University of Salford, Manchester

Robert Kennedy College

Is the Future of Business Software Proprietary or Free and Open-Source? A Macroscopic Information Systems Investigation

Lev Lafayette

Student ID: @00545425

A dissertation is submitted in partial fulfilment of the requirements of The University of Salford for the degree of MSc in Information Systems Management

February 2020

Declaration

By submitting my work here I declare that:

- This is my own work
- The work of others has been acknowledged
- Data, experimental, surveys or other investigative results have not been falsified
- I have read and understood the University Policy on the Conduct of Assessed Work (Academic Misconduct Procedure)

Abstract

Most businesses, for day-to-day applications, makes use of well-known proprietary software systems. But in the last twenty years, servers and embedded systems have increasingly made use of products a variety of free-and-open-source (FOSS) products. Between these two general models there is software projects and aggregate products may exist in the continuum and that have a mix of various licenses structures (e.g., public domain, permissive FOSS, reciprocal FOSS, freeware, proprietary, trade secrets). The question raised is whether there is a trend in software license types, and what effects different licenses have on innovation, economic wealth, and organisational profits. To make a determination this dissertation engages in a multidisciplinary exploration the literature related to software licenses from legal, economic, business, and engineering perspectives and compares the theories that arises with an empirical study of trends, investigation into case studies, and interviews with system engineers. With this macroscopic scale approach to an information systems problem, general trends are identified but also contemporary trends to ensure monopolistic advantage.

Acknowledgements

I would like to offer my sincere thanks to my supervisor, Professor Stuart Wallace for his advice, especially in various aspects of correctness in formatting and structure.

I would also like to thank Chris Samuel, Systems Engineer, for the National Energy Research Scientific Computing Center, United States, for gently leading me into the profession of high-performance and scientific computing and his advice on open standards for interoperability and efficiency.

I would also like to thank Nicolás Erdödy, Chief Executive Officer, Open Parallel, New Zealand, who insists on inviting me to speak each year at an exceptionally high-quality conference with some of the world's leading engineers in providing solutions to the most complex issues in applied computing, much of which contributed to this study.

I would also like to thank John Gustafson, Professor, National University of Singapore, not only for his namesake law, or for his investigations in numerical representation without error, or even for writing the foreword to my book on sequential and parallel programming, but rather for a particular engineering perspective that always seeks to find unconventional solutions.

I would also like to thank Daniel Tosello, friend and co-worker at the HPC team at the University of Melbourne, for his rapid and very deep insight on

matters concerning the interface between organisational processes and information technologies.

Finally, I would like to thank the subjects of the interview survey, all of whom provided examples of real-world examples of the transition between different software licensing regimes. Their requisite anonymity by no means lessens the degree that I am thankful for their time and insightful remarks.

Contents

List of Tables and Figures

List of Abbreviations

AIX: Advanced Interactive eXecutive. A proprietary UNIX produced by IBM. Still in active development.

CMM: Capability Maturity Model. A software development model used by organisations contracted with the U.S. Department of Defence.

CSP: Communicating Sequential Processes. A formal language for describing interactions in concurrent systems.

BSD: Berkeley Software Distribution, a free-and-open-source version of Unix.

FOSS: Free and Open-Source Software. Does not refer to "free", as in price, but rather free in the ability to run the software, and view, modify, and distribute the source-code.

FSF: Free Software Foundation, a non-profit group founded in 1985 to promotes the freedom to study, distribute, create, and modify computer software.

GNU: Gnu's Not Unix, a recursive acronym for the GNU project for free software.

HP-UX : Hewlett Packard UNIX. A proprietary UNIX produced by Hewlett-Packard. Still in active development.

IS: Information Systems. An academic study of connections between the systems of information technology and organisational processes.

MECE: Mutually Exclusive Collectively Exhaustive. A business mapping process that separates objects and processes into subsets that are either mutually exclusive (ME) and collectively exhaustive (CE).

MIME: Multipurpose Internet Mail Extensions. An Internet standard that extends plain-ASCII email to include other character sets and multimedia attachments.

MVP: Minimal Viable Program/Product. A program or product that has all functional features.

MTA: Mail-transfer agent: A software server that can transfers electronic mail messages from one computer to another.

ODF: Open Document Format. The Open Document Format for Office Applications (ODF), an open-format, compressed, XML-based file-format for business documents. Published as an ISO/IEC international standard ISO/IEC 26300.

OEM: Original Equipment Manufacturer. A company that purchases equipment that is manufactured by another company. In computer systems, manufacturers may purchase software with reduced volume licensing generating System Locked Pre-installation.

OOXML: Office Open XML, an open-format XML-based file-format for business documents developed by Microsoft. Published as an ISO and IEC international standard ISO/IEC 29500).

OSI: Open Source Initiative. A US-based non-profit organisation which advocates the use of open-source software.

SaaS: Software as a Service. Provision of application level software as a service through cloud computing.

SME: Small to medium enterprise.

SSM: Soft-Systems Methodology; an approach to organisational process modeling that uses a "system" as an interrogative device for discussion, feasibility, problem identification, and caccomodations.

UNICS/UNIX: Uniplexed Information and Computing Service, was the original name for the Unix operating system coined in 1970, and a pun on Multics (Multiplexed Information and Computer Services), from which it was derived. When capitalised as UNIX, it refers to AT&T the trademarked version.

UML: Unified Modelling Language. A general purpose modelling language for software engineering, designed to provide visual representation of a system.

XML: Extensible Mark-Up Language. A text-based mark-up langauge for encoding documents using arbitrary data structures encoded in a format that is both both human-readable and machine-readable.

VDM: Vienna Development Method. A formal language methods for software and computer system development.

VPAC: Victorian Partnership for Advanced Computing. A consortium of universities from Victoria and registered research agency to provide high-performance computing. Disbanded in 2015.

Chapter 1: Introduction

1.1 A Personal and Academic Interest

The research question is whether the future of business software licensing is proprietary or free-and-open-source (FOSS). This implies an analysis of existing trends (quantification) and providing a theoretical grounding on why these trends exist (qualification). A personal introduction also exists to make motivations and biases explicit, and to indicate prior experience in the subject. There is also an academic interest in an apparent conflict between business reasoning, which argues for an organisation weakening the power of competitors and consumers, and that of economics reasoning, which argues for heightened competition and maximising consumer sovereignty, and also an engineer's interest in the most effective and efficient products. The academic interest comes with a background in four different degrees; a Master of Business Administration, separate graduate degrees in project management and adult education, and an honours degree in politics, philosophy, and sociology. A professional interest is from employment as an engineer and educator in high-performance computing (HPC) for the Victorian Partnership of Advanced Computing (VPAC) and the University of Melbourne for almost 15 years, and committee-level involvement Linux Users of Victoria from 2005 to 2019.

The study is within the discipline of information systems, which needs to be positioned. Some (Checkland, 1988) draw a history in the subject noting an early interests in signal transmission developing into concerns of semantic knowledge, emergent-properties, and meaning-attribution, all resulting in Soft-

Systems Methodology (SSM) in the 1980s. Others (Paul 2007) argue that information systems lacks a distinct identity, located either within business studies or computer science, as the combination of both information technology and organisational processes. This is reflected in the philosophy of formal pragmatics (Apel, 1980, Habermas, 1984), which unites epistemology with ontology by taking a rationalisation of world orientations with specific verification claims. Information Systems is thus a multi-disciplinary science; it includes the world-orientation of objective facts (information technology), and the world-orientation of social facts (institutional processes), which can be readily incorporated into the DIKW hierarchy (data, information, knowledge, and wisdom) (Rowley, 2007). The following figure illustrates the philosophical approach:

Figure 1: World Orientations and Verification in Formal Pragmatics

Unverifiable Metaphysics	Physicalist, Symbolist, Idealist Theology		
Verifiable Reality	Logical and Empirical Philosophy		
Orientations/Worlds (verification)	1. Objective or "The" External World	2.Intersubective or "Our" Social World	3. Subjective or "My" Internal World
1. Propositions of Truth - Sciences (correspondence)	Scientific facts	Social facts	Unverifiable
2. Propositions of Justice - Laws (consensus)	Unverifiable	Legal Norms	Moral Norms
3. Propositions of Beauty – Arts (sincerity)	Aesthetic Expressions	Unverifiable	Sensual Expressions

Elaborated from Habermas (1984, p239)

Many IS inquiries are an application within a particular enterprise, a "microscopic-IS" approach. However, this dissertation is "macroscopic-IS" concern. The core difference is that a microscopic investigations will apply the

assumptions of the discipline to particular instances and are contextually bound. A macroscopic approach applies critical concerns and connections from IS and related disciplines with universal application. Thus, this study is narrow scope in terms only being concerned with software licenses used by organisations, but broad scope in terms of being applicable to any organisation that is affected by software.

1.2 A Brief History and Definitions

Early computer systems were often sold as hardware with the software source-code provided (e.g., the 1951 IBM 701, the 1953 UNIVAC, the 1959 SHARE operating system). A major change occurred in 1974 when the US Commission on New Technological Uses of Copyrighted Works recommended that software could be subject to copyright, which was established in law the 1983 in the United States with the Apple Computer, Inc. v. Franklin Computer Corporation case, and in the same year, IBM introduced an object-code-only policy. During this period Unix gained prominence as an operating system, with the first commercial version, UNIX System V, released by AT&T in 1983 in competition with the University of California, Berkeley Computer Systems Research Group (BSD UNIX). Also in 1983 the GNU Project which was initiated to create utilities and applications that were on Unix systems but under a FOSS license.

Whilst competition between various forms of proprietary Unix-based systems occurred during the 1980s and the early 1990s, the competing new desktop computing systems readily adapted to the new legal standard with

applications such as WordStar and VisiCalc having prominence. As the personal computer market effectively narrowed to IBM-PCs and clones and Apple machines by the early 1990s, so too did operating systems and office applications or "productivity software". Microsoft Office in particular, starting with MS-Word in 1983, became an integrated suite in 1990 and achieved market dominance. More recently it has moved from a stand-alone perpetual license to a subscription license with a client-server and cloud services model with Office365. Today, there is widespread use of proprietary software in various applications used by business workers, including the operating system and utilities, web browsers, databases, customer-relationship management software, enterprise resource planning, and various domain-specific software. However, in the last twenty years, servers, embedded systems, scientific applications, and server-level software have increasingly made use of FOSS products. In the 1990s the combination of the Linux kernel with the GNU application suite began to replace the various proprietary Unix systems. In addition, there is various software projects and aggregate products that have a mix of various license structures.

Software licenses can range from those that are entirely in the public domain to permissive FOSS licenses, reciprocal FOSS licenses, proprietary licenses, and trade secrets. Public domain licenses are defined as those which have no legal exclusion in use. Permissive FOSS licenses in software are typically those which allow for redistribution of the software as desired but with attribution. Reciprocal FOSS licenses require that the freedoms that a software user has received from software must be re-distributed. In comparison, proprietary software is where copyright is applied. In some

cases, a software patent is also applied. Finally, unlike copyright and patents, which have a government record, a trade secret is not disclosed to any public authority.

1.3 Aims and Justifications

There can be little doubt of the importance of the continuing contribution of computer software to scientific and technological developments and administrative efficiency. The software which drives these technologies, and continues to do is, however, subject to different and competing license regimes, which can have varying levels of effectiveness; determining this is a dissertation aim. Whilst it may not be immediately obvious to end-user, any inquiry which can contribute to improved efficiency and effectiveness of software development will have significant importance on the daily lives of all users; establishing this is a justification.

From an organisational perspective, the desire for competing software licenses should vary according to their functional needs. System engineers and software developers should prefer access to source code for purposes of efficiency and effectiveness. Software consumers should prefer functionality, compatibility, and reduced vendor lock-in, all at the lowest possible price. Software vendors should prefer to establish their software as the industry default and extract monopolistic profits. The functional needs of a software engineer have their equivalent in the intellectual expectation that software source code is available for development. Choice theory, whether rational or bounded, is at the core of microeconomic decision-making and contrasts with

the macro-economic expectations of institutional economics. Monopolistic competition, the quest for "competitive advantage", is a core assumption of business reasoning. Finding a theoretical model that satisfies these disciplinary differences in itself a significant and worthwhile interdisciplinary project, which acts as both an aim and justification.

1.4 Overview and Outline

This dissertation explores software license trends supplemented with the use of case studies and expert interviews. The ultimate research objective of the thesis is to draw upon these resources to determine to what degree the future of business software, will follow a free-and-open-source licensing model or whether it will follow a more proprietary model. In exploring this research question this inquiry will (a) conduct a literature review of related works, (b) provide secondary and primary research, and analyse the data of the review and research, and (c) provide a conclusion and evaluation and recommendations derived from the results.

Literature Review

The literature review will cover four major areas. Firstly, there is a discussion of the history and use of various software licenses, which is supplemented by an exploration of the Church-Turing thesis on the nature of computational functions. The application of particular software licenses is supplemented with the business literature related to business profitability. This is compared with the economics of competition, and especially institutional

economics and imperfect competition in relation to software licensing. Both are supplemented by studies in software engineering quality.

