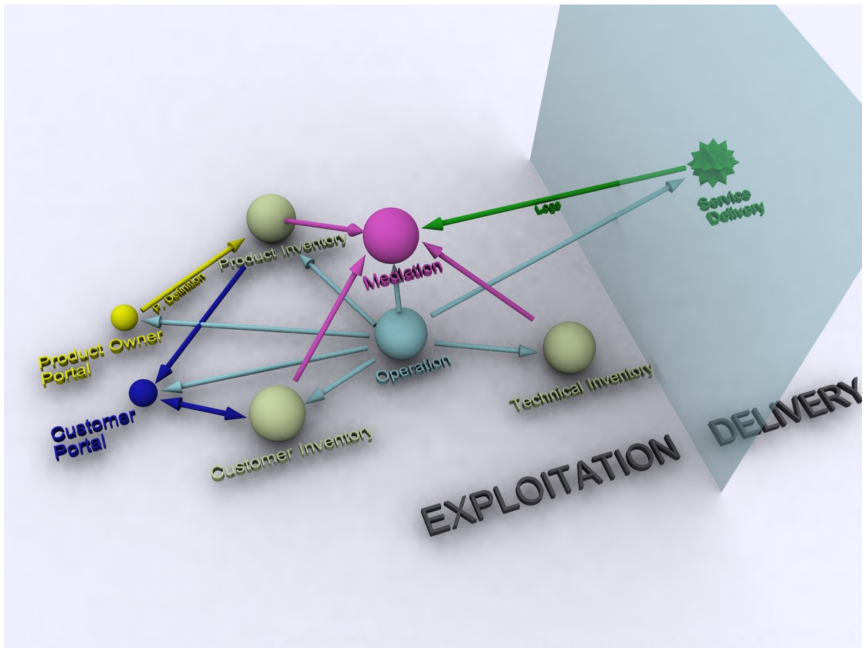
Modern exploitation of Content Businesses

adolfomaria.rosasgomez@gmail.com www.adolforosas.com

October 2014



1 Introduction

Today some say 'content is king' … some others say 'network is king', or 'user is king', or 'data is king' or … whatever thing is king. There are so many 'kings' today in the markets and industry. But you have to concede that content services flourish today, as of mid-2014, in the network. We have more online video than ever before, more shared media: photos, videos, messages (sometimes with embedded media), downloadable apps, streaming games…and more video streaming services than anyone could have imagined just a few years ago.

If we admit that the internet economy is going up, and at the same time we admit that content services are an important part of that economy, it would be reasonable to assume that people have discovered how to exploit content services. How to build services that are not only convenient for consumers: easy to use, appealing, affordable, modern, engaging ….but also convenient for the ones behind the service, convenient for exploitation: easy to run, understandable, reliable, profitable.

To a certain extent it is true that some people learnt to build those services and made them as good as it is conceivable, but only to a certain extent. The fact is you can name a few (less than 5) content services that surpass all others, you can name even less leading search engines, you can name very few leading streaming services, you can name essentially a couple leading movie services, very few first-line music services, virtually only one leading e-book service… The very existence of this clear category of 'leading content services' lets us know that not all the people building services knew how to make them as good as it is possible.

I want to devote this article to the art and science of making content services 'exploitable' in modern terms.

2 Service ideation

Many managers I've known just cannot imagine the 'whole coherent idea' of a new content service. They cannot create a mind representation of a coherent set of actions, definitions, processes that involve all actors (producer, aggregator, sellers, end-users,…), all infrastructure (production stages, repositories, distribution infrastructure, catalogues, front ends, payment systems,…), and all processes(content ingestion, metadata generation, catalogue and frontend provisioning, transaction processing, billing, analytics,…)

Content Business Managers 'not always' understand how involved the 'service idea' or 'service model' is with technology. Some of these managers will let 'technical people' alone build the managers 'current view' of the service. Usually in these cases that 'view' is a short list of requirements that describes some concrete aspects about what the user should see, what should happen when the user does this and that, and what options should 'the system' (doesn't it make you smile to realize how many people refer to almost any part of the service as: 'the system…') provide to end users and administrators.

But please do not take me wrong on this; I would not say that 'technical people' can do it better. In fact most 'technical people' I've known would have even a shorter, less-coherent view of the whole service. The problem is many managers have a tendency to imagine the service only as a money-maker forgetting completely about how it works and many technicians have a tendency to imagine only isolated parts of the service at work and they can't and won't imagine how the service makes money.

What happens usually?

Working on a partial and incoherent view of the service any implementation of that 'view' will be flawed, impossible to evolve once we start asking ourselves very basic questions that test the coherence of the service model. If a technician raises one of these questions, many times he will need the manager to give an answer, to think of choices he never imagined. If the manager was the one to discover something he does not like in the behavior of the service, many times he will need assessment from the technician, making the technician think of options he never considered. It is only a matter of time (very short time) that other managers or even the same manager will need to do very simple variations to the original idea: instead of single items sell bundles, instead of fixed prices do promotions, instead of billing per total traffic by end of month bill per percentile 95/5 min of instant available bitrate, instead of ….

In many cases the first implementation of a content service must be thrown to litter and start all over again just after the first round of 'improvement ideas'. It happens that those 'clever requirements' that we had at the start were mutually conflicting, short-sighted, and did not allow for any flexibility even the most basic.

So, what to do to avoid recreating all the service from scratch every month?

The approach that I would suggest is to do a lot of planning before actually building anything.

Write your ideas down. Draw flow diagrams on paper. Draw mockups on paper. Put them through 'logic tests'. Ask yourself questions in the style 'what if…?' , or 'how can the user do…?', or would it be possible later to enhance/add/change…?' Show your work to others. Rethink. Rewrite. Retest….

Spend a few days or better a few weeks doing this and I can assure you that your requirements list will grow, your idea about complexity of the service will change, your enthusiasm will increase substantially, your understanding of possible problems will be much better, and the time and effort to build and maintain your service will be sensibly reduced.

