Nature of Polyexistentials:

Basis for Abolishment of the Western Intellectual Property Rights Regime

And Introduction of the Libre-Halaal ByStar Digital Ecosystem

Mohsen Banan

PLPC-120033.p3cel

First US Edition

Available on-line at:
https://github.com/bxplpc/120033

Chapter 2

Nature of Polyexistentials

Here we categorize our world into two:

1. Monoexistentials
2. Polyexistentials

There are things in nature that exist in singular and there are things that exist in multiples.

That which exists in nature in singular, we call monoexistential. Examples of monoexistentials include: tangible physical objects, a pencil, land, Internet domain names, bandwidth. Chemistry and physics are the realm of monoexistentials.

That which exists in nature in multiples, we call polyexistential. Examples of polyexistentials include: knowledge, ideas, information, the digital entities. Information theory is the realm of polyexistentials.

This natural categorization then permits us to revisit the question of ownership of polyexistentials which simply maps to the Western Intellectual Property Rights (IPR) Regime. The topics of Western IPR and ownership and restriction of polyexistentials are one and the same.

Our analysis is from the perspective of the possessed. Traditional Western IPR analysis has always been from the perspective of owner/creator/author. The perspective of the possessed represents societal and human liberties.

This is the first introduction of the concept of polyexistentials which leads to a different way of looking and analyzing Western IPR regime. This duality of analysis based on the perspective of author/owner vs. the perspective of possessor/owned perspectives is similar to time domain analysis vs frequency domain analysis or the dual nature of light as particle or light as wave. They are different bases of analysis for the same thing. Results of correct analysis in each domain are equally valid and incorrect analysis in each domain are equally invalid.

There is ample historic precedence for our approach. In the 13th century Ibn-Sina «بو علی سینا» produced "Danesh-namh Alaei" «دانشنامه علایی», [44], in which he classified his world. Based on those classifications, he then used logic to conclude. Ibn-Sina's work became a basis for much of the Western scholarly beginnings. In a sense, what we are doing here is extension of that type of classification and logic for the digital era.

2.1 Polyexistentials Reference Model

In this chapter we begin to develop a reference model. Let us call it the "Polyexistentials Reference Model." Our goal is to introduce a set of concepts and a terminology that can then be used to evaluate merits of Western Intellectual Property laws and to assist us to understand the proper governance model that is needed for polyexistentials. The polyexistentials reference model is independent and outside of the Western IP traditions. This

reference model is based on nature. It reflects science, not beliefs, faith and opinions. This model is independent of societal consensus and is equally valid in the East and the West.

We then put the Western IPR model against the polyexistentials reference model and see that the two are in conflict. When nature and man-made conventions conflict, it is the man-made conventions that are wrong. The polyexistentials reference model permits us to **prove** that Western copyright and patent laws are invalid as any form of property. Such proof is then no longer subject to any dispute because it is rooted in nature and logic – not beliefs and opinions.

We then conclude that the Western IPR model is erroneous. Based on that, we advocate that the Western IPR model should be abolished.

Such analysis needs to start with a clear categorization of monoexistentials, polyexistentials and mixed-existentials.

2.2 Monoexistence, Polyexistence and Mixed-Existence

Examples of monoexistentials are:

Material Monoexistentials: (things, spoon, touchables)

Non-Material Monoexistentials: (spectrum, internet domain name, view)

Rivalry Monoexistentials: [economic term] (Rival Goods: spoon, spectrum)

Non-Rivalry Monoexistentials: [economic term] (Non-Rival Goods: air, fish in the ocean, view) – non-Rivalry goods are often confused with polyexistentials – (e.g., Wikipedia and Jewish analysis have made that mistake).

Public Monoexistentials: [economic term] (Public Goods: roads, national parks)

Examples of polyexistentials are:

Pure Polyexistential: (recording/s, disclosed formula, disclosed idea, text, recipe, algorithm, knowledge)

Digital Polyexistential: (recording/s, formula, idea, text, recipe, software source, software binary)

Polyexistential Content: (mp3, book, cd, video, cookbook, software on a cd)

Polyexistential Service: (Google, By*, Facebook – Polyexistential drived service – monoexistential aspect not dominant)

Examples of mixed-existentials are:

Polyexistential Product: (tivo, viagra, sauce-bechamel, Mixed-Existentials as polyexistential drived products)

We present the concept of "Expressed Formula" as the general form of "primary polyexistential". The digital format presents a "pure polyexistential" form.

Polyexistentials and monoexistentials do mix. Sometimes the dimension of polyexistence is dominant and sometimes the dimension of monoexistence is dominant.

Much of our world is actually a mixture of monoexistentials and polyexistentials – mixed-existentials. In the case of mixed-existentials, the dominant aspect of polyexistence or monoexistence is sometimes clear. In such instances, we will refer to the mixed-existentials based on its dominant aspect.

Consider a book. A traditional book is mixed-existential. The paper and the ink are monoexistentials. But the content of the book (its information) is polyexistential. In the case of a book, clearly the dominant aspect is usually (not always) polyexistential. When you read a book, you are reading its content. A book can easily be

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

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

2.3 Monoexistentials

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

2.3.1 Categories of Monoexistentials

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

We categorize monoexistentials in the following 4 categories.

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

In the following sections we describe each of these.

2.3.1.1 Nature's Material Monoexistentials

Anything material is monoexistential.

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

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

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

2.3.1.1.1 Chemical Elements

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

Our understanding of the periodic table itself is a polyexistential.