Methodology and Methods

Whilst the literature review provides a theoretical understanding and background to the thesis question, additional research provides evidence which contributes to the answer. Prior to the research consideration of methodology and methods is considered; the former representing the theory of method selection. This selected methods include analysis of software trends within organisational contexts, including server software (web-server, mail-servers, domain-name servers) and common personal software (web-clients, office application suites), the latter differentiated by form-factor. The differentiations are based in part on function (server, personal use), human utilisation (operating system, web-browser, productivity suite), and form-factor (server, desktop/laptop, mobile device).

These trends provide quantification, however they do not provide qualification, an explanation of why changes have occurred. Engaging in qualitative examples, case studies also contribute to the research. This includes the forking of open-source code when proprietary impositions are put into place (Herrera, 2012), the establishment of the ISO/IEC 26300 and the ISO/IEC 29500 standards, and the adoption of open-source productivity software by the city council of Munich which switched from proprietary to open-source then back again. Primary research is derived from semi-structured interviews with senior system engineers and developers involved where there was a conscious decisions to change licensing regimes. The

data analysis will seek to draw together this secondary and primary data to provide a conclusion which answers the research question in a theoretically grounded manner.

Chapter 2: Literature Review

2.1 An Overview

The literature review covers initially covers various software licenses, which is supplemented by the an exploration of the Church-Turing thesis on the nature of computational functions. Following this, is a review of business literature related to business profitability and licenses. Providing a contrary perspective, a review of the economics of competition, and especially starting with institutional economics and imperfect competition. Finally, a review of some core texts in software engineering quality.

Providing an overview for software licensing classification and implementation, the most prominent texts are "A Practical Guide to Software Licensing for Licensees and Licensors" (Classen, 2017), "Software Licensing: Principles and Practical Strategies" (Rustad, 2010) and, with a particular emphasis on open-source licenses, "Understanding Open Source & Free Software Licensing" (St. Laurent, 2004).

The classic contemporary study of profitability from a business remains "The Competitive Advantage" (Porter, 1985). An early attempt to apply these principles in software can be found in the doctoral thesis "Sources of competitive advantage in knowledge-based industries" (Levine, 1992), and "Business Process Oriented Implementation of Standard Software" (Kirchmer, 1999), with exploration in a series of essays for small organisations in "Information Technology and Competitive Advantage in Small Firms" (Webb, Schlemmer, 2008). A recent inquiry into competitive advantage in the software

industry can also be found in "Technology Strategy Patterns: Architecture as Strategy" (Hewitt, 2018).

The comparison with establishing monopolistic advantage through proprietary licensing starts with "The Economics of Imperfect Competition" (Robinson, 1933). Whilst this book is quite old it has a powerful influence over any studies which seek to consider realistic economic modelling, rather than the idealised expressions of perfect competition. An application which includes software is "The Economics of Imperfect Knowledge" (Richardson, 1998). More contemporary studies include "The Business and Economics of Linux and Open Source" (Fink, 2003), and select essays from "Handbook of research on open source software" (St. Amant, Still, 2007), and "Advances in Software Economics" (Popp, 2011).

The final area of a literature review is the relationship between software licenses and engineering perspectives. A useful introduction to the subject without reference to particular licenses is found in "A Practical Approach to Software Quality", "Mathematical Approaches to Software Quality" (O'Regan, 2002, 2006), and "Facts and Fallacies of Software Engineering" (Glass, 2002). A popular text on the subject is "The Cathedral and the Bazaar" (Raymond, 1999), which is also compared alongside "Free as in Freedom" (Williams, 2002), and "The Software Paradox: The Rise and Fall of the Commercial Software Market" (O'Grady, 2015).

These texts provide the core theoretical foundations and their elaborations for the debate. As can be expected, they are supplemented by contemporary journal articles and legal cases which explore the conflict. Papers by August et. al., (2008, 2013, 2017) are particularly insightful in this

regard. Atal and Shanker (2014, 2015) explore public good competition and the differences in the permissive and restrictive open-source licenses, whereas development changes are explored by Herrera (2012). The relationship between institutional power and market structure is reviewed by Vatiero (2009) and Gabszewicz (2000).

2.2 Classification and Use of Software Licensees

Classen's "A Practical Guide to Software Licensing for Licensees and Licensors" (Classen, 2017), concerns itself primarily with grounded legal implementation. Of extensive and systemic scope, numerous components of software licenses are provided, differentiating between 'sale' and 'license' (a conditional sale), before working through definitions within a software license, (e.g., "license" and "licensor", source code versus object code), their bounds (e.g., geographic restrictions, transferability), and the major clauses (e.g., indemnification, payment, breaches, remedies, etc).

Not just descriptive, the text also discusses issues of implementation and protection of agreements, of which escrow agreements and elements of U.S. law (e.g., Uniform Trade Secrets Act, Economic Espionage Act) feature prominently. Extensive investigation is carried out on application of the Universal Commercial Code and the attempts to apply the code to software with the subsequent development of the Uniform Computer Information Transactions Act, a curious inclusion given it has only been ratified in two US states. Whilst of a more practical use for users, but less so for a theoretical

review, over half the text is dedicated to providing a range of model forms for various software service agreements.

Whilst an extensive and statement of the necessary components of software licenses, there are issues with Classen. In particular, it assumes that producers of software wish to avoid the First Sale Doctrine, governing resale and redistribution. There is some discussion of open-source licenses, but the author minimises their contribution. In order to discuss this matter with the same sort of rigour one looks to St. Laurent's "Understanding Open Source & Free Software Licensing" (2004). Whilst an older text, and well due for an new revision to cover new FOSS licenses (e.g., GPLv3), it does cover the basics of copyright law, and defines and differentiates between two broad classes of licenses on a functional basis; the MIT, BSD, Apache, and "academic-like" licenses., the GPL, LGPL, and Mozilla Licenses., the Qt, Artistic, and Creative Commons Licenses, along with consideration on non-open source licenses, legal issues of FOSS licenses, as well as considering issues of developing with a FOSS license.

Whilst lacking the line-by-line detail of Classen, St. Laurent also identifies that non-FOSS licenses are a type of rent-seeking, and that FOSS is preferred on engineering grounds of innovation, reliability, and longevity. Open source is defined following the Open Source Initiative (OSI) certification requirements (e.g., free redistribution, source code, derived works under the same license, non-discrimination of persons of fields of endeavour). Starting from the simplest licenses (MIT, BSD, Apache) the author notes how most of these clauses are included, along with warrantee and liability indemnification, to more complex licenses (GPL) which requires stronger enforcement of

requiring the redistribution of works under the same license. The Artistic (Perl) license is noted for providing the right of the copyright holder to enter into a commercial relationship with those who wish to redistribute with further restrictions. The Creative Commons license is specifically designed for products other than software, and provides graduated levels of restrictions of attribution, sharing, commercial development etc. Finally, there is consideration of non-FOSS licenses from the classic proprietary license, to less restrictive licenses such as the Sun Community Source License and the Microsoft Shared Source Initiative.

Of interest is St. Laurent's practical advice concerning legal issues in FOSS licenses. Unlike less free licenses, with FOSS the licensor may not even know who the licensees are and the variety of licenses and the different rights make for some challenges in distribution of a package. The author notes that that the openness of FOSS licenses means that, to a large extent, they are self-regulating, and where they are not (e.g., derived works clauses) that groups such as the Free Software Foundation has policed the (L)GPL with success, an unsubstantiated claim. Another issue that is raised with regards to derived works is that if developers use a license with derived works license, but do not include it, then they run the risk of having their intellectual property returns from their contributions voided (St. Laurent, op cit, p154-158).

2.3 An Excursus: Church-Turing thesis and Software Patents

Named after the mathematicians, Alonzo Church and Alan Turing, the Church-Turing thesis is the hypothesis that a natural numbers function can be calculated, if and only if, it is computable by a Turing machine, a system that controls data manipulation with sequential memory for storage. Whilst the debate is ongoing in mathematics, this discussion on the nature of computability and the relationship to logic, mathematics, and physical systems provided the foundations for modern computational systems (Copeland, Shagir, 2019).

This has a relationship to software patents. Whilst somewhat tangential to the discussion in this dissertation there is an interest in the grounding of intellectual property. Typically, a patent is a applied to the physical embodiment of a process. However, in some cases they have been applied to to software, the earliest probably being for "A Computer Arranged for the Automatic Solution of Linear Programming Problems", which was approved in 1966 (Closa, et al, 2010, p21). Whilst most jurisdictions reject the patent of "abstract ideas", however some (e.g., Canada, Australia, Japan, South Korea, United States) allow software patents where it is an "essential element" of the patent. Jurisdiction that reject software patents include the European Union, United Kingdom, New Zealand, and India. A software patent restricts, in software, the implementation of an process, method, or idea. Whereas with copyright the code is protected according to whatever encumbrances they wish to put on it (e.g., from a public license to a proprietary license), with a patent it is the results, not the code, that is restricted. Some companies with software patents suites use these holdings to target small scale developers with either massive legal fees for a breach of avoidance by paying a

significant (but less) license fee. In other words, a ransom-threat (Bessen, et al, 2011).

Programming is primarily about the implementation of ideas, but if the ideas are patented then software innovation cannot develop. "Software developers, working on their own code, can find themselves paralysed if they wish to be attentive to the existing patent system, or liable if they are not" (Lafayette, 2014). Distinguishing between "discoveries" and "inventions" (Klemens, 2006), it is pointed out that under the Church-Turing thesis that all software is effectively a discovery, as all software is trivially translatable to mathematics. At the same time however, it is argued that the state-machine, that is, the physical result of arrangement of the mathematics, ought to be patentable. In this manner a patent can still be sought and applied, however innovation need not suffer by a legal monopolisation of mathematics.

2.4 Licensing and Profitability

It is a fundamental truism that businesses must be profitable. How profitability occurs is the focus of the work of Porter, of which "The Competitive Advantage" (1985) is a foundational text for an enormous array of literature. From the outset, Porter speaks of the importance of competition, noting that strategy is about finding a favourable competitive position and the need to differentiate between the inherent and sustainable profitability of different industries and the relative competitive position within an industry. An analytical framework is provided (Porter, 1980), which describes the famous five competitive forces in an industry to a business, i.e., the bargaining power

of suppliers, the bargaining power of buyers, the threat of substitutes, the threat of new entrants, and, within an industry, the rivalry among firms.

Following this description Porter outlines "generic strategies" for achieving competitive advantage, based on cost-leadership (via high asset utilisation, volume cost effectiveness, value-chain control), differentiation of products or services, and focus (identifying and satisfying target market segments). The first two strategies are considered appropriate to large organisations, which have the resources to implement them, whilst the last is considered most appropriate for smaller enterprises. There has been significant criticism over Porter's claim that resource allocation demands that a first adopts only one strategy, with empirical examples of successful hybrid strategies (Baroto et al, 2012), and the limited choice of strategies, leaving out for example, entrepreneurial leadership. The critical point however, is that Porter's competitive advantage is actually about firms gaining monopolistic advantage and reducing competition.

Early attempts to apply the principles of competitive advantage are evident in "Sources of competitive advantage in knowledge-based industries: Microcomputer software" (Levine, 1992) and "Business Process Oriented Implementation of Standard Software" (Kirchmer, 1999). Levine explores whether knowledge-based industries have the same characteristics as manufacturing industries in this regard, especially in reference to illegal use and substitutes. Using extensive case examples, Levine argues that contrary to Porter's strategies that problem-solving and knowledge-based skills will have greater importance than volume and pricing, with information-goods having a consistent low fixed cost per unit, thus resulting in a strategic

orientation towards the legal protection of intellectual property claims as a priority, which will not just apply within the software and knowledge industries. Levine extends the Five Forces model to include "illegal" consumers and substitutes based on legal protections of intellectual property, and argues that the rapid technological change will lead to greater internal rivalry in industry. One curious aside to this is the recognition that upstream recommendations from software "pirates" may improve security and profitability (August, Tunca, 2008).

