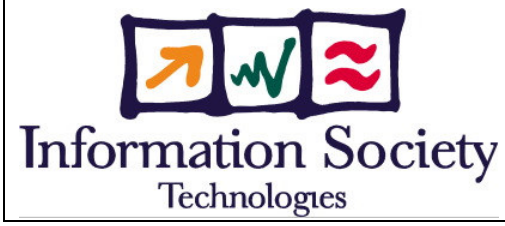
OPAALS PROJECT

Contract n° 034824

Workpackage 8
Open Source and Open Knowledge

Deliverable 8.2:

Inner Source Principles



Project funded by the European Community under the "Information Society Technology" Programme

Contract Number: 034824
Project Acronym: OPAALS
Title: Open Philosophies for Associative Autopoietic Digital Ecosystems

Deliverable Number: Inner Source Principles – D8.2
Due Date: 31/5/2007
Delivery Date: 14/8/2007

Short Description:
The aim of this document is to provide a review and descriptions of the principles of inner source and how this can be leveraged by the OPAALS community.
Partners Owning: University of Limerick
Partners Contributing: University of Limerick

Versioning		
Version	Date	Author, Organisation
1.0	10/7/2007	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick
1.1	12/7/2007	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick
1.2	15/7/2007	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick
Final	17/7/2007	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick

Quality check
1st Reviewer : Evangelia Berdou, LSE
2nd Reviewer : Maha Shaikh, LSE
3rd Reviewer : Luigi Telesca, CreateNet

Dependences:

Work Packages	This deliverable affects a number of work packages namely: WP-7 Community networks and digital ecosystems WP-10 Sustainable community building Tasks:
Partners	This deliverable may be of interest to the following partners: CreateNet University of Kassel London School of Economics
Domains	The domains of, Inner Source, Open Source Software and Global Software Development are extensively discussed in this deliverable.
Targets	Targets of this deliverable include: Domain researchers OPAALS community

Table of Contents

List of Tables

1 Executive Summary

The aim of this document is to outline Inner Source principles and to place Inner Source in the context of both existing open source literature and the ongoing work in the OPAALS community.

The document covers a number of areas which include: Three case studies of Inner Source in use in multiple, large scale global software development companies (IBM, Hewlett Packard and Bell Labs), contrasting traditional open source principles with Inner Source principles, a theoretical grounding from authors such as Goffman and Clarke in the area of different methods of interaction, and the practical issues of managing and organising Inner Source.

These areas were chosen both for the background relevance to Open Source and emerging trends in Inner Source and also for the pertinence to the OPAALS research community.

Upon examination of a number of case studies and surrounding literature a number of cross cutting concerns, repeating principles, problems and issues with implementing an Inner Source solution. In summary, the outcome of this work is that there are many issues involved in the complex area of Inner Source but emerging trends suggest multiple benefits and even go as far as to suggest competitive advantage. We have highlighted the emerging trends in the Inner Source phenomenon and surrounding areas. Awareness of this may be of great benefit to the work of researchers contributing to the OPAALS community both in terms of sustainable community building and in terms of collaboration methods within the community.

2 Introduction

In this deliverable we outline the principles of Inner Source extracted from our initial examination of emerging case studies and surrounding literature. While the case studies have a lot of similarities in that they are for the most part positive experiences of Inner Source, this is probably due to the facts of firstly a low adoption rate in a new emerging technology and secondly perhaps and unwillingness to publicise the failures of initiatives within companies.

These aspects will be examined further in Phase II of OPAALS. This deliverable is the first of two deliverables on Inner Source; the second (in Phase II) will include a more extensive examination based on the emerging questions and issues raised as a result of the work in this deliverable. On review of this deliverable it became apparent that the boundary of these cases and the work in this deliverable was not defined enough, for this reason in an open fashion, we include as an appendix the comments received from reviewers and how we have addressed these comments.

The starting point for Inner Source is an understanding of open source and its many benefits and principles. The benefits of open source (Raymond, 2001) are varying and have a different weight depending on the organisation. More and more we see large organisations focus heavily on open source (Dinkelacker et al 2002). We can see this focus not only by the amount of large corporate investment directed into open source projects[1], (FLOSS Report, 2006) but also by the adoption of open source methods and practices behind corporate closed doors (Gurbani et al, 2005).

An article in the Economist magazine in 2006 cites examples of how the success of open source has changed some of the rules of business. It observes success stories such as Firefox, Google, Amazon, eBay and Wikipedia and the open source methods ideals and philosophies behind these companies. In particular the article looks at how Firefox is gathering momentum and market share (14% in the US and 20% in Europe) on Microsoft's Internet Explorer. The organisational structure of Firefox is also discussed:

> *"From that core group, the open-source method lets a series of concentric circles form. First, there are around 400 contributors trusted to offer code into the source tree, usually after a two-stage review. Farther out, thousands of people submit software patches to be sized up (a useful way to establish yourself as new programming talent). An even larger ring includes the tens of thousands of people who download the full source code each week to scrutinise bits of it. Finally, more than 500,000 people use test versions of forthcoming releases (one-fifth of them take the time to report problems in bug reports)."*

[1] "Firms have invested an estimated Euro 1.2 billion in developing FLOSS software that is made freely available. Such firms represent in total at least 565 000 jobs and Euro 263 billion in annual revenue." (FLOSS Report, 2006)

Wikipedia is also discussed in some detail as an excellent business case where decentralised collaboration can be a very powerful tool. A word of caution obviously (Wikipedia is often held as a modern example of the inaccuracy of popular internet culture) ensues as Wikipedia has been subject to vandalism and prejudice and is an example of how a tool can be powerful if used correctly but if used without question and control, it can almost destroy any benefits gained.

The article is an interesting example of how open source has proliferated and gained interest in the media, by business and beyond software into popular culture.

While open source may not represent a real paradigm shift in software development, it is an extremely successful exemplar of globally distributed development (GSD) and has many other benefits. In particular, in the current climate of outsourcing and off shoring, the open source development model is attracting considerable attention as organisations seek to emulate open source success in traditional development projects, through initiatives variously labelled as 'inner source', 'corporate source' (Dinkelacker et al, 2002) or 'community source'. It has been argued that open source communities can be considered in some cases to be virtual organisations (Crowston, Scozzi 2002; Gallivan 2001). If we consider this assertion to be true then we should be able to transfer the success and benefits of open source to corporate organisations.

Inner source as we use the term here is any leveraging of open source methods within a corporate environment. A variant of this term progressive open source (and within it Inner Source) was first seen published by Dinkelacker et al in 2001. Inner Source is usually employed by companies to capitalise on the success that certain open source projects have enjoyed. There are some distinct differences between open source and closed source development and their respective development communities.

In traditional software development users and developers are typically located in separate departments and locations. This can lead to employees being unaware of development innovations and projects and all too frequently to little mutual respect or voluntary interaction. The user-developer relationship has typically been quite different in open source. While early open source developers were users of actual products, as OSS has developed the situation has changed somewhat. In the absence of a traditional software development company, users need to become more intimately involved in the development process, as technical staff cannot simply send a checklist of requirements to the vendor to ascertain if needs will be met. It is a widely held belief that deploying open source can lead to a sense of shared adventure which is not a common scenario in the proprietary software arena (Raymond, 2001). The transfer of these lessons, where appropriate, to conventional in-house software development is a critical activity and is of much interest to these researchers and we are eager to leverage the lessons learned here for the OPAALS community.

We feel that just as the principles of open source (Raymond, 2001) have helped in understanding, defining and implementing open source, one of the most beneficial (to the OPAALS research community, the larger research community and to companies interested in Inner Source) places to start is with the contrast between open source principles and inner source principles.

This document will now outline a number of case studies (Bell Labs, HP and IBM) to give practical grounding in what initiatives are ongoing in the area. We then set about examining two particular fundamental areas of Inner Source, locally to globally distributed development and managing and organising Inner Source. The first topic becomes apparent as an important issue in Inner Source, as Inner Source solutions are by nature implemented as a solution to leverage global software development. When we consider the move from local software development to global we must consider the existing research in the area of face to face interaction and distance interaction. The second topic examines the important issues of managing and organising an inner source solution which as we will see requires some significant cultural changes. From this background then we will present the inner source principles gathered from the case studies and other surrounding literature examined. The initial understanding and definition of Inner Source gets examined and refined throughout the sections of this document. This examination of previous sections work will culminate in a stronger more defined understanding of Inner Source and its principles. To conclude we present some suggestions as to how this deliverable might relate to the OPAALS community, and finally the conclusion and some closing remarks.

3 Case Studies

We will now examine some of the emerging use of Inner Source in three case studies. The case studies include Bell Labs, Hewlett Packard and IBM. The cases chosen were within a methodological frame of snowballing sampling (Patton, 1990), (Zikmund, 1982), (Crabtree and Miller 1992) and (Babbie, 1995) and are particularly good examples of the use of Inner Source in large multinationals which operate of a global scale. This we feel has the most significant similarities with OPAALS and has the added benefit that Inner Source deployed on a global scale has the most contrast to traditional software development making Inner Source principles easier to extract and define. It is this contrast while not always an absolute truth (there are always exceptions to this case (Cusumano and Selby 1997), (Capilupp and Michlmayr 2007) and (Garcia and Steinmueller 2003) that we feel is of most use in extracting the principles of Inner Source.