It is important that in the ideation process different people work together: marketing & pricing experts, engineering & operations experts, end user experience experts, billing, analytics, support… I would also say: please do not use computers for work in this phase. PowerPoint is great… but your hands with a pencil are far better and much faster. And in a room with no computers there is no possibility for someone to go and check email and thus be out of the group for a while. (Smartphones are computers too and should be prohibited in this phase.) Recall that: 'if you cannot imagine it you cannot draw it. If you cannot draw it with a pencil you cannot draw it in PowerPoint'. If your boxes are not perfectly rectangular and your circles are not perfectly round you can rest assured that at some point later a good computer will fix that with no pain.

If you were not forced to imagine the internals of the service at work you would not find the problems you have to solve and you would never find the solutions. You must realize that everyone that has a voice to claim requirements over the service MUST imagine the service at the SAME time that others do it and SHARE immediately his ideas and concerns with the entire group.

3 Building coherency: a semantic model

Oops…I couldn't avoid the word 'semantic' I've immediately lost half of my readers. I'm sorry. For the brave remaining readers: Yes, it is an academic style of saying that the whole service must make sense and not contradict itself neither in 'concept' nor in 'implementation'.

I've noticed that some people, including some of my colleagues, start to mentally wander when we speak about modelling. But **it is important to create models of services**. Services do not grow on trees. A service is not a natural thing, it is a contraption of the human mind and it must be polished, rounded and perfected until it can be communicated to other minds.

A 'model' is a simplified version of what the service will be once built. **Our model can start as an entity that only lives in paper**. A few connected boxes with inputs and outputs and processes inside may be a perfectly good model. Of course the model starts to get complicated as we imagine more and more functions and capabilities that will be in the service. At some point our model will need to live out of paper, just because paper cannot support the complexity of the model… but recall that the model is always a simplification of the real service, so if you think your model is complicated, what about your not-yet-existent service? Do you really want to start spending money in something you do not really understand?

Have you noticed that you need to **establish a common-language** (also known as 'nomenclature') to communicate 'about' the service inside the service team? Let me show you an example: what happens when you tell another member of the team: "…then the user will be able to undo (some operation)…" But, who is 'the user'? For the designer of the consumer interface it is unequivocally 'the end user consuming the service', for the OSS/support/operations guys it is unequivocally 'the operator of the support interfaces', and for other people 'the user' may even be other kind of human. You probably think I'm exaggerating… you think this is not a problem as every expert has a good understanding of 'the user' in his own context. But what happens if the end-user-at-home must be mixed in the same sentence with the user-of-the-admin interfaces? Let me tell you what happens: each one dealing with that sentence will express it in the way that is more convenient to him. The sentence will be expressed in terms of the mind that is in charge of that document/chapter/paragraph… Names will be applied to distinguish between roles, but these names belong to the experience of the one that is writing the text and they do not always fit in the ideas of other people in the team. These names have never been agreed A-Priori. And worse, in other sentences that also mix the two concepts another different guy can name the same roles completely differently. This is one of the reasons why some technical documents about a service are impossible to read by anyone that just has common sense but lacks 1500 hours of meetings with all the internal stakeholders in the service.

I cannot be sure that I convinced you of the need to stablish a 'common language' about the service, but if you have had to coordinate a documentation department or a big program management office you know what I'm saying. In fact **the problem goes much farther than 'names'. The problem extends to 'concepts'**.

I would never have thought at the start of my career that people could develop so very different ideas of what is 'billing', what is 'support', what is 'reporting', what is 'consuming a content', what is 'consumer behavior', and a few other hundreds of concepts… But over time I learnt. These, and **essentially all concepts that we deal with in our human language do not have the same representation in each other's mind**. That does not usually create problems in our daily life but when we design something very detailed, if we want others to understand it at first attempt we need to agree in the language and concepts that we deal with. We need to start with a model of the service that is semantically correct, and never use terms strange to this model to communicate about the service. The paradigm of the correctness in semantic models is called an 'ontology' (oops bye, bye another half of readers). Ontologies are very hard to 'close' as coherency must be complete but we may do reasonably well with a simpler semantic model that contains **'primitive concepts**' (definitions of objects and actions that will not be questioned), semantic relations between primitive concepts and '**derivative concepts'** which are objects and actions defined in terms of the primitive concepts.

4 Content Service primitive concepts

The word 'service' taken as a 'semantic field' inside human experience can extend to touch many concepts. If we shrink our area of interest just to Telecommunication services the possible interpretations of service are much less, but if we go further and define our domain as 'Telecommunication Content Services', then the number of possible concepts that we will touch and the language appropriate to deal with them gets much more 'manageable'.

'Exploitation model' for a content service: simple semantic model that contains all definitions of entities and actions relevant to describe 'exploitation' of a content service.

We define 'exploitation' as: all actions that pursue the creation and maintenance of value/utility for consumers and all accompanying actions that ensure that service stakeholders obtain 'something' in return.

'Exploitation system' for a content service: information system built to implement the exploitation model of a content service.

The following could be a very simple **Exploitation Model** for Content Services:

- **Content or Content Product**: a piece of information valuable to humans that thus can be sold.

- **Content Service**: a continued action that provides value through Content Products. A service can be sold as a product, but the service is action(s) while Content Product is information. A concrete service is defined by specifying the actions allowed to the consumer (see Capabilities below).

- **Consumer or Customer**: Human that receives value from a product or service. (P.S.: a subtle remark: Consumer gets utility and Customer pays for it. Sometimes they are not the same person.)

- **Service Capabilities**: actions available through a service.

- **Contract**: statement of commitment that links consumer identity + product/service + pricing + SLA

- **Bundle** (of products/services): group of products/services gathered through some criteria. The criteria used to bundle may vary: joint packaging, joint charging, joint promotion … but these criteria define the bundle as much as its components and thus criteria + components must be stated explicitly.