Some sixteen years later, these themes are explored from an internal, managerial approach, looking outwards (Webb, Schlemmer, 2008). Despite limitations in scope (small business only) the inquiry is both theoretically rich and with extensively empirical studies. The authors also take a perspective of competitive advantage primarily from economics rather than business, explicitly referencing Ricardian and Schumpeterian rents, where the former refers to differential economic rents related to the productivity of resources (originally applied to fixed resources, such as land), whereas the latter are economic rents collected by holders of an intellectual property from the period of time between introduction and general diffusion. The relationship between knowledge-industry licenses and the two types of rent tends towards the latter in dynamic utilities and the former in more stable technologies. Wealth-creation in a firm is more due to a firm's resilience by holding internal technological, organisational, and managerial knowledge, rather than engaging in the sorts of strategic actions that prevents new entrants, disrupts competitors, etc. Information technologies themselves cannot generate this sort of competitive advantage, although they are a necessary factor – a very

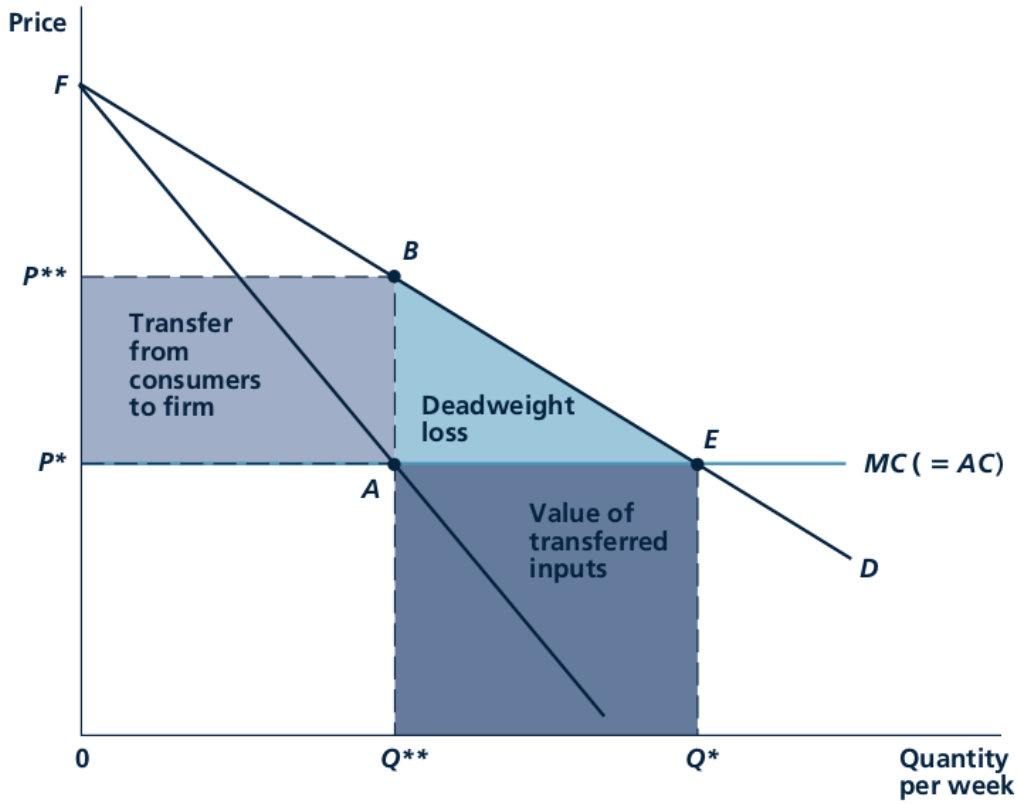
strong endorsement of information systems as a discipline. Further is the importance is the dynamic capacity of a firm to adapt. Competitive advantage thus comes from a complex of both being both resilient and dynamic, in both organisation and technology. An interesting variation on this themes is licensing choice in FOSS (Atal, Shankar, 2015) which correlates license choices with relative market size and reputation benefits.

Finally, for this section is a recent publication that takes technological architecture as the competitive strategy (Hewitt, 2019). "Architecture" is meant here in the sense of a solution designer; the product managers determine what must be done, and the architect describes how to bring that objective into reality in a systemic manner. In the world of software, the critical characteristic is "solid", which can be interpreted as "simple and hard", such as with minimal viable programs (MVPs), in contrast to "complex and fragile", common in "feature-rich" software. Because the objective of business is to gain some competitive advantage, architecture and strategy increasingly need to coincide. Creation patterns are placed into a logical architecture, using logic trees, hypotheses, MECE at the highest level, PESTEL analysis, scenario planning, etc on the World level, SWOT, Five Forces, Ansoff Growth Matrix on the industry level, and so on. High level, and descriptive rather than empirical, the text associates particular strategies in a overarching framework. The approach of technological patterns for competitive advantage is enticing, but requires more convincing empirical evidence. Curiously, the issue of licensing for profitability is barely mentioned only in passing, the author preferring profitability via production rather than monopoly.

2.5 Institutional Economics and Imperfect Competition

Robinson's "The Economics of Imperfect Competition" (Robinson, 1933) remains a classic text for realistic economic modelling. Robinson argues that the model of perfect competition is a model too far abstracted from reality, and proposed an analytical approach to imperfect competition for greater accuracy, making extensive use of geometric modelling and calculus to determine the shape of such curves and degree of monopoly. This can be illustrated by an example of "perfect monopoly" for contrast as follows:

Figure 2: Perfect monopoly



Explanation: "A perfectly competitive industry would produce output level Q* at a price of P*. A monopolist would opt for Q** at a price of P**. Consumer

expenditures and productive inputs worth AEQ*Q** are reallocated into the production of other goods. Consumer surplus equal to P**BAP* is transferred into monopoly profits. There is a deadweight loss given by BEA." (Nicholson, Synder, 2010, p383)

Robinson also famously introduced the concept of "monopsony" to the economic lexicon, where a single buyer controls a market as the major purchaser and many sellers, often been used to describe certain labour markets. Obviously, Robinson offers no explicit comments on software licenses, although there is reference to licensed trades and patents (Robinson, 1933, p93), where the model of imperfect competition does apply. Importantly for this dissertation, the degree of deviance from perfect competition represents a proportional loss in aggregate production. At best, where an monopoly pays the full rent of scare factors, production may come close to perfect competition, but cannot exceed it (Robinson, ibid, p153-154).

"The Economics of Imperfect Knowledge" (Richardson, 1998) supplements this inquiry with a strong concern with investment decisions and risk management. Following Hayek on the knowledge problem, Richardson argues that planned coordination within a firm require management of capabilities as complementary activities through different subsystems, highlighting another weakness in absolutist market perspectives; if markets were perfectly efficient, firm-based planned coordination would not be necessary. This observation was also made as bounded rationality (Simon, 1972) and institutional structures (Coase, 1992, see also Gabszewicz, Thisse, 2000). The alternative for Richardson is market coordination, where the

increasing division of labour results in emergent general-purpose goods which yields economies of scale and which contributes to dynamic efficiencies and productive flexibility. It is certainly appropriate for this dissertation to consider operating system and general purpose business software to be a type of intermediate goods. Interestingly, Microsoft asked Richardson to review competition in the software industry, resulting on an economic analysis of the industry and suggestions for public policy, leading also to the relationship between competition, innovation and increasing returns. Richardson claims that where there is continuous development and innovation existing products are subject to displacement by new ones with a rate of investment equal to the lifespan of the product for normal profits. Pervasive increasing returns, although reducing competition in standard economic theory, becomes possible as market disruptions resulting in competition over needs but monopoly over products (p179).

Whilst some key theoretical economics texts provide critical insight to licensing issues, Fink's "The Business and Economics of Linux and Open Source" (Fink, 2003) addresses the issues in a practical manner, starting with consumer-level advantages (availability of skilled resources, cost, support, vendor independence etc) as well as disadvantages (application availability and maturity, business risk). A good summary of the technical definitions and the various forms of licensing in terms of the various degrees of openness is also provided. Fink argues that the purpose of open-source is not to release code to the public, but rather to bring collaborative development in-house (ibid p141) and, as a logical consequence, "the developer is the user" (ibid, p153), as it the engineers themselves that know, at a very deep level, the underlying

causes that are manifest as user dissatisfaction. Another consequence of this is that the talent in open-source is largely self-selecting, as is reputation among the community. Despite the title, this is primarily an business guide rather than providing a contribution to open-source economics. However, one particular example is given which has economic consequences, and that is a comparison between a software transition to open-source from a proprietary licenses as being like the transition of medicine patents to generics (Fink, op. cit., p160-169); this is an example of Schumpeterian rents, even if the author does not explicitly recognise it as such.

With some fifty-four chapters by individual authors and almost eight-hundred pages, only the most relevant selections from "Handbook of research on open source software" (St. Amant, Still, 2007) are selected. Three chapters (van Reijswoud and Mulo, Papin-Ramcharan and Frank Soodeen, Dudley-Sponaugle et al) are specifically concerned with open-source software in the developing world, one is concerned with political economy (Cunningham), and three with investment and revenue (Langdon and Hars, Puhakka et al, Rajala et al). With regards to the three articles on developing countries extensive case studies (South Africa, Uganda, West Indies, China, India etc.) across the three articles illustrate some common issues. Firstly, a perception of reduced licensing costs is tempered by lack of appropriately skilled staff. This is in itself a opportunity for developing technical self-reliance and local capacity development, but from the user perspective of productivity software, open source software doesn't have the same application features that users are familiar with, and hence there is strong resistance to change, especially given the ready availability of "pirated"

software for users (c.f., Levine, op cit, Vatiero, 2011), and existing contracts or OEM installations. To a large degree China, and to a lesser degree, India, has been able to circumvent these perceived problems through state sponsorship and directives. In nearly all cases, the adoption of free and open-source software has been most successful on the server and infrastructure level, rather than in productivity software.

The chapter on political economy notes the application of copyright law to source code as a property right and is therefore subject to the analysis of political economy. A political divide is observed, with advocates of closed source software tending towards corporate capitalism, whereas the open-source advocates tend towards liberal socialism. The divide also applies to software development models, differentiating economic return for labour input versus creativity and innovation (see also Raymond, 1988) and the notion of the "tragedy of the anticommons" (Heller, 1998), where too many exclusive ownership rights creates *underutilisation* of a resource. For GPL and derived licenses, it is noted that they act as both license (to use) and contract (requiring reciprocal rights in derived works), making it a subversive legal instrument. The claims of political allegiance are juxtaposed with the three chapters on open-source software business models which note high levels of customer-user involvement (CUI) as part of the value chain, making consumers co-producers, with lower-cost and more transparently bespoke applications. However, this has not translated into venture capitalist investments, except in some exceptions (e.g., Red Hat, IBM Linux services. MySQL). In general, income-flows are considered to have a high-value due to savings in time-to-market and licensing, however open-source companies

have a higher discount rate applied due to uncertainties. Several FOSS software models are described, including support and service, loss leadership (including conversion from proprietary to FOSS), brand licensing, and accessorising/"widget-frosting" (adding a proprietary layer on top of an FOSS layer) (see also, Popp, 2011, pp17-40). It is the latter than is considered most successful in commercial ventures (Red Hat, MySQL as examples) (Popp, ibid, 41-61).

2.6 Quality Software Engineering

In "A Practical Approach to Software Quality" (O'Regan, 2002) standards such as the Capability Maturity Model (CMM), Software Process Improvement and Capability dEtermination (SPICE), or the ISO 9000:2000, are reviewed in detail and illustrative examples on the need to deliver innovative software to customers at a competitive price with the desired quality on time. This is a significant challenge as software project overruns indicate, and quality management requires a diverse testing suite with associated life-cycle checkpoints (e.g., acceptance testing of requirements, system testing of specification, integration testing of design, unit testing of code). Acceptance testing should include, for example, developing customer satisfaction metrics across multiple criteria - a repudiation of common systems such as the Net Promoter Score (Reichheld, 2003). whereas software testing and inspection can use particular methods (e.g., Fagan, Gilb). In reviewing various methods, O'Regan's guide is detailed in description, but also includes case studies of impacts, provides practical

examples, highlights issues with different approaches, and includes statistical methods for metrics. The key elements are to have a commitment to quality, a method of implementing quality, and a formal design which can encapsulate states.

As a follow-up text "Mathematical Approaches to Software Quality" (O'Regan, 2006) elaborates on the last chapter of the previous text, by looking at the mathematical approaches in the IBM's Vienna Development Method (VDM), the Z-specification language of the Programming Research Group at Oxford University, Unified Modelling Language (UML), Communicating Sequential Processes (CSP)., et al. Core mathematical components of software engineering (e.g., set theory, propositional and predicate calculus, tabular expressions, statistics, matrix mathematics, state machines, graph theory) are referenced in detail. The focus is on how mathematical techniques that can assist software quality based on rigorous analysis and unambiguous statements which avoids an over-reliance on intuition, which have a low-level of statistical confidence. Rather than organisational examples, implementation examples are given in different software languages and the toolkits available. In both these texts, the question of licensing is not directly evaluated, however it is important to note that in FOSS projects their existence (or lack thereof) is transparent.

"The Cathedral and the Bazaar" (Raymond, 1999) is considered a classic example of technically-informed advocacy in the FOSS development world, based on observation of the Linux kernel and managing the fetchmail project. The title refers to two models of development in FOSS, "the cathedral", where the source code is available for each release but the project

is tightly managed, and "the bazaar" model which has a more open development process, arguing strongly for the latter model under the claim "given enough eyeballs, all bugs are shallow" (the more complete version is ""Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone." (p30). This proposition contrasts, in part, with the argument by Stallman et al. that FOSS is built on the same access principles as open scientific endeavours with code-reuse and refactoring (Williams, 2002). Raymond offers nineteen "lessons" for open-source software development, the most challenging being treating users as co-developers, the principle of "release early, release often" (p28) Combined the principles suggest a rapid development cycle which may start with many errors, but it quickly transformed into solid code through extensive rather than intensive testing. Finally, there is the argument that Brook's Law (Brooks, 1975) does not apply with an Internet-enabled open-source project, and that this will be fatal to closed-source projects: "the closed-source world cannot win an evolutionary arms race with open-source communities that can put orders of magnitude more skilled time into a problem" (Raymond, op cit. p54).

