Nature of Polyexistentials:

Basis for Abolishment of
the Western Intellectual Property Rights Regime

And Introduction of
the Libre-Halaal ByStar Digital Ecosystem

ماهیت چند وجودی‌ها:

دال بر لغوِ آن چه که غربیها نامیده‌اند مالکیت فکری و معنوی

Mohsen BANAN

PLPC-120074.p3cel

First International Edition

Available on-line at:
https://github.com/bxplpc/120074

digitized, in which case it becomes a pure polyexistential. But, if the book was a rare historic manuscript, then the dominant aspect could have been its monoexistential dimension.

In the case of a given factory generated spoon, the dominant aspect is usually the material spoon which is monoexistential and not polyexistential instructions supplied to the numerically controlled machine that produced that particular spoon.

2.3 Monoexistentials

Monoexistentials are bound by their location. At any given time they exist in one and only one specific location. Material monoexistentials can be moved (transported) at physical speed.

2.3.1 Categories of Monoexistentials

In the context of monoexistence versus polyexistence, all that is material is monoexistential. Some non-materials are also monoexistential.

We categorize monoexistentials in the following 4 categories.

- Nature's Material Monoexistentials
- Man Made Material Monoexistentials
- Nature's Non-Material Monoexistentials
- Man Made Non-Material Monoexistentials

In the following sections we describe each of these.

2.3.1.1 Nature's Material Monoexistentials

Anything material is monoexistential.

Matter is the stuff around us. Atoms and molecules are all composed of matter. Matter is anything that has mass and takes up space.

A substance is matter which has a specific composition and specific properties. Every pure element is a substance. Every pure compound is a substance. For example, iron is an element and hence is also a substance. All substances are monoexistentials.

Chemistry allows us to categorize material monoexistentials into: chemical elements, chemical compounds and organic and inorganic.

2.3.1.1.1 Chemical Elements

Each stable chemical element is a monoexistential. This is illustrated in Figure 2.1.[3]

Our understanding of the periodic table itself is a polyexistential.

Our understanding of the periodic table allowed us to predict the existence of elements in nature prior to having discovered them.

Mendeleev used the patterns in his table to predict the properties of the elements he thought must exist but had yet to be discovered. He left blank spaces in his chart as placeholders to represent those unknown elements. The four predicted elements lighter than the rare-earth elements, eka-boron (Eb, under boron, B, 5), eka-aluminium (Ea or El,[2] under Al, 13), eka-manganese (Em, under Mn, 25), and eka-silicon (Es, under Si, 14), proved to be good predictors of the properties of scandium (Sc, 21), gallium (Ga, 31), technetium (Tc, 43), and germanium (Ge, 32) respectively, each of which fill the spot in the periodic table assigned by Mendeleev.

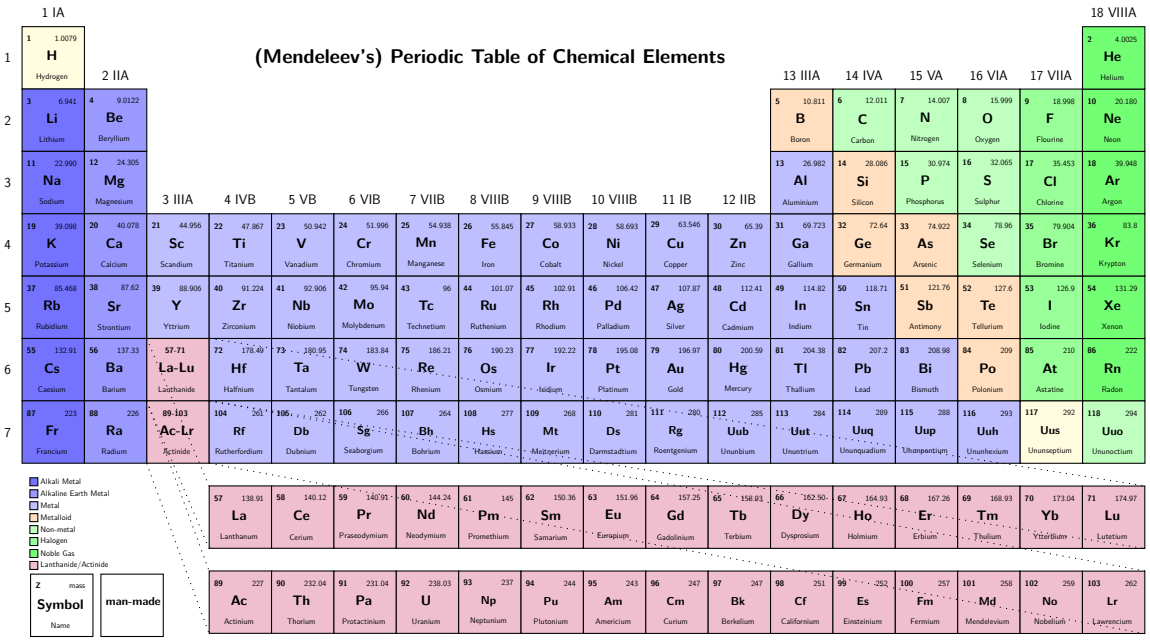

Figure 2.1: Periodic Table of Chemical Elements

Monoexistence of those undiscovered elements was independent of us. Our discovery created new polyexistentials. The monoexistential existed before being discovered.

2.3.1.1.2 Chemical Compounds

A compound is a substance formed when two or more chemical elements are chemically bonded together.

Chemical compounds form much of the matter that is around us.

Beyond basic physical chemistry and inorganic chemistry, when it comes to organic chemistry and biochemistry, at this time we are not adequately equipped to open those analysis. When it comes to DNA in particular, there are some polyexistence similar characteristics which we are not prepared to address at this time.

2.3.1.2 Man-Made Material Monoexistentials

A whole lot of the stuff around us is man-made.

Man-made monoexistentials involve a manufacturing process. The manufacturing process is a polyexistential but what gets produced can have a dominant monoexistential characteristic. When mass produced, each is monoexistential.

If the manufacturing process is relatively simple (say cutting of a tree), then we would consider the result of the manufacturing process monoexistential because the polyexistential component of the end result is insignificant.

If the manufacturing process is complex (say building a gun) then we would consider the result of the manufacturing process a mixed-existential. See Section 2.5 — Mixed-Existentials —, for details.

Strictly speaking one could take the position that all man-made material results are mixed-existentials. There are no pure man-made material monoexistentials.

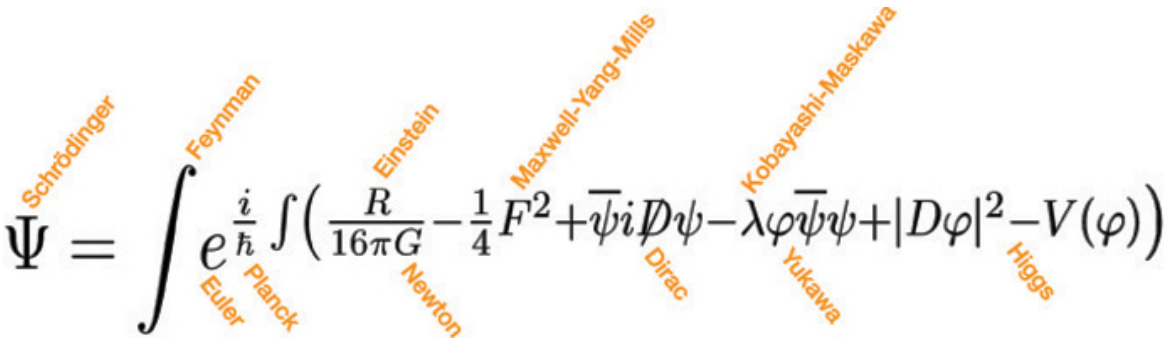

Figure 2.2: Unified Physics Equation With Inventors Labels

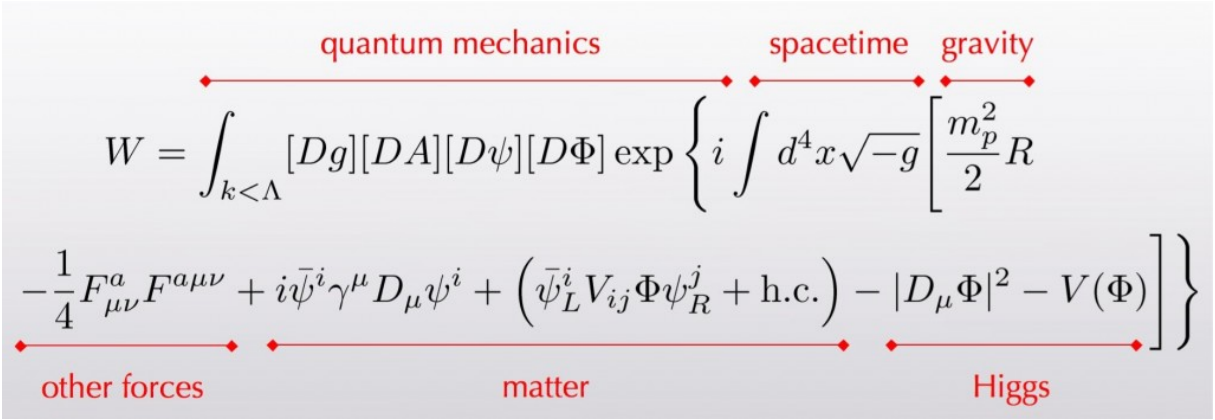

Figure 2.3: Unified Physics Equation With Subject Matter Labels

2.3.1.3 Nature's Non-Material Monoexistentials

Beyond matter there are other things in nature we experience. It is easy to recognize that matter is monoexistential. But it is a mistake to equate matter with monoexistentials. Some monoexistentials are not matter.

There have been many attempts in putting all of our experienceable understandings of the universe into one equation. Figure 2.2 is one such attempt.[4] This equation is annotated by attribution of aspects of knowledge to primary contributors.

All such forces and all such phenomena is monoexistential. They are bound by time and place and exist in singular.

Forces such as gravity and electromagnetic forces are bounded by location. So, things such as radio broadcasting and spectrum are monoexistentials.

Figure 2.3 is another such attempt.[5] This equation is annotated by subject matter labels.

The knowledge of such equations are polyexistentials.

2.3.1.4 Man-Made Non-Material Monoexistentials

Man-made non-material monoexistentials fall into two categories. Man-made physical non-material monoexistentials and man-made social monoexistentials.

Examples of man-made physical non-material monoexistentials are over the air television and radio broadcasts. These all involve energy, electricity, magnetism and waves and they are all bound by time and place.

Social monoexistentials involve creation of uniqueness and scarcities. Social structures and interactions often require uniqueness. As such, humans create non-material monoexistentials. Some examples of man-made non-material monoexistentials are: domain names and national identification numbers such as American social security numbers.

While many copies of an instance of a digital (polyexistential) exist, it is possible to create an association between a specific instance of that digital as its genesis (which we label as original) and its creator (which we label as originator or original assignee). Such associations can then be recorded in public ledgers. This allows for the tracking of all further assignments, so that at any given time it is possible to know the association between the original and the current assignee. This is the concept behind digital assets. An example of digital assets is Non-Fungible Tokens (NFTs). NFTs are typically used to represent digital art, collectibles and gaming items. They are stored on a blockchain and can be bought, sold, and traded on digital marketplaces.

2.3.2 Scarcity of Monoexistentials

Monoexistentials can be scarce or plentiful. Scarcity and plentifulness are relative concepts and depend on the environment and time. It is scarcity of monoexistentials that make them rivalry or non-rivalry.

2.3.2.1 Monoexistentials Rivalry Goods

"Rivalry Goods" is an economic concept.

In economics, a "good" is said to be rivalrous or rival if its consumption by one consumer prevents simultaneous consumption by other consumers.

In general terms, almost all private goods are rivalrous.

A good can be placed along a continuum ranging from rivalrous to non-rivalrous.

2.3.2.2 Monoexistentials Non-Rivalry Goods

"Non-Rivalry Goods" is an economic concept.

Non-rival goods may be consumed by one consumer without preventing simultaneous consumption by others. A good can be placed along a continuum ranging from rivalrous to non-rivalrous.

Many examples of non-rival goods are intangible.

Some broad examples of Non-Rivalry Goods are: air, fish in the ocean, view, roads, national parks, television broadcasts, wind and sunshine.

Non-Rivalry goods are often confused with polyexistentials (e.g., Wikipedia and Jewish IPR analysis make that mistake). Introduction of the concept of polyexistentials fully eliminates this common confusion.

The concept of polyexistentials is a philosophical concept. The concept of Non-Rivalry Goods is an economic term. Basing economics as the primary basis for structuring human laws is wrong. Inclusion of IPR in the US constitution by businessmen (founding fathers of America) is another example of the confusion which amounts to an attempt in creating rivalry goods from polyexistentials – based on artificial scarcity.

Goods that are both non-rival and non-excludable are called "public goods." It is generally accepted by mainstream economists that the market mechanism will under-provide public goods, so these goods have to be produced by other means, including government provision. Polyexistentials are inherently public goods.

The Western IPR regime is the opposite of "Public Goods". In the US constitution we have government provisions creating artificial scarcity against the public good.