Case Study - Bell Labs (Gurbani *et al*, 2005)

Background

This is the first case study in this deliverable and provides excellent insight into what can be achieved with Inner Source. The case study of Bell Labs provides real indication that open source has overcome many GSD problems (Herbsleb and Mockus 2003) and has allowed distributed users to contribute to code with minimum conflict or overheads (Mockus, Fielding and Herbsleb 2002), and many companies are looking to capitalise on the success of open source so that open source can aid companies overcome similar problems. The authors of the Bell Labs case study (Gurbani *et al*, 2005) focus on industry adoption of the open source development methodology. It contributes to the knowledge of Inner Source by ”providing detailed analysis combed from interviews of multiple developers and quantitative analysis of data pertaining to a corporate open source project where multiple organisations contributed synergistically to further and use a common asset”. The case study focuses on a development project in Bell Labs that uses open source software and practices to develop a commercial telecoms internet telephony server. The software was available in Lucent Labs over a four year period for anyone who wished to integrate it into their particular software product line. This particular example is a perfect fit for inner source as the reusability of the code developed is very high.

The differences between open source and commercial development as it is termed (the authors of this deliverable disagree with this term as it over simplifies the open source field in light of the research presented in Deliverable 8.1 on the commercialization of open source) are outlined below in Table 1, in terms of Requirements, Work Assignments, Software Architecture, Tool Compatibility, Software Processes and Incentive Structure. The differences are examined under such headings by the authors as these were the most prominent areas of difference observed in the implementation.

	Open Source Software	Proprietary Software
Requirements	Rely mostly on the user/developer, requests are sent by all users to mailing lists and developers may or may not address them depending on interest and perceived importance (Scacchi 2004).	Considerable effort into gathering and analyzing requirements. Management decide on the priority of requirements based on business needs.
Work Assignments	In open source, developers choose what assignments they want to work on usually based on there own particular need (Shah, S 2004).	Work assignments are assigned by management, with some care taken to match a developer's skills and interests to the domain.
Software Architecture	Open source architectures are more modular than proprietary software (MacCormack *et al*, 2004).	Proprietary software structure is dependant on the organisational structure of the company (Conway 1968) and (Dinkelacker *et al*, 2002).
Tool Compatibility	Open source project exist independently and often have adopted the same set of tools.	Proprietary software generally uses a wider range of tools.
Software Processes	Open Source projects in general have very little in terms of formal process with the software quality control passing through the "walled server" of a few trusted experts or even one respected dictator.	Proprietary software has many highly defined levels of software processes, often accompanied by critical point assessments along the development path.
Incentive Structure	Driven by developer's motivation, including the desire to learn new skills, to create new features that satisfy the developers needs, philosophical benefit of developer contribution and community welfare.	Proprietary software is for the most part profit driven to ensure the sustainability of the company.

Table 1: Differences between open source and proprietary software from Bell Labs case study (Gurbani *et al*, 2005)

Inner Source Applied

The software developed was a Session Initiation Protocol Server (Rosenberg *et al* 2002) which is a text based protocol like FTP, HTTP or SMTP (Gurbani et, 2005). The SIP stack was an emerging trend in communications technology at the time of

development and is a very interesting technology both to developers and to the many areas of software development in the company. The code was written in C and was available on Concurrent Versions System (CVS). This was used for source control and version control. The code ran on Solaris and Linux. The code was originally a SIP server and was later re-factored into a SIP library which hosts a SIP server.

The open source approach in this project arose in three phases; the initial development phase started in April 2000 and was developed and tested for compliance to the IETF standard by two developers. The second stage involved ad hoc distribution of the binary within Lucent Technologies. As maturity of the server grew so too did interest in the servers capabilities within a few select groups. As the maturity of the server grew further, it was no longer a research-only project, and became a full Lucent Technologies product offering. At this stage there was a lot of interest in the SIP server binary, but not much contribution was forthcoming, beyond users reporting their experience to the developers. This use of the binary SIP server increased general use and the profile of the product. User requests and feedback was sent back to developers and prioritised based on business needs of the groups using the product, fortunately a lot of these requests coincided. As SIP grew in profile in the industry and internet telephony became a big part of emerging trends, the SIP server was viewed as an important resource by many groups within Lucent. By 2003 the source code was being studied extensively by many groups in Lucent.

Phase three involved a lot more product specific requests to the author by the individual groups in Lucent, which also coincided with actual code contributions and ideas. This was the breakthrough to the traditional open source model in a corporate setting. The code then took an important change in that it was re-factored to include a transaction library, since all users of the code need a transaction manager. This enabled the creation of multiple user agents that executed on top of the transaction manager (Arlein and Gurbani 2004). With this in place each group could concentrate on the specific functionality they wanted to add to the user agent instead of each group having to also concentrate on the handling of SIP transactions. However in this dynamic environment where open source methods were in use but without the traditional open source social structure and formatting some issues began to arise:

Implications and problems

With many groups using the code base and contributing to the code, different groups took delivery of different code at different times. This meant that the code in use in different groups became isolated and unsupported. Also any additions the group made to the code were unlikely to be fed back to the CVS code and reused by another group in the future. Since many of the groups preferred different tools and didn't use CVS, and instead chose to keep a local copy, this caused further trouble.

Company structure changed as a result of the success and failure of this project. A group was setup to be responsible for a common source code repository; this included another open source term, the "trusted lieutenant", which in this case are key people in each group assigned by management to manage parts of the code.

Choosing to build an open source or inner source resource is an important decision for any company and must undergo cost benefit analysis. In the case in question, the decision was easy since by the time the product need was identified the company

already had, as a resource, a solution that was seen as high quality. Not having to depend on a third party to provide features or bug fixes was also a benefit. However it is usually the case in companies that a cost benefit analysis is outwardly focused and inner source resource would not be valued correctly by the departments responsible for product evaluation. As the concept of inner source or even open source is relatively new or unknown, this can be a problem in a number of departments throughout the company.

Organisation process adjustments were made to adjust to the new paradigm of inner source. Since developers of the inner source project came from multiple groups with different expectations and development styles, this created a mismatch in the development process of the inner source project. This resulted in a very stringent software process including mandatory review and tracking of issues that was not common to all groups. This was sometimes seen as unnecessary by some contributors as the features being added or issues being raised and fixed were often being completely handled by the same group.

Managing contributions was a big issue within the groups as different repositories and tools were used. As a result when issues arose, the problem was magnified. This has highlighted the need for everyone to use the same code repository and toolset.

Coordinating open source and product group development was an issue in development styles between developers working in a corporate environment and traditional open source developers. Pressures existed on developers (as a result of time to market concerns within specific groups) to make changes to the SIP server that satisfied the requirements of the group and not the general changes that would benefit everyone using the SIP server. Changes were labelled of three types; 1) Changes to the general code base and benefiting everyone, 2) Platform specific changes and 3) Product line specific changes. Also it was traditional with groups to apply code freezes coming up to product release, where no new functionality would be added so as not to risk the integrity of other features. This is not really possible in an inner source project where different groups have different release schedules.

Publicity and communication were two particular problems that arose with the company. It was not always the case that employees were made aware of the SIP server resource or when they did, they didn't always know the correct details. While developers of the resource gave talks and explained the code, some argued that even more resources should have been available for publicity. Communication was another issue which in some cases caused two or more developers to work on the same issue with different solutions.

Key Results

The centralised approach to development led to the creation of a very high quality development project where many people experienced the process of emerging technologies development. The project was also used to groom highly motivated staff from other areas in the company who were contributing to the project in a hobbyist approach in their spare time. The project also gained from the classic open source benefit of many eyeballs making the problem look small (Raymond 2001), or even many eyeballs from different perspectives finding more bugs. In particular this

company had the benefit of performance experts suggesting a list of changes to optimise the code, API experts suggested an appropriate API to make the SIP server into a more useable framework and others suggested changes to help port the code to different operating systems.

Analysis

This case study in particular was very successful and makes for a most interesting case or perhaps even poster child for inner source. The success factors can be summarised to be a result of:

- The project successfully implanted a number of changes in technology: a CVS, and some communication tools: for example a mailing list.
- The code quality greatly benefited from many experts having access to the code.
- Communication was an issue which caused a duplication of effort by two different groups.
- Issues arose as groups forked code and subsequent code changes did not benefit the groups depending on the forked code.
- The trusted lieutenant role was required and introduced.
- The cost of the solution was a problem in that it was seen to be under-valued as opposed to a bought solution by the accounting and costing departments.
- Use of an emerging technology that really took off when internet telephony became more standardised and interest in the area grew. For this reason a research department or a department focused on future technologies is seen as a good place to foster growth in inner source development within a company.
- One of the code authors was a leading expert not only in the code but in the IETF SIP standards. Other experts in other areas joined and made the product very strong.
- The code was heavily based in standards, as a result utilising a resource where someone else was responsible for the code's compliance with the evolving standards, was seen as a good way to capitalise on resources.
- The timing of the code's maturity coincided with industry interest.
- The company was liberal with key developer time and also the project seemed to be aligned with the company's business strategy.
- The company was willing to make significant changes to company structure and employee roles to leverage inner source as a useful company resource in the future.

