

Indie Fever

The genesis, culture and economy of a community of independent software developers on the Macintosh OS X platform

Researched and written by
Michiel van Meeteren

Bachelor thesis human geography, University of Amsterdam
Supervisor: Prof. Dr. R.C. Kloosterman
July, 14, 2008

<http://indie-research.blogspot.com/>
michielvanmeeteren@gmail.com

© Copyright 2008, Michiel van Meeteren



This work was supported by a research grant from Sofa BV — madebysofa.com

Acknowledgments

“And, having opened his big register, the geographer sharpened his pencil. The recitals of explorers are put down first in pencil. One waits until the explorer has furnished proofs, before putting them down in ink.”

-Antoine de Saint-Exupéry

No proof can ever be given in the social sciences while observing the world from an ivory tower and therefore there are a number of people who have contributed in making this work possible that I would like to thank over here.

First and foremost, I would like to thank Dirk Stoop who showed me his world, accompanied me in the travels and contributed indispensable knowledge to this thesis. Furthermore I want to thank his colleagues at Sofa: Koen, Hugo, Jasper, Jorn, and Oskar for their support. Additionally, Zillah Kaptein and Loes van der Laan contributed substantially with their reviewing. I am very grateful for the dedicated collaboration of the respondents, in that sense Indies are the best research subjects a researcher could wish for. I would like to thank Mike Lee and Wil Shipley for their extraordinary hospitality while doing fieldwork in Seattle. Lastly, I would like to thank the University of Amsterdam for encouraging and supporting me to go out and satisfy my curiosity.

Introduction	3
Chapter 1: An epistemological framework to understand the ‘irrational’; Ontology and research methodology	6
1.0 Introduction	6
1.1 Epistemological framework	7
1.2 Methodological considerations	13
Chapter 2: Genesis of a figuration; The path dependent process to the OS X independent developer community	16
2.0 Introduction	16
2.1 The legacy of the classic Apples: the Apple II and the Macintosh	16
2.2 The legacy of NeXT computer	20
2.3 Merging the technologies	22
2.4 The .com bust and the maturation of online payment systems	25
2.5 The starting point: the initial figuration of OS X developers	26
Chapter 3: The sensibilities of beautiful software and economic independence; Cultural traits	27
3.0 Introduction	27
3.1 The sensibilities of ‘good software’	28
3.2 The sensibilities of being independent	31
3.3 The sensibilities of good business practice	33
3.4 Lifestyle aspects and spatial preferences	36
3.5 Understanding cultural capital in the Indie Mac world	38
Chapter 4: Communities and networks of practice; The hardware of the Indie figuration	39
4.0 Introduction	39
4.1 Applying cluster theory to Indie developers	40
4.2 Relations between Indie developers	40
4.3 Embeddedness of the interactions	44

4.4 The role of customers	52
4.5 The economics of Indie software; marketing	54
Chapter 5: Placing the Mothership; The role of Apple Inc.	59
5.0 Introduction	59
5.1 'Who' is Apple according to Indies?	60
5.2 Relations between Apple Inc. and Indie developers	61
5.3 Functions of the relations between developers and Apple	64
5.4 Evaluating the balance of power between Indies and Apple Inc.	69
5.5 Inserting Apple into the figuration by an application of Michael Porter's 'diamond'	71
Conclusions	75
Epilogue	77
Appendix A: Defining culture and identity	78
Appendix B: Questionnaire	80
Appendix C: Composition of the qualitative interviewee selection	85
Appendix D: Apple/NeXT/OS X history Timeline	88
Appendix E: OS X market share developments	90
References	93
Online references	99

Introduction

Every industry has its own legends; stories about entrepreneurs, like John D. Rockefeller or Henry Ford, that did not only shape their industry but also exerted influence on our societal and cultural system. The computer industry is no exception to this; the garages of Silicon Valley have not only shaped a multibillion dollar industry but have also created a mythical archetype of what a small startup company can achieve in our contemporary world. Apple, and thereby its founders Steve Jobs and Steve Wozniak, are one of the strongest symbols of this. Although the garage doors of Silicon Valley have long slammed shut and Silicon Valley has turned into a corporate powerhouse, the myth of the computing startup still shapes people's ideal to achieve big goals with small companies.

This thesis is the result of the first phase of a longitudinal research project to understand the social and economical world of small independent software companies that develop software for Apple's Macintosh platform. The aim of this first phase was to simultaneously establish the qualitative contours of this world and develop an epistemological framework that is able to explain it.

The market for software on the Apple Macintosh platform can roughly be divided into two segments. First there is the 'blockbuster' segment. These are the software applications that have a high worldwide turnover, on which economies of scale apply. Three big companies mostly serve this market: Apple Inc. itself, Microsoft and Adobe. Additionally, an enormous amount of small niche markets exist that are made up of more specialized applications of any kind one can imagine: 'getting things done' software, library solutions, cashier systems, budget balancing software, Mp3 or video converters, FTP clients and so on. Ever more of these niches have become economically viable due to the arrival of digital distribution and e-commerce, which brings down the cost of serial production and distribution to near zero. This research project is about the entrepreneurs that serve those markets on the Macintosh platform, focusing on a certain subgroup of entrepreneurs in particular: those that have informally organized themselves under the identity maker of 'Independent software developers'; in their own words 'Indies'.

These often one-man small companies have formed a relatively close-knit community, despite being scattered over the USA, Europe and Australia. They help each other out on common problems through online communication and interact like they would be coworkers. Moreover, throughout the years individuals within the community have created online resources for the

benefit of the collective. Despite these collaborations they still regard each other as competitors although all sorts of unspoken rules apply to the kind of competition that is allowed within the community. In this sense they share a culture that resembles the close-knit craft communities of Northern Italy or the diamond merchants of Antwerp.

The Indies' culture of competition is strongly interwoven with the examples that have been put forward by Apple Inc. throughout its existence and the history of the community is strongly interwoven with the history of the fabled computer company itself. In the second half of the 90's Apple Inc. was resurrected from its corporate grave by the return of its founder: Steve Jobs. In 2000/2001 this resulted in OS X, an operating system that left behind years of inertia and partly reshuffled the composition of developers that were active on the platform. This is why the release of OS X will be taken as a starting point for the community under study.

Such an online community poses some challenges for the social scientist. It requires an understanding of an informal social entity with opaque boundaries. Furthermore it raises all sorts of questions about how the community is reproduced, sustained and how the cultural and economical facets of it intertwine. It also cannot be understood without including Apple Inc. in the picture since it is a potential competitor and technical host to the community in one. Therefore, the choice has been made to frame the research into a network approach, primarily focusing on the interdependencies amongst companies and between companies and Apple Inc., which brings us to the research question subdivided in three parts.

- *How do independent software developers for the Macintosh OS X platform interact with each other and with Apple Inc.?*
- *What kind of functions do the social relations have for the social and economical figuration of both Apple Inc. and the independent developer community?*
- *How are these interdependencies socially and spatially embedded?*

In chapter one the research question will be discussed in relation to the contemporary economic geographic literature identifying the problems at hand in greater detail. An epistemological framework will be constructed to tackle these problems and the research methodology including its advantages and shortcomings will be discussed. Chapter two will explain the genesis of the OS X developer community in detail and will reveal its cultural

antecedents. Chapters three and four will respectively discuss the Indie culture and its social and economical foundations. Chapter five will include Apple Inc. from a broader point of view, allowing to make statements on a more structural level. The research question will be revisited and answered in the conclusion thereafter.

Chapter 1: An epistemological framework to understand the ‘irrational’; Ontology and research methodology

1.0 Introduction

The purpose of this first chapter is to explain the theoretical framework that has been chosen for the research project. This framework and the resulting research question just mentioned, touches upon a few ongoing debates in economic and economic geographic literature. In general the ongoing debates in economic geography have been about assessing globalization. To what extent do the societal changes that accompany the increasing flows of goods and information around the globe influence the geography of economic practices? First it has been noticed that the cultural and semiotic aspects of goods and services have increased (Scott, 2000). As a result of this enterprises aiming at economic success have to take into account the wishes of the creators of the cultural content, the so-called creative class (Florida, 2002). This creative class has one important aspect that needs to be understood to judge their economic performance. They choose their goals on a variety of motivations, which often contradict with profit maximizing rationality of (neo)classical economics (idem, pp.102-115).

Another important aspect of globalization is that economies have to distinguish themselves by providing innovative goods that have a high added value. This influences the economic geography because it is argued that the spatial proximate intensive relations that firms have among each other positively enhance innovativeness and competitiveness: the cluster hypothesis (Porter, 2000). Especially for the information technology (Saxenian, 1994) and the cultural industries (Scott, 2000) these inter-firm linkages are deemed very important.

Indie developers are both part of the IT and the cultural industries; they create new computer software by their own design. For that reason researching the linkages between Indies is a logical starting point to connect to those theoretical repositories. The ‘function’ part of the research question is necessary to see these relations in a wider perspective. If we want to judge the ‘functionality’ of the relationships in terms of innovativeness or competitiveness we have to abstract from the subjective actor level to a more generalized structure level. Lastly, the embeddedness part of the research question is informed by the fact that the functioning of ties between actors for

innovation, etc., seems to be strongly related to the quality of those ties (Sturgeon, 2003, pp. 217-220). Therefore understanding the embeddedness of the relations is crucial for assessing its function. Paragraph 1.1 will establish an epistemological framework that is able to take into account these aspects while paragraph 1.2 is devoted to the methodological choices that were made doing the research and will reveal both its advantages and shortcomings.

1.1 Epistemological framework

The social sciences have a tradition of specialization. Economists study economic issues, cultural issues are studied by anthropologists, social issues are studied by sociologists, etc. The disadvantages of this approach are that the respective theories produced by these disciplines have a difficulty ‘talking’ to each other. They have different axioms about how the world works, different ontologies, and often have research methods that are not interchangeable (Chabal & Daloz, 2006, pp. 310-327). Therefore, if we want to combine the insights of different disciplines, we need an overarching epistemology that is able to bridge these differences. Three of these fundamental social scientific issues stand out in this thesis: the actor-structure problem, the interaction between culture and the economy and the problem of embeddedness.

The problem of abstracting from the subjective, related to the actor-structure problem, is the first one to tackle. The empirical part of the research consists of interviews and field literature that express the position and ideas of the actors involved. This implies that a humanistic ‘lifeworld’ or ‘rich actor’ perspective (Ley, 1980) lies at the heart of the research that is necessary to understand the motivational and rational discourses of the Mac Indie community. At the same time, research that would only address this perspective would have a tendency to overlook the more structural economic and power relations that impinge on- and help sustain such a community (Perrow, 1986, p.258-278). To achieve this double focus, the subjective viewpoints have to be abstracted to a more intersubjective ‘functional’ framework. A ‘functional’ framework takes into account the stakes that actors have in sustaining a social system regardless of the motivation of the actors involved (Elias, 1971, p.85). Hereby we can judge for example if, and in what sense interactions between Indies enhance their competitiveness. Although this does not imply that functions automatically determine the system –as has been argued by advocates of the sociological functionalist school (Perrow, 1986, p.278). Ties between Indies do not have to be created out of such a utilitarian viewpoint nor sustained for that reason. The problem that arises is how to value both positions simultaneously without getting lost in a perspective where one viewpoint determines the other.

The second problem is that of human motivation. One important aspect of the definition of Indie Mac developers (chapter 1.2) is that they are commercial companies; they have the intention to make a living out of their business. Still, their behavior cannot be explained solely from a classic ‘economic man’ perspective. All sorts of cultural sensibilities play a role in the decisions that are made. Why ‘independent’ and why ‘Mac’? To name a few obvious ones. Explaining their motivation requires a combination of cultural and economical considerations. The scientific division of labor has a long tradition in seeing the sphere of the economic and the cultural as separate theoretical fields while in practice they reinforce each other in a variety of ways (Krippner, 2001, pp. 788-802). Therefore, if we want to understand rationality in the Indie Mac world we have to understand their ‘rational’ discourse without resorting to either a cultural or an economic logic.

Third, and strongly related to the other two, is the problem of embeddedness. The ongoing debate about embeddedness in the social sciences¹ relates in its most fundamental form to the contradiction between opportunistic behavior of individuals on the one hand and social order on the other. The rationality of (neo) classical economics implies that it is natural that people commit opportunistic behavior when the perceived rewards are bigger than the perceived costs. While more sociological approaches suggest that human action is triggered by their social context that could include ‘irrational’ behavior if viewed from an economic perspective (Granovetter, 1985, pp. 481-487). Ontologically we know that the two are only separable in a theoretical sense and that in social reality all economic relations, including spot markets, are embedded in social relations in one form or another (Krippner, 2001, pp. 778-782). Despite the theoretical refinements on both sides of the debate it has proven to be very difficult not to end up in either an under- or oversocialized definition of embeddedness. Marc Granovetter (1985, pp. 504-508) tried to resolve the problem by focusing on the interactions between actors, utilizing a social network perspective. Although this was an important improvement in the explanation, mainly because it resolved the actor-structure dichotomy, it did not address the separation between cultural and economic determinants (Krippner, 2001, p.798-802). Partly as a result of that the concept grew increasingly fuzzy as typologies of different, not mutually exclusive, ‘kinds of embeddedness’ were defined (Hess, 2003). The ‘problem of embeddedness’ is especially important in this research because independent Mac developers trust each other while technically, they are

¹ In which I include economics.

competitors. Chances for opportunistic behavior are rampant but seem to happen very rarely within the community.

Two considerations are especially important for applying an embeddedness framework to this research question. The first is related to the geographical implications of embeddedness. To trust each other is related to knowing each other and therefore a logical connection between meeting somebody in person and the development of trust relations exists (Hess, 2003, p.17). A similar argument has been made about tacit knowledge transfers that require demonstration and imitation (Malmberg & Maskell, 2001, p.13). These connections led to a spatial transformation of the embeddedness concept in economic geography where spatial proximity is considered a prerequisite for the development of trust relations and tacit knowledge transfers (Mackinnon et al., 2002, pp. 301-302). In certain economic regions these trust relations are part of, and form localized cultural milieux² that give unique regional advantages that are difficult to reproduce elsewhere (Scott, 2000, pp. 16-29). This is however a highly contested position that is related to the wider 'glocalisation' debate. On the one hand authors are arguing that the technological innovations of the globalization era enables remote working and collaboration, increasingly making face-to-face interaction redundant, and therefore 'flattening' the world (Friedman, 2006). On the other hand there are authors who argue that the same innovations invoke a premium for localization, or spatial clustering, for some assets cannot be sustained over long distance (Sturgeon, 2003). These localized assets are usually subdivided in two different kinds. First, there are urbanization economies, like a specialized labor market, that exist independent of linkages between firms. Second, there are assets that are derived from linkages between firms, social network externalities or localization economies, which rely on social embeddedness properties (Malmberg & Maskell, 2001, p.3). The latter also form the core of Porter's cluster hypothesis (Simmie, 2004, p.1098). Although empirically it is often the case that social embeddedness coincides with spatial properties, it should be remembered that fundamentally, embeddedness is not a spatial but a social relationship (Hess, 2003, pp. 2-5). It is theoretically not impossible to create trust relations over distance with no or limited face-to-face contacts; the amount of these contacts supposedly increase now because of the availability of ever more advanced types of social software. Not the region but the culture, usually embodied in interactions, is the determinant of trust relations (idem, pp.19-21). Culture has a tendency to develop spatial characteristics but a territorial demarcation is not by definition part of a

² Appendix A includes definitions of the culture and identity concepts, based on the conception of anthropologist Clifford Geertz.

cultural system (Mitchell, 2000, pp. 60-65). Territorialized culture is a logical, but not deterministic outcome of social processes and spatial proximity is a logical but not necessary prerequisite for trust formation. Paragraph 4.3 will elaborate on these propositions regarding the field data of this research.

The second consideration in the use of network-based embeddedness theories is the question what exactly constitutes a relation between actors. This can be conceptualized as a mutually acknowledged reciprocal tie between actors as Granovetter (1985, p.496-498) seems to imply. If this is taken to be true, a network perspective is insufficient to explain all forms of embeddedness because it does not take into account that there can be 'one way' traffic of meanings. Human beings do not only learn social rules through direct interactions between humans but they also eventually acquire culture through interaction with a 'meaningful environment', for example: an observed role model embodied in a text, a piece of music, a landscape or a piece of software (Crang, 1998, p.44). This learning implies that cultural meanings in wider society can also influence human motivation (Derné, 1994). Describing network relations as 'interdependencies', which can exist independently of mutually knowing each other, broadens the scope of the network concept to include both cultural and indirect power relations and solves part of the problem (Elias 1971, pp. 11-18). Caution should be taken though; never should it be forgotten that by creating these network models we try to summarize a complex social reality using a network metaphor. The quality of the metaphor depends on its ability to grasp those elements of society, linguistically, that we deem important in explaining the question at hand; we should always be wary of reifying this theoretical construct as something 'that is'.

To summarize this story: a good epistemological framework would need to have the ability to grasp both the actor-structure and culture-economy dichotomies through an embeddedness perspective. Figuration sociology as proposed by Norbert Elias (1971) has the potential to explain the problems at hand without falling for the deterministic trap. Furthermore it is a network-based perspective, which enables this theory to be compatible with other theoretical literature that I deem important for this thesis. Especially evolutionary and path dependent economics, economic sociology in the vein of Granovetter, the cultural theories of Pierre Bourdieu, the work on cultural economic systems by Allen Scott and work of Michael Porter on competition and cluster economics can fit quite easily in the basic propositions set out by Elias.

To Elias, society consists of figurations of interdependent actors. Although the potential scope of a figuration is a worldwide one, the concept can be used to delimit those actors that are necessary to explain for the research question chosen (Elias 1971, pp. 144-146) For example, it can include or exclude the customers of an economic figuration depending on whether they exert enough influence to help explain the situation at hand. Although taken from a different grand narrative, the delimitation of the figuration is very compatible to what Bourdieu calls a 'field'. For example Bourdieu's theory of the cultural field –whose concepts play an important role in this thesis– takes into consideration all the producers, intermediary actors, their outputs and the power positions of these actors that make cultural products what they are (Johnson, 1993, p.9).

In the work of Elias the interdependencies, or linkages, between actors are a mix of economic, affective, cognitive, and formal hierarchical power differentials (Elias, 1971, pp. 149-162), where power is broadly and neutrally defined as the ability to make other actors do something (idem, p.80-87) and consequently, power should be understood as a relational property (idem, p. 81). An actor can be endowed with, for example, economic, affective and cognitive power advantages versus another actor. These power advantages are very comparable to the monetary metaphors Pierre Bourdieu applies in his work; e.g. cultural, linguistic, educational and symbolic capital. He uses these metaphors that resemble prestige and renown, a certain acknowledged competence or other dispositions to explain how other personal assets in societies than just economic differences can empower people (Bourdieu, 1977, pp. 171-183). The distinctions between the different capitals Bourdieu employs are often instrumental and field-specific but they are all based on his same basic concept of symbolic power: power that is not reducible to economic capital (Johnson, 1993, p.7). In this thesis the concepts of cultural power and cultural capital will be used to mean the overarching concept of power that is not reducible to economic differences. Note that Bourdieu's power principle is principally synonymous to that of Elias; it is relational in the sense that symbolic capital is only valid to people who share the same symbols³.

This Bourdieu-enriched Elias-style epistemological framework has the capacity to solve the problems identified. First it explains that people can have

³ Bourdieu's theory has incompatible differences with Elias's theory when it comes to the recognition of social structure. Bourdieu divides society in relatively autonomous 'fields' with their own dynamic that form an intermediary power structure in society. The dialectic oppositions in his theories make it quite 'France specific'. For example the opposition between l'art pour l'art and commercial art (Bourdieu 1993, p.115) would not be valid in the USA.

affective powers related to people they do not know personally: think of pop stars or charismatic politicians. Furthermore it explains relations between different kinds of power: the essence of the work of Bourdieu (1984) is the struggle for formal political power through capital conversions. It also explains different motivations: the theory recognizes for example that peer recognition can be a goal in itself to gain affective or cognitive power in relation to others; the way to do so is conveyed by culture. The actor-structure dilemma is solved by Elias with the explanation that an actor never is fully aware of whom he or she has an interdependent relationship with. For that reason the outcome of social action can never be fully predicted by the actor beforehand, as we can assume that even the most rational of persons will only take in account what he or she knows. Paradoxically, processes of social change, for example revolutions or technological innovations seem to have its own inherent structural logic afterwards. This logic emerges because the interdependencies that are relevant became apparent through the process of social change and even only after intensive social scientific and historic enquiry (Elias 1971, pp.162-171). The same logic can be applied to the economic notion of path dependence; at every stage of economic transformation the power differentials between actors change, opening new opportunities for divergent economic paths and blocking others that afterwards can be recognized as a 'path dependent process' (David, 2000). The advantage of figuration sociology is that it prevents the reification of these sorts of social structures; it can always be reduced to actions of, often unaware, human agents.

It should be noted that the questions raised by the empirical material require explanations that are derived from different social scientific disciplines. Although they all fit into the epistemological framework that was set out before, this does not mean that the authors acknowledge such a framework. For example the Indie figuration has no formal power structure but it could be argued that informal power plays an even bigger role in such a constellation. On several occasions theories will be applied that assume such a power structure to be present in the form of a firm, in particular as evolutionary economics and communities of practice are discussed. A company is a figuration just like the developer community under study is and I think it is theoretically justified to eclectically apply some of the concepts of these theories to this thesis because their validity is the consequence of the figuration aspect and not the formal power aspect of firms.

1.2 Methodological considerations

This paragraph will explain how the research subject is defined, the method chosen to study the figuration and what the advantages and disadvantages of this method are. Since the validity of this thesis rests on the fact that something as a Mac Indie developer figuration –a distinctive group– exists, I have decided to define an ‘Independent developer’ in the same way as members of the Indie figuration do. At the C4 conference, a conference organized by Indie developers for Indie developers in Chicago in 2008, developer Jonathan ‘Wolf’ Rentzsch [10-08-2007]⁴ defined ‘their’ identity:

‘Indies’ are independent software developers. They cannot be characterized as just any kind of third party software developer; two additional criteria have to be met. First, he states that a company needs to be ‘non-large’. What Rentzsch means with this is that an Indie company is not ‘corporate’. For example Adobe or Microsoft would not be Indie, but a company with a few dozen employees could be. What exactly being ‘not-corporate’ means is described in paragraphs 3.2 and 3.3. Secondly, Rentzsch suggests that the company needs to be commercial. They need to sell software of their own to a broad audience, and will not be a captive supplier to another company; selling the software should be done to make money in the marketplace.

From a social scientific perspective this may seem a fuzzy definition but it has proved to be thoroughly recognized during the interviews and therefore useful during the qualitative research process. Although officially part of Rentzsch’s definition, games developers have been left out of this research program for two reasons. Firstly, the development process of computer games usually requires a much greater input in artwork that causes a different, usually bigger company structure. Secondly, they serve a completely different market and in an early stage of the research it was decided to omit them because it would complicate the research too much.

As has been stated in the introduction, this thesis explains the results of the first phase of a longitudinal study of Independent software developers on the Macintosh platform. This first phase consists of qualitative research and a theoretical explanation of the results found. The reason to choose to start with a qualitative phase is because a thorough understanding of the social world of Indie developers is necessary before any meaningful quantitative questionnaire can be developed. Furthermore, there are no readily available

⁴ In the thesis literature sources will be distinguished by Internet sources by the kind of brackets used. () means it is a literature source to be found in the literature reference list, [] means it is an internet source to be found in the separate internet references list.

databases of independent developers. This means that criteria to make a sampling frame, necessary for a large survey, have to be established before any activities can be undertaken. The empirical inquiry of this qualitative phase was achieved in three steps. First a set of informal preliminary interviews were conducted in Amsterdam between June and December 2007, with Dirk Stoop of the Amsterdam based Indie company Sofa BV. These interviews produced hypotheses about the workings of the Indie figuration that were distilled in a questionnaire, which in its general form can be found in appendix B. This questionnaire formed the basis of 15 semi-structured interviews that were conducted on the West coast of the USA in January-February 2008. The questions in the interviews were customized with personal aspects that were derived from Internet activities, like blogs and interviews of the respondents. The interviewees were initially selected to provide an as broad as possible spectrum of perspectives on the Indie development world. This was based on information provided by Dirk Stoop's expertise and Internet searches. The questionnaire was refined between each of the interviews. Subsequent interviewees were selected by snowball 'sampling' methods (Bryman, 2001, pp.98-99) to fill in gaps in the different perspectives. In this way an attempt was made to create interplay between the research findings and theorizing on the figuration as is suggested by grounded theory (Strauss & Corbin, 1998, pp. 15-35). This stage of the research generated more than 45 hours of recorded interviews. Appendix C gives an overview of the different characteristics on which interviewees were selected although they are anonymized for privacy reasons. The third stage of the empirical research involved the analysis of the interviews and the collection of blog posts, video and radio fragments that provide further explanation and validation of the patterns found in the interviews; over 15 hours of audiovisuals were analyzed and over 300 blog posts were collected, some of which are quoted in this thesis. Altogether this results in a viable thick description (Geertz, 1973, pp. 1-30) of the Indie figuration under study that is described in this thesis.

This particular qualitative method has the advantage that the power structure and cultural sensibilities of the figuration can be understood in a very detailed manner. The downside is that at this point it is still uncertain how applicable the research findings are to all Mac developers that fall under the Indie designation. Other companies that do not have such a strong personalized presence on the web may be successful and therefore other cultural preferences could have remained unnoticed. Furthermore the lack of numbers makes it difficult to precisely define the advantages that are derived from cooperating. Subsequent quantitative research will have answer those

questions. Geographically the Indie community that was studied during the research consisted of an unidentified amount of connected companies that were spread out over the US, Europe and Australia. Some connections with Japan were found but they were incidental and usually related to private backgrounds. Therefore it is left open at this point whether the results of the research are valid outside the three regions mentioned. The exact structure of the community and the overlaps in networks will have to be revealed with a network analysis in a later phase of the research. The boundaries of the community are unidentified at this moment because they are fuzzy in their very definition. People do not sign up for a club, they code a piece of software, start to sell it and at a certain time get noticed and connected by the community. From that moment on they create web content and that is how community members were recognized in the research process.

A final shortcoming in the research is that Apple Inc. was not officially willing to participate in the research. Therefore relations between Indies and Apple Inc. are not validated from the perspective of Apple Inc.. Still, the triangulation of the perspectives of the different developers gives a coherent impression about the role of Apple Inc.. It is however impossible to make any claims on the intentions of Apple when it comes to their relations with Indies. The advantages the Indies provide Apple Inc. with are revealed through a functional analysis but it is unclear whether Apple Inc. is aware of this and whether the company creates them on purpose.

