepherd Five integration into CMF/Plone

- We have worked a bit on PlonePAS but some of infrastructure things such as (CMF)PropertySets which enabled Calvin to do his Mozilla work! And a host of projects/applications - https://houston.enfoldsystems.com/browse/

And the most interesting project that is soon to go public is Entransit our Deployment system. Which I believe has a chance to revolutionize and bring a larger audience into the community. And expand the Plone project's visibility.

Thats an awful lot of waxing. In short. I have felt myself to be the person who will land strategic technologies for the next generation of Plone including lots of Zope3/Five work; Entransit; and creating high-value commercial tools that validate Plone and offer functionality that is only available in expensive commercial systems - for a reasonable price (even FREE for Non Profits).

Enfold Systems is a commercial company. It is a great strength of the community. Do we work onPlone? Well. It may not be as visible. Look at FATE and Flon -- ask some of the people using these technologies. Are we bettering Plone? I think we have created the technologies and spotted the trends to make Plone viable in the next versions from 2.2-2.5 (PlonePAS/Five/RDBMS/etc) for the community to play higher in the stack. Maybe I am wrong -- but I some how doubt it.

9. Some have complained that there is a lack of leadership on the code and infrastructure side in Plone. Certainly, the 2.1 release has been pushed almost entirely from the UI side, though there are plans to focus on infrastructure such as PlonePAS for 2.2. Do you feel that there is sufficient vision, guidance and capacity for arbitration when it comes to Plone's underlying technical infrastructure? Is the community equipped to handle such radical changes as a move to Five Views or PAS?

ok. well yes - I have been busy working on less visible aspects. We have experimented with lots of technologies out-of-band from Plone (which is good) and now we have the understanding of how to apply our wisdom/experience. I feel 2.1 has made some great movements in the usability aspects; and I think adding more infrastructure simultaneously would not have been a wise decision. Can the community - on its own land large structural changes in Plone? I believe the answer is two fold:

- Yes the community can do this. But it requires a leader and someone who is committed to owning and sharing the responsibility with others in the community. And they need to have a good idea of where to go *wink*. But I feel there are people in teh community that *can* do this. But its hard to do because most people dont have first hand experience with the technologies (that are the targets for structural renovations for Plone). And that means experimentation. Few people want to experiment or have time to experiment on a large scale in Plone and try out new ideas and can swallow throwing away all the time if the experiment fails. As a friend of mine, kiko says "R&D in Open Source ends up happening on the Weekends -- and that does not scale". I'm paraphrasing but

its true. Plone is large and its hard for community members to co-ordinate and land changes wholesale for a new version of Plone. Most of this I believe is my fault for not passing on the "authority" of doing this.

- I believe the community is equipped. The community is 100% collaboration and adaptability. That is the strength of open source communities and especially python-based communities.

- Enfold Systems wants to help move Plone forward along with the community and other community-oriented companies such as PSOL and COM.LOUNGE (and the tens others: ngeniweb, cignex, etc). Its just a matter of time and priorities. When you are trying to survive and build technologies that will make you competitive for the next 24 months -- its hard to justify lots of time to adding infrastructure. ala: no need for us to add infrastructure that will be thrown out in 2 releases; I have learned a ton of lessons being involved with Plone. And moving too early is a huge problem.

I am quite happy with where Plone is at. Is it moving fast enough? Some people think its too fast. But I believe we need to move swiftly to new architecture including Zope 2.8.2 and future 2.9. We need to clean up and make Plone more competitive. There is no need to port Plone to Zope 3 until we have our usabilitiy, localizations and featureset base line complete. We are *almost* there but we have a few most problems to solve. Import/Export. Version Control. Are the two really big issues. Things like PAS are just removing older components for newer ones (they do have a impact but not as real as say - Version Control and Workspaces). We need these features but why has there not been Version Control? Because no one has championed the winning Way. And as with open source thousands of implementations will bloom and the sweetest smelling will be picked. Once we are complete with localizations and usability -- once we are the Apple of Content Management. We will be ready to evolve AGAIN onto Zope 3 and this time - we will know where we want to end up.

Paul Everitt

Chief executive of the Plone Foundation and founder of the Zope Europe Association

A. About you

1. Briefly, for the record, who are you?

Paul Everitt, founder of Zope Europe Association. Also, executive director of the Plone Foundation and co-founder of Zope Corporation.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become involved in Plone?

I first interacted with Alan and Alex about Plone while still at ZC, near the end of 2001, I believe. I am primarily involved in Plone on the organization side, not the software side.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

I think the biggest asset to Plone is its democracy. Second, its style. Farther down the line, its technology.

For the technology itself, the biggest asset is the OOBE, out-of-the-box-experience. It is very useful and very usable with no extra work. It is also quite attractive.

4. What are the greatest problems with, or threats to, Plone?

1) The technology stack is baroque, brittle, and "proprietary". For every technology (programming language, database, template system, etc.) in the stack, customers have never heard of it.

2) Plone was originally a UI experience around other people's architecture. This scope wasn't too big for a community with two full-time workers, the founders. However, Plone is now also a framework, and the 2 founders have their own companies to run. Thus, there's a hole in the governance model for an architecture that is increasing in scope.

3) There are now competitors that are attracting attention, making Plone in danger of being the old new thing.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

Plone's greatest asset is its democracy. In fact, you are the archetype for this. :^) Your statement at the end of the ECM panel, about feeling wanted, states the case for me precisely.

I think that the Plone roadmap might need some more adult supervision and ownership for the core architecture. Beyond that, I wouldn't try to formalize Plone at all. Being grown-up ain't all its cracked up to be.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

Even though I'm not a developer, the Plone community is a tangible, warm, friendly, inviting thing to me. It is the lens through which I see Plone. In fact, it *is* Plone. The software comes and goes, changes, ebbs, etc. It's the people that make Plone what it is.

I'm also quite interested in the businesses in Plone. I see Plone's democracy as a level playing field for a vast network of small businesses, each able to align their personal/technical ambitions with their professional/career ambitions. It's a powerful combination.

C. Particularly about your role

7. As the executive director of the Plone Foundation and founder of the Zope Europe Association, how do you see their roles in the Plone community? How relevant do you feel that they are to Plone today, and will this change in the future?

I just finished a workshop for people that run open source foundations, so it's an interesting question.

The Plone Foundation exists to Protect and Promote Plone. Not to develop Plone. It shouldn't encroach on the turf of the development community. However, it should make sure the the fairness needed for Plone's democracy has a governance model and long-term owner.

I also hope, deeply hope, that the Plone Foundation provides something like a career path for community people that want to tackle non-technical leadership. Plone's growth won't be based exclusively on software. It would also be around organization, governance, leadership, and marketing. It's my job to help build this machine, and to find people to replace me.

For ZEA, I must admit, this is the Big Idea that animates my activities. 15 years ago, people accepted at face value that only a multi-billion dollar company could make an operating system. The idea that disconnected volunteers worldwide could beat publicly-traded "grown-ups" would have seemed insanely absurd.

Sometimes the conventional wisdom is utterly clueless. Sometimes, perseverance overcomes skepticism, and an alternate history emerges.

I see the same thing on the business side. People think that open source *business* requires expensive, cathedral "grown-ups" for integrators, to make that open source stuff "controlled" and safe. Might it be possible, though, to organize a market using the same principles and benefits from open source software? Could you provide scale, redundancy, choice, fairness, and invigoration in a business model by removing the cathedral?

I think ZEA has legitimate shot at this. We must be viewed as owned by the Plone producers, and servicing their businesses, not creating a new business that disintermediates them. If we had 50k euros to invest in the sales machine, we could become a very interesting story, very quickly. We really are that close.

A business model that unites the producers, and other important links in the value chain, which also supports the foundation and commons, is a really interesting idea. I hope it works. :^)

8. Plone is perhaps uncommon as an open source project, because it has a marketing effort behind it, as will become particularly evident with the upcoming 2.1 release. How do you feel that this effort fits in with the community and its goals, and how important is it? How aware are Plone contributors of the marketing effort, and how much do they value it?

Because Plone has always been about style and pizzaz, I think this fits in quite well. Also, since half the attendees of Plone Con are small businesses, I think the community is deeply interested in seeing the brand grow. It is so much easier to close a deal when the customer has heard of Plone. It is especially easy when they call you asking for work. :^)

I don't think they are aware enough. It's primarily my fault. I waaayyyy overextended myself on organizing stuff at EuroPython. 10 days later I went on vacation, then directly to OSCON. Thus, the 10 weeks in the runup to the 2.1 release didn't give me much time at all.

It's never too late, though. :^)

9. How important do you feel that business interests are to Plone's development and future? Is there (always) a harmonious relationship between Plone and the firms that leverage it?

Tremendous, and yes.

I remember when Zope first took off. IMO, the key was that we convinced hundreds of micro companies worldwide to bet their businesses on Zope. We were viewed as fair, responsive, and fun. I think that mantle has passed to Plone, and because of this, Plone is the real deal. others are spreading the brand, lining up the big customer wins, and investing time in the software.