- (one possible) **List of Content Products** that pretends to define a CDN-based Service:

Streaming VoD (Movies…): content pre-stored and consumed on demand as a stream.

Streaming Live (channels, events…): content consumed live, as the event is captured, as a stream.

Non-streaming Big Object (SW, docs…): pre-stored huge content consumed on demand as a block.

Non-streaming Small Object (web objects…): pre-stored tiny content consumed O.D. as a block.

The following could be a possible minimum description of an **Exploitation system** for the above model:

Portals: (Human Interaction tools)

-Product owner (product definition) portal: meta-portal permits defining prod+capabilities.

-Seller portal: meta-portal permits define 'soft' prod properties: name, pricing…

-Customer portal: handles customer action: consumption and feedback.

-value added portals & tools: customer analytics, trends, help, guides…

Mediation systems

-report gathering & analytics: log gathering & processing, analytics processing

-billing & clearing: 'money analytics'

Operation systems

-customer provisioning: customer identity management

-service provisioning: contract management + resource allocation

-event (incidence) ticketing: link customer + product + ticket + alarms + SLA

-SLA monitoring: link product properties + analytics + contract -> alarms -> billing

Inventory systems

-commercial (product) inventory: database of products + capabilities (properties)

-resources (technical) inventory: database of infrastructure items (OSS)

-customer inventory: (protected) database of identity records

5 Human interactions: Portals

This is usually the first (and sadly sometimes apparently the only) part of service that a company works on.

It is not bad to devote time to design great interaction with your consumers and operators, but you must go beyond nice interfaces. In fact a streamlined interface that does just what is needed and nothing else (extra functionality is not a gift, it is just distracting) is not so easy to build.

As I mentioned before it is important to have 'perfected' the idea about the service. Once that idea is clear and once you have written down your objectives: what 'powers' (capabilities) you want to put in the hands of your consumers and what would it take to implement those capabilities in the unavoidable explosion of support systems, operation portals, processes,… it is time to go and draw a few mockups of the interfaces.

It is not easy to build simple interfaces. The simpler the look & feel the more clever you have to be to handle correctly the interaction. It is especially difficult to get rid of 'bad habits' and 'legacies' in interface design. You must consider that today the interface that you design will be 'interpreted' by devices that have new and appealing possibilities: HD displays with video even in small mobile devices, multi-touch displays, high quality sound … and best of all: always-on connection and social interaction with other 'places/sites/services' . This interaction must be 'facilitated' by making a natural 'shift' from your interface to any other site/interface that you share your consumer identity with at the click (or swipe) of one widget.

5.1 The customer portal

You need interaction with your consumers/customers. They need a place to contract with you (in case you support 'online contract generation' which by the way is the trend), and a place to consume your product that as far as this article is concerned is 'content', that is: information intended for humans.

This portal is the place to let consumers browse products, get info about them, get them… and this is also the place to let them know transparently everything that links them to you: their purchases (contracts), their wishes, their issues/complaints, their payment info, and their approved social-links to other sites…

What makes you a provider of preference to a consumer is: **trust**, **ease of use** and **clarity**.

Through the customer portal the Consumer registers his identity with the service (maybe totally automated or aided by Operations), purchases products, receives billing information and places complaints. We'll see later that there are a number of entities related to these actions: Customer Inventory, Product Inventory, Mediation, Event Ticketing, Operations,…

5.2 The seller (internal) portal

You may run a small company and in that case you run directly your business through a relatively small portal in which you do everything: advertising, contracting, delivery, feedback, ticketing…

Or you may be a huge company with commercial branches in tens of countries, with local pricing in different currencies, portals in many languages and a localized portfolio in every region.

In both cases it is useful to keep a clear distinction between ‘hard’ and ‘soft’ properties of your product.

‘Hard properties’ of your (content) product/service are those properties that make your product valuable to customers: the **content itself** (movie, channel, episodes, events…), the **ease of use** (view in a click, purchase in a click…), the **quality** (high bandwidth, quality encoding, and flexibility of formats…), the **responsiveness** of your service (good personalized attention, quick response times, knowledgeable staff…), etc.

‘Soft properties’ of your (content) product/service are those properties that you have added to make exploitation possible but that are not critical to your consumers : the ‘**names’** that you use to sell (names of your options, packages, bundles, promotions, IDs, SKUs,…), the **prices** and **price models** (per GByte, per movie, per Mbps, per event, per bundle, per channel, per promotion…), the **ads** you use to promote (ads targeting by population segment, by language, by region,…), the social **links** and commercial **alliances** you build, the **themes** and **colors**, the time-windows of pricing, ….

The best way to materialize the distinction between ‘hard’ and ‘soft’ properties of a product/service is to keep two distinct portals (and all their associated backend) for ‘product owner’ and for ‘product seller’.

In the **‘product owner’ portal** you will manage hard properties .

In the **‘product seller’ portal** you will manage soft properties.

The customer portals are built ON the seller portals. That means that you at least have as many customer portals as ‘sellers’ in your organization. If you have branches in 30 countries and each of them has autonomy to localize the portfolio, pricing, ads, names, etc… each branch is a seller. Each branch needs an internal portal to build its entire commercial portfolio adding all the soft properties to a very basic set of common hard properties taken from the product owner internal portal (see below). Each branch (seller) will build one or more customer portals depending upon their internal seller portal.

You can even be a ‘relatively small’ company that licenses products/tools to resellers. In that case you provide an internal reseller portal to your licensees so they can sell your hard product as their own by changing names, prices, ads, links, etc…

5.3 The product owner (internal) portal

This is the sancta sanctorum of the product definition. This is the place where you define the internals of your product. This is the place where you empower your consumers by ‘creating’ fantastic actions over fantastic content pieces and putting all these actions and portfolio in the hands of your consumers.

It is VERY difficult to find a tool that could be flexible enough to take ANY piece of content and link it to ANY possible action that makes commercial sense. In fact it is impossible.