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

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

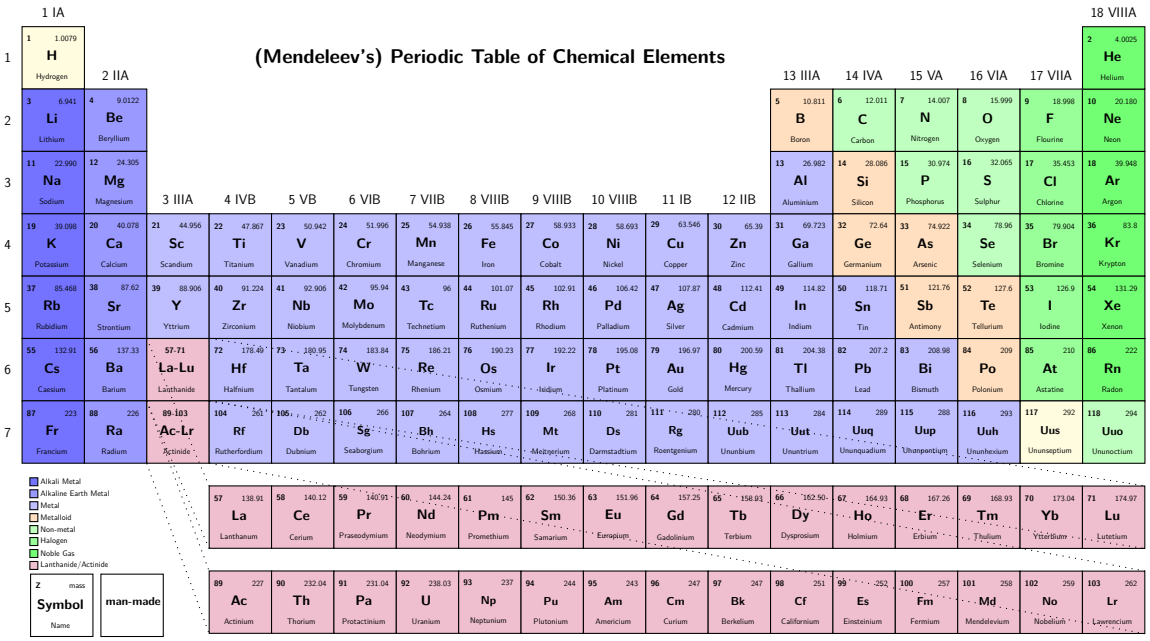

Figure 2.1: Periodic Table of Chemical Elements

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

2.3.1.1.2 Chemical Compounds

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

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

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

2.3.1.2 Man-Made Material Monoexistentials

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

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

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

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

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

Chapter 3

Digital Polyexistence

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

3.1 Digital: A Practical Pure Form of Polyexistentials

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

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

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

3.2 Basic Physical Laws of the Digital World

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

3.2.1 Digitization – Perfect Polyexistential Reconstruction

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

Such transformation involves sampling.

Sampling theorem says:

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

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

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

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

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

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

3.2.2 Encoding of Information Content

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

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

With it came the ideas of:

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

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

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

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

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

3.2.3 Transmition and Remote Copying of Digitals

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

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

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

13.4 Economics Is Not Science

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

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

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

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

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

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

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

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

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

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

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

13.5.1 The Proprietary and For-Profit Quadrant

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

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

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

13.5.2 The Non-Proprietary and Non-Profit Quadrant

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

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

13.5.3 The Proprietary and Non-Profit Quadrant

In theory this quadrant should be empty.

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

In practice, it is where most academics exist.

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

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

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

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

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

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

13.6 Libre-Halaal Polyexistential Capitalism

Thus far, we have emphasized the following key points:

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

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

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

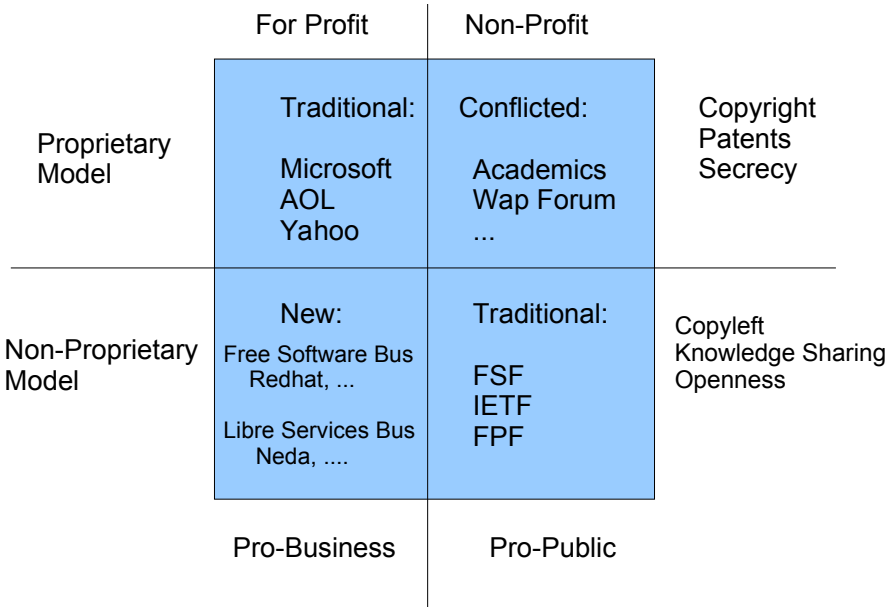

Figure 13.1: Operation in the For-Profit and Non-Proprietary Quadrant

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.