Chapter 2: Genesis of a figuration; The path dependent process to the OS X independent developer community⁵

2.0 Introduction

Apple Inc. announced the Mac OS X operating system in the year 2000.⁶ Although the name suggests continuity with the ‘classic’ Macintosh operating systems, in fact under the hood lies completely different technology. The technologies underpinning OS X were largely developed by NeXT computer in the 1980’s and acquired by Apple when they bought NeXT computer in 1996. Despite technological bridges being built to enable software companies developing on the old Macintosh operating system to port their products, developing for OS X meant convincing developers to swap technologies. It reset the technological clock; all developers had to invest in new technology. This drastically shuffled the composition of the figuration of third party developers for the Mac. This chapter is devoted to providing an explanation of how the OS X Indie developer community came into being, what the cultural legacy is they identify with, how this coincides with the developments in the technology they work with, and how the wider social conditions were that contributed to the genesis of the developer community. In short, it tries to identify the path dependent trajectory that is responsible for the genesis of the current Indie developer community.

2.1 The legacy of the classic Apples: the Apple II and the Macintosh

The early history of the personal computer is strongly intertwined with that of Apple Inc. Apple itself embodies the Silicon Valley myth (Saxenian, 1994, p. 20): a company founded in a garage by two young men who invented a computer that sold millions, just because the world was ready for it and the big computer corporations were sleeping (Linzmayr, 2004, pp.4-10). In those days it was not unusual that a computer user would try to program what he needed himself. The Apple II was a vehicle to do so; before those days mainframe and minicomputers were not something you would own as an individual. The Apple II strongly symbolized the idea that a computer could empower individuals, encompassing the ideals of the homebrew computer club and the liberal counterculture that still lingered in Silicon Valley after the

⁵ This chapter relies heavily on field data in certain aspects. Despite efforts it was difficult to back every historical statement up with a reliable reference.

⁶ Appendix D contains a timeline where the several events are put in chronological order.

1960s (Saxenian, 1994, p.34). Apple has always cherished this image by using commercial slogans like 'the computer for the rest of us' and 'think different' (Cruikshank, 2006, p.116). It is the idea that the computer was a new outlet for expression, like art, or in the words of Apple founder and current CEO Steve Jobs in 1998 (quoted by Cruikshank, 2006, p.24):

“The idea that a scientist can be an artist is very much the Apple philosophy: in the 70's and 80's the best people in computers would normally have been poets and writers and musicians.They went into computers because it was so compelling. It was fresh and new. It was a new medium of expression for their creative talents. The feelings and the passion that people put into it were completely indistinguishable from a poet or a painter.”

The Apple II had a relatively huge developer community in its day mainly consisting of small companies. Even though the Apple II reached a huge audience compared to mainframe or minicomputers it still was a relatively fringe phenomenon in society; economies of scale did not matter that much in those early days. When the personal computer became a mainstream market and the IBM-PC took off, this changed.

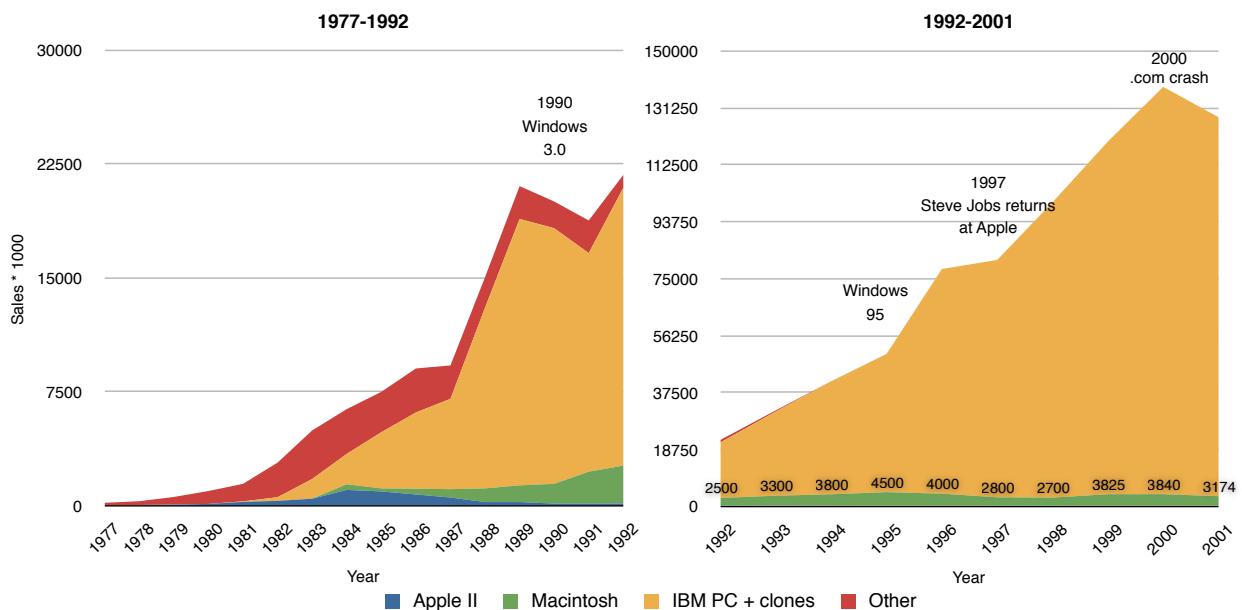
IBM, the biggest company in pre-microcomputer technology, hit the microcomputer market in 1981 when it introduced the IBM-PC. IBM's PC was more expensive and wasn't in any degree superior to Apple's technology – many regarded it as inferior– but within two years it took over Apple's lead in the market share of microcomputers thanks to its strong brand which stated stability, service and reliability (Linzmayr, 2004, p.68). Thus, for the first time, Apple got its David versus Goliath image: the hippy-ish young homebrew computer club style company versus the corporate giant of an earlier era. With a competitor gaining market share in the consumer and business markets Apple had to come up with a successor to its cash cow, the Apple II. Apple developed an expensive innovative computer, the Lisa, to compete in the business market. The Lisa never sold well but it introduced two technological novelties in the mainstream computer market: the mouse and the desktop interface, both developed at Xerox PARC in Palo Alto and 'leaked' to Apple's R&D department (idem, pp.73-78). At the same time Apple was busy researching its new generation consumer computers, the Macintosh, in which the same technologies were included. Moreover, the Macintosh became a computer with a specific design philosophy: it had to be a portable all-in-one computer that was easy to use, had to have an intuitive interface and should make computers accessible for everyone (idem, pp.85-98).

Although the Macintosh ended up being much more expensive than initially planned, it was a revolutionary consumer product in its day. It was the first

consumer-oriented computer that used a graphical-based interface and a mouse (idem, pp.97-98). The computer was introduced with a commercial based on George Orwell's 1984 that put IBM down as big brother and the Macintosh as the computer that would break the spell, reinforcing Apple's underdog image (idem, pp.109-114). Although the Mac sold well it never restored the market lead of Apple Inc. People were not willing to pay the premium for the innovations and the graphical interface, so the computer eventually only became huge in the graphical industry and advertising business. Moreover, most Apple innovations were slowly adopted by Microsoft who became the big corporate winner of the operating systems battle of the 1980s and 1990s and eventually replaced IBM as the arch nemesis of Apple lovers who regarded Microsoft's Windows as a rip-off of 'their' platform (Cruikshank, 2006, pp.134-138).

Apple Inc. has always capitalized on this image of being 'the underdog' that innovated and emancipated computers but was endangered by foul corporate tricks. They established an anti-big corporate image preferring quality over quantity and having beautiful design in high esteem (Cruikshank, 2006, pp. 129-140). Although one can argue about the factuality of these claims it is those virtues that make the Apple brand an identity marker for its users. Apple users are often described by outsiders as 'religious followers' that show a certain devotion to their computer brand. Apple has actively used this devoted following strongly organized in user groups to, for example, keep Apple products available in retail stores during the darker days of its existence (idem, pp. 89-100).

Figure 1, Global Hardware shake-out 1977-2001



Sources: <http://jeremyreimer.com/totalshare0.gif>, Linzmayer (2004)

Despite having this strong brand with its hard-core users Apple Inc. got into trouble in the beginning of the 1990s. Apple's market share was falling steeply (figure 1) and Microsoft Windows was becoming the industry standard operating system in all markets except the graphical industry. Moreover, Apple's signature operating system was no longer unique: most features were adopted in Microsoft Windows (Cruikshank, 2006, p.44). Apple was hampered by technological inertia in the design of the original Macintosh and kept failing to apply radical innovations to its operating system partially because of technological choices made a decade earlier (idem, p.45). The following quote by the founders of Indie company Panic [23-03-2004] illustrates the problems of developing for an operating system that never anticipated the complexity of a personal computer ten years after its design.

Duncan <Davidson, Interviewer> : So what was it like developing for the classic Mac?

Cabel <Sasser, Panic founder>: Oh, it was horrible. When you had a problem, your Mac would freeze up. Every time your application crashed, you'd get MacsBug and that was just a pain. But the funny thing is that even though it sucked, we didn't really know how much better it could be until we got Mac OS X. At the time we just accepted that this was the way things were.

Steven <Frank, Panic founder>: Mac OS X has dramatically shifted the way our development works. Things are so much quicker and faster now.

Although part of the hardware problem was solved with the licensing of IBM technology in 1994, Apple's R&D failed to produce a stable successor to the then 10 year old Macintosh operating system. The most significant try was the Copland project of which parts were used to stretch the life of the classic operating system [Kernelthread/6]. With Windows closing in for the kill, failing R&D and huge losses Apple turned their hopes to external technology that was eventually acquired with NeXT computer (Deutschman, 2000, p. 235).

Meanwhile the software development world had forked with the rise of the mass marketed computer in the 1980s. When personal computers were a fringe phenomenon software development was distributed as code in magazines, or on disks or cassettes in specialized outlets that carried small volumes. As the personal computer business took off economies of scale became important. To serve a mass market one needs mass marketing, logistics and retail that presupposes large organizational entities. This implied that software companies serving the mass market evolved from one or a few people shops to large organizations. At the same time, the 'shareware' concept rose: people would write software and distribute it over bulletin boards or through fringe magazines and send software and, often voluntarily, checks for payment by mail. Shareware could be completely based on the 'honor' system,

receiving the full version with voluntary payment or the software's functionality would be limited in certain ways before payment [Fleishman 03-07-2006]. Although shareware communities –a sort of predecessor of today's open source software– were present it remained a niche phenomenon in which it was difficult to make a living. The barrier of entry to the main market was high and few of these 'hobbyists' could make it their main occupation.

2.2 The legacy of NeXT computer

When Apple founder Steve Jobs was ousted from Apple in 1985 he founded NeXT computer. Set free from the limitations he had felt from his superiors at Apple he set out to build his vision of the 'perfect computer' aimed at the corporate and education markets. Like the Macintosh, the NeXT computers were based on the philosophy that a computer manufacturer should both develop software and hardware in-house. And both on the hardware and software side the company eventually produced computers that were 10 years ahead of its time (Cruikshank, 2006, p.52).

Especially revolutionary and relevant for the present was the NEXTSTEP operating system. It had a number of features that made programming for the platform a much easier and nicer experience than before. Without going too much into technical details, the NEXTSTEP operating system emphasized higher level object oriented programming and adopted the so-called Model/View/Controller paradigm, initially developed at Xerox PARC as part of the Smalltalk user interface and programming environment [Burbeck 1992, ADC 2007]. This means that a lot of common tasks a computer is asked to do are preprogrammed in application frameworks, reusable classes. These object templates make programming a lot easier because it requires fewer coding input on behalf of the programmer. Furthermore, the NEXTSTEP operating system relied on the objective-C programming language that made programming remarkably different from both the Windows and the classic Macintosh platforms that increasingly relied on C++. Objective-C and C++ are both object oriented programming languages based on the 'arch' language C. Objective-C is a simpler language than C++, has less features but is remarkably more easy to learn and use [kernelthread/7]. Steve Jobs' design ethos had entered the world of coding itself (Deutschman, 2000, p.55). In the words of Simson Garfinkel and Michael Mahoney [03-05-2002]:

“Unlike development systems based on the C++ programming language, such as Microsoft's Application Foundation Classes and CodeWarrior's PowerPlant, Cocoa <the later name for the NEXTSTEP development environment> is built on top of the Objective-C programming language -- a language that's simultaneously simpler than C++ and yet

better suited to creating graphical user interfaces. Whereas most people find programming in C++ a chore, most Objective-C programmers find the language to be a joy.”

All of these features allowed the NEXTSTEP environment to be easier programmable than its contemporaries with more beautiful results. For example John Carmack’s Doom game engine [Lynch 1997] and the first web-browser, and with it the World Wide Web, were born on a NeXT computer. The NeXT computer played an important role in enabling the development of the browser, as inventor Tim Berners-Lee [undated] recalls:

“I wrote the program using a NeXT computer. This had the advantage that there were some great tools available -it was a great computing environment in general. In fact, I could do in a couple of months what would take more like a year on other platforms, because on the NeXT, a lot of it was done for me already. There was an application builder to make all the menus as quickly as you could dream them up. there were all the software parts to make a wysiwyg (what you see is what you get - in other words direct manipulation of text on screen as on the printed - or browsed page) word processor. I just had to add hypertext”

Berners-Lee’s quote especially underscores the technological innovations that the NeXT platform offered to developers; developer tools that would play a significant role in the future. But the use of radically different technology made it a divergent operating system from the mainstream in the 1990s and NeXT computers remained a niche in the niche of scientific and corporate computing.

Despite its technological superiority, the integrated NeXT computer never was a profitable product. The decision to produce both the operating system and the hardware in-house and the high material quality standards made the NeXT computers a very expensive piece of equipment compared to its direct competitors. The educational and professional market was saturated with the much cheaper Sun Microsystems and Windows computers that relied on external standardized hardware which was increasingly produced offshore (Deutschman 2000, p.130-132). Therefore, NeXT computer was never able to gain enough market share to sustain itself financially; the estimated total sales during its existence only reaching 50000 units. NeXT’s main customers were government agencies and companies who were especially fond of the fact that the developer tools allowed custom software to be built cheaply [Wikipedia, Nextstep entry]. The consumer market was left to cheaper competitors. This implied that there never was a large market for standardized applications and therefore very few independent software houses existed. The customers of NeXT computer relied on custom internal software and software developers who worked with NeXT technology independently were mainly doing

consulting work on this custom software. These independent consultants were often motivated by a fondness of NeXT technology and showed a very tenacious commitment to the platform. As one interviewee stated when asked about the characteristics of the NeXT developer community:

“There was a joke within NeXT community that they'd rather be sheep farmers than step over to another technology, because it was such a fun cool technology that they did not want to do something else.”

In 1993 NeXT computer had suffered so many losses that it was forced to discontinue its hardware line and the NEXTSTEP operating system went on to compete with Windows NT on Intel processors. The software leg did become profitable with open architecture versions of NEXTSTEP and its successor OpenStep but the technologies were never able to catch on in the mainstream competition before the Apple acquisition (Linzmayer, 2004, pp. 212-213).

2.3 Merging the technologies

When Apple acquired NeXT computer in 1996 Apple Inc. was in its worse shape ever. It had just suffered a net loss of 740 million dollars (Linzmayer, 2004, p.265) and it had almost merged with Sun Microsystems in the previous year. With the acquisition of NeXT, Apple founder Steve Jobs returned to Apple as an advisor but within a year he had regained control of the company and was appointed interim CEO. He replaced most of the executives by his own NeXT confidants (Deutschman, 2000, p.245), including senior vice president for software engineering and chief architect of the NEXTSTEP operating system Avie Tevanian [Rochester Review, 1997]. These changes created expectation that Apple would shortly start to push NeXT technology within the Apple operating system [Siracusa, 02-04-2008]. Despite that, it would take Apple until January 2001 when the 1.0 version of OS X was released. In the meantime Jobs did make the company profitable again by completely overhauling the corporate strategy focusing on just a few products (Linzmayer, 2004, pp.289-299) and in the process lowering the barrier of entry for developers (Cruikshank, 2006, p.70).

The process of creating OS X is often described as a carefully crafted master plan (Linzmayer 2004, pp.278-280); however, at a second glance it is a more chaotic history. Pretty soon after the acquisition in 1997, Apple released Rhapsody. Rhapsody was an operating system that was primarily based on NEXTSTEP but it did require existing Apple developers to rewrite their code bases [Siracusa, 02-04-2008]. At this point Apple needed the support of the big software companies that had kept the company alive in the previous years:

Adobe and Microsoft. Without their Photoshop and Office software, Apple stood no chance in the Windows dominated world. The problem was that these companies had huge code repositories for those programs, which were based on the old Macintosh programming environment, and there was no chance that they were going to rewrite those in Objective-C [idem], thus hardly any software for Rhapsody was written. This proved that any new Apple operating system would only stand a chance to be successful if it supplied the means to port the existing code base of these companies.

What they came up with was a dual solution. The NEXTSTEP programming environment was renamed into 'Cocoa' which allowed software to be programmed in the NEXTSTEP way while they also transferred some parts of the Macintosh toolbox, the pre OS X programming environment, into an environment called 'Carbon'. Old code could be 'Carbonized' which allowed reuse of these old code bases that would get the big programmers on board. Old OS 7/8/9 software could also still be run in OS X under emulation. Despite that it took Apple until the release of OS 10.2 in 2002 (the second installment of OS X) when the critical mass of users finally switched from OS 9 to OS X (Linzmayr, 2004, p.280). Furthermore, Apple started to promote a Cocoa-Java bridge. The meant that people could use the then hugely popular but substantially different Java language to program in Cocoa instead of Objective-C [Garfinkel & Mahoney, 05-10-2002]. It terrified the old NeXT programming community whose attachment to the platform also relied on the elegance of the objective-C programming language.

Another incentive to get developers on board of the OS X train were the developer tools. Previous to the release of OS X, becoming an Apple software developer required an investment of around 1100 dollars on third party software like Codewarrior [Diskovery, 2001]. With Mac OS X Apple Inc. started to distribute the developer tools that they used themselves for free with every new Mac sold (Anguish et al, 2002, pp. 29-30). The tools, Project Builder⁷ and Interface Builder, were derived from the old NEXTSTEP tools to which extensive Carbon functionality was added (Mott, 2001, pp. 1-18).

Meanwhile there also was a lot of confusion amongst the smaller developers. Most of them had quit the platform during the meager years and those that were left showed a strong conservativeness regarding the new technologies. Cocoa was received with such skepticism that for a while it seemed very unlikely that the lot of developers would switch to Cocoa. Carbon, with its similarity to the old pre-OS X ways of developing seemed a lot more appealing

⁷ Which later evolved into Xcode

to most smaller developers. The old independent NeXT consultants and a few Apple ‘convert’ developers were the ones who put in a great effort in convincing the developer community to switch technologies. The first Cocoa programming books were written by the old NeXT developers Scott Anguish and Aaron Hillegas; books that were quoted as a huge inspiration by almost all of the interviewees. The Omni Group, an old NeXT consultancy house, started a Cocoa mailing list and ported games to show the virtues of OS X and Cocoa [Cook, 2001]. A good illustration of the vigor in which these third party developers evangelized Cocoa is the following excerpt of an article by ‘early adopter’ developer Dan Wood [December 2001].

“Since I still have many friends and acquaintances in the “Carbon” camp, I am interested to hear what they think of Cocoa. It’s disheartening to hear how few of these folks express any desire to learn Cocoa. Some claim that they don’t like the syntax of Objective-C. Others say that it’s too much of a learning curve. But I think that for the most part, it’s because longtime Mac developers have spent a long time investing in the Mac toolbox, and don’t want to see that expertise go to waste.....

.....That kind of thinking <switching to Cocoa> may not be rational if you “do the math.” But it’s just as hard to let go of emotional investment as financial investment. I view Cocoa as that new boat that traditional Mac developers are being offered. And it’s hard to let go of the years of honing your investment. And worst of all, I see a great community of Mac developers with “old boats” that are so immersed in their decision not to look at the new technology called Cocoa that they are going to find themselves the hares in the aforementioned fable.....<the Tortoise and the Hare>

There is no real finish line; Cocoa and Carbon will continue to exist for some time. Each has their advantages, but Carbon is the winner when there are business reasons; Cocoa is the winner when there are technical reasons. As time goes on, the business advantages of Carbon will fade away, and the market for Cocoa programmers will be larger than the market for Carbon programmers. The developers that insist on defending their investment in Carbon avoid Cocoa when it comes time to build new projects may find that the Cocoa developers have beaten them to market. I hope that this doesn’t happen. If you are a carbon developer, start learning Cocoa in your spare time.”

Eventually enough developers jumped ship to vindicate Apple's continued development of Cocoa⁸. As the reputation of Cocoa grew more developers found out that developing in Cocoa actually saved them time because the coding efficiency of Cocoa is higher compared to Carbon [Shipley, 5-10-2006]. Currently there are hardly any Indie Carbon applications under development left. The Cocoa-Java bridge was abandoned in 2005 [OSnews, 10-07-2005] and Apple increasingly started to endorse Cocoa. When it was announced in

⁸ It is impossible to state whether the influence of the Indie developers did influence Apple to continue develop Cocoa –although some respondents did suggest that– so I do not wish to impose any causality here. Apple probably wanted Cocoa to be the future and the enthusiasm of these developers surely helped in convincing Apple that they could get the market to innovate.

2007 that 64 bit Carbon support was not going to be part of future operating systems it finally became obvious that Cocoa was the unquestionable future of Mac OS X. But by this time Apple was booming and Adobe could do nothing but grudgingly accept to port its code base [Nack, 2-04-2008].

2.4 The .com bust and the maturation of online payment systems

Two other related events coincided with the release of OS X that are significant in the path-dependent process that gave OS X and its developer community a head start.

First, there was the .com crash in March 2000 [wikipedia, dot-com bubble entry] that ended a period of massive growth in the computer industry. Apple did suffer from the .com crash in terms of sales but because it already had reorganized itself in the previous years the company was never really in the same danger zone that other companies faced (Linzmayr, 2004, pp. 298-301). Apple did recover sooner than other companies and when the iPod was released on November 10, 2001, the new era for Apple really took off (idem, p.300). Meanwhile the .com bust had created huge unemployment with software developers who had gotten used to the idea that venture backed startups would be begging and generously paying for their skills. This proved to be very beneficial for OS X. Developers that were laid off or just coming out of graduate school took advantage of the economic downturn by pursuing their own creative interest while in-between jobs. They checked out Cocoa because Cocoa was a cool new⁹ technology that before the .com bust might not have been economically feasible to invest time in. Due to the .com bust probably more people switched to Cocoa than otherwise would have been the case. At least 4 out of the 15 respondents that were interviewed got involved into Cocoa programming in this way.

The .com bubble also had a second asset for the takeoff of the figuration. The overheated expectations of the .com bubble had created the infrastructure that eased payment over the Internet. E-commerce systems, like Paypal, reached their maturation. Furthermore, as many more people were connected to the Internet and the quality of connections had gotten much better, downloading software from the Internet became something that an average person could do. This meant that all of a sudden individuals could serve worldwide markets out of their own home. And these services remained after the bust. This implied that the retail and distribution barriers of entry to the software

⁹ As the NEXTSTEP history shows, it was not that new at all. But to the mainstream engineer audience who never got acquainted with NeXT's technology it seemed new and fresh.

market were eased, allowing the niche markets of Indie software to become economically viable (Anderson, 2006).

2.5 The starting point: the initial figuration of OS X developers

At the initial stage of OS X development a selection process occurred. The prime movers in the new technology were old NeXT consultants and old Apple developers who were up for renewal. These people became the initial role models for the Indie community and set the standard for what being an independent Mac developer would become. For example the former NeXT company Omni Group and an early convert OS 9 company like Panic are not only perceived as two of the most successful and inspirational Indie companies, they were also two of the prime movers in establishing the Indie communities' collective resources (Chapter 4).

Ironically, the long time it took Apple to integrate the technologies enabled it to tap into the advantages of the .com bust. It ensured a flow of unemployed talent willing to take the risk on a cool new technology they heard about, since they could not get a job in a computer industry that was facing massive layoffs. Furthermore because the .com bubble had created the necessary infrastructure to sell software over the Internet it ensured that some of the companies founded by this new generation ended up being sustainable. Subsequently when Apple took off, they took off as well. Appendix E shows some of the developments in Apple's market share after the release of OS X and illustrates the economical benefits that these prime movers could eventually reap.

Chapter 3: The sensibilities of beautiful software and economic independence; Cultural traits

3.0 Introduction

Even if social change in a figuration of aesthetic production is completely autonomous of economic influences it is still subject to a logical social transformation (Bourdieu, 1993. P.140). Economics play an important role in the Indie figuration, but it remains important to attempt assessing its cultural aspect separately of its economic logic to be able to understand the social embeddedness of the whole Mac Indie environment. According to Bourdieu (1993, P.121) reproduction and evolution of such a purely culturally driven 'field of restricted production' is defined by the outcome of competition for the power so as to grant cultural consecration. Put more simply: there are, autonomously of economic successes, certain ideas about what 'good software' is and what is in fact considered 'good'. At a specific point in time there are certain actors who embody these meanings by doing a very good job in applying these implicit rules and therefore they gain cultural capital vis-à-vis their peers. Intermediary actors, like journalists, who do not develop software themselves, can still obtain a powerful position by being recognized as people 'who get it'. In the Macintosh world journalists tend to be very much tuned into the Indie culture as well [Engst, 11-08-2006] and therefore play an important part in this process. The whole process of 'rule setting' is dynamic: new actors can enter the figuration and strive for recognition either by complying to the rules or changing them (Bourdieu, 1993, p.106-109). Still this struggle often happens incrementally by adhering to some principles and disapproving of others. Therefore cultural change is subject to path dependence as well (David, 1994). One can attempt to change the common sense ideas about what is 'good' but to do so one still needs to be accepted by peers, usually requiring a viable argument.

In the fieldwork recurring patterns of ideas about aesthetics, creativity, how to do business, and life-style within the Indie community were found. Logically, there are variations in these cultural dispositions but mostly they stem from a similar understanding in why and how they are doing what they do: a certain ethos. The importance and history of these ideas often originate from the histories described in the previous chapter. Essentially, this collection of meanings is an ethnography of Macintosh Indie culture; it sheds a light on how cultural capital is gained in the Mac indie community. It should be noted that the cultural aspects that will be discussed are separated for analytical

purposes. In practice they mutually reinforce each other in a myriad of ways and they are furthermore interacting with economic logics. Despite this last remark the importance of the cultural aspect should not be underestimated; most respondents claimed that money was only important insofar as it enabled them to sustain themselves and to devote their valuable time to what they like to do best. This motivation is captured by developer Brent Simmons [19-09-2002] in the following citation:

“One of the reasons I develop for OS X is that, when it comes to user interface, this is the big leagues, this is *the show*. That’s probably what Joel <on software, a famous blogger> would call an “emotional appeal”—and to call it that, that’s fine by me. To switch metaphors: imagine you think you’re a good writer. You think you have the talent to write excellent novels. You also have the talent to be a really great sports journalist—you could probably get a column in some newspaper somewhere and be very comfortable, maybe even win a Pulitzer or two. Which do you choose? Writing novels is the bigger risk and the bigger reward, by far. I choose that (metaphorical) path. How could I not? The other path is honorable and sensible and has its rewards too. But to me it’s the difference between an empty night sky and a night sky with all the stars shining and a big, bright *bella luna*. “Emotional appeal?” Oh yes indeed. And I don’t apologize for that for one second.”