For this reason the ‘owner portal’ lives more time in the realm of developers than in the realm of administrators. (This MUST NOT be the case for the other portals: seller, customer… or you would be in serious trouble.)

What I mean is: it is impossible to design the ‘tool of tools’ that can graphically modify the actions that you make available to your consumers in every imaginable way. The new ideas that you come up with will surely require some new code and unfortunately this code will be at the heart of your exploitation systems. For this reason it is better to cleanly separate your sellers internal backend from the mother company backend and your customers portals from the sellers internal portals.

But do not despair; it is possible to implement a very serious backend for content exploitation, tremendously powerful, and a flexible tool that manages the most common hard properties of a content product /service.

The common 'product owner portal' must implement the following concepts and links:

-the **complete product list**: items not listed here are not available to sellers

-the **complete capabilities list:** actions and options not listed here are not available to sellers

-general 'hard' **restrictions**: internal SKUs, internal options (formats, quality steps, viewing options…), billing options (per item, per info unit, per bandwidth…), SLA (availability, BER, re-buffering ratio, re-buffering events/minute…)

Every Content Product must go through a cycle: **ingestion - delivery – mediation (accounting & billing)**.

The links between consumer id, contract, product id, options for consumption, options for billing, options for SLA, etc… must be implemented in several information systems: databases, registries, logs, CRMs, CDRs, LDAPs…)

From these three segments in a service life-cycle: ingestion–delivery-mediation, most academic articles on content services focus on 'delivery' as this is the aspect of the service that creates the hardest problems and thus it is fertile soil for innovation (CDNs are a great example). This article focuses in all the rest of the effort, everything that is not pure service delivery.   One important goal of this article is to demonstrate that **creating a great online service and later figuring out how to exploit it is a bad idea.**

5.4 Value added portals & tools

These tools and values added are as I've said…'added'. No one needs them to find, get, pay and enjoy a Content Product. But who 'needs' a Movie anyway? I mean 'needing' is not a good word to describe the reasons that drive Content consumption.

Many times it happens that there are actions that no one 'needs' but the value they add to a basic service provides such an attraction to consumers that the service becomes popular and the new actions become a convenience that every other competing service must implement. Examples are: search engines, comparison engines, wish lists, social appreciation lists, ranks, comments … People are buying 'objects' that no one needs to keep himself alive, there is nothing obvious per-se in a Content Product value. We must compute its value before purchase judging on other people appreciation and comments.

All the 'tools' that we can imagine that may help people understand our offer (portfolio) and navigate through it are positive to our business and should be appropriately placed in the customer portals or gathered together in a tools tab , not distracting consumers but helping them know and consume Content.

Some modern tools that help monetize Content: product search, product comparison, social wish list, price evolution over time, price comparison, related products, buyers ranks, social (our group) ranks, open comments, long term trend analysis…

6 Mediation Systems

These are key systems to exploitation but the name is really ugly. What do we mean with 'Mediation'?

We need a middle-man, an inter-mediate, a mediator, when we do not have all the capabilities required to do something or when it is convenient to delegate some task in others that will do it better or at least equally well but cheaper than us… or simply when we prefer to focus in other tasks.

In a commercial exploitation system, '**Mediation'**, usually means '**doing everything that is needed to apply and enforce contracts**'. Sounds easy, yes? OK, it isn't.

Enforcing contracts is 'Mediation' for us because we choose not to identify with all the 'boring' actions needed to apply the contract,… we prefer to identify ourselves with the actions that deliver the service, and that is human. Delivery is much more visible. Delivery usually drives much more engagement in consumers.

Mediation usually involves accounting and processing of data items to prepare billing and data analyses.

Mediation in Content Services includes:

-**log gathering & processing**

-**billing & clearing** (and sometimes payment)

-**analytics processing** and report rendering

In the CDN business Log gathering & processing is a cornerstone of the business. Many CDNs in fact offer edge logs as a byproduct and some sell them. Even in some legal frameworks especially in Europe CDN service providers are forced to keep edge logs for 12 months available to any authority that may demand them for audit.

CDNs are huge infrastructures, usually with thousands of edge machines in tens or hundreds of PoPs distributed over tens of countries. Almost 100% of CDNs bill their customers by the total number of bytes delivered over a month (GBytes/month). Only a few CDNs bill customers per percentile 95 measured in 5 min slots of delivery speed over a Month (Mbps/Month). In any case it is necessary to measure traffic at the delivery points (edge). But the edge is huge in a CDN so lots of log files will need to be moved to a central element for some processing. This processing involves separating CDRs (Customer Data Records) that belong to different customers, different Content Products, different regions, etc…etc… In case a CDN implements percentile 95/5 billing the downloads have to be processed in 5 min slots, average Mbps per slot and customer calculated, rank of slots over the whole month gathered and the percentile 95 calculated per customer.

Usually other interesting calculations are worth doing over edge logs.

We now live in the era of 'Big Data' which is a new buzzword for an activity that has been for long time present in some businesses (like CDNs) and longtime absent in some other businesses (like many online services), this activity is behavior recording (journaling) and offline analysis (trend spotting and data correlation).

Analytics and billing should be related in a CDN. As time goes by more and more data analyses become popular for CDN customers. We started with very basic billing information (traffic/month) and that is still valid but many other analyses become feasible in these days due to increased processing power and due to new and interesting propositions about data correlation.   Online content businesses have appeared over the world in a moment when other online services existed and there were established billing information systems. These billing systems for online services were mostly of two kinds: **continuous service accounting for deferred billing** (CDR based, common in Telephony), **discrete event billing** (common in online shops).

Discrete event billing is easy to understand: **one SKU is purchased –one SKU is billed**. No more time spent.

