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.p4cel

First International Edition

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

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.

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.

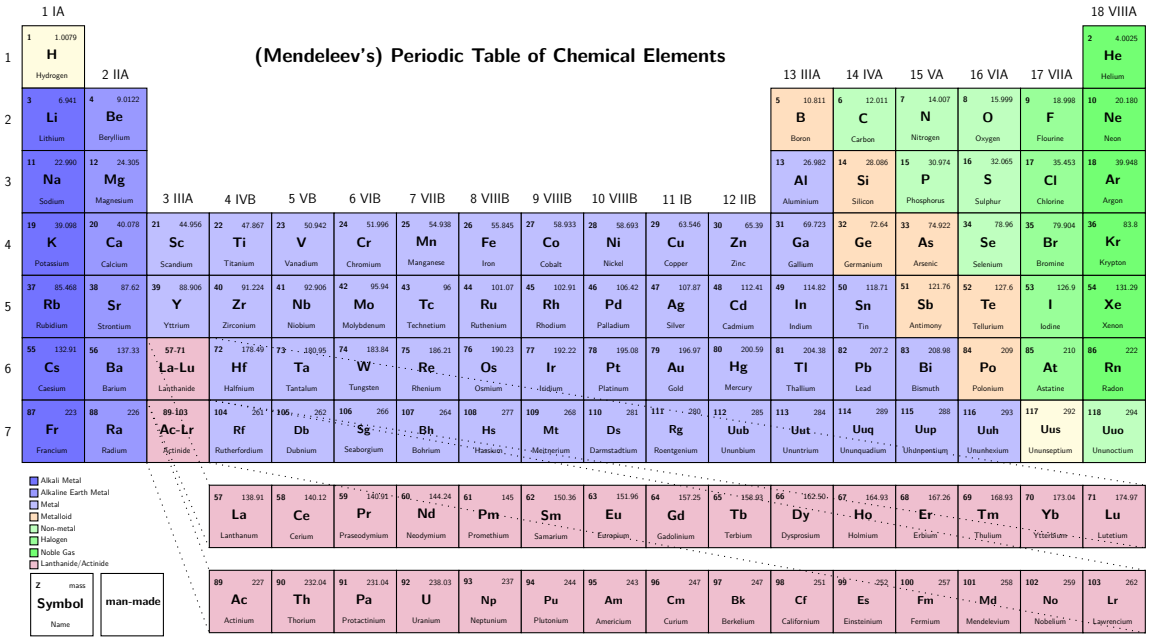

Figure 2.1: Periodic Table of Chemical Elements

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.

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.

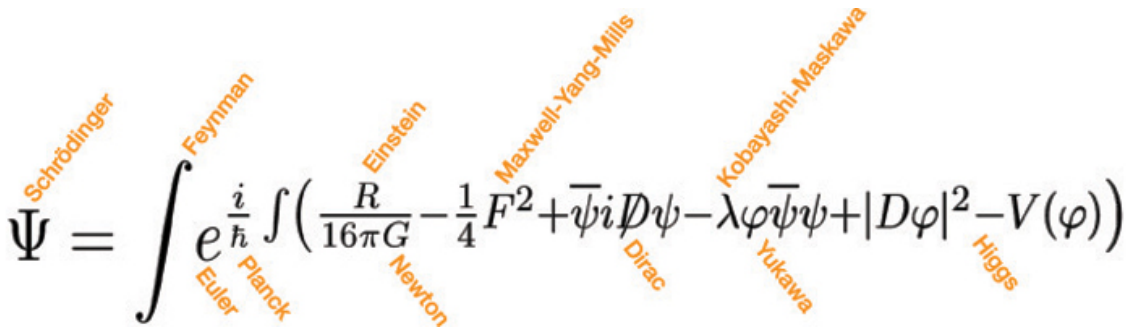

Figure 2.2: Unified Physics Equation With Inventors Labels

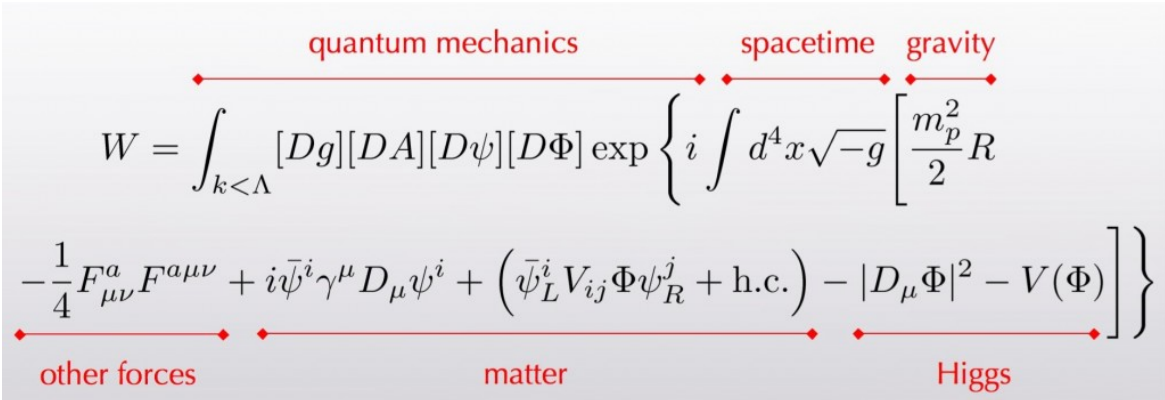

Figure 2.3: Unified Physics Equation With Subject Matter Labels

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.

2.4 Polyexistentials

We present the concept of "Expressed Formula" as the general form of "primary polyexistential". Each formula forms a class of polyexistentials. Each possession of a formula forms an instance of a polyexistential. The digital format presents a "pure polyexistential" form. Unless expressed a formula is not a polyexistential.

The full emergence of digital technology in the middle of the 20th 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. Restrictions on polyexistentials has been harming nearly every aspect of life.

Pure polyexistentials are kept in some form of memory. Polyexistentials are "remembered", "retrieved" with memory. Memory relates to object permanence. While memory is usually material, polyexistentials are always non-material. Memory can be a human's brain or handwritten ink on a piece of paper, machine produced ink on paper (traditional books), digitized information on hard disk. Memory functions as a minimal substrate and is the container of polyexistentials.

An animal can be the producer of the polyexistential and an animal's memory can be the memory for a polyexistential. But polyexistentials are for the most part the result of human activity. Polyexistentials are often expected to be useful. The value of the polyexistentials comes from the impact that they can have on human condition.

expressed our understanding of transmission of digital entities.

We then built on this physical layer understanding and added say six more layers to create the internet.[8] And we now have a global network on which digitals can be transmitted, often without knowing borders.

3.2.4 Cryptography, Encryption and Information Confidentiality

Storage and transfer of digital entities can be in the clear or can be made confidential.

Cryptography, the use of codes and ciphers to protect secrets, began thousands of years ago. Methods of encryption that use pen and paper were used to achieve some secrecy.

By 1949, we had Shannon's theory of perfect secrecy, as a mathematical model for secure communication. It states that if a message is encrypted using a key that is as long as the message itself, then the message is theoretically unbreakable. This is because the key is as long as the message, so it is impossible to determine the key without knowing the message. This means that the only way to decrypt the message is to have the key, which is only known by the sender and receiver.

In parallel with our entry into the digital era, roughly in the 1970s secure cryptography which until then was largely the preserve of governments became a generally available tool. Two events have since brought it squarely into the public domain: the creation of public encryption standards like DES, and the invention of public-key cryptography systems (PKCS). By the 1980s, internationally proposed standards such as X.509 included all necessary knowledge to secure digital information.

Nature believes in encryption. Nature facilitates encryption. It is natural to encrypt.

It is easier to encrypt information than it is to decrypt it.

We have the necessary knowledge to make digital entities private and to make our human communications and human interactions autonomous and private. So, our privacy can be preserved.

3.3 Programming Languages and Manner-of-Existence of Software

One perspective on software and programs is that they are human made set of instructions that computers execute. Another perspective, expressed by Donald Knuth, is: "Programs are meant to be read by humans and only incidentally for computers to execute." Figure 3.1 shows a timeline for the history of high-level-programming languages evolution from 1954 to 2002.[9] Prior to Fortran, most programs were written in Assembly Language. An assembly language is a type of low-level programming language that is intended to communicate directly with a computer's hardware. Unlike machine language, which consists of binary code, assembly languages are designed to be readable by humans. Each computer has its own assembly language and a program written for one computer would not execute on another.[10]

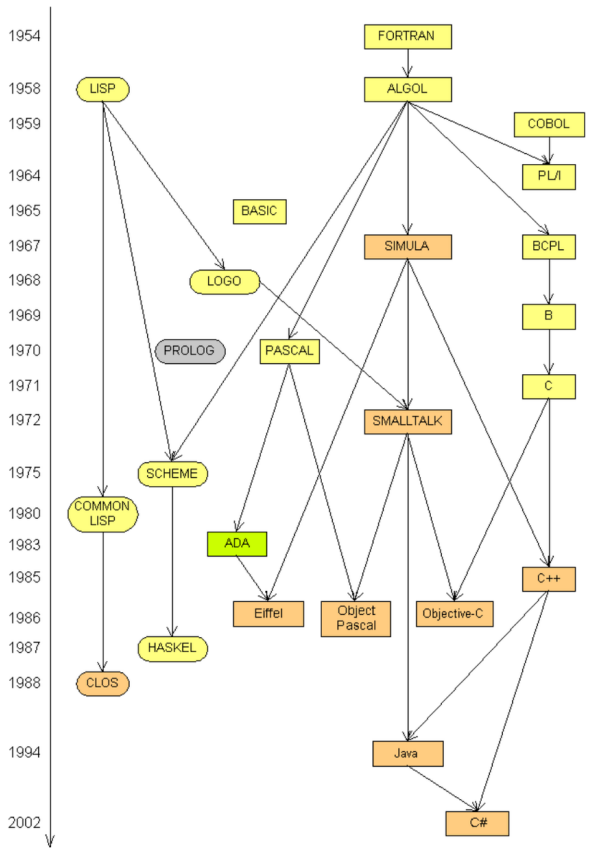

Figure 3.1: A History of Programming Languages

Fortran (FORmula TRANslation) was the first enduring effort in creating a programming language that would produce binaries that would execute on many computers through the use of a Compiler. Fortran was primarily discipline specific. It was for scientific computations.

In our view, the most significant event in the history of programming languages is the publication of the Lisp paper titled: "Recursive functions of symbolic expressions and their computation by machine, Part I" [38] by McCarthy in 1960. Lisp raised programming from the machine domain to human domain. Abstractions of Lisp were no longer computer centric; they were consistent symbols which a programmer could tailor to her subject domain. Various dialects of Lisp continue to be in common use today. In its early days it did not have all the necessary capabilities, but its fundamentals and its structure supported evolution.