3.1 The sensibilities of ‘good software’

To describe what is ‘good’ is giving in to subjectivity. What makes a car a good car or a piece of software a good piece of software? The answer will differ according to whom you ask this question but the answer will always be a combination of utilitarian (it never breaks down) and aesthetical (look at the nut wooden dashboard) properties. But even those aren’t unambiguous categories: is a utilitarian piece of software a piece of software that allows you to do everything possible or those things that you need very often in a simple way? We can find the base line of the utilitarian part of good Indie Mac software, which echoed throughout all the interviews, in the Omni Group’s ‘what is Omni’ [2004-2008] mission statement:

“Software should not just be useful, it should be so easy to use that it’s actually fun. It has to work right. It shouldn’t require you to know anything that it can figure out for itself. It shouldn’t have features you won’t use cluttering up the interface. It should have a narrow focus and do a good job at the stuff it does do, and work well with other programs that do other tasks. It should allow you to do simple things very quickly without learning anything and yet still allow you to do more complex things when you’re ready”.

This sort of ‘less is more’ philosophy can be dated back to the original 1984 Macintosh (Linzmayr, 2004, p.86) and this is still to a large extent the philosophy behind Apple’s software and the successful Indies. But the original Macintosh also set a standard for how these ideals are to be applied in the form of the graphical user interface (GUI). At this moment this GUI is the subject of debate between Indie developers. How much aesthetics, or ‘eye

candy', is allowed in an application for it to be fun without subordinating its functionality? Usability between different software programs is improved by consistency in look, feel and use and for that reason Apple used to impose their 'Human Interface Guidelines' (HIG) on their own software and put on a strong pressure on third party developers to do the same. The HIG involves very precise instructions on how to replicate the Mac 'look and feel'. Adhering to these principles has been so much internalized that a respondent spoke about 'being a good Mac citizen' when talking about implementation of the HIG. But technology evolves, innovations create new possible user experiences and as Apple started to violate its own guidelines in new versions of its software, developers could not resist the temptation to do the same. When Indie prominent John Gruber [Jalkut, 21-10-2006] described the HIG as dead this started a heated debate in the developer community.

"The HIG is dead. Not dead like meaningless, but dead like hopelessly out of date. It can't possibly keep up with Apple's implicit standards because Apple has abandoned some fundamental traditions of the HIG, such as uniformity in appearance and behavior across all applications. The web is proof that they're at least partly justified. People aren't stupid, they know a button when they see a button, regardless of how it's drawn. Would I prefer a consistent UI personally? You bet! Any chance of that happening again? No way. The genie is out of the bottle and developers have two choices: constantly monitor the state of the "implicit UI guidelines" and synthesize your own standardized interpretation, or else go out completely on your own and invent something new. HIG guideline is simply: does this look good?"

Because of these developments, ideas about what a good interface is have become less strict. Most respondents claim that the HIG should be taken for what they are: 'guidelines' and a good starting point for anyone who does not have a clue how the application should look. The new rule is that violations are welcome if applied in a tasteful manner; where tasteful is more determined through peer review than a set of instructions from above. Just as this citation shows, where Scott Stevenson quotes Daniel Jalkut [23-10-2006]:

"Daniel Jalkut wrote up a *little bit* on the Mac world's slow but steady drift away from the human interface guidelines. I guess it never really bothered me that Apple started bending and breaking the rules. After all it's *their* set of rules. Daniel has had a lot of great quotables recently, and this is one of them:

At the very least I think it's time for us crotchety old engineers indoctrinated by System 7 or earlier values to mellow out a bit. I got so used to defending the party line for so many years, that I stopped questioning whether it was worth defending.

First of all, LOL. Really, though, there was a time when the HIG stuff *was* worth defending vigorously. It kept things from descending into the depths of... well... the sort of stuff we see on other platforms"

This cultural shift has left cracks in the developer community. In the absence of an 'objective' set of rules opinions have diverged. Vigorous online discussions on whether a certain developer 'has gone too far' in terms of form versus substance are ongoing. The most prominent discussion is the 'Delicious generation debate' where several young developers have used the fact that their interfaces are 'against the rules' as a marketing strategy for their applications. As the developers of the Delicious Generation claim [Smykil, 31-05-2007]:

"Well it's the whole package, form and function. That's what I call Delicious today," adds Teutschler. "I think a beautiful UI only makes using apps much more pleasurable for most OS X users," says Wagner. "If they don't like GUIs, the Terminal isn't too far away in the Utilities folder..."

Furthermore their motivation seems at least to a small extent influenced by the initial peer rejection.

"I asked Casasanta, founder of the new project along with Sarner, if this group of individuals <the Delicious Generation> and this new application was in some way an answer to the criticism they have received. Casasanta made no real attempt to hide. "Yes, I guess in some part it is."

The above quote shows that these developers have a certain perspective and although they have a different opinion on what a good UI (User Interface) is than older generations of developers they still pay homage to the principles that underlie the conventions, such as usability. Because of this maintaining of principle they are still accepted by at least a significant part of the developer community. This form of cultural evolution with both its continuities and discontinuities is described by Bourdieu (1993, p.107) as 'when newcomers come into existence, i.e. accede to legitimate difference.... they necessarily push back into the past the consecrated producers with whom they are compared, "dating" their products and the taste of those who remain attached to them'. This process of 'dating' does enhance the amount of cultural capital: power, negatively, that 'old fashioned' developers have over the new generation if they are not willing to acknowledge the changing standards. Essentially these new developments enable the figuration as a whole to evolve, preventing a state of lock-in where the codes and styles of cultural performance are so much conventionalized that change is impossible¹⁰ (Scott, 2000, p.38).

¹⁰ This does not imply that change is necessarily 'better'; It just states that new ideas emerge, are sometimes able to get recognition and as a consequence compete for the power of defining standards.

The fact that these innovations in UI are, sometimes grudgingly, accepted might also have to do with two other cultural attributes of the Indie community: that of software as progress and that of software as design. There is a strong feeling that technology allows people to improve their lives and that good software enables people to actually do just that. Many developers like pieces of software that are new and innovative and they acknowledge that conventions must sometimes change to innovate. Furthermore the Apple platform has a strong history in design: making utilitarian things beautiful. The applications of the Delicious Generation might violate UI codes, it might be a bit too much ‘eye candy’, but at least they apply new technologies and do it in a, what many consider, beautiful way. These meanings relate as strong to Apple’s history as does the Graphical User Interface.

3.2 The sensibilities of being independent

We have already defined what an independent Macintosh software developer, Indie in short, is for the research purpose in paragraph 1.2. But despite a ‘formal’ definition it is very important to stress that an Indie is more than an abstract definition. Being Indie means something to the participants; their job is among other factors an important identity marker. The following quote by Buzz Anderson [24-02-2003] explains the ‘Industrial atmosphere’; a set of socio-cultural norms and practices revolving around the production system (e.g Scott, 2000, p.17), of Mac Indie development, better than I could possibly explain myself. This quote is a response to another blog post that points at the stigma of ‘being unprofessional’ that stuck to the then popular ‘shareware’ label.

“In responding to Slava, meanwhile, both Erik Barzeski and Steven Frank (both developers of Mac software) point out what is, to me, a much more problematic aspect of the “shareware” label—it tends to connote a certain level of quality that is far below what people expect from “real” commercial software. This is, of course, unfair to applications like MailDrop and Transmit which may have been developed by small companies but still exhibit a level of quality equal to or greater than that of the larger software houses. The question, then, is what should small developers like Erik, Steven, and me call our software?

As the author of an application that traditionally would fall under the “shareware” heading, and as a developer with aspirations to someday having the level of success of Erik or Steven, I have personally thought about the problems with the “shareware” designation for a while now, and I’ve come to one conclusion: that the term I prefer is independent software. The model I have in mind here is, of course, independent records labels, and I think the comparison works well in a number of ways. Like an independent record label, an independent software company is differentiated from its “big league” competitors mainly by:

1. A much smaller number of employees.

2. A preference for direct distribution channels.
3. An ability to serve niches that aren't economically viable for the larger players.
4. Less sensitivity to, shall we say, industry political correctness.....
5. Greater agility and far more sensitivity to emerging trends.

Most importantly, “indie” record labels don’t bear the stigma of low quality just because of their small size—far from it! Some of the greatest music in the last twenty years has originated from independent labels—for example, Factory Records in the late 1970s and 1980s and Creation Records in the 1990s. In the same way, I believe that much of the most innovative software in recent years (at least on the Mac platform) has been created by independent developers, and that this trend will only continue. Just as Factory and Creation were both there at the birth of new movements that later swept through their industry, I believe that “mainstream” software developers will increasingly find themselves taking cues from their “indie” counterparts.”

The above description has a few remarkable similarities with the business culture that was, according to Annalee Saxenian (1994), crucial to the development of Silicon Valley. In the Indie world a strong belief exists that small entrepreneurs with great ideas can make a difference and therefore Indies show an attachment to the ‘garage myth’ (Saxenian, 1994, p.20) of which Apple itself is a legendary example. Likewise, the idea that they advance a new technology, as a sort of ‘mission’ is strongly embedded in their identity (idem, p.61). Furthermore, Indies have a strong disdain for ‘corporate’ behavior as the remark on ‘industry political correctness’ shows. This disdain especially shows when big companies use their market power to ‘cash in’ with inferior technology, drive small companies out of business or try to hinder market access by, for example, patents inaccessible to small companies [e.g. Shipley, 26-04-2005]. In one sentence: if they do dirty competition that is not based on merit. Another similarity is the group awareness of the Indies: they are in this venture together and there is a mutual sense of respect and emotional attachment towards other companies who produce good work (Saxenian, 1994, p.31), as the mentioning of competitors in the blog post shows. The most striking difference between Indie companies and their Silicon Valley predecessors is the financial conservativeness of most Indies. The developing of their software takes nothing more, in theory, than an Apple computer, knowledge and their own hard work and so they do not need much capital to pursue their initial entrepreneurial ambitions. But even in later stages most Indie developers show a strong reluctance to financial risk, whether it is hiring employees, attracting external capital or quitting the day time job before the break-even point has been reached, even though the Mac market has been booming for the last few years. This financial prudence is something that is openly acknowledged as this quote from Kevin Hooctor [2-10-2007] shows:

“Oooo... *cash flow!* That's a huge balancing act. My company (finally) paid the bills last month, but that's no reason to get cocky and think that this will be the monthly norm from here on out. There was a lot of marketing splash last month with MoneyWell because of its top billing on Apple's downloads website. I don't have that prime real estate anymore so I have to assume that I will need to start building a war chest for the dry months. This company was inspired by the design of Delicious Monster, shortly after I finished listening to Wil Shipley's now famous WWDC speech, and it has very little overhead, but there are still operational costs because this is not a part-time business. No Thirst Software is my only job and I have no desire to update my resume at this point. That's why I have designed it to go the distance and weather any storm the computer market can throw at it; short of Apple going belly up, but let's not even think about that. <shudder>”

The financial prudence can be traced back to two important aspects of being Indie. The first reason is historical; as chapter two shows the .com crash played an important role in the genesis of the current Indie culture. But even more important and less speculative is the fact that being economically independent ensures that developers can pursue their own ideas. They do not want to feel the pressure to compromise their artistic vision to external financing and therefore fall into the trap of ‘corporate logic’. Illustrative here is the following essay by Steven Frank [undated] of Panic software where he explains their motivation to try to live up to the ethos of pioneer software developers in the early 1980s.

“Another thing that seems to have disappeared is the cool software company. Is there a Beagle Bros. <an old Apple II software design company> of the 90's? Most seem really straight-laced and are obsessed with "biz". There are a few with a sense of humor, of course, but they are seldom seen and often overlooked in favor of the "serious" companies. With the software industry being so huge now, compared to the days of the <Apple> II+, is it possible for a software company to be as personal as Beagle Bros.? Is it still possible to build a software company that will capture the imagination of the next generation of computer users? I don't know. But I'd really like to find out.”

3.3 The sensibilities of good business practice

Interrelated with the sensibilities of being independent and what ‘good software’ is, is an informal code of conduct how to behave towards customers and each other has evolved. This also plays a strong role in the amount of peer recognition, and therefore cultural power, that is conveyed upon developers. The most important principle of this is respect and responsiveness towards customers and each other. Behavior on blogs should be sensible and friendly and developers are supposed to behave responsible to their users as well. The following quote by Brent Simmons [28-03-2003] illustrates this consensus.

I'd like to have more resources for other developers. I have a couple tutorials and some open source classes ... and I'd like to do more of that. If I have some success with NetNewsWire and make money, it's good karma to be as generous as I can, not just with people who use my software but with other developers. As a developer I've always

appreciated when other developers do that, so I try to do it too. So, whenever I can make time I plan to add more tutorials and open source Cocoa code and things like that. Well, even if there's no real karma at work at least it makes me feel good and it's fun to do.

Not all developers need to have the karma of Brent Simmons to be respected but the interviews did show that this is the kind of ideal type behavior that made Brent Simmons one of the bigger 'rockstars' in Indie development. And this reputation enhancement does work two ways. When developers get a reputation for offering bad customer service, for example, it negatively enhances their reputation in relation to their peers, although this usually will not be publicly acknowledged. On the contrary, conflicts between developers are not supposed to be fought online but preferably in personal communication. Formal legal action, although it rarely if ever happens, would be an outrage in the community maybe just because it is another thing that 'a big company' would do. Altogether these informal social norms make up a code of conduct of do's and don'ts in the Mac Indie figuration. The strongest example of this code of conduct is in the domains of competition and the pricing of software that led one respondent to describe the Indie community as 'labour union like'.

The problem that Indie developers face when trying to make a living out of their software is that they are dependent on customers paying for it to sustain themselves. Competition with similar products on price incentives is something that is beneficial to no one; especially in a world where there is a flood of often badly made free alternatives to a piece of software. There is a strong fear that users can be 'spoiled' when they get used to good software being cheap or free. Therefore, when a developer of a 'quality application' decides to lower its price there is a feeling that this endangers the sustainability of other developers because the price of software devaluates. The following blog post by Paul Kafasis [09-01-2008] deals with a recent example. He responds to a sentiment of happiness that surrounded the fact that a very popular application, the RSS reader NetNewsWire, had just become free.

"While I understand the sentiment, as someone who makes his living selling software, this is a disheartening thing to read. Yes, it might seem great if all software was free. But while NewsGator <the company had just that bought NetNewsWire> has the financial resources to accomplish this move, most companies do not. Very rapidly, you'd see a shrinking of the market, a loss of innovation, and ultimately, a decrease in quality. There's no market for commercial software on Linux, and the quality of solutions simply isn't on par with what's available on the Mac. By attaching a value to software, we give it value, in a self-fulfilling prophecy."

The devaluation of software is another discussion in which the Delicious Generation group of developers was involved in a negative way for a lot of peers. People in and around this group created a series of bundles and promotions of Indie Mac software, the most well known being the 'Macheist' bundle, that essentially sold licenses of software for sometimes 10% of the normal price. These bundles were hugely successful and raised issues, next to the pricing, about what would be a fair division of the profit since the economics of the promotion turned out to be very profitable for the organizers [Slashdot, 17-12-2006], at least in the first year when they organized it. But not all Indie developers subscribed to that point of view; some considered it beneficial to them and the exposure a good thing for the community as a whole; they stressed the fact that developers can make up their own minds. Furthermore when it came to the point of these debates really 'splitting' the community some developers chose to remain 'neutral' and refrained from mentioning the controversy in public at all. This can be considered a strategy to preserve respect with both sides of the debate, thereby saving cultural capital and preventing themselves from alienating potential users.

Although there are longstanding debates whether the user indeed becomes 'spoiled' if some software is free and whether a promotion like Macheist is bad when all the participants involved do so consciously; it still is agreed upon that competition should not be taken head on by price. Imitations of existing software are not accepted in the community and if you do compete with another application you are supposed to do this by taking another angle on functionality or UI design and have the consumer choose the one that he or she prefers. In this way everybody is supposed to find their own niche and the best applications will surface out of the competition on quality. It is really not done to promote your application on the expense of your peer competitors and, on the contrary, a lot of Indie developers have no problem mentioning their own competitors respectfully as long as they consider their applications good. Mike Lee [16-06-2008] illustrates this by making the comparison between his own application 'Twinkle' and his competitor 'Twitterific'. It should be noted that he does not even consider their vying for the users preference 'competition' which is a matter of definition. In the definition of Michael Porter (2000, p.257) the sort of non-competition Mike Lee mentions could even be regarded a strong competitive strategy. The quote also shows the 'union' characteristic of the community; developers who do not 'follow the rules of competition' are simply excluded from the Indie community and thus of some of its collective benefits (Chapter 4).

“Twinkle and Twitterrific are not actually competing apps. I would go so far as to advance the theory that there actually is no competition among the central core of Mac and iPhone development. Allow me to explain, using Twinkle and Twitterrific as an example.

Craig <the Twitterrific developer> and I have different visions for our respective applications. If we had the same vision, we’d work together. As it stands, Craig is trying to build the best iPhone Twitter client, and doing a hell of a job. I have different plans for Twinkle, which go a lot further than Twitter. When a user chooses between these two apps. It’s not a matter of right versus wrong, or which app is better. Rather, they will choose the app whose vision best matches their own.

Competition does exist in our market, but it’s mainly perpetrated by visionless hacks who see money in an app and try to recreate someone else’s work in an attempt to cash in. These people aren’t really part of the community, and frankly, I’d like to see it stay this way.”

Paradoxically, the above discussion does not apply to open source software. Indie developers are usually strong advocates of open source software and many of them participate in open source projects themselves or open source the non-pivotal parts of their own software. It is another thing that enhances reputation in a positive way. The reason the two business models –Indie and open source– do not bite each other that much is because open source software has a tendency to focus on the infrastructure rather than the end-user experience of software; the geeky stuff. Getting the UI details right and making the software into a finely tuned end product is less interesting to open source contributors and leaves a market for commercial software (Friedman, 2006, pp. 104-112).

3.4 Lifestyle aspects and spatial preferences

Joel Kotkin (2000, p.16) points out that there are generally speaking two faces of the information economy: a ‘hard’ side built around quantifiable sciences and mathematics and a ‘soft’ side that is concerned with the content that flows through the expanding information pipelines of the new age. According to Kotkin these two faces have a distinct geographical articulation. The ‘hard’ activities are to be found in suburban ‘Nerdistan’ while the soft activities show a strong urban preference. According to the interviewees there is a correlation between the kind of software engineer you are and the type of industry you work in, and Indies seem to definitely be on Kotkin’s soft side of the information economy. Some of the people interviewed claimed benefits from a computer science degree on an abstract level –if they have one– but most of them did not consider it crucial for their work. Moreover, a striking number of respondents had a background in the visual arts or music that they considered equally important. It seems that people who end up being an Indie software developer distinguish themselves from their ‘hardcore’ counterparts

by valuing cultural inputs as important or even more important as the abstract data structures and algorithms of traditional computer science. In general this coincides with a preference for urban living. Most Interviewees showed a strong disdain for Silicon Valley as an inspiring place to live or work. It is considered corporate, ugly, expensive and a congested place.

Indie Mac developers appear to be prototypes of what Richard Florida (2002) calls the 'creative class'. On average they do not make a sharp distinction between their personal and professional lives (Florida, 2002, p.14), they appreciate systems of meritocracy and subscribe less to the status of wealth, they seem more motivated by the respect of their peers than by money (idem, p.78) and rally around their professional identity as a software entrepreneur (idem p.80). Indies fall into the category of 'free agents' (idem, p.107): people whom willingly sacrifice the securities of a job to be able to pursue their own creative interests. Even the bigger Indie companies, those with employees, usually have grown out of just a one or two man shop. The interviews show that the owners of these companies dedicate as much time as possible to the actual creative process. Managing other people instead of contributing code or designs themselves is often seen as a necessary evil that has to be prevented as much as possible.

These similarities in lifestyle preferences invoke a tendency for spatial clustering and the research seems to indicate that there is a correlation between where Indie developers live and the places that are identified by Florida as the 'centers of the creative economy' (Florida, 2002, p. 235); the places that research was conducted –San Francisco, Portland and Seattle– all score high on Florida's creativity index (idem, p. 246).

However, it does seem from the interviews that the presence of other Indies plays only a limited role in choosing where to live. The spatial agglomeration of Indie developers seems to evolve more from similar lifestyle preferences than around interaction with peers (paragraph 4.3). Indie developers are relatively footloose: they serve worldwide markets from the Internet, use the same Internet for interaction and from an economical standpoint they can locate anywhere providing there is an Internet connection. For example, the center of Mac development in general is the San Francisco bay area. Despite this, many Indies prefer other urban locations because rents are high and due to the nature of their job they do not need to operate from this particular place. Location choices are to a large extent based on the cultural atmosphere of places and not so much on economic rationality but when it does it is centrifugal as the last example shows. Living in an urban area is an identity

marker and living on the West Coast of the USA is quoted as ‘a receptive atmosphere’ for entrepreneurship in general. However, evidence of economic interdependencies as a casual factor for urban living as suggested by Scott (2000, p.19) in relation to Indies seems very thin at this point.

3.5 Understanding cultural capital in the Indie Mac world

According to Bourdieu (1993, p.95) ‘sincerity is one of the preconditions of symbolic efficacy and can only be achieved if there is a ‘fit’ between the expectations inscribed in the occupied position and the dispositions of the occupant’. In other words: in Bourdieu’s ideal type field of restricted production this implies that the more ‘authentic’ the Indie is according to the dispositions of the group, the more powerful he or she potentially is. Still, this is only true if all developers have the same interpretation of the underlying principles. The Mac Indie world has been growing enormously over the past few years and as a result the ‘old’ common sense rules of doing business are being challenged by new interpretations. Furthermore, there are more variations in the paradigms of quality and because of the growth of the market there is increasingly more money at stake. The different sets of principles about being an Indie cause different overlapping ideal types of Indies to coexist and compete for recognition setting a pace for cultural evolution. From an evolutionary organizational viewpoint (Nelson & winter, 1982, pp. 96-136) the incremental changes in the different sets of principles cause an increasing variation in the routines that govern the social organization of the Indie figuration (idem, p. 130-131). These routines, characterized by the evolving cultural dispositions, form the specific informal institutional setting in which doing business in the Indie Mac world is socially embedded. Especially with the absence of formal hierarchy the actors endowed with the right amalgam of cultural and economic power will set the standards for the social reproduction of the Indie figuration in the future. But because the new routines evolve out of the old ones organically this social reproduction is a logical continuation of the path-dependent process of the development of the Indie culture. How the social reproduction of this culture works and how this is related to interactions is investigated in the next chapter.

Chapter 4: Communities and networks of practice; The hardware of the Indie figuration

4.0 Introduction

In chapter three is attempted to unravel the culture that guides behavior within the Indie figuration. Cultural codes can be regarded ‘the software’ that inform the often common sense codes, rules and instructions on which social interactions thrive (Chabal & Daloz, 2006, p.86). To complete the metaphor: the ‘hardware’ then could be considered the social interactions within the figuration; these are the subject of this chapter. Defining the boundaries of the Mac Indie figuration is troublesome because so many different actors play a role. The core is of a set of interacting actors that are considered part of the ‘Indie developer community’; these are small software development companies and intermediary actors like journalists and influential consultants who provide connections to new members and end users. There is not always a sharp delimitation between these roles: some developers are involved in journalism or hosting community websites that influence their power position in the network. Developers might also have external connections by doing consultancy work for ‘non-Indie’ firms or by being active in the web design world. Also, the fact that Indies are often the core ‘power users’ of each others products should not be underestimated. Furthermore, the end-users and Apple Inc. play important roles. All these actors influence the outcome of group processes and should therefore be considered part of the figuration (Elias, 1971, pp. 145-146). Because there is no formal organization delimiting the Mac Indie figuration it is impossible to define exactly what is part of the ‘internal influences’ and the ‘environment’ as the boundaries between the two are blurred by definition (Perrow, 1986, pp. 192-193). Despite this difficulty it is heuristically still very relevant to frame the interactions into understandable chunks. The main question and empirical research is restricted to interactions amongst developers and between developers and Apple Inc., but the functions of these interactions cannot be assessed without taking the other parts of the figuration into account. To still be able to provide a holistic picture and answer the research question, Michael Porter’s theories on competitiveness and clusters will be used as a framework. Firstly, paragraphs 4.1, 4.2 and 4.3 will assess the interactions and particularities amongst Indie companies, or as Porter (1998a, pp.178-182) would call them: ‘(inter)firm strategy, structure and rivalry’ and ‘related and supporting industries’ (idem, p.166). Secondly, in paragraph 4.4 and 4.5 the ‘demand conditions’ (idem, pp.174-175) will be inserted into the perspective by looking at the role of users and the specific

markets Indies operate in. With the other factors: 'Factor conditions' and another part of the 'related and supporting industries' (idem, p.166) interactions with Apple Inc. play an important role; they are subject of chapter five and will be addressed there.

4.1 Applying cluster theory to Indie developers

Michael Porter (2000, p.254) defines a cluster as a “geographically proximate group of interconnected companies and associated institutions in a particular field, linked by commonalities and complementarities”. Clusters should be seen as an organizational form that is somewhere in between arm’s length market relationships and vertically integrated firms (Porter, 1998b, p.79). Because most cluster participants do not compete directly due to for example the heterogeneous nature of their products, there are economically viable opportunities for collaboration and collective institutions without distorting competition (Porter, 2000, p.255). It should be noted that the sort of competition Indie developers engage in amongst each other (paragraph 3.3) fits this prerequisite well. The main criticism on cluster theory has been its emphasis on the geographical dimension (e.g. Simmie, 2004). The reasons for Porter to emphasize this geographic dimension are local spillover effects (Porter, 2000, p.259) that rely on for example tacit knowledge transfer and trust relations that are difficult to sustain over long distances. This is because face-to-face contact is presumed an essential (Cumbers & Mackinnon, 2004) prerequisite for trust relations and tacit knowledge transfer to occur. It is exactly the point where the debate on social and spatial embeddedness comes into place. Tacit knowledge transfers and trust relations rely ultimately on the properties of the social interactions between actors (Breschi & Lissoni, 2001, p.262). Therefore social proximity, being part of an in-group, instead of spatial proximity is the ultimate determinant of the interactions that Porter considers crucial to the local dimension of the cluster concept. Indie developers show only very limited clustering in the spatial sense; when they do it is more related to similar lifestyle preferences (paragraph 3.4) than to cluster based interactions. Because interactions between developers do occur frequently but are based on temporal physical proximity on conferences and social software such as blogs, Twitter and mailing lists (paragraphs 4.2, 4,3) I feel that use of the cluster concept is justified.