Case Study – HP (Dinkelacker *et al*.2002)

Background

The second case study examines the use of Inner Source within Hewlett Packard. The authors use a number of terms almost interchangeably such as "Corporate Source" (Dinkelacker, J. Garg, P. 2001), Progressive Open Source (Dinkelacker *et al*. 2001; Dinkelacker *et al*. 2002) and finally, Inner Source. All three share the same hypothesis; that it is possible to migrate certain software engineering principles from the open source paradigm to the corporate software development environment. The three terms as defined in this case study differ in terms of access, or how open the source is. HP define Inner Source access to be only within HP and behind the corporate firewall. Corporate Source is HP's implementation of Inner Source and is open to all HP employees. HP's Progressive Open Source (POS) has three distinct levels, Inner Source, Controlled Source and Open Source with respective users as: HP employees, HP partners and everyone with an internet connection.

HP understands the use of Inner Source as the application of open source concepts, perspectives and methodologies in a corporate environment. Open source is seen as being relevant to corporate environments due to its success in software maintenance, reusability and quality (Raymond 2001). Another reason given is that more eyeballs make problems look small. However fundamental conflicts exist. Businesses usually try to retain intellectual property (IP) rights on their software. IP rights is an important issue especially where the Inner Source boundary extends to include a corporate federation. The case study illustrates how inner source can be a good way to realise the benefits of open source while sidestepping possible risks. One risk associated with maintaining a fully open source solution includes the release of untested software. Some of the advantages observed by HP in open source projects include: Community building, open discussions for requirements and features and evolvable and modular designs. However, the contrasting environments of open source and traditional corporate software development have vastly different goals, time constraints, motivations and environments. The biggest difference between the two, as seen by HP, is the pressure of time to market a product, thus making it very important to appreciate how the principles of open source are transferred to the corporate environment. In an effort to make this transfer possible and attractive HP set out a list of expected benefits to be gained through an evaluation of required factors including:

- A toolkit of existing software from which to build new products
- Improved quality of code as many authors will have contributed and reviewed the code
- Community Debugging
- Leveraging of open source tools and methods
- Aid developer's movements within the company as existing developers are aware of the common code tree.
- Faster development schedules with the use of code common to multiple projects.

The challenge faced in implementing inner source is seen as partly being of an organisational nature, for example, how to develop code across organisational boundaries and how to identify and retain module designers.

Organisational challenges include:

- Virtual organisations or group dynamics within a company may work together with various degrees of success, for example if one group always writes high quality code while another group capitalises on this and uses resources elsewhere.
- Leadership is required in all open source like projects. Two distinct problems arise from this in a corporate environment. What happens when a code/module leader leaves the company? What if no leader is responsible for a section of the code?
- Task assignments are much easier to manage in smaller groups where a project manager is able to judge the skillset of each member in the group. It is much more difficult to do this on a large scale.
- Developer indoctrination is possibly the most important and overlooked organisational challenge. If people don't accept the inner source model and really work with it (this includes code standards and general rules of operation), it will be very difficult to manage and run.

Technology requirements include:

- Repository for the code to be held with version support.
- Community Support is important and tools are required to enable community participation and communication.
- Security is a general concern of every company's code, but even more so as the access to code increases so too might the risk in terms of IP rights but also through allowing remote repository access.
- Search and navigation
- IS/IT Support

Inner Source Applied

HP in its' nature as a large multinational company has many developers across the globe. The centre for the corporate source initiative is in HP Labs research Library. The tool support in HP for corporate source is easy to use and is of a familiar format to HP developers. The code is searchable and stored in concurrent versions system (CVS). The code metadata is stored in XML.

Implications and problems

Fundamental Conflicts experienced between open source and business ideals include:

- Time Constraints
  Open source projects are very flexible with time as the cost of not shipping a product version is low. The opposite is true in the business world where penalties may be payable for each day the product is late.
- Intellectual Property rights
  This is a very fundamental difference, where open source developers often believe in the freedom to benefit, use and improve from other peoples ideas. A part of the business world exists solely to capitalise and generate revenue from Intellectual Property.
- Organisational conflicts
  The structure of open source while highly structured with well defined roles is more code orientated and light weight as compared to traditional business organisational structures which have evolved over centuries from military

organisational structures into the industrial age and into what we commonly see in hierarchal structures today. Self managing in open source conflicts with traditional business practice of strong top down management.

- Motivational Conflicts
  The motivation of open source developers can be as simply to learn new skills but can also have more passionate motivation such as "the world needs this product we are working on". In contrast developers in the business world are more career and financially focused.
- Goals
  The goal of a business is usually to sustain itself and make profit and perhaps to improve market position. Open source goals again can be more fundamental with goals such as "to be the fastest web browser on the market within five years".

Key Results

It was envisaged that the results of implementing the above factors would include:

- Greater agility to address new opportunities
- Reduced time to market
- Higher quality code
- Increased consistency between HP products
- Decreased costs of product production

Analysis

- The use of Inner Source in HP is continuously being improved with the focus for improvement being on organisation and cultural issues.
- From a technical viewpoint the fundamental shift in focus is that instead of the binary code being the end-product, now the source code is the end-product.
- Adoption of inner source is more social change than technical change. While technical changes are needed, more important aspects include leadership and developer backing
- There exists a trade off between choosing an area that is a concern or interest to many people and choosing an area that is of high value to a lot of people.
- Code module granularity is important in a corporate source code repository. The module must offer significant functionality to be worth potential inclusion and must not be too large and complicated with overheads and calling sequences. The latter would defeat the purpose of agility and reuse making it easier to write the required functionality from scratch.
- Corporate source has value beyond software product development. Examples include: computing infrastructure and internal decision support tools.
- Some HP projects already make available their code on their own web service and in this case don't see the need to move over to the corporate source system.

Case Study – IBM (IBM Community Source, 2007)

Background

IBM, although a supporter of open source projects[1], has long been a traditional software development company, choosing to develop its software in a proprietary manner. It has now become a leader in adoption of OSS and the OSS model in the corporation (Capek *et al*, 2005). It is this fostering of open source outside the company that has afforded IBM the opportunity to gauge some of the attractive attributes of open source projects and bring them into the corporate environment (Capek *et al* 2005).

Community source is the term used by IBM instead of Inner Source. From the initial data we know, Community Source methods fit into Inner source but not all Inner source principle seem to be present in the Community Source solution.

It is clear that IBM has invested significantly in Inner Source (IBM Community Source, 2007). IBM sees Inner Source as a good way to liberate the creativity of programmers, drive efficiency and bring products to the marketplace faster.

The company sees Inner Source as a new development methodology for the entire globally disturbed company. A strategic decision to decompose technologies into a number of components to aid reuse has been a driving factor in change. The use of Inner Source is seen as a good way to aid cross pollination of expertise, ideas and requirements between all the locations of IBM labs. IBM has essentially borrowed from open source many philosophies, strategies, tools and culture to transform IBM's development practices to further support global component development.

IBM sees Inner source as a means to an end in effect. In the not too distant future every software company will need to componentise their code and Inner Source is a way to do this while giving a lot of freedom to the developer and promoting innovation across the company.

Inner Source Applied

IBM has a hundred projects that are currently using the Community Source infrastructure and has over 2,000 users that are currently registered and using the Community Source tools and processes (Taft 2005). The company has tools that store the actual source logic using repositories very much like SourceForge, documentation and specs, news and bulletins, patches and fixes, and educational material and broader community tools and support. The company states that development is 30 percent faster using an open source model.

[1] IBM has played a large part in the success of open source projects such as the Linux operating system, Apache web server and the Eclipse software development environment. Each of these projects can be considered high profile, successful, OSS initiatives.

Implications and problems
As the IBM program involving Inner Source continues, the company is really trying to harvest the programming culture and practices that have been the product of decades of top class development at IBM, along with these practices IBM are attempting to blend some of these new techniques that have become apparent through exposure to open source communities. Older projects will gradually adopt the new development methods and new projects will benefit from traditional IBM disciplines. This approach is seen as the best way to blend the two worlds and poses the biggest challenge going forward.

Key Results
IBM sees Inner Source as a good way to liberate the creativity of programmers, drive efficiency and bring products to the marketplace faster. The company states that development is 30 percent faster using an open source model.

Analysis
IBM has a clear business vision that focuses on the components of their software and the gradual introduction of open source methods to the company wide development environment. IBM seems to focus heavily on the software development methodology instead of the organisational structure or the people behind the software methods. However little is known about the extent of the programme and the detailed organisational changes needed to implement the system. The information we do know about IBM focuses on the technical changes to code architecture which forms only a part of Inner Source principles.

Section Conclusion
In this section we have examined three cases studies in detail, Bell Labs, Hewlett Packard and IBM. Each case was outlined in terms of its background, how inner source was applied within each company, the implications and problems that arose as a rest of the implementation, the key results drawn from the case study and an analysis of each case study. The cases helped establish ongoing implementations of Inner Source and the issues surrounding such implementations. From this section we have gained further understanding regarding:

- Differences between Inner Source and traditional software development.
- Differences between Inner Source and open source.
- Perceived benefits of Inner Source usage.
- Actual benefits of Inner Source usage.
- Problems with implementing inner source solutions and some of the possible factors causing these problems.
- Principles on Inner Source.
- The cultural changes required for a successful Inner Source solution.

We now move on to the next section entitled “Locally to Globally Distributed Development”. We have seen a number of problems and issues, in the implementation of Inner Source solutions, while some of the problems can easily be attributed to cultural differences; one major difference is that which we will now examine in the next section. In particular we use this section to deepen our understanding of the theoretical understanding of the issues involved in moving from traditional software development to inner source.