Chapter 3

Digital Polyexistence

Sometime in the 20th century humanity entered the digital era. The full emergence of digital technology in late 20th century and early 21st century has moved humanity into an arena where the dominance of monoexistentials ended. We now live in a world where polyexistentials impact nearly every aspect of life.

3.1 Digital: A Practical Pure Form of Polyexistentials

Digital as a practical pure form of polyexistentials permits us to use, apply and produce more potent polyexistentials far more easily.

The aspect of "digital" that we are focusing on in this section is not digital technology or specific digital capabilities. We are concerned with the meaning and ramifications of "being digital". Our focus is the digital as applied math. Digital perspective is that of looking at the world in discrete terms. Analog perspective is that of looking at the world in continuous terms.

Perhaps the most clear moment for our entry into the digital era can be considered the understanding of digital capabilities by the likes of Nyquist and Shannon. We can point to the event that established the discipline of information theory and the digital era, as the publication of Claude E. Shannon's classic paper "A Mathematical Theory of Communication" in July and October of 1948. By then basic physical laws of the digital world were generally understood. Based on that knowledge, we became equipped to convert most information into digital, transfer and broadcast polyexistentials over large distances and store and reproduce exact copies of information.

3.2 Basic Physical Laws of the Digital World

In this section we discuss the basic laws of the digital world that govern data and information (polyexistentials).

3.2.1 Digitization – Perfect Polyexistential Reconstruction

It is possible to convert some of what we can sense (e.g., sound and images) into digital form.

Such transformation involves sampling.

Sampling theorem says:

> A signal can be completely reconstructed from its samples taken at a sampling frequency F, if it contains no frequencies higher than $F/2$:

$$f_{max} < f_{Nyquist} = F/2; \quad \text{i.e.} \quad F > 2f_{max}.$$

This equation is referred to as the Nyquist condition for perfect signal reconstruction.

The lowest sampling frequency F at which the signal can be sampled without losing any information must be higher than twice the maximum frequency contained in the signal; i.e., $F > 2f_{max}$, otherwise aliasing or folding will occur, and the original signal cannot be perfectly reconstructed.

Human perception is limited, therefore achieving perfect capturing in digital form is possible.

For example, the maximum frequency that we can hear is 20KHz and sampling at above 40KHz is very feasible. So, audio can reliably become perfect lossless digital audio which can be digitally encoded, transported, distributed and encrypted.

3.2.2 Encoding of Information Content

In 1944, Shannon for the first time introduced the qualitative and quantitative model of communication as a statistical process underlying information theory, opening with the assertion that:

"The fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point."

With it came the ideas of:

- the information entropy and redundancy of a source, and its relevance through the source coding theorem;
- the mutual information, and the channel capacity of a noisy channel, including the promise of perfect loss-free communication given by the noisy-channel coding theorem;
- the bit—a new way of seeing the most fundamental unit of information.

In information theory, systems are modeled by a transmitter, channel, and receiver. The transmitter produces messages that are sent through the channel. The channel modifies the message in some way. The receiver attempts to infer which message was sent. In this context, entropy is the expected value (average) of the information contained in each message.

Based on the probability mass function of each source symbol to be communicated, the Shannon entropy H, in units of bits (per symbol), is given by equations the like of:

$$H = -\sum_i p_i \log_2 p_i \quad \text{(bits per symbol)}$$

So, at that point the basics of how information can be packed inside of the digital world were understood.

3.2.3 Transmition and Remote Copying of Digitals

Digital entities can be reliably and perfectly transmitted over distances through imperfect and noisy channels. When the received digital entity is stored on the remote end, we have remote copying. We draw a clear distinction between transmission of digital entities and remote copying.

The speed by which digitals/polyexistentials are transferred is very different from the movement of monoexistentials.

Digitals can be transferred at around 70% of the speed of light. The speed of light is about 300,000 kilometers per second or about one billion kilometers per hour. In contrast, the speed of the fastest airplane is about 3500 kilometers per hour. The surface of the earth at the equator moves at a speed of 460 meters per second — or roughly 1,600 kilometers per hour. The speed of transfer of digitals (polyexistentials) can be hundreds of thousand times faster than the speed of movement of monoexistentials.

route around the power of labor to interrupt production. The information vector enabled capital to draw resources from a variety of sources at short notice. The information vector enabled capital to develop productive resources remote from traditional working class communities, with their historic memory and capacity for self-organization.

In the short term, the vectoralist class was helpful to capital in its struggle against labor, but in the long run, it is trying to subordinate capital to itself. Take a look at the top Fortune 500 companies, or the top "unicorn" venture capital darlings of the moment. With a few exceptions, one finds iterations of the same thing: companies whose power and wealth relies on stocks or flows of information, which control either the extensive vector over space or the intensive vector of an archive of commodified information—so-called intellectual property.

Whether it is finance, tech, cars, drugs, food, or chemicals—often the big companies no longer actually make their products. That can be contracted out to a competing mass of capitalist suppliers. What the vectoralist firm owns and controls is brands, patents, copyrights, and trademarks, or it controls the networks, clouds, and infrastructures, along which such information might move.

The rise of the so-called sharing economy is really just a logical extension of this contracting out of actual material services and labor by firms that control unequal flows of information. This control via the information vector is becoming more granular, working now at the level of individual laborers rather than subcontracted firms. At first, the vectoral made capitalist firms subordinate. Now, where they can, the vectoralist class replaces them altogether with individual subcontractors.

Like all previous extensions of the abstraction of private property, this one too produces its own internal antagonist. And like all previous antagonists, it never appears in a pure and self-conscious form. Most peasants tugged the forelock and did what they were told, silently cursing the lord under their breath. Most workers settled for some job security and a weekend. Radical class-based movements are rare.

So it comes as no surprise that the hacker class is not particularly conscious or organized or antagonistic either. But its frustrations are real. The hacker class designs the information tools by which all human effort is controlled and organized by asymmetrical flows of information. The hacker can see her or his own job succumbing to this tendency in the end as well.

The organization of the activity of hackers is built into the form of code itself. Their efforts are compartmentalized and separated — black-boxed. They work on alienated tasks just as workers do. Only they do not work from clock-on to knock-off time. Even when they sleep they work for the boss. They might in some cases be well paid, but in many instances they are not. Their skills date quickly, and they are replaced by others.

Hackers won't necessarily respond to the vectoralist class in traditional labor movement terms. A strike would hardly be effective given that hackers can't shut down production. The most frequent forms of antagonism are more likely changing jobs, or stealing time on the job for one's own projects. Of course many dream of start-up glory, but that dream quickly tarnishes when the hacker gets to see firsthand who usually cashes out first in such schemes.

The significance of platform cooperativism is that it is a movement that can place itself at the nexus of the interests and experiences of both workers and hackers. Why not use the specific skills hackers have to create the means of organizing information, but use it to create quite other ways of organizing labor? Cooperatives have a long history in the labor movement; indeed, in their origins, they looked back to forms of peasant self-organization of the commons.

Why not re-imagine the cooperative on the basis of contemporary forms of information vector—but without the information asymmetries that are the basis of vectoralist class power? That seems like the thread of a political-economic project that both honors past struggles and also addresses the distinctive form of commodification in the age of the information vector as a private property relation.

The vectoral political economy is in many ways worse than the capitalist one. It gives the ruling class of our time unprecedented wealth amid growing poverty and despoliation. It enables that ruling class unprecedented flexibility in routing around strikes, blockages, or communal strongholds. It has made the whole planet appear as an infinitely exploitable resource at precisely the moment when it is also clear that the past products of commodified production are coming back to haunt us.

> And yet every advance in the abstraction of the form of private property also opens up new perspectives on what may be held in common, and how the common might counter-organize. The practical and conceptual experiments of platform cooperativism are a key moment in the advance of this counter-organizing agenda.

13.3.2 American IPR Capitalism Threatens Humanity

Many of the points made in the reproduced previous section are consistent with what we are saying.

The phase that we call American Monoexistential Capitalism, Wark calls "Capitalism as a second-order mode of commodified production". The phase that we call American IPR Capitalism, Wark calls "vectoralism" — a third-order commodified mode of production. Wark identifies (and refers to it properly) "so-called intellectual property" as part of the problem. We consider intellectual property as the root of the problem in American IPR Capitalism. What Wark calls the "hacker class", we call the "global software engineering profession".

Our objection to the positions of American academics and activists, the likes of Wark, is that they are not seeing the full picture. They don't dig deep enough and don't go far enough. Americans have not recognized the full scope and scale of the harm of their celebrated economic model. That harm is not just to themselves, but to all of humanity — as polyexistentials are non-territorial.

Sure, what we are observing is worse than capitalism. But it is not sufficient to just say that. We should go way deeper and much further, and in this book we do.

13.3.3 Containing the American IPR-Based Capitalism

Economic models are human creations for the purpose of bettering the human condition. But humans should not live just for the purpose of bettering the economic model.

Corporations are created and exist for the sole purpose of generating profit. Consider the equivalent of a person whose sole purpose was accumulation of profit and power. Would you not be calling him/her a psychopath? Such an entity should be well bounded and limited, otherwise it will destroy humanity. Based on this understanding, we draw a distinction between American capitalism of unbounded-corporations which we consider inhuman and bounded-corporations monoexistential capitalism to which we subscribe.

In monoexistential capitalism, the existence of the subject of profit is in singular. To a certain extent, this functions as a natural form of bounding and containment of the corporation. Extending monoexistential capitalism into the realm of polyexistentials in the proprietary and for-profit quadrant empowers the unbounded corporation to profit from the unbounded replicability of polyexistentials.

The Western IPR regime has resulted in the transfer of more power and more control to the unbounded Americanist corporations. The overwhelming majority of copyright and patents are controlled by corporations — not individuals.

Highly optimized economic models can destroy humanity. Economic models define human behavior. Money becomes everything. People start living for money. But economics is full of externalities and other aspects of life start being damaged. Before you know it, people become economic creatures. All of this has already happened in the context of American Capitalism. Before you know it, the rest of humanity will become Americanized. Humanity is at risk.

American Capitalism is not just an economic system. It is a value system. It is intertwined with extreme individualism (khodzadegi) and greed.

The economic model of bounding corporations to the halaal manner-of-existence of polyexistentials protects humanity in the context of a profit-oriented highly competitive polyexistential capitalism. This can be the foundation of an economic system that results in bettering the human condition and can protect humanity from corporations.

13.4 Economics Is Not Science

Unlike chemistry, physics, or medicine, economics is not a science. The pseudo-science of economics, uses the trappings of science, like dense mathematics, to pretend it is science. Real science is not ridiculed. In economics there are many ridiculed schools – many consider Reaganomics as "voodoo economics".

To the extent that certain disciplines and theories have been established in economics, they are mostly applicable to monoexistential capitalism. In modern economics, what we have is application of monoexistential dominated theories to the polyexistential dominated environment. This is a recipe for disaster.

These outdated beliefs remain the basis for economic policies that don't work anymore. In response to inflation they continue to raise the interest rate. Sure, that may have been effective in a monoexistial dominated environment, but it is no longer effective in a polyexistential dominated environment.

Economists need to focus more on discovery of fundamentals and less on policy. The fundamentals include the distinctions between economics of monoexsitentials, economics of mixed-exsitentials and economics of polyexistentals. These are governed by different laws of nature and thus demand separate economic theories.

American economists have yet to discover that economics must be subordinate to nature, humanity, law, and justice. In America all of that is completely backwards. American economists believe that economics is the main axis around which everything else should be structured.

Governance of polyexistentials should not be based on economics. This is what the Western IPR economic regime does.

Halaal manner-of-existence of polyexistentials should govern economics of polyexistentials.

13.5 Proprietary vs. Non-Proprietary; For-Profit vs. Non-Profit

A business or other construct may be characterized as either proprietary or non-proprietary. And it may be characterized as either for-profit or non-profit. Generally speaking, these characterizations are orthogonal. Thus, there are four quadrants in which a construct may reside. This is illustrated in Figure 13.1

Typically, the for-profit label represents self-interest orientation and the non-profit label represents public goods orientation. The Western IPR regime has created the proprietary model in the for-profit context. In Polyexistential Capitalism, there is no proprietary model anymore. Both for-profit and non-profit activities produce public goods.

We first briefly describe each quadrant and then focus on the non-proprietary and for-profit quadrant.

13.5.1 The Proprietary and For-Profit Quadrant

The business models for the *proprietary, for-profit* quadrant are well established. The Venture Capitalist business model resides exclusively within this quadrant.

Conventions and regulations for this quadrant are well established. Copyright and Patents are the norm in this quadrant. Venture Capitalists understand it well.

Historically, proprietary and for-profit have been very closely allied, so that the *proprietary, for-profit* model dominates conventions.

13.5.2 The Non-Proprietary and Non-Profit Quadrant

With regard to public research, there are well established, clear and mature procedures for supporting research in the *non-proprietary, non-profit* quadrant. The recipient organizations are typically .edu or .org entities, and the

resulting public-funded research comes back to society in the form of unrestricted, non-proprietary results and assets.