4.2 Relations between Indie developers

Interactions between firms in a cluster can be divided in vertical relations, concerning in- and output relations within the value chain, and horizontal relations between firms that perform the same activities (Malmberg &

Maskell, 2001, p.11). Most interactions amongst developers are of the latter type. Vertical relations do occur but much more on an exceptional basis in the form of the outsourcing of icon, UI design and contracting work. This implies that there is little economical necessity for relations between Indie companies; they do not need each other's output for their own work. The sorts of relations between people of the same profession sharing a collective sense of identity from their work have been labeled 'communities of practice' (Brown & Duguid, 2000, pp.126-128). Communities of practice are defined as groups of workers informally bound together by shared experience, expertise and commitment to a joint enterprise using the same repertoire of resources in their work (Sole & Huysman, 2001, pp.22-23). It explains how new members become 'part' of the community: by learning to be in practice. Furthermore, this explains the correlation between one's profession and identity (Brown & Deguid, 2000, p.153). The concept of communities of practice was derived from organizational sociology and therefore mostly applied to informal organized social groups within a firm (Gertler, 2003, p.86). Despite that, the concept shows its significance when we compare its properties with the sort of interactions Indies have.

Most of the interactions between Indie developers involve the sharing of information: they help each other out on problems they encounter while programming and usually they have no problem of sharing snippets of code with their peers even though this is technically their intellectual property. But also information sharing on how to run a business, how to organize credit card processing, who to appoint graphic design to, speculation on the behavior of Apple Inc., and other rumor sharing occurs between Indies. Most of this information sharing is online: people answer questions on mailing lists, write blog posts on problems they encountered and if relations become stronger it expands to one-on-one media such as chat clients. An intermediary form of social software called Twitter plays a very important role within the Indie community. Twitter allows you to post short 140 character messages called 'Tweets' which are received by all the people who have subscribed to your 'Twitterfeed'. Interviewees described the role of Twitter as a sort of 'virtual water cooler'. It allows developers, who mostly work at home alone, to ask casual questions to a lot of people at the same time. It can be categorized as a digital solution to the lack of peer support that characterizes working at home alone (Brown & Deguid, 2000, pp. 76-78). The research also gave the impression that this social aspect of the interactions was more important to the small and decentralized bigger Indie companies than to the centralized Indie companies in which the employees share an office. Thus, it gives support to Brown & Deguid's findings. Nevertheless, these small, often chit-chat like

conversations form the backbone of interactions within communities of practice (idem, pp. 102-109) and social software allows Indies to do this while being physically apart.

It should be noted that the Indie community has grown to such a size that it would probably be better to speak of a network of practice or a constellation of communities of practice (Sole & Huysman, 2001, p.25), since not all Indies know each other that well. We have seen in chapter three that there is a certain hierarchy within the Indie community: some actors are famous, more experienced and therefore more powerful. There is a limit to the amount of people one can have a reciprocal personal relationship with (Gladwell, 2000, pp.178-180) and as such each developer is only connected in the sense of a stronger tie to a limited number of other developers. For example mailing lists that have too many newbie questions are abandoned by experienced developers who no longer learn anything themselves. Instead, they create new ways of more restricted communication channels to communicate on their own level, like IRC channels¹¹ [Stevenson, 22-11-2004]. However, barriers of entry to these more restricted institutions seem to be more related to skill as to anything else and it seemed from the interviews that such a hierarchy of resources should be considered a ladder: newer, now intermediary experienced developers replace the older generation to answer newbie questions. Despite this stratification there seems to be a sense of openness in the community in which all developers are accessible to the others and each developer interviewed seemed at least to be plugged into the network to a certain extent.

A remarkable finding in the interviews was that most Indie developers considered their competitors friends. Apart from providing each other with information and solutions the interaction has a lot of ordinary camaraderie that shows the strong affective elements of the interactions. They barely make a distinction between work life and private life or between colleague and friend. This quote by developer Lee Fallin [8-11-2007] underscores the discussion above and furthermore it points out the strong role one's personality plays in Indie development.

“Blogs and Twitter are amazing ways to share your thoughts and get feedback. For indie mac developers Twitter is especially nice. Being an indie developer can be a lonely experience sometimes, and Twitter provides a sort of virtual set of coworkers that you can use as a stress reliever..... An interesting question related to this ability to shout out to the world as it were, is where does your company's voice end and your personal voice begin?

¹¹ IRC stands for 'Internet Relay Chat', a form of multi-person online chatting.

Especially when you are an indie developer and in a lot of people's eyes (you == your company).”

The mailing lists and IRC channels already have been mentioned as some collective resources the Indie community has at its disposal. Apple supports some of these resources^a but throughout the years Indie companies have developed several independent collective resources as well. To learn how to program in Cocoa there are tutorial sites like Cocoa Dev Central and Stepwise^b. For technical questions there are non-Apple initiated mailing lists, for example MacOSX-dev run by the Indie company Omni Group and the wiki site Cocodev^c hosted by Steven Frank, one of the founders of Panic, which is another ‘famous’ Indie company. Furthermore there are various blogs almost solely devoted to technical issues like: Cocoa is my girlfriend, Chis Hanson, Ridiculous fish and Katidev^d. Besides those it is not unlikely that Indies themselves will post technical solutions on their blogs that are also devoted to other matters. There is a mailing list devoted to business questions^e and several podcasts are devoted to Cocoa programming^f. These podcasts also have a role in reproducing the Indie culture by actively reporting on the non-technical issues of being an Indie company. These collective activities seem to play a large role in the Indie figuration: apart from their utilitarian usefulness they strengthen a sense of common identity and confirm the status of important developers. To illustrate the effectiveness of the resources and the fact that they are interlinked the following post by Matt Long [19-04-2008] on the ‘Cocoa is my girlfriend’ blog is insightful, once again preaching the expectations on design that is embodied in the Macintosh experience.

“I’m no graphic artist. This has always been the case and probably always will be. One of the first keys to being successful with any venture is to realize your limitations and weaknesses and be prepared to pay the professionals who can do the work and provide what you need. While watching the macsb mailing list, the name Jordan Langille came up several times as a graphic artist who could provide the shiny candy coated Macintosh icons and logos that people expect. I contacted Jordan at One Toad Design and within a couple of weeks he provided me with this icon for my application.”

^a [<http://lists.apple.com/mailman/listinfo>]

^b [<http://www.cocodevcentral.com>], [<http://www.stepwise.com>]

^c [<http://www.omnigroup.com/mailman/listinfo/macosx-dev>], [<http://www.cocodev.com/>]

^d [<http://www.cimgf.com>], [<http://chanson.livejournal.com/>], [<http://ridiculousfish.com/blog/>], [<http://www.katidev.com/>]

^e [<http://tech.groups.yahoo.com/group/macsb>]

^f [<http://www.cocoacast.com/>], [<http://www.cocoaradio.com/>], [<http://www.mac-developer-network.com/>]

This blog post also illustrates the last important function of the collective institutions: the fact that it provides the infrastructure for information on contracting work and collaborations. Most developers do not have the variety of skills that are necessary to make a good application independently, implying that the artwork for applications is often outsourced. Moreover there are overlaps between the Indie community and the worlds of web design and contracting work for programming. These reputations of subcontractors and job offers are transferred through the social network of the Indie figuration.

4.3 Embeddedness of the interactions

On several occasions in this thesis the logic of the connection between social and spatial embeddedness has been questioned. It has been stressed (paragraphs 1.1, 4.1) that presumed 'local' spillover effects are essentially a social rather than a spatial phenomenon. But this is only a partial answer because it still leaves the question open on how this social embeddedness comes about. Answering that question in regard to the Indie community is the purpose of this paragraph. The importance of local embeddedness is stressed in the academic literature for three interrelated functions: trust relations, tacit knowledge transfer and local culture, all of which supposedly require intensive face-to-face contacts between actors and are therefore confined to the local scale (MacKinnon et al, 2002, pp. 300-301, Saxeninan, 1994, pp.161-164). Consequently, when we want to argue that the physically separated Indie Mac community is able to sustain trust relations, tacit knowledge transfer and a distinct culture, we have to answer how.

Trust relations

Why do people invest a lot of energy in building a Cocoa resource site that makes the work of their competitors easier? Why –as fieldwork revealed– do people share code, their intellectual property, with people they have never met? All these collective resources, which are potentially beneficial to the whole figuration, rely on a strong sense of trust and reciprocity on the community level, as this blog post by David Sinclair [23-04-2007] shows:

“I've benefited from advice and comments on mailing lists like CocoaDev, and like to give back to the community. In the past I released some of my Classic Pascal code, and now I'm doing the same for some of my Cocoa code. I am a chronic generic code writer, which I know many people frown on, but with six applications to maintain, shared code is very useful and efficient. An important part of this shared code is my Cocoa categories, that extend Apple's classes with convenience methods and new functionality.

These categories are now available to other Cocoa developers to use in your own products, if desired. The code was written over the last several years, so some of it could be replaced

with more modern techniques, but hopefully a lot of it will remain useful. They are certainly used a lot in my products.....

.....You can view the code online and copy select snippets if you wish, or download .zip archives for each, or a single archive with all. This code is generously licensed; you are welcome to use it in your own products, including commercial; all I ask is a mention in your credits or website, and that you tell me you're using it. Use as little or as much as you wish.”

Trust is a very elemental concept in human relations. The expectation that all participants keep their promise, reciprocity, is the basic condition of any transaction. A fear of opportunistic behavior, taking advantage of that expectation, is regarded a fundamental reason why collective arrangements that would benefit all participants do not happen. Overcoming this dilemma of collective action requires a confidence that the other people involved can be counted on regarding their contribution to the collective (Granovetter, 1985, p.490). As transactions get more complex and costly, the need for trust grows: there are more frequent and more complex information exchanges and deeper investments by the participants (Perrow, 1986, p.240). The reasons why face-to-face contacts are regarded crucial in these sorts of interactions are twofold. Firstly, frequent interactions between people helps get to know each other better; know that they will run into each other again and therefore have an incentive to live up to the expectation. According to Granovetter (1985, p.492) the reason diamond merchants do not swindle each other is because they monitor one other's behavior closely in a tight knit community. Distance works as a barrier because it presumably makes interaction less frequent, but as the empirical research shows the lack of frequency of interactions between people can be –and is– solved by social software. The stakes may not be as high as with diamond merchants but opportunistic behavior will be noticed even in the lack of face-to-face interaction because of frequent interactions and peer pressure via the Internet. The second reason relates to sensory quality of the interaction: the compulsion of proximity (Giddens, 2006, p. 154-158). Face-to-face interaction or co-presence supplies much richer information on how people think and feel and gives a stronger impression of sincerity. Innovations in social software go very fast, Twitter being an important improvement, but this lack of interpersonal bandwidth is something that respondents recognize and actively seek to solve. Obviously collective arrangements do come about in the Indie figuration and there are mechanisms at work that solve this problem. Academic literature, supported by the empirical findings, gives four probably interrelated additional explanations why it is not impossible that trust relations and collective goods occur in the Indie figuration.

Firstly, it should be noted that cyberspace is very efficient for the production of collective goods; the production function –the relationship between the proportion of the group contributing and the proportion of the group benefiting– is high (Kollock, 1999, p.226). Any piece of information posted on the Internet is a public good to all people who can decode the content of that information. The size of the group necessary to produce public goods is often only one on the Internet because of the limited resources that are necessary. Therefore Internet communities have a much larger chance of becoming a ‘privileged group’; a group of which one member is able and willing to provide a public good by him or herself (idem, p.226). Only one Indie developer needs to be willing to invest and start a wiki site like Cocodev to get the ball rolling and provide the whole community with an online code repository. Furthermore, these collective contributions often have a lower cost than one would expect. Open sourcing a piece of software that someone developed for private use but isn’t marketable hardly adds a lot of cost for the developer so, if there are non-material benefits to open sourcing, collective and private interests could easily converge (Kollock, 1999, p.229).

Secondly, not being physically co-located does not automatically imply that there are no face-to-face contacts at all. Two significant patterns of face-to-face contacts have been found in the interviews. The first are local organizations of developers: Cocoaheads¹⁸ and Xcoders¹⁹ and more informally: Cocodevhouse²⁰. These are sort of show and tell meetings between developers that are actually localized institutions; San Francisco, Seattle and Portland all have such a group. In the interviews a lot of attention was given to these groups just because they are exactly the kind of institutions that are supposedly significant for localized spillovers. In spite of what was expected, the result of this inquiry was that they did not really matter too much for sustaining the Indie community as such. These meetings were considered a nice way of spending time; not as much because of the technical content but because of the socializing with peers afterward; the beer drinking. Still, they should be regarded as supportive rather than pivotal institutions. Developers easily skip such a meeting if time is short and others do not even think it is worth the drive. They are probably more important to developers just starting out to get to know some local people that have the same interest; it really is a social occasion with like-minded persons with a limited scope. On the contrary, the second pattern of face-to-face interactions found is very

¹⁸ [<http://cocoaheads.org/>]

¹⁹ e.g. [<http://www.seattlexcoders.org/>]

²⁰ [<http://cocodevhouse.org/blog/about/>]

important: the yearly conferences being Macworld, WWDC and C4. Macworld is an Apple trade show in San Francisco, WWDC (World Wide Developer Conference) is the yearly developer conference organized by Apple and C4 is a small but influential Indie conference organized in Chicago. Each of these conferences has a different official purpose: interacting with users at Macworld, learning to apply new technology at WWDC and C4 is a specialized Indie conference that is inspired by a Hacker conference –MacHack– that used to be very important to the Indie community but ceased to exist after 2003 [Rentzch, 08-09-2006]. However, the interviewees cite all three occasions as events where necessary complementary face-to-face contacts take place. Conferences, next to their official functions, seem to function as a catalyst of interactions between developers. They complement the missing aspects of social software and therefore the hypothesis that conferences can substitute for some of the localized cluster effects (Maskell et al., 2004) seems justified.

Conferences, however, do not explain all trust relations in the Indie world, the community is simply too big for that as the following blog post by Scottish developer Frasier Spiers [2-11-2007] shows – so complementary explanations remain relevant.

“When I was younger and less sure of myself, I used to depend on chance social interactions to do some professional networking. My default position was that none of these gods of Mac development would deign to answer my email, so I had better coincidentally bump into them at some event in order to get on their radar. In retrospect, that was kind of a stupid assumption.

My experience of the Mac developer community has been that help and encouragement is freely given, but genuine respect is earned through shipping code. In some ways, living far from the centre of the tech/social scene provides a bit of space, time and quietness to get to the point of actually shipping and supporting code.

I’d love to be able to visit the US a few more times each year, just to do the socializing aspect. WWDC is great, but an insanely large and busy event. There’s practically no possibility of randomly bumping into the people you know, unless you know the tallest Mac developers out there (and right now I’m wondering whether that’s Craig Hockenberry or Blake Seely :-).”

The post of Frasier Spiers also points out a third reason why there is a low necessity to co-locate: people need each other but not continuously. The main body of work when developing software is writing the actual code that requires, in Spiers’s words, ‘space, time and quietness’. Because developers are not dependent on each other’s output they are not depending on each other’s time schedule. This loosely coupled characteristic (Perrow, 1986, p. 202) of the Indie network eases the necessity of trust relations somewhat.

A final explanation for the trust relations in the Indie figuration lies in the community concept. Defining your industry as a 'community' already presupposes some form of collective self-definition, a shared identity and the presence of reciprocal norms. The very genesis and culture of the Indie figuration (chapters 2, 3) provides the group attachment that can distinguish a virtual space as a community (Wellman & Guila, 1999, pp. 177-178). Peter Kollock (1999, pp.227-228) provides three motivations for people to contribute to collective goods in such a virtual community even before considering altruism. His first explanation is that of 'reciprocity in a system of generalized exchange'. This means that people have faith in the fact that others will compensate for the favor granted later in time. One builds up credit in the community by helping other people that later can be 'exchanged' if one needs help him or herself. The second explanation is reputation enhancement: by showing off your knowledge and willingness to contribute one can rise in status as perceived by peers. His third explanation is a sense of efficacy: when people recognize that their behavior has a positive influence on the environment it may support their self-image as an efficacious person. Kollock notes that for all these motivations a self-definition of the group and its environment is a prerequisite for collective action. Besides this, Kollock stresses the fact that there needs to be a way to define the contributors so that they can be rewarded. These remarks seem to both strongly support the arguments on power within the figuration in chapter three and stresses the path dependent genesis of the Indie community as a necessary basis for the observed system of reciprocity.

The interviews showed that establishing trust relations in the Indie community usually follows a common trajectory. First a developer is noticed by the others because he or she releases a piece of software, writes an interesting blog post that gets linked from a blog that is already renowned and respected in the community, or does a good contribution on the mailing list. Whether this person is trustworthy is 'read' in the online contribution. The online track record as it is recorded by google functions as a business card that allows the peers to establish an initial conclusion on their reliability and personality, a process that has been identified by Judith S. Donath (1999, pp. 38-39). The criteria on which people are judged are for example the quality of their software and their friendliness in responses: the sensibilities that are described in chapter three. Subsequently, the interactions on the conferences give a face-to-face conformation to a relation that was already established online. Interviewees point out that these meetings often result in correcting the picture they had of a particular developer: people they disliked on the basis of their online personality turned out to be friendly persons. The

conferences provide an opportunity to test a first impression that has been made online. After the conference, the now stronger relation is again sustained with social software until the next conference. Wellman & Gulia (1999, p.179) argue that such a mix of remote and personal communication was already strong enough to sustain strong ties in the age of telephone. Therefore it is only logical that social software with its ever-improving variety of means and intensity of communication only positively enhances this basic proposition.

Distinct culture

Throughout this thesis it has been argued that a distinct culture plays an important role in the Indie figuration. Therefore it is not the question whether there is one but rather how it is sustained and reproduced. According to Allen Scott –following Salais and Storper– such a culture evolves in response to the peculiar tasks and problems that interrelated groups of producers face at every turn (Scott, 2000, p.17), which brings us back to the communities of practice framework. This raises two related questions. First, there is the question whether the Internet with its limited sensory richness can function as a carrier for the presupposed shared identity of communities of practice. Secondly, we need to understand how new members of the community learn how to become a member –get socialized– to such an extent that they consider the Indie ethos part of their own identity through primarily online means.

Asensio & Hogson (2001, pp.67-70) argue that the semiotic content on the Internet can be regarded as a set of cultural artifacts strong enough to bind groups together around a shared common sense web of meanings, which according to Clifford Geertz (1973, p.5), is the essence of culture. As Asensio & Hogson (2001, p.67) put it: “groups of users, most of who had never had face to face contact, devise complex systems of symbolism and textual significance to ensure that they achieve common understanding and richness of communication despite the lack of visual cues. These complex systems are variable, emergent and unique to that particular group and that is what constitutes the identity of the community.” These findings give support to the impression that Indies are able to sustain their own distinct culture primarily through online communication. Socialization in communities of practice involves a process of ‘learning to be’ instead of just ‘learning about’ (Brown & Deguid, 2000, pp. 117-146). When people try to acquire knowledge to become, in our case, an Indie software developer the utilitarian information of how to solve certain problems is embedded in the social context through which they learn. The identity that you develop determines what you pay attention to and

what you learn and so identity formation influences information gathering and the other way around (idem, p.138). New members learn how to define a situation and decipher meaning in online socialization in the same way as one would in a face-to-face situation. The only difference is that it requires a greater effort on behalf of the newcomer; they have to take an active approach to information acquisition (Ahuja & Galvin, 2003, p.174). But because information seeking is usually the incentive to connect to the community resources this seems to happen automatically.

To give a concrete example of the socialization process: If a developer learns Cocoa through the tutorial sites and the mailing lists, they do not only learn how to develop a piece of software but they also learn how to behave on the mailing lists to get a good answer, they also learn how peers do their business. If you have a question on how to handle a certain problem you probably end up at an informing blog post of an Indie developer, like this one from Gus Mueller [25-12-2005]:

“You've got to make it look and feel nice as well. Make it a Macintosh™ application. Someone commented to me the other day that both Brent Simmons (maker of NetNewsWire) and myself seem to have a sense for making usable user interfaces. I then pointed out that we both just try and figure out what Panic <another Indie company> would do..... Emulate who you want to be like, but don't copy because that's lame*. I guess that's lesson #7 1/2.

So that's it! Gus's risk-free-no-money-out-of-your-own-pocket path to sticking it to The Man! Just plan, set realistic goals, meet those goals, diversify, save up, add four cups of patience, and have fun. And most importantly- work your ass off. It's not difficult, it's just not easy. It takes time and patience and hard work. Now it's your turn- go make a better widget. (Just don't go making another note taking application because that's what we don't need another one of.)”

The online resources set examples of how things are done in the Indie world. So it is primarily through peer influences expressed online that new developers acquire the knowledge of how to ‘be’ an Indie developer because the Indie Ethos permeates all the Indie resources that are put up there by the community. That this results in social reproduction might be illustrated best by this comment on a blog post by developer Wil Shipley [15-01-2006]:

“Trevor Fancher said...

I'm a 17 year old wanna be independent Mac developer and this post so applies to me that it is crazy. I don't know why I haven't emailed Wil before. :)

I love how guys like Wil Shipley, Gus Mueller, and Brent Simmons are always writing about their experiences with developing on the Mac and running a company. You guys are

my heros. If my future company takes off I will definitely be sending you all a big fat check. (maybe ;)"

Knowledge spillovers

The third reason mentioned in the literature why firms in a cluster are often presumed to be locally embedded is knowledge transfer. Various kinds of knowledge are supposed to be 'sticky'; they cannot easily be transferred over long distance (Malmberg & Maskell, 2001). Often, this stickiness is contributed to the tacit aspect of knowledge. This concept, originally from Michael Polanyi, underscores that knowing how to do something is, largely, acquired through experience. This implies that to a large extent we are not even aware that we use certain internalized skills to accomplish something and even if we do, it might be very difficult to transfer that knowledge in codified form; usually demonstration and imitation is required to show what you mean (Gertler, 2003, p.89). This last dimension gives one obvious reason why tacit knowledge is considered sticky: demonstration and imitation is greatly enhanced by co-presence. Tacit knowledge is an important concept to the communities of practice framework; learning to 'be' is in essence acquiring the tacit skills to tap the common sense, non-codified knowledge of a collective group of practitioners (Brown & Deguid, 2000, pp. 124-133). Thus there is a strong correlation between the possibility of tacit knowledge transfer and the distinct culture as described in the previous section. This relation is summarized by Meric Gertler (2003, p.78) as a reflexive relationship because "on the one hand, tacit knowledge is an essential complement to explicit knowledge, in the sense that it supports the acquisition and transmission of explicit knowledge through tacitly held constructs such as the rules enabling speech, reading and writing. Furthermore... the routines, customs and conventions that govern much economic behavior contain a strong tacit element". The unconscious dimension of tacit knowledge is what Nelson & Winter (1982, p.77) call a 'skill'. By becoming part of an organization one acquires the tacit skills of an organization by internalizing the routines of that organization (idem, pp. 134-136), in our case the Indie figuration (Chapter 3). Therefore this dimension of tacitness can be sustained over longer distances like the Indie culture can be sustained over longer distances as has been argued.

This does not resolve the 'demonstration and imitation' logic why tacit knowledge is sticky. Even if socialization in the community of practice is possible over a long distance, demonstration and imitation is hampered by the limited sensory possibilities when communicating over long distances. This is acknowledged and that is why Apple organizes its WWDC conference and

developer kitchens (Chapter 5) and why there is a market for personalized tutoring like the Big Nerd Ranch [<http://www.bignerdranch.com/>].

At a second glance, however, it might not be that important with software engineers as in other practices. The most important aspect of software development is coding. ‘How to’ in software engineering is actually how to write something; how to codify. This means that sending code snippets as a demonstration of how to solve a certain problem is possible. The social context of that snippet: why it is an elegant or the ‘best’ possible solution to a problem remains tacit but that does not presuppose co-location. The following testimonial on the blog of Scott Stevenson [08-06-2006] illustrates these facets:

““Scott Stevenson's in-depth knowledge of Cocoa and his clear, easy to understand instructions, and tutorials were instrumental in getting me up and running with Objective-C and Cocoa Bindings in a very short time. Scott was very responsive, and always willing to help. Scott creates clean, well-commented code, and his experience with Cocoa conventions meant that the code samples not only taught how to get something done, but the best way to do it.”

- Noah Lieberman, creator of LogTen”

The above three sections show that the specific culture and industry structure of the Indie figuration in combination with social software enable the Indies to sustain trust relations and tacit knowledge transfer over longer distances than what is often supposed in the economic geography literature. The presumed ‘local spillover’ effects in cluster theory appear to be, in this case, much more related to the social proximity of a community of practice than the actual co-location of these actors. This explains why cultural preferences seem to play such a strong role in the location of Indie companies (paragraph 3.4): in the absence of economic advantages of co-location, cultural preferences go uncontested.

4.4 The role of customers

Customers play an important role in the Mac Indie world and the qualitative differences between a ‘Mac customer’ and another kind of software customer are regarded a crucial aspect of the figuration of Indie software. Developer Wil Shipley [21-07-2005] summarizes this as:

“I love the Mac user base because they tend to be people who are into trying out new software and recommending it to each other and giving the little guy a chance.”

Generally speaking, like the developers themselves Macintosh users tend to be immersed by the cultural and functional meanings that are embedded in the Macintosh user experience; they do not pay a premium for their computer for nothing. The users appreciate the personal communication and qualitative attention that is devoted to details in functionality and the design of Indie software. This aspect of ‘chance’ that Wil Shipley mentions should not be taken lightly. Mac Indie software consumers are highly critical before they are willing to pay for their software but a significant portion does eventually pay, as blogger and developer Scott Stevenson [23-10-2006] points out in a response to changes in the UI paradigm:

“Here's what I've learned: Mac users, on the whole, have a sense of taste. If an app's experience isn't right, they simply won't use or buy it. The Mac user base is its own filter. It's not just theory, we've seen it happen.”