These claims have been subject to substantial criticism. Whilst code reviews are well-recognised as adding to software quality (e.g., Pfleeger, et al , 2003), making code public itself only increases the *potential* number of eyes and does not guarantee formal reviews. The critical vulnerabilities on the transport layer of OpenSSL known as "Heartbleed" was unnoticed for over a year, and the small volunteer team which manages this piece of critical Internet infrastructure have appealed for more publicly-supported positions

(Felten, Kroll, 2014). This is a classic "free-rider" problem; millions use the code, but few contribute to its maintenance. Likewise the claim that there is a pool of open-source developers "orders of magnitude" larger than closed-source developers is optimistic at best. Certainly however large-scale collaborative patching of errors occurs as it discovered, and most typically at the point of integration.

In "Facts and Fallacies of Software Engineering" (Glass, 2002) a more human-centric approach to the issue of software quality management is provided in preference to formal procedures. Whilst the style is unbearably chatty, there is a structure consisting of proposition, discussion, controversy, and extensive source material backing each claim. Being interested in factual statements the text is heavily orientated towards the practical rather than theoretical approaches, and has some particularly critical remarks on various attempts to market approaches contrary to reality with caustic wit.. There is explicit commentary on FOSS development, noting an assumption that users can be the error checkers, making use of debugging tools as often as non-FOSS programmers (p100, 175). This is in partial contrast to models which show efficient contributors encourage a FOSS model (August, Shin, Tunca, 2013).

Finally, "The Software Paradox" (O'Grady, 2015) takes a multi-disciplinary approach to a paradox; as software is becoming increasingly more important and more disruptive, "eating the world" as Andreessen (2011) vividly put it, the commercial value is declining. The evidence, derived on comparing market capitalisation of technology companies to software revenues of major software firms, is that competition between products is

increasingly becoming competition on service and implementation. Four main causes for the industry transformation are identified; (1) the rise of open-source software and opportunities for high-quality and rapid collaboration., (2) the emergence of various "(Software, Infrastructure etc) as a Service" managed-service models of provision., (3) the harnessing of what was "waste data" into business intelligence in order to better adapt to customer desires., and (4) the increasing strength of developers due to complexity and their preferences for open-source products on availability and convenience. The evidence of the transformation is broad and conclusive, as well as gradual and relentless: This provides opportunity for firms to adapt. The argument is not so much that it is increasingly difficult to generate revenue from software, but rather there is a change in how the revenue is generated, with diversification and adoption of the SaaS subscriptions is the most probable immediate solution.

2.7 A Literature Review Synthesis

The use of an extensive, multi-disciplinary literature review in this dissertation is necessary due to the subject-matter. The fact that there are disciplinary differences and even outright contradictions in perspectives from the different disciplines should be considered an opportunity to develop a new synthesis, and the extensive nature of the review provides insights for this synthesis that would otherwise not be available from a narrower inquiry. Incorporating the legal, business, economic, and engineering perspectives can lead to the following summary:

Software may be released with a variety of licenses, with varying degrees of permissiveness and contractual reciprocity. Among non-permissive licenses standard models of competitive advantage are extended to include an additional cost of license enforcement, and extension of license rights beyond Schumpeterian rents. Economically, this constitutes a "damaged good" in an attempt to derive monopolistic profits, which organisations with institutional strength will attempt to leverage as vendor lock-in. This results in reduced aggregate economic welfare and underutilisation. As the market develops there is an increasing need for general purpose goods which tend towards a single application and complex bespoke implementations which are more prone to competition. Permissive software licenses compete against non-permissive equivalent applications by reducing the marginal cost of reproduction of the information good towards zero deriving income from the bespoke implementations, and combining general goods for infrastructure and value-added service for context-bound content.

In the evolution of software deployments one witnesses some many examples of these general principles. Microsoft, long-dominant in the desktop and laptop operating system market, seemed to forget that the tendency to a single general product meant different products for different devices. Whilst various implementations of MS-Windows maintains around 90% market share for the desktop and laptops (as the next chapter will illustrate) , their presence on tablets and mobile 'phones is extremely small, with the market dominated by Google's Android and Chromebook (both of which uses a Linux kernel) and Apple's iOS (which uses a UNIX-BSD derived operating system). As a

fully open-source product, GNU Linux has never made inroads to the desktop or consumer device market, but instead has come to dominate supercomputing and server infrastructure, as both general software products and bespoke implementations. In productivity software, there has been a recent move to placing such software in cloud based services, as G Suite competes with difficulty against Office365, which has the advantage of familiarity, and whilst the fully-FOSS product LibreOffice remains a device product with minimal installation share. Finally, in browser share, a long competition between Chrome, Safari, Firefox, and Internet Explorer resulted with the former establishing a dominant position. Further confirmation of the general principles can be shown empirically through case studies and interviews; it is these examples that will make up the next two chapters.

Chapter 3: Research Methodology and Selected Methods

3.1 Research Methodology and Methods

A methodology is the general theoretical understanding of the appropriateness of a particular methods to a study (Howell, 2013), despite an increasingly common "pretentious substitute for the word method" (Frankfurter, 2007, p2). Methods are the particular tools employed in a research investigation; a methodology investigates and informs the choice of those methods based on paradigmatic principles. Combining ontology and epistemology, a methodological approach must consider what the problem is, before describing what tools to use in the task of investigation. History is a particular good example of distinguishing between methodology and methods, as the discipline of historiography is a discipline-based methodology. Historical sources (Thurén, 1997) are evaluated with relics having priority over narratives, with originality having greater credibility. A source close to the event is more credible than a distal one. A primary source, any information-artefact that was created at the time under study, is more credible than a secondary source, which builds and comments on primary sources. To minimise the possibilities of bias, non-partisan sources are more credible than pre-disposed advocates or detractors, although this begins to move into the interpretative historiography of hermeneutics, rather than source criticism *per se*.

To answer the research question the literature review drew upon material from the disciplines of law, business, economics, and engineering to

determine a synthesis and a general statement. From this literature review a hypothesis is drawn, that firms with an existing institutional strength in intermediate goods (e.g., operating systems and general purpose business software) and non-permissive licenses have an anti-economic imperative to derive Ricardian rents from software to establish monopolistic advantage. This imperative is in conflict with a marginal cost of of reproduction that tends towards zero, the presence of general infrastructure software with permissive licenses and reciprocal contracts that is preferred by engineers for efficiency, convenience, and collaborative purposes, and by service-orientated firms.

3.2 Selected Methods and Time-Scales

In order to test the core dissertation question and the propositions that arise from the literature review, appropriate methods have to be selected. In answering the core question trends in select software provide a primary empirical research data. Qualitative and interpretative analysis of this data, informed from the insights of the literature review, provides supplementary secondary research to the empirical data. In selecting the primary data the widest possible data collection sources are chosen; if interpolation is necessary that will form part of a secondary analysis. A second method is select case studies where there has been significant changes or challenges to software licensing regimes that affect the research question. The purpose of this is to elucidate the reasoning involved when a licensing change is made, and whether this fits the trends and the literature review suggestions. Again, while the factual content is primary and the interpretation secondary,

this is more qualitative rather than quantitative. Supporting these case studies is new primary research material based on interviews of system engineering staff involved the changeover of licensing modes. The purposes of the interviews is to determine whether the decisions were based on sound technological or commercial principles.

The time-scale for this data and analysis is from 2000 onwards, whenever possible. Apart from being a convenient date and of an length of time to ascertain trends, it can also be used to test a further hypothesis on whether changes the computation processing in hardware equates with a change in licensing. If this has any veracity, the changes from the dominant proprietary UNIX licenses in the early 2000s and prior to FOSS Linux licenses by 2005 should be replicated in other device sets as their performance improves. It also serves as a useful way-point in the development of mobile operating systems, illustrating a competitive period in the operating system environment before concentration.

3.3 Research Method Detail

Initial primary evidence is selected from trend in utilisation of software and license across particular products. These are differentiated by three levels; server-level software, desktop and laptop productivity software, and software for mobile devices (e.g., tablets and mobile phones). It is recognised that the sheer quantity of units will be increasingly greater from the mobile devices, to the personal devices, to the infrastructure systems. This itself is subject to significant empirical evaluation; as devices there were relatively few

mobile devices prior to 2000; today, they make up the majority of devices. The reason for the differentiation is to discern whether there is significant difference in the adoption of proprietary vis-a-viz open licenses according to technology function.

The first set, server-level systems, is differentiated into three subsets. The first subset is usage share of high-performance computing, taken from the metrics of the Top500 project. This whilst more related to scientific and research software, it is included as it represents both "big iron" as a trend where computing power is heading and in recognition that many business are engaged in research and development, evident by the modest number of commercial bodies found in this metric. The second subset is concerns itself with various Internet servers (DNS servers, email servers, web servers). A number of sources (e.g., W3Techs, Security Space) provide data of these servers. Finally, on the opposite end of the form factor, but still constituting server-level systems are embedded operating systems in infrastructure computers such as switches and routers, which have a variety of CPU targets; quantification here is taken from VDC surveys.

As personal devices, the second and third sets actually have equivalent subsets, although the form-factor is different and they are also differentiated by their time of mass-introduction. Whereas the first set is primarily a matter of infrastructure utilisation, which users only interact with indirectly, the second and third sets are a matter of primary interaction. The first common subset is operating system, on which all applications depend on. Whilst there is a myriad of potential applications to evaluate, two further subsets are evaluated, that is web-browser client and productivity software

suites. These are selected on the basis of user utilisation and in direct response to the dissertation question. Access by sites by web-clients collected by StatCounter have provided a rich source of data form the operating systems of desktop, laptops, and mobile devices. For productivity suites unfortunately only Spiceworks offers survey data that is methodologically sound, i.e., based on business-deployment and a large scale survey. In all cases the selection of software reviewed is taken from most common actual use; everyone who uses a computing device, for example, must use an operating system. Many common functions are carried out with a web-browser as a clients accessing websites which require a web-server. Everyone who receives email does so with a mail-server, even if accessed through a website front-end. Some common software that was excluded included file servers and directory services, as metrics were not available, ERP/CRM systems (which are highly dependent on general databases), and task specialist applications (e.g., accounting software, CAD/CAM, graphics or video manipulation programs etc) due to their lack of general use.

In addition to trend data and analysis, case studies are investigated which differentiate between the primary facts associated with the events and the secondary interpretative analysis. The case studies are selected where changes in the license regime occurred or was attempted, and that change has been part of a *conscious* decision on the part of the part of management, thus necessitating a qualitative inquiry. The interviews, too, have an association with the case studies in terms of providing this qualitative investigation. Trends and case studies provide examples of *what* happened,

but do not provide an qualitative explanation of *why* a decision wad made. A lack of change explicable through habit and herd behaviour endemic in the human species: "We think an act according to habit, and the extraordinary resistance offered to even minimal departures from custom is due more to inertia than to any conscious desire to maintain usages which have a clear function" (Lévi-Strauss, 1958)

A major issue is when a conscious decision is made to change the software that they are originally provided with or, if they make a change and revert back to a prior system. Conscious choices the critical driver here, especially when new technologies (mobile phones, tablets, cloud computing) are introduced where no prior habits exist. As a result, the preferred primary research data collection used here will be nine semi-structured interviews of target audiences where changes or new technologies exist. The structured component will seek answers to the most fundamental questions of why a particular software and license model was adopted (e.g., whether the engineering performance dominated, or financial issues, or the license model etc.) whereas the unstructured component will seek elaborations of interest that come out of the interview process. Interview subjects are selected from a combination of experience in depth (involvement in quantity) and technology type (involvement in quality). Interviews were conducted in a combination of synchronous and asynchronous methods, depending on geography. A considered decision has been made to conduct these interviews with the systems engineering staff rather than managers and decision-makers as they have a more detailed insight to transformational technologies, from which policy must ultimately be based.

3.4 Ethical Considerations and Method Limitations

As interview primary evidence is used in this dissertation, ethical questions are raised concerning data privacy and anonymity. Although the interviews constitute a relatively small part of the research data a high level of ethical consideration is nevertheless required. These individuals will be part of a qualitative and semi-structured interview process to elucidate the system reasons on why a particular decision was made and what alternatives were considered. In preparation for conducting these interviews the guidelines and case studies of the British Sociological Association have been consulted, along with the literature review of standards (Allmark et al, 2009., Sanjari et al, 2014).

Subjects are provided participant information which outlines the purpose of the study, why they have been invited, the format and procedure of interview, any risk or benefits, problem resolution, confidentiality, contact information. Participants are provided with verbal and written explanations of the research project and given the opportunity to ask any questions at all stages of the research process to ensure that they are fully aware of what is involved and how this affects them before they agree to participate. It is stressed that should participants are free to withdraw from the research process at any time, without need for explanation.

With the written permission of the participants, interviews will be recorded and summarised for qualitative research and analysis. All data obtained during this research project will be subject to the provisions of the

Data Protection Act 1998 UK and will be stored on the researcher's computer passphrase-protected files to which only the researcher has access and will not be shared, except with the specific participant on request. The passphrase-protection will use GnuPG (GNU Privacy Guard), with AES128 symmetric cipher. Participant information clearly states that all personal information will be given the strictest confidence, and this is re-iterated at the commencement of the interview.