13.5.3 The Proprietary and Non-Profit Quadrant

In theory this quadrant should be empty.

With regard to research, supporting public research in the *proprietary, non-profit* quadrant makes no sense at all. Here the results of the research are shut off from the public in terms of both ownership and business: the results are privately held and make no contribution either to society or to commerce.

In practice, it is where most academics exist.

13.5.4 Operation in the For-Profit and Non-Proprietary Quadrant

The business models for the *non-proprietary, for-profit* quadrant are not well established. The Venture Capitalist constituency does not understand this quadrant, nor does it believe in it, or have any experience in it.

Business operation within the *non-proprietary, for-profit* quadrant is still very unusual at this time, and mature business models for this quadrant do not yet exist. Therefore, our own Open Business Plan may be considered the most complete business analysis of the non-proprietary model in existence today.

The notion of a non-proprietary construct, residing and operating within the for-profit sector, is new and different. Historically, the for-profit sector has been closely associated with proprietary ownership of assets. Hence the Internet Services industry and the likes of Blackberry as we see them today. Also historically, management of non-proprietary or public assets has been primarily associated with the non-profit sector. Hence the current orientation of the FOSS Movement, operating largely within the non-profit sector.

We propose a radical shift of the Internet Services industry from the for-profit, proprietary quadrant, to the for-profit, non-proprietary quadrant. In this space the entire software for an Internet service remains a communal public resource in the trust of the engineering profession, while service deployment is driven forward by the full force of for-profit commercial motivations.

This radical shift to the *non-proprietary, for-profit quadrant* causes a major industry reconfiguration, with significant winners and losers. The losers are the existing vested proprietary interests, whose economic hegemony vanishes. But the winners are the many more companies who can now enter the Libre-Halaal Software and Libre-Halaal Internet Services market —and the end-users. The impact is immense both in economic terms and in societal terms.

13.6 Libre-Halaal Polyexistential Capitalism

Thus far, we have emphasized the following key points:

- Western IPR regime is the wrong model for governance of polyexistentials
- Libre-Halaal is the proper model for governance of polyexistentials
- Recent recognitions of negative effects of American Capitalism reflect the symptoms of the IPR mistake

Earlier, we introduced Libre-Halaal as a replacement model for Western IPR. But we have not introduced an economic model to replace American IPR Capitalism.

So, now we want to draw the contours of what should replace American IPR Capitalism. Its short name is“Libre-Halaal Capitalism”. Its full name is: “Libre-Halaal Oriented Polyexistential Capitalism”.

The full text of Attribution-ShareAlike 4.0 International license is available at:
https://creativecommons.org/licenses/by-sa/4.0/legalcode

As such, where copyright applies, we use the CC BY-SA 4.0 license for Libre-Halaal Content.

In civilized societies that reject the Western IPR model and adopt the Libre-Halaal model instead, all public content is Libre-Halaal Content. This would be similar to having all public content be subject to the Attribution-ShareAlike license in the IPR regime.

15.2.3 Definition of Libre-Halaal Internet Application Services

First, we need to define what we mean by internet services.

Internet application services allow for remote execution of software and remote delivery of the results of that execution. So, to begin with, for the internet application service to be considered Libre-Halaal, the remote software that is executed must be Libre-Halaal.

Our scope here is internet application services at layer 7, not services related to internet connectivity at layer 3. For brevity, sometimes we refer to internet application services as internet services.

As mentioned in Section 4.6.3, internet services are Libre-Halaal Internet Services (Libre Services) if they have all of the following attributes:

1. Every software component included in the service must be Libre-Halaal software.
2. The software for the entire service must be Libre-Halaal software. The entire primary source code for the entire service must be available to all software engineers, so that the entire service can be reproduced.
3. All protocols used by the service must be transparent and unrestricted.

In a document titled:

> **Definition Of The Libre-Halaal Internet Services Label**
> **Defining Halaal Manner-Of-Existence Of Internet Application Services**
> **A non-proprietary model for delivery of Internet services**
> http://www.by-star.net/PLPC/180045
>
> http://www.libreservices.org

we provide definitional criteria for halaal manner-of-existence of internet services.

A primary intent of this definition is to allow for the software engineering profession to fulfill its guardianship responsibilities to society and humanity. One of these guardianship responsibilities is verification of claims by internet service providers against what the service actually does (based on analysis of source code of the software). Another guardianship responsibility is to produce more and better internet services in a collaborative environment. Another intent of this definition is to maximize accrued benefits from the internet services to society. These goals are in harmony with the nature of software as polyexistentials.

Use of internet services impact autonomy of the individual in ways that use of software does not. With software you are not restricted with the choices that the service provider has made on your behalf. Use of software is more convivial than use of internet services.

Use of internet services impact privacy of the individual in ways that use of software does not. When you run and control the software of your own email server, your emails are in your own exclusive possession. When you use Gmail, your emails are also in Google's possession. When you run a software on your own computer, the logs of what you requested and how the software accomplished your instructions are in your exclusive possession and logs of your activities are private. When you use an internet application service logs of what you requested and how the software/service accomplished your instructions are in possession of the service provider. The service

provider has the ability to monitor your activities. You may then have less privacy. Furthermore, when you use internet application services and interact with other individuals or organizations, logs and perhaps content and details of your interactions are also in possession of the service provider. As a result, the individuals with whom you corresponded may then have less privacy as well. When you use internet services, both you and those that you have been interacting with are potential subjects of surveillance at layer 7.

Surveillance capitalism drives the American proprietary digital ecosystem and loss of privacy and autonomy of the individual are considered natural by products of technology in Americanism.

In Libre-Halaal digital ecosystems, things are different. We protect against loss of privacy and autonomy by providing proper definitions for Libre-Halaal Internet Services.

Therefore, the definition of Libre-Halaal Internet Services also involves protection of privacy and autonomy of the individuals. To provide for such protections, we need to go beyond the halaal manner-of-existence of internet services and regulate internet application service providers.

Based on capabilities and services that internet services provide, we can identify different type of internet services. These include:

- Possession Assertable Libre-Halaal Services — Abstract real individuals. Email, Messaging, Autonomous Web Publication
- Content Syndication Libre-Halaal Services — YouTube like services
- Information Provider Libre-Halaal Services — Google Map and Google Search like services
- Group Collaboration Libre-Halaal Services — Github, Zoom like services
- Mediated Inter-Autonomus Communication Facilitation Libre-Halaal Services — Craig's List, dating, eBay like services
- Bazar Oriented Libre-Halaal Services — Amazon, AliBaba, Walmart-like services
- Locale Oriented Libre-Halaal Services — Uber, Airbnb-like services
- Social Oriented Libre-Halaal Services — Facebook, Twitter-like services
- Brokerage Oriented Libre-Halaal Services — booking.com-like services
- Escrow Oriented Libre-Halaal Services — escrow.com-like services

In Libre-Halaal digital ecosystems, each of these types of internet application services demands its own specific set of regulations.

Surveillance capitalism driven American proprietary internet application services are inherently centralized. The service provider sits in the middle, observes and controls everything on a large scale and exploits individuals on the periphery.

The type of regulations that are needed are very different for each type of Libre-Halaal Services. At this time, we focus on just two types that are being implemented in ByStar. Future updates to this document will include more details with respect to regulations of each type.

Possession Assertable Libre-Halaal Services (PALS) are the main building blocks of Libre-Halaal digital ecosystems, through which internet services are forced to be de-centralized.

15.2.3.1 Definition of Possession-Assertable Libre-Halaal Services (PALS)

The concept of Possession-Assertable Libre-Halaal Services (PALS) does not materially exist in the American proprietary digital ecosystem.

16.2.2.6 ByStar Convivial User Environments – Blee

Users experience ByStar Services through ByStar User Environments.

ByStar services can be accessed in a variety of ways. In addition to the traditional browser based model, ByStar provides for rich and deep Software-Service integration.

Initially we are focusing on two convivial user environments for ByStar.

Blee (ByStar Libre Emacs Environment) – [27] – is a layer above Emacs that integrates ByStar platform (Debian GNU/Linux) capabilities into Emacs and provides for integration with ByStar Services.

An overview of this User Environment is provided in:

> **Blee and BxGnome:**
> **ByStar Software-Service Continuum Based Convivial User Environments**
> http://www.by-star.net/PLPC/180004 — [27]

The deep integration between Libre-Halaal Software and Libre-Halaal Internet Services creates a Libre-Halaal Software-Service continuum, far superior in capability to any Proprietary/Haraam Software/Service combination.

16.2.2.7 ByStar Content Generation and Content Publication Facilities

ByStar offers a rich environment and a number of facilities for content generation.

Autonomous Content Publication facilities are a well established feature of ByStar.

In the document titled:

> **ByStar Content Production and Publication Facilities**
> http://www.by-star.net/PLPC/180038 — [28]

we describe capabilities and features of ByStar content generation facilities and ByStar autonomous content publication facilities.

Autonomous self-publication can then be augmented by information aggregation federated services such as ByTopic, ByContent and BySearch.

16.2.3 ByStar Architecture Principles

The ByStar Digital Ecosystem is driven by a set of engineering architecture principles. We summarize some here.

16.2.3.1 Tools for Conviviality

Our primary criteria for software component selection and service design is “conviviality”.

By conviviality we refer to the concept of “Tools for Conviviality” as Ivan Illich introduced it.

In the document titled:

> **Introducing Convivial Into Globish**
> http://www.by-star.net/PLPC/120037 — [5]

we introduce the concept of ”Convivial” into Globish.

Briefly, in Illich’s words:

> Tools are intrinsic to social relationships. An individual relates himself in action to his society through the use of tools that he actively masters, or by which he is passively acted upon.
>
> To the degree that he masters his tools, he can invest the world with his meaning; to the degree that he is mastered by his tools, the shape of the tool determines his own self-image. Convivial tools are those which give each person who uses them the greatest opportunity to enrich the environment with the fruits of his or her vision. Industrial tools deny this possibility to those who use them and they allow their designers to determine the meaning and expectations of others. Most tools today cannot be used in a convivial fashion.

The dynamics of the Proprietary American Digital Ecosystem are such that they produce industrial tools.

The Libre-Halaal ByStar Digital Ecosystem is designed to fully reside in the Libre-Halaal-Convivial quadrant, [9].

16.2.3.2 ByStar End-To-End Philosophy vs Rise of the Middle

The dominant model of interaction between people and the model of access to information in the Proprietary American Digital Ecosystem is the centrally controlled Rise-of-the-Middle model – which puts the service provider at the center of all interactions so that it can exploit users and traffic information.

The ByStar model, in contrast, is end-to-end oriented and is based on the following principles:

- Make Services Autonomous Whenever Possible (peer-to-peer oriented)
- Invest and Focus on End-to-End communications facilities amongst Autonomous Libre Services
- When a Federated Service functions as an intermediary, limit its role to the bare essential of hooking the two ends. Thereafter, communications can be end-to-end.

16.2.3.3 Choice of Software Components

The Libre-Halaal software model is a flourishing creative environment, generating a constant stream of new and better software packages, duplicating and surpassing the capabilities of an ever-increasing portion of proprietary software territory. Indeed,for any particular item of functionality, there are typically multiple alternative free software packages available.

In this environment the model for implementation of By* service functionality is not one of original software development. Rather, it is a process of intelligent selection and integration of functional components from the Libre-Halaal software creative environment.

So, in creating By* our task has not been to write functional software components—in fact we have written almost none. Our main task has been to make careful engineering choices among the available free software components, and integrate these properly into the By* framework. In making these choices we consider not just the features and capabilities of each software component, but also the compatibility of the component within the overall By* architecture.

The main considerations in our choice of software components have been:

- Conviviality
- Scalability
- Libre-Halaal Mainstreamness
- ByStar Consistency

Virtually all the initial By* service functionality has been created this way. The following are some of the basic By* features that have been included by this process:

- Debian GNU/Linux.
- Base: djbdns, daemontools, ucspi, multilog, ...
- Mail: qmail, courier, spamassassin, ezmlm, ...
- Web: apache, zope, plone, geneweb, squirellmail, jquerymobile, galleria, ...

We will continue to select and incorporate additional software packages as these materialize within the free software environment. We will not create so much as we will harvest. Or to paraphrase the common industry dictum: *Good programmers write good software; great programmers reuse and integrate.*

16.2.3.4 Confidentiality, Anonymity and Privacy

By confidentiality we mean: ensuring that information is accessible only to those authorized to have access.

By anonymity we mean: the characteristic of any interaction where an individual protects his or her identity from being shared with another person or with a third party in the context of particular communications. In other words, people know what you do but not who you are.

By privacy we mean: the ability of an individual or group to seclude themselves or information about themselves and thereby reveal themselves selectively. In other words, people know who you are but not what you do.

ByStar Autonomous Services are designed to provide tangible confidentiality, anonymity and privacy on a large scale. All of Libre-Halaal ByStar Digital Ecosystem has this inherent design.

The basic assumption in the ByStar Digital Ecosystem is that all communications and traffic is subject to eavesdropping and traffic analysis.

Fortunately, the nature of digital information is such that it is easier to encrypt than it is to decrypt.