4 Locally (face-to-face) to Globally Distributed Development

Interaction is an important factor to be considered in Inner Source as it can often be the case that companies wish to capitalise on the success of open source in the area of globally distributed software development (Melian, 2007). We have chosen here to examine this area in particular due to fact that it is inherent to our observations of Inner Source that corporations move from a local style development and communication to a global style of development and communication. This is certainly an expected outcome in the cases we examined. While we do not expect this to hold true for all future cases of Inner Source (it is certainly envisaged that companies with experiences of Global Software Development would also have knowledge of this subject), we do however feel that in examining some of the theory in this chapter will shed more light on the underling factors which cause the effects observed in the case studies of Section 3.

A major problem for dispersed organizations conducting research and development is to organise work so that the participants can effectively use one another's expertise and knowledge without the use of frequent face-to-face interaction. Collaboration often promotes innovation and it is this collaboration and innovation which the businesses examined in our case studies have also tried to create by fostering an Inner Source environment.

There are at least three important features of face to face interaction that influence communication (Bavelas *et al*. 1997), these include face to face dialogue with unrestricted verbal expression, meaningful non verbal acts such as gestures and facial displays and instantaneous collaboration between speaker and listener. It is considered that face to face communication is a basic skill set and the base from which communication can grow. Communication tools other than from the realm of face to face communication require special skills and education to master.

It is understood that in using the communications tools of open source to communicate as opposed to face to face interaction that a huge deal of information may be "lost in the translation" For example in face to face communication we often use paralanguage to imply meaning. Paralanguage is metadata applied to the verbal communication, such as the use of inflection in a sentence. This has the ability to completely change the meaning of the sentence. Another example of information lost in non face-to-face communication is kinesics or body language. This is what Goffman (1959) referred to as "give-offs" – the often non-verbal signs that help to situate and verify the things we say. Give-offs, which are essential to human communication, are largely lost in ICT mediated communication due to the difference between face-to-face conversation and conversation through and by means of ICT as discussed below.

To elaborate this issue we will turn to ten features of a casual face-to-face situation suggested by Clark (1996). Below we will discuss how the inner/open source communication situation may be understood in terms of these characteristics. Clark's

features are as follows: co-presence (participants share the same physical environment), visibility and audibility (they see and hear each other), instantaneity (they recognize each other's action with no perceivable delay), evanescence (the medium fades immediately), recordlessness (actions do not leave any record or artefact), simultaneity (participants may receive and produce at once and simultaneously), extemporaneity (actions are formulated and executed in real time), self-determination (participants determine for themselves what actions to take and when), and self-expression (participants take actions as themselves). Clark (1996) describes the first four of these features as related to the immediacy of face-to-face conversations, the next three as related to the medium, and the final three as related to the control of the conversation. In the remainder of this section these features are discussed from the perspective of communication through ICT tools in a community. Table 2 below summarizes the interpretation of Clark's (1996) features from the open/inner source perspective as used in the following discussion. The comparison and interpretation is based on the notion that an inner-source project/community represents a special type of norm-based context and that the introduction of ICT tools implies a special type of medium for conversation.

	Feature	*Face-to-face conversation*	*Inner-source project*
Immediacy	Co-presence	Participants share the same physical environment	Participants may not share the same physical environment
	Visibility	Participants can see each other	Participants may not see each other
	Audibility	Participants can hear each other	Participants may not hear each other
	Instantaneity	Participants perceive each other's actions with no perceptible delay	Participants may perceive each other's actions with considerable delay
Medium	Evanescence	Medium is evanescent – it fades quickly	Medium is persistent – it may stay until the system is shut down
	Recordlessness	Participants' actions leave no record or artefact	Participants' actions leave a record in the system (logs, databases, etc)
	Simultaneity	Participants can produce and receive at once and simultaneously	Participants either produce or receive as separate acts
Control	Extemporaneity	Participants formulate and execute their actions extemporaneously, in real time	Participants formulate and execute their actions/utterances reflectively during extended amounts of time
	Self-determination	Participants determine for themselves what actions to take when	Project norms and ICT design determine what actions to take when
	Self-expression	Participants take actions as themselves	Participants may take action on behalf of other people and their organization

Table 2: Features of face-to-face conversations (Clark, 1996) compared to communication through and by means of ICT in an inner source project/community (adapted from Ågerfalk, 2004).

Implications of ICT enabled communication in Inner Source Development

When communicating though ICT, participants in a conversation are typically not co-present. When lacking co-presence, implicit references to anything outside of the system are likely to lead to confusion and misunderstanding. In the context of inner source there is typically neither visibility nor audibility. 'Typically' since participants may see and hear each other with the adoption of multi-media type of interfaces even though such means of communication are rare in open source projects, but could perhaps be adopted more readily in inner source projects. As indicated above, when participants cannot see or hear each other, gestures, sighs, and other audiovisual clues (give offs) are not possible to utilize.

In contrast to the recordlessness and evanescence of face-to-face conversations, one of the advantages of ICT is that communication may actually leave traces in the system. For example, conversations may be logged and even semantically tagged for easy retrieval. In a sense, ICT based systems can be understood as the set of social interactions they afford and support – they provide an action potential (cf. Carroll, 1996). That is, actions are only self-determined so far as the system design allows (see Goldkuhl & Ågerfalk, 2002). So, while turn taking is self-determined in face-to-face settings and constrained by the very fact that any system restricts possible actions to perform (see above), business rules and norms (often codified as a software development process or method) largely determine what actions to take when in a development project. Developer interactions are part of an 'action structure' and the set of possible actions to perform at any particular point in time changes as the interaction proceeds.

When performing actions through and by means of an ICT system, the user interface is typically not evanescent but persistent for as long as participants desire. Unless an error occurs, a displayed message (such as a posting to an e-mail list) will stay displayed until it is deliberately turned off. This condition gives rise to the opportunity to formulate and execute actions reflectively during extended amounts of time. The lack of co-presence (see above) also suggests that systems should be designed as to take into account where and how messages are received; receipt and interpretation should be possible at desired places and in desirable ways.

Even though ICT systems are interactive participants may not perceive each other's actions instantaneously, but rather with considerable delay. Consider, for example, a question about a particular software module posted during the night (which may actually be during the poster's day). Such a question may not be displayed by a relevant peer until the next day. When using an ICT system, participants either produce or receive – but not simultaneously, as in a face-to-face setting. This implies that developers must understand that no feedback will be given until a message has been delivered and interpreted. For obvious reasons, the design of the system and the project as a whole should strive to minimize such delayed feedback.

All communication has an ingredient of self-expression. Nonetheless, developers on a project, as well as the ICT systems themselves, may take action on behalf of other people and their organization. This means that social obligations may be created through an ICT system without the responsible parties' direct interaction with the system. This condition suggests that responsibilities should be allocated to human actors and ICT systems so that users gain maximum support (e.g. in terms of decision support *vs.* automated actions), that description and explanation of the system's

performed and scheduled future action(s) are readily available, and that all actors involved are aware of their action relationships even though they may not be directly interacting with the system.

The majority of ICT based tools for distance communication highly depend on textual communication and a result can be a challenge for creative thinking people, who tend to be more visual thinkers than textual. Employing the use of new innovative visual distance communication tools could be one way to foster innovation in Inner Source communities and to bring the inner source conversation context closer to the "ideal" face-to-face setting.

In this section we have examined some of the underling fundamental changes that occur and the issues that arise with the introduction and use of Inner Source. We will now move from the more theoretical aspects examined here in this section to the more pragmatic issues that have arose as a result of our examination of Inner Source; namely that of managing and organising Inner Source.

5 Managing and organising Inner Source

In this section we look at how traditional organisational and management structures must change differs from that which would be of benefit to Inner Source.

In HP the employees were conscious of posting to the Inner Source community as their messages were going out to a wider community. Managers felt that the use of the Inner Source solution would lead to an increase in quality due to the open nature of the Inner Source solution. People were insecure about posting their work for a company wide audience but eventually felt a lot of pride associated with contributing quality work (Melian, 2007).

Visibility is an important issue to consider in Inner Source. As traditional projects move from traditional style development, where the code and communication is usually shared only within the project group where face to face communication is common, to Inner Source style development visibility of the developer's work and textual communication increases significantly. The issues surrounding this may include an increase in quality of work as a result of many people being able to give helpful feedback but it also takes more time and effort for people to communicate in this environment.

As this visibility increases so too must the companies' security efforts, especially in the case where various contractors are employed within different areas of the company. Where in the traditional development environment, the access of people to code is localised within company departments and easier to manage, in contrast the very openness sought after in the use of inner source makes security and access control a lot more difficult. In HP digital badges were used to control access and vendor access to digital badges was something new that had to be managed as a result of the use of inner source. This creates an additional issue to be dealt with when bringing in a new vendor, as not only must the new vendor have a physical id but also a digital badge for inner source access (Melian, 2007). This creates a certain level of trust within the company that the code is safe but when using external vendors the perception is that they have no control over what security the vendor is using once the vendor is given access. Some of the fears encountered as a result in the increase in openness include fear of granting access to external access of the code to vendors and also by allowing other groups to see what we do, we run the risk of someone seeing an opportunity to downsize our department and take over our work. The other fear among developers is that by everyone in the company potentially having greater access to the inner workings of the company any employee that leaves permanently may release trade secrets to competitors and as a result threaten jobs in the company.