This implies in general that producing qualitative software is also rewarded in a monetary sense but the interviewees pointed out that their own personalities seemed to play a big role in this as well; the whole idea of consuming qualitative ‘handcrafted’ software in which you could personally communicate with the developer seems to enhance the value of the software. It is not just a product that is sold but it is also a whole experience that enhances the value of software above its utilitarian value (Pine & Gilmore, 1999). The following post of developer John Pannel on the MacSB mailing list [14-06-2005] communicates this very well:

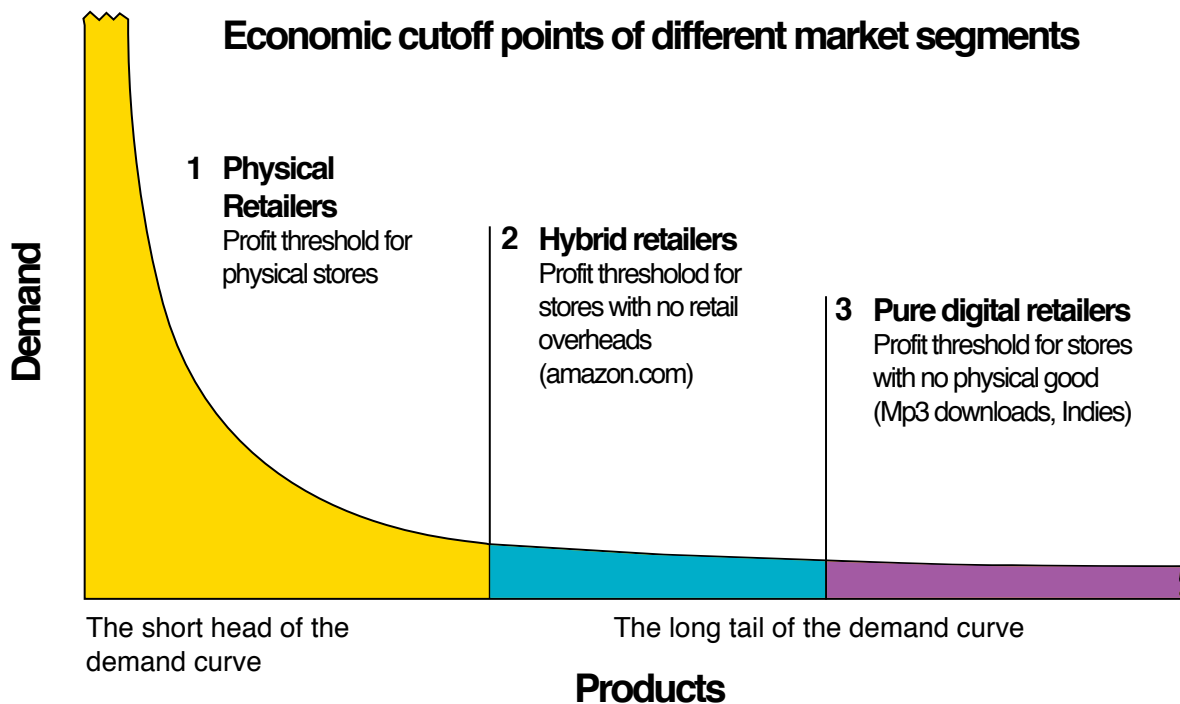
“I often get positive feedback and good mojo emails from people like the tone of my site, and who appreciate the personal touch of getting near immediate responses to email requests from a person named "john", not "support". Registered users feel more like they are involved with and supporting a personal cause than an informal corporation. Some of the most successful shareware companies of size ≥ 2 seem to still keep the personal touch rather than hiding behind "we" (i.e. Panic, Omni). "We" becomes a signpost that says "there's only one of 'us' here, otherwise 'we' would share 'our' names".”

The fact that the virtues of ‘Indieship’ are transferred to customers becomes apparent by the fact that a lot of developers get their localizations of the software for free by foreign customers. They offer to translate software in their native language out of what seemingly is an altruistic effort. This is contributed by developers to the fact that users want to ‘be part’ of the story. The clue of experience economies is that participation of the consumer is crucial to the consumption process (Pine & Gilmore, 1999, p.3) and apparently this role can play such a big part in identity formation to some (Glassner, 1975, pp. 5-26) that it is worthwhile for people to formalize that role.

4.5 The economics of Indie software; marketing

Indies generally operate under the conditions that Chris Anderson (2006) defines as 'long tail economies'. The demand curve of a commodity, in this case Macintosh software, is a power law distribution (figure 2). This implies that any market has potential revenue in niche markets, the tail of the curve, as long as there are little or no costs involved in distribution and retail (Anderson, 2006, p.24). Online distribution, in which the costs of distribution approach zero, has the potential of tapping the revenue of the tail because it is relieved of the scarcity and therefore costs of shelf space. This enables niche products to become economically feasible because they serve worldwide

Figure 2. Schematic representation of the theory of the long tail



Source: Anderson (2006)

markets independent of physical distribution constraints.

The software products of Indie companies often serve niches. Most Indie companies develop their products by 'solving their own problems': they find that the platform is lacking some functionality and write software to patch that. For example Checkout, a point of sales piece of software initially developed by Sofa BV, was conceived out of the frustration members of the development team experienced when working at a Macintosh retail store in Amsterdam. They developed a program that would fit their ideal of a good

retail experience. A beginning software company does not have the resources to build a solution that solves all the retail problems in the world; their version 1.0 had for example no network capabilities. This implies that they solved the problems of little stores that 1) own a Mac and 2) have only one cash register. On a worldwide scale there are enough small niche retail outlets that fit the description and have a good experience with the product to sustain the company. But if they all had to be served by a physical distribution and retail network it would have suffered from market failure. For these sorts of niches Internet enables a product to be economically sustainable.

To make a long tail economy work a system needs to be in place to inform potential customers about the availability for a solution to their problem. Taking away the physical transaction costs does not take away informational transaction costs. A good OS X developer has the cognitive resources to make a good product but it does not necessarily have the advertising budget to reach its customers.

This problem is solved by so called ‘aggregators’; for example a website that lists all available Mac software, pricing and where to obtain it. This allows potential customers to look for and compare useful software in a centralized place. The pioneers of OS X development used aggregator sites like [www.versiontracker.com] or [www.macupdate.com] to get the word out about their application. The problem is that as the number of developers has grown there is a lot more software vying for people’s attention and the signal-to-noise ratio of these sites has grown to the point where potential customers cease looking for useful software on these channels (Anderson, 2006, pp. 115-119). There is no quality control and therefore an aggregator loses its function of revealing the gems in a sea of less useful software. Developer Buzz Anderson [22-02-2003] explains it as follows:

“The atrocities that amateur developers (usually drunk on the newfound power bequeathed to them by their copy of RealBasic) unleash upon the world through that venue make Apple’s UI transgressions look downright venial! VersionTracker as a major source of exposure, have long bemoaned the extent to which deserving apps get lost in the sea of crap on the site. As the author of an application with a great many competitors (many of which are very, very poorly done), I have certainly been among the complainers.”

What does still function as an important aggregator resource is Apple’s OS X third party download page [<http://www.apple.com/downloads/>]. In this case Apple does the quality control but it does imply that developers are directly dependent on Apple for the listing; programs that hack the system or iPod software are not listed there.

The problem with these sorts of aggregators is that a developer is ultimately dependent on those customers browsing these databases; they are often specifically looking for something. Beside these resources developers have developed a system that is described as ‘echo chamber’ marketing. Actors within the figuration utilize their cultural and symbolic capital as ‘famous developers’ or ‘famous journalists’ endorsing other developer’s achievements by pointing towards new interesting pieces of software, for example on their blogs. This functions as a peer reviewing process in which the applications that are considered ‘good’ by the standards of the figuration (chapter 3) get known within the subset of Mac users that are ‘tuned in’ to the figuration. Mac journalist John Gruber [20-11-2006] explains it as follows:

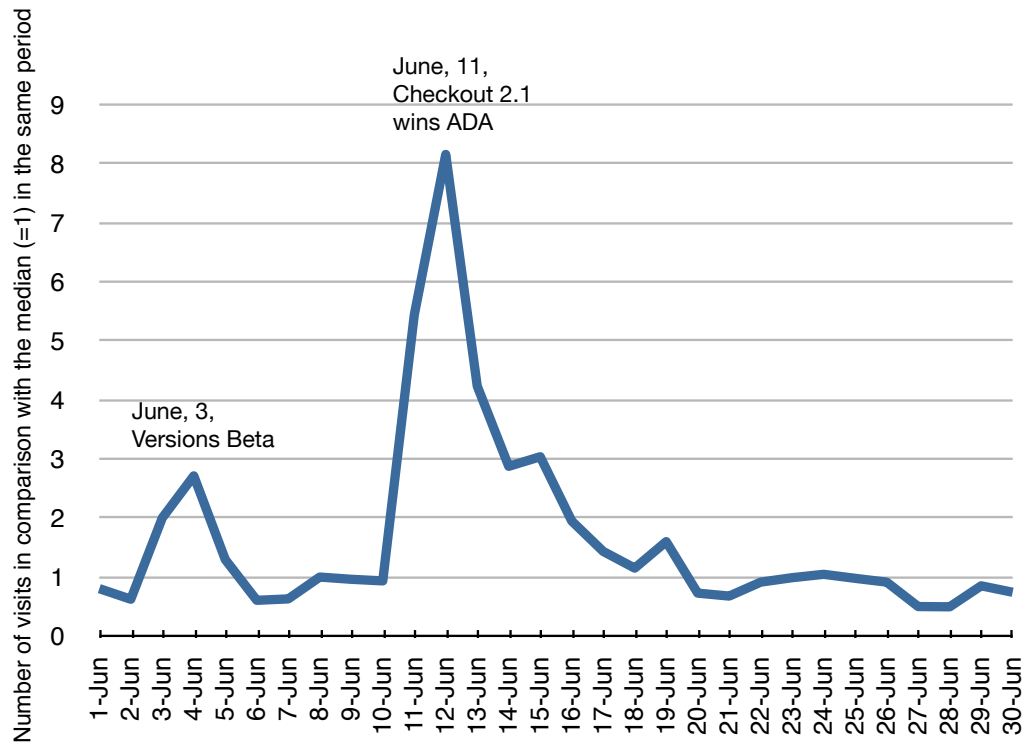
“In broad terms, there are two major steps for any new commercial app to be successful: breaking out of obscurity, and breaking into the mainstream. The good news is that if your app is well done and the idea is appealing, it won’t languish in obscurity for long. (But the contrapositive is true: if your app is languishing in obscurity, that means it isn’t well done or isn’t appealing, or both.) There does exist a middle ground between obscurity and the mainstream, where the app is well-known amongst, say, the sort of people who read *Daring Fireball*, but not known by the vast majority of Mac users who aren’t Mac enthusiasts. Sorry to say, I’m not aware of any magic formula for making that jump.² Some of it is merit-based, but some of it is just luck.”

However, as the Mac community has grown even this “grassroots support” can have enough carrying capacity for a small developer to sustain itself economically. For Example John Gruber’s RSS feed has a subscription of 50.000 ‘hard-core Mac users’ [Alperin, 17-04-2008]. Once a developer is sustaining itself it gets the resources to improve the product, forge stronger ties with other ‘tastemakers’ and in general raise the chances to Gruber’s ‘second jump’. By recommending each other’s software and thus endorsing it, developers are able to gain the critical mass of users. This is what Malcom Gladwell (2000) calls ‘tipping’. Software tipping involves getting your message past the signal-to-noise ratio of enough potential users to make it become a self-sustaining business. In Gladwell’s terminology: the actors in the figuration perform the ‘law of the few’ providing the network through which the knowledge spreads while the peer reviewing process ensures that only high quality relevant ‘sticky’ messages are sent through the network sustaining the meritocratic environment.

As an illustration of the marketing power of internet references figure 3 shows visitor statistics of the Checkout product website in June 2008. June was a very busy month for Sofa. First they released the beta version of ‘Versions’ –a Cocoa application which was widely expected in the Indie world– on June 3. The numbers in the graph have been standardized, but it shows that the

people who read the Checkout site on that day almost tripled in comparison with the median number of visitors on a day that month. These were people who followed two sequential links from the Versions website, most of these extra visitors initially came to look for something else. Then on June 11, Checkout 2.1 won an Apple Design Award (ADA) at the 2008 WWDC. This resulted in a day peak of more than eight times the median number of daily

Figure 3. Number of visits on www.checkoutapp.com between 1-07-2008 and 30-07-2008



Source: Sofa BV

visitors that month.

It should be stated that developments in the figuration are a dynamic process. As the developer community grows there is a growing variety of cultural dispositions on software quality (paragraph 3.1). This implies that the network of communities of practice has seen a variety of different ‘routines’. More and more software that becomes a hit is not necessarily fully endorsed by all of the (culturally) powerful actors in the Indie community. For example ‘hype marketing’ has become increasingly important. This involves raising expectations with the Mac press on the basis of some very visible features of your application so that the word of mouth goes around through the hype, very often even before software is released. There is a lot of discussion in the community about whether these are good developments: it tends to give a premium to eye candy versus functionality and, moreover, it circumvents the

necessity of endorsement of other developers. Another argument is that it gives potential power to intermediaries for example with hyped bundles like Macheist [Mike Lee, 07-02-2008]. On the other hand it raises the visibility of Indie software in an overall way, increasing the number of potential customers for all Indie software houses. All these developments are the result of a growing platform with increasing markets and stakes and only time will tell whether something like an 'echo chamber' niche-marketing instrument is viable for more than niche segments of a long tail economy. Subsequently, with the increase of the platform the number of potential customers has exploded, making the stakes of being a well-known developer much higher. This also means that there is pressure on being a small shop. A large part of the marketing is being personal to your users and once you have to support a base of thousands of customers it becomes increasingly difficult to be responsive. The dilemma many developers face is how to grow their company without losing this 'personal' touch that is essential to their business model.

Chapter 5: Placing the Mothership; The role of Apple Inc.

5.0 Introduction

In a recent issue of Wired magazine Leander Kahney [18-03-2008] wrote a profile of Apple Inc., describing the company as “bearing more resemblance to an old-school industrial manufacturer like General Motors than to the typical tech firm”. Kahney concludes this because Apple is a vertically integrated firm that designs both its hard- and software in-house and sells it as a closed system. Furthermore he emphasizes the fact that Apple is a very secretive company that does not allow its employees to speak about their activities, which serves the company because it keeps its innovations secret until the day of release. This picture is indeed the antithesis of the ‘typical’ Silicon Valley tech firm that supposedly reaps the advantages of the inter-company tight knit networks that makes the Valley one of the world’s important centers of innovation (Saxenian, 1994).

With such a journalistic representation one starts wondering about the role of Indies in this picture. How ‘independent’ can you be if you depend on the operating system and hardware of a vertically integrated, secretive General Motors-like corporate behemoth? The question rises whether Indies are not more than sharecroppers filling the niches of Apple’s corporate agenda.

Apart from all the normative arguments Kahney poses it is true that the discerning feature of Apple versus other software platforms is that Apple Inc. is both a software and hardware company. Except for a failed experiment in the 1990s (Linzmayr, 2004, pp. 245-257) Apple Inc. has always resisted the market pressure to split its software and hardware division and kept its software locked onto its proprietary hardware. Ironically this feature discerns the relation between Indies and Apple Inc. from the example that gave Indies their name: the music industry. In the music industry, independent record labels were always direct competitors of the major firms and the history of the two is one of competition over the means of record production and marketing that determined the cultural output in the hit charts (Hirsh, Peterson & Berger quoted in: Perrow, 1986, pp. 183-189 & Scott, 2000, p.37). The difference is that Apple Inc. might be a potential competitor with Indies on a software level but their interests converge on the hardware level: more switchers to the Apple platform means more potential customers for both the Indies and Apple’s software division. This feature complicates the power relationship

between Apple Inc. and Indies to a great extent. These relations are the subject of this chapter.

5.1 'Who' is Apple according to Indies?

To Indies Apple Inc. has two faces. On the one hand it is referred to as 'the Mothership' and they are well aware that it is Apple that provides them with their tools and is responsible for their 'ecological environment' in the first place. Moreover, Indies usually have a strong reverence towards the platform they develop on and criticism on Apple in the blogosphere has to be very nuanced to be tolerated by peer developers.

On the other hand there is the picture of 'evil' Apple. There have been incidents in the past where Apple at least was 'inspired' by (others say stole) ideas of Indies. The most infamous example of that is the 'Watson history': Watson was software which was built to complement a piece of Apple software: Sherlock. Shortly after Watson won an Apple Design Award in 2002, Apple released a new version of Sherlock that incorporated many of Watson's features [Smith, 29-07-2002]. To this day, Apple is every now and then accused of 'sherlocking' Indie software. These cases and the extent to which a developer was responsible for the copying are widely debated in the community [e.g. Gruber, 30-06-2004]. Sometimes, the developer is blamed for stepping too much into the mainstream market. If you develop software that is likely to be developed by Apple itself some consider this as your own fault. To prevent it from happening developers have their own ways of 'reading' Apple. Having more bits and pieces of knowledge and access to some technologies that are still hidden to the greater public they have a better educated guess in what Apple Inc. is going to do next. To illustrate all this ambiguity it is interesting to read the remarks of Watson developer Dan Wood [06-01-2006] when Apple seemed to 'sherlock' him again with the web-design program he developed after a short period of working at Sun Microsystems:

"I did not enjoy writing Java "Swing" (user-interface) code, nor working for a big company. So in August 2004, I went back to work with Terrence on our web-building application. Yes, I still would rather create programs in Cocoa for the Mac than any other platform.....

Yesterday, an Apple page describing their upcoming iLife '06 slipped out listing a single new application: iWeb. Here we have been moving closer to release (ironically, not feeling like there would be any competition from Apple because I have been finding bugs in Apple's WebKit that implied that Apple couldn't possibly be doing any kind of Webkit-based website-editing application).....

At least this time, I'm not expecting that Apple's offering will be a clone of our application, but the timing is certainly unfortunate. Although at times I feel like we are Apple's unpaid "Research and Development" department, the truth is that Karelia's product ideas just happen to be mainstream, like Apple's. I won't be surprised if iWeb turns out to be as lame as Sherlock was, so we can get off the tracks, let Apple pass by, then get back on the tracks and back to work.....

With that single word, on that one leaked page, our plan changed yesterday.... Because we are a small company, we are nimble — much more so than a Fortune 500 company or a bowl of petunias.”

Developers quote this nimbleness very often; do not compete with Apple and if it happens, change gears and find another niche or make your program into a ‘power user’ version of an Apple program. Overall, Indies regard their relationship with Apple as symbiotic: Apple provides them with great tools, a special kind of customer and in overall a good business environment (paragraph, 5.3) at the price that somebody gets bitten every now and then, or in the words of a respondent:

“Sometimes the little birds on the big rhino have bad things happen to them but generally speaking we are beneficial members of the same ecosystem.”

At the same time the general impression from the interviews was that Apple Inc. had become more responsive to Indies in recent years. Quite often Indies who get burned seem to get compensated, often ending up with a job at Apple and never disclosing the benefits they gain from that deal. There is a good chance that their software is further developed within Apple, the most famous example being Soundjam which eventually turned into iTunes [Sasser, 2007a]. Furthermore a lot of interviewees have the impression that, at least nowadays, if good Indie software fills up a niche that Apple considers crucial, Apple refrains from entering that niche because of the Indie offering. It seems that Apple has become more aware of the value of small developers in the last few years. Relatively more attention is being given to them compared to the big developers and it seems that Apple realizes that small developers on the platform are important assets. This correlates with the greater interaction among Apple employees and Indies as described in paragraph 5.4, but whether there is any causal relation is a matter of speculation.

5.2 Relations between Apple Inc. and Indie developers

The first remark made by most respondents when interviewed on this topic is that you have to make a distinction between ‘official’ connections with Apple and ‘unofficial’ connections with Apple; the latter being an important resource

to a lot of Indies. Formal relations with Apple come in two varieties: relations that any developer can have and a few 'privileges' that some developers have.

Formal relations

The developer tools that are required to write software on a Macintosh come free with every Mac sold. This means that the only barrier of entry someone has to take before he or she can start to become a Macintosh developer is to buy a Mac and install the tools. Official communication between developers and Apple run through Apple's ADC –Apple Developer Connection- network [<http://developer.apple.com>], the basic subscription is free but more advanced paid memberships offer for example: hardware discounts, personal help from Apple's technical support and developer seeds of new versions of for example the operating system. ADC offers mostly programming tutorials, access to mailing lists and other 'generic' troubleshooting materials.

Furthermore Apple's WWDC conference is in principle accessible to anyone who pays the entrance fees. The technical part of WWDC is largely devoted to tutor developers in new Apple technology who receive some hands-on help on coding problems. All information and seeds received on technology that is not yet officially released is officially under NDA –Non Disclosure Agreement– with Apple Inc., and seldom do developers publicly proclaim any information under NDA. Still, information under NDA is privately discussed amongst Indies. Despite some minor complaints most Indies seem to regard this official help of Apple as sufficient.

Formal privileged relations

Apart from these relations accessible to everyone there are relations with Apple that are more or less privileged. First of all there is the Apple Evangelism team. These Evangelists have an official job to 'evangelize' new technology with developers, show them the advantages of the technology and persuade them to apply it in applications. Apart from this official function they are also gatekeepers to Apple engineers for the Indies that have access to them: the evangelists introduce the Indie to the appropriate Apple engineer. Furthermore some Indies use them as a channel to get feedback on their applications from Apple [Shiple, 10-08-2007]. The Indies that have access to the Evangelists generally prefer this road over the official ADC road.

Evangelists are the most likely way for Indies to get in touch with higher up Apple executives and even then only indirectly. Access to these Evangelists is not formally restricted, but it seemed from the interviews that Indies who develop an application that is more beneficial to Apple are more likely to have good contact with the Evangelists. Evangelists are also involved in organizing so called 'developer kitchens' at the Apple campus in Cupertino. These are

invitation-only events –free of charge– that are a sort of mini-WWDC that allow the developers invited to get a head start in developing with new technology. The Evangelism team also plays an important role in organizing the Apple Design Awards (paragraph 5.5)

All this at least implies that Apple Inc. is aware of the content that developers outside of the corporation develop for the platform. It shows that some developers get a form of official preferential treatment by Apple if it's deemed important²¹. This is hardly acknowledged because Apple has a reputation of being very tough if someone violates his or her NDA. The following quotation on the iPhone SDK²² [Evans, 18-02-2008] is a rare example of the outside world that gets to hear something of these interactions. It should be noted that this event occurred during the fieldwork and that respondents were speculating on the existence of a such a 'list of approved developers', suggesting that most Indies did not know what was going on exactly and at least confirming the fact that admitting that you know something is already a violation of the NDA.

“The developer also confirmed his original loose-lipped website posting concerning its work on the iPhone to have attracted punishment from Apple, with the developer removed from the list of approved developers receiving iPhone SDK beta's. "TinyCode.com was ordered to be removed from operation by Apple, because by releasing firmware versions and stating I had possession of the firmware and SDK was apparently a violation of the Non-Disclosure Agreement I agreed to when I accepted a copy of the SDK and firmware," the developer explained in a post on the MacRuwebsite”

Informal relations

Apart from these 'formal' relations with Apple Inc., a lot of interviewees also mentioned that their relations with individual Apple employees proved to be at least as important. When Indies go to conferences or kitchens they get in touch with individual employees who give the classes. Furthermore, both participate in the social occasions surrounding these events. Apple engineers are also users of Indie software, not a very unlikely event as they are often power users of software themselves. Also, Apple employees are noted for their participation on mailing lists, solving problems posted there on an informal basis. It is not unusual that personal relationships evolve out of these encounters, as Apple employees also seem to be attracted to the ethos of Indie developers. The following blog post by Apple employee and blogger Chris

²¹ It is impossible to say at this moment to what extent this is institutionalized policy.

²² Software development kit.

Hanson [01-04-2006] emphasizes²³ that cultural attractions similar to the Indies play a role in choosing to work for the Mothership.

“I don't think anything outside my family has wielded anywhere near the influence on my life that Apple has. From the time in the third grade when I first learned to program in Logo, a huge part of my life has been spent not just working with Apple technology but steeped in Apple's — Woz and Jobs' — values.”

These relations can be very useful for Indie developers. Being able to email the actual engineer who is responsible for a bug you come across can provide a good way to circumvent the official channels for bug reporting; this saves time and usually provides a better answer. Apple employees are also subject to very detailed NDA's surrounding their work. The number one rule if you have contact with Apple employees as an Indie is that you never even take the slightest risk in disclosing to others what an Apple employee might have told you. Indies know that employees are under NDA and will most likely lose their jobs if any rumor is traced back to them. In this way Apple retains its image of secrecy to the outside world and the press but it gives opportunities for Indies to prepare themselves to what might come. It should be noted that the kind of knowledge that developers might need from Apple are no ready-made rumor site chunks. It usually involves cryptic cues which allow Indies to anticipate on changes in the development frameworks that often are specific to their application. It can be assumed that the availability of these informal contacts is probably restricted to those Indie developers who have already established a name in the Indie community.

If we analyze these relationships through an embeddedness perspective it seems that they work very much in the same way as relations amongst developers work. They are not daily interactions, are often started face-to-face or on the initiative of the employee who contacts the Indie out of enthusiasm for the Indie's software. After that, the relations are usually sustained over social software and renewed on the conferences just like the relations amongst Indies.

5.3 Functions of the relations between developers and Apple

Before we can say anything about the balance of power between Indie developers and Apple Inc. we need to define in what ways they are dependent upon each other and what the functions of the relations are.

²³ It also shows that it is not forbidden to have a blog as an Apple employee, you just have to be careful about what you write about because of the NDA.

Functions of Apple for Indie developers

The intellectual property surrounding the platform, the means to develop in Cocoa, are owned by Apple Inc.. This is the most obvious dependency of Indie developers to Apple. Apple is responsible for their developer tools, Apple decides how the operating system evolves, what sort of methods get deprecated and eventually become obsolete and which parts are supported by Apple Inc.. An Indie can use unsupported features of the programming environment –custom code– but by doing so there is always a chance that Apple Inc. changes something in the operating system update, ‘breaking’ the Indie software. Apart from this, Apple Inc. and Indies have a common stake in growth of the platform. Whenever Apple sells more Macs and converts new users to switch this will increase the potential numbers of users for Indie software. As already has been mentioned, Apple has a function for marketing for Indies as well. Developers who are featured on Apple’s site reported that a significant part of their referrals and sales originate from that site. It could be regarded as an important way to break out of the limited echo-chamber market that only reaches customers tuned in to the Indie world, thus Apple is an important aggregator. This function is likely to increase in the future since Apple has announced that sales of Indie iPhone (see epilogue) software will all run through Apple’s App store²⁴ effectually increasing this dependence [Kafasis , 24-07-2008] but potentially opening up a huge market for those who are included in this sale.