With nature on our side, ByStar Digital Ecosystem provides large scale countermeasures which include end-to-end data confidentiality and traffic flow confidentiality.

ByStar federated services are governed by transparency and well understood logging expectations and audit trail protections which are oriented towards preserving privacy.

All of this is in stark contrast to how confidentiality, anonymity and privacy are in the American Proprietary Digital Ecosystem. There, they have become a currency.

16.2.4 ByStar Central

The basic design of ByStar is very distributed. Services are generally autonomous and interactions are usually end-to-end.

This means that ByStar is centrally light. But there are some fundamental, infrastructural, and foundational organizations and services that are required at the center of ByStar.

The following infrastructure and foundational organizations have been put in place towards administration, guardianship, direction setting and facilitation of collaboration and growth of ByStar.

16.2.4.1 Libre-Halaal Foundation - non-profit, non-proprietary

Libre-Halaal Foundation is the non-profit legal entity that facilitates collaborative development, maintenance and administration of ByStar.

16.2.4.2 Neda Communications, Inc. – for-profit, non-proprietary

Neda Communications, Inc. is the for-profit legal entity that has developed Libre-Halaal ByStar Services. The core of ByStar software is subject to the Affero v3 General Public License and also the Neda Commercial License (dual licensed). Neda plans to profit from widespread usage of The Libre-Halaal ByStar Digital Ecosystem in a variety of ways.

16.2.4.3 LibreCenter.net

LibreCenter.net is Neda's data center. It is distinct and different from other data centers in that is built purely on Libre-Halaal Software. At this time most ByStar Services are hosted at Libre Center.

16.2.4.4 BySource.org

BySource.org is the Software Distribution Center for ByStar software in source form.

16.2.4.5 ByBinary.org

ByBinary.org is the Software Distribution Center for ByStar software in binary form.

16.2.4.6 ByStar Name and Number Assignment Authority

ByStar Name and Number Assignment Authority is responsible for central assignment of names and numbers for ByStar services.

Design of ByStar as an ab initio independent separate digital ecosystem permits ByStar to expand beyond the Proprietary American Digital Ecosystem. This is desired and possible for two main reasons. First, ByStar ideology may demand certain separations. Second, end-to-end purity of ByStar software-service continuum enables ByStar to do things that are not possible in the Proprietary American Digital Ecosystem.

ByStar's Public Key Infrastructure (PKI) and the possibility of a ByStar Alternative DNS Root, and ByStar Digital Currency are some examples.

16.2.5 Current ByStar Services and Capabilities

ByStar Services are vast in scope. They are designed to be ever growing. Basic structures of ByStar are in place and many services are built or partially built. The Libre-Halaal Services collaborative framework allows for ByStar to grow dynamically.

Thus far our focus has been in making sure that the overall architecture of the ByStar Digital Ecosystem is sound. We have been designing big and implementing gradually. A complete stable system is in place. It is now a matter of expanding and improving it.

In ByStar today, for email we don't use gmail, yahoo, msn, outlook.com, aol or other proprietary centrally controlled mail services. We use BystarMail. Similarly, for web presence, content publication, photo and video galleries ByStar has existing capabilities in use.

Here we provide a summary of where ByStar services stand today.

A snapshot of the organizations, services and software that form the ByStar Digital Ecosystem today are shown in Figure 16.2.

Libre-Halaal Foundation central resources are shown in violet in Figure 16.2. Neda resources are shown on the top. Current ByStarEntity generators are shown under the "ByStar Autonomous" label and ByStar federated services are shown next to them. ByStar software consists of three major layers, these are shown in the lower part.

The current status and growth of ByStar falls into four broad categories:

1. Current capabilities of ByStarEntity (ByStarServiceObject) – what any autonomous services are capable of offering.
2. Current span of ByStarEntity generators – what type of autonomous services (ByName, ByArtist, BySmb, etc) can be readily generated and supported?
3. Current scope of ByStar Federated Services.
4. Scale of user base – how many people are using ByStar?

16.2.5.1 Current Capabilities of ByStarEntity

Every ByStar autonomous service is anchored in a ByStarEntity. Every ByStarEntity can be provisioned to provide any of the current capabilities enumerated below.

- ByStarEntityIds and credentials – single password. [Unix account based]
- PKCS – ByStar Public Key Infrastructure (PKI) – Credentials.
- Autonomous VPN services and ByStar overlay networks. [openvpn based]
- Large amounts of autonomous disk space. [secure ftp based]
- Autonomous synchronization and version control facilities. [git – and also svn and cvs based]
- A Content Management System based website – with both public and private access. [Plone based]
- A conventional public website. [Apache based]
- Mobile websites. [jQuery Mobile based]
- Content publication services. [Plone based]
- A photo gallery. [galleria based]
- Genealogy web services. [geneweb based]
- Mail transfer service (MTA). [qmail based]
- Mail access service. [Secure Courier IMAP based]
- WebMail service. [SquirrelMail based]
- Mailing list services. [Ezmlm based]
- Mailing distributions. [Gnus based]
- LibreTexting. [qmail and emsd based]
- Matched User Environment Profile. [Blee based]

Various other capabilities are in the works. With the ByStarEntity model in place, addition of features is quite simple.

16.2.5.2 Current ByStar Services Sites

Current ByStar services sites are depicted in Figure 16.1.

ByStar services sites are organized by "types" in Figure 16.1. The *Autonomous ByStar Services* are PALS (Possession Assertable Libre Services). An example of *Autonomous ByStar Services* is ByName.net. The *ByStar Central* sites support the infrastructure of ByStar.

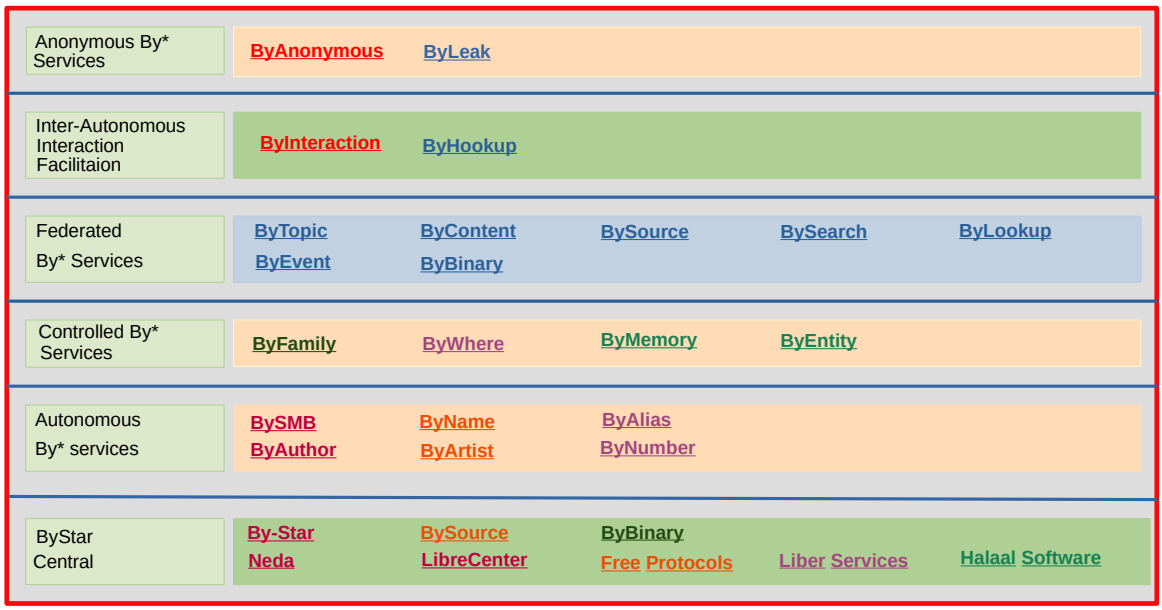

Figure 16.1: ByStar Domains Table

16.2.5.3 Current Status and Span of ByStarEntity Generators

A number of ByStarEntity Generators—the machinery required for fully automated creation of new service instantiations—are in place for a number of ByStarEntityTypes. Current ByStarEntity Generators are shown in Figure 16.2 under the "ByStar Autonomous" label. We thus have the ability to create unlimited numbers of new accounts in batch mode, or at any time we can "enable" the services, to permit self-service account creation by individual and business users.

16.2.5.4 Current Status and Scope of ByStar Federated Services

A number of sites are in place for facilitating inter-autonomous relations. Current Federated Services are shown in Figure 16.2 under the "ByStar Federated" label.

Our initial focus amongst federated service is those used for information aggregation. These include ByTopic, ByContent and BySearch.

16.2.5.5 Growth of user base: timing

An important consideration is the point at which we will begin to accept the burden of significant numbers of users.

In the case of a conventional service deployment there is typically a major emphasis placed on early and rapid growth of user base, to demonstrate demand and marketplace viability of the service, and lay claim to a particular portion of functional territory. This was the modus operandi during the dot con era, where claims of user base numbers were an integral part of spin-and-flip and pump-and-dump models. Some of those attitudes still persist.

However, we are not following this standard early proof-of-service approach. This may be appropriate for a conventional new service, where service functionality is the central and most critical issue. But for ByStar, a different timing strategy is required.

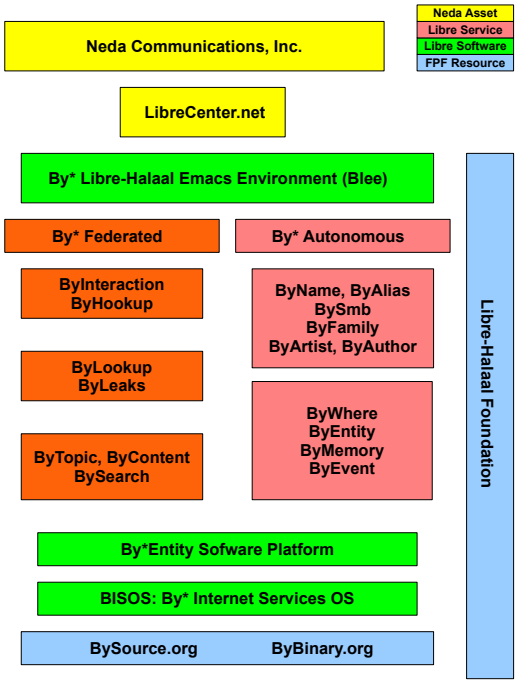

Figure 16.2: Libre Services Supporting Organizations

First, as a superset of numerous existing services, proof of service for By* in functional terms is already demonstrated by the Internet Services industry as it exists today. It is far more important to prove the model itself rather than its functional manifestations, and hasty creation of user base does little to accomplish this.

Instead, we have provided a coherent and complete description of the model in this and our other documents. The theoretical basis for the model is solid, and this will be clear to anyone willing to invest the time to understand it. In addition, a number of working By* implementations are already in place; examples are provided. Though the scale of usage remains small, these are sufficient to demonstrate the viability of the Libre-Halaal model and the ByStar design, and the value of the resulting services to paying clients.

But a far more important consideration is that installed base is very costly in terms of maintenance and support, and premature exposure to these costs can jeopardize the more critical work of building the underlying model machinery. Therefore, we will not take on the burden of user base until the time and/or context is right for this. This means either that we are fully ready to accept the associated costs of ownership, or that the user base is being taken on in an appropriate context, such as a suitable business partnership.

Under either scenario our strategy is the same: at the right time we will populate the services at large scale by mass creation of By* service accounts for large existing user bases.

16.2.6 Relationship With Existing Realities

The Libre Services and By* models are revolutionary, and can be expected to have a revolutionary effect on Internet usage. But these models are about service development and functionality, not about technological infrastructure. We are not reinventing the Internet protocols, or any other technical aspect of Internet operation.

What is being presented here is not a tear-down and rebuild operation.

Libre Services and By* imply no discontinuity, in terms of either technology or service deployment. The implementation model for Libre Services and By* is wholly evolutionary—there exists a continuous migration path from the proprietary model of today to the Libre model of tomorrow.

16.2.6.1 Relationship With the Proprietary American Digital Ecosystem

Based on ideology, the Libre-Halaal ByStar Digital Ecosystem fully avoids proprietary software and proprietary services. We simply avoid The Proprietary American Digital Ecosystem.

But, any and all of our services can be used in the Proprietary American model.

The core of ByStar software is subject to the Affero v3 General Public License and also the Neda Commercial License (dual licensed).

In a document titled:

> **A Strategy For Rapidly Becoming An Internet Application Service Provider**
> **Joining, Adopting and/or Licensing ByStar**
> **A Public Unsolicited Proposal**
> http://www.by-star.net/PLPC/180040 — [26]

We describe various options for those interested in joining, adopting and/or licensing ByStar.

16.2.6.2 Relationship With FOSS / FLOSS Movements

Libre-Halaal ByStar Ideology and FOSS Ideology have a great deal in common and we closely collaborate with our FOSS brothers and sisters, but the ByStar Libre-Halaal Ideology is distinct.

We invite our "Free Software" and "Open-Source" brothers and sisters to recognize that the "Libre-Halaal Software" model is a more complete model and that the "Libre-Halaal Software" label is a better label.

16.2.6.3 Active Private Parallel Digital Ecosystems – Example: NSA

What we want to do on a very large scale and in the open has been done in medium-scale in private.

