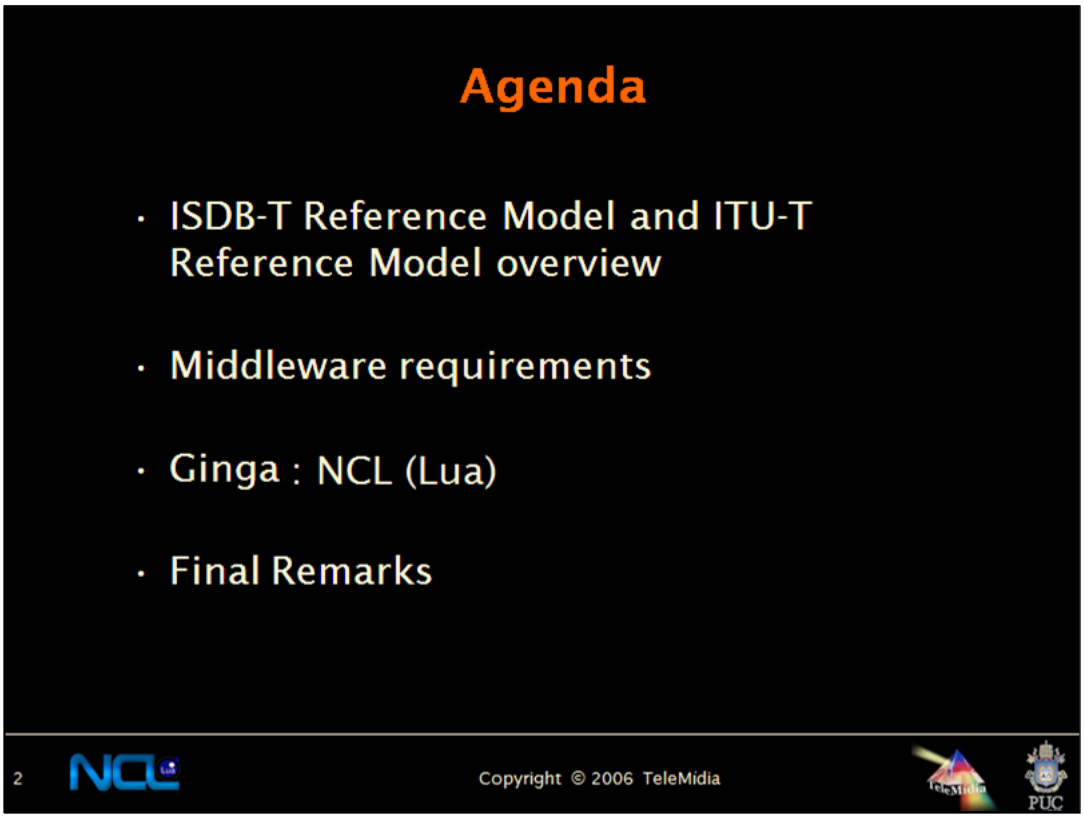
Welcome to this tutorial: **An Introduction to DTV and to Ginga-NCL**



I've divided my presentation into four parts.

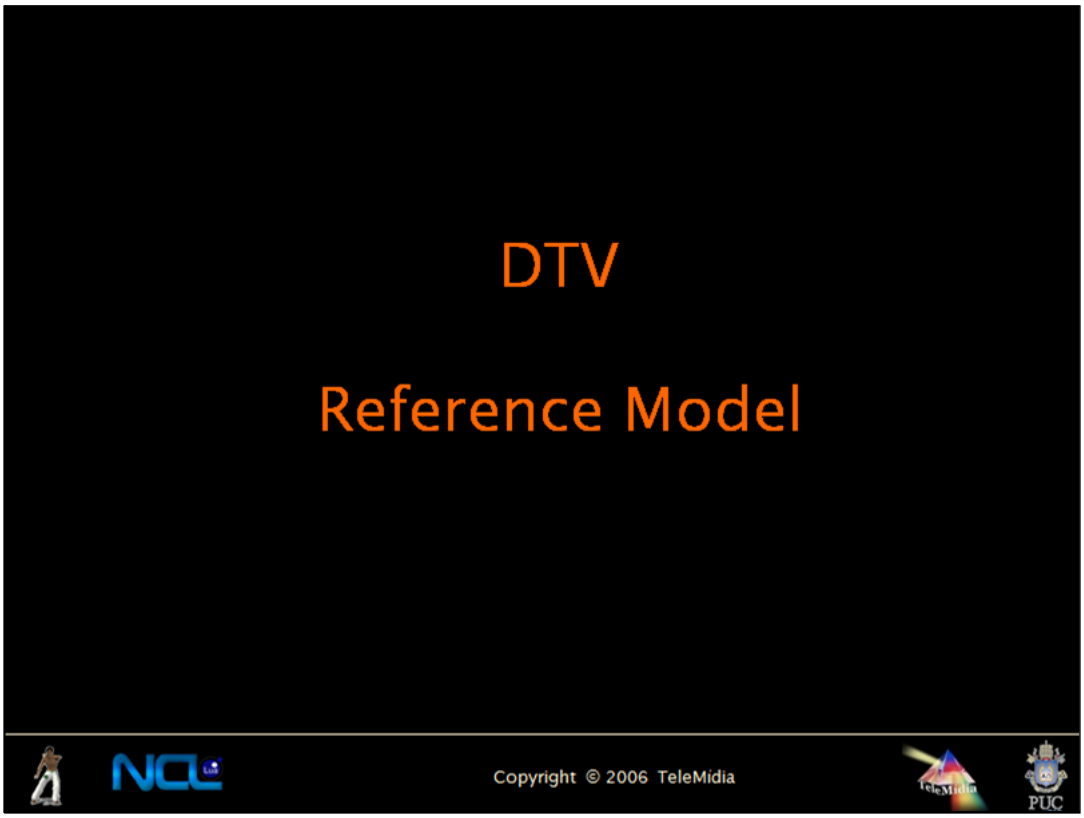
First, I'd like to quickly overview the typical Reference Model for DTV . In particular, I'd like to introduce the International Standard for Digital Broadcasting – ISDB-T Reference Model, in its profile adopted in all Latin-American countries, and also to introduce the IPTV ITU-T Reference Model.

Second, I'd like to go through some middleware design decisions, considering the support offered to applications.

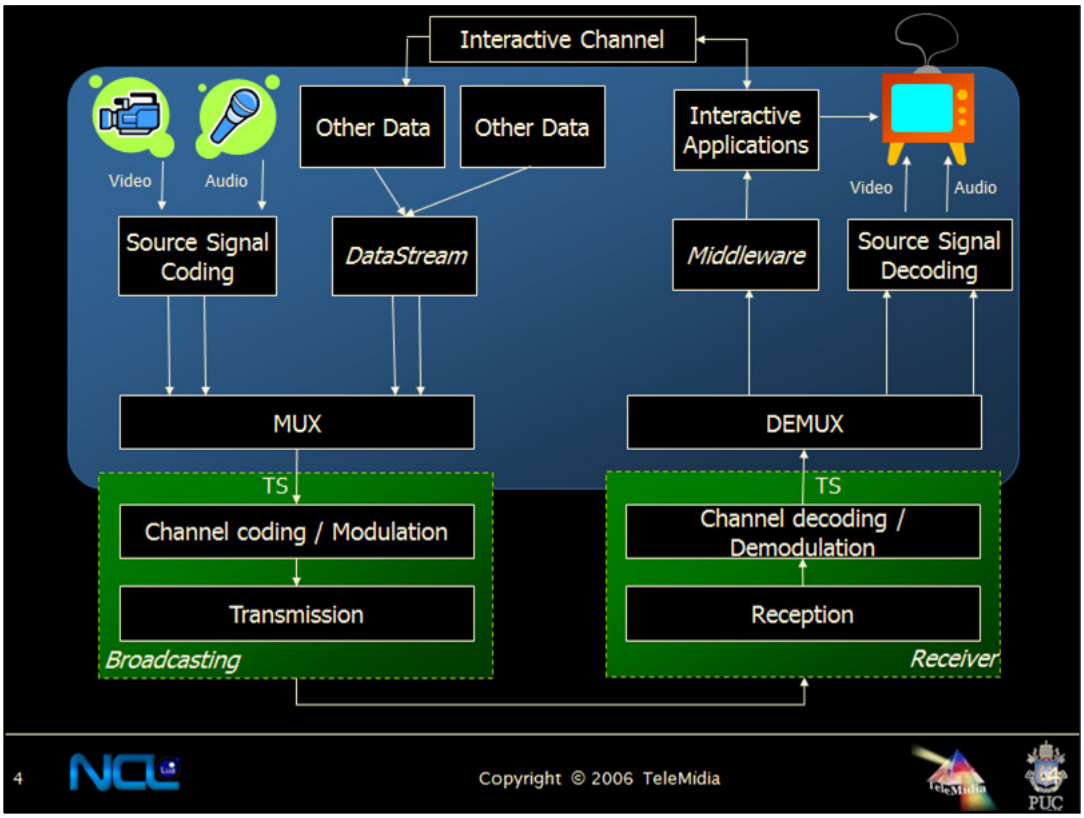
Then, I'll present the Ginga middleware architecture. I'll focus much more on the declarative environment of Ginga, since this is the single required subsystem of Ginga and, indeed, its main innovation. I'll try to stress some differences with regards to other declarative middleware solutions, and to highligh some of the NCL and Lua features.

NCL (Nested Context Language) is the declarative language of Ginga. Lua is the scripting language of NCL.

Finally, I will try to address some future research directions established for the TeleMidia Lab, where Ginga-NCL has been designed.



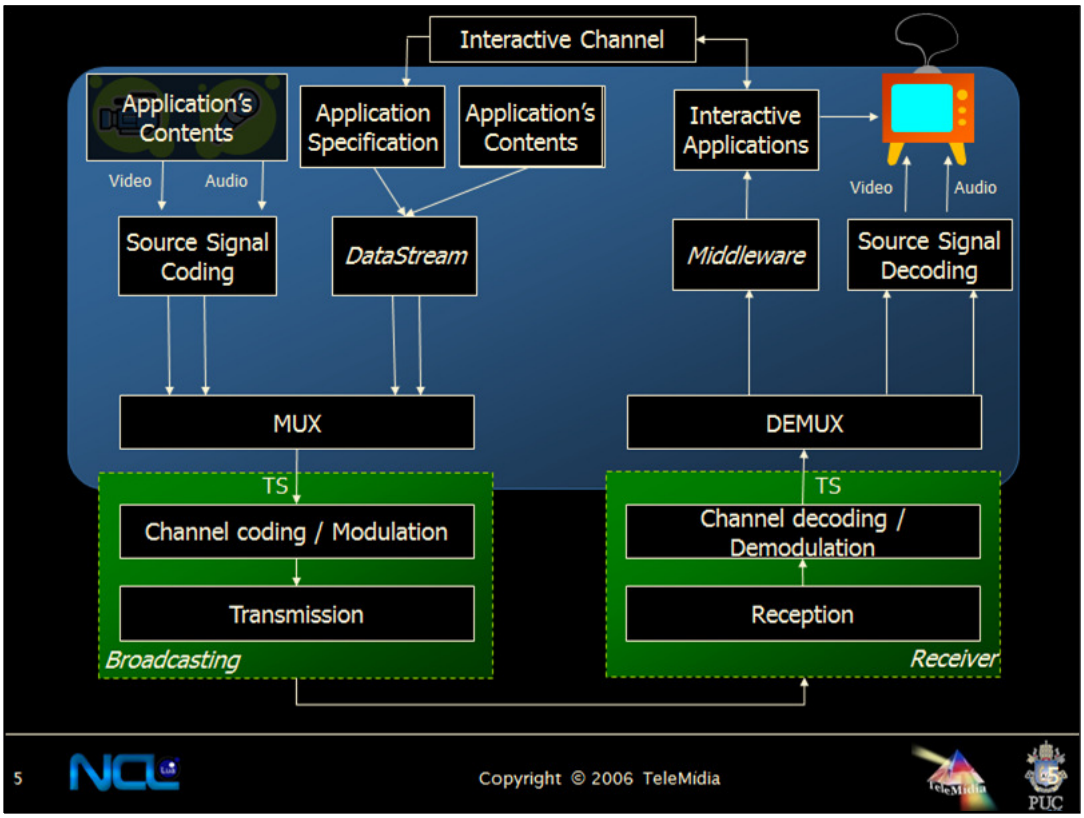
So, let's start briefly presenting the ISDB-T Digital TV Reference Model and the ITU-T Reference Model for IPTV services . The goal here is only to put things into a context before presenting the middleware architecture, the main focus of this tutorial.