All research has limited methods according the extensiveness and practical size requirements of a study. The use of multiple studies at scale for trend analysis is as good as can be acquired on the public record. A more thorough study could have applied very specific tests against legal or technological decisions, however that would be overkill for a trend analysis. The selection of case studies is admittedly subjective, but based on a qualitative assessment across the ontological categories used in trend analysis for relative importance. There is a high degree of confidence that the selected case studies do represent the most important examples in relation to the dissertation question. Finally, there are significant issues with the generalisability of the interview questions, in particular the numbers of subjects selected for the study. If there is a near unanimity in the subjects concerns then this could certainly act as an impetus for further and more detailed studies on the issue of a conflict between managerial decisions and engineering expertise.

Chapter 4: Data and Analysis

4.1 Introduction to Data and Analysis

As initiated in the introduction chapter and developed in detail in the preceding methodology and methods chapter this chapter begins with an review of various trends in software licenses according to installations on products differentiated by purpose and form used in a business environment. If indications from the literature review chapter are correct, the trends should see a move towards a consolidation of software for each service coupled with monopolistic actions (c.f., Richardson op. cit, Robinson op cit), along with the adoption of FOSS tools in infrastructure (O'Grady op, cit., Raymond op. cit.). This is followed with case studies in the change of proprietary forms of Unix in server-level systems to FOSS operating systems, the forking of open-source code when proprietary impositions are put into place (Joomla and MariaDB), the establishment of the ISO/IEC 26300 Open Document Format for Office Applications and the ISO/IEC 29500 OfficeOpen XML standard and the adoption of open-source productivity software by the city of council of Munich which switched from proprietary to open-source then back again (Porter, op. cit., Levine, op. cit.,). There is some expectation that, following the literature review case studies of user-experiences (St. Amant op cit., Still op. cit.) that familiarity will be considered of greater importance that engineering. Semi-structured interviews with engineers involved in where changes in license regimes serve to ascertain where the technical considerations are made and their perspective of user or management decisions.

4.2 Trend Data and Analysis

Datasets

The first set of trend data is server-level operating systems which consists of three subsets. The first is the high-performance compute infrastructure for scientific research from which commercialisation is possible, from which metrics from the Top500 project are used, taken from the November iteration of the bi-annual survey. It is noted that the Top500 is based on calculations of floating point operations which is, of course, not the only metric of computer performance. It does not, include the read-write speed to disk, for example. However it is a convenient single metric and is the most well-known. Another restriction is the metric is that it only counts publicly submitted systems. It is not considered probable that either of these elements will have an effect on the overall trend in the software licenses used.

The second subset is various Internet servers (DNS servers, email servers, web servers) which provide business and organisational infrastructure in an outward looking perspective. These use metrics from a number of sources, including W3Techs, Security Space etc. DNS Surveys are limited, initially from Daniel J. Bernstein's survey of 2000, Don Moore's survey of 2004, and the Centre for Applied Internet Data Analysis for 20050-2010. From this point onwards DNS surveys were no longer acceptable as it became common practise not to publicise advertise what software or version was in use for security reasons (Takano, 2001). This is not the case for mail server records where a lookup of MX records is possible; Security Space provides an comprehensive survey in terms of quantity (all known mail servers visible on the Internet) and over a time period from 2006 onwards.

Security Space also conducts the same sort of survey for web-servers, however, with an increasingly large group of "Others". Comparison with the W3Tech's survey suggests that is primarily Nginx and Cloudflare Server; the former is an open-source product, whereas the latter is a reverse-proxy provider, that can obfuscate the actual web-server software. For embedded systems there is surveys from Venture Development Corporation (2004, 2008, 2013, 2015); commercial and in-house operating systems are combined as a single category to ensure data consistency.

With regards to personal devices, from desktops and laptops in one set to various mobile devices such as mobile phones, tablets, and consoles in another, surveys are taken from the main activities on such devices. Quantification is achieved primarily through collection of data from website visits from personal devices. For desktop and laptop operating systems and web-browsers StatCounter Global Stats calculates metrics from over two million websites and ten billion monthly page views, with removal of 'bot activity. For desktop systems StatCounter's values are limited from 2009 and for mobile devices from 2012; the month of August is used for the latter, rather than January for the annualised value. For office productivity suites only the survey by Spiceworks in 2017 was considered suitable, with a useful differentiation between individual and cloud installations. Other surveys were limited to particular business sizes, or market share by revenue, or by cloud-deployment etc., which whilst interesting in a different context did not directly contribute to the research question. Determining office productivity software also has the difficulties of (a) multiple software installations on the same device (e.g., MS-Office and LibreOffice both being installed), (b) the lack of

ability to automate what is installed or in-use, and (c) local installation and utilisation of a productivity suite and of a cloud-based service (e.g., Office365, GoogleSuite, and LibreOffice all being used). The Spiceworks survey was the only survey identified that captured these issues to a satisfactory degree. In the metrics the highest value is used for each suite (whether local installation or cloud), the survey was of business use, and aggregate values may total to over 100%.

All surveys are rounded to one decimal place and exclude non-responsive data. Operating system or application metrics that have less one percent of market share are included in "others/unknown" (except for the HPC survey which is based on absolute quantity). "Others/unknown/blocked" includes any rounding effects. For annual evaluations where monthly reporting is used, January is used as the canonical month, with the exception of mobile devices as noted. None of these survey limitations has a significant effect on trend analysis.

Table 1: Top 500 HPC Systems by Operating System

	UNIX	BSD	Linux	Windows	MacOS	Others or Mixed
2000	427	16	54	0	0	3
2001	443	12	39	1	0	5
2002	412	12	71	0	0	5
2003	275	11	198	1	1	14
2004	179	10	305	1	2	3
2005	99	4	372	1	5	19
2006	88	3	376	0	3	30
2007	29	2	427	3	2	37
2008	24	1	439	0	1	31

2009	25	1	448	3	0	23
2010	20	1	460	3	0	16
2011	30	1	457	1	0	11
2012	20	1	469	3	0	7
2013	11	1	482	2	0	4
2014	12	0	486	1	0	1
2015	6	0	494	0	0	0
2016	2	0	498	0	0	0
2017	0	0	500	0	0	0
2018	0	0	500	0	0	0
2019	0	0	500	0	0	0

Data source: https://www.top500.org/statistics/sublist/

Table 2: DNS by Server Software

	BIND	TinyDNS	PowerDNS	Embedded Linux	MS-DNS	Others/Blocked
2000	74.5%	8.5%				17.0%
2004	70.1%	15.6%	1.9%		6.2%	6.2%
2005	81.3%					18.7%
2006	74.9%		4.4%	15.6%	4.5%	0.6%
2007	70.4%		6.6%	19.3%	2.7%	1.0%
2008	58.7%	4.5%	4.6%			32.2%
2009	73.8%	2.6%				23.6%
2010	53.2%	1.9%				51.3%

Data sources: Daniel J. Bernstein's survey of 2000 http://cr.yp.to/surveys/dns1.html; Don Moore's survey, 2004 http://mydns.bboy.net/survey/; Centre for Applied Internet Data Analysis for 20050-2010 (http://dns.measurement-factory.com/

Table 3: Email (Message transfer agent) by Server Software

	Exim	Postfix	Sendmail	MailEnable	MDaemon	Microsoft	Imail	Others
2007	18.5%	13.3%	32.1%	2.2%	2.4%	21.0%	4.1%	6.4%
2008	21.6%	15.8%	27.0%	2.5%	2.4%	22.2%	2.9%	5.6%
2009	26.6%	18.0%	22.4%	3.0%	2.5%	20.8%	2.0%	4.7%
2010	31.6%	19.9%	18.3%	3.1%	2.4%	19.1%	1.5%	4.1%
2011	37.9%	22.1%	14.5%	3.2%	2.2%	15.9%	1.1%	3.1%

2012	44.2%	23.4%	12.2%	3.2%	2.0%	11.8%		3.2%
2013	47.4%	25.6%	10.8%	3.0%	1.9%	8.5%		2.8%
2014	50.6%	28.0%	9.2%	2.9%	1.7%	5.5%		2.1%
2015	52.5%	30.3%	7.9%	2.7%	1.6%	3.3%		1.7%
2016	53.6%	32.8%	6.4%	2.4%	1.5%	1.9%		1.4%
2017	55.7%	33.1%	5.2%	2.4%	1.3%	1.2%		1.1%
2018	56.8%	33.8%	4.4%	2.2%	1.0%			1.8%
2019	56.9%	34.4%	4.2%	2.2%				2.3%
2020	57.0%	39.9%	3.9%	2.1%				11.1%

Data sources from: From SecuritySpace
http://www.securityspace.com/s_survey/data/man.20[06-20]01/mxsurvey.html

Table 4: Websites by Server Software (Security Space)

	Apache	Microsoft	Netscape	WebSite	Zeus	Others
2000	56.0%	26.8%	5.2%	1.4%		10.6%
2001	58.1%	28.7%	3.7%		1.0%	8.5%
2002	64.7%	25.6%	2.2%		1.1%	6.5%
2003	61.7%	22.3%	1.2%			14.8%
2004	69.8%	22.3%				7.9%
2005	72.3%	19.2%				8.5%
2006	71.9%	22.2%				5.9%
2007	73.2%	20.3%				6.5%
2008	73.6%	19.0%				7.4%
2009	72.1%	17.6%				10.3%
2010	72.0%	16.7%				11.3%
2011	71.0%	16.2%				12.8%
2012	68.8%	15.1%				16.1%
2013	68.3%	15.0%				16.7%
2014	64.9%	15.6%				19.5%
2015	58.5%	15.2%				26.3%
2016	53.6%	23.4%				23.0%
2017	44.5%	29.6%				25.9%
2018	46.3%	16.8%				36.9%
2019	45.9%	15.0%				39.1%
2020	39.5%	17.4%				43.1%

Data sources from: Security Space
http://www.securityspace.com/s_survey/data/20[10-20]01/index.html

Table 5: Websites by Server Software (W3Tech)

	Apache	Nginx	Cloudflare	Microsoft-IIS	LiteSpeed	Google	Others
2010	71.5%	3.9%		20.6%	0.6%	0.6%	2.8%
2011	69.7%	5.9%		20.1%	1.0%	0.7%	2.6%
2012	66.7%	10.1%		18.4%	1.4%	1.0%	2.4%
2013	63.9%	14.1%		16.8%	1.8%	1.3%	2.1%
2014	64.8%	15.6%		14.6%	2.0%	1.3%	1.7%
2015	58.8%	22.9%		13.3%	2.1%	1.3%	1.6%
2016	55.5%	26.7%		12.4%	2.3%	1.4%	1.7%
2017	50.9%	32.1%		11.6%	2.3%	1.3%	1.8%
2018	47.8%	36.4%		10.5%	3.1%	1.0%	1.2%
2019	44.6%	40.7%		9.0%	3.7%		2.0%
2020	42.1%	31.3%	12.3%	8.0%	5.5%	1.0%	0.0%*

Data sources from: W3Tech
https://w3techs.com/technologies/history_overview/web_server/ms/y
* Rounding error in original data.