For instance, the United State's National Security Agency (NSA) has created a separate parallel private digital ecosystem for its own use. NSA operates the private .nsa TLD; many NSA internal email addresses are of the form username@r21.r.nsa, mirroring the NSA organizational group structure. NSA has a particular ideology for its digital ecosystem which includes a large element of security, confidentiality and secrecy. The NSA, through use of its own particular software and services has created a completely different environment in parallel to the internet.

The precedence of such private parallel digital ecosystems combined with the proven power of Libre-Halaal software demonstrates that widespread realization of ByStar digital ecosystem is very viable.

16.2.6.4 Relationship With Piecemeal Privacy Oriented Software and Services

Some engineers kind of get it and have been trying to build various piecemeal privacy and autonomy software and services. Such efforts have always stayed limited in scope and scale. That is primarily for two reasons. First, the engineers have failed to connect with society. And second, piecemeal solutions don't work.

We build on these piecemeal privacy and autonomy software and services and bring them into ByStar as integrated and complete large scale services.

An example of a piecemeal privacy effort is PGP - Pretty Good Privacy. A bunch of engineers and technologists use it amongst themselves, but PGP never penetrated society at large. ByStar comes with Public Key Infrastructure (PKI) as an integral part of the service and equivalent of PGP is an inherent part of ByStar.

Another example of a piecemeal privacy effort is:
Tor `https://www.torproject.org`.
Tor attempts to accomplish traffic flow confidentiality just through redirection. Traffic flow confidentiality is an inherent part of ByStar which includes redirection plus layer 3 and layer 7 padding as well.

Proprietary vs Libre	Libre-Halaal Ecosystem	Proprietary Ecosystem
Laws, Values and Model	Patent-free	Patented
	Copyleft	Copyright
Software and User Env	Transparency	Secrecy
	Public ownership	Private ownership
Internet Services	Privacy, Autonomy	Surveillance Capitalism
Content	Guardianship	Exploitation

Table 16.1: Engineering vs. Business Polarization

16.4.2 War of Ideas – War of Words

ByStar is huge, powerful, and viable. But given the entrenched vested interests in opposition to it, the promotion of ByStar amounts to a kind of war. ByStar has the inherent characteristics to prevail in this war – we have moral superiority, intellectual correctness, and a construct that is viable in every respect: technological, economic, societal etc. But it is essential that all this be communicated effectively.

The ByStar Wars (to coin a phrase) will be fought on multiple fronts. But as a revolutionary movement, to a significant extent it will be fought as a war of words and ideas. This means that the movement is advanced effectively in words, defended against attack in words, and extremely forceful and effective counter-attack made against its detractors.

Please refer to `http://www.by-star.net/bxPublications` for a list of publications that we have felt is necessary for ByStar to be equipped with.

16.5 Joining ByStar

Successful Digital Ecosystems are dynamic. They grow and are ever evolving.

In the early stages of the evolution of ByStar we have adopted the strategy of limiting the size of our user base. A large active user base requires more support and is more difficult to maintain when service changes are frequent and when structural corrections may be needed.

At a certain point we would invite the public at large to use fully automated services to obtain ByStar accounts and start using ByStar. But that is not now. ByStarEntityGenerator web services such as ByName.net ByAuthor.net ByWhere.net, etc. are in place. However, at this time we screen account requests individually.

16.5.1 Individually

Any individual wishing to join ByStar can make a request and we usually activate accounts for these requests. Please see http://www.by-star.net/joiningByStar for details.

16.5.2 En Masse

Groups of users (Autonomous ByStarEntities) such as students or staff at a university or High School or a church can join ByStar en masse and obtain ByName services. Other ByStarEntity abstractions can also join ByStar en masse and obtain associated ByStar services.

In an article titled:

> **Joining, Adopting and/or Licensing ByStar**
> **A Strategy For Rapidly Becoming An Internet Application Service Provider**
> **A Proposal**
> http://www.by-star.net/PLPC/180040

We describe various options for those interested in joining, adopting and/or licensing ByStar.

Chapter 17

Technology of ByStar: BISOS (ByStar Internet Services Operating System)

In the previous two chapters, we described the requirements for a healthy digital ecosystem and its functionality. In Chapter 15 — Theory of Libre-Halaal Digital Ecosystems — we provided definitional criteria for the manner-of-existence of relevant parts of Libre-Halaal digital ecosystems. In Chapter 16 — The Libre-Halaal ByStar Digital Ecosystem, we described the functionality of ByStar. In this chapter, we focus on the technology of ByStar: the architecture, design and implementation of ByStar.

The engineering design and implementation of the ByStar Digital Ecosystem is documented in:

> **The Universal BISOS: ByStar Internet Services Operating System**
> **Model, Terminology, Implementation And Usage**
> **A Framework For Cohesive Creation, Deployment and Management Of Internet Services**
> http://www.by-star.net/PLPC/180047 — [30]

In that overview document, we present a vast model and process that can redirect the manner-of-existence and functionality of internet application services to protect humanity. In this chapter we include some extracts from that document.

Our audience for this book is all of humanity. Anyone who is willing to read and and who is willing to think independently. However, this chapter of the book is aimed primarily at fellow engineers and software-oriented readers and those who are curious to learn about the internals of ByStar. Here, we provide a simplified overview of BISOS. This overview includes the components we have selected, how we have arranged them, and the abstractions we have created to structure BISOS as an integrated platform. Additionally, we describe how ByStar uses BISOS. This overview does not aim to describe the inner workings of BISOS.

For those wishing to dig deeper into ByStar, we provide a reading road map in http://www.by-star.net/bxRoadmap. In ByStar Publications List: http://www.by-star.net/bxPublications, we provide pointers to ByStar related articles. These documents evolve as ByStar evolves, and the publications list will be kept up to date. The ByStar publications list is structured primarily for reference.

The internet services industry of today has three characteristics that greatly limit its capabilities, its usefulness and its health.

First, virtually all existing internet services are based on the traditional proprietary opaque model. So far, the FOSS movement has no formal presence within the services domain. The internet Applications Services Provider (ASP) sits in the center and controls and owns almost every aspect of our (user) communications.

Second, the current proprietary central model of American internet services has taken us to live in a world where our use of the network is mediated by organizations that often do not have our best interests at heart. This has led to the rise of surveillance capitalism.

Third, the internet services industry has arisen in a highly disorganized, unstructured way, driven by a multitude of uncoordinated commercial initiatives. The various industry capabilities have been built in an *ad hoc* manner, based on immediate business expedience, rather than by any sort of overarching engineering design. The result is the internet services industry as it exists today: chaotic, uncoordinated, and falling far short of its true potential.

The solution to these limitations consists of three main components:

1. We need to require the Libre-Halaal manner-of-existence for internet services. In other words the entirety of our public internet services should be internally transparent. The entire software of our own internet services should be Libre-Halaal Software (FOSS, FLOSS, Open-Source, Free Software).
2. We need a "Unified Autonomy and Privacy Oriented Digital Model" that is built on a "Universal Internet Services OS" and provides us autonomous services — that belong to us and are controlled by us.
3. We need a "Universal Internet Services Operating System (OS)" to bring consistency and cohesion to ower digital environment.

Here by "our" and "us" we are speaking of society at large when it is represented and protected by the Internet Engineering Profession.

Thus far we have been describing the contours of the problem and the contours of solution in abstract terms. We now present a specific implementation, that makes our proposal concrete.

17.1 Concept of the Universal Internet Services OS

The concept of an internet services operating system, or a common foundation, platform, and framework for the development of internet services, is not new. Proprietary internet service providers have their own proprietary and closed Internet Services OS. However, on the non-proprietary internet services side, this concept has not been formalized, structured, and cultivated. There is some precedence for this, and we can use this as a starting point.

Shortly after the internet started to impact society (say in 1994) and shortly after Linux became widespread, the idea of a server-side Internet Services OS appeared as "The LAMP Stack".

17.1.1 The Early LAMP Precedence

LAMP is an acronym that stands for "Linux, Apache, MySQL, Perl/PHP/Python". Packaged together, they create an application stack that is both free to use and open source which functions as a general purpose web server.

In 1994, the Common Gateway Interface (CGI) was introduced in CERN httpd, allowing for the server-side execution of code to create dynamic webpages. In a sense, this can be considered the genesis of internet application services. This made it possible to create a LAMP stack (the free general-purpose web server) using Linux, CERN httpd, and server-side programming languages such as Perl. However, it wasn't until the release of Postgre95 that a free database was available. Finally, in 1996, MySQL was released online, completing the LAMP stack.

Validity of the LAMP stack as a server-side web services generic OS was established through its widespread use in the late 1990s. Many of the dot-con era firms ran their websites with LAMP.

We recognize what is generally labeled "The LAMP Stack" as a very rudimentary Internet Application Services OS. LAMP had the following characteristics.

1. LAMP was a layer on top of Linux distributions
2. LAMP was a server-side stack

3. LAMP addressed a certain segment of internet application services. Its scope was websites development.
4. LAMP focused on a very specific profile of the Linux distribution — Apache and MySql.
5. LAMP focused on a specific programming language — one of Perl, PHP or Python.

Extending and improving the concept of LAMP can lead to the notion of "A Universal Internet Services OS".

Such an extension involves two dimensions:

1. An Internet Services OS should cover all internet services — not just web services.
2. An Internet Services OS should fully cover all sides — clients, servers, things in the middle and software-service-continuums.

By "Universal" we are referring to this notion of "covering all sides" from phones and tablets to mainframes and sever-clusters. This idea of "Universal Services OS" builds on Debian's concept of "The Universal Software Operating System".

17.1.2 Operating System, Internet Application Service and Digital Ecosystem

Almost everyone uses email. Email is a widely used application. To make things more explicit, we will use email as an example of an application service.

In Figure 17.1, let's consider email in the context of operating systems, internet application service and digital ecosystems.

First, let's take a look at what is happening in the proprietary universe. The five major American proprietary tech companies (Google, Microsoft, Apple, Facebook, and Amazon) have created five distinct digital ecosystems as competing enclaves. In Figure 17.1, ByStar and Proprietary American Digital Ecosystems, we are focusing on the first 3 and each of their office and email environments. These ecosystems are mostly separate and isolated from one another, and the economic model of these proprietary digital ecosystems is "Surveillance Capitalism". As such, when users sign up for a free email account, they are voluntarily forgoing much of their privacy. Sadly, the rest of the world is becoming Americanized through the American Internet. Each of these enclaves also have Mail User Agents that are fully integrated into their digital ecosystems, providing users with address books, calendars, time management and planning tools, multi-lingual authoring tools, and more.

Now, let's focus on the right side of this picture. On the non-proprietary side, based on the FOSS model, we have ended up with lots of components. We have Debian as a platform, we have Emacs as an editor-centered office environment and lots of great applications. But on the non-proprietary side we don't have anything that can reasonably be considered a digital ecosystem.

We need non-proprietary digital ecosystems. And that is what ByStar is.

In proprietary digital ecosystems, the scope of the operating system (Chrome, Android, Windows, MacOS) is limited to the usage-side. The service-side OS is unknown due to the proprietary services being opaque. The concept of an Internet Services OS is well established inside of each of the proprietary services providers. Each has their own and parts of their Internet Services OS are exposed to their "Cloud" users.

On the FOSS side, the scope of the LAMP style operating systems is limited to the service-side, with the usage-side being considered agnostic. ByStar and BISOS provide a powerful and universal solution, covering both the service-side and the usage-side.

17.2 Overview of BISOS and ByStar Digital Ecosystem

BISOS (ByStar Internet Services OS) is a reification of the abstraction of "A Universal Internet Services OS". ByStar is a concrete form of the abstraction of "A Unified Autonomous Digital Ecosystem".

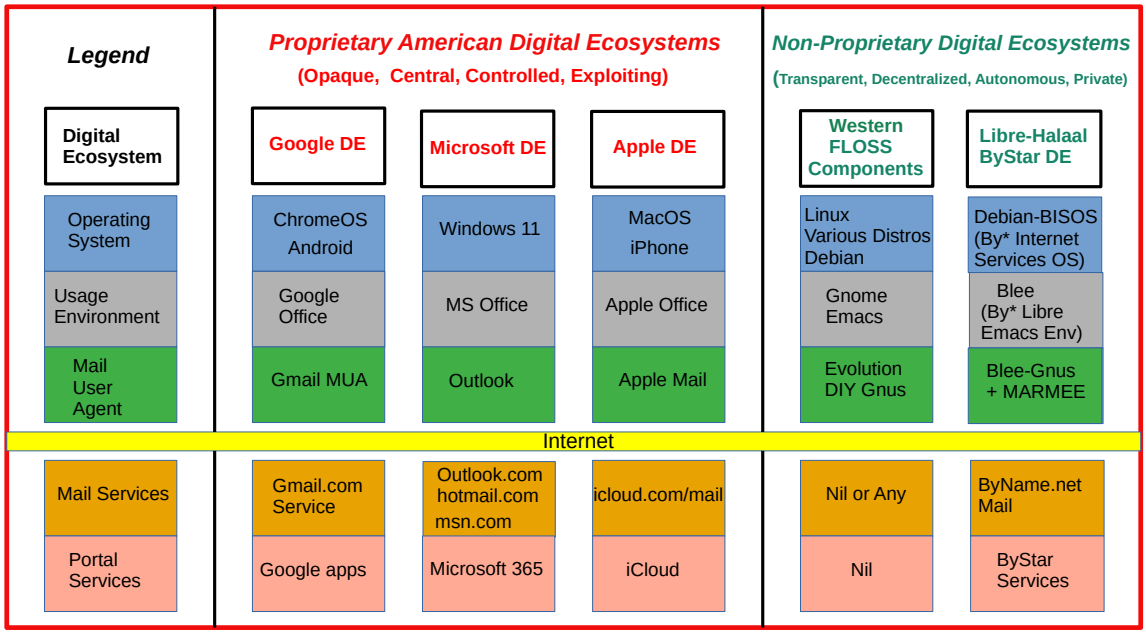

Figure 17.1: ByStar and Proprietary American Digital Ecosystems

BISOS has the following key characteristics.