The Foundation is a really important step in making sure the magic of fairness and self-interest remain. I think we bootstrapped the foundation well, and thus, people feel like it is *their* foundation. As such, if a non-harmonious business conflict arises, the foundation has the legitimacy to make a ruling that stands a chance of acceptance.

10. How important, in terms of its sustainability as a project and quality as a product, is it that Plone is open source? The path Plone has followed has obviously been very different to that of Zope, which you helped make open source. What implications have these differences had? What can other business-focused software platforms learn from these experiences?

Each approach has a different set of pros and cons. Without a doubt, if Hadar hadn't been the most enlightened VC of his day, Digital Creations would have died before Zope was hatched. Thus, getting financing can really help the beginning.

However, it can be at the expense of the end. Investors have to get liquid. There has to be an exit. And thus, there is a bit of a zero-sum, winner-take-all mentality that might be unsolvable.

It is soooo hard to get a vortex created. Meaning, enough mass, buzz, governance, and momentum to separate yourself from the 400 open source CMS projects and actually have a shot at closing mainstream sales. Somehow Plone got that vortex without financing. As such, I think its future trajectory won't be constrained, meaning, the size of the plant isn't constrained to the size of its pot.

I think the biggest lesson for other businesses to learn is in one word: "real". The open source thing has to be real. You have to really, deeply be ok with the idea that your child grows up and moves out the house one day. After we got our C round of 12.5M, we had a grown-up CEO installed. He once said, "Open source is like the big-boobed booth babe you bring to a show, parade around, and send home after closing." He just viewed it as an angle, not a principle.

If it isn't real, then eventually the conflicts of interest will surface and you won't get the network effects.

Joel Burton

President of the Plone Foundation

A. About You

1. Briefly, for the record, who are you?

I'm a software developer/manager in the US. My background is in corporate IT, and my educational background is in Computer Science and Women's Studies.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

I become involved with Plone in January, 2003, when I started answering questions on the mailing list. My first serious contribution was the documentation for SQLStorage.

I've worked on documentation and marketing for Plone, along with a few bug fixes and coding. Mostly, nowadays, I do community work for the project--helping keep plone.org in shape, speaking at conferences, etc. I serve as the Chair of the Board of the Plone Foundation.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

Our community and the add-on products. We have a large audience of people who respond positively to the community we've build and the self-marketing that the community itself does.

Secondly, I think Plone has a "obviousness" to our interface that people really react to, visually and intellectually. It's interesting how often people say that "I downloaded Plone, and I just knew I anted to use it/learn about it"

4. What are the greatest problems with, or threats to, Plone?

Our competition swirls around two languages that have larger followings: PHP and Java. Especially with our Java competition, there's a "single stack" mindset that some companies have that rules out Plone.

A further away threat is that the concept of "content management" as a discrete product is slowly disappearing. Concepts like idea lifecycle management, and deep integration with legacy datastores, CRM, etc., are going to change the content management concept, and a lot of that moves into areas it may be difficult for Plone to follow.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

The community is absoluetly essential to Plone.

The community is a few things:

- a core audience of about 50 people, who are the "regulars" in IRC, conferences, etc.

- a broader audience of a few hundred who have committed in any way to the project

- a large audience of thousands who silently (and not so silently) cheer from the sidelines, advocate for us, implement us locally, etc.

The first group is obviously critical. I think our biggest strength, though, is the third group: we have a passionate set of users, something that some open source products don't have.

As for fragmentation: we're building a big, complex piece of software that's good for many things. So many of us come from very different perspectives on what we do: (a) university students that see Plone as a place to hack and learn (b) small-scale consultants (c) corporate developers (d) free software ideologues. So, not all of us are trying to build the same thing, exactly. But I don't feel like the community has any dramatic rifts.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

I do feel like part of the community. My role is a bit fuzzy--although I'm a programmer by trade, my contributions to Plone have tended to be less technical, as I was getting plenty of on-keyboard action in my consulting practice. As I've recently moved to a full-time position that's more managerial/sales than development, this may change, and I may start doing more technical stuff with Plone.

C. Particularly about your role

7. As the president of the Plone Foundation, how do you see its role in the Plone community? How relevant do you feel that the Foundation is to Plone today, and will this change in the future?

I have a different position here than some others. I don't think the community model that drives Plone is broken, nor do I think the Foundation should be the meta-owner for all things Plone.

Rather, I see that Foundation's role as a focused one: be the legal owner/organizer for the Plone IP, and coordinate efforts to market the product/project. On those fronts, we're making good progress: we've got the legal arrangements (at long last) moved to the Foundation, and we've got a fairly active group of people to help with the marketing, most notably the extensive offer of pro-bono help from Porter Novelli.

In the future, more of the Plone "world" may fall under the Foundation umbrella, but I'm not certain if this is a good thing or not; so far, I've resisted efforts for the Foundation to do things like certify consultants, standardize training, etc. I think those things are better handled by interested commercial ventures, rather than through the foundation (plus, some of those things threaten our nonprofit status, and I have to be the one to worry about that sort of thing).

8. What led to your election as Foundation president? What skills do you feel are most important in this role?

During the formation of the Foundation, I was a kind of gopher (does that word translate well? In US English, it refers to a helpful, background, kind of person, who runs around and does miscellaneous things). I had no formal role--I wasn't on the formation board--but was tapped as "Assistant Secretary" to the Foundation so I could sit in on board calls. I got Porter Novelli to agree to serving as the marketing pro-bono company, and helped with the launch announcement at CAWorld.

I decided to formalize my role by running for the Board at the Vienna conference. The way the Foundation works, you don't run for board positions--the board itself elects its officers. However, I

had been one of the more active people in the starting of the Foundation, so I offered myself as president, and (I think) everyone agreed.

The most important skill, I think, is realism about what the foundation can accomplish, and how much to tinker with the the way the community works now. This requires good communication skills. It also helped that I was seen by many as a trusted person in the community, who didn't have a particular agenda.

9. As a Plone consultant, do you ever feel any conflict of interest between the work you do for profit and your role in the Foundation?

Little. I was quite fortunately placed when I become Board chair: I was a consultant with (basically) one large client (Porter Novelli) who got work companies that were already their clients. So it never happened that PN was bidding against (say) Enfold or PloneSolutions for work. So I felt that my consulting work was more like a full-time job than a consulting practice, as far as competition went.

Nowadays, though, I work for CIGNEX, which is actively competing to get Plone work, and has competed against other Plone companies. CIGNEX believes (as I do) that we'll do best by growing the size of the Plone world, rather than trying to capture a greater percentage of work from the same world, so I hope that our goal--to get business--benefits everyone more than hurts everyone.

In the event, though, I do think there's an inherent risk with having a Foundation officer who has a commercial interest in the product, but it's also a benefit--I wouldn't be nearly as interested in Plone, nor able to help as much, if I didn't earn my living through it.

Helge Tesdal

Chief executive of Plone Solutions A/S

A. About you

1. Briefly, for the record, who are you?

Helge Tesdal, developer/CEO of Plone Solutions.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

I'm a core developer, foundation member, also making standalone Plone products that are not included in the Plone core package. I do Plone for a living.

I knew Alexander Limi way back when we were both studying in Trondheim. We were looking into starting mp3.no, but needed a CMS. After ditching PHP, we started using Zope and Portal Toolkit which later grew into CMF. While Alexander moved to Oslo and worked on what would become Plone, I finished my degree in Trondheim. After finishing my degree, Plone was getting closer to 1.0 release, and when I moved to Oslo I started looking into it again, and attended EuroPython 2002 where it got quite some interest. I worked on a Java web project that lasted almost a year, and when the project ended it was clearer than ever that I wanted to do Plone rather than Java/Struts/Cocoon/JSP or whatever was the buzz at the time. After some thinking, Alexander Limi, Geir Bækholt and me founded Plone Solutions in September 2003.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

- Python, which is a great langauge, fits my brain

- Builds on a well tested battle hardened platform

- Extensibility and the way you're able to plug your own products into it

4. What are the greatest problems with, or threats to, Plone?

- There was a problem with lack of process, but it seems to have improved a lot with the PLIP process and tools to support that process

- The problem with lack of process and direction still holds for some of the packages Plone depends on, like AT and ATContentTypes

- Lack of resources to work on Plone core to do the tedious tasks

- If we don't get new talented people into the community regularly

Of course there is some cruft and products that turn out to be pure and utter crap making me want to tear all my hair out at times, but that goes for any piece of software/framework out there, and fixing it is just a matter of time as long as there is a good community.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

It is a strong, fairly large and growing community, with lots of professionals. I believe the professionals and the regular meetings (sprints, conferences) makes it a friendly and including community, less prone to flame wars, but occationally lack of response and feedback as people are busy working. It also helps with direction and a common understanding of what Plone is and should be.

The community is what makes Plone great, and gives an edge over alternative systems that might be more sophisticated technologically.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

I feel part of the community, and attend as many events as possible. The community is important for peer recognition and acknowledgement when we do something cool, and makes us want to spend more time on Plone.