There is also the argument that in an environment where there exists a centralised repository where a lot of information is freely available on a company wide basis and if free to traverse and browse, that this environment could make people reluctant to use digital media for some forms of communication. This was certainly the case in HP where developers rejected the suggestion of the use of video equipment to record idea generation and brainstorming in a meeting direct because of the likelihood that the

media would be made freely available in a company wide system where the media could live for a significant period of time.

Traditional Management and Organising	**Inner Source Management and Organising**
Silo Mentality: Conflicting Interests internal actors, adversarial external relationships	Big family: Blend of interests founded on equality, established interdependent relationships, collaboration and co-learning.
Garage mentality: Restricted access to knowledge/information even internally.	Soccer team metaphor: Progressive more open access to knowledge and information
Cathedral: Top down imperative control	Bazaar: Creating learning communities
Machine metaphor: Fixed structures and procedures	Brain metaphor: Flexible structures, emergently progressing
Bureaucratic organizing: Detailed and task related assignments,	Self managing/regulating groups, creativity and innovativeness, empowered work. Behaviour and action based on a common understanding.

Table 3: Traditional Management Vs Inner Source Management (Melian, 2007)

Introduction of an Inner Source system that is perceived to be open and a controlling factor by management causes serious undermining of developers. If Inner Source is to be used effectively and to promote innovation then introduction of an Inner Source system can aid decentralized and geographically dispersed development teams to work together on a project. The openness of the Inner Source solution deployed in HP caused developers to be a lot more careful about the quality of their work and comments they were contributing. The openness of the system also caused some users to become very concerned over job security. It is clear that management and organisational structure must change and adapt to the use of Inner Source as must the users of the system.

Table four below outlines the problems that arose in the examined case studies and literature on Inner Source. The table outlines the background difference between traditional software development and Inner Source software development and the corresponding problems arising as a result in the change in environment. While some of these problems are a direct result of implementing inner source not all problems have a net negative effect. For example the problem of forking of code, depending on the situation is probably better than the previous situation where code was developed in parallel with know information about ongoing development of other project available. At least in the case of code forking the developers are aware of other code existing and possible solutions to bugs existing.

Traditional Development Environment	**Inner Source Development Environment**	**Inner Source Problems and Issues as a result of change.**
Localised approach to development	Centralised approach to development.	Forking of code.
Procedures and knowledge of how to account for software based on set formula assuming certain structures are present.	Absence of procedure of how to attribute value and cost when resources and input is company wide.	Inner source solutions not being valued or accounted for correctly.
Local tools and standards based on each project preferences and closest development partners.	Company wide standards, tools and practices.	Pushing of company wide code practice, standards and tools seen as excessive and unnecessary.
Defined timescale of development and testing, with complete control of the code being held by the local project group.	Constant development of code with no group having the ability to stop development for a testing cycle.	Code freezing not possible before project release.
Communication and code access on a local level.	Open CVS, mailing lists available company wide.	Privacy of ideas and conversations an issue.
Tight time constraints	No time constraints	Possible delay in time to market.
IP a closely guarded secret	IP released company wide	Every employee knowing the business of the entire company a threat to jobs when that employee leaves.
Heavy hierarchical business organisation structures	Structured more loose roles	Concerns over project groups taking over the responsibilities of other groups.
Focus on the binary shipping	Focus on CVS commits	Possible, delay in time to market.

Table 4: Problems encountered in implementing Inner Source

6 Principles of Open Source versus Inner Source

In this section we will discuss how we define open source and how we define inner source. This section draws from the output of all previous sections, to accurately define in detail principles of Inner Source. We also look at the principles of open source and compare and contrast them with the principles of Inner Source.

How do we determine if a software product, piece of code or any other entity is entitled to be labelled Open Source? There are many meanings of open source (Gacek, 2004) and there are of course practices that are seen in many open source projects (Halloran *et al* 2002). One definite metric is if it has a licence recognised by the Open Source Initiative (OSI). Sometimes this is not necessarily a clear enough attribute to look for as we have seen in the case of Sugar CRM which claims to be an open source product as its' code is available but in the case of Sugar CRM which is licensed to include the stipulation that the Sugar logo must be used on every single page of any derivative work. Many argue in this case Sugar CRM is not in fact open source as not only does this go against the idea of open source but it is also not a recognised license. Recent changes have been made by both Sugar CRM and OSI that have seen the Sugar CRM product fully recognised by the OSI.

However, the emerging area of inner source does not have an organisation such as the OSI to look to for an approved licence with which they can release products under and so we have chosen to look more clearly at the principle of Open Source and the contrast and similarities that exist with Inner Source.

Some observations on the principles of Inner Source have become apparent upon review of the areas.

- CVS and associated repository tools, along with community communication tools (mailing lists, web forums, etc) were part of a common solution.
- The quality of the code increased as a result of the inner source implementation.
- Communication issues were common.
- Forked code was a problem that effected support of the common infrastructure.
- Costing of code produced in the new method was not valued correctly.
- A common area to choose for inner source initial projects was that of research or emerging technologies.
- The use of domain experts from across the company can increase the product's quality significantly.
- Significant organisational structural changes were required.
- The changes often would lead to reduced time to market for products using inner source.
- The changes often would lead to reduced costs in producing a product using inner source.
- There needs to exist a large degree of overlap between community values and governance structure.

- Motivation of company and contributors needs to be clear but crystal clear for the company.
- Boundaries are especially important, where developers can work and where contribution is not really requested. Case example Maemo and Nokia has a one way code release (any open source developer contribution will not feed back into the Nokia code for a number of reasons).
- Success factors that play a role in open source include simple communication mechanisms with a low learning curve.
- Computer mediated groups generate more ideas (Nunamaker *et al*, 1991)
- The breadth of ideas is a lot greater in asynchronous communications media such as is found in open source projects (Benbunan-Fich *et al*, 1999).
- Open source can often be seen as an experimental approach to software development in that coordination takes place after action.
- Open source can be described as constellations of communities who can work on what they choose rather than a formal set of tasks mandated by business objectives. Formalisation inhibits innovation (Klincewicz, 2005).
- Open source development is co-evolution of code, knowledge and community.
- As with implementing any change in a company and particularly open source changes to a company there is evidence to suggest that employees will resist change. (Jaaksi, 2007)

The issue of community is an important issue in Inner Source and was mentioned multiple times in the case studies examined, however the how do we define "community"? Should the "community" have certain traits in order to be able to support Inner Source? Hillery (1955) found 94 different definitions of community with only one common element that of people. Since 1955 we have seen a sea change in our world which only serves to complicate the definition of community. The issue of community in Inner Source needs further examination that is best served in the second deliverable on this topic where we can revisit cases with these questions in hand.

From our initial examination of the area and this issue in general we have observed the community to include all developers and staff with code interaction duties at least and possibly all employees. The issue of characteristics of suitable communities of Inner Source did not arise as the case studies had an inclusive approach which everyone in the company has access to and an experimental approach where they envisage a better product is everyone has input and see if implementing it results in a positive experience.

Open Source Principles (Eric Raymond)	**Inner Source**
1. Good programmers know what to write. Great know what to rewrite (and reuse): Promotes parallelism and redundancy. Latter leads to knowledge sharing and transfer	This principle holds true for Inner Source, code reuse is a key benefit to any organisation attempting to employ Inner Source.
2. To solve an interesting problem, start with one interesting to you: knowledge generating. Intrinsic self-selection.	This principle does not hold true from the developers standpoint, as developers cannot in all situations choose the most interesting problem to work on, however there are exceptions to the rule which we will see later
3. Release early. Release often: reward feedback. Prompt feedback leads to tight development, debugging and testing (Maintenance problem is that you don't even recognise your own code 6 months later).	This principle is as true in Open Source as it is in Inner Source.
4. Given enough eyeballs, every bug is shallow: community feedback, author and community review again – double-loop learning. Self organisation of community and knowledge	This is a major benefit of Inner Source and the aspect of leveraging the community feedback extends in Inner Source to include bugs, community feedback and user requirements.
5. Provided coordinator had medium such as Internet and knows how to lead without coercion, many heads are inevitably better than one (Brian Fitzgerald: 2 principles here. Internet and low coercion).	(A)The coordinator must be strongly connected to the group, and Inner source communication tools provided to aid this. (B) Low coercion is also a principle of Inner Source but in a very different way, in that high coercion is inherent in a corporate environment but the Inner Source environment should adopt a low coercion ideology.

Table 5: Open Source and Inner Source principles

Table six below compares and contrasts traditional software development to inner source software development based on the information gathered from our findings in the case studies examined in Section 3 and the expectations and perceptions of the people aiming to capitalise on open source. We use this contrast of traditional software development and inner source software development not to suggest that the two terms are polar opposites or by implication, that traditional software development and open source software development are polar opposites, but we instead use this comparison in order to provide the most useful areas, that aid in the identification of activities that typically occur in Inner Source. There are of course cases of cross-over between both types of development for example (Garcia 2007) argues that much of the characteristics found in traditional development are found in open source development, and in particular (Capiluppi and Michlmayr, 2007) argue that many open source projects never migrate from centralized development to distributed

development which are commonly held polar opposites between traditional development and open source development (Raymond 2001). Another example of this cross-over is when companies attempt to build a community around a corporate product such as examined in (West and O'Mahony 2004) where companies which released developed code to a public community much in the same way Nokia release code to the Maemo community. Maemo is Nokia's linux based platform for handheld devices, where code is released to the community but no changes or applications developed by the community are accepted back into the Nokia product offering.