The **CDR** (Customer Data Records) are **tiny pieces of information that must be collected over a period (monthly usually) to help reconstructing the 'service usage history'**. Each CDR is as much as possible an independent piece of information intended to be later processed by a 'billing machine'. When creating the CDR we must not rely in that any context information will be later available and thus the CDR must contain everything needed to convert it in money: customer ID, service ID, units consumed, time-stamp, and other 'creative specific billing data'. The fact is that **there is always some context needed at processing time** so no CDR system is perfect, but the whole idea of keeping CDRs is to **reduce the context** that exists at the time of CDR creation and in this way we will be able to post process adding information that was not available at consumption time (in case this information ever appears).

Consolidation of CDRs is a cumbersome process that allows great flexibility in Telco billing but it does not come for free. In fact this 'flexibility' has created one of the biggest problems in data back ends: processing of CDRs usually cannot start until the billing period has ended (explanation below) and at that moment CDRs in the billing system can be thousands of millions of records. Huge datacenters have been built for billing. They are expensive, they are slow, they are complex, they are unreliable (no matter how enthusiastic the vendor is and how small he claims is the amount of 'impossible to charge for' records). Why is this? Creativity in business models for 'Telecom utilities' has been enormous in recent times, especially since the advent of mobile communications. A subscriber is charged usually at the end of a month, and in the middle he can contract, refuse, modify, consume a variety of communication products, receive promotions, discounts, obtain fidelity points, redeem points… All this complexity of actions that affect the monthly bill must be recorded, enriched with context, time-stamped, stored… and a consolidation process must be run at the end of the billing period to transform CDRs in a bill per customer. This high complexity is supported willingly by Telcos today. They seem to have a preference for creating a plethora of different promotions, personal plans, personal discounts, special discounts, special surcharges, different pricing time windows... It seems that currently this complexity is good for Telco business, but the other side of it is that you need CDR billing.

Now you should be questioning yourself about this: **Business-wise, is a Content Service more like a shop or more like a mobile Telco service?** Will we do better with discrete-event billing or with CDR billing? That may be a tricky question. In my own humble opinion any **Content Service must be better thought of as a shop**, and a CDN is no exception. CDNs create an interesting paradox: the Customer (the one who looks for the service and eventually gets it and pays for it) usually is not the same human that 'consumes' the service. The typical CDN customer is a company that has some important message to make through internet. There can be millions of 'consumers' on demand of that message. There can be thousands of millions of consumption actions in a billing period, exerted by millions of different humans. This fact distracts many people from other more important facts:

-the Service Capabilities are completely determined and agreed before starting to serve

-the SLA is completely established and agreed before starting to serve

-pricing is completely clear and agreed before starting to serve

-no matter there are millions of termination points, it is perfectly possible to track all them to the CDN service and bill all the actions to the proper customer

-a Telco service is strongly asymmetric: the customer is many orders of magnitude less 'powerful' than the service provider; a CDN is not. For a CDN many customers may be in fact bigger financially than the service provider, so there is space for initial negotiation, and there is NO space for wild contract changes in the middle of the billing period just because the service provider gets creative about tariffs or whatever.

So I would say that **CDR billing for a CDN does only complicate things**. Logs of edge activity are the ultimate source for service audit and billing but there is no point in separating individual transactions, time-stamping each one, adding all context that makes a transaction independent from all others, and storing all those millions of records.

A **CDN deserves something that rests midway between event-billing and CDR-billing**. I like to call it '**report-based-billing**'. Some distributed processing (distributed along the edge and regions of the world) may allow us to separate 'reports' about the bytes downloaded from the edge and accountable to each of our customers. These reports are not CDRs. These reports are not either 'unique events' to be billed. These **reports are partial bills** for some time-window and for some customer. We may do processing daily, hourly or even finer than that. We will end up having the daily (for instance) bill for each customer in each region. This daily bill can be accumulated over the month easily so we will have the **bill up to day X in month Y** with little added effort over daily processing. These reports support 'easily' corrections due to failures in service that will have an effect on billing (compensations to customers, free traffic, promotions…) and also support surgical amendments of daily report consolidation in case (for instance) some edge log was unrecoverable at the time of daily processing but was recovered later.

By implementing this '**continuous consumption accounting and continuous report consolidation**' it is **possible to bill CDN (or any content business) immediately after the billing period ends** (month usually), but most important there **is no need to process thousands of millions of CDRs** to produce our bills nor is it needed to have a huge datacenter for this specific purpose.

7 Operation Systems

This concept of 'operation' leads us to an interesting discussion. In the Telco world operation is always present. No system or service can work with 'zero operation'. This concept of **operation goes beyond 'maintenance'**. Operation means **'keeping the service up'**. This is a very varying task from one service to another. One may think that the better the service was imagined the less operation it needs… and that is not a bad idea. It is true. But in the real world 'zero operation' is not yet possible.

Put simply, the services we create have **so many actions** inside that affect so many machines and lines of code that we cannot really believe they can work without **keeping an eye on them**. Taking care of that is '**monitoring'**, and by the way **we never really discovered how to accomplish some tasks automatically** (customer contact, contracting, support calls, replacement of SW versions, etc…) and that is '**support**'. These human concepts of '**monitoring'** and '**support'** have been named in the Telco world: **OSS** (Operation Support Systems) and **BSS** (Business Support Systems), but in real life there is high overlap between them.

How could you possibly think of a task that means operation of a service without being a support to the business? Have you ever seen any business that has operations that do not carry costs? Do you have operations that do not produce business? (If you answered 'yes' to any of the two questions you better review your business…).

The most important (in my view) OSS/BSS in Content Services are:

-**customer provisioning**: customer identity management

-**service provisioning**: contract management + resource allocation

-**event (incidence) ticketing**: link customer + product + ticket + alarms + SLA

-**SLA monitoring**: link product properties + analytics + contract -> alarms -> billing

7.1 Customer/Consumer provisioning

This kind of system, an information system that acquires and handles human identity records has evolved enormously in the recent years. '**Managing Identity'** is an action incredibly powerful for a business and it carries great responsibility that will be enforced by law. However only very recently we are seeing some part of the power of Identity Management in real life.