This slide presents the workflow of a DTV program, since its conception to its consumption.

The main video and the main audio are captured and digitally encoded generating an elementary video stream and an elementary audio stream. Other data that compounds the new non-linear TV program is serialized (sometimes encapsulated in IP packets), producing other elementary streams.

In a non-linear TV program, the main audio, the main video and other data are related in time and space, compounding a **DTV application**.



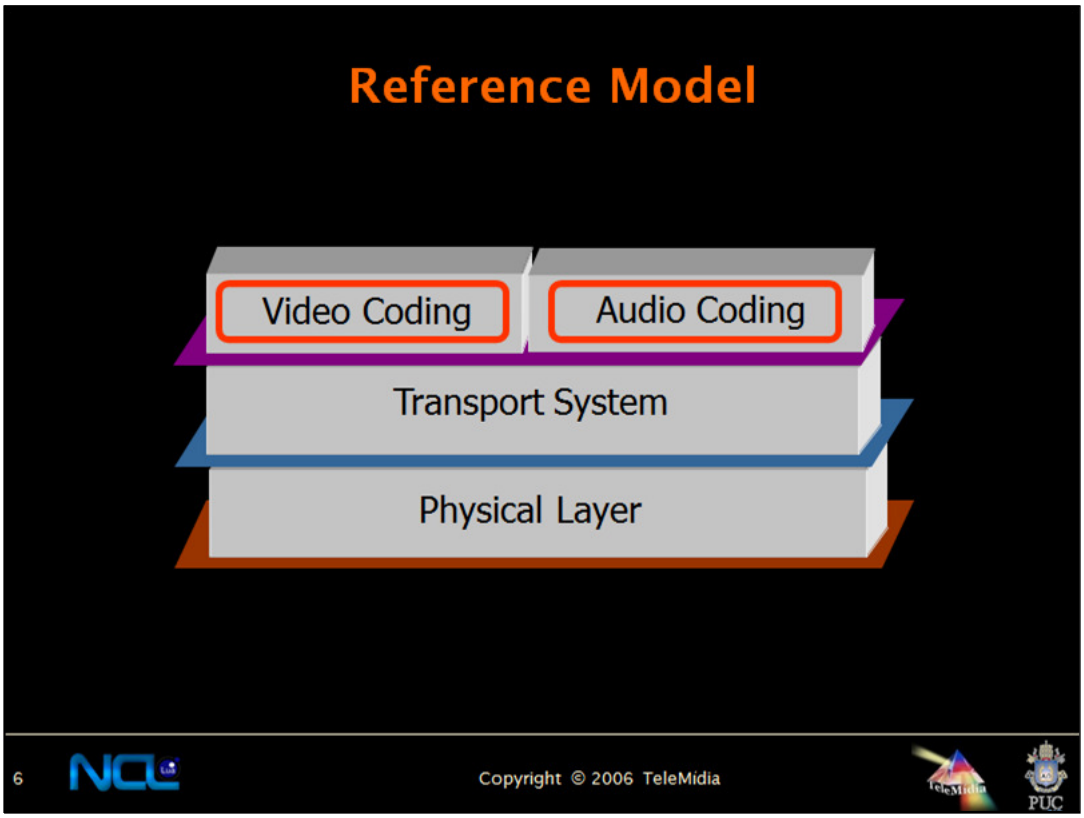
These relationships among media content are part of a DTV-application **specification**.

All these streams are then multiplexed into a single stream called Transport Stream, or simply TS.

This stream is then modulated and transmitted by broadcasting in a terrestrial DTV system, or transmitted by multicasting (or unicasting) in IPTV or P2PTV systems.

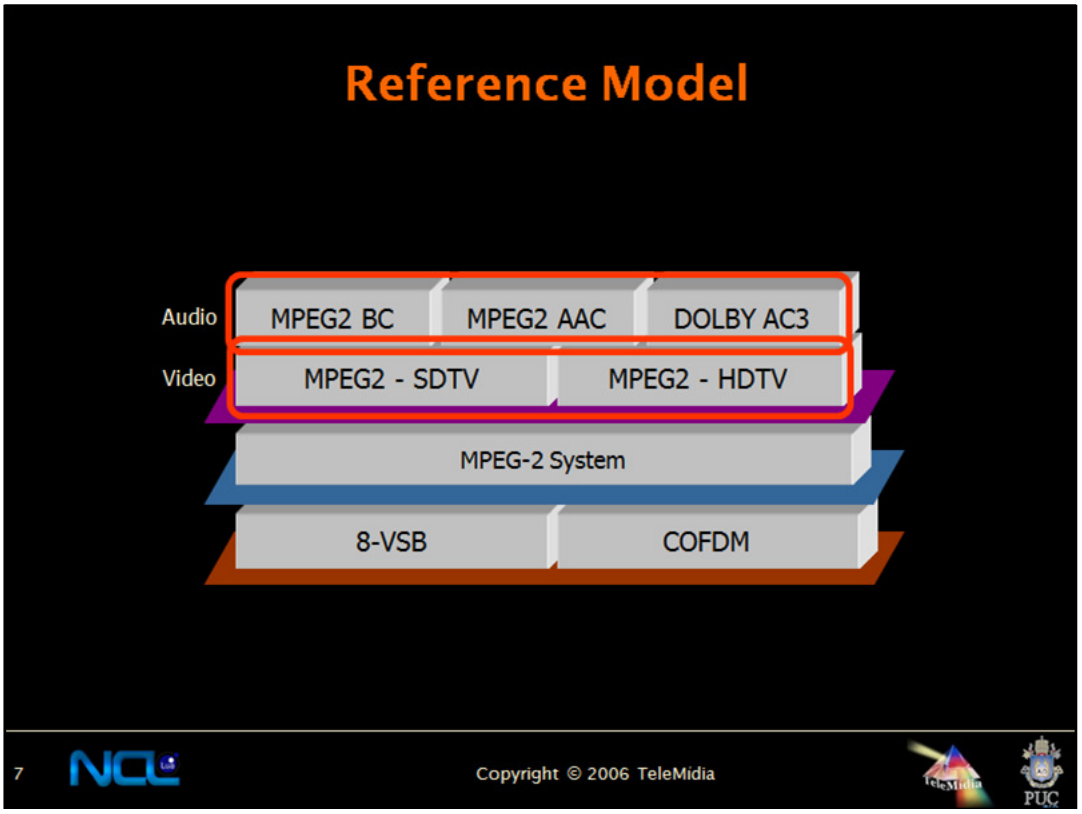
An inverse process is performed in the receiver side.

All these steps are based on well-established standards,

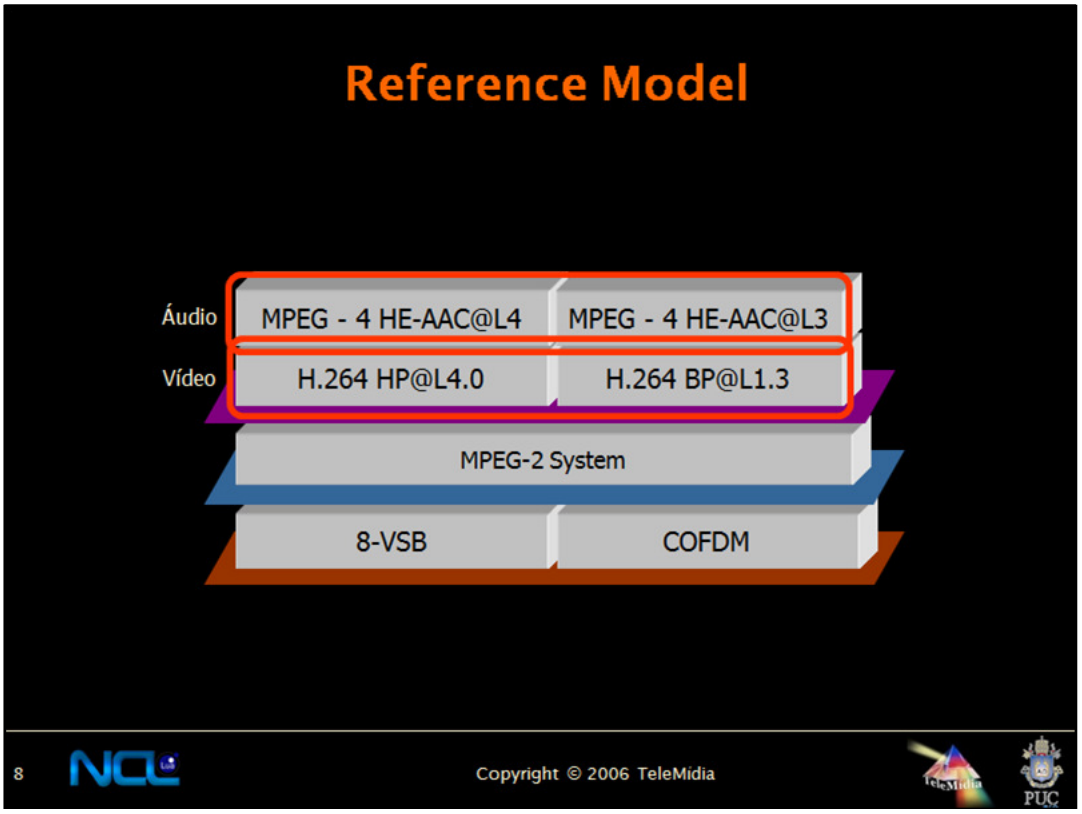


which compound a DTV reference model.