Table 6: Embedded Operating Systems

	Free Linux	Commercial Linux	Other FOSS Real-Time OS	Commercial or In-House OS
2001	15%			85%
2004	15.5%			84.5%
2008	23%			77%
2012	56.2%	6.3%	7.6%	29.9%
2017	64.7%	5.0%	5.5%	24.8%

Data source: VDM reports, in bibliography

Table 7: Desktop Systems by Operating System

	MS-Windows	MacOS	Linux	ChromeOS	Android	Unknown/Others
2009	95.4%	3.7%				0.9%
2010	93.8%	5.2%				1.0%

2011	92.0%	6.6%				1.4%
2012	89.6%	7.3%			1.7%	1.4%
2013	91.0%	8.0%				1.0%
2014	88.9%	8.4%	1.1%		1.4%	0.2%
2015	88.2%	9.1%	1.5%			1.2%
2016	85.2%	9.0%	1.5%			4.3%
2017	86.4%	11.2%	1.6%			0.8%
2018	82.3%	12.8%	1.4%			3.5%
2019	75.5%	12.3%	1.6%			10.6%
2020	77.7%	17.0%	1.9%	1.5%		1.9%

Data source: StatCounter https://gs.statcounter.com/

Table 8: Mobile Device Operating System

	Android	iOS	KaiOS	Nokia	MS-Windows	Series40	Symbian	Blackberry	Samsung	Others
2012	25.1%	36.7%			1.0%	12.0%	10.1%	3.9%	5.2%	
2013	32.8%	37.2%				9.8%	6.8%	2.8%	3.8%	
2014	40.0%	33.6%			1.8%	9.3%	3.5%	2.5%	3.3%	
2015	52.3%	30.3%		1.2%	2.0%	4.71%	1.4%			
2016	62.0%	25.1%		2.2%	2.0%	2.3%				
2017	67.8%	24.0%		1.4%	1.0%					
2018	71.3%	23.0%								
2019	70.8%	25.5%	1.0%							
2020	72.5%	26.5%								

Data source: StatCounter https://gs.statcounter.com/

Table 9: Desktop Webbrowser Client

	Chrome	Firefox	Safari	Edge	IE	Opera	Other
2009	1.4%	27.0%	2.6%		65.4%	2.9%	
2010	6.0%	31.6%	3.8%		55.2%	2.0%	
2011	15.7%	30.7%	5.1%		46.0%	2.0%	
2012	27.4%	24.8%	6.6%		37.4%	2.0%	
2013	38.1%	22.5%	5.1%		32.2%	1.2%	
2014	46.6%	20.4%	5.1%		24.6%	1.3%	
2015	51.7%	18.7%	4.9%		21.2%	1.7%	
2016	57.8%	16.0%	4.6%		16.0%	2.0%	
2017	62.1%	14.8%	5.3%	3.4%	10.5%		
2018	66.0%	11.9%	5.9%	4.1%	7.3%		
2019	70.9%	9.5%	5.2%	4.4%	5.7%		
2020	68.8%	9.8%	8.6%	4.7%	3.7%		

Data source: StatCounter https://gs.statcounter.com/

Table 10: Mobile Webbrowser Client

	Chrome	Safari	Samsng	UC Browser	Opera	Android	IEMobile	Nokia	Blackberry	Netfront	Dolfin	KaiOS	Others
2012	1.0%	35.6%		6.2%	15.5 %	21.7%		8.1%	3.7%	3.0%	1.0%		4.2%
2013	2.1%	36.4%		7.5%	11.9 %	27.0%		5.7%	2.6%	2.1%			4.7%
2014	8.4%	31.9%		9.3%	11.8 %	24.8%	1.7%	4.8%	2.1%	1.8%			3.4%
2015	26.7%	28.4%		10.0%	8.0%	18.6%	1.7%	2.3%					4.3%
2016	35.2%	23.4%		16.5%	9.8%	10.9%	1.7%						2.5%
2017	43.2%	22.2%	6.0%	14.9%	5.8%	5.5%							2.4%
2018	48.8%	21.1%	5.1%	14.1%	5.2%	3.2%							2.5%
2019	53.6%	24.2%	6.8%	6.9%	3.4%	2.2%						1.0%	1.9%
2020	60.2%	24.3%	6.1%	4.6%	2.1%	1.0%							1.7%

Data source: StatCounter https://gs.statcounter.com/

Table 11 : Productivity Suite

	Office/365	G Suite/G Docs	Open/LibreOffice etc	iWork
2017	82%/53%	26%/16%	16%	3%

Data source: Spiceworks https://community.spiceworks.com/software/articles/2873-data-snapshot-the-state-of-productivity-suites-in-the-workplace

Analysis

In all cases, where trends are available, there is an increasing move towards FOSS licenses. This is immediately noticeable and most significant in HPC environments where proprietary UNIX licenses were over-taken by Linux in 2003, which then eventually became the exclusive operating system by 2017. Two issues of note; firstly, a period where "mixed" systems were of a small but significant number during the change from Unix to Linux. Secondly, following a cultural principle of familiarity (Levi-Strauss, ibid), the Linux systems were sufficiently close to the UNIX systems to make transitions relatively easy.

In DNS servers, as much as can be derived from the available data, the permissive-licensed software (BIND in Mozilla Public, PowerDNS is GPL, TinyDNS is public domain) is dominant and increasingly so. Among mail-servers, despite a significant use of the proprietary Microsoft-IIS systems at 21.0% in 2007, there was a sustained decline to under 1% by 2018. Other proprietary systems included MailEnable and MDaemon. All the other MTAs use some form of public license (IBM Public or Eclipse public for Postfix, GPL for Exim and Imail; Sendmail has its own license that is considered open source). In web-server software, FOSS licenses are also dominant and increasingly so. Again with a significant share at the start of collection metrics, Microsoft's IIS has a declining trend in both the Security Space and W3Tech surveys, although there is the recent modest rise of LiteSpeed, a proprietary-licensed Linux-based web-server is of note, although it also has an open-source license for a less feature-rich version. The major competition is now between two competing FOSS products, Apache and Nginx. Finally, in

embedded systems is quite evident that operating systems have moved significantly towards FOSS licenses; for embedded system producers there is enormous benefits in cost-efficiency, functionality, and interoperability.

In the web-browser usage share the two most notable trends in the collapse of the proprietary licensed Internet Explorer and the rise of Chrome, which whilst proprietary is largely based on the open-source Chromium project. Even Microsoft's new web-browser, Edge, is also largely based on Chromium. Firefox, with a stronger public license, has also experienced a significant decline, whereas the proprietary Safari has experienced modest growth. On mobile devices a there is a similar story with the rise of Chrome, although there is also notably the absence of Firefox and the presence of UC Browser, a Singaporean-Chinese proprietary mobile browser that is popular in south-east Asia.

In terms of personal devices there is a similar trend, but with less severity. There is not as much an influence in desktop and laptop operating systems where FOSS-licensed systems are of a very marginal share, although it is noted that the rise of the proprietary MacOS does have a degree of similarity with Linux, as a "UNIX-like" system. As mobile devices now represent the majority of computational devices, there is a notable trend to partially FOSS products in preference to proprietary licensed products. An initial period of several alternative competing proprietary products and software the field narrowed down to the closed-source iOS and and the partially open-source Android. The trend is clearly in favour of the latter.

Finally, and as an exception due to a lack of available trend (n=1), office productivity software is overwhelmingly proprietary (whether as MS-

Office/365, GoogleDocs/Suite, or iWork), whether as local installations or as a Software as as Service (SaaS) through the cloud or local network servers. Indeed, it is in the SaaS cloud environment that proprietary software is sandbagged against other strong FOSS trends in other environments, and LibreOffice Online has not developed to the same extent as other productivity suites, lacking a global cloud infrastructure. Whilst there is incredible advantage to distributed productivity software in the cloud in terms of access and collaboration (of which FOSS development tools such as git certainly illustrate) this does come at cost based on network bandwidth and latency.

These trends broadly concur with the aggregated theory from the literature review. Developers and engineers have a strong preference for FOSS-licenses for functionality and convenience which means that this would be come into effect most prominently in server level systems. In the case of embedded systems there is certainly a lot of indifference among most business-consumers on what operating system is running, but among producers of such goods the FOSS model has become primary, again satisfying the primacy of the trend in the engineering space. As expected, this does not apply as strongly in the space of personal devices. Familiarity in operating systems and office productivity software has meant that FOSS revolution has not penetrated as far or with the same extent as in the server environment, although interestingly there is some confirmation of the hypothesis that software trends in the most powerful computer systems does percolate, albeit to a lessened degree, to the personal devices. The differentiation of personal devices has been justified by the results, which clearly shows that being a personal device itself by no means ensures all

devices will follow the path of familiarity; Microsoft's failure to recognise this with the rise of mobile devices was quite extraordinary, and provided a huge opportunity for both Google and Apple. It can also be suggest that this confirms with prior theories in the literature review (Richardson, op cit) which argued for a trend towards monopolisation in intermediate goods, but competition in needs. Microsoft's recent moves to protect flagship proprietary products with provision of a subscription-based, SaaS model is currently increasingly successful due to the market penetration and geographical distribution, as there is no FOSS-based institution that can provide the sort of international cloud provision for this sort of model; Google's G-suite is the only possible competitor. From many years FOSS advocates for many years have argued that the cloud is "somebody else's computer". As much as this is true, it will not prevent an aggressive campaign by proprietary software vendors to push consumers into cloud-based proprietary application, even if these servers in the cloud run a FOSS operating system and deployment suite (Vaughan-Nichols, 2016) .

4.3 Case Studies and Analysis

Case Studies: Forking FOSS Software

Many FOSS products are the initiatives of businesses who seek to extend the level of community involvement (op cit., Fink, 2003). If the product develops in popularity the business gains increased market value. Sometimes, either through sale or an internal management decision, an

attempt is made to re-establish a less open license. Three particular examples of this behaviour are noteworthy, on the basis of significant utilisation. The first is Mambo Open Source, a content management system, the second OpenOffice, and the third, MySQL. In each of these cases the developer community forked the source-code, and created a competing product. As important warning for business practice, there are zero examples of a successful open-source product that has been turned to a successful proprietary license.

Mambo was established as a closed-source product by the Australian company Miro Construct Pty Ltd. In 2001 it released part of the product as Mambo Site Server under a GPL license, and shortly afterwards offers dual products, one under a proprietary license, and the other under an open-source license. In August 2005 Miro created the Mambo Foundation as the controlling organisation for Mambo, without sufficient consultation with the developer community. By the end of the month, the core developer group forked Mambo, and established Joomla. Whilst Mambo continued with a few security fixes until a final stable release in 2008, Joomla is still in very active development as a top content-management system, and has expanded its features with now over 8,000 free and commercial extensions.

In the case of OpenOffice.org, more commonly known as OpenOffice, it began life as StarOffice, a proprietary office-suite developed by the German company Star Division from 1985. In 1999 Star Division was purchased by Sun Microsystems and in 2000 Sun announced that it had become an open-source product, specifically to encourage community development and as a FOSS alternative to Microsoft Office, with the XML file format adopted as an

ISO standard in 2006. When Oracle acquired Sun in 2010, they reduced their contribution of developers. A majority of developers then left the project and forming The Document Foundation, forking the OpenOffice code, and releasing LibreOffice. Further, whilst under Sun the policy of OpenOffice was was governed by the Community Council, comprising of community members, under Oracle the Community Council was composed only of Oracle employees (Blankenhorn, 2010). Shortly afterwards, Oracle ceased development of OpenOffice altogether. In contrast, LibreOffice soon became the default on most major Linux distributions, is in active development, has been made available in 115 languages, with hundreds of extensions.

Finally, MySQL was established as relational database management system (RDBMS) by the Swedish company MySQL AB in 1995, which provided a core component of the software stack used in the majority of websites, including Facebook, Youtube, Twitter, and MediaWiki. MySQL was provided under a dual-license system whereby MySQL AB made MySQL available under the GPL, but also sold it with bespoke extensions under closed-licenses to clients when requested. In 2008 MySQL AB was acquired by Sun Microsystems which in turn was acquired by Oracle in 2010. On the announcement of Oracle's impending control over MySQL the code was forked with the new database named MariaDB. At the time MySQL was seen as a major competitor to Oracle's flagship database product. Since the takeover, Oracle has split MySQL into a community (FOSS) edition with less features and performance, and a enterprise (proprietary) edition, has closed public access to the list of bugs, and has limited developer involvement. MariaDB, whilst originally designed as a drop-in replacement with MySQL,

does have a growing list of incompatibilities, but a greater range of storage engines, and remains committed to community development and a FOSS license. Notably however, Oracle has not made MySQL closed-source - and it remains in active development and as the second-placed competitor to the Oracle database, but within the company Oracle. MariaDB, whilst having a respectable following and some major deployments, has not replaced MySQL.

Case Study: ISO/IEC 26300 and ISO/IEC 29500 Standards

In 1993 "SGML Open" was established as a non-profit body to encourage the adoption of the Standard Generalized Markup Language and entity exchange. This became OASIS Open in 1998 to be inclusive of XML and in the following year it was approached by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) to jointly develop a new set of specifications for electronic business communication. One of the many technical standards that it developed included The Open Document Format for Office Applications (ODF), an XML-based file format for spreadsheets, presentations, word processing documents, etc. From 2002 to 2006 an OASIS technical committee, with participation from a variety of diverse organisations, contributed various requirements to allow for a standard for various business documents that were transferable and could be archived for prosperity. This was published as an international standard as ISO/IEC 26300.

Notably absent from this process was Microsoft Corporation, who had a near-monopoly on business document software. Whilst the ODF was a freely available standard, Microsoft Office relied heavily on being closed-source. Over a two-year process from 2006 to 2008 Microsoft engaged in a two-part strategy. Firstly to partially adopt some of the requirements of ISO/IEC 26300, but mainly to have their own format, Office Open XML (OOXML), adopted a competing standard. In this they succeeded, firstly by the Ecma International consortium (ECMA-376) and then by the ISO/IEC (ISO/IEC 29500:2008). In achieving the latter, Microsoft and its employees engaged in significant lobbying, including a memo suggesting to the Swedish Standards Institute (SIS) suggesting it should vote in favour of OOXML in return for "marketing contributions" (Goldberg, 2007), and a Standards Australia delegate who was a developer for Microsoft was offered payments to promote the OOXML format in Wikipedia entries (Gedda, 2008). In Norway, an initial "no approval" vote in 2007 became an "approval" vote in 2008, after the administration of Standard Norge expanded the membership of the deciding technical committee; over half the members resigned in protest (Paul, 2008).

Whilst the morality of these actions is a candidate for analysis, for the perspective of this study what is interesting is how it fits the model for establishing and maintaining monopolistic advantage and monopolistic profits. Any organisation, when confronted with potential competitors and increased consumer power, will expend effort (in economic theory, equal to the long-term super-profit margin) to ensure that their monopolistic position is retained, regardless of the aggregate social cost; that is everybody else's problem.