In a series of internal research articles that I wrote seven years ago I was promoting the idea of a '**partially shared identity**'. At that moment the idea was certainly new as some syndication of 'whole identities' was entering the industry and some more or less 'promising standards' were in the works. We built a demonstration of three commercial platforms that were **loosely-coupled by sharing fragments of the whole identity** of the end user.

Today I'm happy to see that the once 'promising' standards which were overly complex have been forgotten but the leading commercial platforms and the leading identity management platforms (social networks) now allow **cross-authentication by invoking APIs** inspired by the idea of 'set of identity records and set of permissions'. The platform that requires access to your identity data will let you know what 'items' it is requesting from your authenticator before you allow the request to go on. This is a practical implementation of '**partial identity'**.

But let's focus in the simplest purpose of the 'Customer Provisioning': we need to acquire a hook to some human so we can recognize her when she is back, we can give service to some 'address', we can take her feedback and we can send her bills and charge her account for the price of our service.

As I've said the most intelligent approach to knowing our users today is …going directly to our would-be-customer and saying: … 'Do you have a social network in which you are well known? Good, please let me know which one. By the way I have bridges to the biggest three. You can choose the one you prefer to authenticate with me and I will not bother you a single minute entering your data.'

Usually social networks do not hold information about payment methods (VISA, PayPal, etc…) so fortunately for the peace of mind of our customer/consumer that part of the personal data cannot be shared. But taking the more general concept of a 'platform' in which a consumer has a personal account, it is imaginable a business relationship with another platform in which the **consumer would occasionally like to do a purchase but he does not want to rely on them to handle his payment**. In case the consumer gives permission the charge could be sent to the first platform that is already trusted by the consumer. The first platform will handle consumer's money and the new (or second) platform will just be a provider of goods to the first platform, sending these goods (in our case Content Products) directly to the address of the consumer. In this way the consumer obtains the good effects of sharing his payment data without actually sharing them.

I have to say that I'm also happy to see this concept today implemented in Amazon Marketplace. In case of virtual goods (Content) it could be even easier to implement (or more complicated it depends on the nature of content and the kind of delivery that is appropriate.)

7.2 Service Provisioning

This is hard stuff. As I mentioned at the beginning of this article '…today we are not talking about delivery…' But in fact delivery is the most attractive part of content businesses from a technological perspective. It is also the biggest source of pain for the content business. It is where you can fail, where you can be wrong, have the wrong strategy, have the wrong infrastructure, the wrong scale… and you see… It is a hard problem to solve, but this is the reason it is so exciting. CDNs are exciting. **Service Provisioning is directly related to how you plan and execute your content delivery**.

Provisioning more service is a daily problem in CDNs. It may be due to a new customer arriving or because existing customers demand 'more service'. It cannot be taken lightly. Customers/Consumers can be everywhere through your footprint, even worldwide, but you do not have PoPs everywhere and your PoPs do not have infinite capacity. **Service provisioning must be the result of thorough thinking and data analysis about your current and projected demand**.

As I commented in a different article, a CDN takes requests from essentially anywhere and then has to compute '**request routing'** to decide per request which is the best resource to serve the request. Resources are not 'anywhere'. There is a '**footprint'** for a CDN. There are many strategies to do this computation, and there are many high level strategies to geographically distribute resources. As of recently the edge of CDNs starts to be less distributed. Or it would be better to say that **the original trend of 'sprawling the edge' through the world has been greatly slowed down**. CDNs nowadays enhance the capacity of their edges but they have almost stopped branching finely the edge. There is a reason for this behavior: the most consumed content in CDNs is VoD (per byte) and pre-recorded content delivery is not very sensible to edge ramification. With appropriate buffering a few-PoPs-edge can do very well with VoD. On the contrary live events and low latency events depend very much in proper branching of the edge.

When the probability of dropping requests in our request routing due to the misalignment of our demand and our resources capacity/position gets above a certain threshold we will need to increase our service.

In a CDN there is usually dynamic allocation of resources to requests. There is no static allocation of resources to some requests, for example to some customer. But there are always exceptions. In a sophisticated CDN **it is possible to compute the request routing function with reservation of resources for some customers**. This technique of course makes global request routing much more complicated but introduces new business models and new possibilities in SLAs that are worth considering.

In case your CDN applies capacity reservation then a new customer with a reservation will have an immediate impact in service provisioning.

Other impacts in service provisioning emanate from the very nature of some CDN services. For example, when a CDN starts caching a domain of a new customer it is usually necessary to inform the caches of the name of this domain so they (the caches) change their policy to active caching. This action should be triggered by a proper service provisioning system.

7.4 Event ticketing

In any online service it is important to keep track of complaints and possible service failure. I would say that this is not a very special part of a Content service. (Please understand me right: being a Content Service does not make this special over other Services.) Essentially it is a **workflow system that will let you account for events and link them to: Customer Identity + Operations work orders**. Easy as it is to implement a simple workflow it is worth the time to use alarms and time stamps to implement a 'promptly communication policy'. Once you have received notice of a potential problem clock starts ticking and you must ensure that all stakeholders receive updates of your action in due time. The ticketing system does exactly that. It creates 'tickets' and manages their lifecycle. A **ticket is a piece of information that accounts for a potential problem**. As more details are added to the ticket all stakeholders get benefits from the existence of the ticket: the customer gets responses and corrective actions, operations get information to address a problem, the whole system gets repaired, other users avoid running into problems, your data backend and analytical accounting get info about your time to solve problems and number of problems and cost of repairing.

All in all the ticketing system is your opportunity to implement transparency and a 'communication bus' that works for emergencies and gives the right priority to many different events and incidences.