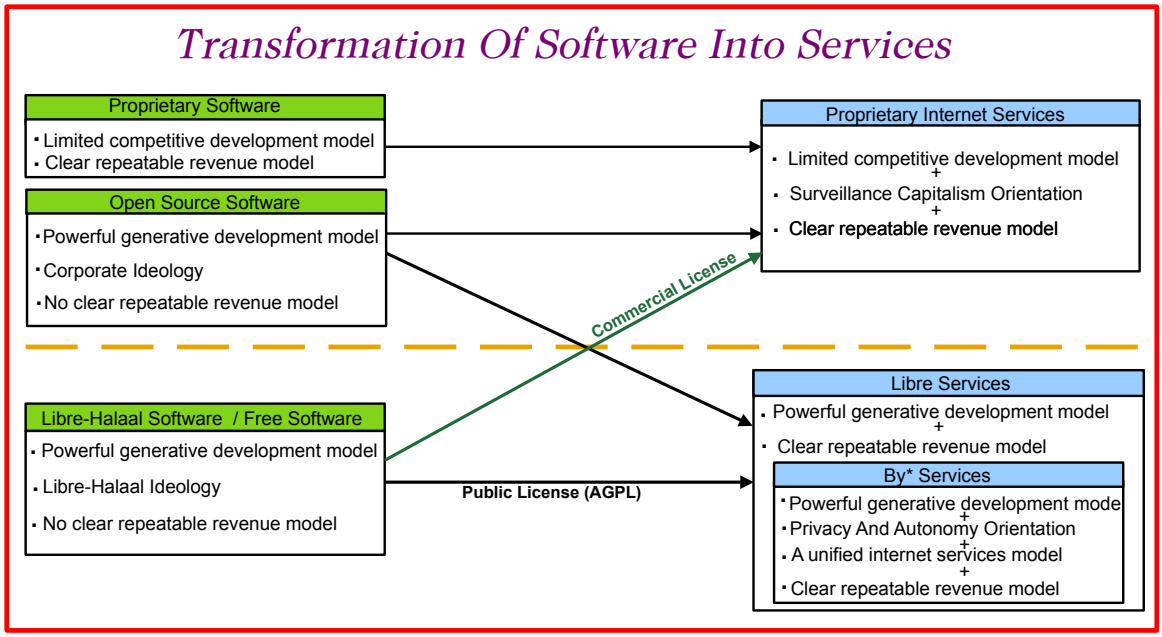

Figure 13.2: Business Ramifications of Software to Service Transformation

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.

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.

Possession-Assertable Libre-Halaal Services are used to represent real individuals and their belongings in the real world.

PALS are Libre Services that belong to their users. The owner-user should be able to self-host the service if she wishes to. A PALS can also be externally-hosted, in which case the PALS service provider must conform to a set of operational constraints. The PALS model provides for portability and transferability of Libre-Halaal Services between network abodes.

When externally-hosted, the PALS service provider must conform to the following rules. First, it must run the PALS without any modifications – the software of PALS must be known to its owner-user. Second, upon demand by the owner-user, the PALS and all user data should be transfered to the owner-user. Third, after delivering the services and the data to its owner-user, PALS operator must delete all of the user data from all of its resources and retain none of the owner-user's data.

15.3 Problem: Individual's Autonomy and Privacy are Being Crushed

Today, the internet services industry is almost entirely owned and controlled by proprietary commercial interests. Google, Yahoo, MSN, LinkedIn, YouTube, Facebook, Apple, and virtually every other Internet service—these are all proprietary for-profit corporations, with no obligation towards the public welfare.

This represents a grave hazard to the broader interests of society. The existing proprietary digital ecosystem is well on its way towards the destruction of humanity. Under immediate threat of destruction are the privacy and autonomy of the individual.

Loss of autonomy and privacy are symptoms of the basic model of the Proprietary American Digital Ecosystem. At a societal level, autonomy and privacy cannot be preserved just with new technology. There are no band-aid technical solutions.

The basic model of the Proprietary American Digital Ecosystem is all wrong.

There is already the beginning of a dawning realization within society of the growing danger to the individual's rights and freedoms.

Various attempts at blowing the whistle are made by some, but these are often crude and lack an understanding of the root of the problem.

15.3.1 Early Shallow Recognitions of the Problem

Some such superficial expressions of the problem include:

Julian Assange (in 2012) puts it like this:

> The world is not sliding, but galloping into a new transnational dystopia. This development has not been properly recognized outside of national security circles. It has been hidden by secrecy, complexity and scale. The Internet, our greatest tool of emancipation, has been transformed into the most dangerous facilitator of totalitarianism we have ever seen. The Internet is a threat to human civilization.

Eben Moglen (2011) says:

> Zuckerberg has done more harm to the human race than anyone else his age.

Moglen (2011) also says:

> Facebook is Wrong. It should not be allowed. You technologists should fix this.

Scott McNealy is quoted (1999) as saying:

> You have zero privacy anyway. Get over it.

Tim Berners-Lee (2014) says:[50]

> We need to re-decentralise the Web.

Tim Berners-Lee (2014) is even willing to think of responsibilities of the "geek community as a whole" — but as a Westerner, he is apparently unable to recognize Software Engineering as a global profession with global responsibilities. So, he says:

> It's important to have the geek community as a whole think about its responsibility and what it can do.

Donald Knuth says:

> A mathematical formula should never be "owned" by anybody! Mathematics belong to God.

Steve Jobs has said something along the lines of:

> If you're not paying for something, you're not the customer; you're the product being sold.

Even the British Sir Elton John, who has made his fortunes from copyright restrictions, now kind of gets it. When it comes to pharmaceutical companies profiting from the miseries of the sick through patent restrictions, Elton John says:

> We must end the greed of these corporations.