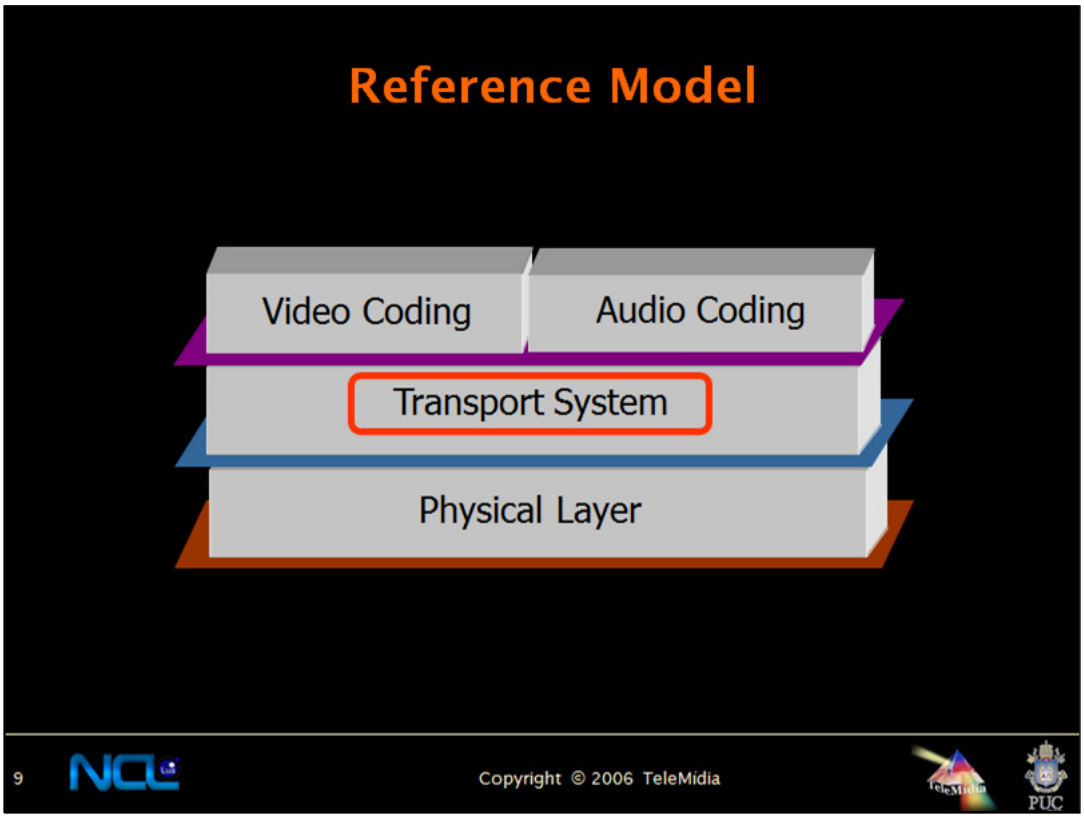
As for the video and audio coding,



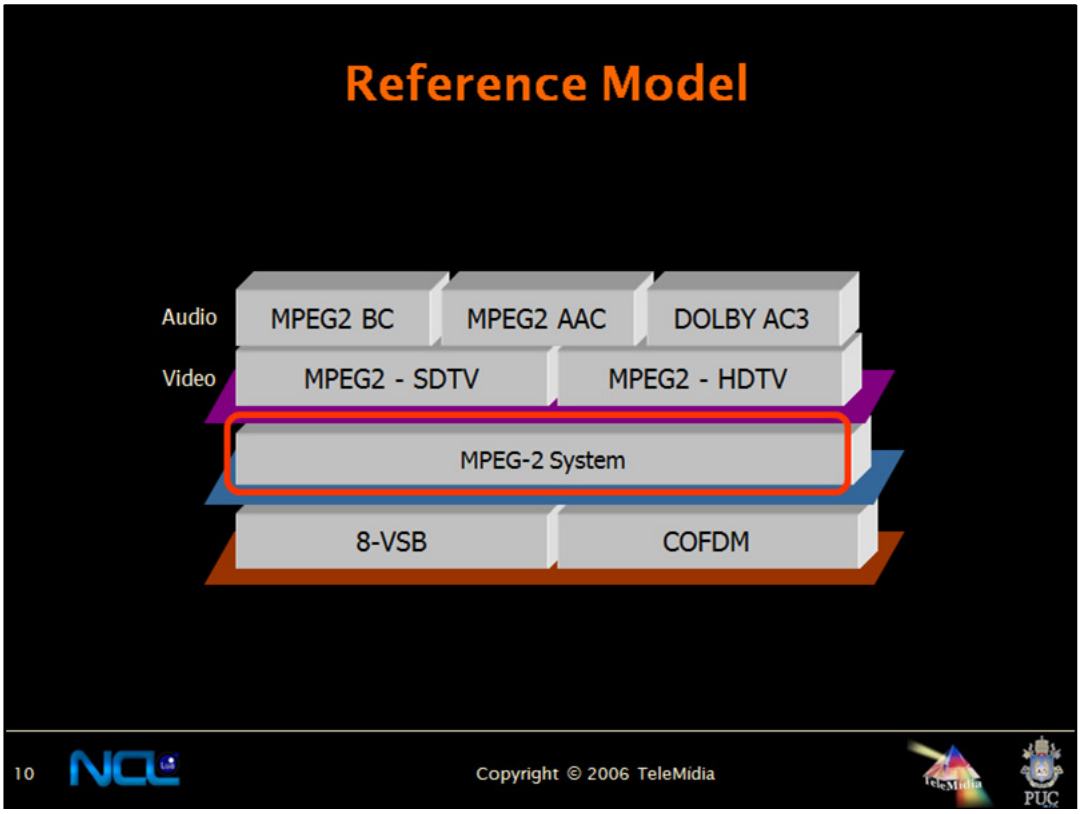
different from all three main systems (the European, the Japanese, and the American),



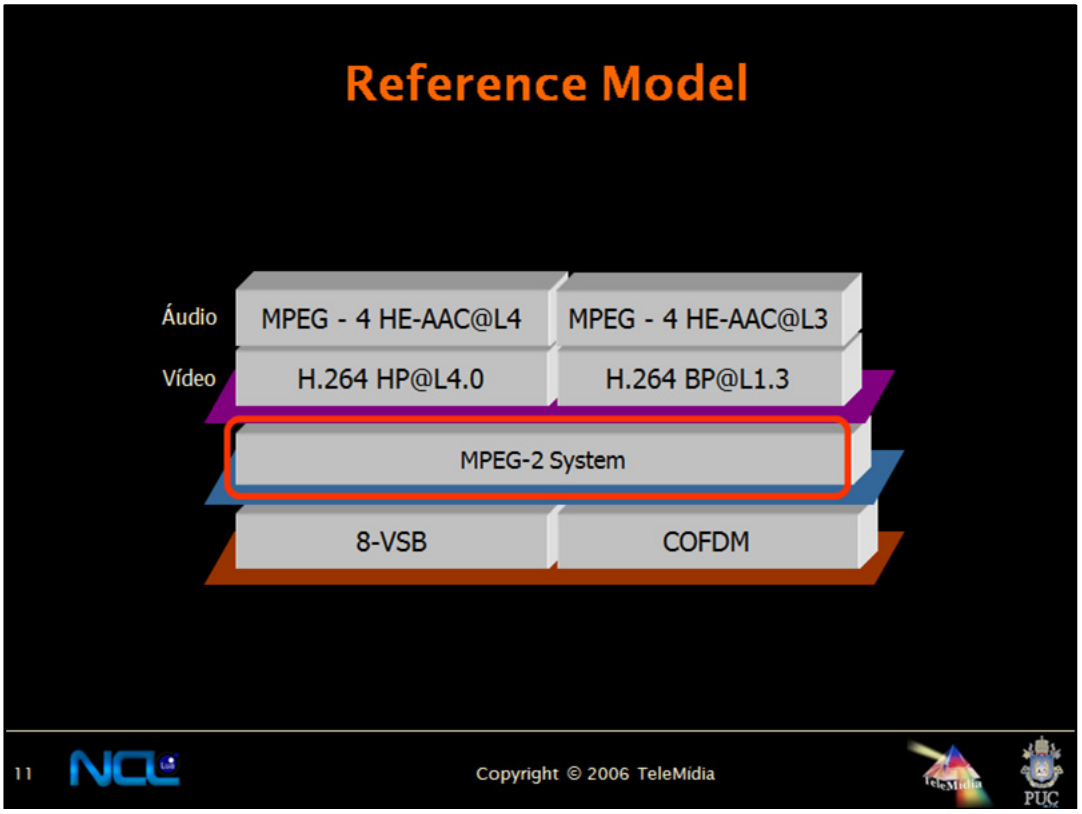
in the ISDB Reference Model adopted in Latin-American, MPEG-4 was chosen as the audio and video coding, both for fixed and for portable receivers. Of course the new profile had to take advantage of a newer coding technology.



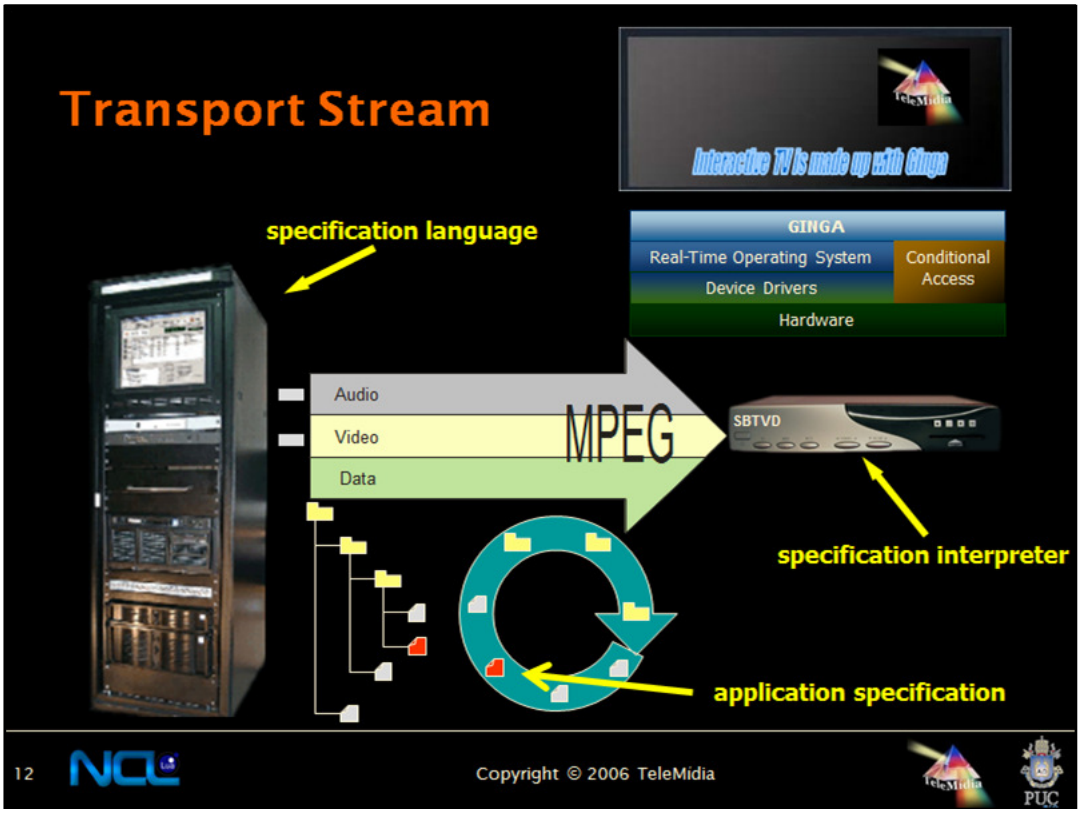
As for the Transport System,



similar to all main terrestrial DTV and most IPTV systems,



in the ISDB Reference Model adopted in Latin-American , MPEG-2 System is responsible for generating application data streams, and for multiplexing them with the main audio and video streams in a single flow, named Transport Stream (TS).



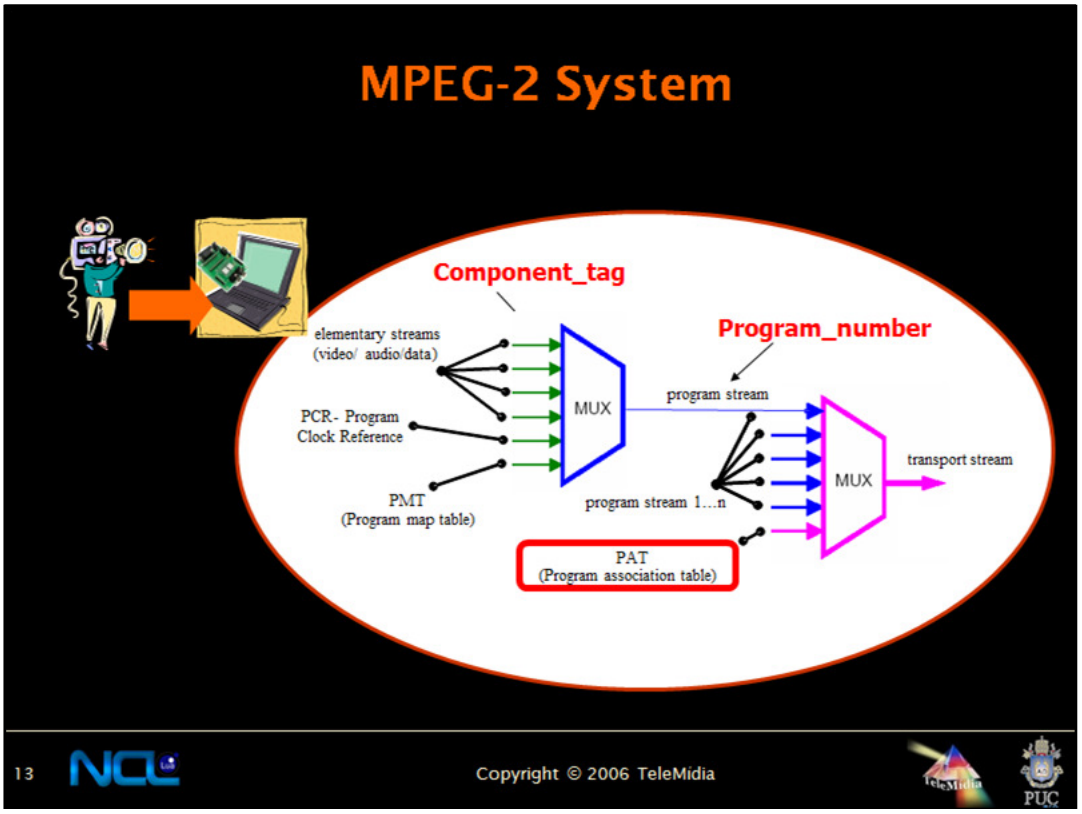
Applications' content (media content and the application specification) can be transmitted on demand (as in VoD services) or as unsolicited data (as for example, in data carousels, transmitted time after time, as in live TV). So, in a DTV system it is common to have both pushed and pulled data transmissions.

Pushed contents that are not directly transmitted as elementary streams must be placed in a file system that is transmitted in a carousel, time after time. All temporal and spatial relationships among media objects that compound a non-linear TV program are specified in a document (also part of the application), which may also be placed in the carousel.