C. Particularly about your role

7. PloneSolutions is one of the more active companies in the Plone community, particularly because it employs Alexander Limi and Stefan Holek. As an employee of PloneSolutions, do you feel loyalty to the community as well as the company? Is there ever a conflict of interest between your paid-for work and work you do or would like to do for the community?

I believe everyone in psol, including me, feel loyalty to both the company and the community. Naturally the board of psol feels loyalty to the community on behalf of psol. I can't think of any situations right now where something good for psol would be bad for Plone, or the other way around. There are times when we in psol disagree with other people concerning what we believe to be best for Plone.

Whenever possible, we try to make the paid work into community contribution. This is mainly possible with code, especially new products. We always do our best to convince our customers that it makes sense to release the framework type of code, and it hasn't really been a problem yet.

It's been difficult to get any direct return on releasemanagement and when limi reorganizes plone.org, encourages people and makes sure Plone still looks and feels like Plone. In the long run we believe these contributions are important to give us a higher profile and get us the interesting jobs. Although psol pays our bills in the short run, we need Plone to succeed in the long run if we want psol to pay our bills in a couple of years as well.

8. How often are you able to contribute code to Plone and its supporting technologies (e.g. Archetypes, CMF, etc.)? Are these contributions driven principally by your paid-for work, or do you also contribute work for free? Why/why not?

I haven't been able to contribute a lot of code directly recently. As you mentioned before, we have Stefan and Alexander spending a lot of unpaid time on Plone, and we're not able to keep the company running if all of us spend that much time on unpaid work. In addition, there is paperwork and other unpaid company time. We have some contributions that have been driven by paid work, like the ExtendedPathIndex/navtree, PlonePortlets, the GRUF optimizations before Plone 2.0 and so on, but we rarely start new projects unless we see some opportunity for sponsorship somewhere down the road. We sometimes have to unfuck stuff because we need it in customer projects and are pushing Plone to its limits, which enables us to do some bugfixing as paid work.

I consider time spent on Plone to be company time, and it enables me to have more fun at work than I think most people have. Company time sometimes extends into the evenings, but I don't do Plone at all in what I think of as my spare time.

9. Is your motivation to work for PloneSolutions at all influenced by the Plone community and its ideals? If you had the choice, would you do any more or less community-benefiting work?

I'd say that my motivation to work for psol is very much influenced by the community, and that psol enables me to be a part of the community.

We want to do as much community-benefiting as possible, most of all new cool infrastructure. :)

Geir Bækholt

Funding partner of Plone Solutions A/S

A. About you

1. Briefly, for the record, who are you?

Geir Bækholt, Norway, 30 years old.

I am one of the three founding partner of Plone Solutions AS, a Norwegian Plone consulting company.

I am also one of the people that were (although only sporadically) involved in the Plone project from the very start.

I am an active developer of Plone itself and surrounding, supporting technologies, and an active member of the Plone Foundation.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

see 1

I was involved in some sense of the word from before the project started. I was involved in much of the stuff that led to becoming plone as time passed. I was busy elsewhere and not very active in the development of Plone itself until we were approaching the 2.0 release. Before that i mainly made javascripts and helped with forms when it was needed.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

The community , the collaborative, innovative spirit shared, and the fact that Plone is turning into a "brand"

4. What are the greatest problems with, or threats to, Plone?

Fragmentation, lack of release handling and a proper roadmap. The fact that people tent to reinvent the wheel a lot. the fact that most of the community prefers making things their own way instead of leveraging existing standards. Zope(2?) is becoming more and more dated as a base platform, and we are all noticing the pain of working with it.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

The community *is* Plone. It is very fragmented, and still there is a strong sense of a common way forward when needed. Without the community there would be no Plone the way it is today.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

I feel very much as a part of it. It influences me a lot, and i feel i have my fair share of influence back.

C. Particularly about your role

7. PloneSolutions is one of the more active companies in the Plone community, particularly because it employs Alexander Limi and Stefan Holek. As an employee of PloneSolutions, do you feel loyalty to the community as well as the company? Is there ever a conflict of interest between your paid-for work and work you do or would like to do for the community?

We have actively decided to be community-involved. The Plone community is our main channel of marketing as well as our friends and co-workers. We do our recruitment from the community. We feel a great amount of loyalty to the community, and sometimes the sense of conflict arises when paid work has to be prioritised before free work that helps everyone. Still, the choice is easy, and long-term it helps the Plone community more to have me only partially available but working with Plone than to have me go find a job in the game industry again....

8. How often are you able to contribute code to Plone and its supporting technologies (e.g. Archetypes, CMF, etc.)? Are these contributions driven principally by your paid-for work, or do you also contribute work for free? Why/why not?

Continually if possible. Sometimes paid work that does not produce software general enough to share will get in the way, but generally i try to help out at some level as often as possible.

Most of my work is probably as part of my paid-for-work, as that defines most of my time that is in front of a computer, but when i have spare time (like for sprints) i contribute for fun or because i feel it is needed. Both ways are good, and both have their ups and downs. for free is more fun, because i get to make the stuff i want, but paid pays my mortgage and feeds my kids. There is a fine balance ;)

9. Is your motivation to work for PloneSolutions at all influenced by the Plone community and its ideals? If you had the choice, would you do any more or less community-benefiting work?

Given the choice, we'd like all our work to be community-benefiting. If we could channel every piece of paid-for work back to the community as free software, that would of course be the best solutions possible. We strive for this goal, and try to influence our customers to pay for generic-re-usable solutions that can be released to the public. We still have a way to go, but i have a feeling we are moving (slowly) in this direction. :)

Stefan H. Holek

Current Plone release manager, employee of Plone Solutions A/S

A. About you

1. Briefly, for the record, who are you?

Stefan is working with (and on) Zope for more than 5 years now. He is the author of the widely used ZopeTestCase test framework for Zope2. Stefan is a regular speaker at conferences in the Python and Zope world. He is the release manager for Plone 2.0 and 2.1.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

My first contact with Plone was at the Castle Sprint (September 2003) when the Plone developers convened at Phil's castle, about 50km from where I live. I couldn't let that one pass. ;-)

Currently I am the release manager for Plone.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

It's pretty and it's easy to use. Once you've wrapped your head around Zope it's even reasonably easy to develop for.

4. What are the greatest problems with, or threats to, Plone?

The codebase was/is developed by a *multitude* of developers. Code quality varies accordingly.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

Plone *is* the community, and vice versa. Plone being community driven is probably the main difference to CPS, Silva, and others.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

Very much so. The great bunch of Plone developers was my main reason for getting (and staying) involved.

C. Particularly about your role

7. As the Plone release manager, to what extent do you feel responsible for Plone and its survival as a product? Does this responsibility extend only to the codebase, or do you also feel responsible for the Plone community at large (e.g. in terms of drumming up support, resolving conflict between developers, inspiring new developers to join etc.)? If so, how is this manifested?

I deliberately try to stay out of the cheerleading business. There are people who do this a lot better than I do. I concentrate on keeping an eye on the codebase, and yes, I feel responsible for that.

8. How does your work as a Plone consultant and developer influence your interaction with the Plone community? Is there ever a conflict of interest between what you do for profit and whatever role you may feel that you have in the community?

Unfortunately yes. For one there is a time tradeoff, i.e. many times consulting work does not translate into contributions to Plone.

9. How often are you personally able to contribute code to Plone and its supporting technologies (e.g. Archetypes, CMF)? Are these contributions driven principally by your paid-for work, or do you also contribute work for free? Why/why not?

I do not (or very rarely) contribute actual features to Zope/CMF/Plone. What I do all the time is fix and maintain things. I do a lot of free work too. Why? Beats me ;-)

Alec Mitchell

Independent IT consultant and core Plone contributor

A. About you

1. Briefly, for the record, who are you?

I am Alec Mitchell, a Los Angeles, CA, USA based IT consultant who uses Plone in client projects as much as I am able, and of a late a Plone core developer.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become involved in Plone?

I develop custom applications for clients on top of Plone using Archetypes, and more recently have become a Plone core developer. I started toying with Plone during the 2.0 beta cycle (late 2003, though I had poked around with it minimally before that), while considering it as the basis for a customer project. I had been involved in some small pure Zope projects before that. I became directly involved in developing the Plone core at the Amsterdam UI Sprint in March 2005, where I worked with a number of core developers on polishing and optimizing Plone 2.1.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as atechnology?

A clean template language, good separation of interface and content, heavy use of cross-platform web standards, and of course Python. Also, Archetypes provides an incredibly nice framework for creating content types with potentially complex behavior and relationships. Mainly, Plone provides a clean easily customizable interface on top of Zope/CMF which is itself a great set of technologies.