Alan Kay has famously described Lisp as the "Maxwell's equations of software". The universality of syntax of Lisp makes it unique in ways that are absent on other programming languages. Lisp is homoiconic. It treats code as data. This means that it is able to create domain specific structures (through a powerful macro system) and become very extensible.

Over the years software engineers have come up with a number of programming languages which emphasize various desired characteristics (efficiency, object orientation, robustness, ease of use, etc.) These have formed families of programming languages which are depicted in Figure 3.1.

From the perspective of polyexistence, what is of significance for us is manner-of-existence of software. Software has two forms, binary and source. The binary form of software is for execution by computers. The source form of software is for use by humans — software engineers.

Based on societal laws, for general use, software can be available in binary form only. Or, software can be available in source form and its resulting binary form. When software is available in source form, based on societal regulations modification and distribution of software can be limited. These types of limitations and permissions define manner-of-existence of software.

Through the Western IPR regime, Microsoft Windows is available in binary form only. This has immense ramifications on health of society which impact the autonomy and privacy of individuals.

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.

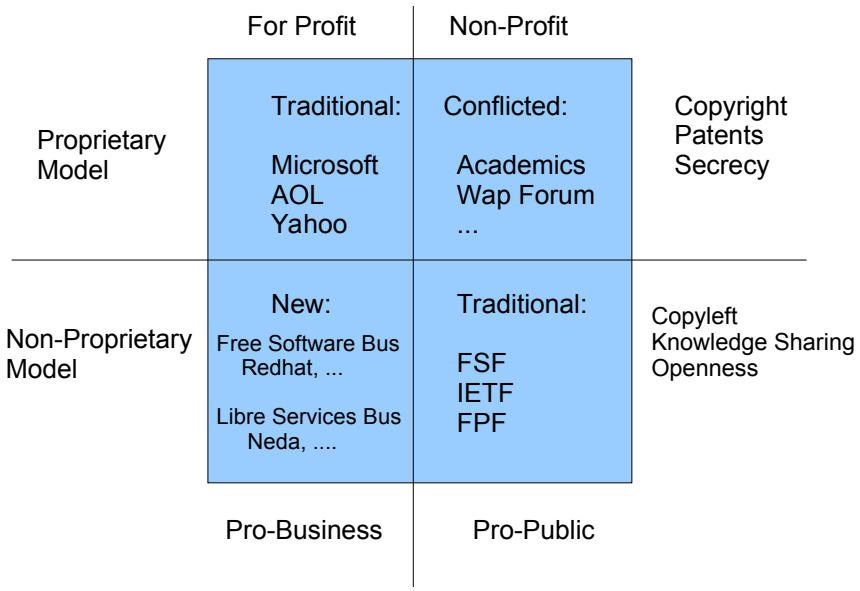

Figure 13.1: Operation in the For-Profit and Non-Proprietary Quadrant

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 scope of Libre-Halaal Capitalism is all polyexistentials. The economics of currently patented medications are within scope of Libre-Halaal Capitalism. The economics of Monsanto patents for genetically modified soybeans are within scope of Libre-Halaal Capitalism.

But initially we focus on Libre-Halaal Capitalism in the digital domain. By that, we mean:

- Software — Based on Libre-Halaal Software
- Internet Application Services — Based on Libre-Services
- Digital Content — Based on Libre-Halaal Content

13.6.1 Transformation of Software into Services

In Section 13.5 we introduced two dimensions of Proprietary vs. Non-Proprietary and For-Profit vs. Non-Profit. To those two dimensions now add another dimension. That of Software Vs Internet Application Services (Internet Services).

Part of the debate about FOSS is now over, while part continues. The part that is over is any question about the viability of FOSS as a development model for creating large-scale, complex, relevant software systems. GNU/Linux is a fully viable free software alternative to the proprietary Microsoft Windows operating system, against which it continues to make steady inroads.

And apart from such well-known and high-profile projects, behind the scenes the FOSS movement has become 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.

And the fundamental FOSS creative dynamic has now also become very well understood: the FOSS development model allows *unrestricted creative reuse of existing assets at essentially zero cost.* It is from this dynamic that the FOSS model derives its tremendous generative power. FOSS is thus fully established as a generative engine and an industry reality and is here to stay.

But the part of the debate that continues is whether or not this has any meaningful commercial dimension. Within the proprietary software domain, a powerful revenue-generating engine exists in the form of the traditional software licensing model. But this revenue source is absent under the FOSS model. In its place there are a number of possible business and revenue models, but in all cases, these lack the large-scale repeatability that makes things really interesting from a business perspective.

There thus remains a conceptual gap, a puzzle, about how the powerful generative forces of FOSS can be turned into a large-scale, repeatable, revenue stream. But this puzzle is now solved.

Business Dynamics Of Internet Services

Within the Internet Services industry the business and revenue models are quite clear and obvious. The largest and most obvious are the subscription fee model of generalized service providers, and the advertising model of numerous specialized no-cost service providers, demonstrated most spectacularly by Google. Both the subscription fee and advertising models are unlimitedly scalable, thus resulting in the gigantic commercial Internet of today.

But the Internet Services industry of today is a fundamentally proprietary construct. While proprietary service providers can and do make frequent use of FOSS components within their services, they do not espouse the FOSS development model itself, and their technical development process remains competitive and proprietary. Though they may incorporate FOSS components, Facebook and Google are certainly not FOSS values oriented.

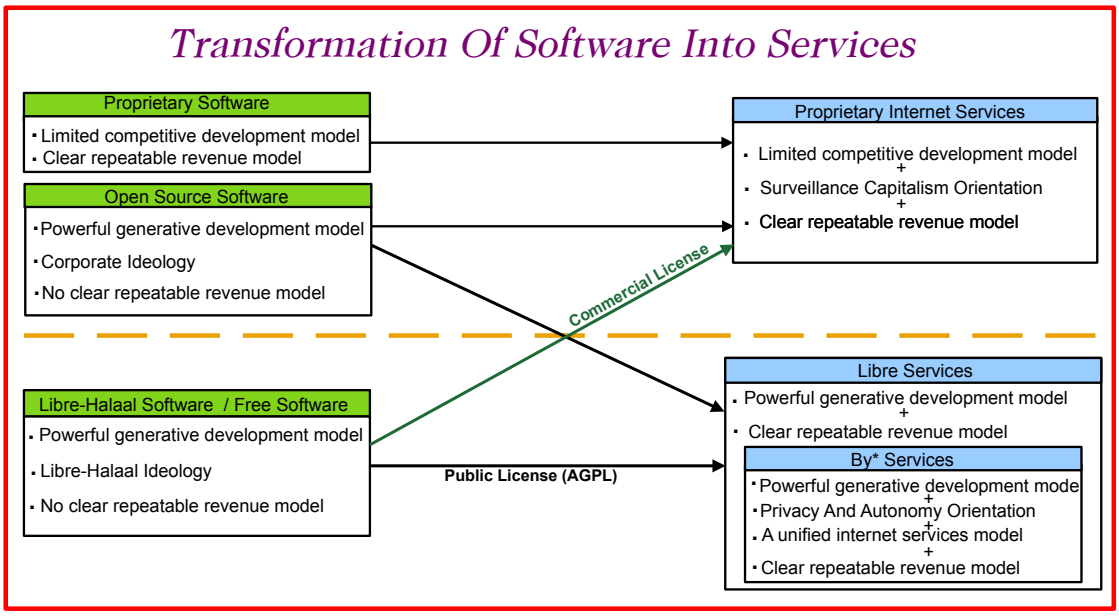

Figure 13.2: Business Ramifications of Software to Service Transformation

Thus, as we look at the software and internet industries of today, we see two largely disjointed cultures. As illustrated in Figure 13.2 we see the FOSS domain, with its powerful generative and propagative development model, but lacking any clear large-scale monetization model. And separate from this we see the proprietary Internet Services domain, with enormous revenue and business consequences, but handicapped in scope and scale by its competitive development model.

Now we are witnessing a further transformational event in the evolution of the internet: a shift of traditional software applications towards a service-based implementation, or what is sometimes called the "transformation of software into services." This is the critical event that now solves the FOSS revenue puzzle. This development unites the generative power of the free software domain with the proven revenue models of the services domain. The transformation of software into services *allows the powerful generative model of FOSS to be invested directly into the powerful revenue model of the Internet Services industry.*

The dashed horizontal line in Figure 13.2 represents two different models and two two different ideologies. The upper part of the dashed horizontal line represents the proprietary American digital model and the convenient convergence of the open-source and corporate cultures. We described some of these dynamics in Section 12.1.6 — Corporatization of FOSS.

The lower part of the dashed horizontal line represents the Libre Services model and the consistent Libre-Halaal Software and Free Software ideologies. In Figure 13.2, note that open-source software feeds into both Proprietary Internet Services and Libre Services. Some software engineers who choose the Libre-Halaal public licensing model choose not to be agnostic and recognize that the moral and ethical ramifications of not cultivating the proprietary internet services are very important. At this point (in 2023), the Libre Services industry is insignificant compared to the proprietary internet services. This is not because of the inherent economics of the two models. It is because of lack of understanding of the economics of Libre Services, business comfort with the traditional proprietary model, and American and Western societal values.

13.6.2 Libre-Halaal Internet Services Capitalism

With the above understandings of:

1. For-Profit/Non-Profit and Proprietary/Non-Proprietary Quadrants
2. Transformation Of Software Into Services

We now add another dimension to the square and turn it into a cube.

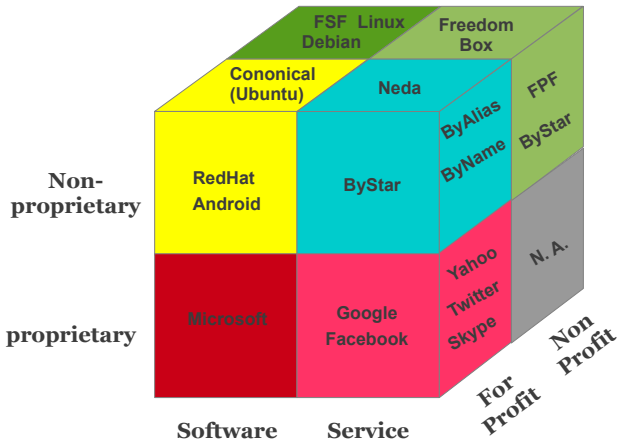

Figure 13.3: The For-Profit Non-Proprietary Quadrant For Internet Services

So, we now have a cube as shown in Figure 13.3. The Libre-Halaal services are positioned in the For-Profit Non-Proprietary Quadrant for Internet Services. Note that in the non-proprietary layer, re-use and collaboration is far richer than the proprietary layer. For example, in the Software slice, Debian and Ubuntu cross progress. In the Services slice the same can happen. For example, ByStar and FreedomBox can cross progress.