1. BISOS is both purposeful and general purpose. BISOS is ideology driven. The general purpose of BISOS is to facilitate the creation of digital ecosystems that prioritize autonomy and privacy. The specific purpose of BISOS is to facilitate creation of the Libre-Halaal ByStar Digital Ecosystem.

2. BISOS is layered on top of the Universal Debian software.

3. BISOS facilitates secure and private possession and portability of the user's information through the abstraction of ByStar Portable Objects (BPO).

4. BISOS enables the two-way transfer of Libre Services from the user's own possession to Libre Service providers and between Libre Service providers through the Possession Assertable Libre Services (PALS) abstraction.

5. BISOS creates software-service continuums through universality on both server-side and usage-side.

6. BISOS services integration and usage integration structures are self-confined to select languages: Python, Bash, Elisp and C/C++. Each language environment is augmented with BISOS native frameworks. The primary integration framework of BISOS is Python-Command-Services (PyCS).

7. The primary usage interface for BISOS is Blee (ByStar Libre-Halaal Emacs Environment), which is comprehensive and extends to development environments.

8. BISOS server-side PALS features are based on specific profiles from Debian packages collection. The profiles primary focus on autonomous email and autonomous content publication.

9. BISOS usage-side capabilities are based on specific profiles from Debian packages collection. The profiles primary focus on email handling and content production.

10. BISOS platforms are automated to be recreatable from BPO contained information as physical and virtual images. Linux KVM is the only supported virtualization model.

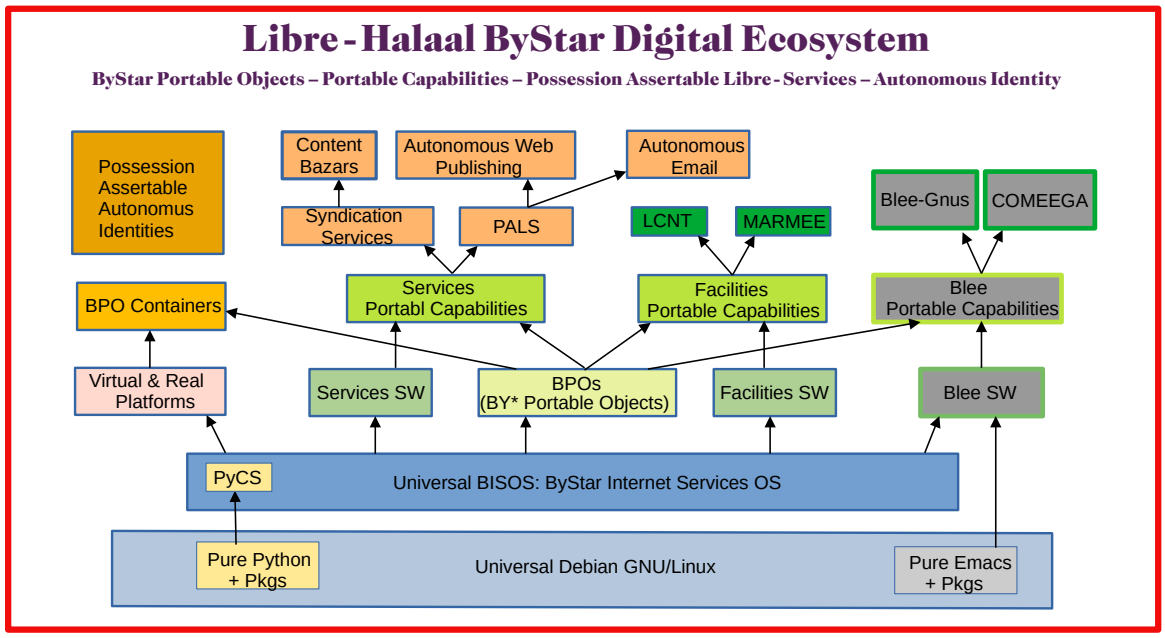

Figure 17.2: ByStar Portable Object Capabilities

11. BISOS's basic unit is a site. A BISOS-Site includes a private git server and a registrar.

BISOS facilities are used to create the infrastructure of ByStar and various types of ByStar services.

Figure 17.2 depicts layerings of BISOS and of ByStar services. The Universal Debian Gnu/Linux is our foundation on top of which BISOS resides.

The box labeled "Services SW" refers to instances of BISOS service-side debian packages. The box labeled "Facilities SW" refers to instances of BISOS usage-side debian packages. Configuration information for packages reside in BPOs (By* Portable Objects).

The combination of "Services SW" and its relevant configuration within a BPO, forms a "Portable Services Capability". The combination of "Facilities SW" and its relevant configuration within a BPO, forms a "Portable Facilities Capability".

Possession Assertable Libre Service (PALS) is a type of `Portable Services Capability`. Multi-Account Resident Mail Exchange Environment (MARMEE) is a type of `Portable Facility Capability`.

Possession Assertable Autonomous Identities (PAAI) are types of BPOs which include the identifiers (e.g., domain names) that enable PALS to become Realized Services.

The stack on the right side of Figure 17.2 depicts BISOS's usage environment which we describe in Section 17.10.

The stack on the left side of Figure 17.2 depicts evolution of platforms in BISOS. A BISOS-Platform is a Debian computer loaded with BISOS software. A BPO-Container is a BISOS-Platform which has received (contains) some BPOs. A PAAI-Container is a BPO-Container which ontains one or more PAAI-BPO.

17.3 BISOS Engineering Philosophy and Ideology

BISOS is purposeful and ideology driven. Parts of BISOS ideology are rooted in health of society. BISOS also reflects a particular engineering philosophy. Figure 17.3 depicts our choices in adoption of philosophical characteristics from various software development groups, with some adjustments.

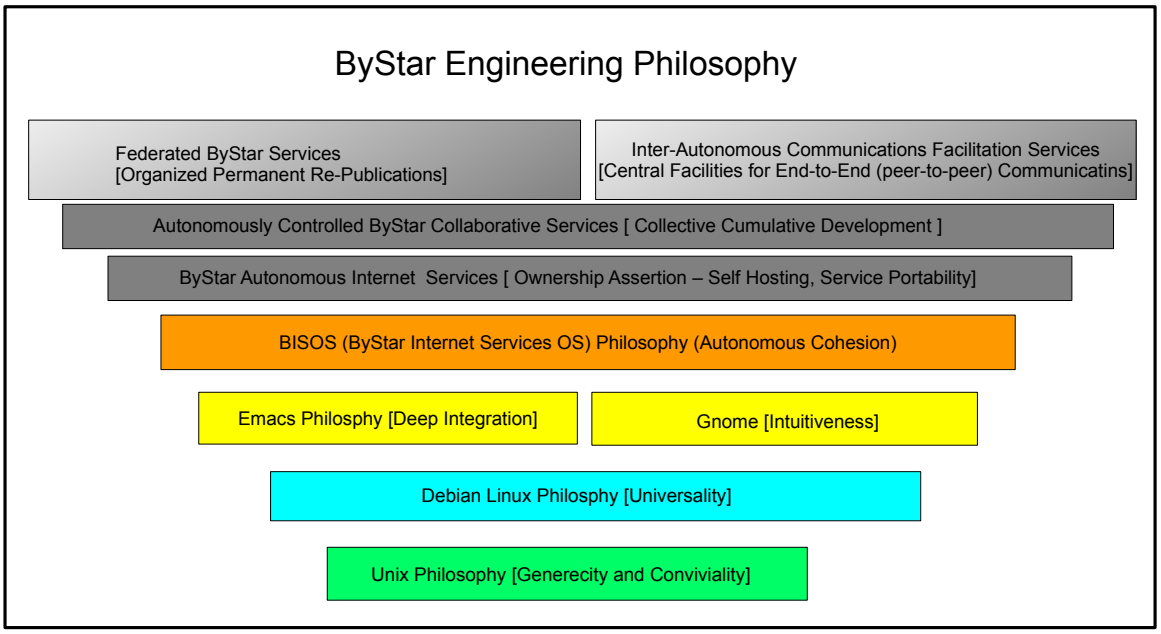

Figure 17.3: ByStar Engineering Philosophy

Unix's Genericity and Conviviality

BISOS is based on the "Unix" model. Not the "Linux" model. We draw a distinct differentiation between "Unix Philosophy" vs "Linux Philosophy" vs "Business Philosophy". Unix Philosophy is a set of cultural norms and philosophical approaches to convivial software development and usage. Unix Philosophy has been well articulated by Ken Thompson, Doug McIlroy, Kernighan, Pike and others.

Linux Philosophy is a laissez faire adaptation of Unix Philosophy that results in software bloat.

BISOS is firmly rooted in a Unix Philosophy and discounts the Business Philosophy and the Linux philosophy.

Debian's Universality

Debian insists on running on everything. By everything we mean a large number of CPU architectures. This is accomplished on methodic and durable reliance on primary source code. By everything we also mean the range of very constrained environments to super computers.

This is important for ByStar because BISOS inherits Debian's Universality.

Emacs's Deep Integration

Blee, BISOS's usage environment, is based on Emacs. Some Emacs builds include a kitchen-sink icon. It is the one feature not yet implemented in Emacs.

Emacs is an integral part of BISOS. It is a framework for consistent integration of internal and external applications. This in turn results in a very convivial usage environment which spans software development, content creation, interpersonal communication and integrated internet application services access.

17.4 BISOS: an Over Debian Pure Blend

Debian defines Pure Blend as: "a subset of Debian that is configured to support a particular target group out-of-the-box. One way to understand this is a list of packages that gets installed to provide a focus of use."

The lower layers of BISOS can be considered a Debian Pure Blend. BISOS-service-side has one deb-pkgs-profile and BISOS-usage-side has another deb-pkgs-profile.

But BISOS goes beyond that. BISOS and Debian are not peers. BISOS is a layer on top of Debian. BISOS provides services-oriented facilities that go beyond the scope of Debian. BISOS has its own policies and practices that are a super set of Debian policies and practices. While the basic unit of Debian is a computer, the basic unit of BISOS is a BISOS-Site.

17.5 BISOS's Basic Unit: BISOS-Site

Typically, the basic unit of an Operating System is one computer — depending on the context the computer is called: a host, a system, a platform, a box, etc.

With BISOS the basic unit is more than one computer. We call BISOS's basic unit: BISOS-Site. Fundamental BISOS abstractions are based on BISOS Portable Objects (BPO) which are implemented as git accounts. Some BPOs must be private. So, a BISOS-Site must include a private git server — which is implemented as a Gitlab instance. BISOS's use of BPO is purely through a Python API interface. Gitlab GUI is hardly ever used. BISOS also relies on the uniqueness of names and numbers. BISOS therefore needs an automated registrar for some private names and numbers. For BISOS to fully operate, at a minimum it needs those services.

A BISOS-Site also provides facilities for creation and management of Virtual Machines (VMs) and a simple BISOS-CMDB (configuration management database) — a central repository for storing BISOS-Site related resource. For creation and recreation of VMs (image management), BISOS uses Vagrant.

17.6 BISOS Portable Objects (BPO)

A fundamental abstraction of BISOS is the concept of BISOS Portable Objects (BPO). BPOs are packages of information. There are some similarities between BPOs as packages of information and software packages such as deb-packages or rpm-packages.

Like software packages, BPOs are named uniquely and can depend on each other and can be collectively installed and uninstalled. BPOs are used for many things similar to how the files system is used for many things. BPOs can be used to hold the complete configuration information of a system. BPOs can be used to hold configuration information for software packages. BPOs can be used to hold private user data. BPOs can be used to hold collections of content and source code.

For its own operation, BISOS uses various BPO types. Other types of BPOs can be created or generic BPO types (for example the Project type) can be used.

Each BPO consists of a number of Git Repositories (hereafter called "repos"). Each of the BPO's repos can be synchronized using generic Git tools, but we use Blee/Emacs's MaGit exclusively.

BPOs are implemented as Gitlab accounts. Gitlab accounts are Unix non-login shell accounts. BISOS's interactions with Gitlab is exclusively through an API (Remote Operations). Each Gitlab account then can contain repos subject to common access control mechanisms. Gitlab accounts map to BPO-Identifiers (BPO-Id). Each BPO-id then maps to Unix non-login shell accounts. The Unix account then becomes the base for cloning of the repos in the corresponding Gitlab account.