4. What are the greatest problems with, or threats to, Plone?

Before 2.1, scalability was a big issue, hopefully that will not be the case with 2.1. Plone lacks a good versioning system, which should be considered essential, and a decent forum, which some people seem to consider essential. Plone seems to have a strong brand, but more traditional RDBMS backed systems like Ruby on Rails seem to be gaining mindshare (I can't stand to look at those RoR templates myself, too much like PHP). There are perhaps a few ways by which the bar for entry into the Plone world may be lowered. Note that all of the recent news pieces on Rails are a about how trivial it is to do this or that, while most of the available introductory information on Zope/Plone highlights its steep learning curve in some way. Though I'm not entirely sure that's too important for the market to which Plone aspires. It is not a platform for toy MySQL apps, and we don't want to make it appear as such.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

The developer community is on the small side compared to some projects like KDE, but compared to most open source projects it seems quite large and vibrant. It doesn't appear to be dominated by a few strong personalities, but seems very dynamic and merit driven. The user base appears to be quite large for a product of this sort (judging from mailing list traffic), and very well behaved. I have yet to witness any real fights within the community, and there doesn't seem to be much in the way of fragmentation (though like any community it seems to have a curmudgeon or two). :) The

community appears to be entirely essential to the development of Plone. I can't really imagine where 2.1 would be without the involvement of a number of dedicated community members, both old and new.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

Certainly. It's very nice to get immediate feedback on one's work, and discussions on IRC and through email can be a very effective way to work out a new feature or solve some other sort of problem.

C. Particularly about your role

7. You have recently become quite active in the Plone community, investing a lot of your time. Why do you do it?

Foremost to have stable scalable platform to work with, and to help make Plone recognized as an important player in the CMS/Application framework space. By increasing Plone's quality and marketability, I hope that my own marketability will be improved. I'm semi-consciously making the gamble that spending time/effort to become a recognizable and respected figure within an Open Source project of growing importance will lead to more and more interesting work opportunities. It's also nice to have one's work recognized with such immediacy, by a group of one's peers. Plus it's just a good group of folks, it seems.

8. Do you feel that your status in the community has changed, or is changing, as a result of your contributions? If so, in what sense?

My status has changed quite a bit. Before I started contributing directly, I would write an occasional, hopefully helpful, post on the mailing list and submit a patch/bug report here and there; nobody within the community really knew or cared who I was. Now, apparently, I'm a Rock Star. ;) Such recognition isn't terribly important to me, but it's not at all a bad thing, and potentially a very good thing (as noted above).

9. Do you intend to continue to be involved with Plone in the future? Why/why not?

Yes, as long as I can find Plone projects to work on and time to work on Plone itself. It's a fun community, and a fantastic framework. I love working with it and on it, and I think it has a great deal of unrealized potential, which I hope to play a part in realizing.

Florian Schulze

Student, independent IT consultant and core Plone contributor

A. About you

1. Briefly, for the record, who are you?

I'm Florian Schulze, a 26 years old student of computer sience form Alsfeld in Hessen Germany. I'm currently writing my master thesis and starting my own business.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

I recognized Plone for the first time at the end of 2003. I started to use it around version 2.0.1. I first did rather simple websites with it and now do more and more complex things with it. I consider Plone vital for my future in the web application business, that's why I started to contribute quite much around the beginning of June 2005 to help with Plone 2.1. I digged through bug reports and did my own categorization. Soon after that, I started to tackle javascript problems and added the first unit tests for them.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

The standard conformance, the accessability and the UI. From a developer point of view, Zope as the basis (especially because of ZPT) and Python as the language of choice for the implementation. Python is a very strong point for me, I don't think I would still be programming if I hadn't discovered Python. I got frustrated from all those other languages.

4. What are the greatest problems with, or threats to, Plone?

The release/development process was a weak point, but it got much better. From the short time I'm involved in Plone, I think there also aren't enough good developers who can invest enough time.

Another point, which seems to get better, is the slow evolution to newer technologies like CMF 1.5.

What's also missing or isn't done cleanly yet, are addons like Forums. Most of them don't seem to fit into Plone and they don't follow it's philosophy or don't interact well enough. Some of this will be much easier to fix with Plone 2.1 because of ATCT and related work.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

I'm involved for a rather short time, but from what I saw, the core team seems to be rather small, but strong. I don't have a clue about the user base of Plone, I don't know how big it is or how they think of Plone.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

Yes, I felt like I was a part from almost the beginning. I know everyone from IRC only, but all seem to be really nice and respect each other. I'm really looking forward to see people in real-life.

C. Particularly about your role

7. You have recently become quite active in the Plone community, investing a lot of your time. Why do you do it?

I need Plone 2.1 and saw that it needed help. It's fun to work with Plone. Soon after my first contributions a felt that I was able to improve Plone even more and that my improvements were well received. There is still much to do and some work is not so fun, but if those obstacles get removed others are able to contribute again. There is another developer who did some really needed cleanups to i18n and I feel like I'm doing the same for the javascript side of things.

I also like the fame ;-)

8. Do you feel that your status in the community has changed, or is changing, as a result of your contributions? If so, in what sense?

Oh, yes! People are honoring my contributions and listen to my suggestions.

9. Do you intend to continue to be involved with Plone in the future? Why/why not?

Yes, there is much to improve Plone even further and the technology around it evolves. As long as I'm doing webdevelopment and I don't find anything better I will work with and contribute to Plone. I'm really looking forward to get Plone on top of Zope 3 step by step.

Hanno C. Schlichting

IT consultant and core Plone contributor

A. About you

1. Briefly, for the record, who are you?

My full name is Hanno Christian Schlichting, born in October 1980 in Hamburg/Germany, where I still live. Besides working for a local hospital, I am studying computer science and philosophy at Hamburg University.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become involved in Plone?

After running a small IT company in my late years at school, I started studying at Hamburg University five years ago and to work as a system administrator for the hospital I am still working for. In this position I had to deal with an old and unmaintained intranet, that merely consisted of some linked html pages. As most users didn't know any HTML, this was a definite case for a CMS. But the budget was short and so I looked into the open source scene for any promising products.

My first search narrowed down to Typo3, Midgard and Zope. As I knew PHP and hated it because of its lack of object orientation support, I went for Python and Zope. As I needed something more out-of-the-box usable I looked at CMF, which was more like what I needed but just plain ugly. So I discovered Plone, which was to me at that point just a nice presentation layer for CMF. This must have been somewhere in early 2002 at Plone 0.9.5, CMF 1.2 and Zope 2.4 or 2.5.

As documentation was really hard to find back then, I got myself subscribed to the plone mailinglist. As mailinglists were split to users and developers some month (or years?) later, I read on both but as traffic got to high on users I unsubscribed from that one. One of the first things I did was to translate Plone to German, as otherwise I couldn't use it in my company. This was also the first thing I activly got involved in with Plone. After migrating my intranet to 1.0.5, I had not much time anymore, because the hospital I am working for merged with another one, which included three complete relocations of the whole company and building up a complete new area, which was quite time intense.

So after three years, we had finally managed this complicated process and I had again time to take a look at my intranet. It was in a really messed up state and needed a general overhaul. As I began to update it, I found that many features I wanted where implemented in Plone 2.1. So I got a closer look and found it in the usual 'everybody-wants-features-but-noone-wants-to-fix-bugs'-state it has been before any release so far. So I had the choice of either going for 2.0 and adding lots of features myself or help to get 2.1 production ready. As I dictate my time schedule myself, I took the second option, signed the contributer agreement and rescheduled some other internal projects, I would have spent my evenings on instead ;)

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

For me this gets down to a really short list: Open Source, standards compliance, great usability, easy maintainance and always integrating thrilling new technologies.

4. What are the greatest problems with, or threats to, Plone?

Plone is even after introducing the PLIP mechanism unreliable, in repect to release and feature planning. Another problem is customizing the UI to your needs. Either you have to stay very close to the original look or do a complete redesign.

I can't see any major threats out there. Plone has found it's market and has lots of companies around the globe based on it. The only problem might be, that these companies are getting too occupied doing client work and don't spent enough ressources on Plone itself.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

Plone has started as a community spinning of companies. The heart of Plone has always been the IRC-channel, combined with the various sprints and lately the business network Zope Europe Association. I personally think the story of Plone is best told as a story of a few people. These have build up a really strong community which Plone's success is mostly build on. I can't really tell what the size of this is, as I don't know to what I should compare it, but have the impression of a rather small core group with a mass of part time contributers.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

As I have been around for a very long time now but only in a passive way, I would call myself part of the community but definitly not of the core group. Possibly you have to drink some beers in real live with other members to get that status ;) As I know of the history of Plone I'll have a certain respect for the people that are responsible for this. It's their work and I am only one guy using it, so the community is definitly influencing my behaviour.

C. Particularly about your role

7. Leading up to the 2.1 release, you have done a lot of work, particularly on the i18n and l10n code. Why do you do it?

As stated earlier this prevented me from doing lots of work. Besides I had always a strong interest in web technologies and Plone is some kind of playground to learn, stay updated and use these to me. My normal job work consists only to a very small amount of programming, but I really like to do it. I have therefore done lots of i18n work as I realized that this was the part of Plone I could best contributing back on.

8. Do you feel that your status in the community has changed, or is changing, as a result of your contributions, now and in the past? If so, in what sense?

Definitly. First of all there are some people knowing my name now. As I am so far only communicating virtually with everybody my role is determined through my actions. As I havn't done much in the past, how should anybody had known me? Now I am working on the same project with everybody which includes communication, which is combined with ones actions the base of one's role in a group of people. So quite logically my status has changed.