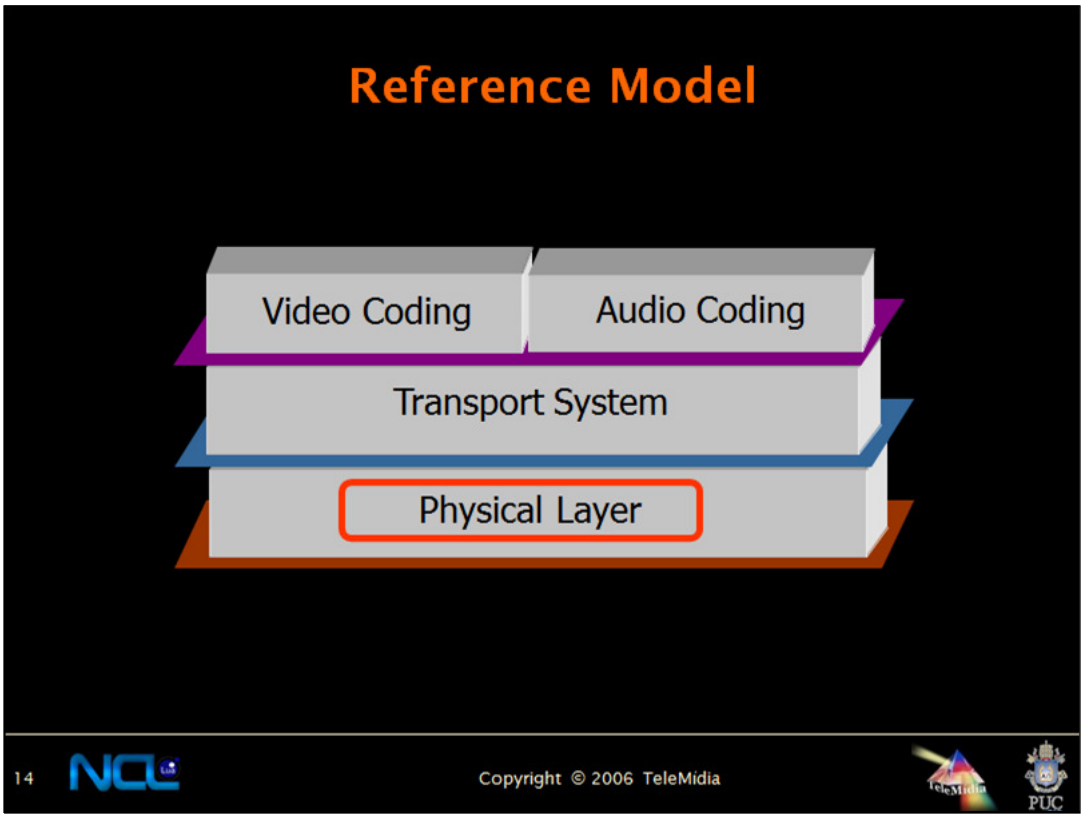
This kind of service, defined and supported by the application specification, is known as "asynchronous service", and it is the only way to have media synchronization with unpredictable events, like viewer interactions.

In the content producer side, the specification document is carried out using some specification language, and in the receiver (client) side, this document (the application specification) is parsed and interpreted by a **formatter**, or presentation user agent (supported by the client-side middleware).

In live transmissions, a stream event may be responsible for triggering the application. Stream events are also used for live editings.

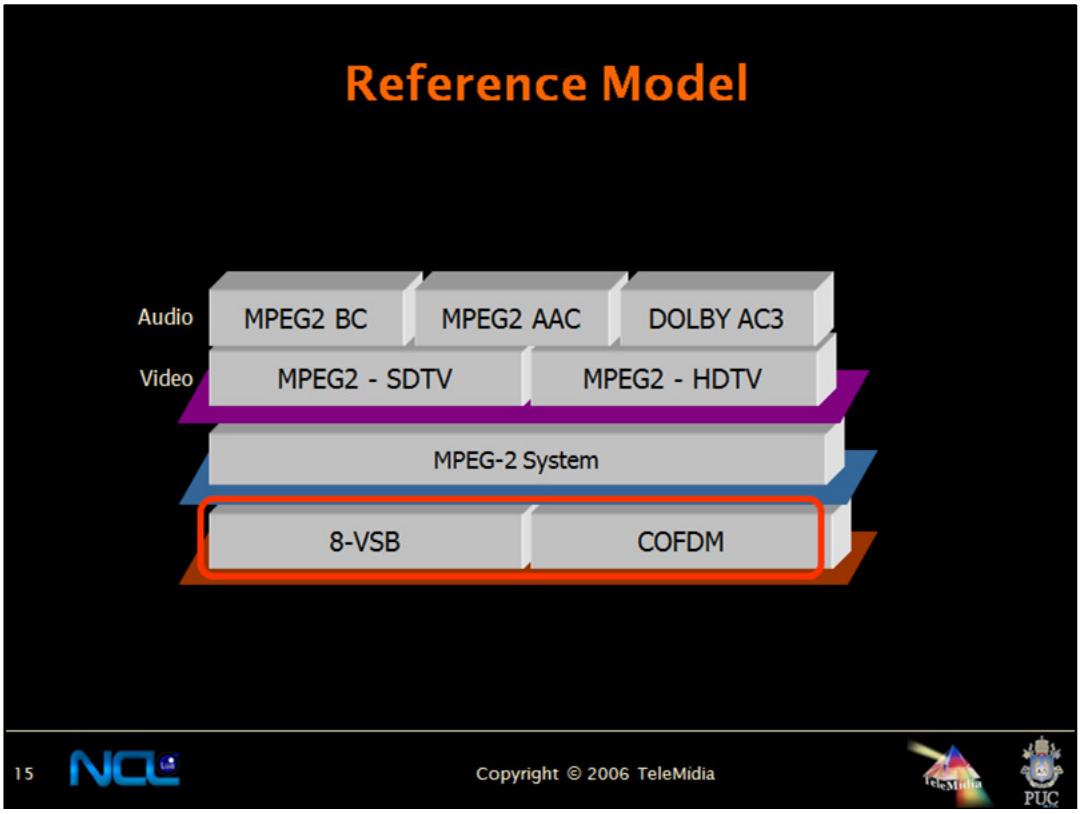


In MPEG-2 multiplexing, a table, named PAT (Program Association Table) identifies each program stream, which is composed by several elementary streams (including those transporting carousels associated with an application), which in their turn are identified by a table named PMT (Program Map Table).

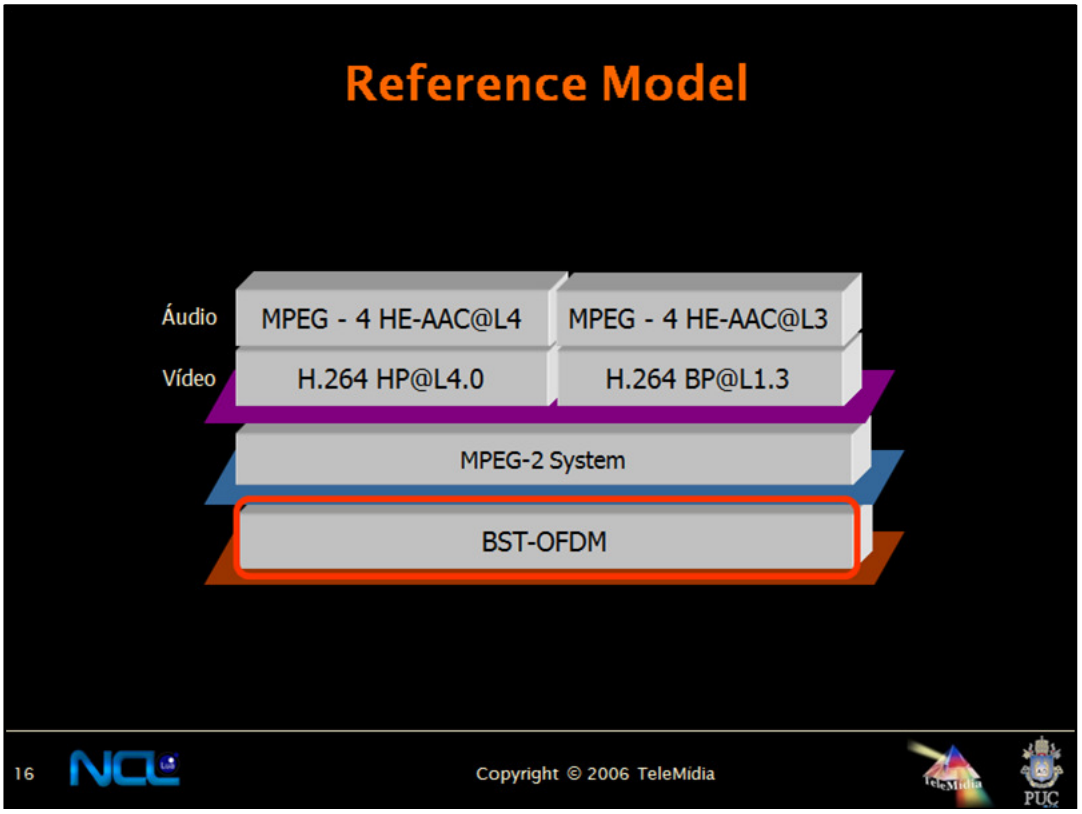


The Physical Layer differs for each DTV system.

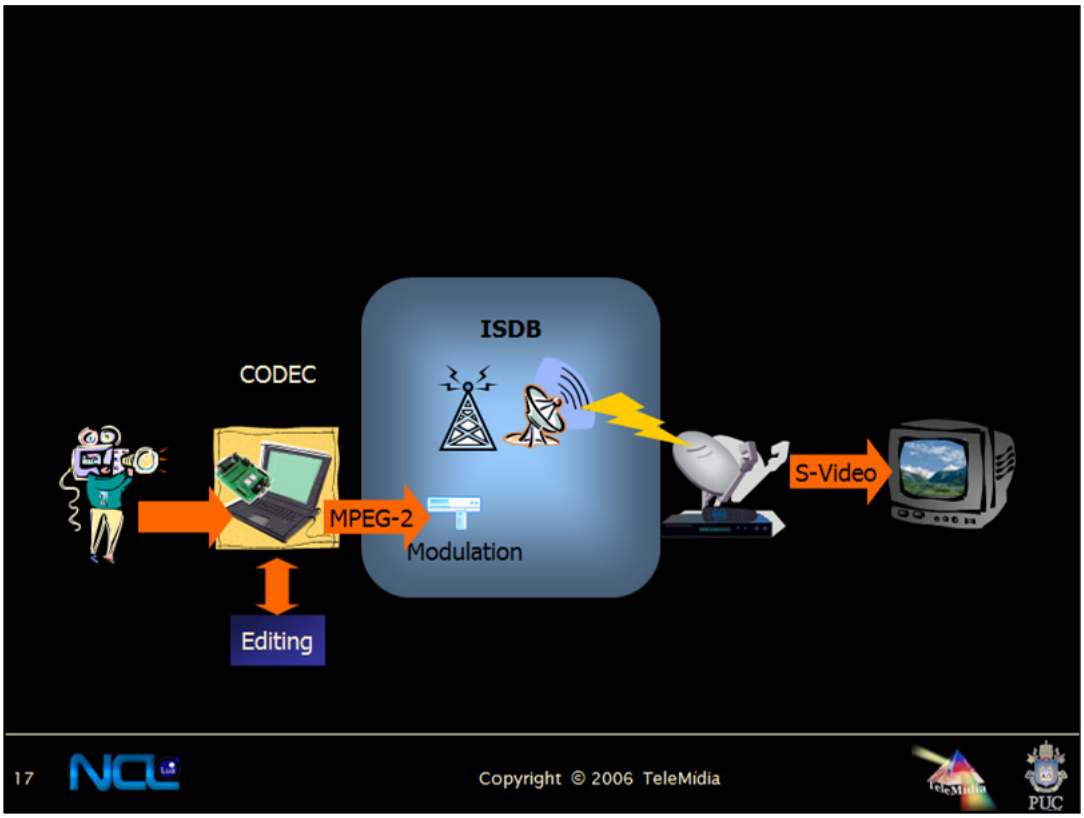
In terrestrial DTV systems, there are several options to modulate the transport stream to be broadcasted,



as the ones used by the American and the European terrestrial DTV systems.