The Libre-Halaal Services deployment model breaks both these traditions. It represents 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.

13.7 Attribution Based Economics (ABE) — Instead of Ownership

Ownership and attribution are two separate things.

Through ownership, the IPR model includes attribution. In the IPR model, the one (or the ones) who creates the original copyrighted polyexistential becomes the owner and the polyexistential is attributed to the owner.

In the Libre-Halaal model, the one (or the ones) who creates the original polyexistential becomes the origin of that polyexistential. And the polyexistential is attributed to that origin. In the Libre-Halaal model, by rejecting ownership we are not rejecting attribution. Attribution is an integral part of the Libre-Halaal model.

Capitalism operates on supply and demand as the basis of value. There, open and unlimited availability translates into zero market value. In that model, non-rivalry goods are worthless. So, in that traditional economic model, polyexistentials are not economically sound.

In the context of Libre-Halaal software (FOSS) and internet application services the common revenue generation model has been that of providing value added services rather than from the original development of the core open software. Since, as the core Libre-Halaal software is unrestricted polyexistentials, from a market value standpoint, that developed software is worthless.

Yet, the Libre-Halaalness of polyexistentials leads to limitless availability of useful works. This is a profoundly good thing from the perspective of maximizing value, and thus suppressing it is deeply misguided.

Attribution-based economics is a new model that aims to remedy this state of affairs by changing the basis of value from supply and demand to collective recognition. This is facilitated by a process of "inheritance

attribution" where we collectively agree on the extent of inheritance of ideas and works in other (e.g., derivative) ideas and works, by means of transparent and evolving standards. This model is capable of recognizing a much larger set of valuable contributions, including forms of value that cannot be coerced into a supply-and-demand equation. That is, in this model, there is no need to artificially restrict availability in order for something to be considered valuable. By virtue of the curious property that innovations on the process are themselves subject to the process of recognition in a self-reflective way, we gain accuracy, and by the property that agreed-upon standards apply equally to all, we gain fairness —guarantees that are at best tenuously present in today's economic systems.[45]

13.7.1 Attribution Based Economics (by Sid Kasivajhula)

This book is a collection of thoughts and beliefs. The role of the author is to organize and direct these thoughts and these beliefs. When we use the word "we", this is what we mean.

The concept of Attribution Based Economics is relatively new and the best way to present it is to use the words of one of its origins and attribute it to him.

At the emacsConf-2022, a virtual Emacs conference, Sid Kasivajhula presented some thoughts related to the concept of Attribution Based Economics (ABE). We reproduce parts of that presentation below:

> When we think about the problems of the world we see global warming, war, appropriation, poverty, and among numerous other problems, also the inability to make a living as an open source developer.
>
> Now this last problem may seem a lot less consequential compared to the other ones, but what if I told you that the solution to this problem and the solutions to the others are one and the same?
>
> And it's because there's a common underlying problem at the heart of all of these problems. I'm going to tell you what that problem is in one sentence. You ready for it? It is … the deviation of market value from true value. Let's think about this in the context of existing economic systems such as capitalism and communism.
>
> And of these, I want to focus on capitalism because it is the only nontrivial economic system, really. Communism is more sort of a political means to achieve economic ends. And the other economic systems exist sort of on a spectrum between these two. So let's focus on capitalism. Capitalism has as its basis of value supply and demand. And consequently, there is a great emphasis on this idea of ownership. Now ownership is an idea that made some kind of sense when you have goods and services that are constrained in some way, that are essentially finite in supply. But when you have things like works of software, art, and music, which are essentially infinite in supply, the idea of ownership and supply and demand don't make sense anymore. And yet we employ the institution of property to constrain supply and introduce the idea of supply just so that we can induce a market value in terms of supply and demand in a capitalist economic system. And it's wrongheaded.
>
> How many of us have written copyright declarations like these on our work. It's a lot of work! Especially when we have version control. Now in this example, almost every line is written by a different person, so who owns the code in this case? Who owns the copyright here? Is it some of them, is it all of them, do they share it in some way? It doesn't really make sense, especially when the reason we're employing copyright and ownership in this case is to approximate the idea of attribution, which is what we really care about here. And that brings us to the nature of the solution, which is to move away from an economic system based on ownership and supply and demand, to an economic system based on attribution, instead. That is, moving away from who owns what to who did what and how important was it. And we can do this by the process of Dialectical Inheritance Attribution, which just means that we do it in a collective way using common collectively agreed upon standards that are applied transparently to all. And when we have an economic system that is based on attribution as the source of value in this way, we call it attribution based economics. Now, once we have that, it gives us fairness, effective empowerment of expertise, freedom through incentives rather than through coercion. And privacy as well.
>
> But I could tell you all of those things and some may still say, "Why should I care about this?" There are those who would say that fairness is not a good goal, and that might makes right,

Others kind of see the problem but consider it inevitable. More recently, discussions of loss of privacy in the context of Internet services has become a daily occurrence in mainstream Western press. None of these discussions have any depth and no meaningful cure is even searched for. Many articles and books have been written about the "End of Privacy." Shallow, subdued nagging — that is the position and role of American press on the problem.

15.3.3 Contours of the Cure

In order to cure this disease, we need to conceptualize it in its totality — that of a "Digital Ecosystem".

The Proprietary American Digital Ecosystem cannot be fixed. Its dynamics are taking it to a particular eventuality — the destruction of civilization and humanity.

Instead, we need to erect an alternative digital ecosystem to stand against it.

15.4 Overview of Digital Ecosystems

Our use of the term "Digital Ecosystem" is very broad and includes inter-related software, systems, services, content and societal frameworks including: philosophical, moral, societal, social, economic, business and legal practices – that shape it and are shaped by it.

Here we describe digital ecosystems in four parts.

Ideology — Societal Frameworks:
Digital Ecosystems exist within societal frameworks. Digital Ecosystems are shaped by societal norms and shape people and society.

Very important aspects of societal frameworks which have immediate impacts on the shape of digital ecosystems are laws and models governing polyexistentials. Societal Agreements governing all that is digital (and more broadly polyexistentials) in the West is based on the Intellectual Property (IP) regime. This has shaped the entirety of Western Digital Ecosystems.

Software and Usage Environments:
Software is the digital form that controls other digital forms. As such, it is the foundation of digital ecosystems.

Internet Services:
Internet Services consist of *software execution accessed through a network*. As such, software may no longer be in the immediate possession of the user. Internet Services are therefore a distinct part of digital ecosystems – separate from Software.

Information and Content:
A primary purpose of digital ecosystems is to facilitate production and communication of information and content. In addition to the content itself, facilities and rules governing production, publication and access to content are distinct parts of digital ecosystems.

15.5 The Proprietary American Digital Ecosystem

The broad label that we use for Internet services and software as it exists and is practiced today, is: "The Proprietary American Digital Ecosystem". We include the term American in this label not just because it is dominated by America, but because it is rooted in American values and American rules. The American purely economic model and widespread practices which are based on Western IPR regime have led to dominance of Internet by large American corporations and the governance of the internet through Corpocracy. These corrupt values and models are now being exported and forced on the rest of the world in the name of Internet.

The manner by which the Proprietary American Digital Ecosystem is shaped by American societal norms is multi-faceted. To better understand this, in Figure 15.1 we provide a layered model.

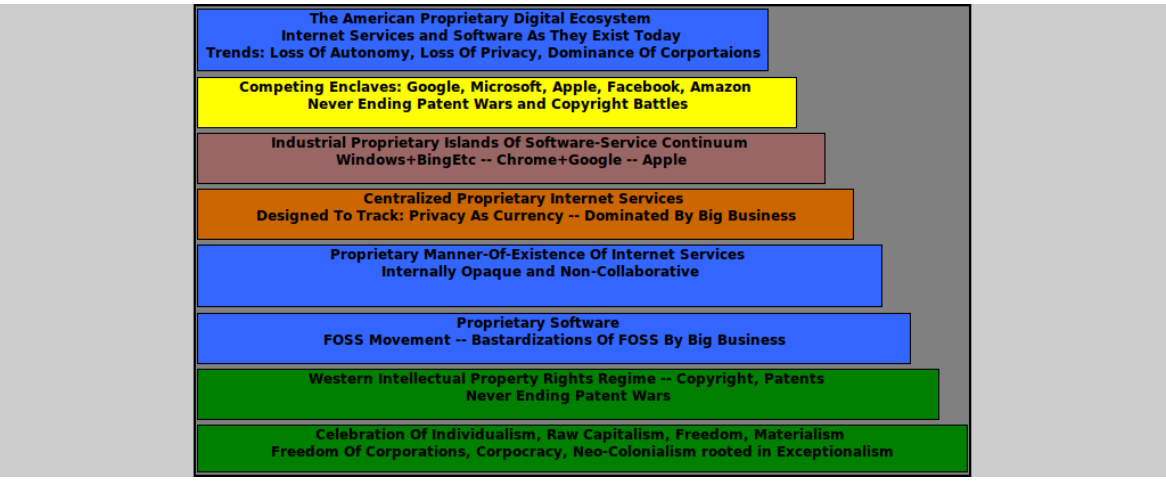

Figure 15.1: The Proprietary American Digital Ecosystem (Layered Model)

The very same eight layers that are presented in Figure 15.1 are the basis of the layered model that we present as a moral alternative in Section 16.2. Note that without recognizing and tackling the underlying root causes of the problems that the proprietary American digital ecosystem present, it is not possible to cure these problems. When the underlying nature of any public digital ecosystem is proprietary, it poses a danger to health of society.

In the following sections, we focus on specific aspects of the above layered model.

15.5.1 Competing Proprietary Digital Enclaves

The Proprietary American Digital Ecosystem comprises of a number of competing Proprietary Digital Enclaves. The proprietary Microsoft digital enclave is one such example. The Microsoft enclave has had its roots in the proprietary software business and is now trying to bring in proprietary services. The proprietary Google digital enclave is another example. The Google enclave has had its roots in the proprietary search business and is now trying to integrate with more software and services. Apple, Facebook and Amazon are examples of other American Digital Enclaves. What they all have in common is that they are all competing in a locked environment driven by Patent and Copyright laws. None of these enclaves were designed ab-initio to be digital environments for humanity. All of these enclaves exist primarily to generate profit for their owners.

This model of being governed by competing proprietary enclaves is normal and even desired by most Americans. The American medical system is similarly structured and so is the American food system. From the outside, many view Americans as purely economic creatures that exist in an industrial context who are fully committed to supremacy of money. While the proprietary American digital ecosystem my be fine for Americans, it may not be for the rest of the world. Bits are without borders and this American disease has been spreading.

Ramifications of manner-of-existence of the proprietary digital ecosystem, matter in two important ways. It matters in terms of service functionality—what the service itself is actually doing. And it matters in terms of policy—what the service provider is doing.