Edward Snowden (2013) says:

> "if a surveillance program produces information of value, it legitimizes it. … In one step, we've managed to justify the operation of the Panopticon."

The Panopticon is an architectural concept for a prison where the guards can watch, unseen by the inmates, from a tower in the middle into all cells build in a circle around the tower. It leaves the inmates in a perceived state of permanent surveillance. The French philosopher Michel Foucault described the effect:

> Hence the major effect of the Panopticon: to induce in the inmate a state of conscious and permanent visibility that assures the automatic functioning of power. So to arrange things that the surveillance is permanent in its effects, even if it is discontinuous in its action; that the perfection of power should tend to render its actual exercise unnecessary; that this architectural apparatus should be a machine for creating and sustaining a power relation independent of the person who exercises it; in short, that the inmates should be caught up in a power situation of which they are themselves the bearers.

The original Panopticon, like the digital versions the likes of NSA and Microsoft are building, takes away all feeling of privacy. Even when one is not watched, knowing that the possibility of being watched is always there, creates uncertainty and leads to self disciplining and self censorship. It is certainly a state the powers that be would like everyone, except themselves, to be in.

To call these signs of deterioration of humanity is an understatement.

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

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

16.2.3.4 Confidentiality, Anonymity and Privacy

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

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

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

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

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

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

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

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

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

16.2.4 ByStar Central

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

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

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

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

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

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

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

16.2.4.3 LibreCenter.net

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

16.2.4.4 BySource.org

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

16.2.4.5 ByBinary.org

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

16.2.4.6 ByStar Name and Number Assignment Authority

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

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

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

16.2.5 Current ByStar Services and Capabilities

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

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

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

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

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

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

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

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

16.2.5.1 Current Capabilities of ByStarEntity

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

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

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

16.2.5.2 Current ByStar Services Sites

Current ByStar services sites are depicted in Figure 16.1.

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

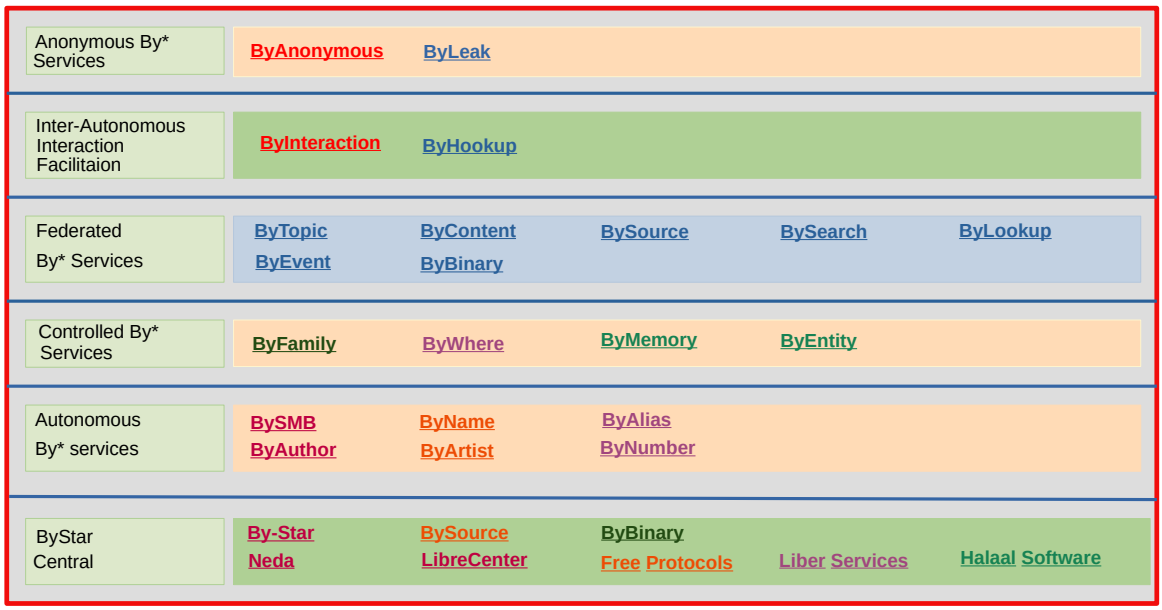

Figure 16.1: ByStar Domains Table

16.2.5.3 Current Status and Span of ByStarEntity Generators

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

16.2.5.4 Current Status and Scope of ByStar Federated Services

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

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

16.2.5.5 Growth of user base: timing

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

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

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

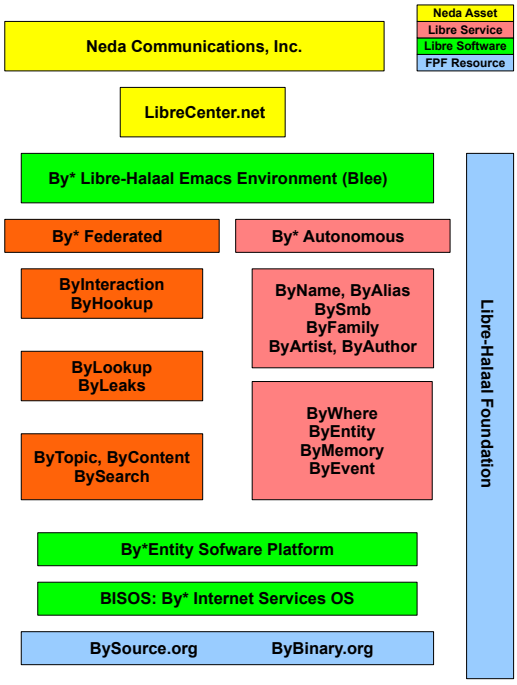

Figure 16.2: Libre Services Supporting Organizations

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

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

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

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

16.2.6 Relationship With Existing Realities

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

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

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

16.2.6.1 Relationship With the Proprietary American Digital Ecosystem

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

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

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

In a document titled:

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

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

16.2.6.2 Relationship With FOSS / FLOSS Movements

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

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

16.2.6.3 Active Private Parallel Digital Ecosystems – Example: NSA

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

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

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

16.2.6.4 Relationship With Piecemeal Privacy Oriented Software and Services

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

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

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

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

16.2.7 ByStar Economics

Having introduced the Libre-Halaal Bystar Digital Ecosystem in philosophical, moral, societal and engineering terms, we now turn our attention to the economic and business dimensions.