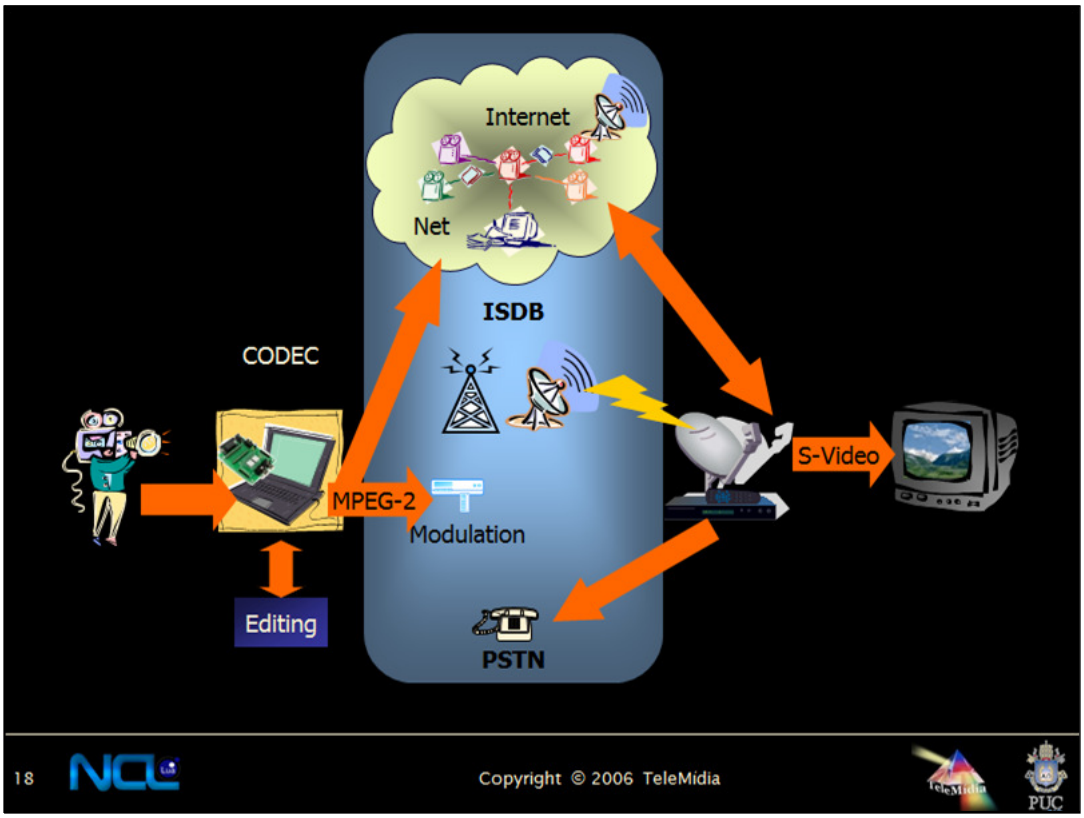
All ISDB-T profiles use the same modulation procedure known as BST-OFDM.



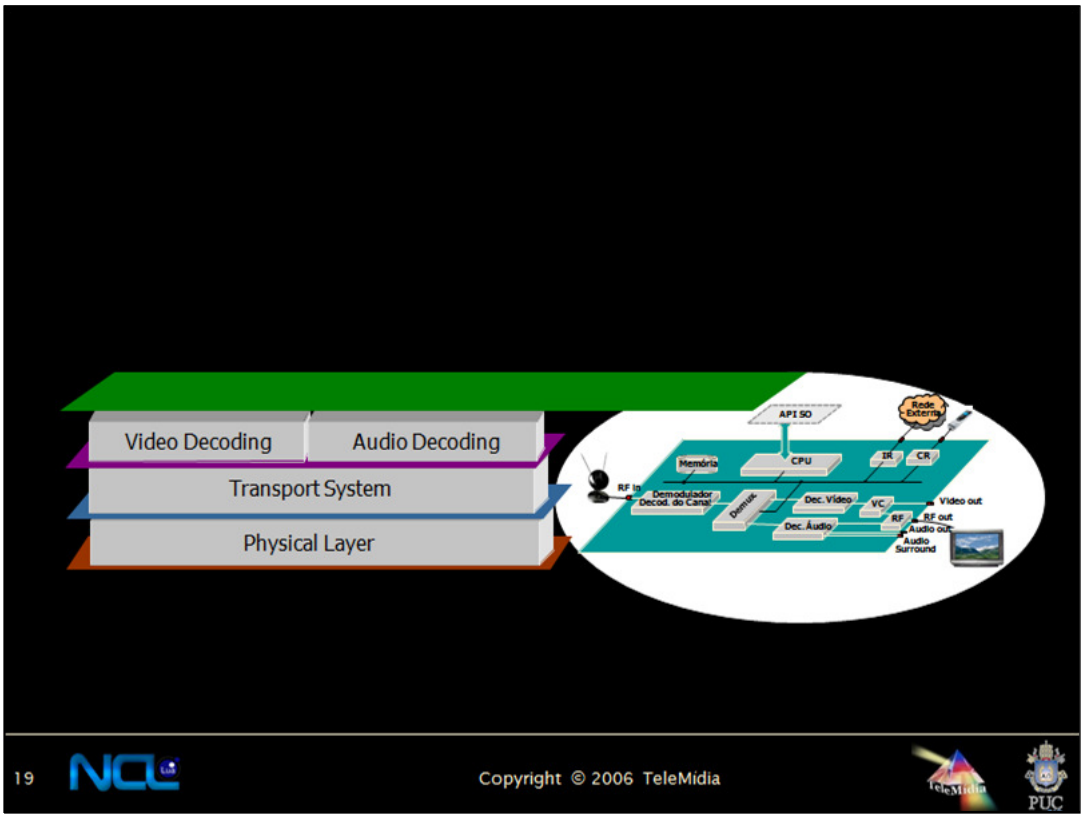
A terrestrial DTV system may have no other network than the broadcast network.

Even in this case, we can still have interactive applications, but the navigation scope is limited to the data transmitted in the carousel.

This option is usually called local interactivity or wallet garden interactivity.

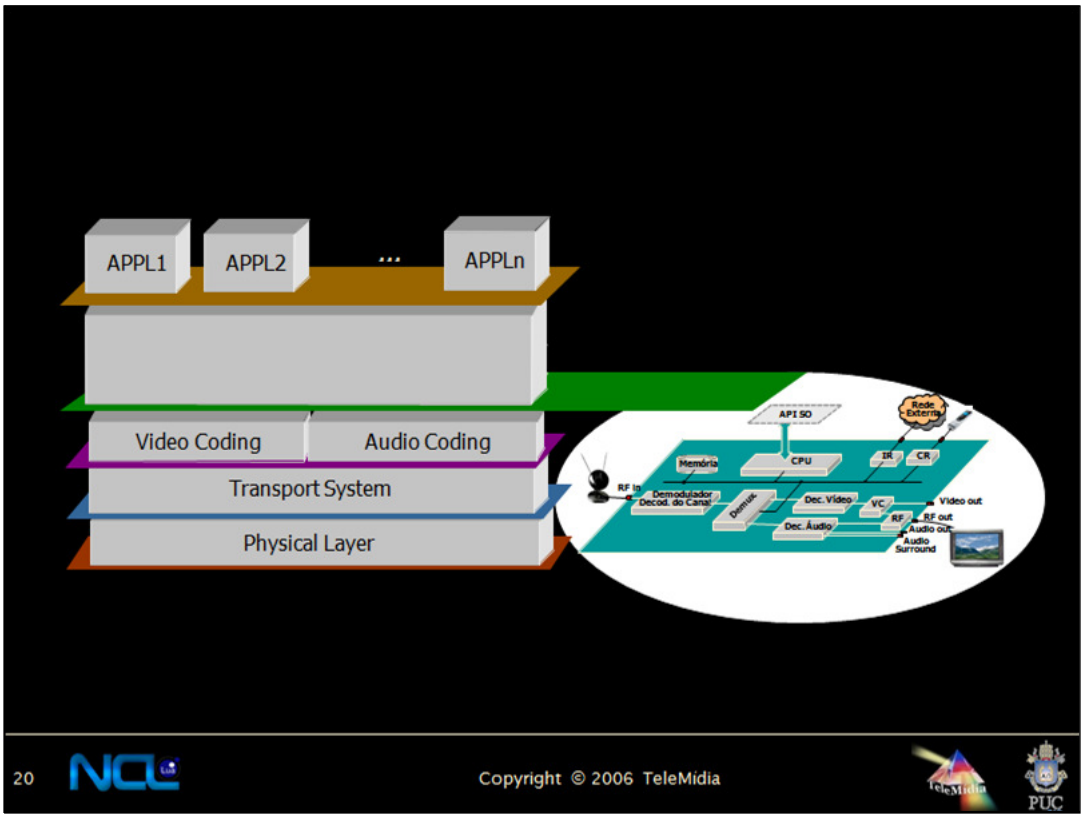


The other extreme is to allow viewers to have access to networks from where they can receive applications' content and to where they can upload data.

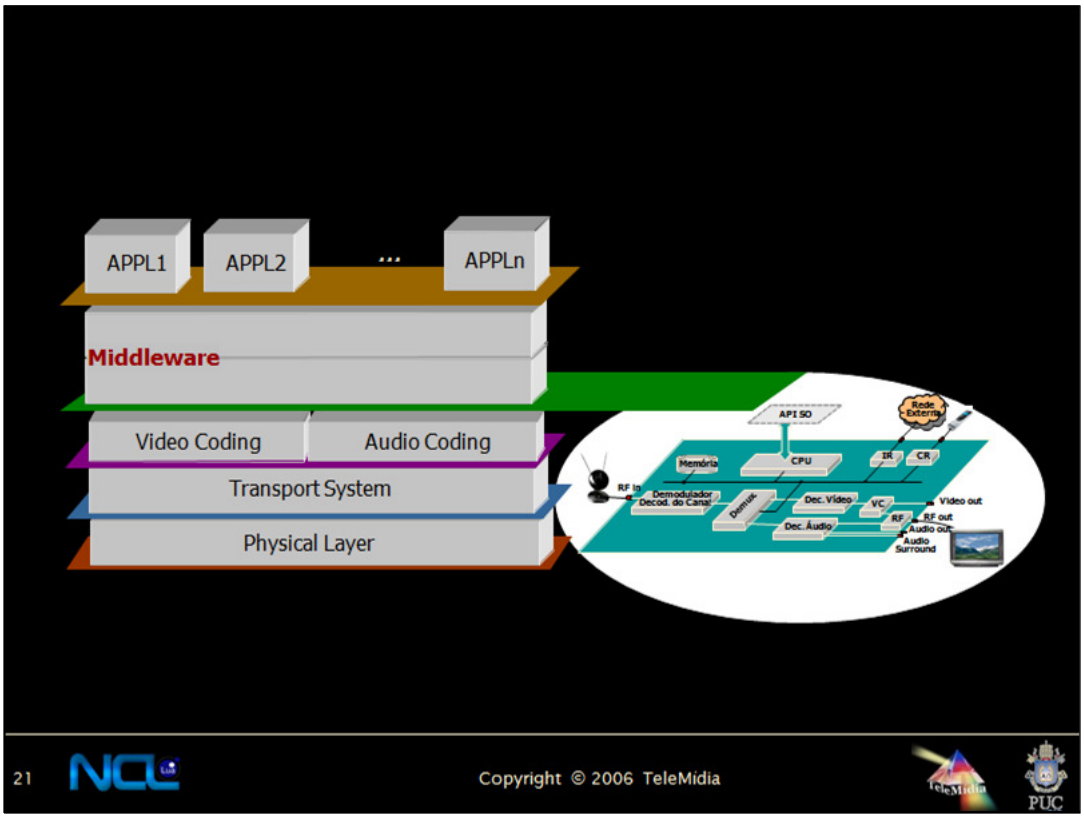


In almost all receivers, the digital reception process (including the demodulation), the data and audiovisual stream demultiplexing, and the audio and video decoding are implemented in hardware.