15.5.2 Ramifications on Service Functionality

Regarding service functionality: existing proprietary services such as Google, Yahoo, YouTube, Facebook, Microsoft, Apple, and virtually every other service—these are strictly controlled assets of their owning companies, heavily defended by patents and copyright. The software that runs the service is closed, such that the true service functionality is unknown. This means that as the user of the service you have no knowledge of what the service is actually doing behind the scenes. For example, you have no knowledge of what the service is doing with your personal information. Every item of information you provide to the service, either implicitly or explicitly—every communication, every search query, every website visited, every mouse click—can be used by the service provider for unknown purposes, without your knowledge or consent.

Anonymous By* Services	ByAnonymous	ByLeak			
Inter-Autonomous Interaction Facilitaion	ByInteraction	ByHookup			
Federated By* Services	ByTopic ByEvent	ByContent ByBinary	BySource	BySearch	ByLookup
Controlled By* Services	ByFamily	ByWhere	ByMemory	ByEntity	
Autonomous By* services	BySMB ByAuthor	ByName ByArtist	ByAlias ByNumber		
ByStar Central	By-Star Neda	BySource LibreCenter	ByBinary Free Protocols	Liber Services	Halaal Software

Figure 16.1: ByStar Domains Table

- 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.

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.

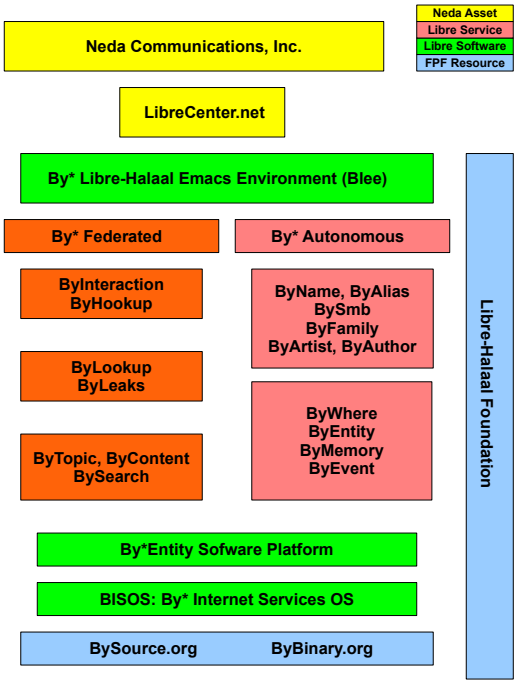

Figure 16.2: Libre Services Supporting Organizations

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.

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.

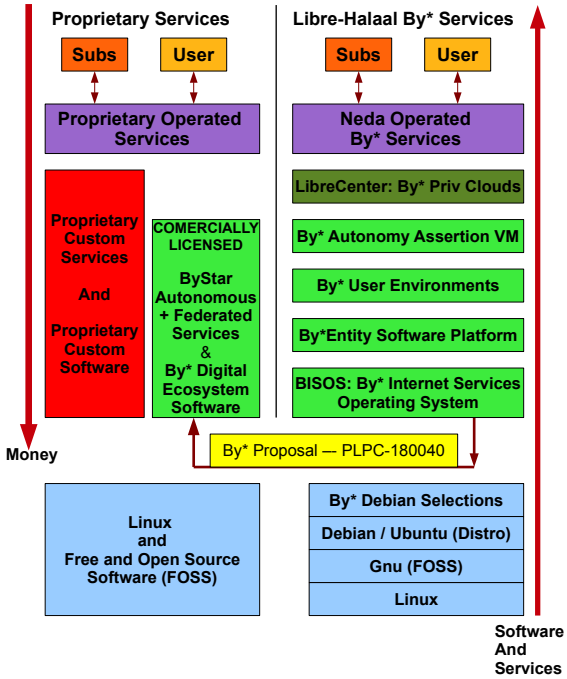

Figure 16.3: ByStar Value Chain

where the user is more committed to the service. So, all our investments in ideology are actually also business wise.

All of the ByStar value chain software is Libre-Halaal (Free and Open Source) software. ByStar software in Figure 16.3 is shown in two different places.

The software in the lower part represents Debian and/or Ubuntu GNU/Linux and the specific software packages that we have chosen. These are externally developed open-source software packages which are typically subject to the free software GPL license (or similar) which permits their inclusion in proprietary services. This is often referred to as ASP loophole.

The software in the middle is the software that Neda has developed. It is subject to the "Affero General Public License Version 3" (AGPL3) and Neda Commercial License (Dual Licensed). AGPL3 closes the ASP loophole. Any ASP which uses ByStar software must subject its changes and improvements to AGPL3 and make its changes and improvements publicly available. Those ASPs not wishing to do so, can use ByStar software through the Neda Commercial License.

In the left column of Figure 16.3, we illustrate a typical proprietary ASP who is incorporating ByStar as part of its services based on the Neda Commercial License.

In this environment the model for implementation of By* service functionality is not one of original software development. Rather, it is a matter of selection and integration of already available software packages. Virtually all existing By* service functionality has been created this way—in building By* we have written almost no new software components at all.

Thus we are not so much in the business of software development, as we are in the business of software integration. But the integration of software components to produce a coherent service is far from trivial. We have created a sophisticated technical integration environment for this purpose, called the **BISOS: ByStar Internet Services Operating System** [30].

Design of BISOS and the **ByStarEntity Software Platform** recognize the evolution of underlying external software (bright blue) in the ByStar value chain. This is the extraordinary magic of Libre-Halaal software and services: the ability to take things and reuse them at extremely low cost. This is the fundamental growth dynamic of Libre Services, and the powerful generative force that is lacking in the proprietary model. This is the key dynamic that causes the By* Libre Services eventually to surpass the proprietary model entirely in terms of features and functionality.

16.2.7.4 ByStar Open Business Plan

The halaal manner-of-existence of software creates a powerful generative development model for Halaal Internet Services. This generative model is absent from Proprietary/Haraam Internet Services. Thus Libre-Halaal Internet Services have a major advantage and can compete directly with Proprietary/Haraam Internet Services in terms of development.

The basic recurring revenue models for Libre-Halaal Internet Service providers are essentially the same as for Proprietary/Haraam Internet Service providers. Thus, in terms of revenue generation, Halaal and Haraam services are on an equal footing.

As part of our responsibility to create a viable implementation construct, we have fully analyzed the business dimension, and we have formulated the business model in the form of an Open Business Plan, titled:

> **The Libre-Halaal ByStar Open Business Plan**
> **An Inversion to the Proprietary Internet Services Model**
> **Neda Communication Inc.'s Open Business Plan**
> http://www.by-star.net/PLPC/180014 — [29]
> http://www.neda.com/strategicVision/businessPlan

ByStar open business plan is available in 3 forms; the Condensed Summary (about 12 pages), the Executive Summary (about 15 additional pages) and the full plan (about 85 pages).

Our business plan is viable because we understand the critical dynamics of polyexistentials. The current direction of the internet services industry does indeed present a grave hazard to humanity, and we will indeed safeguard humanity against this. These extraordinary claims provide a unique and powerful marketing message. And they also happen to be true.

16.2.8 Understanding the Full ByStar Picture

We have given you a brief summary of ByStar above.

This summary is over simplified and captures the gist of a series of articles that we have developed to analyze and describe various aspects of ByStar.

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. Below we provide a description of how these documents collectively draw a comprehensive picture.

The big ByStar picture is shown in Figure 16.4. Each of the layers in this figure represents either a conceptual definition (shown in blue), or an actual software/service implementation (shown in orange). Each layer builds on the layers beneath.

The layers in Figure 16.4 are color coded. Each of the layers are either:

A Conceptual Layer. Representing concepts. Layers 1,2,3,4,7 and 8 are in Green, Blue and Yellow.

A Tangible Layer. Representing software/service implementations. Layers 5 and 6 are in Orange and Brown.

The tangible layers are bound by the conceptual layers underneath them and receive legitimacy from those concepts.

The conceptual layers are validated by the tangible layers.

The green layers (1 and 2) at the bottom are philosophical, moral and societal. Their scope is wider than the moral digital ecosystem that we are after. Generally speaking, these are not the domain of engineers. They are the domain of ethicists, philosophers and sociologists.

The blue layers (3, 4 and 8) are philosophical, moral, societal, social and engineering aspects of digital ecosystems that require direct involvement of engineers and the engineering profession. The yellow layer (7) addresses economics and business dimensions of ByStar.

The orange/brown layers (5 and 6) are engineering constructs. They are in-use software and in-use Internet application services.

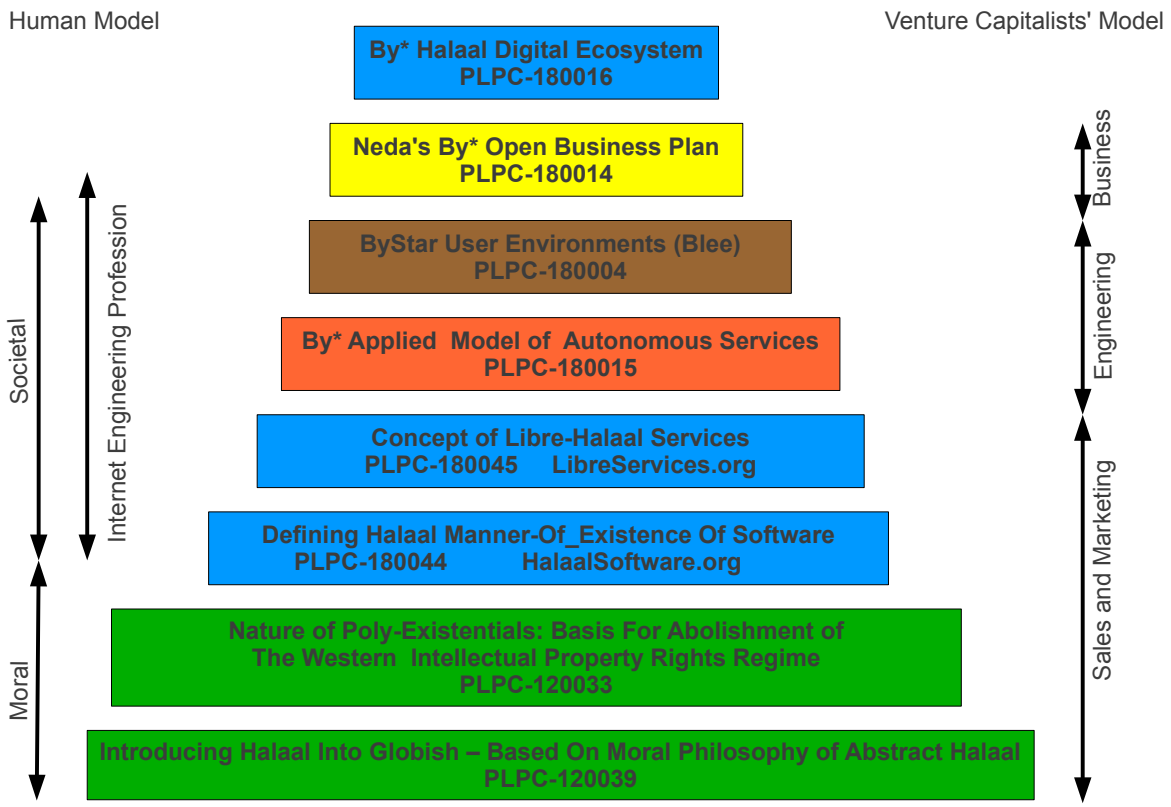

Figure 16.4: The Libre-Halaal ByStar Digital Ecosystem Conceptual Layering

In ByStar Roadmap: http://www.by-star.net/bxRoadmap
we provide a reading roadmap to ByStar related articles.