9. Do you intend to continue to be involved with Plone in the future? Why/why not?

I can't look into the future, but as long as I use Plone in my company I probably contribute something back and therefore will be a part of this community. I think this community is as any

community build on people which share an interest or a hobby. Doing something with people sharing an interest is to me some of the thing in life making it worth living. So I'll probably stay as long it is some fun ;)

Jens Klein

IT consultant and Archetypes release manager

A. About you

1. Briefly, for the record, who are you?

Jens Klein, Software Architect, Plone and Zope-Developer. Free Software and Open Access Activist.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become involved in Plone?

'am involved since June 2003. Its a full time job to me since January 2004.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

a) FOSS from the base (python) up to its componente and lots of extensions,

b) works out of the box,

c) full-featured CMS, but not reduced to it, because

d) a full integrative Application Server works behind it.

e) a real large good working business community/network drives it,

f) which gives free support in irc,

g) and finally: its scalable.

4. What are the greatest problems with, or threats to, Plone?

a) Zope 2 has a bad model (derivation instaed of modularity) and so

b) to many dependend layers;

c) slow and eats up memory;

d) bad marketing, doesnt divide into devs and customers;

e) documentation still a bit chaotic (got better).

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

Community is the strong driver. w/o community Plone is dead. Community is Plones greatest advantage over all other factors. Its large but organized in small efficient networks working together in bigger networks whcih are reaching wide over Plones sphere.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

Yes. I build my business on thsi networked working and have my small network arround me where contracts are made. arround this bigger non-formal networks are existing. all i release is FOSS - thats only possible with such a working network where other do release FOSS too.

C. Particularly about your role

7. As the Archetypes release manager and co-developer of ArchGenXML, do you feel that these tools are intrinsically linked to Plone and the Plone community, or are they separate technologies in their own right with their own user base, support mechanisms etc? Would you like to see closer integration or more separation between Plone, asthe "end-user" product, and technologies such as Archetypes, ArchGenXML and CMF?

Indeed they are linked to Plone's community. Archetypes is the toolbox

you need to easyli extend Plone. ArchGenXML is the simplification, the

dev-speedup and is part of bigger process. The user-bases of thems are a sub-set, a small sub-network of Plones Community. This networks are linked tight. Otoh if you look with a birds view on the whole Plone's Community is just a sub-network and ArchGenXML - at least what the community around learned with it -will be part of Zope 3 in future. The relation of Z3 and Plone isnt defined yet. there will be one, at least the communities will have. I think what i'd like to see is Plone people working closer together with CMF. Same for Archetypes and that includes me. But Its a matter of time. its difficult enough to get enough information to understand and maintain archetypes. so, doing the same for CMF is impossible to me. ArchGenXML is a layer above. it almost independent and just uses both.

8. How often are you able to contribute code to Plone and its supporting technologies (e.g. Archetypes, CMF, ArchGenXML)? Are these contributions driven principally by your paid-for work, or do you also contribute work for free? Why/why not?

say 80% payed vs. 20% for free. It depends on. most Archetypes maintainment i do is for free, but 95% of my AGX-Work is payed. And 95% of my products i write/design/... are payed. I do it payed because I need to feed my family. and theres not much time left to do things for free.

9. How does your work as a Plone consultant and developer influence your interaction with the Plone community? Is there ever a conflict of interest between what you do for profit and whatever role you may feel that you have in the community?

In Plone community most are professionals earning money by supporting plone and its sphere. I'am in that role too. so it doenst conflict. If I need to work for a customer I'll try to do it in a generic way, so it can be reused, built in Agx, Archetypes or as a separate product. Its an FOSS-business model and almost everybody accept and respect it.

Matt Hamilton

Technical Director of Netsight Internet Solutions Ltd.

A. About you

1. Briefly, for the record, who are you?

Matt Hamilton, Technical Director of Netsight Internet Solutions Ltd, based in Bristol UK. Netsight is a full service web development agency that produces websites for a range of public and private sector clients. All the technical aspects of our work are based on Zope and/or Plone.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

We have been using Zope since 1999, ie before Plone existed. We have so far developed 100+ sites on top of Zope. We became interested in Plone in 2002 and launched our first Plone 1 site (for Warwickshire Police Force) in early 2003. Since then we have attended both the Snow Sprints, two UK zope 3 sprints (one we hosted), and Plone Conf 2&3 and EPC 2004 & 2005. I was a founding director for the Plone Foundation, and founding member of the Zope UK association.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

Plone has very good separation of content, presentation and logic, it also have very good accessibility and multilingual support. And of course the community is great... more on that later...

4. What are the greatest problems with, or threats to, Plone?

Plone is only starting to be used in large organisations and generally works in a very different way to them, hence care needs to be taken not to scare them. (e.g. don't call the ZODB a 'database', call it an 'object store' as large companies will instantly liken it to Oracle and set their DBAs on it).

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development

The Plone community is great. They are generally a very friendly bunch of people, and I think take more care than other communitys might in helping newbies. One of the weaknesses of the community is that there are a few very key individuals that carry a lot of weight and the community has no real way to ensure their continual support.

The Plone community has a wide breadth of users, so you get not only coders, but usability gurus, interface designers, business people etc.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

Yes, I feel a strong part of the community. Whenever we go to develop something we look around to see what the community is doing, not just to find out if someone has done what we want before, but the general direction the community is going with various trends.

C. Particularly about your role

7. NetSight is one of the most established Zope consultancies. How do you use Plone in your work? What are the advantages and disadvantages of using Plone over custom-built Zope solution? In addition to technical rationale, please consider the role of the wider Plone community in terms of support, the sharing of open source code, etc.

The vast majority of the sites we build are corporate websites that use very simple content management, ie. a database of news articles. Originally we used to design custom front ends for people to manage these within Zope. The problem was that each wanted a slightly different look, which under Zope is very difficult. Plone gives us a good interface out of the box that is simple to use and can have un-needed features switched off, but that can also be customised to suit particular users. It also give us room to grow so if a client wants more features we have a common framework to build them on.

As mentioned before, we always look to the community to see what has been developed before. We are still not the best at this and will still often write code ourselves instead of re-using existing code for simple things. But we are slowly re-using more code out there.

Two heads are better than one, and in the sprints much of the work is pair-programmed. Whilst at the Snow Sprint earlier this year I worked with Geir Bækholt on a new product, ResourceRegistries. We both had different skill-sets and learned a lot from working with each other, and in the end we developed a much better product.

8. How often are you (and other developers at NetSight) able to contribute code to Plone and its supporting technologies (e.g. Archetypes, CMF, ResourceRegistries)? Are these contributions driven principally by your paid-for work, or do you also contribute work for free? Why/why not?

We actually contribute very little back :(Unlike many Plone companies we have a design department as part of our company and our main skill is in producing sites with both a sound graphical design as well as technical functionality. On that basis, much of the work we do is customisation and writing specific functionality for clients that is specific to their business and wouldn't make sense distributing.

Where I think we make the most contribution is in promoting Plone and working on Plone within larger organisations. We are better off leaving the core development to better developers. Having said that we have written a Windows NTLM Authentication product for Zope which we may release soon.

9. How does your work as a Plone consultant and developer influence your interaction with the Plone community? Is there ever a conflict of interest between what you do for profit and whatever role you may feel that you have in the community?

No I don't think there is a conflict, however as usual paid work gets priority over community work, and hence sometimes community work can get neglected if we have a rush on. I am however always keen to pass on knowledge that we have gained from work with clients to others.

As we have been using Zope from the beginnings, I have quite a lot of experience in the lower levels of Zope that many Plone companies do not. This can be invaluable for debugging things like memory leaks etc.

Matt Lee

Senior Developer at NHS Connecting for Health

A. About you

1. Briefly, for the record, who are you?

I'm Matt Lee, Senior Developer with NHS Connecting for Health,

Delivering the IT infrastructure that will modernise the NHS. In addition to this, I am involved with the Free Software Foundation and CNUK Media Foundation, both of which also use Plone.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become involved in Plone?

I started using Plone in June 2004, after some time spent trying (and failing) to get Microsoft Content Management Server to produce valid and accessible websites.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

* GPL licensed - this is good, because it means that nobody can ever take Plone away from us. It also means we have the freedoms, as a community to do whatever we want or need to do to get it working.

* Zope/Python - Built on existing free software, Plone doesn't try and reinvent the wheels.

4. What are the greatest problems with, or threats to, Plone?

* Lack of really good error database - sometimes, you can have a problem that's just a weird little flaw. It can be a real nightmare to figure it out, because everything can seem like it's fine.

* Keeping to update, and worrying about dependencies - It'd be really good if there was some kind of packaging manager for Zope/Plone, like Debian or Ruby have.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

* Community - the Plone community is a strong one, with clearly defined structure. Also, many of the key people behind Plone are making a living from it, which is why Plone is so consistent.

* Foundation - it is a good thing that there is a Plone foundation, and

I feel more should be made of it.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

I have code in the Plone collective, and I can answer some questions in the IRC channel. I feel the Plone community welcomes everyone who can do anything to contribute back. I feel I am a part of thecommunity.