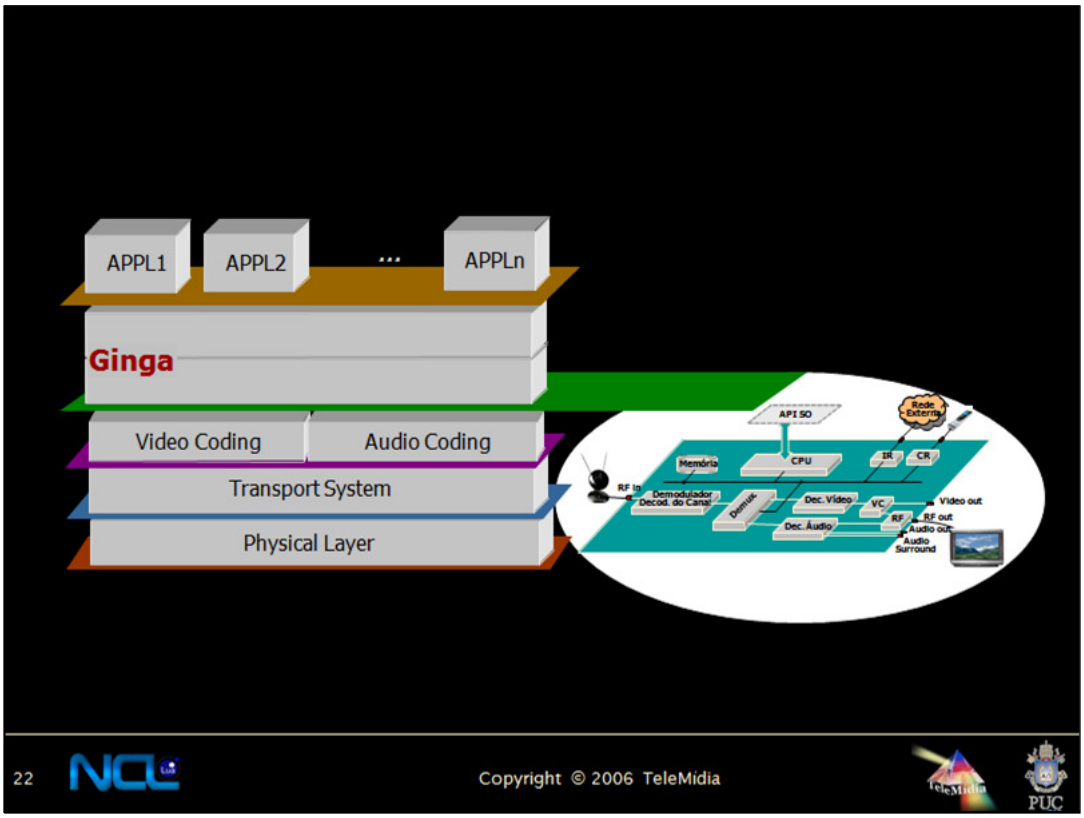
In order to have applications independently from the hardware and operating system platform, and also in order to offer special support to the specification of these applications,



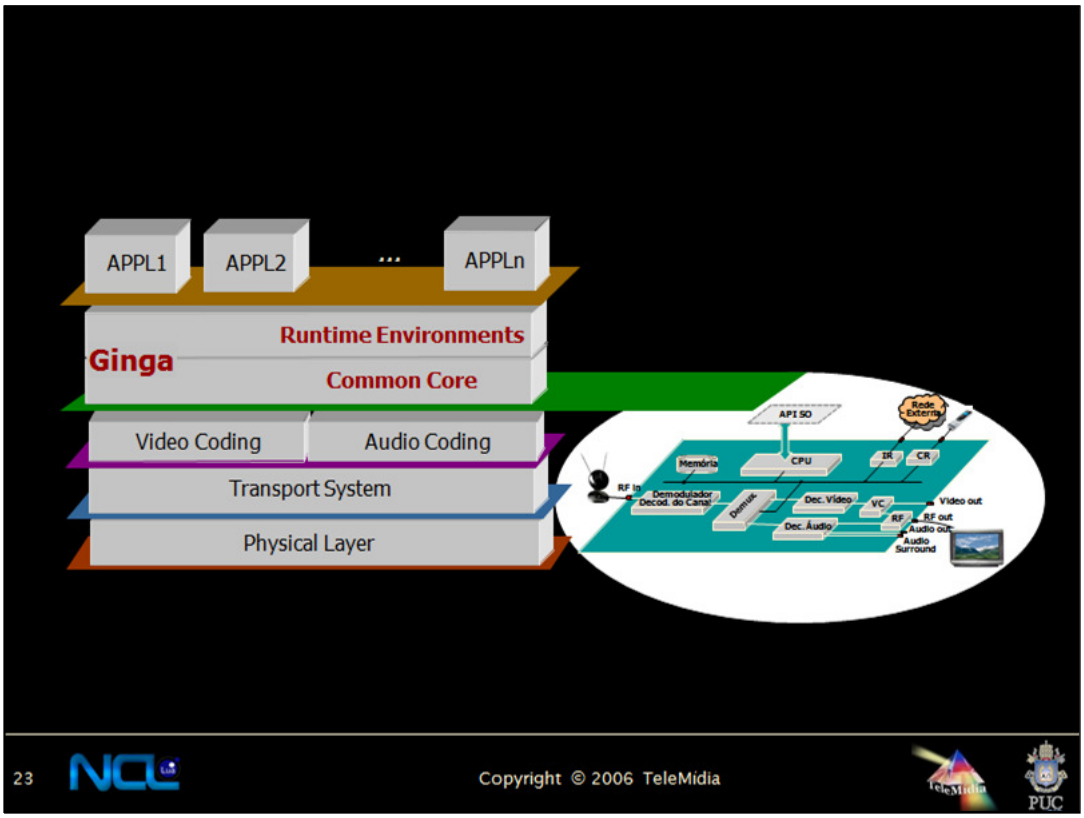
another layer is introduced in the system architecture,



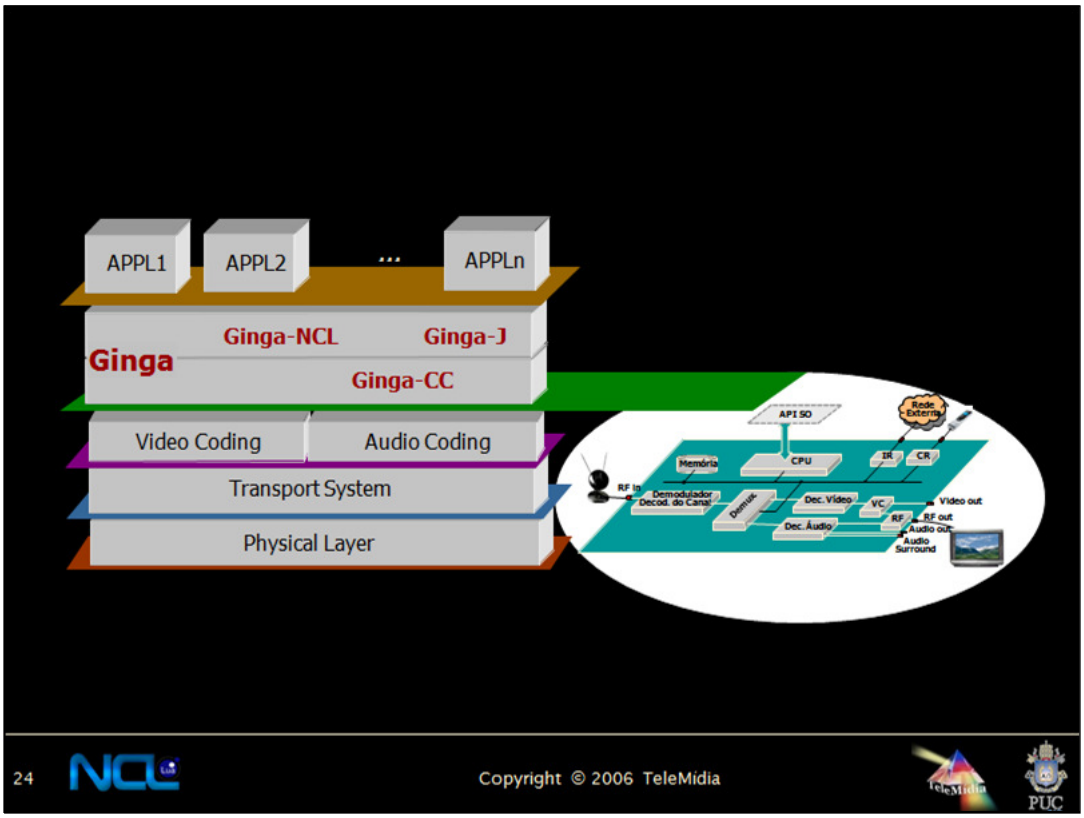
the middleware.



And Ginga is the name of the terrestrial ISDB and the ITU-T H.761 IPTV middleware .



The runtime environment of Ginga is a logical subsystem that processes

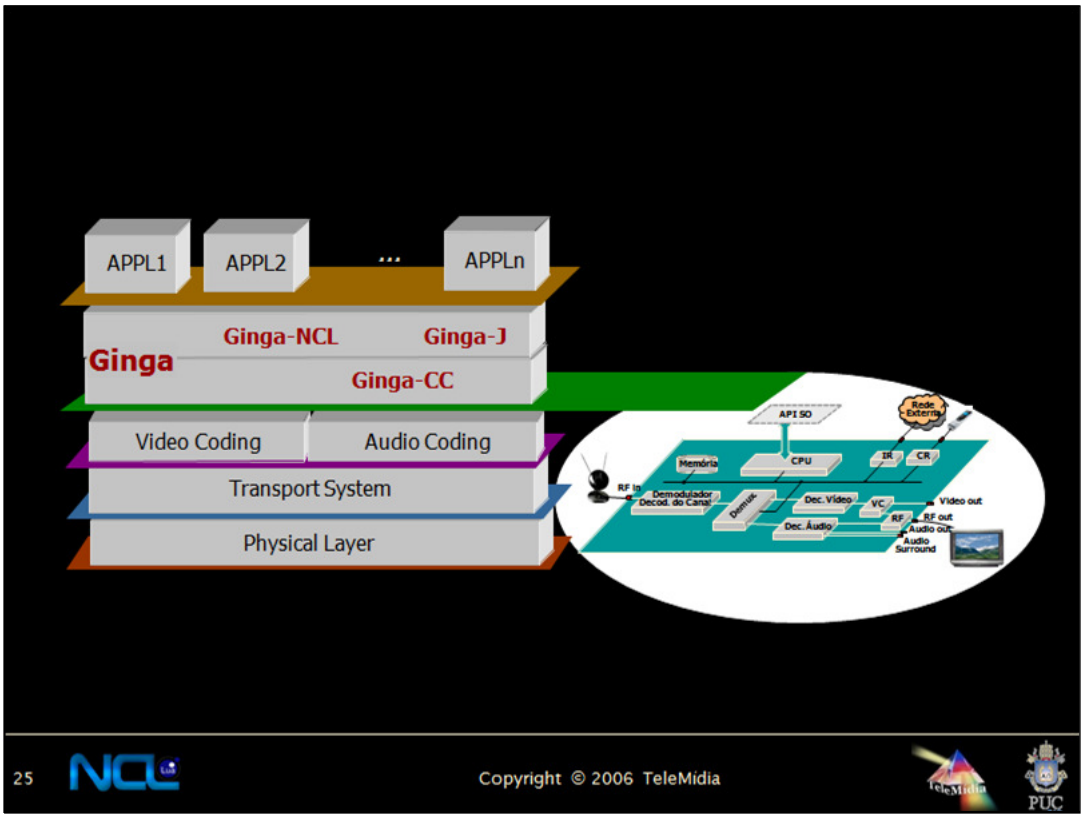


NCL- Lua based applications (the Ginga-NCL declarative environment) and imperative based applications (the Ginga-J imperative environment in the case of ISDB-T).

These environments are implemented using the services of the Ginga Common Core Module.

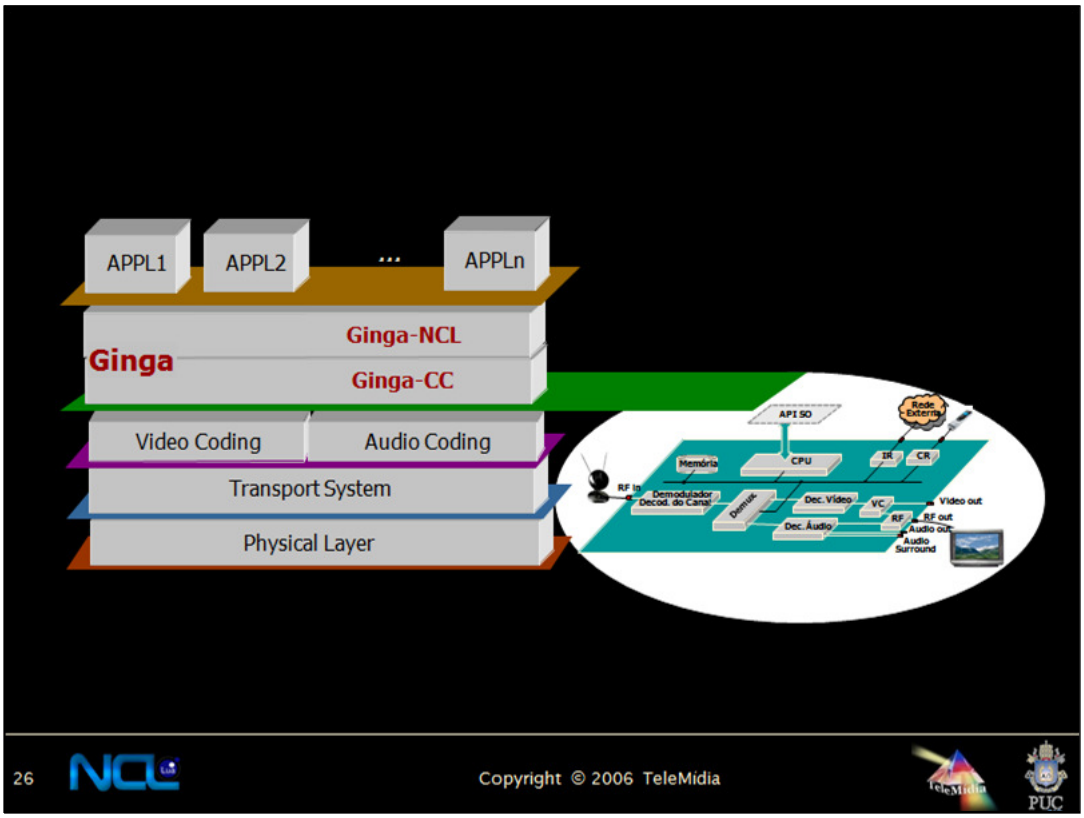
The Common Core is composed of:

- common content decoder/players
- procedures to obtain contents transmitted by broadcasting, multicasting or unicasting.
- the conceptual display graphic model defined by the DTV system;
- And other optional modules: conditional access, network protocol stacks, context manager, update manager, etc.