Figure 16.4 shows how the moral, legal, societal, engineering, economic and business dimensions of the ByStar Halaal Digital Ecosystem are layered as described above.

Note the differing characterizations of this layering on the left and right. Both characterizations are valid, but they reflect entirely different viewpoints. The left side characterization is called "The Human Model," and reflects the philosophical, moral and societal elements of the model. It also identifies the role of the engineering profession in maintaining these elements. The right-side characterization is called "The Venture Capitalist Model," and is very different from the "The Human Model." The same elements are present, but now represent their significance as part of an investment strategy. Thus, the moral and societal concerns within the human model are now viewed as a sales and marketing opportunity. This makes clear that when dealing with Venture Capitalists, issues of morality and societal welfare are not the topic of discussion. In this regard Venture Capitalists need only understand that human beings are in fact concerned with vital moral considerations such as "privacy" and "autonomy," and that these considerations have powerful sales and marketing consequences. And that our unconventional strategy of overturning their sacred-cow – Copyright and Patent model – gives us a huge competitive advantage.

The gigantic picture we have drawn in Figure 16.4 is a blueprint. It represents a complete framework for collaborative work towards an alternative to the current proprietary digital ecosystem. By aligning ourselves with the natural forces and dynamics of polyexistentials, and by means of large-scale unrestricted collaboration, we can achieve this.

16.3 ByStarEntity Realization Models — By Way of Some Examples

Let's explore ByStar in the context of a few examples. All these examples are completely fictional.

Let's consider Bob Smith, a 46-year-old university researcher and his 15-year-old daughter, Alice Smith. Alice is a freshman (9th grader) at Garfield High School (GHS).

Bob wishes to have the following in ByStar.

1. An Autonomous ByName BxEntity for his private and public use – 5.bob.smith.byname.net (he is the 5th bob.smith requesting byname services).
2. An Autonomous ByFamily BxEntity for his family – 8.smith.byfamily.net.
3. A Controlled ByFamily BxEntity for his daughter Alice – alice.8.smith.byfamily.net (Alice is a minor and Bob wishes to have the option of overseeing her communications).
4. A Controlled ByWhere BxEntity for their condo in Kirkland, WA – 1-98034-3681-74.bywhere.net (say for reliable driving directions).

There are 3 different realization models for Autonomous BxEntity-s.

- Shared Cloud Autonomous Model
- Hosted Private Cloud Autonomous Model
- Premise Private Cloud Autonomous Model

Bob is concerned about privacy and prefers the "Hosted Private Cloud Autonomous Model" over the "Shared Cloud Autonomous Model". He trusts the ByStar model enough not to need the "Premise Private Cloud Autonomous Model".

In the following sections we describe ByStarEntity realization models in the context of Bob and Alice's example.

As we go through these examples, we will also be comparing them with their counterpart in the Proprietary American Digital Ecosystem.

16.3.1 ByStarEntityId Registrations

Through ByStar, Bob needs to have an Autonomous ByName Registration, an Autonomous ByFamily Registration and a Controlled ByWhere Registration.

So, Bob goes to http://www.byname.net and provides his name "Bob" "Smith" and an email address and agrees to conform to ByStar usage policies and in return, he receives:

- 5.bob.smith.byname.net – BxEntityId=23. 1.2.7.3 .32674 – BxEntityIdPassword=

Similarly, Bob goes to http://www.byfamily.net and provides his autonomous BxEntityId=23.1.2.7.3.32674 and gets:

- 8.smith.byfamily.net – BxEntityId=23. 1.2.9.5 .4689

He then provides his autonomous BxEntityId=23.1.2.7.3.32674 and gets:

- 1-98034-3681-74.bywhere.net – BxEntityId=27. 2.2.6.4 .4689

for a ByWhere controlled entity.

All of the above were external registrations. In the ByStar model, Bob himself has now become a registrar for some ByStarEntitys.

Under the 8.smith.byfamily.net domain, Bob now registers

- bob.8.smith.byfamily.net – as BxEntityId=23. 1.2.9.5 .4689 .1

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".

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.

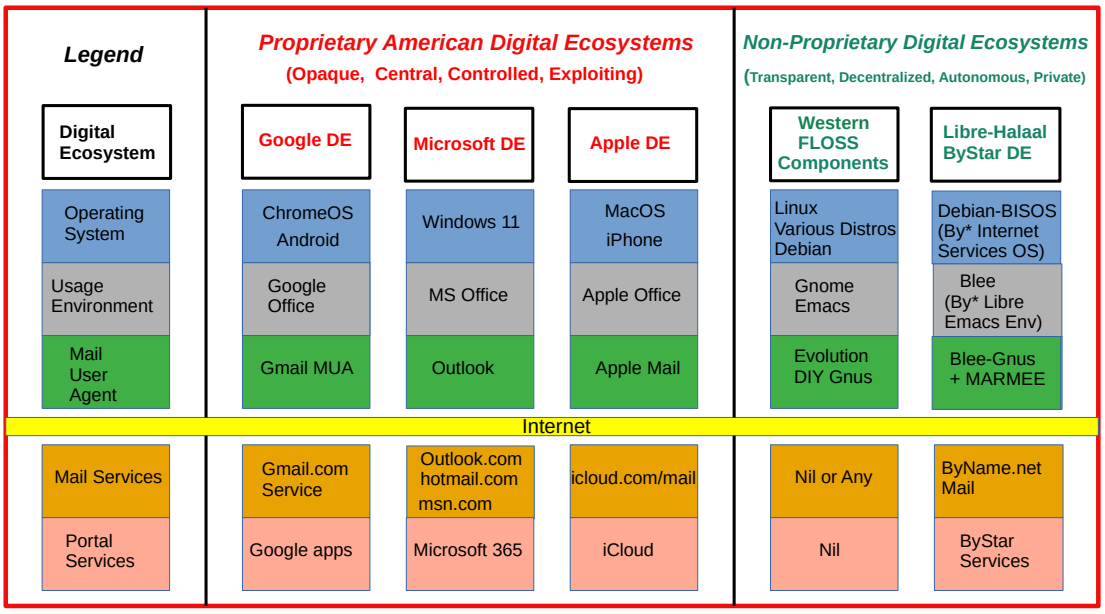

Figure 17.1: ByStar and Proprietary American Digital Ecosystems

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.

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).

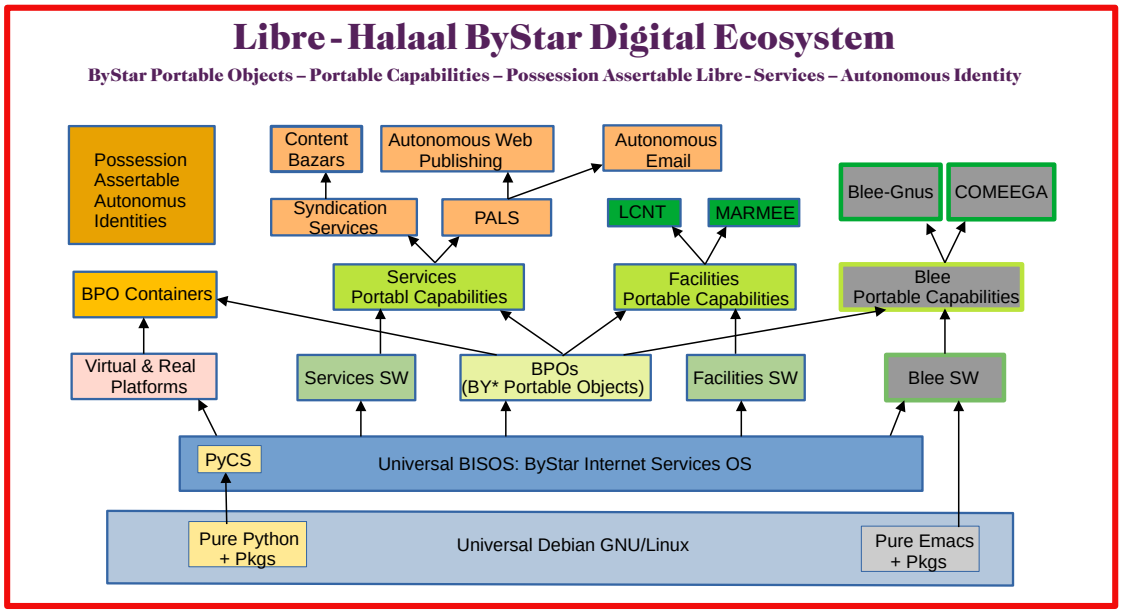

Figure 17.2: ByStar Portable Object Capabilities

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.

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.

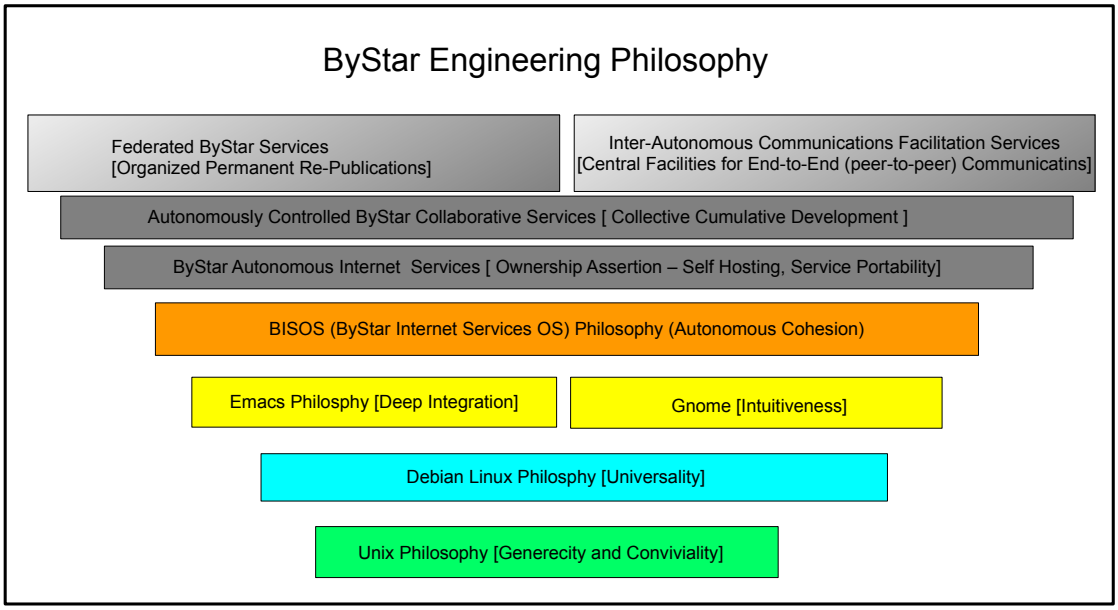

Figure 17.3: ByStar Engineering Philosophy

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.