Traditional Software Development	**Inner Source Software Development**
Code held in a local context	CVS and associated tool usage
Focus on release binary	Focus moves to producing source code for CVS submission.
Much face to face communication	More towards distance interaction
Code review takes place at key points	Constant code review
Code quality is dependent of a few key people within a project	Code quality is a concern of everyone across multiple projects
Tools and methods vary from project to project.	Corporate wide policy of tools and methods
Costing of a project is easily tracked with defined resources and costs being easily tracked with each project group.	Costing of inner source becomes much more difficult as code contributors can potentially come from any department or project
Code exists in an isolated manner with the possibility of major company resources being used to solve a problem multiple times.	Resources are less likely to be wasted on solving a problem many times as anyone can check if someone is attempting a solution with the correct communication structures in place.
Breadth of ideas and thinking likely to be a lot smaller.	Breadth of ideas greatly increased.

Table 6: Traditional Software Development and Inner Source Software Development

In this section, we examined Inner Source in more detail than previously outlined through outlining observations that hold true across multiple case studies. We examined the principles of open source and compared and contrasted them with the principles of Inner Source. We also examined the principle differences between traditional development and Inner Source development.

7 Inner Source for OPAALS

This section examines the potential for Inner Source usage in the OPAALS community, particularly for researchers for whom software development is not the primary focus. We have attempted to address the relevance of this work to OPAALS in as a complete a fashion as we have been afforded, however as we will be examining further case studies on Inner Source in greater detail in Phase II of this project we encounter a boundary beyond which detailed assertions to the applicability of Inner Source in areas of OPAALS could prove to be a negative experience. This is certainly the case in projects releasing code to the open source community too soon (Michlmayr, 2007). We feel this could also be a factor of major importance both in the implementation of Inner Source in software engineering and in implementing Inner Source concepts beyond software.

As was mentioned in the HP case study, inner source methods have uses beyond software and technology specific areas. In fact elements of inner source can all readily be seen either in the OPAALS OKS or in requirements raised by the OKS user group or indeed issues raised in the OPAALS meetings. This goes beyond some of the obvious practical benefits that come with Inner source usage such as code reuse, use of common components and increase in quality of code. In particular the principles of Inner Source that stand out as being most transferable to OPAALS include: transparency and awareness of work ongoing, community, focus shift from final output submission to submitting quality work to the community at an earlier stage for constructive community input and increase in quality of work as a result. Table seven below examines in more detail some of the transferable principles of Inner Source to OPAALS.

Where Inner Source works well seems to be in areas of software, that is investigating up and coming areas of software or specifically technology research departments. Another characteristic of inner source is the interest it generates in a large group of people where a cross cutting concern or aspect is present. So in the case of Lucent everyone wanted to use the high quality product to integrate with their own product line. There were many benefits to participation and so the incentive was high. This is a very important trait which would be of huge benefit to the OPAALS community. If similar characteristics could be identified for the research community then the authors believe that inner source could have a significant resonance within the OPAALS community.

Examples of cross cutting concerns of all researchers in OPAALS may include:

- Emerging trends in research domains
- Research methods
- Peer review
- Quality publications

Other examples of cross cutting concerns may exist in sub domains within OPAALS and these two could be capitalised on. As mentioned previously there exists a trade off between finding a common area to work together on, that is a concern or interest to

many researchers, and finding a common area that is also of high value to a lot of researchers. The success factors of inner source examples however are not just limited to cross cutting concerns. There are other elements such as excellence or quality of the commons, timeliness in content and domain experts willing to improve the commons with or without immediate reward. The issue of management structure is probably the biggest issue to overcome, not in the management of OPAALS, instead the management or organisation structures inherent in the various different OPAALS researcher's domains.

	Traditional Software Development	*Inner Source Software Development*	*OPAALS Research Environment*
Implemented in OPAALS	Code held in a local context	CVS and associated tool usage	Participants may not share the same physical environment
	Much face to face communication	More towards distance interaction	More distance interaction.
	Breadth of ideas and thinking likely to be a lot smaller.	Breadth of ideas greatly increased.	Breadth of ideas greatly increased.
	Focus on release binary	Focus moves to producing source code for CVS submission.	Researcher focus should shift to OKS submission.
	Code quality is dependent of a few key people within a project	Code quality is a concern of everyone across multiple projects	Research quality is a cross cutting concern of all OPAALS partners.
Planned	Code review takes place at key points	Constant code review	Constant content review may or may maynot be plausible.
	Code exists in an isolated manner with the possibility of major company resources being used to solve a problem multiple times.	Resources are less likely to be wasted on solving a problem many times as anyone can check if someone is attempting a solution with the correct communication structures in place.	In the long term a global OKS would be a great reference point to examine current research.
Not Applicable	Tools and methods vary from project to project.	Corporate wide policy of tools and methods	Tools and methods still vary from partner to partner.
	Costing of a project is easily tracked with defined resources and costs being easily tracked with each project group.	Costing of inner source becomes much more difficult as code contributors can potentially come from any department or project	Costing not necessarily an issue but attributing authors might be a comparable issue.

Table 7: Traditional Software Development and Inner Source Software Development and OPAALS Research Environment compared.

8 Conclusion

In conclusion we have examined a number of case studies involving various inner source solutions. We looked at the expected results of the inner source solution and the motivations behind the deployment. We then looked at the actual benefits and the problems associated with the deployment.

We examined the area of the change from locally to globally distributed development and in particular face to face interaction versus distance interaction for the reason that an inner source solution brings and makes transferable traditional development to a globally distributed solution, and for that reason we examined these interaction types and the implications for inner source.

The issues of managing and organising inner source were also examined. These issues are especially important to inner source and as we have seen some critical changes were required in the successful implantation of inner source. In particular we have seen that to create innovation a looser approach to managing is required and the organisational structure must undergo some key changes to foster growth of inner source. One particular resounding finding is that organisational structure is the biggest issue that must be constantly examined when implementing an inner source solution.

Inner Source principles were set out next from the background material covered in previous sections. We looked at emerging principles in terms of how they relate to open source and also how they relate to traditional software development. These principles help to characterise the inner source phenomenon more clearly and set out the scope of our future research and also help us to explore the potential cross over of inner source principles beyond software.

The change occurring in the corporate world in order to leverage the successes of open source is a phenomenon not limited to software. It is this potential of inner source beyond software that has implications for the OPAALS research community. We have outlined the emerging trends on inner source and the structure of some inner source solutions. We feel that some of the ideas expressed here could enjoy success in the OPAALS domain and could make a profound impact in how we work together in the future. However as in the corporate world, transitions are not easy and require cultural, organisational and structural changes if the success of open source is to be transferred successfully.

9 References:

Ågerfalk P J (2004). Investigating Actability Dimensions: A Language/Action Perspective on Criteria for Information Systems Evaluation, Interacting with Computers, 16(5), pp. 957–988.

Ågerfalk, P.J. (2005). Studying Persistent Conversations in an Open Source Context: A Conceptual Framework, In Proceedings of Promote IT 2005, (Eds, Bubenko jr., J et al.) Studentlitteratur, Lund.

Arlein, R. and Gurbani, V., An Extensible Framework for Constructing Session Initiation Protocol (SIP) User Agents. Bell Labs Technical Journal, 9, 3 (November 2004), p. 87-100.

Babbie, E. (1995). The practice of social research (7th ed.). Belmont, CA: Wadsworth.

Benbunan-Fich, R., and Roxanne Hiltz, S. "Impacts of Asynchronous Learning Networks on Individual and Group Problem Solving: A Field Experiment," Group Decision and Negotiation (8) 1999, pp 409-426.

Bergquist, M. and Ljungberg, J. "The Power of Gifts: Organising Social Relationships in Open Source Communities" Information Systems Journal (11:4) December, 2004 pp.305-320.

Capek, P.G., Frank, S.P., Gerdt, S. and Shields, D. (2005) A history of IBM's open-source involvement and strategy, IBM Systems Journal, 44, 2, 249-257.

Capiluppi, A. and M. Michlmayr (2007) 'From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects' in IFIP International Federation for Information Processing, Volume 234, Open Source Development. Adoption and Innovation, eds. J. Feller, Fitzgerald. B., Scacchi, W., Sillitti, A. Boston: Springer, pp. 31-44.

Carmel, E. and Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. IEEE Software, 18 (2), 22-29.

Carroll, J. M. (1996). "Becoming Social: expanding scenario-based approaches in HCI." Behaviour & Information Technology 15(4): 266–275.

Clark, H. H. (1996). Using Language. Cambridge, Cambridge University Press.

Conway, M.E., How Do Committees Invent? Datamation, 14, 4 (1968), p. 28-31.

Crabtree, B. F., & Miller, W. L. (Eds.). (1992). Doing qualitative research: Research methods for primary care (Vol. 3). Newbury Park, CA: Sage.

Crowston, Kevin, Scozzi, Barbara, Open source software projects as virtual organizations: Competency rallying for software development. IEE Proceedings Software, 149(1), 3-17. (2002).