Combinations of profiled deb-packages for internet application services and their configurations in the form of BPOs can then create Libre Services that are possession assertable, portable and transferable.

17.7 BISOS Possession Assertable Libre Services (PALS)

Based on capabilities of BPOs and the capabilities of service-side profiled Debian packages, we can now create Libre Services.

BISOS Libre Services can be thought of four parts:

1. Libre-Halaal software of the services (usually a Debian Package)
2. Configuration information for the software for the service (often as a repo of a PALS-BPO)
3. Names and numbers for binding of services (as a repo of a PAAI-BPO)
4. Service owner data (in the form of one or more BPOs)

This model provides for portability and transferability of Libre Services between network abodes. For example, a Libre Service at a provider can be transferred to its owner to be self-hosted.

There are some similarities between PALS-BPO and container virtualization (Docker and Kubernetes). PALS-BPOs include comprehensive information for construction of services and these can be mapped to container virtualization. However, at this time BISOS does not use container virtualization, as it is redundant. BISOS uses BPOs to create and recreate Kernel-based Virtual Machines (KVM) inside of which PALS-BPOs are deployed.

Self-hosting is the practice of running and maintaining a Libre Service under one's own full control at one's own premise. BISOS Possession Assertable Libre Services (PALS) can be initially self-hosted and then transferred to a Libre Service provider. PALS can also be initially externally hosted and then become self-hosted on demand. The concept of "transferability" between network abodes is well supported in BISOS.

17.7.1 Network Abodes and Transferability

In the proprietary American digital ecosystem, the concept of network abodes is mostly vague. Names such as cloud and edge are used without much precision, and, the concept of transferability simply does not exist. You cannot self-host your Gmail service.

Within ByStar and BISOS, we have precise definitions for where Libre Services can be realized and where they can be transferred to. This is depicted in Figure 17.4

Let's define "edge" as point of demarcation between the public digital world and the physical world (and its associated private digital environment). In Figure 17.4 this is depicted as a dotted red circle. When by physical world, we mean "things", then in the American Internet, we have the culture and lingo of IoT (Internet of Things) Edge Computing. But what if by the physical world, we mean people — individuals?

The three concentric circles on the outer side of the edge are called "Rims". These are:

1. Exposed Rim.

   Systems in the Exposed Rim are on your premise, and they are externally visible. Wifi hotspots, routers and VPNs are usually in the Exposed Rim. Self-Hosting occurs in the Exposed Rim. Systems in the Exposed Rim should be well secured as they are vulnerable to direct attacks.

2. Inner Rim.

   Systems in the Inner Rim are on your premise behind a firewall. private desktops, fileservers, private Gitlab and private registrars are usually in the Inner Rim. Systems in the Inner Rim are usually physically stationary.

3. Outer Rim.

   Systems in the Outer Rim are usually portable devices and at this time they are on your premise behind a firewall. Laptops, Pads, Mobile-Phones (with wifi access) are usually in the Outer Rim. Systems in the Outer Rim are usually portable devices.

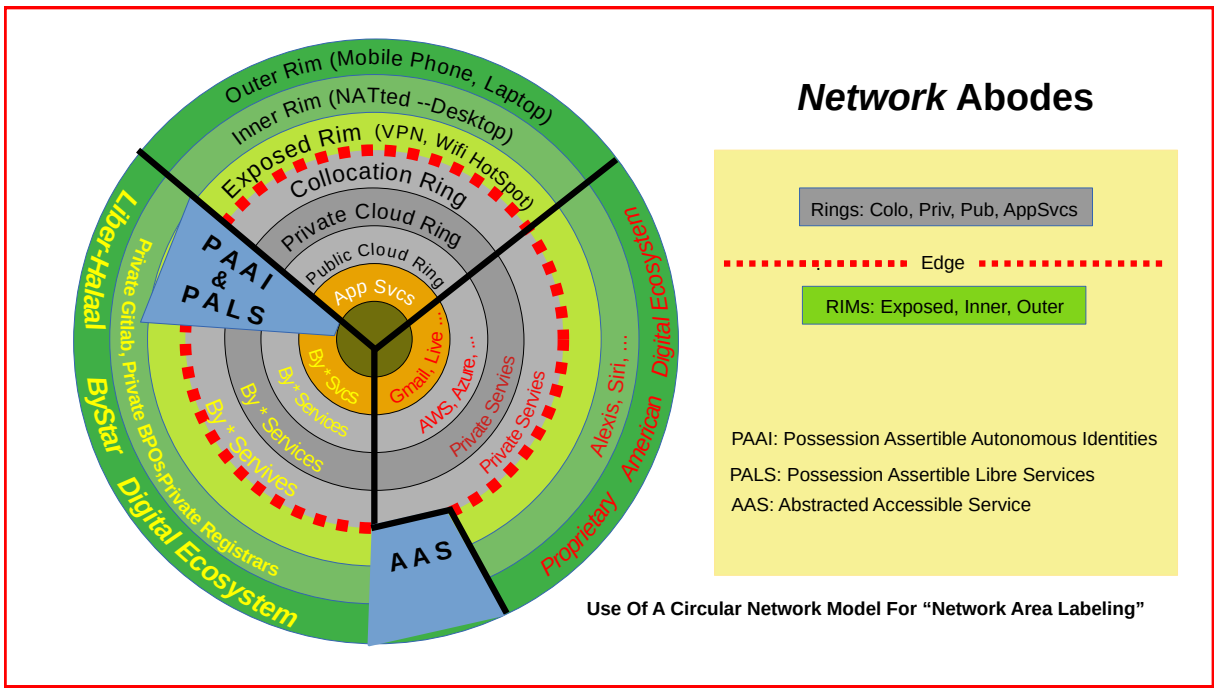

Figure 17.4: Network Abodes: A Circular Model For Network Area Labeling

The four concentric circles on the outer side of the edge are called "Rings". These are:

1. Collocation Ring.

   Systems in the Collocation Ring are on somebody else's premise (usually a data center), but they belong to you (or are rented by you). A collocation data center is a physical facility that offers space with the proper power, cooling, network connectivity and security to host other people's computing hardware and servers. There is a certain aspect of self-possession in the Collocation Ring.

2. Private Cloud Ring.

   Systems in the Private Cloud Ring are usually virtualized and are under your exclusive access.

3. Public Cloud Ring.

   Systems in the Public Cloud Ring are usually virtualized and are under your access.

4. Public Internet Application Services.

   Examples of Public Internet Application Services in the proprietary American digital ecosystem are Gmail, Facebook and Instagram. You pay for public proprietary internet application services by becoming the product, through your privacy.

In the model of the proprietary American digital ecosystem, a given internet application service typically permanently resides in the ring abodes and is not transferable to other service providers. The service belongs to the service provider and it is locked.

In the ByStar model, the service belongs to its user and it is the user who decides where she wants to realize it. This transferability is accomplished through the abstractions of BPOs (BISOS Portable Objects), PALS (Possession Assertable Libre Services) and PAAI (Possession Assertable Autonomous Identities). In Figure 17.4 the segment labeled "PAAI & PALS" spans the Exposed Rim, the Collocation Ring, the Private Cloud Ring, the Public Cloud Ring and the Application Services Ring. This means that a BISOS based Libre Services can be transferred between any of those network abodes.

BISOS can also be used to provide access to proprietary internet application services. This is shown in the segment labeled "AAS" of Figure 17.4. Abstracted Application Services (AAS) are facilities that allow for abstraction of some proprietary internet application services to be used by BISOS. One such internet service is Gmail. Gmail can be used through Blee-Gnus and BISOS-MARMEE.

17.7.2 Ramifications of Libre-Halaal Edge-Oriented Strategies

To illustrate the privacy and autonomy-oriented benefits of the PALS model, let's compare and contrast The American Internet with ByStar in the context of a very simple but very important human application: "email". To be more concrete and specific, in the context of the American Internet, let's use the fictional example of an American politician called "Hillary Clinton". In the context of ByStar, let's use the fictional example of an Iranian engineer called "Mohsen Banan".

In the American Internet environment, the individual typically has at least two email addresses. One is through her work, say at the State Department, as: "hillary.clinton@state.gov". The other is for personal use, as: "hillary.clinton@gmail.com". Paying attention to her email addresses, we note that "hillary.clinton" is always on the left side of the "@". This means that "gmail.com" has risen in the middle and controls "hillary.clinton@" — and millions of others. This means that Google has full possession and full control over Hillary's personal emails. Her "hillary.clinton@gmail.com" emails are neither autonomous nor private. Now, since Hillary Clinton is an intelligent and powerful American politician, she has recognized that her privacy and autonomy are important and that her email communications should be under her full control. She is rich, so, she goes ahead and sets up her own email server in her basement. We don't know if that email server was based on proprietary software or not, but we do know that as an individualistic American, she was only focused on addressing her own email autonomy and privacy concerns. Email autonomy and privacy of society at large was not her concern.

In the ByStar environment, the individual similarly also has two sets of email addresses. Mohsen's work email may well be under the control of his employer, but his private email service and email addresses are under his own control. For personal use, Mohsen has registered and obtained
`mohsen.banan.byname.net` for himself.
Notice that while `byname.net` is part of ByStar,
`mohsen.banan.byname.net` belongs to Mohsen. Based on that, he can now create a series of email addresses for himself.
For example, he can use "bystarPlan@mohsen.banan.byname.net" for matters related to distribution of this document.
He can use "card@mohsen.banan.byname.net" on his visit cards.

Now, let's compare and contrast the email addresses "hillary.clinton@gmail.com" and "myDesk@mohsen.banan.byname.net". The right-part of the '@' signifies ownership and control. The right part of '@' controls the left-part of '@'. So, `gmail.com` controls "hillary.clinton". While `mohsen.banan.byname.net` controls "myDesk" and Mohsen, owns `mohsen.banan.byname.net`. Notice that `gmail.com` controls millions of people through their left-part. In ByStar, millions of people can obtain their own right-parts and then control their own left-parts — and own their own portable full email addresses.
Notice that while `gmail.com` has positioned itself in the middle of the network, `mohsen.banan.byname.net` has positioned itself in the edge of the network. Longer domain names which fully take advantage of DNS's hierarchical design are manifestations of edge-oriented strategies.

Next, let's compare and contrast the software of the `gmail.com` service against the software of `mohsen.banan.byname.net`. The software of `gmail.com` service is proprietary. It belongs to Google. We don't know what it does. When you hit the delete button for a particular email, you can no longer see that message. But perhaps Google is keeping all of your deleted messages somewhere, forever. Because it is all proprietary software, you just don't know what is actually happening with the emails that you may think are yours. The software of `mohsen.banan.byname.net` services is part of the public ByStar software. It is part of BISOS. It is a public resource. That entire software is internally transparent. On your behalf, the engineering profession knows what it does and what it does not. When you delete one of your own email messages, it can be known that it was truly deleted — forever. This is what having a Libre-Halaal Service means.

With ByStar in place, all the Hillary Clintons of this world can have their own email communications under their own full control. We invite Hillary Clinton to join ByStar. As an American politician, perhaps she can start

thinking about solving her society's email problems — not just her own. We welcome her assistance in promoting ByStar.

Consider the privacy and autonomy of such edge-to-edge email communications between "myDesk@mohsen.banan.byname.net" and "myDesk@hillary.clinton.byname.net".
The mail protocol traffic is of course end-to-end encrypted between `mohsen.banan.byname.net` and `hillary.clinton.byname.net`. The message itself can additionally be encrypted. At no point is any third party in possession of the clear-text message. Logs of the message transfer are only in the possession of the two edges. And all of this can be realized on an internet-scale.

All ByStar individual services are intended to be end-to-end and edge-oriented. However, they don't need to reside on the "Rims" side of the network edge. Since ByStar individual services are possession-assertable and portable, they can also be provisioned in the "Rings". See Figure 17.4 for the references to Edge, Rims and Rings. This provides for options of self-hosting or external-hosting of individual services. So, `byname.net` can be made to be as convenient as `gmail.com` yet preserves the guarantees of autonomy and privacy through being possession-assertable, portable, Libre-Halaal, and edge-oriented.

While here we focused on the email service as an end-to-end edge-oriented strategy, similar approaches can be applied to other internet applications and intra-edge applications. In the edge-oriented ByStar model, when you control the thermostat in your own house, that can all happen as a ByStar intra-edge application without loss of privacy and autonomy.

17.8 BISOS Model of Platform Universality and Software-Service Continuums

Earlier we made several points about the universality of BISOS. We pointed out that BISOS inherits Debian's universality, and that our design philosophy includes relying on a singular Unix with full cohesion.

We have Service-Side BISOS for creation of internet services and we have Usage-Side BISOS for usage of internet services. These two create the BISOS software-service continuum. This is very powerful because the two sides are very consistent. This is depicted in Figure 17.5.

Note in Figure 17.5 that although the lowest layer (hardware) of the two stacks is very different, most of the rest of the stack is very common. Also note that on the top parts, capabilities are complimentary based on the common lower layers.