7.5 SLA Monitoring

This is an information system that I rarely see 'out-of-the-box'. You need to build your own most of the times. Many vendors of video equipment and/or OVPs sell 'probes' that you can insert in a variety of points in your video distribution chain. These probes can give you a plethora of measures or 'quality insights' about your service. Many other vendors will provide you with network probes, traffic analysis, etc…It is advisable to have a **solid background in performance analysis** before trying to use the vendors' suggestion of a set of **SLO** (Service Level Objective) to build a **SLA** (Service Level Agreement) for a customer. It happens many times that the understanding that we get from the written SLA is not the same that the customer gets. And it happens even more frequently that the measures that the probes give us DO NOT implement what we have announced in our SLA. It is key to clear any doubt about what is measured and how, exactly, it is measured. (For more in depth information you may want to read my previous article: *CDN Performance Management*.)

The **SLA is our commitment in front of our customer to grant certain characteristics** of content traffic. Today no one will be selling a Content Service on the 'soft promise' that the Service will scale seamlessly with demand, the traffic shaping will be correct, the delay low, the losses inexistent, the encoding quality superb,… All these 'fuzzy statements about service quality' are simply not admitted. The 'reach' of the service in terms of what that service can really do cannot be an aspiration. It must be documented in an SLA. This **SLA will state clearly what we can expect from the service using quantitative measures**.

There are very serious differences between 'cheap' CDNs / content services and 'serious' 'high quality' services. Even when the finished product may occasionally look the same: video on demand, channels, events… there is a whole world of complexity about **preparing the service in advance** to support any eventuality. A quality service provider may spend easily 5X to 10X more than a cheap provider preparing for variations in load and preparing for all kinds of performance threats. Of course taking care of performance in advance is expensive. It involves lots of analysis of your systems, constant re-design and improvement, buying capacity in excess of demand, buying redundancy, hiring emergency teams, buying monitoring systems…how can a business survive to this investment? . This investment is an opportunity for positive publicity and for a business model based on quality and SLAs**. If you are highly confident in your performance you can sign a very aggressive SLA**, promising high quality marks and accepting penalties for casual infringement.

There used to be a huge difference in the delivery options available to a Content Service in the early days of CDNs (15 years ago). At that moment it was:

Option 1: **Plain carrier connectivity** service: no content oriented SLA. Use it at your own risk. Only percentiles of drop packets and average Mbps available were eligible to describe quality. Nothing was said about integrity of individual transactions.

Option 2: **CDN**. A 'shy' SLA, promising a certain uptime of the service, certain bounded average transaction-latency, a certain set of content-related quality KPIs: buffering ratio, time to show, a certain level of cache-hit ratio…

At that moment option 2 was much more valuable than option 1 (no surprise…), and for that reason prices could be 10X raw Carrier traffic prices for CDN customers.

Today after years of CDN business, after continued improvement in Carrier services, but also after a serious escalation in demand of Content and a serious escalation in typical Content bitrate…SLAs have to be different and CDN prices vs traffic prices have to be in a different ratio. Anyway this is a matter for a much longer article.

What happens today is that **SLAs are now a much less impressive sales tool**. Almost all CDNs show very similar SLAs. I've been able to notice a trend that is very interesting. **Some CDNs are getting increasingly 'bold'**, promising to achieve certain SLOs that are close-to-impossible to grant. This is probably an effect of **the way most customers check SLAs: they check them only in case of serious failure**. Or even disastrous failure. There is not a culture of reviewing the quality of the traffic when there are no complaints from end users. Companies that commercialize SLA-based services have noticed this and they may be in some cases **relaxing their vigilance on SLAs**, moving resources to other more profitable activities and reacting only in the rare case of a disastrous infringement of the SLA. In that case they just refund the customer and go on with their activity. But at the same time they keep on selling service on SLA promises.

My own personal view about managing SLAs is not aligned with this 'react only in case of problem' style. It is true that the underlying carrier services are today more reliable than 15 years ago, but as I've said **Content Technology keeps pushing the envelope** so it would be better to redefine the quality standards. We should not assume that IP-broadcasting of a worldwide event 'must' carry a variable delay of 30s to 60s. We should not assume that the end user will have to live with a high buffering ratio for 4K content. We should not assume that the end user must optimize his player for whatever transport my content service uses.

It is a good selling point to provide **SLA monitoring reports** for all services contracted by the customer on a **monthly** basis. These reports will show how closely we have monitored the SLA, and which margin we have had across the month for every SLO in the SLA. Of course these reports also help our internal engineering in analyzing the infrastructure. A good management will create a **cycle of continuous improvement** that will give us a bigger margin in our SLOs and/or the ability to support more aggressive SLOs.

SLAs and their included SLOs are great opportunities for **service differentiation**. If my service can have seriously low latency, or no buffering for 4K, let us demonstrate it month by month with reports that we send for free to all customers.

So having SLA reports for all customers all the time is a good idea. These reports can usually be drawn from our Performance Management Systems and through mediation can be personalized to each Customer.

8 Inventory Systems

These are of course core components of our exploitation. As commented above we must keep track of at least: tech resources, customers, products.

I like to start with the hardcore components of a good delivery: tech resources

8.1 Technical Inventory

This **technical inventory** is a concept that comes very close to the classical OSS inventory of machines. I say close and not identical because a content service should go beyond connectivity in the analysis of the technical resources.

The technical inventory must contain a **list of all machines in the service** (mostly delivery machines in a content service) with **all their key characteristics**: capacity, location, status ... These are **long term informative items**. Real time-load is not represented in the inventory. An alarm (malfunction) may or may not be represented in the inventory. It may be there to signal that a machine is out of service: status out.

Having a well-structured tech inventory helps a lot when implementing **automated processes for increasing the delivery capacity**. In a CDN it is also especially important to regularly compute the resource map and the demand map. In fact the request routing function is a mapping of the demand onto the resources. Ideally this mapping would be computed instantly and the calculation repeated continuously.

The technical inventory is not required to represent the current instantaneous load of every machine. That is the responsibility of the request routing function. But the request routing is greatly supported by a comprehensive, well-structured technical inventory in which a 'logical item' (like for instance a cache) can be linked to a hardware part description (inventory).