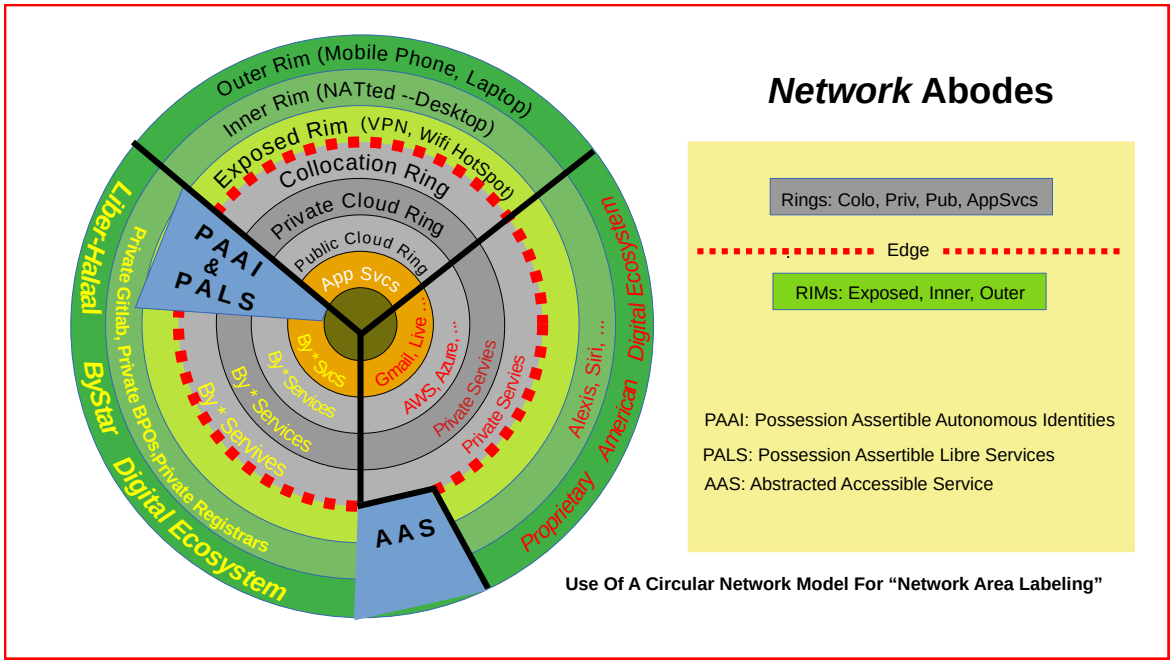

Figure 17.4: Network Abodes: A Circular Model For Network Area Labeling

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.

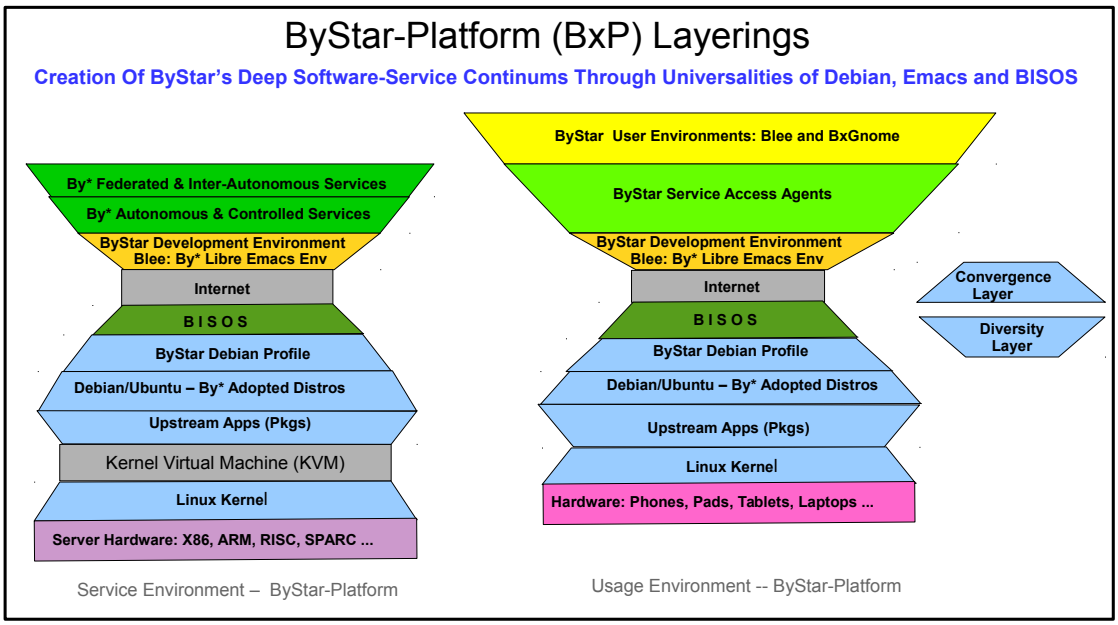

Figure 17.5: ByStar Platform Layerings and Software-Service Continuums

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?

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.

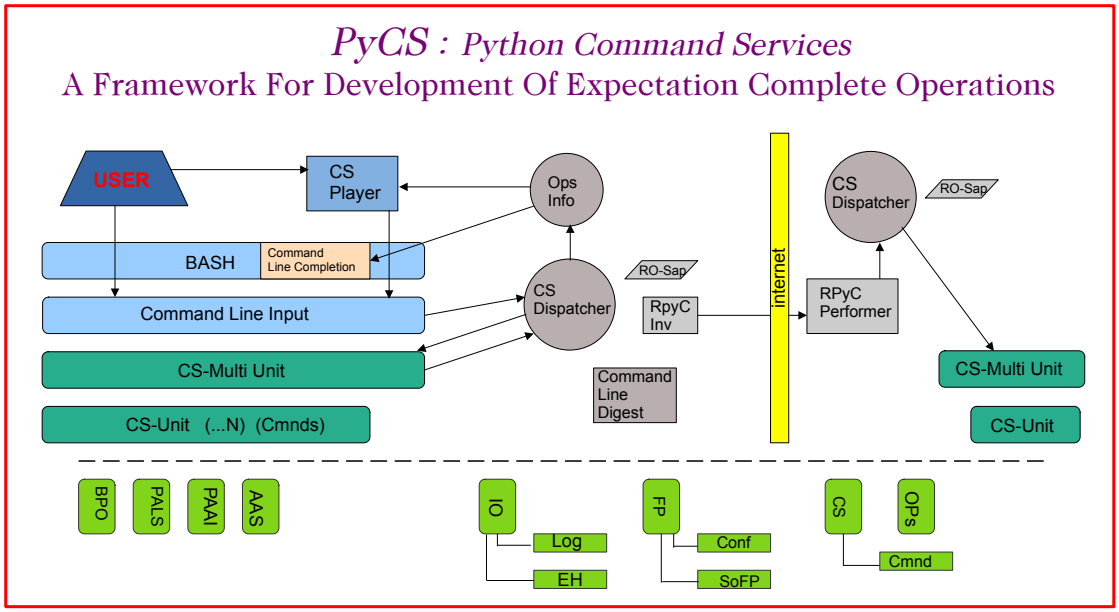

Figure 17.6: BPyF (BISOS Python Platform) and PyCS

- 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".

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.

17.10 ByStar Libre-Halaal Emacs user Environment (Blee)

Blee, ByStar Libre-Halaal Emacs Environment, is ByStar's primary usage environment. It is fully integrated with BISOS and Blee is aware of all ByStar conceptual constructs.

Conventional OS wisdom calls for separation of OS functionality from user-interface/usage-environment. But BISOS is not a traditional OS and Emacs is not a traditional usage-environment.

The concepts of universal platform and software-service-continuum that we presented have ramifications on usage and user experience. ByStar services can thus be greatly enhanced by providing the user with a "matched" environment—a user environment that is closely integrated with the service. This provides the user with features and capabilities that go far beyond what is possible using the traditional generic browser access.

By fully integrating BISOS and Blee, we accomplish a degree of cohesion and conviviality within the ByStar Digital Ecosystem that is absent in the American internet environments. Blee is significantly more broad and sophisticated than other usage environments.

In Figure 17.7 we depict that Blee is part of BISOS and that Blee includes Emacs. Think of Figure 17.7 as a containment hierarchy. The Libre-Halaal ByStar Digital Ecosystems contains both Usage-Side BISOS platforms and Service-Side BISOS platforms. The Usage-Side BISOS platform contains Blee. And Blee contains Emacs.

Emacs is a 40-plus years old editor centered usage environment, with a Lisp engine at its core and an extremely powerful display and editing engine in its nucleus. Emacs is one of the oldest Free Software in continuous use. Over the past 40 plus years, sophisticated engineers have added support for anything and everything to Emacs. Emacs's well designed fundamental abstractions make it the most convivial usage environment. Emacs is a multi-lingual editor that supports most human languages. But out of the box, Emacs is clunky and difficult to use.

Blee serves two purposes:

1. Blee integrates with BISOS and ByStar services and ByStar concepts.
2. Blee makes Emacs less clunky and easier to use without losing any of Emacs's conviviality.