We are devout monoexistential bounded-corporations capitalists. We believe in proper ownership rules, free markets and proper regulation. We are pro-business.

The existing capitalist model for monoexistentials is generally correct, in both philosophical and economic terms. But the extension of the monoexistential capitalist model into the domain of polyexistentials, based on the Western IPR regime, is a grave mistake. Philosophically it is wrong. Societally it is harmful to humanity. And economically it is unstable and vulnerable, since it can be displaced by disruptive business models like ours. The ByStar Open Business Plan explains how this will come about, and how we will profit from this.

We expand on this in Chapter 13 — Global Polyexistential Capitalism.

16.2.7.1 Revenue model for Libre-Halaal Software

The Proprietary-Haraam software model, operating under Western copyright restrictions, includes a highly effective recurring revenue generation model: the proprietary software licensing model.

But the Halaal manner-of-existence of software eliminates all restrictions on the distribution and use of software. Thus, the Proprietary-Haraam recurring revenue model is also largely eliminated. Recurring revenues under the Libre-Halaal software model are much less than under the Haraam software model.

16.2.7.2 Revenue model for Libre-Halaal Internet Services

The Halaal manner-of-existence of software creates a powerful generative development model for Libre-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. Therefore in terms of revenue generation, Libre-Halaal and Proprietary-Haraam services are on an equal footing.

16.2.7.3 ByStar Value Chain Analysis

ByStar value chain is a chain of activities that we perform in order to deliver valuable internet services to the market. It is a high-level model of how we take raw externally developed Libre-Halaal software as input, add value to these software packages through various processes, and sell finished services to our customers.

In Figure 16.3, we illustrate the ByStar value chain on the left column and its inter-mixing with proprietary value chains on the right column.

Focusing on the right column of Figure 16.3, notice that "Neda Operated By* Services" establish a direct relationship with Subscribers and Users at the very top. Note that the scope of these Internet services is everything – the * in By* – and that the intended scale of these services is planet-wide. By definition, no Internet services opportunity can be bigger than that.

The arrows between Neda Services and User/Subscriber in Figure 16.3 include an element of "Trust, Loyalty, and Respect" which is the result of "ByStar Ideology" that we presented earlier. The element of trust and respect is fully absent in the left column. In business terms, Trust and Respect, translate into "stickiness" – 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

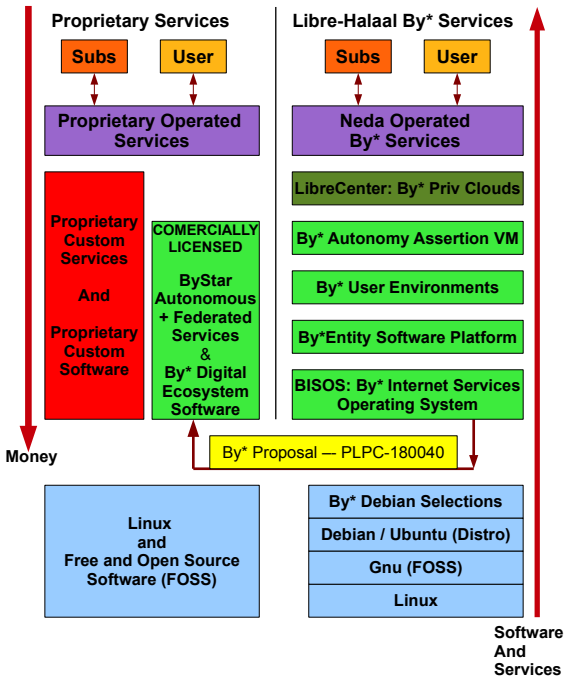

Figure 16.3: ByStar Value Chain

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.

Chapter 17

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

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

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

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

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

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

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

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

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

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

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

The solution to these limitations consists of three main components:

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

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

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

17.1 Concept of the Universal Internet Services OS

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

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

17.1.1 The Early LAMP Precedence

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

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

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

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

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

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

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

Such an extension involves two dimensions:

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

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

17.1.2 Operating System, Internet Application Service and Digital Ecosystem

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

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

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

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

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

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

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

17.2 Overview of BISOS and ByStar Digital Ecosystem

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

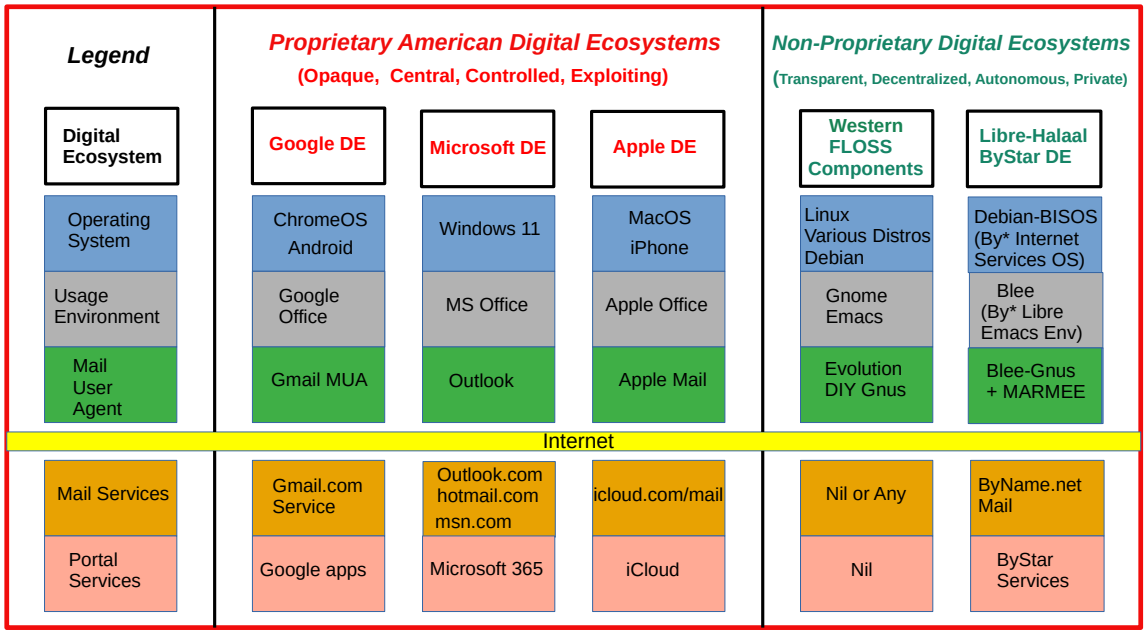

Figure 17.1: ByStar and Proprietary American Digital Ecosystems

BISOS has the following key characteristics.

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

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

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

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

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

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

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

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

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

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

17.4 BISOS: an Over Debian Pure Blend

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

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

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

17.5 BISOS's Basic Unit: BISOS-Site

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

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

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

17.6 BISOS Portable Objects (BPO)

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

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

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

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

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

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

17.7 BISOS Possession Assertable Libre Services (PALS)

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

BISOS Libre Services can be thought of four parts:

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

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

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

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

17.7.1 Network Abodes and Transferability

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

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

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

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

1. Exposed Rim.

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

2. Inner Rim.

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

3. Outer Rim.

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

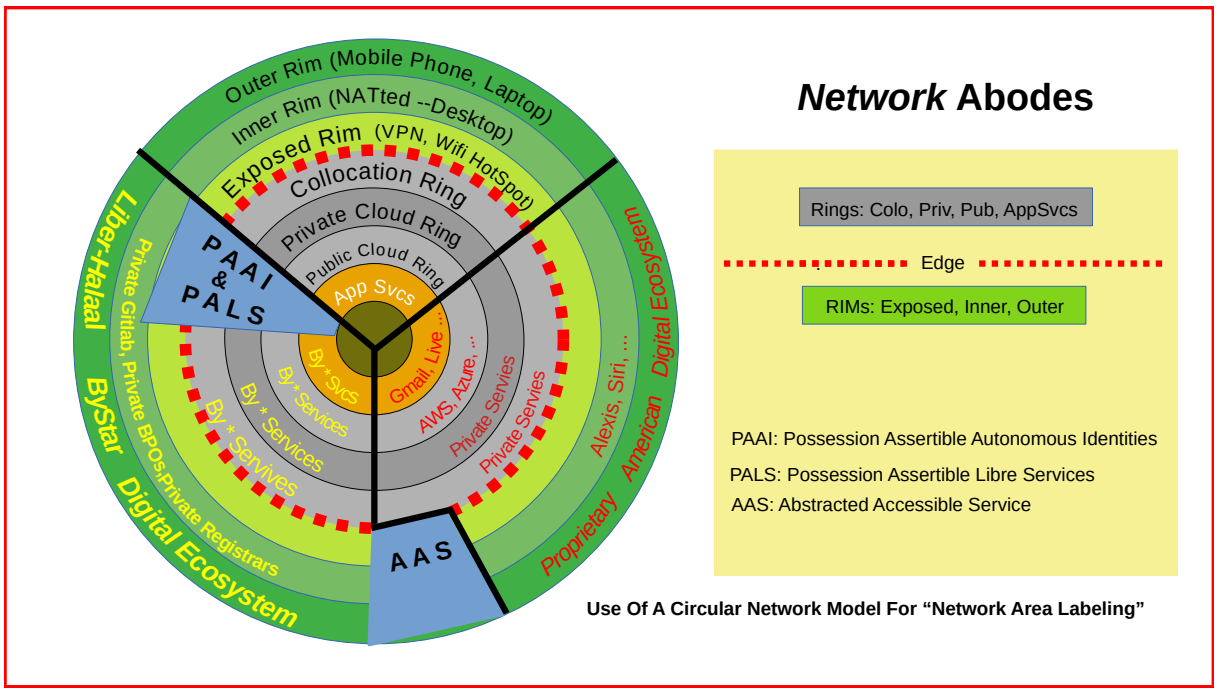

Figure 17.4: Network Abodes: A Circular Model For Network Area Labeling

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

1. Collocation Ring.

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

2. Private Cloud Ring.

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

3. Public Cloud Ring.

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

4. Public Internet Application Services.

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

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

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

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

17.7.2 Ramifications of Libre-Halaal Edge-Oriented Strategies

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

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

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

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

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

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

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

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

All ByStar individual services are designed to be end-to-end and edge-oriented. The concepts of end-to-end and edge-orientation are integral to ByStar's decentralized design, which stands in stark contrast to Gmail's highly centralized approach. However, these edge-oriented services don't need to reside on the "Rims" side of the network edge. Since ByStar individual services are possession-assertable and portable, they can also be provisioned in the "Rings". See Figure 17.4 for the references to Edge, Rims and Rings. This provides for options of self-hosting or external-hosting of individual services. So, `byname.net` can be made to be as convenient as `gmail.com` yet preserves the guarantees of autonomy and privacy through being possession-assertable, portable, Libre-Halaal, and edge-oriented.