Cubranic, D. (1999). Open-Source Software Development. In 2nd Workshop on Software Engineering over the Internet, 7p.

Cusumano, M. and R. Selby (1997) "How Microsoft Builds Software", Communications of the ACM, 40 (6), pp. 53-61.

Gallivan, M.J. "Striking a Balance between Trust and Control in a Virtual Organization: A Content Analysis of Open Source Software Case Studies," Information Systems Journal (11:4) 2001, pp 277-304.

Gurbani, V., Garvert, A., and Herbsleb, J., A Case Study of Open Source Tools and Practices in a Commercial Setting, Proceedings of the 5th ACM Workshop on Open Source Software Engineering (WOSSE), (May 2005), pp. 24-29.

Dinh-Trong, T. and Bieman, J.M. (2004) Open Source Software Development: A Case Study of FreeBSD. In Proceedings of the 10th International Symposium on Software Metrics, IEEE Computer Society.

Dinkelacker, J., Garg, Corporate Source: Applying Open Source Concepts to a Corporate Environment. HP Technical Report 2001. http://www.hpl.hp.com/techreports/2001/HPL-2001-135.html?jumpid=reg_R1002_USEN

Dinkelacker, J., Garg, P., Miller, R., and Nelson, D., Progressive Open Source. HP Technical Report 2001. http://www.hpl.hp.com/techreports/2001/HPL-2001-233.html.

Dinkelacker, J., Garg, P., Miller, R., and Nelson, D., Progressive Open Source. In Proceedings of the 2002 ACM International Conference on Software Engineering (ICSE'02), pp. 177-184, May 2002.

Erenkrantz, J.R. and Taylor, R.N. (2003). Supporting Distributed and Decentralized Projects: Drawing Lessons from the Open Source Community. In The 1st Workshop on Open Source in an Industrial Context, p. 21-30, <wwwbib.informatik.tu/muenchen.de/infberichte/2003/TUM-I0319.pdf>

Fielding, R.T and Kaiser, G. (1997). The Apache HTTP server project. IEEE Internet Computing, 1 (4), 88-90.

FLOSS Report EU http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf

Gacek, C. and Arief, B. (2004). The Many Meanings of Open Source. IEEE Software, 21 (1), 34-40.

Gallivan, M.J. “Striking a Balance between Trust and Control in a Virtual Organisation: A content Analysis of Open Source Software Case Studies” Information Systems Journal (11:4) December, 2004 pp.277-304.

Garcia, M.J. and E.W Steinmueller (2003) ‘The Open Source Way of Working: a New Paradigm for the division of labour in Software development?’ INK Open source working paper No.92, SPRU Science and Technology Policy Research, University of Sussex, available at: http://www.sussex.ac.uk/spru/1-6-1-2-1.html, last accessed 15/08/07.

German, D.M. (2003). GNOME, a case of open source global software development. In International Workshop on Global Software Development, p. 39-43, <gsd2003.cs.uvic.ca/gsd2003proceedings.pdf>

Goffman E (1959) “The Presentation of Self in Everyday Life,” Doubleday: Garden City, New York.

Goldkuhl, G. and P. J. Ågerfalk (2002). Actability: a way to understand information systems pragmatics. In Coordination and Communication Using Signs: Studies in Organisational

Semiotics 2. K. Liu, R. J. Clarke, P. B. Andersen and R. K. Stamper. Boston, Kluwer Academic Publishers: 85–113.

Gurbani, V.K., Garvert, A. and Herbsleb, J.D. (2005). A Case Study of Open Source Tools and Practices in a Commercial Setting. In Proceedings of the 5th Workshop on Open Source Software Engineering, p. 24-29, ACM.

Halloran, T.J. and Scherlis, W.L. High Quality and Open Source Practices. in Meeting Challenges and Surviving Success: 2nd Workshop on Open Source Software Engineering. 2002. Orlando, FL.

Herbsleb, J.D. and Grinter, R.E., Architectures, Coordination, and Distance: Conway's Law and Beyond. IEEE Software, Sept./Oct., (1999), p. 63-70.

Herbsleb, J.D. and Mockus, A., An Empirical Study of Speed and Communication in Globally-Distributed Software Development. IEEE Transactions on Software Engineering, 29, 3 (2003), p. 1-14.

Hillery, G. A. (1955) "Definitions of Community: Areas of Agreement", *Rural Sociology,* **20** pp. 111-123.

IBM Community Source Interview, http://www.betanews.com/article/IBM_Turns_to_Open_Source_Development/1118688437 2007.

Jaaksi, Ari (2007) 'Experiences on Product Development on Open Source Software' in IFIP International Federation for Information Processing, Volume 234, Open Source Development. Adoption and Innovation, eds. J. Feller, Fitzgerald. B., Scacchi, W., Sillitti, A. Boston: Springer, pp. 85-96.

Kirsch, L.J "Deploying Common Systems Globally: The Dynamics of Control" Information Systems Research (15:4), December 2004, pp. 375-395.

Klincewicz K. (2005), Innovativeness of Open Source software projects. MIT Working Papers.

Krishnamurthy, S. (2002). Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. First Monday, 7 (6), <www.firstmonday.org/issues/issue7_6/krishnamurthy/>

MacCormack, A., Rusnak, J., and Baldwin, C., Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code, in Harvard Business School Working Paper. 2004: Boston, MA 02163.

Melian, C. Progressive Open Source, PhD Thesis, Stockholm School of Economics 2007.

Michlmayr, M. Quality Improvement in Volunteer Free and Open Source Software Projects, PhD Thesis, University of Cambridge 2007.

Mockus, A. and Herbsleb, J.D. (2002). Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source?. In Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering, p. 19-25, <opensource.ucc.ie/icse2002/MockusGerbsleb.pdf>

Mockus, A., Fielding, R., and Herbsleb, J.D., Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11, 3 (2002), p. 309-346.

Moon, J.Y. and Sproull, L. (2000). Essence of Distributed Work: The Case of the Linux Kernel. First Monday, 5 (11), <www.firstmonday.org/issues/issue5_11/moon/>

Nunamaker, J. F. Dennis, Alan R. Valacich, Joseph S. Vogel, Douglas. George, Joey F. Electronic meeting systems, Communications of the ACM, v.34 n.7, p.40-61, July 1991.

Open Source Business, The Economist, March 16th 2006
http://www.economist.com/business/displayStory.cfm?story_id=5624944

Patton, M (1990) Qualitative evaluation and research methods, Sage Publications, Newbury Park, California.

Rosenberg, J., et al., SIP: Session Initiation Protocol, IETF RFC 3261, July 2002, http://www.ietf.org/rfc/rfc3261.txt

Sagers, G "The Influence of Network Governance Factors on Success in Open Source Software Development Projects. Proceedings of the Twenty-Fifth International Conference on Information Systems. Agarwal, R and Kirsch, L. (eds.) Washington DC, December 2004, pp.427-438.

Sarma, A. (2005). A Survey of Collaborative Tools in Software Development, ISR Technical Report # UCI-ISR-05-3, Institute for Software Research, University of California, Irvine <www.isr.uci-edu/tech-report.html>

Scacchi, W., Understanding the requirements for developing open source software systems. IEE Proceedings on Software,149, 1 (February 2002), p. 24-39.

Shah, S. Understanding the Nature of Participation and Coordination in Open and Gated Source Software Development Communities. In Annual Meeting of the Academy of Management. 2004.

Raymond, E., The Cathedral and the Bazaar. O'Reilly Publishing Company, First Edition, (February 2001).

West, J. and C.S. O'Mahony (2004). 'Contrasting Community Building in Sponsored and Community Founded Open Source Projects', available at: http://opensource.mit.edu/papers/westomahony.pdf , last accessed 06/08/07.

Woods, D., and Guliani, G. Open Source for the Enterprise, O'Reilly Media, Sebastopol, CA, 2005.

Zikmund, W. G. (1982). Exploring marketing research (4th ed.). Orlando, FL: Dryden Press.

10 Appendix (Reviewers Comments)

We have included below all reviewers comments. These comments are greatly appreciated and have focused our attention on areas of the deliverable that we needed to improve and clarify. We have addressed most of the comments below except in the case of a small number of comments that arose as a result of our not defining the scope of this deliverable and its focus. As mentioned earlier in the text, this is the first deliverable of two in the area on Inner Source and the second deliverable in phase two will examine in further detail the issues raised in our work here in this deliverable.

There are a number of interesting issues and areas that we wish to examine in phase two and these have been outlined in previous sections. In particular the issues raised in reviewers comments that will be part of our further investigations in cases studies on Inner Source include:

	Comments	**Reasoning**
Evangelia Berdou (A)	The context within which the case studies are situated does not provide a well grounded view on how inner source principles relate to developments in the free/open source world and their influence on corporate practices. Although the dichotomy between open source principles and traditional software development, especially as presented by Eric Raymond, can be a useful starting point, it should be further elaborated to reflect the complex and often dialectic relationships that exist between corporate and community driven development as indicated by different studies.	Agree completely and have addressed this issue to a limit extent in the final deliverable text. The continued work in Phase 2 examines in more detail issues of this exact nature.
Evangelia Berdou (B)	I think several of the points that you make will come across clearer if you develop a typology of the stages and extent to which inner source principles are adopted in a corporate setting. For instance when and under what conditions is the development opened up to a greater community? How is this community defined? Are there, if any, any additional costs in inner sourcing (such as learning)? Some of these issues have been touched upon within the case studies, but I think that they need to be more formalized.	The questions raised here are extremely relevant and appreciated, we will address these questions to the maximum extent of the knowledge available to us at this time, however as with the previous point we will be looking at these issues in more detail in the case studies of Phase 2. In particular it is very difficult to formalise these issues without examining other cases of a similar nature.