Case Study: Linux on the Desktop in Munich

In 2003, as Microsoft's support for Windows NT 4.0 was coming to an end, the Munich City Council commissioned a report for a successor operating system and applications for their offices. The report offered two alternatives; migration to MS-Windows XP or using a Linux-based system. A majority of council members voted for the Linux and FOSS solution, which lead to the development of the "LiMux" distribution which became the first Linux desktop distribution certified for industry use (ISO 9241) by the Technischer Überwachungsverein (TUV) (Technical Inspection Association). Initially the project was delayed due to concerns relating to software patents. Finally, in late 2006, the actual migration began, starting primarily with web-browsers and email clients, then the office suite, and then the operating system itself, reaching the objective of c15,000 desktop PCs (of c18,000) by October 2013. However, in November 2017 the Munich city council, with advocacy by the Lord Mayor, decided to revert to MS-Windows by 2020 with all systems being replaced by MS-Windows 10. Entirely coincidentally, in 2013 Microsoft announced their willingness to move their German headquarters from the town of Unterschleißheim to Munich by 2016.

The initial decision to move from MS-Windows to Linux included a five-year average cost of the comparative transitions from MS-Windows NT to either Linux or MS-Windows XP. A particular strong case however was made over security concerns, not just from the possibility of unauthorised viruses but also the “call home” function in MS-Windows XP, whose use was not sufficiently disclosed (Linux Voice, 2014). To assist in the office process an

extension to OpenOffice called Woolmux was developed to ensure consistency in letterheads, form templates, document version etc, which itself was released as a FOSS product in 2008. The default distribution based was switched from Debian to Ubuntu, as whilst the former had impressive stability for servers, it was not up-to-date with newer desktop hardware. The Council was a highly heterogeneous body with some 22 units each with their own IT department, with approximately 300 common office software applications and an additional 170 specialised applications different council roles, resulting in some 50 different MS-Windows configurations.

Nevertheless, by 2013 the Council estimated that the switch from MS-Windows to Linux had generated a net saving €11m on hardware and software licensing costs (Heath, 2013) and had even resulted in a reduced number of issue tickets (Essers, 2012). Not everyone was happy however, with a new mayor declaring that the heterogeneous environment was not sustainable (Schwarzbuch, 2017) as some systems were mandated to use MS-Windows or virtualisation. However, a review of the IT system, conducted by Microsoft's Alliance Partner of the Year for 2016, (Accenture und arf, 2016) came the conclusion that the main IT problems confronting the City were due to the dispersed number and lack of co-ordination between the the number of IT departments, rather than the use of FOSS. The estimated cost of switching back to MS-Windows will cost €50m (Heath, 2017).

Of course, the reversion by the City of Munich occurs in a very different environment now compared to 2003, as many other bodies have moved in the other direction. Following the success of moving off proprietary office applications, the French National *Gendarmerie* started moving some 90,000

desktops from Windows XP to Ubuntu in 2007, saving €50 million on software licensing between 2004 and 2008 alone. As of 2018, approximately 90% of new desktop systems arrive without an operating system and 82% have GendBuntu installed. Another major example is army of the People's Republic of China (Gertz, 2009), have developed their own desktop distribution, Kylin, originally from FreeBSD, but since release 3.0, using Linux. It is now installed on 40% of Chinese Dell computers (Griffiths. 2015).

4.4 Interviews and Analysis

Two systems engineers were interviewed in the high-performance computing domain, both of whom were involved when the industry was undergoing the transition from various forms of proprietary UNIX to Linux. It was noted that during this period even organisations that had their own versions of proprietary UNIX (e.g., IBMs Advanced Interactive eXecutive or AIX), were actively encouraging the use of Linux instead, as AIX was considered an internal fiefdom within IBM. There were encounters of specialist code written in some proprietary UNIXs (in the interview case, Hewlett Packard UNIX, HP-UX,), however the high level of compatibility between Linux and UNIX meant that only minor modifications were required to the code base and re-compilation. The results of these interviews can be understood from elements of the literature review and other data analysis. From the perspective of engineering, the high level of compatibility with UNIX and performance can match with a competitor's approach to Porter's Five Forces; the proprietary UNIXs were unable to compete with any sort of

engineering advantage with a FOSS competitor. What is surprising, and not addressed in the literature review, was the degree of internal competition that can arise within a company and how a corporation can deliberately act against its own competitive advantage if it perceives that a wider market advantage can be gained.

Seven other interviews were conducted with senior engineers engaged in large scale server transitions with client interfaces, in four cases over dozens of sites, specifically in the field of database servers, a large data science and modelling team, and a number of general small-to-medium (SME) server implementations. Three were from a proprietary licensed system to a FOSS licensed system, and the other was from FOSS license to a proprietary. In the former, both were driven by a combination of both management and engineer needs. From a management perspective there was continuing concern with mounting license costs, slow improvements in the proprietary code-base change requests. Engineers raised issues with a lacking of ability to scale with the proprietary products, security issues, a concern with falling behind technological trends, and the ability to integrate with these trends. In each case there was some resistance from a user level, although these were invariably based on false assumptions (e.g., thinking that they would not be able to use their existing operating system on their client machine). The main issues faced in two of the four cases was the initial set-up costs and the initial cost of building expertise and training with the new tools. In the one case where the transition was from a FOSS licensed system to a proprietary system the decision was driven by management with a desire "to be more enterprisey", and to outsource skills rather than have them

developed in-house and engineering concerns of security or bespoke changes were largely ignored. In one the SME transition from a proprietary license to a FOSS license was reverted back to a proprietary license to have an standard approach with the corporate head office, despite the financial and technological successes of the FOSS implementation.

Again we find a situation where real-world implementations partially confirm the theoretical models expressed in the literature review, albeit these are matched with contradictory trends. The desire for cost-efficiency in the transition to FOSS licenses is evident enough, but in multiple cases there was examples of engineering perspectives (e.g., Raymond 1999 op. cit, Glass 2002 op cit). From the perspective of those implementing systems from a FOSS to a proprietary license system there is an application of a monopolistic perspective (Porter 1985 op cit), but as a recipient; for as much as one vendor may seek to reduce competition and to derive monopolistic profits (Robinson, op cit) there must also be a consumer who has willingly reduced their market power under the assumption that efficiencies can be gained by having the same user-interface and processes that are found elsewhere. This is, of course, a classic example of monopolistic leverage, with all that implies for aggregate economic wealth. It is notable that in these cases not only were the license costs obviously greater, but the possibility of bespoke requirements were ignored. Whilst it may seem that these could be just poor management decisions, if a monopoly position is achieved by the vendor then switching costs can be highly problematic, as the following illustrates:

> "The Windows API [application programming interface] is so broad, so deep, and so functional that most ISVs [independent software vendors] would be crazy not to use it. And it is so deeply embedded in the source code of many Windows apps that there is a huge switching cost to using a different operating system instead. It is this switching cost that has given customers the patience to stick with Windows through all our mistakes, our buggy drivers, our high TCO [total cost of ownership], our lack of a sexy vision at times, and many other difficulties.… Customers constantly evaluate other desktop platforms, [but] it would be so much work to move over that they hope we just improve Windows rather than force them to move. In short, without this exclusive franchise called the Windows API, we would have been dead a long time ago. The Windows franchise is fueled by application development which is focused on our core APIs"
>
> – Aaron Contorer, Microsoft general manager for C++ development, in an internal memo to Bill Gates, Feb 21, 1997. (European Commission, 2004)

An illustrative example for this was an interview with an IT manager and engineer for a large community organisation who attempted to introduce OpenOffice as an alternative to Microsoft office on client machines. Their motivation was similar to those of other engineers and managers who have introduced server-level changes; they wished to reduce existing licensing costs, they wanted greater software efficiencies, and they wanted a product that they could sustain, etc. However significant problems arose with minor

matters of compatibility with files sent and received from other organisations, including government. Whilst OpenOffice could export and import Microsoft Office documents, spreadsheets etc., formatting was often not a good replication of that from the proprietary alternative (LibreOffice has improved significantly since this attempt). Training costs was also raised as an issue, although with each new change to the look-and-feel of MS-Office requires re-training as well, whereas Open/LibreOffice has had almost the same interface for approximately twenty years. Overall, there is an issue of a monopoly producing sufficient habit and familiarity among a user-base, which will strongly resist change regardless of real advantages in engineering and system costs. It is notable that this has not occurred with the adoption of more open-source client applications which has stronger open-standards and are less reliant on file transfers to and from others (e.g., web-browsers). It is notable that this attempted implementation was shortly after the events in the ISO/IEC 26300 and ISO/IEC 29500 case study. With two international standards it seems that one a market monopoly position will retain or extend that position, rather than satisfy the objective of interoperability.

Chapter 5: Conclusion, Recommendations, and Evaluation

5.1 Review and Conclusions

The dissertation began with the history of the research topic noted that a legal change that allowed the introduction of proprietary licenses was quickly adopted by business, and whilst the common form-factor of computing devices changed from the mainframe, to the personal computer, and more recently to the mobile device, competing interests have led to a variety of license models. Engaging in a critical and macro-information systems approach, the dissertation engages in the application of information technology and institutional processes with insights from software engineering, intellectual property law, and economics. The research topic is justified that efficient and effective software will have on the lives of everyone affected by software, but also in the academic sense of providing a theoretical model that satisfies diverse disciplinary differences.

The literal review explored theories and examples that could contribute to these objectives. This included a study of software licenses and exploration of the functional differences between various proprietary to more permissive licenses. The excursus on the Church-Turing thesis made the fundamental claim that software is best covered by copyright ("discoveries") whilst physical devices by patents ("inventions"). On the issue of licensing and profitability the dissertation is heavily influenced by the theories of the various forces to establish monopolistic advantage, and their application to software which suggests a strong strategic orientation toward protecting and

extending intellectual property rights. An attempt to overcome the apparent conflicts between profitability, engineering, and aggregate wealth is made with the distinction between Ricardian and Schumpeterian rents, where the economics of imperfect competition can illustrate the effects within a context of institutions, leading to the trend towards monopolistation in intermediate goods, but competition in needs, which can results in a dual-license structure where there is a FOSS core and proprietary or bespoke additions. As the literature review explored the components of quality software engineering, the claim that FOSS was innately of higher quality was taken to task, with the recognition that there is a higher *potential* quality through transparency. The increased adoption is also witnessing a reduction in commercial value, of which SaaS subscription-models provides both a barrier and platform to move into areas where there is now FOSS dominance.

Aggregating the theories from the literature review presents a hypothesis that those firms with an existing institutional strength and monopolistic market share in general purpose software with non-permissive licenses are in conflict with a falling marginal cost of reproduction and collaborative needs of engineers and the desire for lower costs for service firms. This is a "race condition", where multiple parallel processes are racing to an end point, which will determine the system state. In reviewing the status of this race, multiple categories of computing systems are reviewed, deriving from needs and practise. The status of peak systems provided a test for the hypothesis that events in the HPC space would be followed in less powerful systems, more common in a business environment, differentiated by server-level software, desktop and laptop productivity software, and software for

mobile devices. As far as can be ascertained, this is the first time that such trends have been published in such an aggregated manner, even if it is clear that further survey data is required, especially in the field of productivity software. The data trends indicate an overall trend towards FOSS, firstly in the HPC environments and subsequently in server-level software, less so in desktop and laptop operating systems, but more strongly in web-browsers and mobile devices. In response, there is a current move towards "sand-bagging" proprietary licenses in the field of productivity software suites which also confirms with the literature review (O'Grady, op. cit., 2015).

Case studies were selected where there was major changes attempted in a software license regime. The first was based on forks of existing open-source code where changes in license or governance has resulted in new products. A second case study, a study of the establishment of two alternative ISO standards for office documents, illustrated the degree to which an organisation will act to retain a monopolistic advantage, which was also evident in the third case study of a change and eventual reversion in desktop environments. The chapter concluded with a number of interviews with senior engineers involved across different domains for changes in software licenses regimes. The results illustrate a strong engineering preference for FOSS solutions which succeed on the server level, but are less successful on a user environment due to familiarity. There is also a strong difference between the decisions of management who are technically aware and those who follow an “enterprise” solution.

A conclusion must return to the original research question; whether the future of business software is proprietary or free-and-open-source? Stated in

a simple manner, the research of this dissertation argues that there is an ongoing trend towards the open-source model over the past twenty years, most significantly in the high-quality HPC and server space, satisfying dissertation aims. For proprietary vendors the SaaS model of in a cloud-based subscriptions is a promising defensive line as it provides the advantages of economies of scale, avoids the problem in software of a falling marginal cost of reproduction in software, and leverages a competitive advantage in institutional infrastructure deployment. As this advantage may only be temporary (e.g., if an infrastructure competitor offers as FOSS alternative commercialised through bespoke support contracts) and ironically is often deployed using FOSS operating systems and cloud-deployment tools. Nevertheless, it is already evident that an aggressive contractual campaign is under way, which consumers ought to be wary of; they are, after all, reducing their market power when agreeing to such implementations.