I also feel as I can bring my experiences of the free software world to the community.

C. Particularly about your role

7. The NHS has recently shown a lot of interest in Plone. Why is this? What alternatives were considered, and why was Plone chosen?

The wider NHS and their interest in Plone is largely a result of what we're doing. NHS Connecting for Health and the work we do is one of the biggest things to happen in the NHS in recent years. We're doing a lot of work, and if people can see us using something like Plone – something that is vaguely accessible to them in their job, and it's useful and not difficult to setup and configure - then they're going to seize the opportunity. I think that's one of the most positive aspects of our Plone usage. We had a Plone day event in our Exeter office, and invited people from the wider NHS to come and spend the day, talking about Plone.

As far as why we chose Plone, it was more a case of needing a CMS, quickly, that did a lot of what we wanted out of the box. Plone did this. Whether we continue to use Plone in the long term, I don't know - certainly at the moment, I don't see anything else that can do what Plone can do, either proprietary or as free software. The CMS project, was a project of the old NHS Information Authority, and over a period of about 18 months we evaluated many of the proprietary CMS systems, as well as a few of the PHP-based free software ones. None of them offered us a consistent level of functionality while complying with web standards. When I found Plone, it was out of desperation, but I'm certainly glad I found it.

8. Has the NHS evaluated the role of the Plone community in relation to its dealings with Plone? Does the NHS wish to play a role in this community? If so how and why? If not, why not?

Certainly within my team at NHS Connecting for Health, we embrace the Plone community for all its work. We also intend to 'share-alike' with the community, making as much of our own code as free software as possible. The Plone community also teaches us things, and our work with Plone companies like Netsight, Fry-IT, Blue Fountain and Plone Solutions have set us in good stead for the future.

“Mr. X” (anonymous)

Of a large organisation using Plone as part of a product offering

A. About you

1. Briefly, for the record, who are you?

I'm X, I'm employed at X as a manager in development. My title is Director of Development.

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

I became involved around February 2004 when I was asked to evaluate open source document management projects. Plone was not on my list because it was web content not document management. However, certain key executives pushed it and my objectives were overruled. I became the development manager of a new project to build a document management system product using Plone as the base.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

It's use of Python, the out of box experience(on Windows at least) and the somewhat ok UI(at least for an open source project).

4. What are the greatest problems with, or threats to, Plone?

Zope 2 is overly complex and hard to understand(not pythonic). Plone is difficult to integrate with non-Zope stacks.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

Great community, it is Plone's greatest strength.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

I did while my project lasted. It did in a number of ways: We used community written add-ons as part of the product. We collaborated on one feature, managed external files with Alan Runyan

and his company. We participated in a few sprints. The community's decision to stay with GPL licensing also influenced our product.

C. Particularly about your role

7. X have invested some money directly in the Plone project. What was the rationale for this?

I can't say more than what has been said in press releases.

8. Does X wish to play a proactive role in the Plone community, for example to shape the development roadmap? If so, how and why? If not, is it a conscious decision not to?

X has decided not to continue development of a Plone based product.

9. Is X involved in or intending to get involved in other open source ventures? If so, how does Plone fit into this?

X is involved in other open source projects(Web address, removed). I can't speak about future ventures.

“Mr. Y” (anonymous)

Of a large organisation using Plone as part of an internal initiative

A. About you

1. Briefly, for the record, who are you?

Y

2. What is your relationship with Plone and the Plone project? How and how long ago did you become in Plone?

'become in Plone'? I guess you mean 'become *involved* with Plone'. We use Plone as the basis of our content management system. I started working on the project a year ago, and Plone had already been selected as the platform.

I learned Python, Zope, CMF and Plone all at the same time. That was fun, coming from a background of ASP, VB and SQL Server :)

To be honest, there's not a lot of Plone left in our system any more - it's been mostly customised away.

As far as my relationship with the Plone project goes, I hang out on IRC (#zope,#plone,#archetypes - x) and help people when I have time.

B. About Plone in General

3. What do you think are the most positive aspects of Plone as a technology?

For me, the main thing was that I had a ready-built system to start with, and could change a bit at a time. The capabilities you get with AT are great (though I have the odd issue with the implementation, see below).

It also has an enthusiastic userbase - pretty invaluable for an OSS project.The support you get online is probably the best thing about Plone.

Note that I've not used the internationalisation features of Plone, they aren't requirements for this project. Even Plone's detractors are pretty impressed by these features. So I may be missing out on a big chunk of Plone goodness :)

4. What are the greatest problems with, or threats to, Plone?

Brace yourself :)

Layers and layers and layers of complexity, poor implementation, and nonsensical things like subclassing CMF tools and adding no methods or attributes, just changing the meta_type. That's pure cruft, completely confusing to the newbie. Archetypes (which, since ATCT ships with Plone 2.1 I consider to be part of Plone) makes the situation worse. Schema-driven development is a great idea, but the implementation that AT provides is pretty bad. Some points off the top of my head:

- The AT default accessor should be fast. It's not.

- ClassGen.py is a horror. Check out generateZMICtor. Because that string is simply exec'd, you can't debug it. That sort of stuff can be done with proper Python introspection.

- Too much dark magic. Why is so much stuff passed around using **kwargs? The _initializing_ hack?

- base_edit, widgets. It took me a good six months to understand how all the ZPT metal declarations fit together. Autogenerated forms are a great idea but base_edit.cpt is *really* tough to understand for a newbie.

- __implements__, yet another implementation of interfaces, fantastic - and have you ever stepped through a call to implements()?

- ReferenceEngine. Nothing wrong with references as a concept, but even the AT guys know the current monster is unmaintainable. After being bitten by it, we stripped all reference engine calls out and replaced them with simple catalog keyword indexes indexing UID() or the physical path as appropriate.

Plone's divergence from CMF HEAD as also been worrying - the CMF team shouldn't have to make CMF releases on an obsolete branch to support Plone! The disjointed efforts to get Plone onto 1.5 are pretty indicative of the leadership problems with the project. Christian Heimes did a lot of work to do the migration, and Raphael Ritz; Jens Vagelpohl was drafted by Alan to do it - and nobody really seemed to take the bull by the horns and actually get things moving for ages. I believe whit did some work on it a while back, too.

Plone also has some way to go before it can be considered 'enterprise ready', in no particular order:

- It needs to run on Zope 2.8 for MVCC, the absence of it *really hurts* on big installations.

- It needs to ship with userfolders that talk the common auth protocols: LDAP, NTLM, etc.

- It needs to be able to connect reliably to enterprise RDBMSs: SQL Server, Oracle, DB2 etc.

- It needs to be fast out of the box, not slow (we have to serve 150,000 authenticated users globally, no anonymous users for us).

- It needs to support revisioning of documents (I appreciate the difficulty in covering all use cases here)

- For the UK market, it needs to be DDA compliant.

- Limi's right, Plone needs a WYSIWYG editor by default. It also has to cope with people pasting hundreds of pages of content from MS word into it. (I think this has been addressed in 2.1, so that point may no longer be relevant). It should look like and function like MS Word.

- In a similar vein, plone needs a spellchecker.

breathe :)

Clearly some of the above are at the Zope-level, but they are holding Plone back.

Right, threats to Plone? Lack of organisation and under-the-hood QA. Publicity mishaps like the recent CA stuff.

Don't get me wrong - without Plone as my learning environment, our CMS would not have happened. But there's a lot of technical ugliness under the skin, particularly if you regard AT as part of Plone.

If I had done Zope/CMF development before this project, I would have started out with bare CMF and gone from there. I think our end product is sufficiently dissimilar from Plone to warrant that. Plone's great for fairly light customisation.

So maybe some of my comments are a little unfair on Plone because we've pushed it pretty hard.

5. What is your perception of the Plone community (i.e. weak/strong, cohesive/fragmented, large/small, etc.)? How important do you feel the community is to Plone's development?

I think I've answered some of that in the previous section!

As far as Plone development goes, I would like to see is QA. There's a lot of bad code going into the Plone core.

The community is good for helping newbies, I spent a lot of time on IRC in the early days. It feels pretty large, and people are helpful.

6. Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

Involvement with the community has been difficult. Corporates don't really understand open source, and it's difficult to motivate them to make them understand. I think that's changing slowly - here, driven by Linux.

C. Particularly about your role

7. Your company has recently invested in a large-scale Plone project.

Why is this? What alternatives were considered, and why was Plone chosen?

Interesting question :) I'm afraid I can't really go into specifics on this one, other than to say the usual enterprise CMS candidates were considered. Zope's cost of acquisition (no pun intended) was low, and the consultants were cheaper. We were interested in Zope, and the consultant we ended up using introduced Plone as a way to try to solve our problems using Zope.

8. Have you evaluated the role of the Plone community in relation to its dealings with Plone? Do you wish to play a role in this community? If so how and why? If not, why not?

I'll answer the question backwards :)

As a company, Y will not endorse Zope, Plone or any of the technology stack, and our name must not appear in any marketing or promotional material. That said, I've managed to get the complete cloak of secrecy a little, which I why I'm allowed to answer these questions :)

That makes getting involved in the community directly difficult. Personally I would like to, but can't.