Having this rich data HW inventory allows us to implement an **automated capacity forecasting process**. In case a new big customer wants to receive service we may quickly calculate a projection of demand and determine (through the inventory) which is the best place to increase capacity.

It is also very useful to **link the inventory to the Event ticketing system**. In case a machine is involved in a service malfunction that machine can be quickly identified, marked as out of service, and retired from our delivery and request routing functions. At the same time our OSS will be triggered for a repair on site, a replacement… or simply we may mark the datacenter as eligible for end of the month visit.

The **tech inventory must be also linked to our cost computation process** that also takes data from our **mediation systems and our purchases department**. We want to know the lifetime of each machine that we operate and which is the impact of each machine in our costs. This impact has CAPEX and OPEX components. Having these links between analytic systems allows us to implement a long term profitability analysis of our business.

8.2 Product Inventory AKA Commercial portfolio

As we saw when talking about the service ideation there is a portfolio of different products. In case of Content Products this portfolio maps to a **list of titles and a wealth of 'actions'** or capabilities that our customers buy the right to execute. We may package titles with actions in the most creative way that anyone could imagine : Channels, Pre-recorded content on demand ,events…any combination of the mentioned with an idea of quality through 'viewing profiles' (bitrate, frame size, audio quality, frame rate, codec, color depth,…), monthly subscription, pay per view, hour bonus, plain tariff, premium plain tariff,…whatever. But **how do we map all these 'products' to our other systems**: technical inventory, customer inventory, mediation, analytics, portals, SLAs monitoring, event ticketing…

The best solution is to **build links from the Product Inventory to all the systems** in a way that makes sense. And that 'way' is different for each of the exploitation system components that we have described.

For instance, when designing a VoD product we should map it to the Technical Inventory to be sure that the list of codecs + transports is supported by our streamers. If we have an heterogeneous population of streamers in which some support the new Product and some not…we need to link that knowledge to customer provisioning so we do not sell a footprint for that product that we cannot serve…. If that same VoD product will be billed through a monthly plain tariff with a cap in traffic and with a closed list of titles but we allow premium titles to be streamed for an extra fee… we must include informational tips in the Product inventory so the link to the Mediation can build properly the monthly bill for this customer. If we want to apply different pricing for different places in the world we need to include those tips in the Product inventory and use them to link to Mediation and to link to Customer provisioning and to link to Portals.

Of course the most obvious link of the Product inventory is to the Product Owner Portal. The Product Owner Portal is the technical tool that is used to design the product capabilities (actions) and as I've said it is a system that lives at the core of the Exploitation system, in a dungeon where only developers and a few product owners can touch it. As it goes through frequent updates to provide new and exciting capabilities the same happens to the Product Inventory. This inventory evolves with the Product Owner Portal, to reflect and account for every new capability and to store the tips that are used to link via many processes to the rest of exploitation system components.

8.3 Customer Inventory

As we have mentioned before today having information about our customers is an asset that has turned to be more powerful than ever before. In fact there is serious fight for having the personal data records of customers among commercial platforms. For this reason 'sharing' part of the customer identity is the new trend.

Anyway, let's assume that we are the primary source of data about one particular customer. In that case we need to **account for enough information to legally approach our customer**: full **Name**, full **address**, fiscal **ID**, **payment data**. On top of that we may pile up whatever data we dare to ask our customer about himself.

And on top of **'what our customer knows we know about him'**… we will add a ton of **'insights'** that we can get about our customer just watching his public activity. Important: watching a public activity means taking notes on actions that we are supposed to notice as service providers… It is not and will never be spying on other activities of our customer of course! There are many insights that online businesses do not exploit, or at least exploiting them was cumbersome and not very fashionable until recently. The 'Big Data' age is changing that. Profiling customers is a hard task that involves lots of interesting algorithms to correlate data sources, but the good part is that we already have the data sources: purchases, timestamps of purchases, traffic, clicks on media players, 'behavior' at large. And the other good thing about collecting all these insights is that it is lawful and it is a 'win-win' action that benefits equally the service and the Customer.

The Customer Inventory is of course linked to Portals, to Mediation, to event ticketing, to some Analytics and to SLA monitoring.

9 Conclusions

We have seen that **Exploitation Systems are a set of 'systems' that rival in complexity with the core service systems, usually called 'service delivery infrastructure'**. But Services must be exploitable…easy for the service provider.

We have seen that we **cannot buy Exploitation Systems off the shelves.** OK, we can. But is it good to go with an all-purpose exploitation suite with 100+ modules that are designed to behave equally when selling cars, houses, apples …movies? My guess is that Content Businesses have some specifics that put them apart from other Services, and even there are specifics that separate one Content Service from another. If we buy a famous exploitation suite for online businesses **we MUST have a clear design in mind** to customize it.

We have seen that **some formality when thinking at the design stage helps later**. I suggest creating first a **Service Exploitation Model** and implementing a **Service Exploitation System** after it.

We have decomposed the typical pipeline for exploitation of Content Services in major subsystems: **Portals**, **Mediation**, **Operation**, **Inventories**.

We have reviewed the **major subsystems** of the exploitation and analyzed the **good properties** that each subsystem should have for Content Services and also have discussed **trends in design** of these subsystems.

While reviewing the desired properties of subsystems we have noticed **the links and processes** that we need to create between them. We have noticed the huge possibilities that we get from **linking subsystems** that in other Service views (alien to Content) are kept separated. These links are key to the coordinated behavior of the Exploitation and they **must be instrumented by adding information that makes the subsystems cooperate.**

As a final remark I would like to emphasize how important it is to **apply innovation, analysis and continuous improvement methods to the Exploitation of Content Services**. I know it looks fancier to deal with the infrastructure for the delivery of video but there are a lot of interesting and even scientific problems to solve in Exploitation.

Best Regards. Adolfo M. Rosas