Functions of Indie developers for Apple

It should be noted that what Apple sells is in essence a ‘platform’; a substitution for other computer platforms like Windows and Linux. A particularity of these platform markets is that they are subject to network externalities (David, 1990, p.356) and there are increasing returns to adoption for users; the more people use a certain platform, the more the value of that platform as a whole increases. More products based on the platform become economically viable and the skills required to use the platform standardize (Boschma et al., 2002, pp.63-68). This implies that these sorts of markets have a tendency to lock-in: you use Windows because Windows is the standard regardless of the alternatives. That this standard can be a suboptimal solution to a problem is shown by the most famous example of such a technological lock-in: the practically very inefficient ‘QWERTY’ keyboard

²⁴ [<http://phobos.apple.com/WebObjects/MZstore.woa/wa/viewGenre?id=36&mt=8>]

(David, 1985). On a platform level²⁵ Apple faces an uphill battle in competition with Microsoft because of these network externalities. This is the point where the importance of long tail economies re-enters the story. The most important applications for a computer –office software, a web browser and, in the case of the Apple platform, software for the creative industries– are mostly produced by the ‘big three’ companies on the Apple platform: Apple, Microsoft and Adobe. These companies serve the ‘short head’ –the blockbuster products– of the market and they are the ideal companies to do so: competing in short head markets requires economic power to acquire shelf space and big advertising budgets (Anderson, 2006, pp. 147-168). But to compete on a platform level covering just the short head will not do. Contrary to some common thoughts Windows is not the standard because of its Office suite but because there all sorts of little niche solutions available on that platform. For example, a professional human geographer requires analysis software for qualitative and quantitative research methods, complex software to do networks analyses and a variety of different geographical information systems; all these have to be very feature-specific to do the detailed analyses a scientist deems necessary. Most of these software titles are often very specific solutions that are in this case not readily available on the Mac platform; essentially locking in a whole university department to Microsoft’s Windows, no matter the qualitative differences between platforms. Apple cannot serve these niches by itself; it is a publicly traded company that cannot take such risks. Furthermore it is impossible for a big company to identify all the niches that make the network externalities of a platform. This is where the total output of the Indie software companies comes in. Because of their nimbleness they fill up the economic viable niches, often identified because of a personal encountered deficit in the available software on the platform. A lot of interviewees claimed that their function to Apple was that their software had the ability to sell Macs. But in essence it is the collective output of Indie software that fills the long tail of the platform, giving Apple the means to compete with Microsoft on the platform level²⁶.

The long tail economies also work in the advantage of Indie developers in a second manner. Apple, being active in the short head of the demand curve

²⁵ Microsoft has always been one of the most important suppliers of software for the Apple platform (Cruikshank, 2006, pp.136-138); which implies that the competition between these companies has a variety of dimensions as well which are left out here for clarity reasons.

²⁶ I am aware that emulation software increases this competitive advantage for Apple even more. Because Windows is installable on an Apple computer but OS X is not installable on a PC, Apple is able to circumvent some of the logic of Network externalities. Still Apple needs to have high quality ‘native’ software for users to prefer a more expensive Apple computer over a PC (if viewed in an entirely utilitarian calculation).

ultimately relies on 'blockbuster' economies (Anderson, 2006, p. 153). Its profitability depends on massive economies of scale for a few mass produced products: computers and operating systems²⁷. To remain profitable it needs to reproduce the sales figures of those few products by convincing customers to upgrade their operating systems or seduce them into buying a new computer; in economic terms Apple needs product innovations to reset its product life-cycle to maintain high profits (Dicken, 2007, pp. 93-94). Customers need to have a demand for these new innovations that Apple supplies and Indie software plays a role in this. If Indie software adopts new features that Apple includes in new versions of its soft- or hardware, sometimes even breaking the Indie software on older systems, customers have an incentive to buy new Apple products. Therefore Apple has a stake in convincing Indie developers to adopt their new technologies.

A completely different but nevertheless important dependency on the Indie community is that the community functions as a talent pool for Apple Inc.. Apple needs Objective-C proficient engineers and people who understand Apple's design style, its interface design in particular. To this day, both skills are severely under served in professional education. An estimate by Mike Lee [10-04-2008] is that there are about 3000 Cocoa engineers on the planet. Even if he is slightly underestimating it, as I suspect, it is not a ubiquitous labor market. Apple has been noted for recruiting its engineers and designers from the Indie world and most interviewees have been at one point in their career approached by Apple Inc. with a job. The most illustrative case of this is the history of Indie company Delicious Monster which at one point had 100% of its ex-employees hired by Apple [Shipley, 18-12-2007].

At the same time career moves in the opposite direction are increasing as well, as the following blog post of Indie developer and former Apple employee Daniel Jalkut [10-01-2008]²⁸ shows.

"Something is changing. In the past few years, more and more of my developer friends have started talking about "going indie." That is, going out on their own to develop, market, support, and profit from their own software. Many years ago, while I was working at Apple, the notion of striking out on one's own was not even on the table for most developers I knew.....

Going indie provides the opportunity to be a very big fish in a very, very small pond. In fact, the only way to extend the metaphor successfully is that as a new indie entrepreneur,

²⁷ In the case of Apple, iPods and iPhones also make up an important segment of the economies of scale.

²⁸ Which is also the blog post that inspired the name of this thesis,

you're going out on a limb, hoping that you'll find a pond even big enough to support yourself and your family. You might roll around in muddy shallows from time to time, gasping for air and hoping for rain, but in any case, you're the big fish and at least you run the show.

Within the past week, both Peter Bierman and Jens Alfke, two long-time Apple employees and passionate Mac developers, have decided to become big fish in their own ponds. They've gone indie.

What? Did these guys get together and coordinate this? I don't think so. The fever is running rampant through the Mac community, and people are catching it everywhere, including inside Apple. I know at least two other people personally, who work at cushy jobs in multi-billion-dollar companies, who are also threatening (read: building up determination) to strike out on their own.....”

Apple is a very demanding employer and Apple employees seem to see working for the Mothership as a tradeoff: you get to work on exciting technical innovations, and while doing you learn a lot about Cocoa programming but you are not allowed to talk about it. Furthermore Apple has increasingly anonymized its employees; they do not get public credit for their achievements and giving a talk at WWDC is probably the best chance of letting the world know of your achievements. The lack of individuality is credited by former Apple employees [e.g. Alfke, 10-01-2008, Anderson 19-04-2007] as an important reason why they went Indie. These kinds of non-monetary arguments are supposedly a trend in the choices regarding career planning within the creative class (Florida, 2002, pp. 102-115) and therefore Daniel Jalkut could be right in claiming that going Indie is becoming a more frequent phenomenon. The result of this increased job mobility between Apple Inc. and the Indie world at least implies that the number of personal linkages (paragraph 5.2) is only likely to increase and likewise that the awareness between Apple and the Indie world will increase.

A probable consequence of this is that the more frequent linkages between the Indie community and Apple also 'inspire' Apple to adopt Indie inventions. Within the Indie community a number of rumors circulate where Apple actually incorporated user interface details from Indie software into the platform, the toolbar of Panic's Coda application being the most famous example [Sasser, 10-09-2007]. Apple's R&D is very decentralized. It is subdivided in small project teams who have a far greater autonomy in their design than often is assumed [Rentzsch, 10-08-2007]. That autonomy gives likelihood to these rumors because apparently individuals within Apple that are inspired by Indie software have enough internal leverage to let them make use of that inspiration. These knowledge spillovers also work in two ways,

former Apple employees going Indie bring routines from within Apple to the Indie figuration.

5.4 Evaluating the balance of power between Indies and Apple Inc.

Economic reductionism that is too strong could make someone easily misinterpret the power relations between an Indie and Apple Inc.. Technically Indies, if they want to remain an independent programmer in Cocoa/Objective-C, are subject to Apple's will. In a situation where the only political leverage is the 'exit' option, such a relation would be labeled very asymmetrical and in favor of Apple (Ruigrok & van Tulder, 1995, p.74). However, the picture changes if we look at the kind of dependence Apple has vis-a-vis Indies. Apple is dependent on Indies in an economic sense: they need Indie software to keep their platform competitive. On the contrary, the dependence of Indies versus Apple Inc. is cultural. Indies are nimble companies with relatively low overhead costs. This means that they can switch far more easily to a different platform than a big company would if they disapprove of the position they are in; they are often into Apple for affective reasons in the first place. Furthermore Apple Inc. needs to continuously convince new developers to develop for the Apple platform whose potential market is, at least at first glance, smaller than its foremost competitor, Windows.

Therefore Apple has to utilize 'softpower'; power based on affective and cultural properties, instead of 'hardpower'; power based on economic or formal political properties (Nye, 2008), to 'inspire' Indie developers to make them do it Apple's way.

The best illustration for this softpower is probably the Apple Design Awards (ADA) event [<http://developer.apple.com/wwdc/ada/index.html>]. These are awards that Apple hands out to developers at its worldwide developer conference (WWDC) each year. The Apple Design Awards are held in high esteem in the developer community; the interviews pointed out that winning an ADA greatly enhances the amount of peer recognition an Indie developer receives. Indies discuss what sort of applications win an ADA as a signal of what gets 'the approval of the Mothership' and discussions on what wins and what should have won play a big role in the informal conversations surrounding the WWDC [e.g. Lee, 16-06-2008]. Apart from that, ADAs are used to 'read' the opinion of Apple Inc. in regard to what kind of applications Apple would like to see coming out of the Indie community. It is Apple that sets the criteria and selects the winners for the Design Awards and adopting new Apple technology in an application is usually a part of those criteria. The

interviews suggested that the kind of applications winning an award have changed during the years: they used to be handed out for applications that made good use of the human interface guidelines but nowadays a lot more ‘flashy’ and experimental design applications²⁹ win the awards.

Another important example of softpower application is the policy regarding the developer tools. Very consequently Apple has lowered the barrier of entry by making tools readily available and free. Developing software for a platform used to be an exclusive business in which developer tools and support had to be bought and they have now all become free. Jonathan ‘Wolf’ Rentzsch [10-08-2007] estimates that the startup costs for starting Mac programming, besides owning a computer, used to be about \$1500 and had variable costs between \$300 and \$3000 a year. Those barriers to entry all have been lifted and the cost is near zero.

In addition to that Apple has put great effort in increasingly simplifying programming for OS X. As chapter two has shown the shift from classic Apple technology to OS X already gave huge improvements to developers productivity wise, but Apple has only increased the productivity of Cocoa programming with every release. With ‘productive’ is understood that it became more high-level. Each subsequent release of OS X offered more actions pre-programmed into the system, meaning that a developer has less work to get things done³⁰. This makes developing for OS X an increasingly pleasant experience with every new OS X release. With every new improvement of the developer tools it becomes easier to do complex things and more deficiencies of the programming environment are ‘patched’. This means that developers have advantages of upgrading as well, which they have to leverage against the stakes of users. This results in developers eager to update and not always being able to wait until all the users do so, as the following blog post [Anderson, 25-06-2003] illustrates:

“Wowee—I sure wish I wasn’t bound by this NDA, because I’ve seen a lot of things here at WWDC that make me very happy! This morning I attended the “Cocoa Update” session, which has really been the first session to delve deeply into Panther’s <the nickname for OS 10.3> Cocoa changes, and it’s becoming pretty clear that a Panther-only release of PodWorks is all but unavoidable. There were just too many new things that made me say “Oh, I need that!” or “Oh, that would make life so much easier” or “That gives me an idea for a whole new feature!” I was like a kid in a candy store!”

²⁹ This does not include opinions on the 2008 Design Awards, and there have been some remarks in the community that the emphasis has shifted again.

³⁰ For example: Cocoa bindings in 10.3, CoreData in 10.4, Core animation in 10.5, and forthcoming, Grand Central in 10.6 [<http://apple.com/macosx/snowleopard>]

To Wil Shipley [24-07-2005] this is one of the more distinguishing factors between Apple and Windows. He points out that that Microsoft Windows has changed its developer environment every few years, which makes investing in platform-specific knowledge a risky enterprise. He also points out that there are a lot of ways to program in Windows, but none of them work like he thinks it should. In his words: “In Cocoa you have time to innovate, in Windows you’ll spend your time getting buttons to draw correctly”.

I do not have the authority to make any claims regarding the factuality of that judgment but it should be noted that the overall impression is that the Apple developer community has seen a steady inflow of young developers and ‘switchers’ from the Windows platform recently; WWDC was sold out in 2008 for the first time in its existence³¹. Without having the figures to back it up, the suggestion that lower barriers of entry and the pleasant ‘easy’ way of doing things in the Apple programming environment gives some credit to a logical correlation between the two. Fact is that the number of Indies is increasing. Apple benefits from more developers on the platform producing quality software and therefore it has a strong incentive to keep improving the programming environment.

5.5 Inserting Apple into the figuration by an application of Michael Porter’s ‘diamond’

According to Porter (1998a, p.115) competitiveness ultimately depends on the capacity of an industry to innovate and upgrade³². Serving the market with innovative products and getting the market to upgrade to new technology ultimately allows an actor to ask for monopoly profits, since its products are a unique asset of the actor (Nelson & Winter, 1982, p.266). This has always been Apple’s corporate strategy, even in the darkest days in the 1990s as is put by Steve Jobs biographer Alan Deutschman (2000, p.232):

“Apple’s stewards were trying to coast on a reputation that no longer matched up to reality, and arrogantly, they were demanding a premium price for its products that no longer were much better, or different, from what hundreds of other PC makers were offering”

The dark days are over for Apple and Apple is once again honored for its innovative capabilities as the distinctive topping or near topping on world

³¹ Although this probably also has to do with the introduction of the iPhone SDK.

³² In this case, Porter makes this statement on the competitiveness of nations but the logic can as easily be applied to clusters, as Porter has done himself.

rankings on innovativeness show [Digg, 10-12-2007]; a status which largely is credited to Apple's CEO: Steve Jobs.

Innovation has two sides. Economist and innovation theorist Joseph Schumpeter (as quoted in Cooke & Morgan, 1998, pp.11-12) started out with the assumption that innovation was driven by visionary entrepreneurs but eventually ended up acknowledging that innovation is to a great extent a collective social endeavor (idem, p.33). This does not mean that there is any reason to doubt Steve Jobs' capacities as a CEO but it does invite to focus on how the Indie figuration plays a role in this platform wide innovation process.

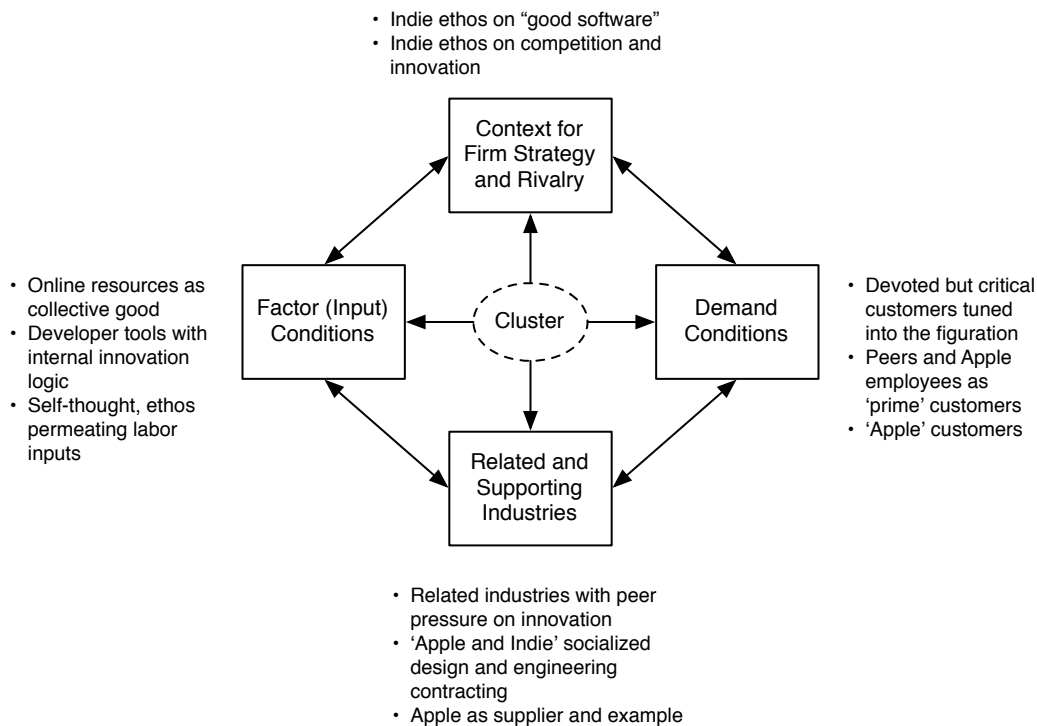
The collectiveness of an innovation process ultimately relies on the diffusion of inventions. It does not suffice to invent a certain technology or process that has the potential to innovate; ultimately all the involved actors have to implement a certain invention before it can reap its market potential and is called an innovation (Nelson & Winter, 1982, pp. 265-272). Innovation by its very nature involves receptiveness towards change; a will to continuously challenge existing routines in favor of a better possibility. This means that it requires two properties: the knowledge of other possibilities and the will to apply them even if it is against a short-term rationale (Porter, 1998a, p.165). However, they should be regarded as properties of a system level (Hotz-Hart, 2000). How do the mechanisms work that support such a 'change' dynamic on the level of the whole Indie figuration that supports Apple to diffuse its inventions?

To assess such a system heuristically applying Porters cluster model to the Indie figuration yields interesting insights. Porter's theory assumes that four interrelated facets can measure the competitiveness of an industry: factor input conditions, context for firm strategy and rivalry, demand conditions and related and supporting industries. The cluster, in our case a figuration of *socially* proximate actors, could be considered a manifestation of the interaction between the 4 factors (Porter, 2000, p.258). Figure 4 provides a schematic overview in Porter's style of how we can interpret these facets in the Indie figuration.

In chapters three and four was explained how the Indie figuration sustains a dynamic but solid ethos amongst its members. It consists of an informal set of rules, which involves both an idea that software should be 'good' and 'innovative' and moreover it explains an ethos on competitiveness, which holds competition on quality in high esteem and competition on price in low

esteem. These informal institutions can be considered the context for firm strategy and rivalry (Porter, 1998a, pp. 178-182) in the ‘diamond’ model.

Figure 4. Schematic representation of the application of Porter’s ‘diamond’ on the Indie figuration



Sources: Porter (1998a), fieldwork

Porter (1998a, p.175) considers the buyers of a product important in enhancing competitiveness if the buyers relatively close to the producer are more critical of the product than the average buyers on the mass worldwide market. This applies to Apple and the Indie figuration in multiple ways. Firstly it should be noted that the prime buyers of Indie software are a select group who are critically ‘tuned in’ to what is produced by Indie companies. Often peers are the first buyers actively giving feedback to the producer. These peers are socialized in the dynamic ethos of the Indie community and evaluate their peers’ software accordingly. Even within Apple Inc. -from the figuration perspective an upstream supplier– there is a significant number of users tuned in with what the Indies produce, meaning that these facets are coupled with each other. Moreover, even the more distant ‘Apple customer’ is regarded a more critical customer than an average software customer. This means that even if Apple grows further beyond its established customer base there remains a critical mass of innovation appreciating buyers.

The two factor inputs that are particularly important when it comes to Cocoa programming are the developer tools and the human capital input. Porter argues (1998a, pp.172-174) that a scarcity in these endowments can foster innovativeness. This is especially true in the labor input. In the absence of formal education channels most Cocoa engineers are self-thought, based on a strong motivation. Furthermore, the resources for learning Cocoa offer a high probability that the engineers become socialized in the culture and ethos that surrounds Mac programming. The other crucial input, the developer tools, has its own structural innovation logic; improving the developer tools with new functionality is crucial for Apple to get Indies on board. If the Indies choose to use new features and technologies introduced in the latest OS X release will eventually require end-users to upgrade their system. The last important factor inputs are Apple's and the community's collective resources. These also compete to certain extent; Apple has an incentive to provide good education to developers because there are alternatives outside Apple's control and at the disposal of Indies.

The related and supporting industries are important to the degree in which they are by themselves internationally competitive (Porter, 1998a, p.176). The supporting industries are the suppliers of input for Indie software, in our case Apple and to a certain extent contracting workers or designers which are in their own degree all tuned in to innovative competition. Designers working for Indies are for example the same designers that could vie for a job at Apple, which is not an unusual career path. The related industries are the competitors that according to Porter should be creating pressure on companies to innovate and improve (Porter, 1998a. p.181). We have seen in chapter three that peer pressure plays a significant role in stimulating Indie companies to innovate.

As these examples show the linkages between the factors play an important role in explaining why the factors work as a competitive advantage. They mutually improve each other and even put a pressure on Apple to stay innovative regardless of its charismatic leader. In that sense Apple is less of a vertically integrated behemoth than the description in the beginning of this chapter suggests. Ultimately, applying the cluster model to the Indie figuration sheds light on the functional relationships that help sustain the cluster. Regardless of human motivation the figuration as a whole has a collective function that helps it sustain itself, of which the interdependencies between Apple and Indies provide an indispensable part.

Conclusions

In the introduction was stated that the social world of a virtual community of entrepreneurs is best to be understood by researching it through a network approach that was distilled in the following research question:

- *How do independent software developers for the Macintosh OS X platform interact with each other and with Apple Inc.?*
- *What kind of functions do the social relations have for the social and economical figuration of both Apple Inc. and the independent developer community?*
- *How are these interdependencies socially and spatially embedded?*

This was elaborated upon in chapter one where it was explained that such a community could be described as a figuration that allowed to make statements on both the micro and macro level regarding economical and cultural properties without resorting into a form of determinism.

This approach yields results in the sense that it is able to explain that a particular path-dependent trajectory led to the genesis of a developer community of interdependent actors. This community is primarily based on cultural properties: a common culture and a shared sense of identity influence the kind of software Indies make and the sort of competition they allow. The figuration approach also explains how this ethos evolves through time and how this is connected to the power positions of actors within the figuration.

The whole community sustains itself by a combination of interaction on conferences and online interactions and by those means they are an example of how supposedly local spillover effects can go virtual, including the socialization of new members of the community and collective institutions. The combination of these interactions and their shared ethos constitutes the way in which these relations are embedded. The Internet allows the activities of Indies to be economically viable because high costs in distribution and retail would otherwise cause market failure. The community assists by functioning as a marketing instrument and supplying role models to optimally achieve that.

When relations with Apple Inc. are inserted in the figuration this enriches the perspective. Indies and Apple have a common goal where success of the platform is concerned and possess complementary competences to achieve

that goal. Apple needs the Indies' nimbleness to raise the scope of the platform and in return provides the Indies with the means to do so. In that sense the relationship is symbiotic and the power relationship between Apple and Indies is much more egalitarian than the size of the respective companies would predict due to the two different kinds of power: softpower versus hardpower. These relations with Apple Inc. are furthermore lubricated by the same ethos that in part originates from Apple. At this moment the ethos and culture seem increasingly reciprocal as interactions and flows of people between Apple Inc. and the Indie community increase. To assess the functionality of this symbiotic relationship the Indie figuration was heuristically framed in Porter's theories on competitiveness and clusters. The result of this assessment is that the Indie community, in a socially proximate cluster kind of way, provides an interface through which the whole figuration helps in sustaining the platform's competitiveness when approached from an evolutionary economic perspective.

The results of the research show how relatively autonomous economic and cultural logics reinforce each other and in this case allow all the participating actors to be able to fulfill their personal or corporate motivations, while supporting a collective endeavor regardless of whether they are aware of it. Furthermore the results give support to the logic of cluster theory without principally adhering to the spatial logic that is often presupposed. The results should be regarded as a starting point for further research: a theory with a closed internal logic that requires additional and in particular quantitative testing.

Epilogue

It can be especially difficult when doing social scientific inquiry in a sector that is continuously innovating and changing to keep up with the pace of developments. Researching a community that, in its current form, started in 2001 is a short history for a discipline which is more used to look at decades than years, let alone months; things change faster than one can study them.

On October 17, 2007, Apple announced that they would release a public iPhone SDK [Winer, 17-10-2007], allowing third party developers to develop software for the device and on March 6, 2008, just 3 weeks after returning from fieldwork the SDK was released to developers [Krazit, 6-03-2008]. The iPhone is particularly interesting to Mac Indies, software for it can be developed in Cocoa/Objective-C. Furthermore Apple announced that it would only allow third party applications to be installed on the iPhone through their online App store. Despite being locked into Apple this does mean that, for the first time, Apple will actively sell Indie software through their sales channels; something which is widely believed to positively enhance the marketing of Indie software to new types of users. Lastly, for the first time venture capitalists took notice and with the SDK a 100 million investment initiative in iPhone applications was announced [kpcp.com, undated].

This implies that with the announcement, the labor market for Cocoa/Objective-C engineers has at least doubled, evident from all the initiatives to develop software for the new platform. What will happen to an Indie community which suddenly finds itself in the center of attention of the whole tech industry? How will the Indie culture evolve under the potential of much bigger profits? Will a peer review system for iPhone applications evolve because of the sheer amount of offerings on the App store? How will the community deal with so many new 'outsiders' attracted to the iPhone that have little notion of the platform's history? The follow-up research has just received its initial questions.

Appendix A: Defining culture and identity

Culture is a very ambiguous and contested concept. Because of the ambiguity it is important to give a clear definition when it is inserted in an explanatory framework. Especially when it is given such an important role as in this thesis.

Insights from symbolic interactionism teach us that social behaviour, at least to a very large extent, is based on the meanings that events, objects, or other social behaviour have for people (Blumer, 1969, p.12). This implies that social action is not directly derived from the way “things are” but from the way “things are perceived”; our definition of the situation. How we define the situation is informed by what we learned throughout our experiences in the past.

In social sciences the acquiring of all learned behaviour, both conscious and unconscious is conceptualised in the notion of “socialization”. This means that all the characteristics a person acquires socially and not biologically have been socialised. When a child socialises it learns the customs of its surroundings: parents, peers and other social institutions like schools and media instruct the new member of the society what is proper behaviour socially (Giddens, 2006, pp. 163-169). A person learns a language, how to communicate, and with it what things mean. Most of this learning happens unconsciously, one does not question the “naturalness” of their surroundings; the knowledge is not “reflexive”. This set of “common sense” cultural assumptions unselfconsciously held seems to people a natural transparent view of the world. People do not trouble them with what is “real” as long as everything around them makes sense to them (Derné, 1994, pp. 268-271). Not all knowledge is just common sense and people can become aware of their roles; become reflexive (Cragg, 1996, p. 159) especially when learning actively.

Culture then, is a system of commonly shared meanings acquired through socialization; in the words of Max Weber as paraphrased by Clifford Geertz (1993, p.5) “as the web of significance that man himself has spun”. Geertz continues to explain that therefore cultural analysis is “no experimental science in search of law but an interpretative one in search of meaning”. The key word here is understanding. As a cultural scientist you try to understand how other people define their situation through deciphering meanings.