Having said that - we hire developers from the community, which feeds them, so they can go and write more OSS :)

I'm not entirely sure what you mean by 'Have you evaluated the role of the Plone community in relation to its dealings with Plone?'. So I guess the answer to your question is 'no'. I suspect the answer to the question you're trying to ask is contained above in my comments about technical QA and who commits to the core.

9. Is open source an explicit part of the your IS strategy, or is the case of Plone more of an isolated one?

Are you asking - does my organisation have a specific open source strategy, or are you asking whether Plone is an isolated case of an open source product?

I'll answer both :)

No, the bank has no open source strategy. Quite the opposite - it has a closed source strategy. All the code and IP I write here is owned by the bank (obviously we can't - and don't - redistribute our Plone customisations). Therefore I could not write OSS while employed here.

We also have to tread pretty carefully with licenses and IP ownership by contractors.

As for the other half of the question - yes OSS is pretty widely used. Linux, Apache, stats analysers, that sort of thing. Plone is higher up the stack (ie. application level) than I think OSS is used generally.

Let me know if anything I've said needs more detail or clarification - I'm pretty tired right now and so I may only just make sense ;) And apologies if the middle section came across as a rant, but these are all things that we - as a customiser - have been hurt by.

My frustration is that I can't actively contribute fixes/improvements back to Plone because of the aforementioned corporate IP policy! Oh well...

Beatrice During

Of the PyPy project, researching “sprint”-based development

1. Most descriptions of open source projects emphasise the conundrum that participants typically never meet in person, yet in the Python, Zope and Plone worlds, sprints and other forms of offline collaborative development are very important. What do you see as the role of sprints? What would be missing from the equation if there were no sprints being held?

The short answer is (IMO): it makes distributed development more collaborative and gives the communities more endurance.

If you look at the primary factor behind succesfull teams ("The wisdom of teams" - Katzenback, Smith) you will understand why sprints evolved and why it works. If a group can focus around challenging goals and always wrap process and communication issues around the actual work, the result you will have the basis for a high performance organization (+ of course other issues of group dynamics, self management etc).

To me sprints is the very basis of this. A group coming together, focusing on challenging goals and always, always _working_ as a way of maturing the group. The focus isnt teambuilding or conflict resolving (like you have in many other organizations - working and then meeting up for a specific process issue - many times confusing the participants instead.)

In this the group is evolving and build endurance for the challenges that the distributed work will put the team through. Thus the result will be an increase of productivity.

BUT: this also has to do with what kind of group you have.

Would you say that the communities sprinting today is a homogenous group (education, skills, interests etc etc)?

By this I mean that sprinting will probably work well in this way for some groups but will have to be wired differently for others. This also is connected to your question about management. To explain this I will refer to the leadership research being done by Hersey and Blanchard. Groups being oriented towards tasks or social interaction needs different management (note also by this means self management - not just the official role) and they describe this in their theory and connect it to the group and its maturity.

So if you take sprinting (challenging goals and resultoriented) as an example - this could fit groups that are both of the "telling" kind and the "delegating" kind. The "telling" kind of group is to me what you described the Alex (?)person stepping in and getting the group focused around the results and "telling" the group.

The results where very good - the group respected a high focus on results and would probably not respect or be as productive during that sprint if someone had started up the sprint with a 1-2 days "brainstorming - lets talk about our intentions etc etc" start up session....

But as the group matures and evolves (achieves results and sees it - this kind of management (or self management) would probably be contraproductive and yield problems.

Also - in this you have potential traps - if a group is very oriented towards relationships and a unclear picture of the goals or goals that doesn challenge them the group can get stuck and dont produce results. If this happens more than once the group will loose "self confidence" - like Martin mentioned the brainstorming sprint they had....

So - for distributed teams this is a key activity to start up the evolving of the team and the community.....buit I think that the form of sprinting needs to evolve with the group - othervise it will be a risk that it will feel contra productive later on. But I dont think we have any communities or projects that have been sprinting for that long yet - or they are starting to show the needs for it now. That would be fun to research and support.....

Some loose thoughts there and maybe not the answer to your question - but its a start and we can take it from there - please ask more question if you think it was unclear....

Moving on ;-)

2. During the method panel, you mentioned that there is a lack of sprint-like activities when it comes to the design and vision stages of development. Can you elaborate on this? What do you feel is missing?

Well - when a group has proven to itself through sprinting that they can reah results, both during the sprints and between sprints, I think they will start to ask other questions and maybe sift focus.

Since the results are there and ongoing the group will probably start investigating metaprocesses and get strategic and visionary (or problemoriented)....thus the sprint will shift focus.

If this isnt somewhat a consciuos decision on when to have these different "outlets" the outspoken purpose of the sprint will collide with the current needs and focus of the group - thus risking that the technique of sprinting will be seen as contraproductive and "not working" any more.....

This is connected to what I mentioned above.

Also - the purpose of sprinting so far is to produce code (i.e result) but results can be many things. So if the group clearly focus on stating the goals around the results and what work they are to do - sprinting is a collaborative technique for doing that.

In requirement processes you have a technique called JAD (IBM) which is very similar.....but then it maybe shouldnt be called sprinting any more?

But I think developer communities (especially distributed ones) need to come together and _work_ around different results - not just code. If a team is getting used to sprinting they might identify and use that technique for other aspects of the work.....

Interesting though - this aso applies wether the team is the client or the client is external,,,,,wether you design up front or not,,,,I found that entire requirement discussion very revealing ;-)

3. Have you investigated the role of sprints and other offline collaboration in terms of recruiting new members to projects and getting an infusion of new talent? If so, what are your observations?

We are discussing this right now actually ;-)

The PyPy project sprinted before they got funded and in our proposal to the EU we specifically described our method of sprinting as a way of disseminating pypy to people not active in this

particular dev work/community and also as a platform for contribution and interaction with the non funded developers in the community.

Now - we have a client - the EU and deadlines/deliverables (the funded aspects) as well as the ongoing dev work that has been going on for over 2 years. We then have to manage the fact that there will be pressure for results and thus some frustration for not having the time to tutor and disseminate to newcomers to the sprint/project...

So now we are discussing different sprints for different purposes in which sprints with more focused results (close to releases and deliverables) being mainly for participants used to the codebase (and to python!) and sprints in conncetion to conferences being more of a tutorial/recruiting/disseminating purpose.

As I said - we are discussing this right know. But to answer your question - sprinting is a very good technique to recruit and get people up to speed with the project (both for the aspects mentioned above as well as for the accelerated team process I mentioned in my talk at EP).

4. Following a sprint or other in-person interaction, do you believe participants are noticeably better equipped to interact and collaborate online as well? Does the long-term performance of a project improve beyond purely the productive output of the sprint as a result thereof?

Yes - definitely. But you also have to weigh in factors of the group evolving/maturing over time and the technique with the group. Otherwise the result can be contraproductive for the entire project and thus the community....

If you sprint in a distributed development team you will probably have a long term effect of 2 +2=5 ;-)

5. Do sprints need managers and action plans, or are they typically self-organising in the same way that online collaboration in open source projects typically is? What are the negative and positive effects of imposing structure, task lists etc. on sprint participants?

This is an interesting paradox and also connected to both Hersey/Blanchard and team building theories. New people have certain needs in the group (see the different stages and the bounce effect I mentioned) and since the sprints will most of the times have unique constellation of particpants each time this will differ within the group - and maybe causing pressure.

Also - if you add pressure due to deadlines and challenging goals the result oriented people in the group will call for structure and control/tracking.

In the evaluations done so far in the PyPy project both newcomers to the project as well as core developers (been involved in the project for over 2 years) have called for somewhat more structure and clearer process (thus asking for management of some sort) - but they do it for different reasons.

I would say though that the sprints in PyPy are organized by the core developers and they are doing a very good job with it. I dont think it is something we can impose formal management around, but we can evolve the process and mature in our sprinting.

Appendix B: Anecdotes from the community

This section contains some messages copied from Plone's mailling lists, referenced as supporting material in the main body of this research. For archives of the Plone mailing lists, please see http://plone.org/about/contact

Included in this appendix are the following:

- *Appreciating contributions*: spontaneous display of appreciation by a member of the community
- *Difference in tone after meeting in person*: the effects of off-line participation on subsequent on-line communication
- *Code critique*: Post raising concerns about a recent change to the software by the release manager
- *Comments from an outsider*: Differences in tone and style between the Plone community and the "outsider" defending an competing open source product
- *Goldegg primer*: Initial post about the Goldegg initiative to the Plone mailing list

Appreciating contributions

Spontaneous show of appreciation from a community member for the author's contribution of an add-on component to the public "Collective" repository; plone-user, August 15th 2005.

Don't be sorry to us all; I for one love developers who care enough to release multiple versions per day/week when necessary. And you are (clearly, from your voluminous contributions and helpful postings) one of those developers who truly care.

I look forward to trying out RichDocument sometime soon... it sounds nearly perfect for a little project (non-calendar-oriented) I have in mind.