5.2 Recommendations

There was both a theoretical and practical interest in this study, motivated by what might be conflicting desires for more effective and efficient software to be produced for organisational use, and for software developers and system administrators to be paid well, and for software source-code to be available for re-use, review, and elaboration. The possibility that the dissertation was an exercise in cognitive dissonance has not been ignored. Fortunately, in the course of the inquiries a number of solutions to these seemingly contradictory motivations have been identified, although they are

subject to significant institutional pressures. As has been recognised organisations with a monopoly advantage will protect that position and which requires a technological development which undermines the existing entrenchment or political interventions. With this caveat in place, two major recommendations on the macro-IS level are presented:

Firstly, the competing standards between ISO/IEC 26300 and ISO/IEC 29500 Standards ought to be revisited with an interest in resolving the contradictions between the two that do not allow for interoperability and backwards-compatibility. Where extensions beyond the universal standard are provided they should degrade gracefully, as is taught (albeit not always carried out) in both web-programming and with email extensions with Multipurpose Internet Mail Extensions (MIME).

Secondly, legislation and multi-lateral agreements need to be introduced when removes the current incentives towards permanently monopolies and super-profits through Ricardian rents to a regime where temporary monopoly rights are granted and *may* be exercised to the extent that they recover investments and generate normal profits, i.e., a Schumpeterian rent. This can be achieved through a copyright regime that ceases to be "all or nothing", which encourages institutions to promote extensions, but rather one which gradually tends towards public domain over time, following the same principles as open-source licenses today (e.g., academic only, attribution, public domain). This will allow for a correlation between innovation and investment, but also ensure that diffusion of technological advancements can take place.

From a micro-IS perspective, differentiation occurs from the perspective of the software vendor and software consumer. Whilst on one level the purpose of a software vendor in a capitalist political economy is not to make software; that is but a means to an end, and the true purpose is to extract monopoly profits. Nevertheless, some vendors should give serious consideration to the advantages that FOSS brings in terms of cost, convenience, collaboration, and efficiency. From the software consumer perspective, reduced cost, standards compliance, security, and avoidance of vendor lock-in should be crucial to decision-making.

4.3 Evaluation and Further Studies

In addition to the recommendations noted, there is a potential for further research. Directly associated with the dissertation itself is the need for additional data on the use of productivity suites in business environments according to form-factor and especially with the rise of competition between Office365 and G-Suite, which will intensify in coming years. As an elaboration on the dissertation material there is potential for applying the licensing insights and recommendations provided here for other instances of knowledge production, in addition to software. For example, in film and music, literature, technology patents, games etc., the question can be raised to what degree that monopolistic intellectual property rights have damaged economic and cultural development, and whether there is an alternative that provides for normal profit returns on investment from innovation whilst also encouraging diffusion.

Finally, there is also a contribution to information systems as a discipline. Unhappily located between computer science and business studies, IS can be strengthened by the development of internal and focussed micro-IS and a multidisciplinary and critical macro-IS orientations. Such a bifurcation, rather than weakening the subject, will strengthen both focus and independence. Indeed, it can be no other way; if focussed studies can identify massive failures and successes in information systems (Ballard, 2013), so too general studies must identify successes and failures of orders of magnitude. Evidentially, there is a great need to take up the task even if it challenges powerful institutions. Thus, it is with just a hint of institutional recursion, that this final quote is given:

> "Few men are willing to brave the disapproval of their fellows, the censure of their colleagues, the wrath of their society. Moral courage is a rarer commodity than bravery in battle or great intelligence. Yet it is the one essential, vital quality for those who seek to change a world that yields most painfully to change."
> – Robert Kennedy, Cape Town, South Africa, 1966

Word Count (excluding figures, tables, major quotations etc) =14972

Bibliography

Accenture und arf, (2016). *Externes IT Gutachten: Untersuchung der IT der Landeshauptstadt München* (LHM) (External IT Report: Investigation of the IT of the City of Munich (LHM)), City of Munich

Allmark, P.J., Boote, J., Chambers, E., Clarke, E., McDonnell, A., Thompson, A., Tod, A., (2009). *Ethical issues in the use of in-depth interviews: literature review and discussion*, Sheffield Hallam University

Andreessen, M., (2011). *Why Software Is Eating The World*, The Wall Street Journal, August 20

Apel, Karl-Otto, (1980). *Towards a Transformation of Philosophy*, Routledge and Kegal Paul, FP 1972

Atal, V., Shankar, K., 2014. *Open source software: competition with a public good*. Atlantic Economic Journal, 42(3), 333-345.

Atal, V., Shankar, K., (2015). *Developers' incentives and open-source software licensing: GPL vs BSD*. The BE Journal of Economic Analysis & Policy, 15(3), 1381-1416.

August, T., Tunca, T.I., (2008). *Let the pirates patch? an economic analysis of software security patch restrictions*. Information Systems Research, 19(1), 48-70.

August, T., Shin, H., Tunca, T.I., (2013). *Licensing and competition for services in open source software*. Information Systems Research, 24(4), 1068-1086.

August, T., Shin, H., Tunca, T. I., (2017). *Generating Value Through Open Source: Software Service Market Regulation and Licensing Policy*. Information Systems Research, 29(1), 186-205.

Ballard, Mark. (2013), *Universal Credit will cost taxpayers £12.8bn*, Computer Weekly, 03 June.

Baroto, Mas Bambang., Abdullah, Muhammad Madi Bin., Wan, Hooi Lai., (2012), *Hybrid Strategy: A New Strategy for Competitive Advantage*, International Journal of Business and Management. Vol. 7, No. 20, p120-133

Bessen, J., Ford, J. and Meurer, M.J., (2011). *The private and social costs of patent trolls*. Regulation, 34, p.26.

Blind, K., et al, (2005). *Software Patents. Economic Impacts and Policy Implications*. Edward Elgar Publishing.

Brooks, F., (1975). *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley

Checkland, P.B., (1988) *Information Systems and Systems Thinking: Time to Unite?* International Journal of Information Management, 8, p239-248

Coase, R. H. (1992). *The Institutional Structure of Production*. American Economic Review. 82 (4): 713–19.

Copeland, B.J., et al. (eds.) (2013). *Computability: Turing, Goedel, Church, and Beyond*, MIT Press.

Classen, H. W., (2017). A Practical Guide to Software Licensing for Licensees and Licensors (6th ed). American Bar Association.

Closa, D., Gardiner, A., Giemsa, F., Machek, J., 2010. *Patent Law for Computer Scientists: Steps to Protect Computer-Implemented Inventions*, Springer Science & Business Media

Copeland, J., Shagrir, O., 2019. *The Church-Turing Thesis: Logical Limit or Breachable Barrier?* Communications of the ACM, January 2019, Vol. 62 No. 1, Pages 66-74 10.1145/3198448

Essers, Loek (2012) *Munich Mayor Says Switch to Linux Saved Money, Reduced Complaints*, PC World

European Commission (2004), *Commission Decision of 24.03.2004 relating to a proceeding under Article 82 of the EC Treaty (Case COMP/C-3/37.792 Microsoft)*

Gabszewicz J., Thisse J-F. (2000). Microeconomic theories of imperfect competition. Cahiers d'économie politique, 37.

Gedda, Rodney (20 February 2008). "Microsoft developer joins Aussie OOXML standards delegation". Australia: Computerworld.

Gertz, Bill, 2009 "China blocks U.S. from cyber warfare". The Washington Times. 12 May 2009.

Glass, R.L. (2002). Facts and Fallacies of Software Engineering. Addison-Wesley.

Goldberg, Microsoft pressade partners att rösta ja, Computer Sweden, 2007-08-29 05:44

Griffiths, James (2015) A Chinese OS at last? More than 40 per cent of Dell PCs in China now running homegrown Windows alternative

Habermas, J., (1984) *The Theory of Communicative Action, Volume 1: Reason and the Rationalisation of Society*, Beacon Press, [FP 1981]

Heath, Nick., (2013). How Munich rejected Steve Ballmer and kicked Microsoft out of the city, Techrepublic.com

Heath, Nick (2017). Windows 10 switchover will cost Linux champion Munich €50m. Techrepublic.com

Howell, K. E. (2013) Introduction to the Philosophy of Methodology. Sage Publications

Klemens, B., 2005. *Math you can't use: Patents, Copyright, and Software*, Brookings Institution Press

Felten, Edward W., Kroll, Joshua A., (2014) "Heartbleed Shows Government Must Lead on Internet Security" = "Help Wanted on Internet Security", Scientific American 311:1 (June 17)

Fink, M. (2003). The Business and Economics of Linux and Open Source. Prentice Hall Professional.

Frankfurter, G., (2007), Theory and Reality in Financial Economics: Essays Toward a New Political Finance. World Scientific

Herrera, J.T. (2012). El conflicto en el Software: una aproximación a la incidencia del «fork» en los proyectos de software libre. ArtefaCToS. Revista de estudios sobre la ciencia y la tecnología, 5(1), 83-122.

Hewitt, E., 2018. Technology Strategy Patterns: Architecture as Strategy. O'Reilly.

Kirchmer, M. (1999). Business Process Oriented Implementation of Standard Software: How to Achieve Competitive Advantage Efficiently and Effectively (2nd edition, FP 1996). Springer.

Klemens, B., (2005). Math You Can't Use: Patents, Copyright, and Software. Brookings Institution Press.

Lafayette, L., 2014. *The Innovation Patent Review and Free Software*, Presentation to Linux Users of Victoria, July 2014

Linux Voice., (2014) How Munich switched 15,000 PCs from Windows to Linux

Leith, P., (2006). Software and patents in Europe. Cambridge University Press.

Levine, B. (1992). Sources of competitive advantage in knowledge-based industries. Doctoral thesis. University of Michigan.

Lévi-Strauss, C. (1958), "Structural Anthropology", Allan Lane

Nicols, K., (1998). Inventing Software: The Rise of "Computer-Related" Patents. Quorum Books.

O'Grady, S. (2015). The Software Paradox: The Rise and Fall of the Commercial Software Market. O'Reilly.

O'Regan, G. (2002). A Practical Approach to Software Quality. Springer.

O'Regan, G. (2006). Mathematical Approaches to Software Quality. Springer. Pfleeger, Charles P.; Pfleeger, Shari Lawrence (2003). Security in Computing, 4th Ed. Prentice Hall PTR. pp. 154–157

Popp, K.M. (ed), 2011. *Advances in Software Economics. A Reader on Business Models & Partner Ecosystems in the Software Industry,* Books On Demand.

Porter M.E, 1980, *Competitive Strategy: Techniques for Analyzing Industries and Competitors*, Free Press.

Porter, M. E., 1985. *The Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press.

Raymond, E.S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly.

Richardson, G. B. (1998). *The Economics of Imperfect Competition: Collected Papers of G.B. Richardson*. Edward Elgar.

Robinson, J., (1933). The Economics of Imperfect Competition. McMillan.

Rustad, M., (2010). *Software Licensing: Principles and Practical Strategies*. OUP USA.

Rowley, Jennifer (2007). T*he wisdom hierarchy: representations of the DIKW hierarchy*. Journal of Information and Communication Science. 33 (2): p163–180.

Ryan, Paul., (2008) *Norwegian standards body implodes over OOXML controversy*, Ars Technica, 2008

Sanjari, M., Bahramnezhad, F., Fomani, F.K., Shoghi, M., Cheraghi, M.A., (2014), *Ethical challenges of researchers in qualitative studies: the necessity to develop a specific guideline*, Journal of Medical Ethics and History of Medicine 7: 14.

Schwarzbuch, (2017), *Kein Glück mit der Software*„LiMux“

Simon, Herbert, 1972. "*Theories of Bounded Rationality*". Chapter 8 in C. B. McGuire and R. Radner, eds., Decision and Organization, North-Holland Publishing Company.

St. Amant, K., Still, B. (2007). Handbook of research on open source software: technological, economic, and social perspectives. IGI Global.

St. Laurent, A. M., (2004). *Understanding Open Source & Free Software Licensing: Guide to Navigating Licensing Issues in Existing & New Software*. O'Reilly.

Takano, Y., Ando, R., Takahashi, T., Uda, S., Inoue, T. (2013), *A measurement study of open resolvers and DNS server version*, Internet Conference 2013, Japan Society for Software Science and Technology

Thurén, Torsten (1997), *Källkritik* (Source Criticism), Almqvist & Wiksell, 1997

Vatiero M. (2009). *An Institutionalist Explanation of Market Dominances*. World Competition. Law and Economics Review, 32(2):221-6.

Vatiero, M. (2011). *The institutional microeconomics of positional goods*. Working Paper.

Webb, B., Schlemmer, F. (2008). *Information Technology and Competitive Advantage in Small Firms*. Routledge.

Vaughan-Nichols, S., (2016) *IT runs on the cloud, and the cloud runs on Linux. Any questions?*, ZDnet, June 24

Venture Development Corporation (2004, 2008, 2013, 2015), *The Embedded Software Strategic Market Intelligence Program*, VDC

Williams, S., (2002). *Free as in Freedom: Richard Stallman's Crusade for Free Software*. O'Reilly.