It implies that culture is not primarily about norms and values but of the meanings, most of the time lying in the unreflexive common sense, that inform people’s definition of the situation (Chabal & Daloz, 2006, p.86). Because everyone’s socialization is factually different, every individual will

have a slightly different cultural background although the sheer commonalities within the socialization process explain why people with similar backgrounds share so many cultural traits. Therefore, there does not exist something as a reified 'superorganical' culture (idem, p.53). It is a concept to understand the commonalities in socialization between people which can never be deterministically applied from a group abstraction to an individual; that would be applying the ecological fallacy (Mitchell, 2000, p. 74).

Related to culture is the notion of identity. Contrary to culture, identity is a reflexive concept. Giddens (1991, p.76) defines identity as: "autobiography – particularly in the broad sense of an interpretive self history produced by the individual concerned, whether written down or not– is actually the core of self-identity in modern social life". People identify according to that part of their socialization (as understood above) that they deem an important part of the narrative of the self; but can be influenced by others as well. Identity often rallies around certain 'markers': language, ethnicity, class, profession on which people define social in-groups (Allport, 1954, pp. 31-35). Especially identity markers based on roles instead of markers based on descent have gained in importance throughout the 20th and 21st centuries (Glassner, 1975, pp. 5-26). But also, increasingly, consumption preferences have been observed as important identity markers (Goss, 2005).

Appendix B: Questionnaire

The semi-structured interviews in the US were conducted with a basic questionnaire which was customized to the interviewees. If their web history revealed interesting questions on a certain topic the questionnaire was changed accordingly. Depending on the time available in the interviews some questions and topics were prioritized above others depending on the background of the interviewee. Furthermore, explicit room was made to elaborate on certain topics by asking new follow-up questions. New fruitful insights were added to the basic questionnaire which were used in the subsequent interviews. The questionnaire revealed here is the generic questionnaire that was at the basis of all the interviews at the end of the fieldwork period. Each main question is spelled out. These were asked literally and meant to invoke an as open answer as possible. The sub questions are paraphrased. Depending on the initial answer they were used as building blocks for follow-up questions. Some interviews were group interviews. In those cases the biographical part was done for each interviewee individually while the other topics were open for group discussion. The questionnaire is divided into four topics, five if it was a big company, on the following subjects: biographical data, horizontal interdependencies, vertical interdependencies, the role of space and place and, with bigger companies, the inter-company interaction.

Topic 1: Biographical questions

- When and why did you decide to start developing and selling Mac software on your own?

- Could you tell us about how you became a developer?
 - What is your formal education?
 - How is your education related to your work as a developer?
 - What kind of resources do you use to improve this?
 - How did you learn Cocoa? (what means did you use?)

- You have been around since the NeXT days, how did you come in contact with NeXT computing and Objective-C?

- Could you explain a bit how you got involved with Apple computers and how this relates to the decision to focus on Cocoa programming?

- About role of Carbon. You have been around the maturation of the cocoa platform. Could you reflect on the relation between Carbon and Cocoa. Back in the early days of OS X and now?
 - How does it compare to the debates surrounding the switch from OS 9 to OS X?
- From your website, it appears that you have a background in the Arts. Could you tell us how this plays a role in your programming?
- What is the role of your work in your life?

Topic 2: Horizontal interdependencies (or interaction between developers).

- Could you tell us something about your relations towards and with other indie developers?
 - Importance of Blogging
 - Importance of Chat clients (iChat, Skype)
 - Importance of tutorial sites/Wiki sites.
 - Importance of Networking Sites (Linked in, Twitter)
 - Mailing lists
 - Do you ever share code over the web?
 - Do these relations differ between Mac and non-Mac developers?
- How important is it to actually meet other developers in person?
 - Frequency
 - In relation to trusting each other, is an online reputation enough?
 - How important is it that other developers live close by?
 - Importance of user groups, Cocoaheads/Xcoders meetings
 - Big conferences like WWDC, Macworld, C4
- For what kind of reasons is it important to know other developers?
 - To solve technical problems
 - To exchange business information
 - Rumor sharing
 - Friendship
- In what ways has blogging been important to you and your work? And did this change throughout the years?
 - Importance for customers
 - Importance for peers
 - Importance for reputation
 - Importance to get noticed by Apple

- Could you reflect on the concept of an ‘Apple’ culture, in general and related to developers specifically?
- Could you reflect on the idea that the Indie developer world is a community?
- You have been around the Indie world for a relatively long time. Could you tell us how the independent developer community changed during that time?
- You have in active on different platforms. In what sense does the Indie Mac world differ form other domains of software engineering you are active in?
- What does it take for you to trust a business partner or another developer?
- From my web preparation it seems to you have contributed a lot of technical (coding) knowledge to the indie community. In what sense are those contributions important?
- How did you get in the position that magazines review your app, and what has the result of the reviews been?
- Could you tell a bit how you promote your software?
- In what sense is it important that you do retail?
- In what sense does interaction with clients matter?
- There seems to be an ongoing discussion between Mac developers on where and how to apply the Apple human interface guidelines, what are your thoughts on that?
 - In what ways do you sense any pressure in the community to follow the guidelines?
- Can you reflect on the discussion between the so-called ‘delicious generation’ and ‘old school’ Mac developers?
- Are controversies like Macheist changing the nature of the Mac community?
- A big part of the debate seems to be concerned about “what an independent application should be”. What are your thoughts about that?

- Function versus form.
- From researching the community, I get the impression that there is some sort of special ethos to being a Mac developer. Could you reflect on that?

Topic 3: Vertical interdependencies (Interactions between developers and Apple Inc.)

- What does an Apple design award mean to you?
- Do you have, and how are, your relations with employees/executives from Apple Inc.?
- Could you tell us something about your experiences with Apple developer relations?
- Could you tell us something about your experience with the Apple evangelism team?
- Could you tell us something about your experience with Apple's developer technical support team?
- Do you get the feeling that your work is "noticed" by Apple?
- How do Apple's new releases affect you?
 - How is the communication with Apple about new releases?
 - Could you talk a bit about the choice of which version of OS X to program for?
- Could you reflect on the influence of Apple's HIG on your work?
- What is the role of the big 'Apple' events like WWDC and Macworld for your relations with Apple Inc. in particular?
 - Could you reflect on how these conferences changed throughout the years?
- Could you tell about your ideas on in what ways Indie developers are important to Apple?
- Could you tell us something about your experiences with Apple dedication towards Indie developers in general? And how these evolved throughout the years?

- Do you have the idea that Apple is lowering their barrier of entry to new developers?

Topic 4: The role of space/place

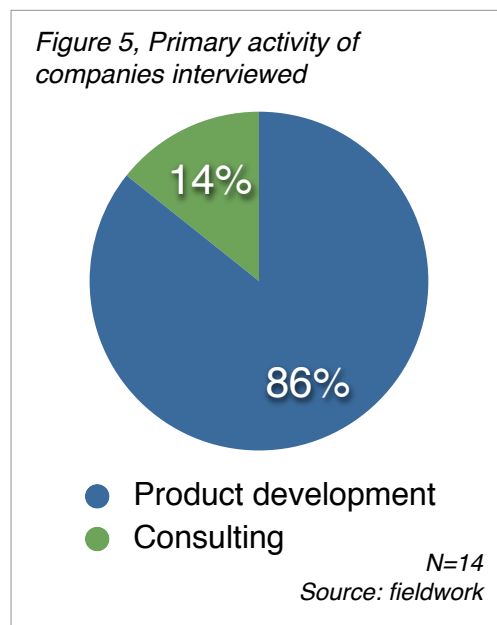
- What is your impression on the Cocoa 'world' in your place of residence?
- In what degree are there local Mac resources in local area, like user-groups or developer meetings?
 - In what ways do they matter to your work?
 - How much does the local cultural atmosphere play a role in your work?
 - How often do you see other developers face-to-face in your place of residence?
- In what ways is it important both for yourself and your work to live in an urban area? (and if you are not in an urban area, do you miss it?)
- How important is it to be close to Cupertino, and the San Francisco Bay area tech world?

Topic 5: Intra-company collaboration

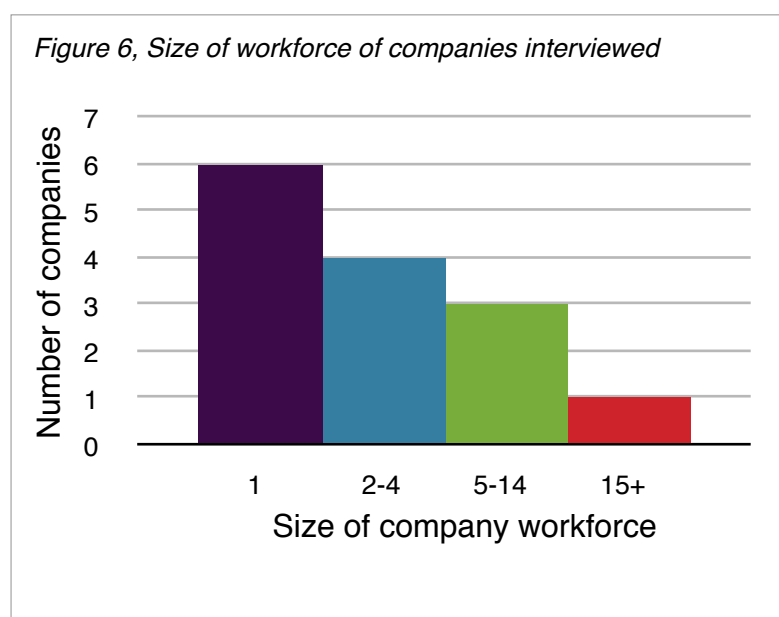
- Could you tell about the way you work together in the development process?
- How does the division of labor work within the company?
 - How did the company mature, also in regard to the division of labor?
- (For geographically dispersed companies) How often do you meet face to face, and what is the importance of those meetings?

Appendix C: Composition of the qualitative interviewee selection

The goal of the qualitative phase was to assess the Indie community from as many different perspectives as possible. Therefore different types of companies were selected. Fifteen interviews were conducted at fourteen different companies. Figure 5 shows the primary activity of these companies. The consulting firms interviewed were active in design and/or engineering contracting work. A second variable on which the companies were selected was the company size, as is shown in figure 6. We attempted to keep a representative balance between the various company sizes that exist in the



Indie world. Figure 7 shows the deviation of the number of applications that the product developing Interviewees have. This indicates whether they have a diversified or a focussed company strategy. Subsequently, the non-one man shop companies can be subdivided into centralized companies, with a central office, and decentralized companies who primarily utilize teleworking. The division is shown in figure 8. Figure 9 shows the technological background of the individual respondents. Effort has been made to highlight the NeXT and 'old'



Apple perspectives while ensuring that enough 'young' developers who started after OS X had come out were interviewed as well. The last selection criterium is the different platforms the interviewees professionally develop for, the percentages are shown in figure 10.

Figure 7, companies by number of products.

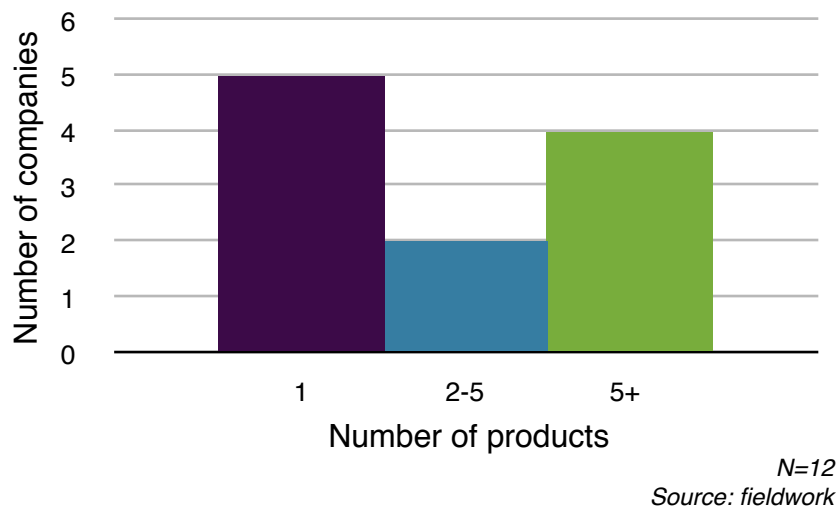
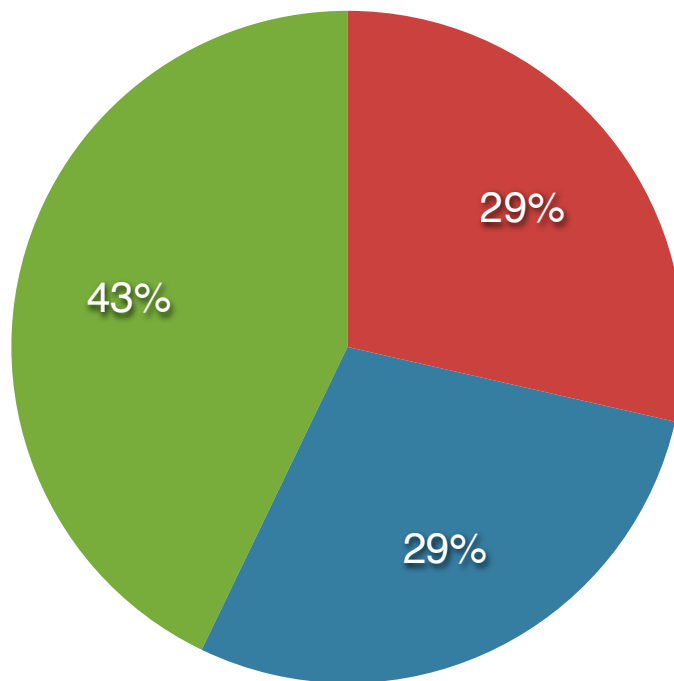


Figure 8, centralization of companies



- Centralized
- Decentralized
- Not Applicable (one person)

N=14
Source: fieldwork

Figure 9, Technological background of respondents

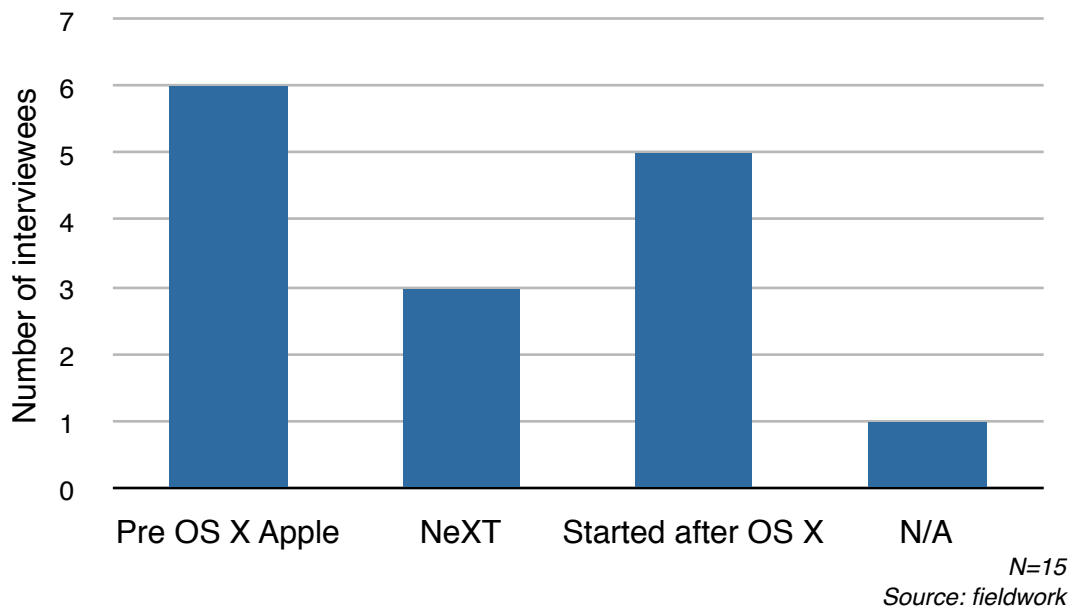
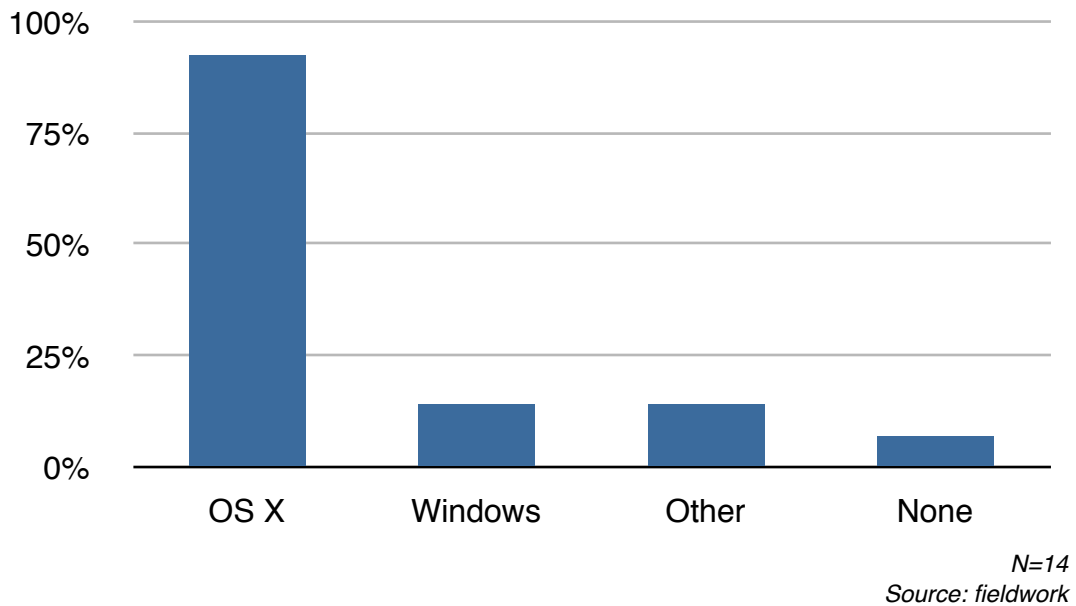


Figure 10, Platforms that interviewed companies develop for



Appendix D: Apple/NeXT/OS X history

Timeline

Sources: Linzmayer 2004, Wikipedia

1977

- Apple founded
- Apple II introduced

1979

- First of two Xerox PARC visits

1981

- IBM introduces IBM-PC

1983

- Lisa introduced

1984

- Macintosh introduced

1985

- Steve Jobs resigns from Apple
- Steve Jobs founds NeXT
- Microsoft releases Windows 1.01 and licenses Mac technology

1988

- NeXT computer (cube) introduced

1989

- NeXT releases NEXTSTEP 1.0

1990

- Microsoft releases Windows 3.0
- Tim Berners-Lee develops the first Web browser on a NeXT computer

1991

- Apple announces system 7.0 (operating system)

1992

- NeXT releases NEXTSTEP 3.0

1993

- NeXT discontinues hardware sales
- NEXTSTEP for Intel processors released
- OpenStep released

1994

- First Power Macs (with IBM technology) ship
- Apple announces Copland operating system

1995

- Microsoft releases Windows '95
- Sun and Apple discuss merger
- Apple releases Copland beta
- NeXT reports first annual profit

1996

- Apple reports \$740 million loss
- Apple Inc. announces purchase of NeXT
- Steve jobs returns as advisor at Apple Inc.

1997

- Apple Inc. acquires NeXT
- Steve Jobs named Apple Inc's interim CEO
- Apple releases OS 8.0 with pieces of Copland

1998

- Rhapsody released

1999

- Apple releases OS 9

2000

- Steve Jobs becomes Apple CEO
- Dot com crash
- Microsoft releases Windows 2000
- Mac OS X public beta released
- Apple buys Soundjam, Indie app. which would evolve into iTunes
- Paypal introduced

2001

- OS X 10.0 (Cheetah) released
- Apple Inc. releases developer tools for free
- OS X 10.1 (Puma) released
- First generation iPod introduced
- Microsoft releases Windows XP

2002

- OS X 10.2 (Jaguar) released
- Mac OS X named default Mac operating system, OS 9 discontinued.
- The 'Watson history'

2003

- OS X 10.3 (Panther) released
- Windows version of iTunes released
- Last MacHack conference

2005

- OS X 10.4 (Tiger) released
- Apple Inc. announces switch to Intel processors
- Apple Inc. abandons Cocoa-Java bridge

2006

- First C4 conference
- First Intel Macs ship
- Twitter launched

2007

- Windows Vista retail version released
- OS X 10.5 (Leopard) released
- iPhone released
- iPhone SDK announced
- 64 bit Carbon support dropped

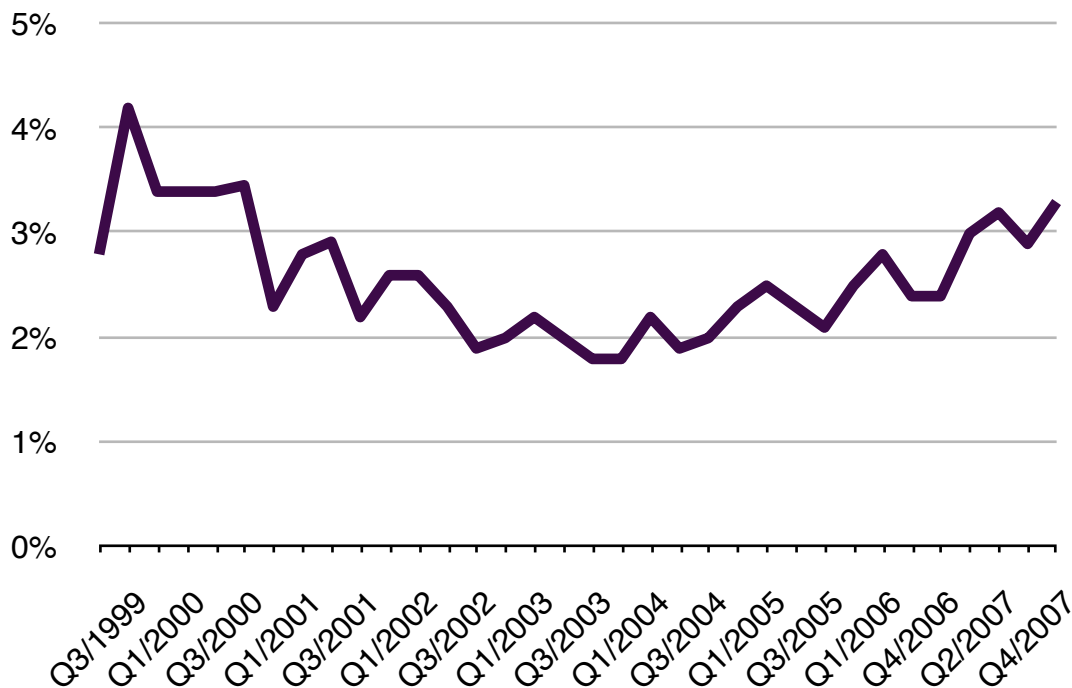
2008

- OS X 10.6 (Snow Leopard) announced
- iPhone App store opens

Appendix E: OS X market share developments

This section aims to provide an outline of the growth of the Apple platform in the last few years by assessing Apple Inc.'s market share developments. Not all Apple customers are Indie customers. Apple provides extensive basic functionality bundled with a computer which might be sufficient for the basic user. Even if Indie software could enrich this type of users' computing experience it still is the question whether the customer would know about the existence of Indie software. Therefore these figures should be interpreted as the potential market for Indie Mac software rather than the actual market which will differ according to the company and the type of software analyzed.

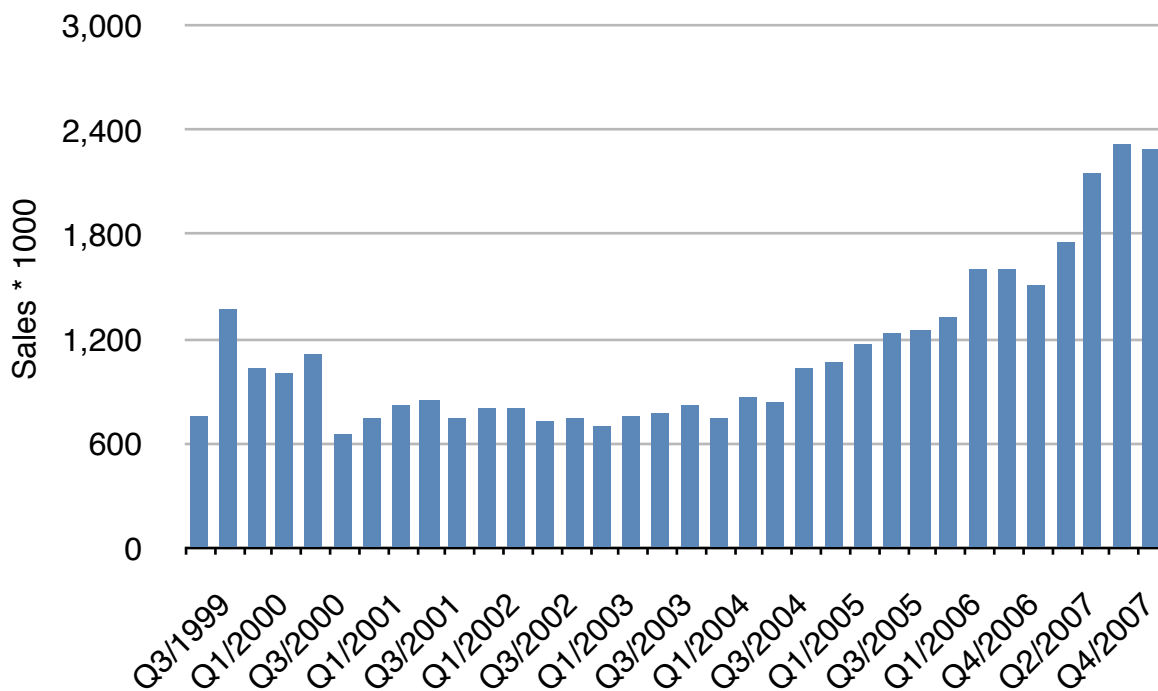
Figure 11, Relative Macintosh worldwide market share, 3rd quarter 1999-1st quarter 2008



Source: http://www.systemshootouts.org/mac_sales.html

If we assess Apple's market share in terms percentage of computers sold, which is the 'traditional' way, the results do not look particularly spectacular (figure 11). Apple's market share has been growing over the last few years, but it has been revolving around the 3% of the total market and has yet to surpass its peak just prior to the .com bust. However, to Indies the absolute figures on computers sold are much more interesting. They rarely operate on the other platforms and therefore what matters to them is the number of Macs that are being used worldwide. Figure 12 shows the absolute sales figures of Macs in the same period.