The degree of consistency and cohesion that this universality creates if far superior to what exists today in the proprietary American digital ecosystem.

17.9 PyCS: BISOS's Integration Framework

BISOS is largely focused on configuration and integration of related software packages towards creation of consistent services. This is typically done with "scripts" that augment the software packages in a consistent way. By scripts, we mean programs that are executed at command line. At times we also need to build Remote Operations (RO) to accommodate remote invocation of central services.

There are three fundamental important choices to be made:

1. What programming language should we use for integration?
2. What command-line framework should we use?
3. What Remote Operations (Web Services, REST, Micro Services) framework should we use?

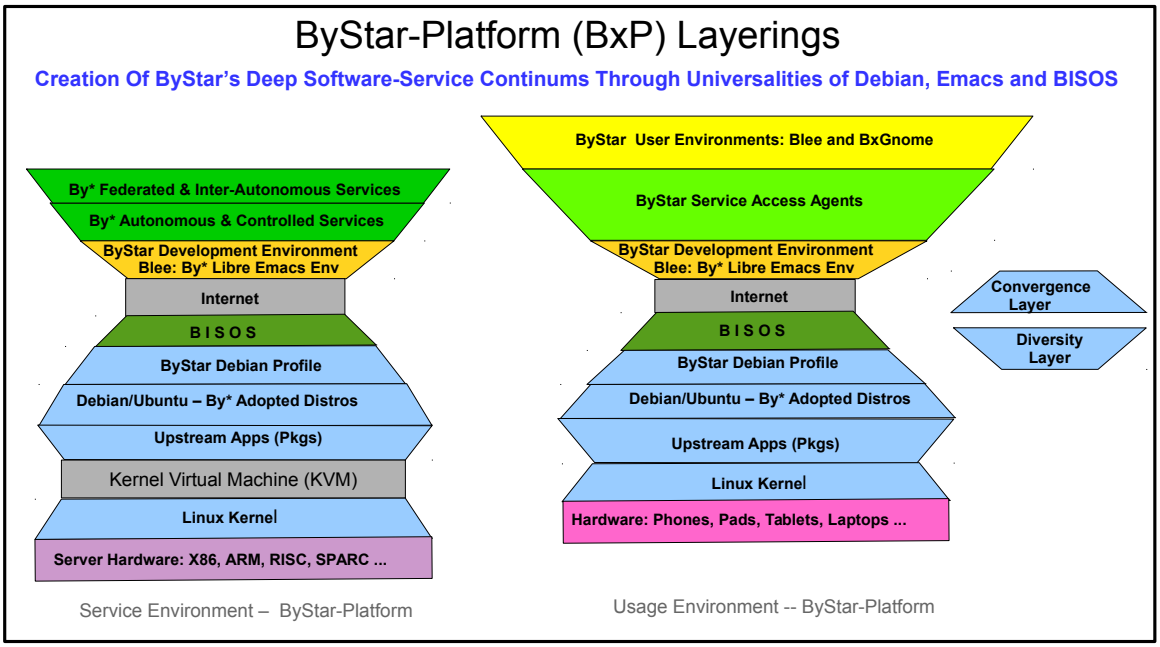

Figure 17.5: ByStar Platform Layerings and Software-Service Continuums

BISOS primarily uses Python and some Bash for scripting.

There are various Python frameworks for command-line and web services. These include click, FastAPI, Flask, Django, RPyC and various others. None of these provide a comprehensive enough framework for BISOS. BPyF (BISOS Python Framework) is a comprehensive integration framework of BISOS that combines existing capabilities from various Python frameworks.

As depicted in Figure 17.6, BPyF consists of five major parts.

- Common facilities — logging, io, error handling, etc.
- File Parameters (FP) and Schema of File Parameters — BISOS's data representation and configuration model
- PyCS: Python Command Services
- BISOS Abstractions
- CS-Units and CS-MultiUnits

In Figure 17.6, boxes under the dashed line represent various libraries. General purpose libraries (on the right side is light green) provide common facilities such as IO, logging, error handling and configuration management which are used throughout BISOS. Various libraries that represent BISOS abstractions in Python such as BPOs, PALS and PAAI. These are shown on the left side in darker green.

For data representation, BISOS uses its own model called File Parameters. The equivalent functionality of File Parameters is often provided by Yaml and Json in typical open-source software packages.

PyCS is rooted in the model of Expectation Complete Operations (ECO), which allows for local invocation of an ECO to map to command-line invocation and remote invocation of an ECO to map to the microservices model and Remote Operations. This universality of ECOs allows for command-line facilities to become microservices.

Facilities for command line invocation are depicted above the dashed line, on the left side of "internet". Facilities in support of service (Remote Operation) performers are depicted above the dashed line, on the right side of "internet".

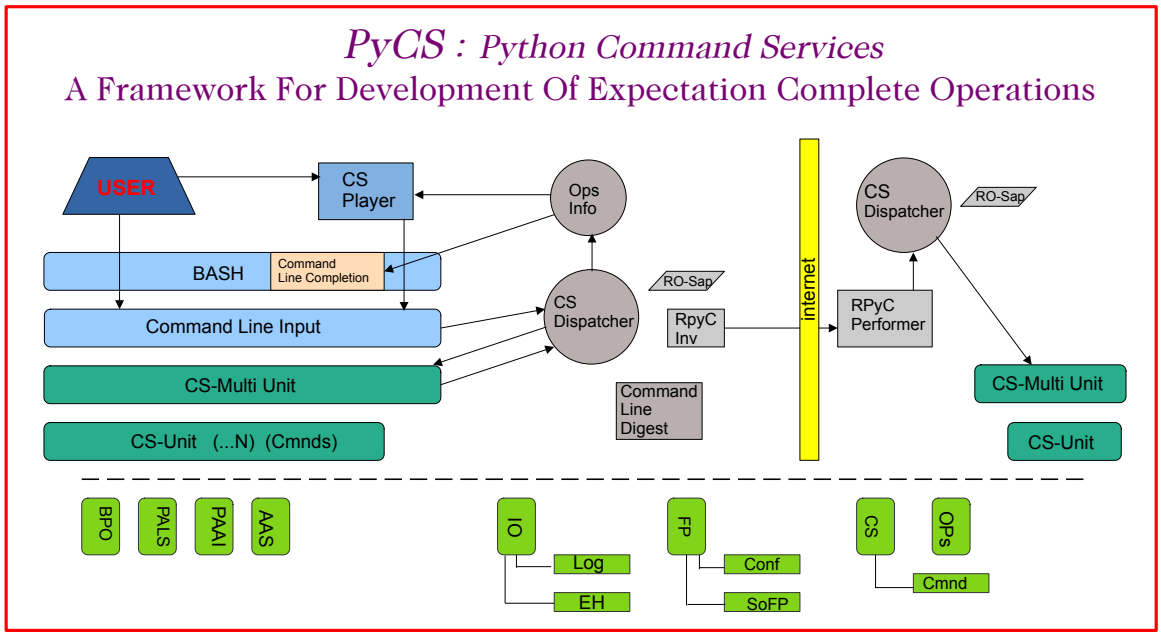

Figure 17.6: BPyF (BISOS Python Platform) and PyCS

Expectation complete operations are specified and implemented in CS-Units. A CS-Multi-Unit represents a collection of CS-Units. Notice that CS-Unit and CS-Multi-Unit boxes are replicated on both sides of "internet". This indicates that both commands and remote operations map to expectation complete operations.

Each ECO is capable of describing everything expected from the operation in full detail which includes all typing information. The information in Expectation Complete Operation includes:

- Name of the operation
- All input parameters
  - List of optional and mandatory parameters
  - List of positional arguments
  - Stdin expectations
- All outcome parameters
  - All result parameters
  - All error parameters

The information of expectation complete operation then maps to command-line verbs, parameters and arguments, and similarly for remote operations. The list of available verbs is specified by the CS-Multi-Unit. Since CS-Multi-Units are capable of describing all of the expectations of all of their operations, very powerful automated user interfaces for invocation of operations can be built. The "CS Player" box in Figure 17.6 illustrates that.

Remote operations are implemented using RPyC. RPyC or Remote Python Call, is a transparent library for symmetrical remote procedure calls, clustering, and distributed-computing. Use of RPyC is depicted with the line going through the vertical box labeled "internet". Names used by invokers and performers are shown in the boxes labeled "RO-Sap" (Remote Operation Service Access Point).

PyCS framework provides a solid foundation for transformation of software into services and integration of software and services in BISOS.

Halaal and Haraam, 5
halaal manner-of-existence, 68, 121, 161
Halloween, 125
haraam manner-of-existence, 121
holocaust, 88
human activity, 21, 24
human beings, 95

Ibn-Sina, 15
Ibrahimic, 65, 71, 150
Imam Khomeini, 150, 151, 237
Imam Khomeini, 67, 68, 236, 237
imitation, 105
important aspect, 103
inherent characteristic, 96, 165
intellectual, 104
Intellectual Property, 1
intellectual property, 64, 71, 138
Intellectual Property Rights, 1
internally transparent, 29, 104
internet engineering, 63
internet services, 35, 121, 143, 159, 162
IP, 1
IPR, 1
Iran, 59, 68, 235–237
Ivan Illich, 4, 46

Jalaal Al-Ahamad, 6
James Madison, 82
Jean-Paul II Encyclique, 152
Jewish, 20, 97, 153
Judaism, 29
Julian Assange, 41

Kamikaze, 24
Khomeini, 67, 68, 93, 151, 236, 237

labor theory, 73
libre, 117
Libre-Halaal, 5, 104
local law, 68

made, 16, 21, 25, 28, 29, 39, 61, 95, 122, 138, 159, 163, 279
man-made, 18
manner, 29, 35, 40, 62, 103, 118, 120, 127, 138, 160, 166
manufacturing process, 18, 25
Melvinia, 78
Michelle Obama, 78
mixed-existential, 16, 23, 25, 28, 31, 75
mixed-existentials, 1, 16, 18, 24, 28, 32, 128, 135
Monoexistential Capitalism, 9
monoexistential component, 25, 28, 75
monoexistentials, 1
monopolistic ownership oriented restriction, 31
monopolistic polyexistential restrictions, 29
moral alternative, 105, 167
Muhammad Ali, 82
multi-possessablity, 25, 63, 67
multiple places, 29
multiple possessors, 29

Najaf, 150
natural collaborative environment, 63
natural dynamics, 35
natural rights, 26, 34, 65, 71, 73
non-proprietary, 115, 122, 133, 139, 143, 147
non-proprietary model, 140
non-proprietary software, 116, 131
Non-Rivalry, 20
non-rivalry goods, 24, 29, 144
non-signatory, 68
Nyquist, 37, 38, 49

opaque, 35, 41, 64, 103, 116, 121, 132
open source, 117, 122
open-source, 21, 29, 115, 119, 121, 125, 143
open-source movement, 115, 120
open-source software, 21, 117, 130, 143
Orientalism, 5, 6
ownership laws, 28, 30, 71
ownership models, 104
ownership rules, 95, 130, 133

patent law, 25, 32, 35, 64, 115, 130
patent laws, 16, 32
patent system, 35
physics, 15
Piracy, 66
Polyexistential Capitalism, 9
polyexistential component, 18, 25, 28, 31, 75
polyexistentials, 1
polyexistentials reference model, 16
private polyexistential, 23
private use, 122
produce, 21, 24, 37, 40, 63, 116, 139, 159
programming languages, 40
proper governance, 15
property, 16, 28, 34, 41, 64, 70, 73, 104, 144
property rights, 28, 65, 75
proprietary model, 122, 139, 143
proprietary ownership, 104, 140
proprietary software, 121, 130, 141, 167
Proprietary-Haraam, 10, 49, 169, 220
public goods, 20, 35
public restrictions, 30, 32
pure polyexistential, 16, 21, 121
pure polyexistentials, 21, 35

real, 29, 32, 67, 69, 105, 163
reserved rights argument, 75
resources, 67, 75, 105

revenue models, 123, 142
rigged misnomer, 65
Rise-of-the-Middle, 178

secrecy, 39, 65
Secular, 150
self-interest, 95, 134, 139, 165
Shannon, 37, 38
societal, 15, 29, 34, 40, 61, 64, 67, 70, 103, 105, 116, 119, 125, 140, 143, 165
societal level, 103, 117, 163
software engineering, 116, 126
software engineering profession, 41, 118, 122, 138, 159
software engineers, 40, 63, 116, 126, 143
source code, 21, 41, 118, 121, 159
source form, 21, 40
specialized skills, 35
Stuxnet, 237, 238
Supremacy, 96

tangible, 15, 31, 41, 70, 75
tear points, 104
Theological Research, 151
trademark law, 65
trademark laws, 65
Twitter, 236

unauthorized copying, 66
universal, 25, 35, 61, 65, 67, 71
universal basic human rights, 26
Unix, 181
useful arts, 34, 95, 132

value system, 95

walled gardens, 35
West-toxication, 6
Western, 5, 61, 104, 160, 165
Western Intellectual Property Rights, 68
Western Trade Organization, 237
WTO, 237

Yahoo, 163

Zuckerberg, 163