Figure 17.7 depicts that Emacs contains a very powerful display engine, a very powerful Lisp engine, a very powerful input methods engine and a very powerful applications development framework. Emacs is primarily known as a textual environment. But it is more than that. Emacs is now capable of handling multimedia (images/audio/video) as well. Emacs's display engine supports bidirectional (bidi) text and is fully

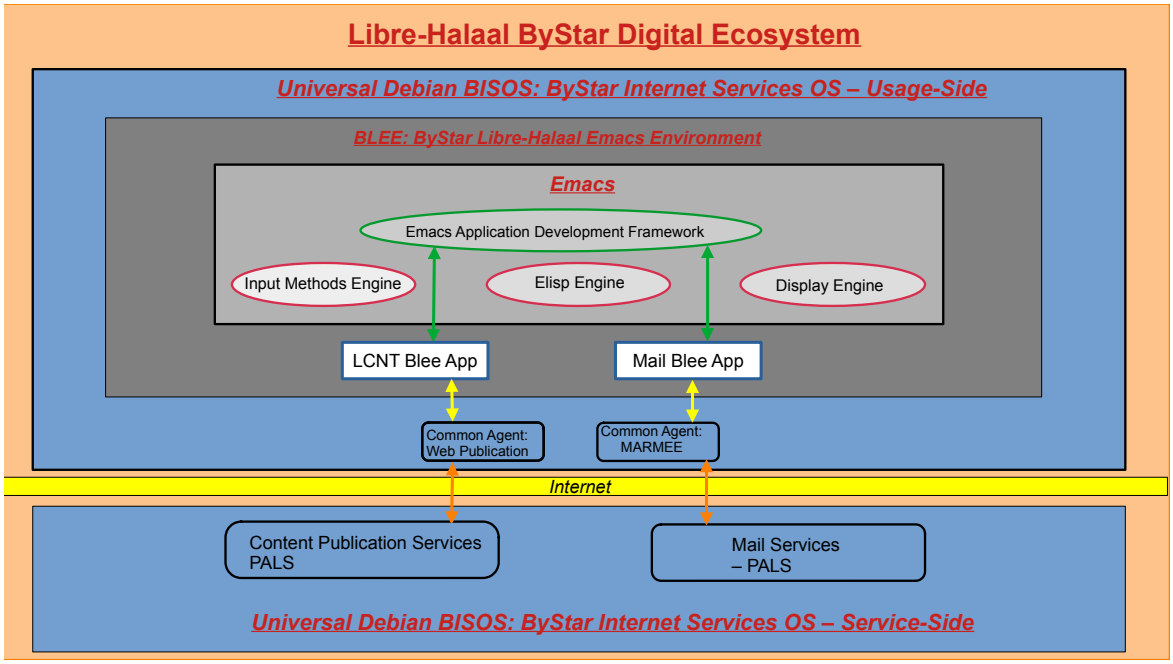

Figure 17.7: A Blee Centric Perspective Of By* Digital Ecosystem

multilingualized. Emacs supports input methods for many human languages. Emacs's Lisp engine and its applications development framework allow for convenient development and customization of applications.

Blee builds on Emacs.

Figure 17.8 shows some of the salient features of Blee. For each of the programming languages of BISOS (Python, Bash, Elisp, LaTeX, Web environment and C/C++) Blee provides Interactive Development Environments (IDEs) that go beyond the language and include the frameworks and libraries of BISOS.

All coding and all writing in BISOS is based on a model called: COMEEGA (Collaborative Org Mode Enhanced Emacs Generalized Authorship).

COMEEGA is the inverse of Literate Programming, where code is written in native programming mode and then augmented with comments and doc-strings in org-mode. COMEEGA provides the necessary tools to switch between native-mode and org-mode conveniently and is used in BISOS and Blee to ensure a high degree of consistency.

The usage of BISOS's Integration Framework (PyCS) described in Section 17.9 is facilitated in Blee through Blee Command Services Players. Each Command Service, whether it is a command-line or a remote-operation (microservice), is expectations complete and can be run more conveniently through Blee.

Of course, all of BISOS and Blee is self-documented. The documentation takes the form of Blee-Org-Panels which take the form of related org-files. Unlike typical documentation, Blee Org Panels are active. You can modify, configure and customize BISOS and Blee from within Blee-Org-Panels. Additionally, Blee-Org-Panels can be used by users to organize their own information and applications.

All of the key abstractions of BISOS (BPO, PALS, PAAI, AAS), can be managed through Blee.

The combination of Blee and BISOS fully wraps development, management and usage of ByStar services. Such universality facilitates continuous growth of ByStar.

17.11 BISOS Software-Service Continuum Apps

Thus far, we have provided an overview of the BISOS infrastructure. Based on these, there are various capabilities that the owner-user can profit from. In BISOS, we call these capabilities "Software-Service Continuum Applications" (SSCA).

As described in Section 17.8 — BISOS Model of Platform Universality and shown in Figure 17.5, part of the

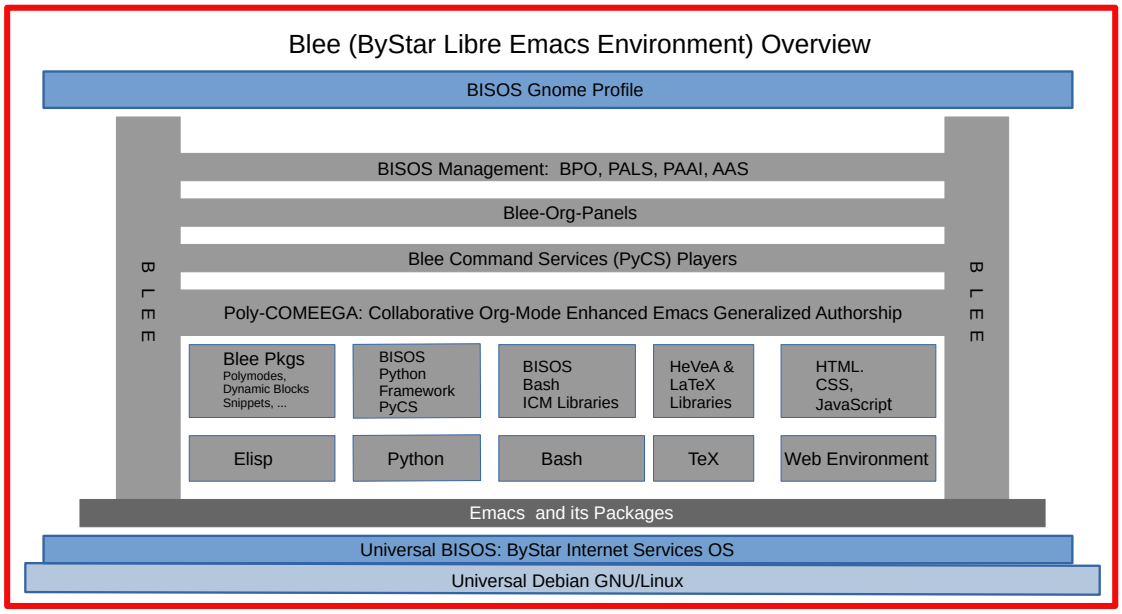

Figure 17.8: Overview of Blee Features

capability is realized in software on the user side and part of the capability may realized on the services side. Since both the user-side and the service-side are based on the universal BISOS platform the resulting combined capability is consistent and flexible.

There are many BISOS software-service continuum applications and the model is open ended. There is an SSCA for genealogy, for photo galleries, and much more.

In BISOS, Software-Service Continuum Applications have a common structure. They typically consist of a three layered stack.

1. BISOS-Svc-Layer: BISOS Services Layer runs as a service-provider and interacts with the BISOS-Sw-Layer.
2. BISOS-Sw-Layer: BISOS Software Layer that facilitates work of Blee-SSCA-Agent and interacts with BISOS-Svc-Layer.
3. Blee-SSCA-Agent: Emacs-Lisp Code of Blee which the user interacts with.

The general model of interactions between BISOS-Sw-Layer and BISOS-Svc-Layer is typically that of Remote Operations where BISOS-Sw-Layer assumes the invoker role and BISOS-Svc-Layer assumes the performer role.

There are two BISOS software-service continuum applications that are foundational. These are email processing and content generation and self-publication.

17.11.1 BISOS Email Software-Service Continuum App

Email is a foundational application. BISOS Email SSCA is structured as follows: The Blee-SSCA-Agent for email is called Blee-Gnus. The BISOS-Sw-Layer is called MARMEE (Multi-Account Resident Message Exchange Environment). BISOS-Svc-Layer is called BISOS-Mail-Service.

Figure 17.9 depicts Blee-Gnus and MARMEE in the context of split-MUA (Mail User Agent) Blee-Gnus is the usage environment and MARMEE addresses mail protocols processing. Gnus is a very flexible mail processing environment which is integrated into Emacs.

BISOS uses a modified version of qmail called BISOS-qmail as the MTA (Mail Transfer Agent). When used it as a traditional MTA, we refer to it as PALS-qmail. And on the usage side we call it MARMEE-qmail. For incoming mail within MARMEE, BISOS uses offlineimap.

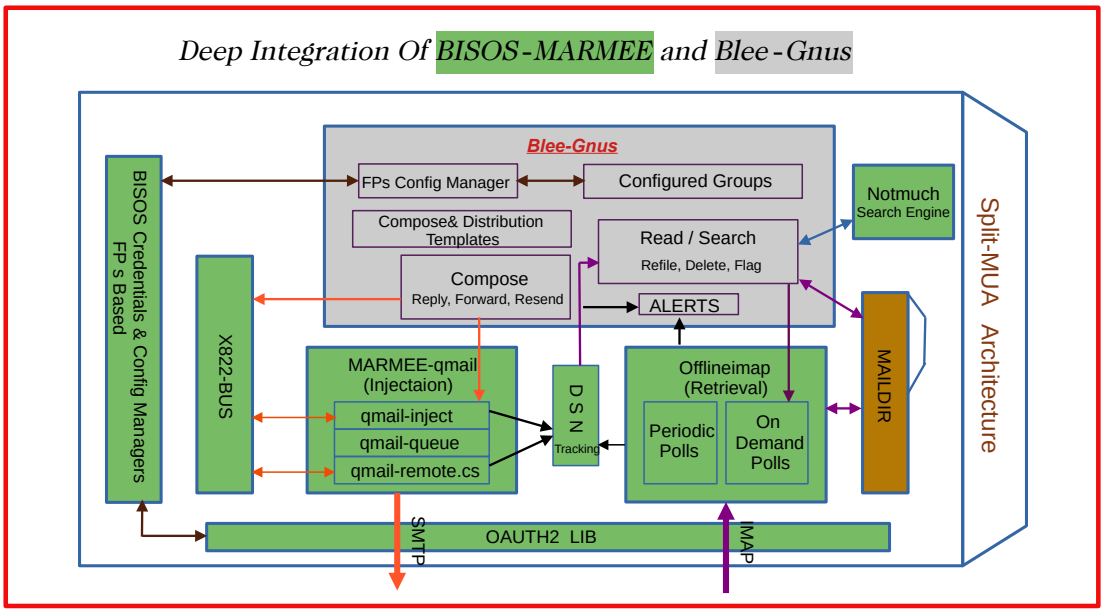

Figure 17.9: Blee-Gnus and MARMEE as a Split-MUA

It is possible to use MARMEE and Blee-Gnus to access other email services. This is done through configuration of an AAS (Abstracted Accessible Service). For example, in addition to ByStar email, an owner-user can also access her gmail account with Blee-Gnus.

17.11.2 BISOS Content Generation and Self-Publication

BISOS software-service continuum application for content generation and self-publication is called LCNT (Libre Content).

The content generation capabilities of LCNT are akin to Microsoft-Word and PowerPoint. But the model of content generation in BISOS is very different from Microsoft-Word and Microsoft-PowerPoint. We use LaTeX for document processing and COMEEGA-Blee for authorship.