Fig 12, Quarterly Macintosh sales, 3rd quarter 1999-1st quarter 2008



Source: http://www.systemshootouts.org/mac_sales.html

As the figure shows the number of quarterly Macs sold has more than tripled since its lowest point in the third quarter of 2000. Because the worldwide personal computer market has been growing this does not appear as strong in the relative numbers. However, the last figure is more important to Indies since their market of potential customers has more than tripled since the release of OS X.

Market share in computers can be measured in terms of computer sales but that explains little of the actual use of computers. With the arrival of the Internet and the availability of web statistics another way of measuring global market share has been developed; based on web statistics. If we look at Apple's market share from that perspective the picture changes again. According to these figures (table 1) Apple's market share has risen from 4.29% to 7.83 % in the period between July 2006 and May 2008. This is quite a different figure from the slow growth mentioned in figure 11. The reason these figures are so different could be explained by the hypotheses that Macs in general have a longer life cycle than Windows machines and that Macs are on average used more intensively than Windows computers, at least for browsing the Internet. This suggests that the average Mac user uses his or her computer more intensively than the average Windows user. Therefore, all the figures suggest that the potential market for Indie software has seen a substantial gradual growth ever since the release of Mac OS X.

Table 1, Market share Apple Inc. measured through worldwide Internet traffic

	Windows	Mac	Linux	Other (incl. iPhone)	Total
July, 2006	95.19%	4.29%	0.44%	0.08%	100%
August, 2006	95.11%	4.34%	0.47%	0.08%	100%
September, 2006	94.80%	4.72%	0.40%	0.08%	100%
October, 2006	94.31%	5.22%	0.39%	0.08%	100%
November, 2006	94.16%	5.39%	0.37%	0.08%	100%
December, 2006	93.86%	5.67%	0.37%	0.10%	100%
January, 2007	93.33%	6.22%	0.35%	0.10%	100%
February, 2007	93.05%	6.38%	0.42%	0.15%	100%
March, 2007	93.40%	6.09%	0.40%	0.11%	100%
April, 2007	93.21%	6.24%	0.41%	0.14%	100%
May, 2007	92.94%	6.48%	0.43%	0.15%	100%
June, 2007	93.34%	6.03%	0.43%	0.20%	100%
July, 2007	93.28%	5.99%	0.46%	0.27%	100%
August, 2007	93.06%	6.18%	0.47%	0.29%	100%
September, 2007	92.47%	6.63%	0.49%	0.41%	100%
October, 2007	92.49%	6.58%	0.50%	0.43%	100%
November, 2007	92.42%	6.80%	0.57%	0.21%	100%
December, 2007	91.79%	7.31%	0.63%	0.27%	100%
January, 2008	91.50%	7.57%	0.64%	0.29%	100%
February, 2008	91.58%	7.46%	0.65%	0.31%	100%
March, 2008	91.57%	7.48%	0.61%	0.34%	100%
April, 2008	91.64%	7.38%	0.63%	0.35%	100%
May, 2008	91.13%	7.83%	0.68%	0.36%	100%

Source: <http://marketshare.hitslink.com/>

References

Ahuja, M.K. & J.E. Galvin (2003): Socialization in virtual groups. In: *Journal of Management*, Vol.29, no.2, pp. 161-185.

Allport, G.W. (1954): *The nature of prejudice*. Boston: the Beacon press.

Anderson, C. (2006): *The long tail*. London: Random House business books.

Anguish, S., E.M. Buck & D.A. Yacktman (2002): *Cocoa programming*. Canada: Pearson technology group.

Asensio, M. & V. Hodgson (2001): Virtual communities in education. In: M. Huysman & P. van Baalen (eds.): *Communities of practice*. Trends in communication series, vol. 8. Amsterdam: Boom. pp. 65-77.

Blumer, H. (1974): *Symbolisch interactionisme, perspectief en methode*. Meppel: Boom [1969, *Symbolic interactionism*, Englewood Cliffs: Prentice Hall. Translated from English by H.O. van den Berg].

Boschma, R.A., K. Frenken & J.G. Lambooy (2002): *Evolutionaire economie, een inleiding*. Bussum: Coutinho.

Bourdieu, P. (1993): *The field of Cultural production*. Cambridge: Polity press.

Bourdieu, P. (1984): *Distinction, A social critique on the judgement of taste*. Cambridge: Harvard University press. [1979, *La Distinction: Critique sociale du judgement*, Paris: Les editions de Minuit. Translated from French by R. Nice]

Bourdieu, P. (1977): *Outline of a theory of practice*. Cambridge: Cambridge university press, [1972, *Equisse d'une théorie de la pratique, précède trois études d'ethnologie kabyle*, Switzerland: Libraire Droz. Translated from French by R. Nice]

Breschi, S. & , F. Lissoni (2001): Localised knowledge spillovers vs innovative milleux Knowledge “tacitness” reconsidered. In: *Papers in regional science*, vol. 80, pp. 255-273.

- Brown, J.S. & P. Deguid (2000): *The social life of information*. Boston: Harvard Business school press.
- Bryman, A. (2001): *Social research methods*. New York: Oxford university press.
- Chabal , P. & J.P. Daloz (2006): *Culture troubles; politics and the interpretation of meaning*. Chicago: University of Chicago press.
- Cooke, P. & K. Morgan (1998): *The associational economy*. Oxford: Oxford university press.
- Cumbers, A. and D. Mackinnon (2004): 'Introduction: clusters in urban and regional development', *Urban Studies*, vol.41, no.5-6, pp. 959–969.
- Crang, M. (1998): *Cultural geography*. London: Routledge.
- Cruikshank, J.L. (2006): *The Apple way, 12 management lessons from the world's most innovative company*. New York: McGraw-Hill.
- David, P.A. (2000): *Path dependence, its critics and the quest for 'historical economics'*. [online]. URL: [<http://www-econ.stanford.edu/faculty/workp/swp00011.pdf>]. Last visited: May, 30 2008.
- David, P.A. (1994): Why are institutions the 'carriers of history'? Path dependence and the evolution of conventions, organizations and institutions. In: *Structural Change and Economic Dynamics*, vol. 5, no. 2, pp. 205-220.
- David, P.A. (1990): The dynamo and the computer. An historical perspective on the modern productivity paradox. In: *The American Economic Review*, vol. 80, no.2, pp. 255-361.
- David, P.A. (1985): Clio and the economics of QWERTY. In: *The American Economic Review*, vol.75, no.2, pp. 332-337.
- Derné, S. (1994): Cultural conceptions of human motivation and their significance for culture theory. In: D. Crane (ed.): *The sociology of culture*. Cambridge/Oxford: Blackwell.
- Deutschman, A. (2000): *The second coming of Steve Jobs*. New York: Broadway books.

Dicken, P. (2007): *Global shift, 5th edition*. London/Thousand Oaks/New Delhi: Sage.

Donath, J.S. (1999): Identity and deception in the virtual community. In: Smith, M.A. & P. Kollock (eds.): *Communities in Cyberspace*. London: Routledge, pp. 29-60.

Elias, N. (1971): *Wat is sociologie?*. Utrecht/Antwerpen: het Spectrum. [1970 *Was ist Soziologie?*, München: Jeventa Verlag. Translated from German by J. Vollers & J. Goudsblom].

Friedman, T.L. (2006): *The world is flat, The globalized world in the twenty-first century*. 2nd edition, London: Penguin.

Florida, R. (2002): *The rise of the creative class*. New York: Basic books.

Geertz, C. (1973): *The interpretations of cultures*. New York: Basic books.

Gertler, M.S. (2003): Tacit knowledge and the economic geography of context, or The undefinable tacitness of being (there). In: *Journal of economic geography*, vol.3, pp. 75-99.

Giddens, A. (2006): *Sociology, 5th edition*. Cambridge: Polity Press.

Giddens, A (1991): *Modernity and self identity, self and society in the late Modern age*. Stanford: Stanford university press.

Gladwell, M. (2000): *The tipping point*. London: Abacus.

Glassner, W. (1975): *The Identity society, revised edition*. New York: Harper & Row publishers.

Goss, J. (2005): Consumption geographies. In: Cloke, P. et al. (eds.): *Introducing human geographies, 2nd edition*. Abingdon: Hodder Arnold, pp. 253-265.

Granovetter, M. (1985): Economic action and social structure: the problem of embeddedness. In: *American journal of sociology*, vol. 91, pp: 481-510.

Hess, M. (2003): "*spatial*" relationships? Towards a re-conceptualisation of embeddedness. GPN working paper 5. Online, URL: [<http://www.sed.manchester.ac.uk/geography/research/gpn/gpnwp5.pdf>], last visited, 2-07-2008.

Hotz-Hart, B. (2000): Innovation Networks, Regions and Globalization. In: In G.L. Clark et al. (eds): *The Oxford handbook of economic geography*. Oxford: Oxford university press, pp. 432-455.

Johnson, R. (1993): Editors introduction: Pierre Bourdieu on Art, Literature and Culture. In: P. Bourdieu, *The field of cultural production*. Cambridge: Polity press, pp. 1-29.

Kollock, P. (1999): Economies of online cooperation. In: Smith, M.A. & P. Kollock (eds.): *Communities in Cyberspace*. London: Routledge, pp. 220-243.

Kotkin, J. (2000): *The New Geography, how the digital revolution is reshaping the American landscape*. New York: Random House.

Krippner, G.R. (2001): The elusive market: Embeddedness and the paradigm of economic sociology. In: *Theory and Society*. vol.30, pp: 775-810.

MacKinnon, D., A. Cumbers & K. Chapman (2002): Learning, innovation and regional development: a critical appraisal of recent debates. In: *Progress in human geography*, vol. 26, no.3, pp. 293-311.

Malmberg, A. & Maskell, P. (2001): *The elusive concept of localization economies*. Paper for the industrial clusters revisited: innovative places or uncharted spaces? Seedion. AAG annual conference NY Feb 27 - Mar 3 2001. [online]. URL: [http://www.utoronto.ca/isrn/publications/WorkingPapers/Working01/Malmberg01_Elusive.pdf], last visited: 14-08-2008.

Maskell, P., Barthelt, H. & Malmberg, A. (2004): *Temporary Clusters and knowledge creation: The effects of international Trade Fairs, Conventions and other personal gatherings*. Paper presented at the 100th annual meeting of the association of American Geographers. [Online]. URL: [<http://ir.lib.cbs.dk/paper/ISBN/x645152756>], last visited: 26-06-2008

Mitchell, D. (2000): *Cultural geography, a critical introduction*. Malden/Oxford/Victoria: Blackwell.

- Mott, T. (2001): *Learning Carbon*. Sebastopol, CA: O' Reilly.
- Nye, J.S. (2008): Public diplomacy and Soft Power. In: *The annals of the American academy of political and social science*. Vol. 616, pp. 94-109.
- Ley, D. (1980): Geography without human agency: a humanistic critique. In: J. Agnew, et al. (1996): *Human geography, an essential anthology*. Malden/Oxford/Victoria: Blackwell.
- Linzmayr, O.W. (2004): *Apple Confidential 2.0*. San Francisco: No Starch Press.
- Perrow, C. (1986): *Complex organizations*. 3rd edition. New York: Random House.
- Pine, J & J.H. Gilmore (1999): *The Experience economy*. Boston: Harvard Business School press.
- Porter, M. E. (2000): Locations, clusters and company strategy. In: G.L. Clark et al. (eds): *The Oxford handbook of economic geography*. Oxford: Oxford university press, pp. 253-274.
- Porter, M.E. (1998a): *On competition*. Boston: Harvard business school publishing.
- Porter, M.E. (1998b): Clusters and the new economics of competition. In: *Harvard Business Review*. Nov-dec 98, pp. 77-90.
- Ruigrok, W. & R. van Tulder (1995): *The logic of international restructuring*. London/New York: Routledge.
- Saxenian, A. (1994): *Regional advantage. Culture and competition in Silicon Valley and Route 128*. Cambridge/Mass./London: Harvard university press.
- Scott, A.J. (2000): *The cultural economy of cities*. London: Sage.
- Simmie, J. (2004): Innovation and clustering in the globalized international economy. In: *Urban studies*, vol. 41, no. 5/6, pp. 1095-1112.

Sole, D. & M. Huysman (2001): Knowledge, practice and the role of location. In: M. Huysman & P. van Baalen (eds.): *Communities of practice*. Trends in communication series, vol. 8. Amsterdam: Boom. pp. 21-35.

Strauss, A. & J. Corbin (1998): *Basics of qualitative research, 2nd edition*. London: Sage.

Sturgeon, T. (2003): What Really Goes on in Silicon Valley? Spatial Clustering and Dispersal in Modular Production Networks. *Journal of Economic Geography*, vol.3, pp. 199-225.

Wellman, B. & M. Gulia (1999): Virtual communities as communities: Net surfers don't ride alone. In: Smith, M.A. & P. Kollock (eds.): *Communities in Cyberspace*. London: Routledge, pp. 167-195.

Online references³³

ADC (2007): *The Model-View-Controller Design Pattern*. URL: [http://developer.apple.com/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaDesignPatterns/chapter_5_section_4.html], last visited: 11-08-2008.

Alperin, K. (22-04-2008): *Daring did*. URL: [<http://www.heliumfoot.com/blog/50>], last visited: 20-06-2008.

Anderson, B. (19-04-2007): *Apple: a romance*. URL: [<http://buzz.vox.com/library/post/leaving-apple.html>], last visited: 06-08-2008.

Anderson, B. (23-07-2003): *Oh My Gawd...it's Panther-Cocoa!*, URL: [<http://weblog.scifihifi.com/2003/06/25/oh-my-gawdits-panther-cocoa/>], last visited: 06-08-2008.

Anderson, B. (22-02-2003): *useless software*. URL: [<http://weblog.scifihifi.com/2003/02/21/useless-software/>], last visited: 20-06-2008.

Anderson, B. (24-02-2003): *F*** the mainstream*. URL: [<http://weblog.scifihifi.com/2003/02/24/f-the-mainstream/>], last visited: 20-06-2008.

Alfke, J. (10-01-2008): *Gone Indie*. URL: [<http://mooseyard.com/Jens/2008/01/gone-indie/>], last visited: 06-08-2008.

Berners-Lee, T. (undated): *The WorldWideWeb browser*. URL: [<http://www.w3.org/People/Berners-Lee/WorldWideWeb>], last visited: 11-08-2008.

Burbeck, S. (1992): *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)*. URL: [<http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>], last visited: 11-08-2008.

Cook, B. (2001): *Worth the wait, The Omni group and Mac OS X games*. URL: [http://www.apple.com/games/articles/2001/04/omni_group/index.html], last visited: 04-07-2008.

³³ Dates are notated in European notation, DD-MM-YEAR, 09-11-1986 means November 9, 1986

Davidson, J.D. (23-03-2004): *Dev to Dev: James Duncan Davidson interviews Panic*. URL: [<http://www.macdevcenter.com/pub/a/mac/2004/03/23/panic.html>], last visited: 3-07-2008.

Digg (10-12-2007): *Apple most innovative company – 3rd year in a row*. URL: [http://digg.com/apple/Apple_Most_Innovative_Company_3rd_Year_in_a_row], last visited: 7-08-2008.

Diskovery (2001): *Online Codewarrior advertisement*. URL: [<http://www.diskovery.com/codewarrior.html>], last visited: 11-08-2008.

Drunken Batman (21-07-2005): *On Being and Deliciousness, with Wil Shipley*. URL: [<http://www.drunkenblog.com/drunkenblog-archives/000581.html>], last visited: 20-06-2008.

Engst, A. (11-08-2007): *Hacking the press*. Keynote given on the C4[1] conference. 10/12 Aug 2007. URL: [<http://www.viddler.com/explore/rentzsch/videos/12>], last visited: 21-06-2008.

Evans, J. (18-02-2008): *iPhone SDK information leaks, Third party developer lets Apple iPhone SDK secrets slip*. URL: [<http://www.macworld.co.uk/ipod-itunes/news/index.cfm?newsid=20472>], last visited: 5-08-2008.

Fallin, L. (7-11-2007): *Do not quote me on this*. URL: [<http://leefalin.com/blog/2007/11/08/dont-quote-me-on-this/>], last visited: 23-06-2008.

Fleishman, G. (03-07-2006): *“Ware” labels perplexing*. URL: [<http://community.seattletimes.nwsourc.com/archive/?date=20060603&slug=ptmacc03>], last visited: 12-08-2008.

Frank, S. (undated): *Beagle Bros of the 90’s?* URL: [<http://www.panic.com/extras/essays/>], last visited: 12-08-2008.

Francher, T. (15-01-2006): Comment on: *Thinking, boxes & what kittens can do*. URL: [<http://wilshipley.com/blog/2006/01/thinking-boxes-what-kittens-can-do-to.html>], last visited: 29-06-2008.

Garfinkel, S. & M. Mahoney (05-10-2002): *Steve Jobs and the history of Cocoa, part two*. URL: [http://www.macdevcenter.com/pub/a/mac/2002/05/10/cocoa_history_two.html], Last visited: 04-07-2008.

Garfinkel, S. & M. Mahoney (03-05-2002): *Steve Jobs and the history of Cocoa, part one*. URL: [http://www.macdevcenter.com/pub/a/mac/2002/05/03/cocoa_history_one.html], Last visited: 04-07-2008.

Gruber, J. (20-11-2006): *Pinprick*. URL: [<http://daringfireball.net/2006/11/pinprick>], Last visited: 18-06-2008.

Gruber, J. (21-10-2006): *The HIG is dead*. Lecture on the C4 conference 21-22 october 2006, Chicago. As transcribed by Daniel Jalkut. URL: [<http://www.red-sweater.com/blog/213/c4-abridged>] Last visited: 18-06-2008.

Gruber, J. (30-06-2004): *Dashboard vs Konfabulator*. URL: [http://daringfireball.net/2004/06/dashboard_vs_konfabulator], last visited: 5-07-2008.

Hanson, C. (01-04-2006): *Happy 30th Apple!*, URL: [<http://chanson.livejournal.com/tag/apple>], last visited: 12-08-2008.

Hector, K. (2-10-2007): *It's all about Balance*. URL: [<http://kevinhector.blogspot.com/2007/10/it-all-about-balance.html>], last visited: 18-06-2008.

Jalkut, D. (10-01-2008): *Indie Fever*. URL: [<http://www.red-sweater.com/blog/447/indie-fever>], last visited: 06-08-2008.

Kafasis, P. (24-07-2008): *Open questions for the app store*. URL: [<http://blogs.oreilly.com/iphone/2008/06/open-questions-for-the-app-sto.html>], last visited: 06-08-2008.

Kafasis, P. (09-01-2008): *What a free NetNewsWire means*. URL: [<http://www.rogueamoeba.com/utm/posts/Article/NNWFree-2008-01-09-19-00.html>], last visited: 20-06-2008.

Kahney, L. (18-03-2008): *How Apple got everything right by doing everything wrong.*, In: *Wired Magazine*, vol. 16, no.4. [online], URL: [http://www.wired.com/techbiz/it/magazine/16-4/bz_apple?currentPage=all], last visited: 12-06-2008.

Kernelthread/6: *Quest for the operating system*. URL: [<http://www.kernelthread.com/mac/oshistory/6.html>], last visited: 03-07-2008.

Kernelthread/7: *The Next Chapter*. URL: [<http://www.kernelthread.com/mac/oshistory/7.html>], last visited: 03-07-2008

kpcp.com (undated): *iFund™*, URL: [<http://www.kpcb.com/initiatives/ifund/index.html>], last visited: 13-08-2008.

Krazit, T. (06-03-2008): *Live blog from Apple's iPhone SDK announcement*. URL: [http://news.cnet.com/8301-13579_3-9886460-37.html?hhTest=1], last visited 13-08-2008.

Lee, M. (16-06-2008): *WWDC, 08*. URL: [<http://www.atomicwang.org/motherfucker/Index/2EB9528C-DD18-4731-BD2A-760E7DA5A74B.html>], last visited: 20-06-2008.

Lee, M. (10-04-2008): *Bubble Boggle*. URL: [<http://www.atomicwang.org/motherfucker/Index/B9CoC61F-37FA-436D-A6B4-27545F009BBA.html>], last visited: 06-08-2008.

Lee, M. (07-02-2008): *Mad bundle*. URL: [<http://www.atomicwang.org/motherfucker/Index/2DB12377-20C5-4CF7-80D7-42FEC83EE43F.html>], last visited: 20-06-2008.

Long, M. (19-04-2008): *From Hacker To microISV: App Names And Icons*. URL: [<http://www.cimgf.com/2008/04/19/from-hacker-to-microisv-app-names-and-icons/>], last visited, 27-06-2008.

Lynch, P. (1997): *NEXTSTEP tech review*. URL: [<http://www.paullynch.org/NeXTSTEP/NeXTSTEP.TechReview.html>], last visited: 11-08-2008.

Mueller, G. (25-12-2005): *How to become an independent programmer in just 1068 days*. URL: [<http://gusmueller.com/blog/archives/2005/12/25.html>], last visited 29-06-2008.

Nack, J. (2-04-2008): *Photoshop, Lightroom and Adobe's 64 bit roadmap*. URL: [http://blogs.adobe.com/jnack/2008/04/photoshop_lr_64.html], last visited 04-07-2008.

Omni Group (2004-2008): *What is Omni*. URL: [<http://www.omnigroup.com/company/whatisomni/>], last visited: 20-06-2008.

OSNews (10-07-2005): *Apple drops the Cocoa-Java bridge*. URL: [<http://www.osnews.com/story/11165>]. Last visited: 04-07-2008.

Pannel, J. (14-06-2005): *Re: Company description - 'We' or 'I'?* URL: [<http://tech.groups.yahoo.com/group/macsb/message/1498>], last visited: 20-06-2008.

Rentzsch, J. (10-08-2007): *Indie Ethos*, Keynote given on the C4[1] conference. 10/12 Aug 2007. URL: [<http://www.viddler.com/explore/rentzsch/videos/3>], last visited: 02-07-2008.

Rentzsch, J. (08-09-2006): *C4: Chicago Mac Developer Conference*. URL: [<http://rentzsch.com/c4/zero>], last visited: 12-08-2008.

Rochester Review (Winter 97-98): *Alumni Gazette, polishing Apple*. URL: [<http://www.rochester.edu/pr/Review/V60N2/gazette.html>], last visited: 11-08-2008.

Sasser, C. (10-09-2007): *Coda toolbar and the three pixel conundrum*. URL: [<http://www.cabel.name/2007/09/coda-toolbar-and-three-pixel-conundrum.html>], last visited: 06-08-2008.

Sasser, C. (2007a): *The true story of audion*. URL: [<http://www.panic.com/extras/audionstory/>], last visited: 06-08-2008.

Shipley, W. (18-12-2007): *Transitions and Epiphanies*. URL: [<http://wilshipley.com/blog/2007/12/transitions-and-epiphanies.html>], last visited: 06-08-2008.

Shipley, W. (10-08-2007): *Monster marketing*. Keynote given on the C4[1] conference. 10/12 Aug 2007. URL: [<http://www.viddler.com/explore/rentzsch/videos/4/>], last visited: 05-08-2008.

Shipley, W. (5-10-2006): *Pimp My Code, Part 12: Frozen in Carbonite*. URL: [<http://wilshipley.com/blog/2006/10/pimp-my-code-part-12-frozen-in.html>], last visited: 11-08-2008.

Shipley, W. (24-07-2005): *Student Talk from WWDC 2005*. URL: [<http://wilshipley.com/blog/2005/06/student-talk-from-wwdc-2005.html>], last visited, 06-07-2008.

Shipley, W. (26-04-2005): *Longhorn: Today's technology, Tomorrow!*. URL: [<http://wilshipley.com/blog/2005/04/longhorn-todays-technology-tomorrow.html>], last visited: 21-06-2008.

Simmons, B. (28-03-2004): Interview Brent Simmons, by John Gruber. URL: [http://daringfireball.net/2003/03/interview_brent_simmons], last visited: 24-06-2008.

Simmons, B. (19-09-2002): *Why I develop for OS X*. URL: [http://inessential.com/?comments=1&postid=2138], last visited: 20-06-2008.

Sinclair, D. (23-04-2007): *Dejal Developer pages: free Cocoa code*. URL: [http://www.dejal.com/blog/2007/04/dejal-developer-pages-free-cocoa-code], last visited: 26-06-2008.

Siracusa, J. (02-04-2008): *Rhapsody and blues*. URL: [http://arstechnica.com/staff/fatbits.ars/2008/04/02/rhapsody-and-blues], last visited: 04-07-2008.

Slashdot (17-12-2006): *Macheist "week of Mac Developer" Causes Schism*. URL: [http://apple.slashdot.org/article.pl?sid=06/12/17/2022219&from=rss], last visited: 24-06-2008.

Smith, B. (29-07-2002): *Exclusive - Watson Developer Speaks Out Against Apple; Plans Port To Windows*. URL: [http://www.macobserver.com/article/2002/07/29.7.shtml], last visited: 05-07-2008.

Spiers, F. (2-11-2007): *On Location*. URL: [http://speirs.org/2007/11/02/on-location/], last visited 26-06-2008.

Stevenson, S. (23-10-2006): *thoughts on human interface guidelines*. URL: [http://theococoa.com/document.page/326], last visited: 20-06-2008.

Stevenson, S. (08-06-2006): *Cocoa Help: Mentoring and immediate answers*. [http://theococoa.com/document.page/274], last visited: 01-07-2008.

Stevenson, S. (22-11-2004): *cocojox on IRC*. URL: [http://theocacao.com/document.page/50#], Last visited: 23-06-2008.

Smykyl, J. (31-05-2007): *The Delicious Generation strikes back*. URL: [http://arstechnica.com/journals/apple.ars/2007/05/31/the-delicious-generation-strikes-back], last visited: 20-06-2008.

Wikipedia-Dot-com bubble entry: *Dot-com bubble*. URL: [http://en.wikipedia.org/wiki/Dot-com_bubble], last visited: 04-07-2008.

Wikipedia-Nextstep entry: *Nextstep*. URL: [<http://en.wikipedia.org/w/index.php?title=Nextstep&printable=yes>], last visited: 03-07-2008.

Winer, D. (17-10-2007): *Apple's iPhone SDK announcement*. URL: [<http://www.scripting.com/stories/2007/10/17/applesIphoneSdkAnnouncemen.html>], last visited: 13-08-2008.

Wood, D. (2006): *The long story behind Karelia's new logo*. URL: [http://www.karelia.com/news/small_and_nimble_the_long_s.html]. Last visited: 5-07-2008.

Wood, D. (2001): The tortoise and the hare, why Carbon developers should start learning Cocoa. In: *Mactech magazine*, vol.17, no.12. URL: [<http://www.mactech.com/articles/mactech/Vol.17/17.12/Dec01MacOSX2/index.html>], last visited 04-07-2008.