Evangelia Berdou (G)	Section 7 'Inner source for OPAALS' provide a very limited discussion of the insights of the deliverable in relation to the OPAALS community. I think such discussion should take into account the different dimensions of the tasks undertaken by the partners as well as the project's distinct institutional framework.	While it was not our intention to focus specifically on Inner Source for OPAALS when we wrote 8.2 we are more than happy to examine this in more detail in the future. There is also the issue of releasing the pertinent information too soon an effecting OPAALS negatively, especially if acted on directly.

Reviewer : Evangelia Berdou, LSE

The deliverable focuses on the adoption of open source tools, principles and processes of collaboration in companies and provides some discussion on how these can be leveraged within the context of OPAALS. The deliverable is on the right track and makes some interesting points regarding the difficulties in transferring open source principles in a corporate environment. *It lacks, however, the depth and breadth of analysis that would allow us to critically position these developments within the context of the evolution of free/open source and the changes that it has initiated in corporate driven, globally distributed software development. In addition the discussion of the relevance of inner source principles for the OPAALS community is too limited to be of substantial value to other partners.*

Major Issues:

a) The context within which the case studies are situated does not provide a well grounded view on how inner source principles relate to developments in the free/open source world and their influence on corporate practices. Although the dichotomy between open source principles and traditional software development, especially as presented by Eric Raymond, can be a useful starting point, it should be further elaborated to reflect the complex and often dialectic relationships that exist between corporate and community driven development as indicated by different studies. Examples of these are:

Garcia, M.J. and E.W Steinmueller (2003) 'The Open Source Way of Working: a New Paradigm for the division of labour in Software development?' INK Open source working paper No.92, SPRU Science and Technology Policy Research, University of Sussex, available at: http://www.sussex.ac.uk/spru/1-6-1-2-1.html, last accessed 15/08/07. In this paper it is argued that open source exhibits many characteristics of traditional software development. The article is of additional interest in that it critically reviews many commonly held views on open source development, including Raymond's.

West, J. and C.S. O'Mahony (2004). 'Contrasting Community Building in Sponsored and Community Founded Open Source Projects', available at: http://opensource.mit.edu/papers/westomahony.pdf , last accessed 06/08/07. This is an interesting paper in that it compares development practices in community driven and corporately initiated open source projects.

Capiluppi, A. and M. Michlmayr (2007) 'From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects' in IFIP International Federation for Information Processing, Volume 234, Open Source Development. Adoption and Innovation, eds. J. Feller, Fitzgerald. B., Scacchi, W., Sillitti, A. Boston: Springer, pp. 31-44. In this article it is argued that not all open source projects reach the distributed development phase of the bazaar and that in several instances the open source development process remains centralized.

Jaaksi, Ari (2007) 'Experiences on Product Development on Open Source Software' in IFIP International Federation for Information Processing, Volume 234, Open

Source Development. Adoption and Innovation, eds. J. Feller, Fitzgerald. B., Scacchi, W., Sillitti, A. Boston: Springer, pp. 85-96. This article outlines Nokia's strategy of open source development. Jaaksi makes some interesting observations on the sources of 'resistance' within companies.

b) I think several of the points that you make will come across clearer if you develop a typology of the stages and extent to which inner source principles are adopted in a corporate setting. For instance when and under what conditions is the development opened up to a greater community? How is this community defined? Are there, if any, any additional costs in inner sourcing (such as learning)? Some of these issues have been touched upon within the case studies, but I think that they need to be more formalized.

c) Furthermore, the deliverable focuses only on relatively successful instances of inner source. It would perhaps be of equal value to examine failed instances of inner source adoption.

In addition:
d) No rationale is provided for the selection of the analyzed case studies.
e) The case study of Firefox is far too limited and should be taken out.
f) I am not entirely certain on how section 4 'Face to face Vs distance interaction' fits into the deliverable, since both of these types of interaction can take place in a corporate setting regardless of the type of development (traditional vs inner source) being adopted. This section appears to be more relevant to a discussion on the differences between collocated and distributed development rather than on the difference between open source and traditional software development principles. A better rationale of why this section adds to the deliverable needs to be provided.
g) Section 7 'Inner source for OPAALS' provide a very limited discussion of the insights of the deliverable in relation to the OPAALS community. I think such discussion should take into account the different dimensions of the tasks undertaken by the partners as well as the project's distinct institutional framework.
i) Section 5 was the most interesting one.

Other issues:

In some points in the document claims are being made that are not supported by relevant references.
j) For example in page 7 it is stated that: 'It is a widely held belief that deploying open source can lead to a sense of shared adventure, which is not a common scenario in the proprietary software arena'.
k) Another example of the same kind can be found in page 27: 'Table five below compares and contrasts traditional software development to inner source software development based on the information gathered from our work thus far'. What exactly is the work that is being mentioned here?

I think issues a), b), d), e), f), g), j), k) need to be addressed before the final submission to the Commission.

Reviewer : Luigi Telesca, CreateNet

The deliverable presents Inner Source as an understanding of Open Source and provides an interesting issue for OPAALS research.

Minor changes are suggested:

1. Homogeneity of characters, sizes and text alignment should be verified.

2. Grammar review is needed.

3. Provide more information on the origins and use of the term "Inner Source". The introduction gives some rapid info, but since Inner Source is the core of the document, it should be better explained (i.e. who is using it? others..).

4. The choice of the case studies should be better explained.

Reviewer : Maha Shaikh, LSE

This deliverable reads well and explores an area that has not been academically studied. The growing importance of inner source to the software industry and far beyond is easy to understand. Deliverable 8.2 makes concrete headway into guiding our understanding of inner source in relation to open source, and traditional commercial software development.

I have some suggestions for change (below) that will hopefully strengthen this deliverable:

Executive summary needs to include some mention of the case studies and a brief sentence summing up the main findings derived from those studies. The summary also forgets to make any mention of Goffman's or Clark's theoretical ideas that have been used in this work.

The Introduction Section includes a definition of inner source. This definition appears to be very broad. Having read through the entire document I understand that you want a better meaning of inner source to emerge from your case study material, however this is not apparent to the reader initially. Perhaps this point could be clarified in the document outline paragraph on page 8. It could add context to your material and provide us with your logic behind the current flow and structure of the deliverable.

Bell Labs case study – This has a table of 'differences between open and proprietary software. If this emerged from the Bell Labs study then I see why it has to be in this section (but then you need to add a reference in the caption to that effect) but if not then this table could be better placed in the Introduction Section.

IBM case study – the key results section is somewhat repetitious.

Firefox case study – remove this case and perhaps add the content or some of it to the introduction section of the deliverable. It doesn't in its current form qualify as a separate case study.

The case studies are very interesting and provide us with a view to the different strategies of inner source adoption and sustainability in commercial settings today. However, what would be equally interesting and probably illuminating about issues in inner source strategy adoption are some unsuccessful cases. D8.2 presents three (the fourth is not really a case study) cases of mostly successful fostering of inner source ideas in a commercial setting. Finding more cases of inner source use in companies is one way to strengthen the points you make but the same effect can be managed if you highlight the problems section you have for each case. You have indicative data but perhaps you could organize it in some easy to read and prominent fashion? One possible way could be to create a table showing the problems that commercial companies face with inner source while highlighting the lessons learnt. This could be added to Section 5 – Managing and organizing inner source.

Section 4 – the move between the various cases and Section 4 is a little bumpy. Each section needs to have a concluding paragraph which summarizes the main issues dealt with so far and then links it with the next section. This will help the transition process between sections and the reader will understand the reasoning behind the given structure.

Title of Section 4 – rather than call this face-to-face versus distance interaction could you contextualise the title to make more sense for inner source? I mean something along the lines of 'local to global' or 'locally to globally distributed development' so that we understand why we are now reading about Goffman and Clark.

Section 4 – the theoretical ideas of Goffman and Clark need to be drawn together in a better way. We have concepts from both given to us but they are not knit together to create a theoretical framework. All we need really is a paragraph to explain why, for the current study, we need both Goffman and Clark and why one alone will not suffice.

Table 2 is very good and informative. I would highlight that the paragraphs that follow are an analysis of inner source development through the theoretical lens of Goffman and Clark. Basically a short subheading to that effect will do the trick.
Section 5 – again we need linking paragraphs from one section to another.

Section 6 – these principles have emerged from your study but this is not apparent to the reader immediately. A linking paragraph will help here as well but the position of this section in the document needs to be explained in the Introduction section. It could be argued that this section needs to be in the Introduction section of this deliverable but if you have a reason for placing it nearer the end then please explain it in the section outline on page 8 of this deliverable.

Section 6 is a very good section and summarizes many of the key arguments you make in this work,

Section 7 – I feel there is more you can say about the relevance of this deliverable for OPAALS. It might be useful to strengthen this section through elaborating some of the points you raise.

Minor issues:
There are some typos and grammatical errors in the document which need to be ironed out with a good edit.