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

17.8 BISOS Model of Platform Universality and Software-Service Continuums

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

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

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

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

17.9 PyCS: BISOS's Integration Framework

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

There are three fundamental important choices to be made:

1. What programming language should we use for integration?
2. What command-line framework should we use?

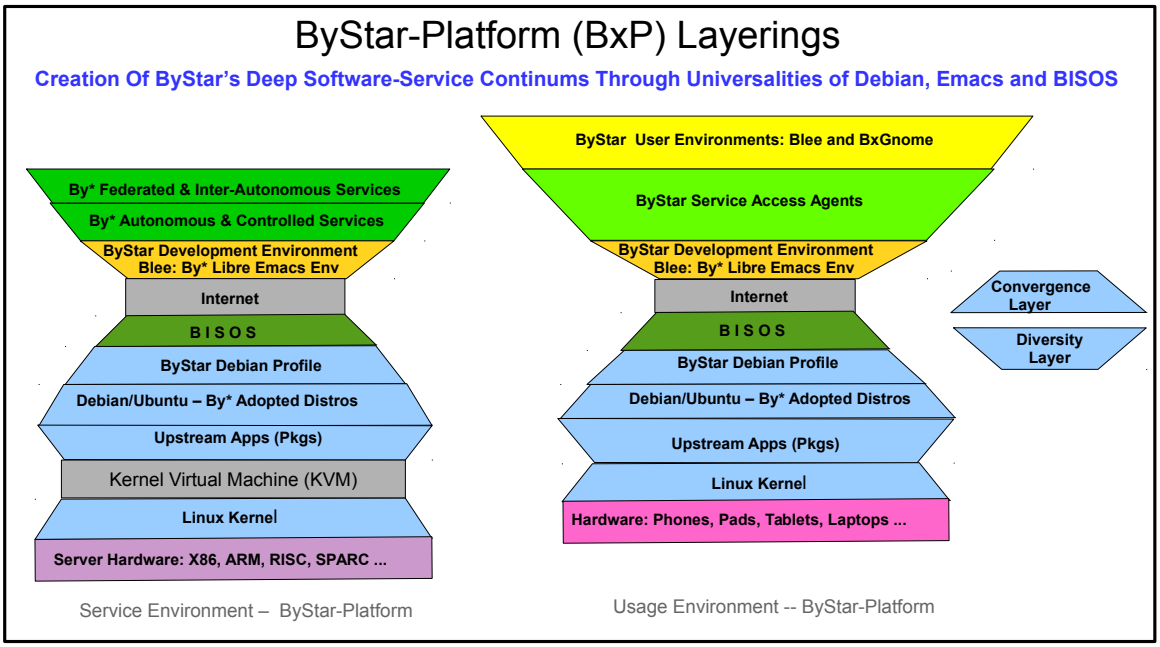

Figure 17.5: ByStar Platform Layerings and Software-Service Continuums

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.
- 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.