A pictorial overview of multi-media content generation is provided in Figure 17.10. A single LaTeX source file is used to embed text, images, audio and video. This single source file is then processed in a variety of ways with a variety of tools including XeLaTeX and HeVeA to produce a variety of outputs including pdf and html. Multimedia frames/slides are then disposed using reveal.js.

BISOS-LCNT also includes facilities for self-publication where the above mentioned generated content can be pushed to owner-user's web sites and can also be syndicated.

17.12 Privacy, Security and Regulatory Ramifications of BISOS

Technological design of BISOS is very different from the technological design of proprietary American internet application services.

BISOS capabilities revolve around the abstraction of the individual and its belongings and delivery of possession and control of those abstractions to the individual. In BISOS, you own and possess your own data and you can own and possess your own services.

BISOS's philosophy is privacy by design.

Privacy by design is the antithesis of the proprietary American internet application services model, which is based on surveillance by design. Surveillance by design leads to centralized architectures and control, while privacy by design architecture leads to distributed architectures and autonomous control.

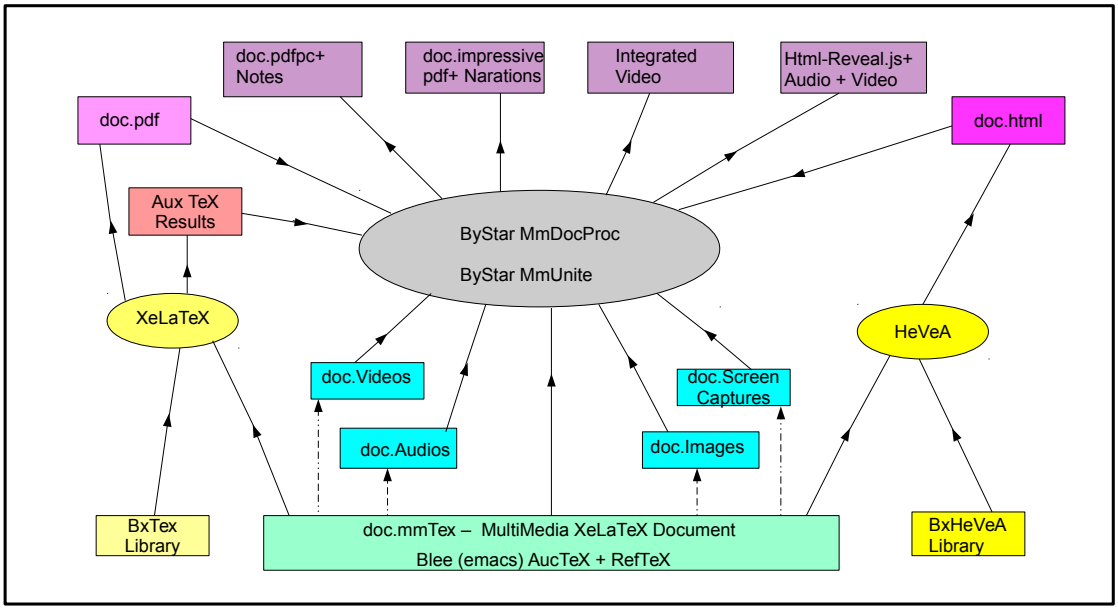

Figure 17.10: ByStar Multimedia Document Authorship And Generation

BISOS's fundamental design has immense security ramifications. Combinations of BPOs, PALS and service recreations capabilities of BISOS render many traditional security models inapplicable. In conjunction with being transferable, autonomous Libre Services are very easily recreatable. In many instances, upon detection of intrusion (or even periodically), after capturing the context of an exploit, a fresh new service replaces the contamination. All of this can be automated.

Since proprietary American internet application services are fundamentally designed for surveillance, the needed societal regulations are complex and ineffective. Since ByStar and BISOS are fundamentally designed for privacy, societal regulations are very simple and effective. ByStar is designed to be self-regulating. ByStar promotes proactive regulations as opposed to the current model of reactive regulations. The engineers have done the work. The politicians just need to understand. The bulk of the needed regulations can amount to exclusive use of PALS Libre Services as defined in Section 15.2.3.1 — Definition of Possession-Assertable Libre Services.

17.13 ByStar and Uses of BISOS

The specific purpose of BISOS is to facilitate the creation of Libre-Halaal ByStar Digital Ecosystem.

Let's see how ByStar uses BISOS to realize the underlying model and capabilities of the Libre-Halaal ByStar digital ecosystem.

- ByStar is about redecentralization of the internet. Control and ownership is transferred from central corporations to distributed individuals (as autonomous entities). Rise-of-the-middle model is rejected in favor of the autonomous edges model.

  BISOS was designed for all of that.

- ByStar software and internet services are un-owned/publicly-owned and internally transparent.

  BISOS 'is Libre-Halaal software subjected to AGPL. The entirety of ByStar Individual Services can be reproduced based on their available sources.

- Broadly speaking, ByStar services fall into these 3 categories:

  1. ByStar Individual (Possession-Assertable/Autonomous) Services.
  2. ByStar Content Syndication Services.

3. ByStar Facilitated Direct and Assisted Inter-Autonomous Interaction Services.

BISOS PALS address (1) and (3). BISOS's Libre Content (LCNT) addresses (2).

- ByStar individual services represent real individuals in the real world. In ByStar, real individuals have real autonomy, real control and real ownership of their own ByStar individual services. ByStar individual services are edge-oriented and can be externally-hosted or self-hosted. When externally hosted, ByStar individual services are regulated to be portable and possession-assertable. For example, Mohsen's ByStar individual services is:
  `mohsen.1.banan.byname.net`.
  You can have your own as: `first.last.byname.net`. Since you own your domain and since you can fully possess the service and your data at will, you have real autonomy.

  BISOS PAAI is designed to support deep domain names and PALS are transferable.

- ByStar individual services are "Possession-Assertable". A portable hosted service can be transferred to the individual who owns it where the individual becomes her own Application Service Provider. For example, people can run their own fully private email servers in their own houses. Just like Hillary Clinton.

  Some early examples of ByStar possession-assertable individual service factory domains are: `ByName.net`, `ByFamily`, `BySMB`, `ByMemory`, `ByAlias`, `ByWhere`, `ByAuthor` and `ByArtist`.

- Direct inter-autonomous relations such as Facebook style photo sharing are accomplished through the individual's own possession-assertable authorization services (individualized OAuth services). Healthy equivalents of capabilities of typical social networks can be created with PALS authorization services where each individual uses his own OAuth service to grant access to his own resources.

  BISOS-OAuth supports this.

- Syndication services such as Youtube style content publication are clearly regulated and integrated with ByStar content production capabilities of individual services. Some early examples of ByStar syndication services are: `ByTopic`, `ByContent`, `ByLookup`, `ByEvent`, `BySource`, `ByBinary`, `BySearch`.

- Facilitated inter-autonomous interaction services such as dating, auction and trade services, are clearly regulated and well integrated with ByStar identity services. Some early examples of ByStar inter-autonomous facilitated interaction services are:
  `ByInteraction`, `ByHookUp`, `ByEntity`.

- ByStar also functions as a hierarchical registrar. For example, Mohsen Banan's registration of `mohsen.1.banan.byname.net` with the `byname.net` registrar results in ownership of `mohsen.1.banan.byname.net` by Mohsen Banan. This domain registration is independent of the service provider that is hosting the portable and possession-assertable individual service. The combination of the portable owned domain and the portability of publicly-owned ByStar individual services allows for transparent transfer of an individual service from one hosting service to another hosting service. This accomplishes the equivalent of Wireless Local Number Portability. Such fundamental user freedoms are absent in the American internet.

  BISOS PALS are portable and transferable.

- ByStar is mostly self-regulated. Upon assertion by the user-owner, the ByStar individual service provider must fully and permanently delete the possession-asserted service and all her data. Or otherwise, ab initio let the owner know that her data will be maintained. Within applicable jurisdictions, ByStar service providers must comply with Lawful Interception (LI) and satisfy regulatory requirements and legal obligations towards Law Enforcement Agencies. Syndication and facilitated inter-autonomous relation providers are subject to known and clear regulations and restrictions.

Mohsen BANAN is an Iranian software and internet engineer.

The software and internet services that he publicly offers all conform to the definition of Libre-Halaal Software and Libre-Halaal Internet Services. All of his public writings are web published and unrestricted.

He has never applied for a patent. As an expert witness he has assisted in legal efforts involving invalidation of a number of patents.

He is the principal architect of the ByStar Digital Ecosystem and the primary developer of BISOS (ByStar Internet Services OS) software.

He has written this book in the context of his responsibilities to his profession.

Knowledge, know-how, uses of know-how, ideas and information are inherently non-scarce. They are **polyexistentials**. Unlike monoexistentials which exist in singular, polyexistentials naturally exist in multiples and are inherently not scarce.

What is abundant in nature is being made artificially scarce through man-made ownership rules called copyright and patents. These ownership rules then permit a certain group, which usually takes the form of corporations, to economically profit from these unnatural and economically motivated artificial scarcities.

In this book we analyze the topic of Intellectual Property Rights (IPR) from a new perspective. The topic of restriction of polyexistentials and Western IPR are one and the same. Yet, the concept of polyexistentials has not appeared in prior discussions of this topic. This is the very first time that the concept and the word "polyexistetials" are being introduced.

The concept of Intellectual Property is invalidated by the model of polyexistence. This proof is based on logic that is rooted in nature of existence and nature of possession and the requirement for ownership to be in harmony with the nature of possession and existence.

Having rejected the Western IPR regime as an erroneous model for govenrnance of polyexistentials, we introduce the **Libre-Halaal model of governance of polyexistentials** towards facilitating conviviality of tools.

We then focus on the digital world and introduce the **Libre-Halaal By* (ByStar) Digital Ecosystem**, as a moral alternative to the existing proprietary American digital ecosystem. We need non-proprietary digital ecosystems. And that is what ByStar is. Equipped with a multidisciplinary blueprint, we offer our initial implementation of the ByStar digital ecosystem as a starting point towards concrete solutions.

It is only through full rejection of Western IPR regime and its deep roots in Americanism that humanity can be rescued.

Distinct And Different

What we are trying to do in this book is distinct and different from the many other books and articles that have been written about Intellectual Property. First, most books on this topic are written by Western lawyers, politicians, economists, journalist, sociologists and academics. We are engineers and we are not Western. Second, this is the very first time that the philosophical concept and terminology of polyexistentials is being used to analyze this topic and this domain. Third, our treatment of this subject is genuinely independent. We are not doing it for money or hype. Finally, unlike most other writings on this topic which amount to naggings of impotents, we are putting a blueprint for a cure on the table. As engineers, we are offering explicit multidisciplinary solutions.

ISBN 978-1-960957-15-3

9 781960 957153