Today we have two possibilities for Ginga implementation in the Nipo-Brazilian System.

For fixed receivers, like a television set, a set-top box, etc., we shall have both Ginga-NCL and Ginga-J.



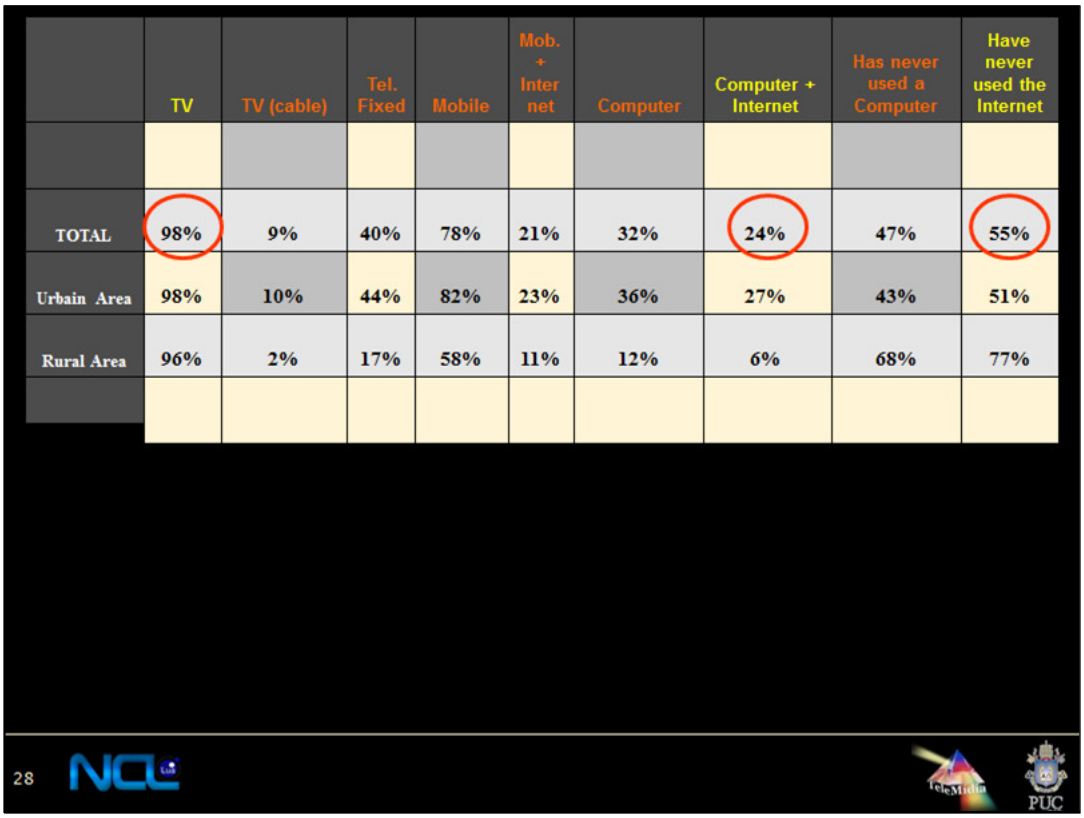
For portable receivers, like a PDA, a mobile phone, etc., only the Ginga-NCL runtime environment is required.

Also, for IPTV services that follow the ITU-T Recommendation, only the Ginga-NCL runtime environment is required.

This brings me to the next topic of this presentation: the middleware requirements.

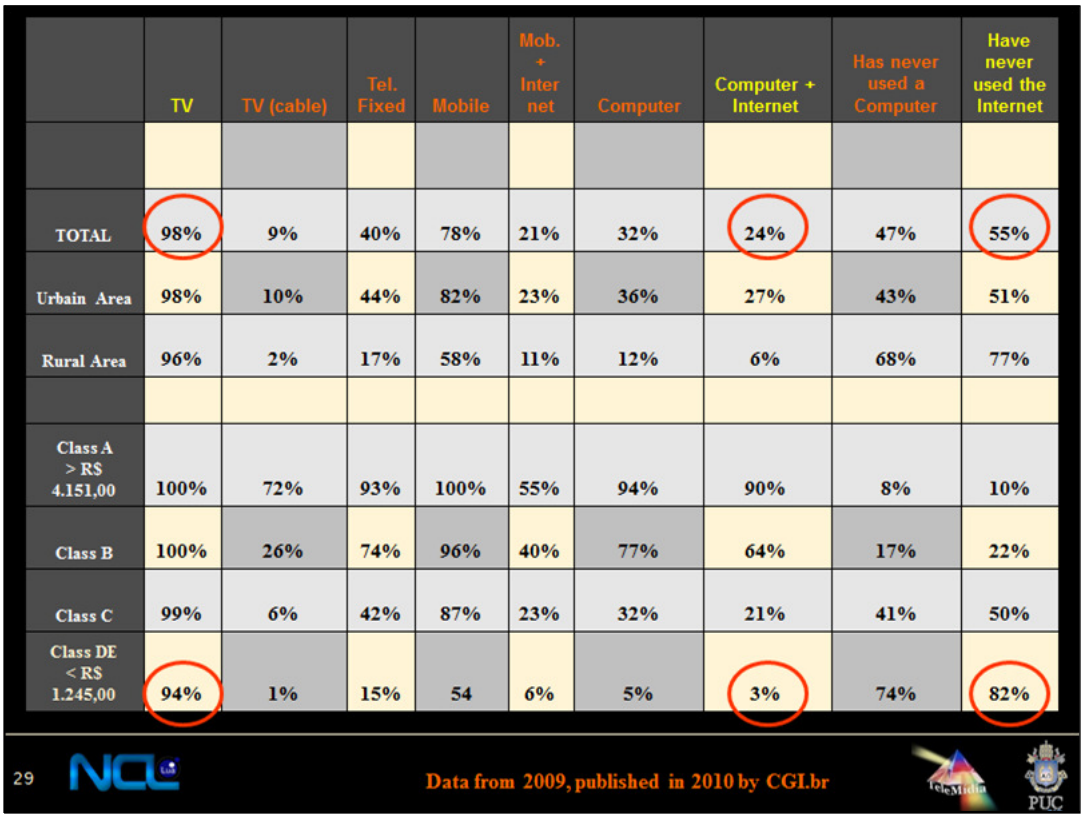
In this new DTV scenario, the viewer gains an active and central role, and this new media cannot be ignored when we talk about social inclusion.

As for example, the Brazilian System has some singular characteristics starting from the users it must give support



Today we have a total of 54,61 millions of households in Brazil.

In 2009, 98 percent of households have a television set, while only approximately 24% have a computer with Internet, and about 55 percent have never accessed the Internet.



	TV	TV (cable)	Tel. Fixed	Mobile	Mob. + Internet	Computer	Computer + Internet	Has never used a Computer	Have never used the Internet
TOTAL	98%	9%	40%	78%	21%	32%	24%	47%	55%
Urbain Area	98%	10%	44%	82%	23%	36%	27%	43%	51%
Rural Area	96%	2%	17%	58%	11%	12%	6%	68%	77%
Class A > R$ 4.151,00	100%	72%	93%	100%	55%	94%	90%	8%	10%
Class B	100%	26%	74%	96%	40%	77%	64%	17%	22%
Class C	99%	6%	42%	87%	23%	32%	21%	41%	50%
Class DE < R$ 1.245,00	94%	1%	15%	54	6%	5%	3%	74%	82%

The disparity is worse in the poorer classes, as it is shown in the slide.

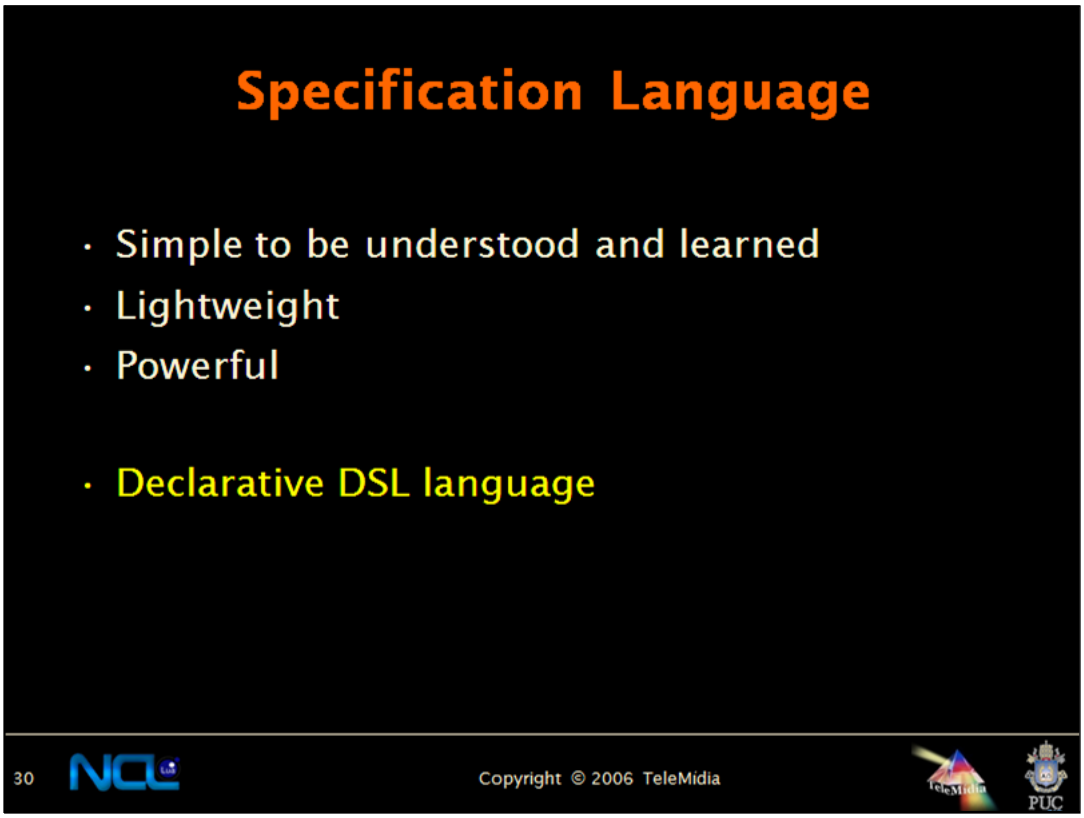
From this statistics we can see that digital TV should play an important complementary role when we talk about social inclusion.

So, at least in the special case of Brazil, the middleware must offer a good support to what we call “Inclusion Applications”, like

T-Learning, T-Government and T-Health.

However, social inclusion is not only offering access to information, but also providing knowledge about how to generate information.