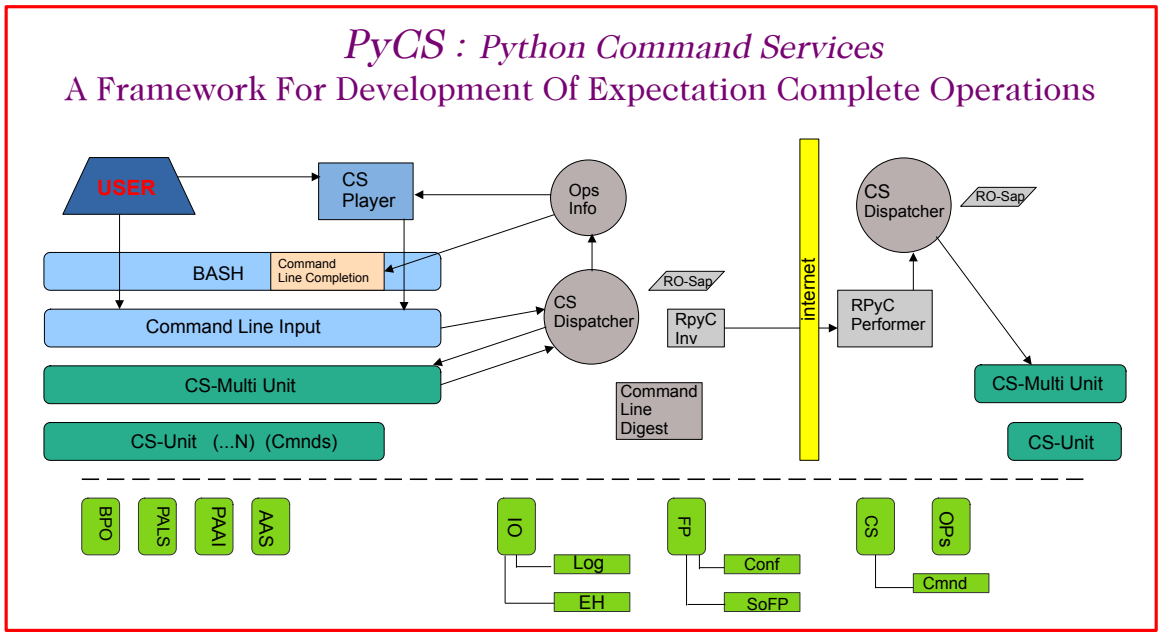

Figure 17.6: BPyF (BISOS Python Platform) and PyCS

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

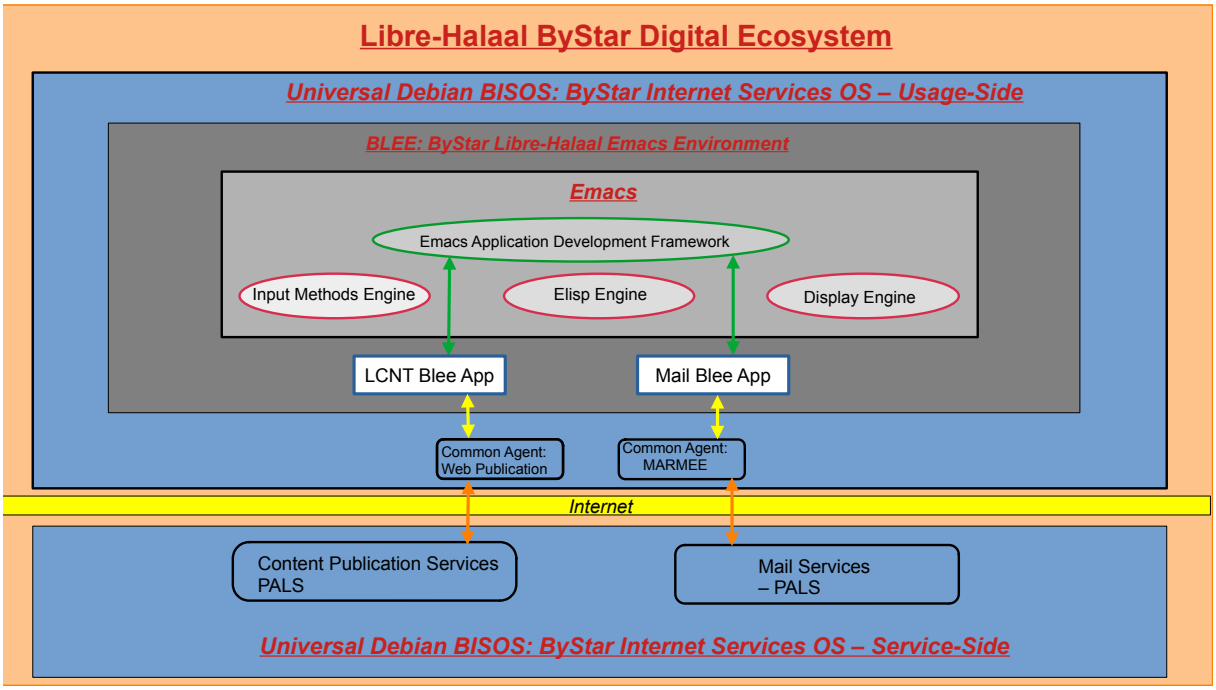

Figure 17.7: A Blee Centric Perspective Of By* Digital Ecosystem

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.

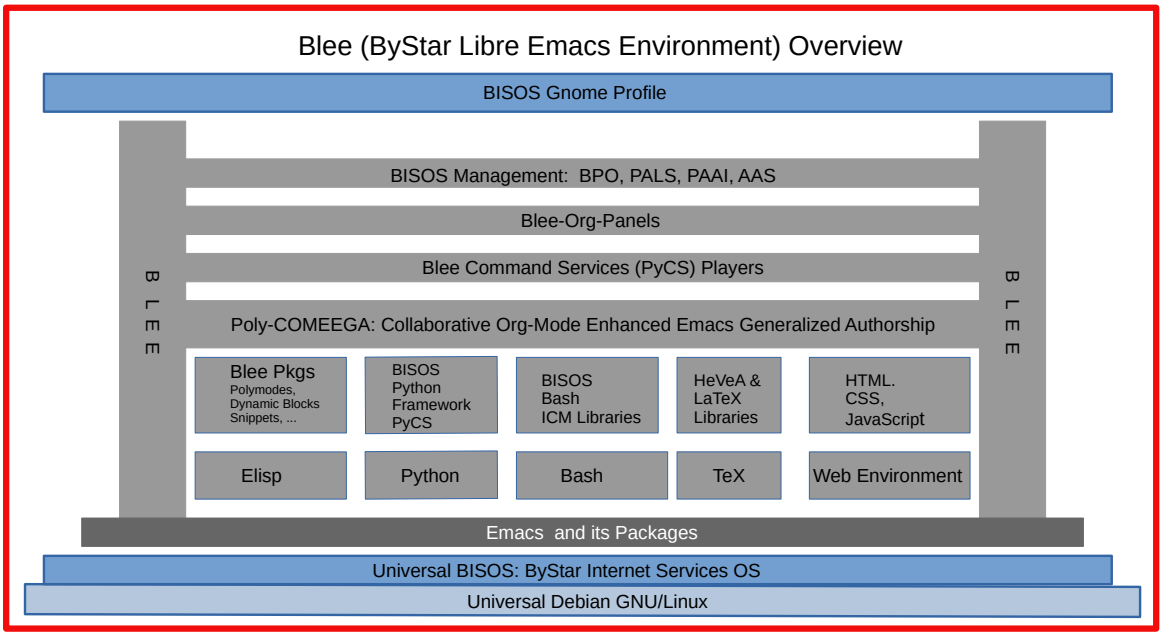

Figure 17.8: Overview of Blee Features

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 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 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.

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.