Cheers!
+lupa+ CalendarX.org

At 03:37 PM 8/15/2005, you wrote:

>> I have released RichDocument 2.0b1, to be found at
>> http://plone.org/products/richdocument/releases/2.0b1.

> Actually: http://plone.org/products/richdocument/releases/2.0b2

Sigh... http://plone.org/products/richdocument/releases/2.0b3

(sorry)

Difference in tone after meeting in person

Reply to the author by release manager Stefan Holek a few days after meeting in person for the firs time, politely cracking the whip to avoid further delays in the release cycle; plone-devel July 8th 2005.

F**k[18] off Martin ;-)

Beta has been delayed by two (2!) weeks already because we wanted to get CMF 1.5 in there. I agreed to that and still think it was a good move. However we are beta now, and let me remind everybody that beta means "no more f**king around". I am going to get really angry. The UI porn can very well wait until 2.1.1. Now is bug fixing time, and bug fixing time only. It is not my problem that the UI people slacked.

If I catch anybody wasting time on "features" instead of fixing bugs, I'm gonna personally whip that person's sorry ass. No more control panels, kupu drawers and shit -- this time has passed. Yes, you too, Limi

We need to get stable AT and ATCT out too, BTW. We have enough on the plate to need everybody's attention, I would think.

Stefan

On 8. Jul 2005, at 16:21, <the author> wrote:

> Secondly, and probably more important long-term, is what degree of quality in terms of Plone-the-product do we want for 2.1? I think dropping out control panels and other polish now for the sake of (probably well-placed) conservatism at this point would be a big mistake (maybe even bigger than slipping the release schedule, though neither is really acceptable imho). If you think from an architecture/platform perspective, it makes sense, for sure, but in terms of making Plone-the-product impressive to users, that polish (e.g. by not referring people to arcane property sheets in the ZMI) will be vital to the impact the release makes. Experience suggests it is the "wow" factor of Plone-the-product that brings us users and extension developers, so let's not marginalise that.

[18] Asterisks added to protect the sensitive reader

Code critique

Post by Danny Bloomendal with critique on a recent change committed by release manager Stefan Holek; plone-devel August 23rd 2005.

Lurker, the only UI you should touch is the UI of your whip
This change is not right. The oldest items needs to be reviewed first. Hopefully supermarkets in Austria
don't treat customers this way at the cash register

shh42 wrote:

> Author: shh42
> Date: Tue Aug 23 09:35:55 2005
> New Revision: 7972
>
> Modified:
> CMFPlone/branches/2.1/HISTORY.txt
> CMFPlone/branches/2.1/skins/plone_scripts/my_worklist.py
> Log:
> Reversed sort order in review portlet to show the most recent item on top,
> just like in all the other portlets.

Comments from an outsider

Comments from a community "outsider" supporting a competing open source project in a thread for a user asking for a comparison between these, highlighting differences in tone. plone-user August 20th, 2005.

I am not too familiar with Plone and therefore cannot help much in terms of a comparison. I am familiar with DotNetNuke however, and would like to point out some factual errors in some of the comments that may influence your decision:

1) DotNetNuke Skinning: This is by far the most flexible skinning solution for any ASP.Net environment period. You have one constraint -- to have an element with an ID="ContentPane". Other than that you are free to let your designers go to town and create any type of layout. Browser compatibility is constrained only by *your* code and the code emitted by ASP.Net natively. DotNetNuke gives you two easy choices -- create a skin in HTML using regular apps such as DreamWeaver etc. or if you are a developer, you can create your skin as an ASP.Net user control.

2) VB.Net vs. C#: Most viewpoints on this are generally based on incomplete/inaccurate information. DotNetNuke is written in VB.Net, however it places no constraint on your development language for creating modules, providers, skin objects and HttpHandlers. If the language produces a .Net compatible assembly, it's good to go. If you want to change the core, yes, you would need to do this in VB.Net. However, at that point you lock yourself out of an upgrade path and this is not advisable. I code in C# almost exclusively (99.5% of the time) and have had no issues creating modules/skin objects/providers/httphandlers/httpmodules for DotNetNuke. Any reservation about coding for DotNetNuke in languages other than VB.Net is entirely due to (a) developer ignorance, (b) developer's emotional ties to language, or (c) developer's elitist attitude. None of these have any technical merit.

3) Development: A developer with decent ASP.Net development skills can create DotNetNuke modules fairly quickly. However, it isn't necessary to create modules to add functionality to DotNetNuke. You can leverage the framework just as easily by creating script-only ASP.Net controls and embed them as skin objects. The bar for this is low enough that even developers with questionable competency can have add-on functionality in DotNetNuke in an hour or less. With two books in print, a very detailed documentation section on the dotnetnuke.com site and many community sites offering templates for DotNetNuke modules, there is adequate documentation for even newcomers to ASP.Net development. Generally speaking, if an ASP.Net developer is having problems with DotNetNuke development, it is because they have a hard time making the leap from page-centric application development to control-centric application development. Also, the provider model, which is something that is the general trend in ASP.Net applications, is supported by DotNetNuke to allow flexibility for different databases etc. Most developers who only skim the surface don't realize that although the framework supports the provider model to give flexibility, they don't have to build their modules with added support for the provider model if the app is for in-house use where the database and

membership providers are known and unchanging. This lowers the bar significantly because creating a DotNetNuke module becomes an almost trivial exercise of converting their existing ASPX pages into ASCX controls. If a developer can't figure that out, then the only reasonable conclusion to draw is that the developer is unskilled.

The following post demonstrates the tone of the Plone community in the same message thread:

I have never worked with DNN, but I think I can share my experiences at my company, as I started working with Plone and Zope in May, and on an exclusively Windows environment.

The ability to learn Plone is dependant on your ability to learn DTML and Python, and I found both of these simple to begin work on. I had previous experience in ASP and C++, but had never worked with any of the technology that Plone uses. However, I found that many of the concepts I was already familiar with translated almost perfectly to the new technology. Not only that, but Python offers a few programming tricks that I remember wishing I could do in C++.

The learning curve is certainly sharp, as so much of Plone has been built on previous layers. It's entirely possible that a method returning an error exists a dozen layers up your inheritance, and the cause could be something equally obscure. A program called DocFinder is indispensible, as it lets you look up methods based on object types in your site, and can help you locate the code that is returning errors.

3.5 months later, our company has customized almost every default aspect of Plone, and every day it's getting better. Don't let a Microsoft environment discourage you; although the development is primarily done on Linux, it is by no means necessary. As for acquiring the skills, the mailing list combined with Andy McKay's book have answered almost all of the questions I've had. As for whether it's worth diverting resources, that depends on how much you would need to change. Certainly the more you customize the default installation, the longer it will take and the more resources you will need.

Goldegg primer

Initial post by Paul Everitt explaining the Goldegg funding initiative. See http://www.goldeggstack.org/about/primer for further details; plone-dev August 25th, 2005.

Howdy Plone developers. Munwar and I would like to start public discussion about Goldegg, a funding initiative for advancing Zope 3 in the Plone/CMF/Five software stack. In summary, Goldegg provides funding to get Zope 3 into the near-term roadmaps of the stack. Specifically, the funding initiative helps get the leaders of these roadmaps working in coordination on the next releases.

As background, a few months ago I circulated a "CMF++" proposal, where money was pooled to produce 2 CMF releases that leveraged some of Zope 3. Munwar Shariff from CIGNEX convinced a Plone customer to spend money on an expanded version of this proposal that included Plone's Zope 3 near-term roadmap. CIGNEX asked me to be project co-coordinator with Munwar.

We have talked quite a bit with many of the stack leaders and we're now ready to open up the discussion. We have some content ready and are preparing a small little site for news. The next few emails will discuss the proposed projects for the Goldegg One funding cycle covering the next 3 months.

I just sent a very similar email to the CMF mailing list. For Plone, though, we have quite a few things to discuss, related to how the medium-term framework gets planned and decided.

In the coming week we will provide more information about the roadmap items we're helping to support. However, Goldegg isn't a new software effort. Rather, it funds existing efforts. The first cycle of Goldegg funding, in fact, slots most of the money into already-discussed CMF activities. Thus, most of the conversation will happen in the CMF, Five, and Plone mailing lists.

We will, though, start talking to other companies that want to pool money. If you make money off of the stack and want to see its architecture improve in the next release cycles, Goldegg is the best chance you'll get to have an impact. Including the impact of shorter release cycles!

I'm available to chat to anybody that has questions about this. Munwar and I have listened to lots of people and we're trying hard to do things the best way possible. It's a tough job to balance all the interests while still giving this customer the right outcome. Thus, we need to ask you for patience on things we mess up. We might not get it right the first time, but we'll listen and keep trying.

I also need to give huge kudos to CIGNEX for making this happen. They produced the customer funding and even matched it with internal funding. Thus, CIGNEX has primed the pump for a near-term Five-ification roadmap, and they're doing it by giving money to existing leaders for existing ideas. This is a brilliant and generous step. It's up to the rest of us -- businesses with funding contributions, community with code and ideas -- to show whether monetary investment can fill medium-term needs.

--Paul and Munwar