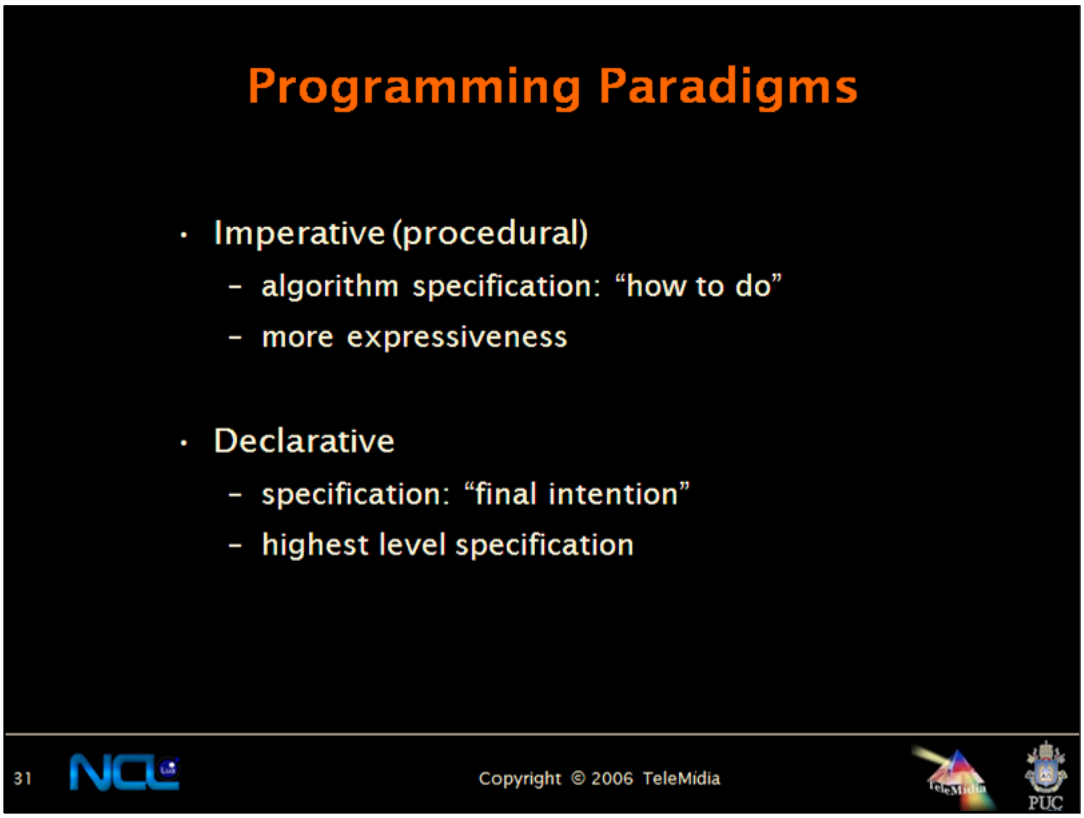
So, a DTV system should offer an easy language to design applications and services.



This language should be simple enough to be understood and learned by common people, since, now, viewers can be part of the content generation process, as in social network applications, for example.

Moreover, this language should be lightweight since its interpreter should run in low cost receivers, and thus, with limited resources.

However, this language should also be powerful enough to not limit the creativity.



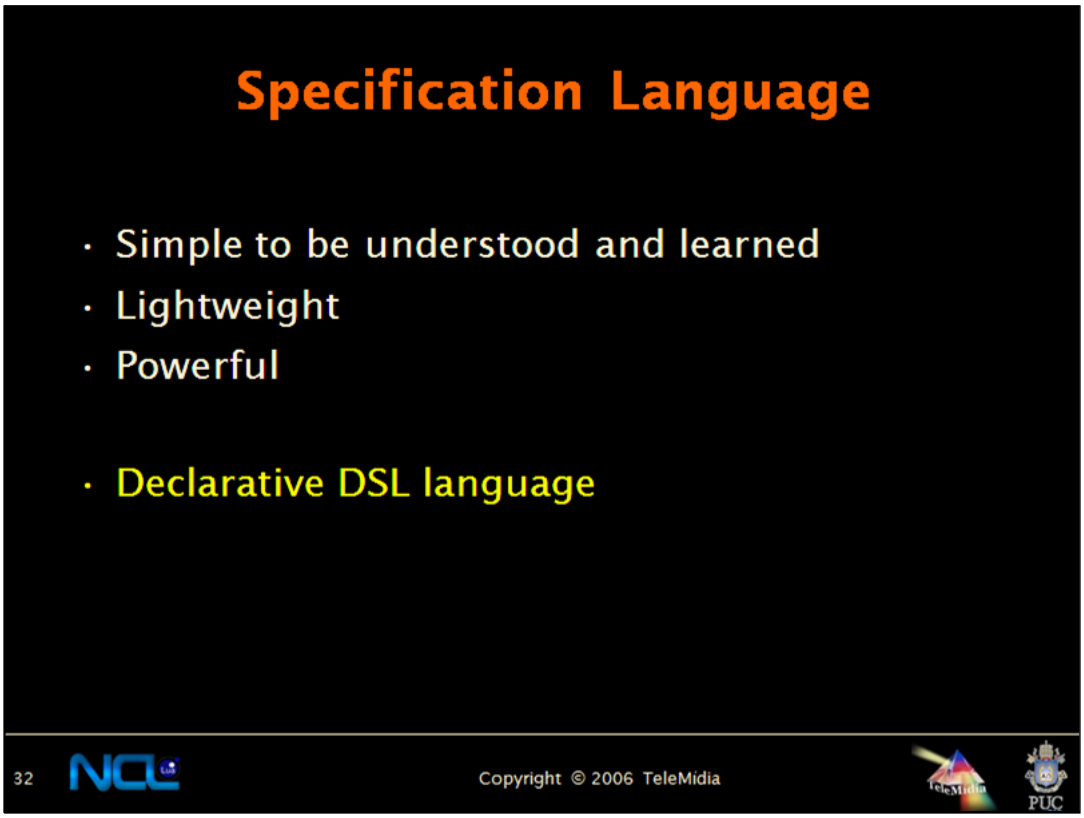
A TV application development can follow different programming paradigms:

The imperative one, the conventional programming style, where applications are decomposed into computation steps that give us an algorithmic specification,

and

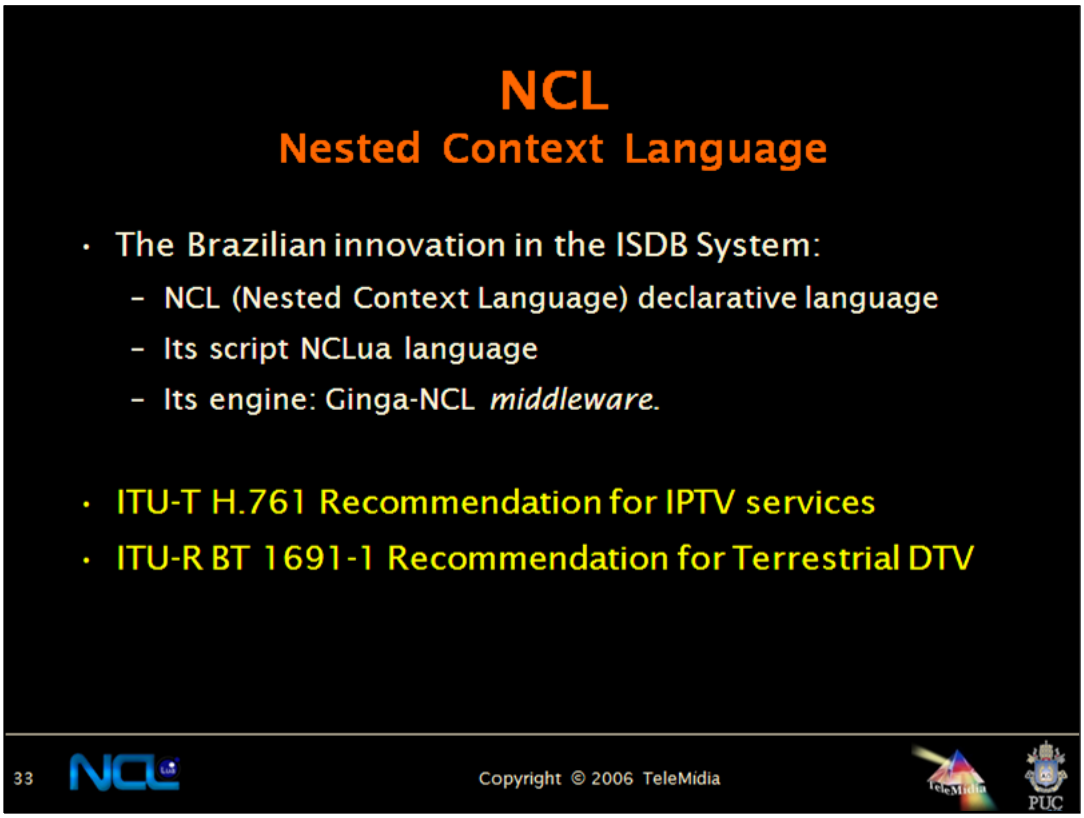
the declarative one, which emphasizes the declarative description of a problem, rather than its decomposition into an algorithmic implementation.

Such declarative descriptions are closer to a high level specification, and, thus, easier to be designed than the procedural ones, which usually requires a programming expert.



All previously mentioned requirements lead to choose a specification language as close as possible to the way human beings express their ideas, and their creativeness.

It should be a declarative language in which the complex algorithmic steps to accomplish a task are left to the machine and are not under user responsibility. A declarative domain specific language (DSL) for TV application conceptions.



These were the requirements that guided the NCL design.

But an issue still remains: what should be the tasks this declarative DSL should focus?

Looking at DTV applications, we see that they are composed of scenes, as usual. However, different from an analog TV, a scene is composed not only of the main video and the main audio,

But, in addition, of other media objects

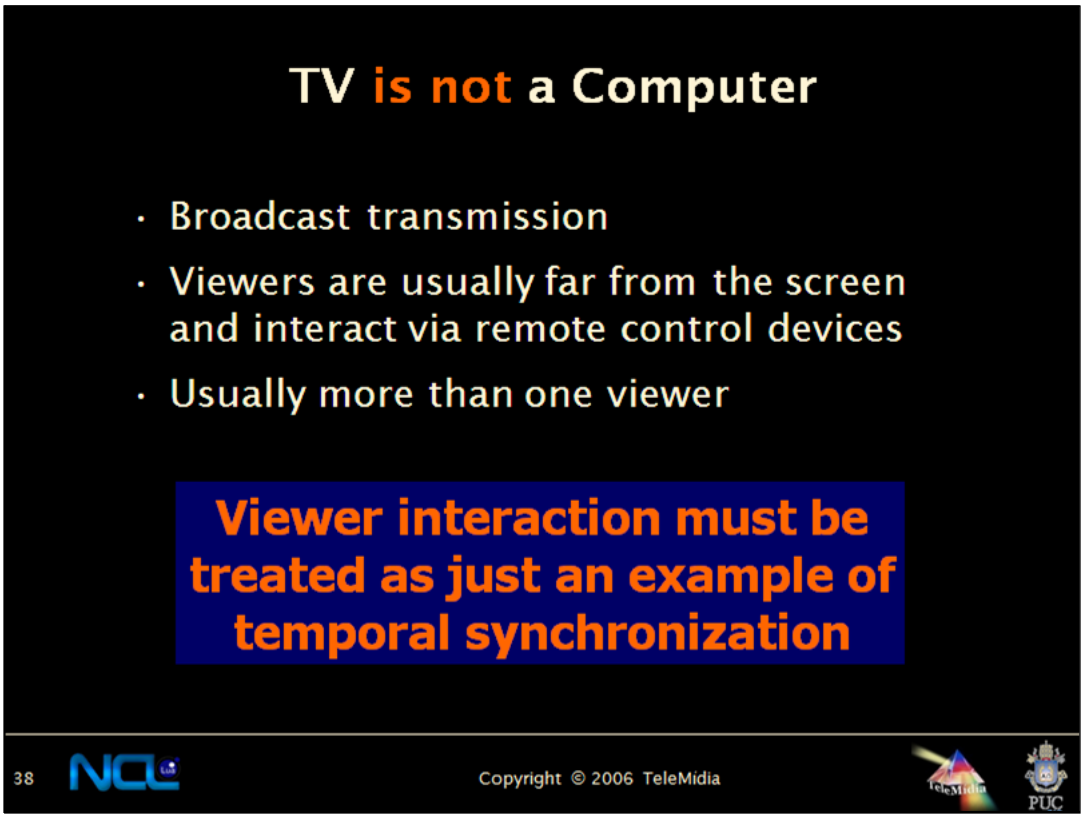
(images, text, other videos and audios) that are synchronized in time and space.

Temporal synchronization is an important key issue.

Different from the analog TV, in a DTV, scene changes can be non-sequential and can even depend on viewer interventions.

However, viewer interactions should not happen so frequently, in comparison with applications designed for computers.

TV is not a computer.

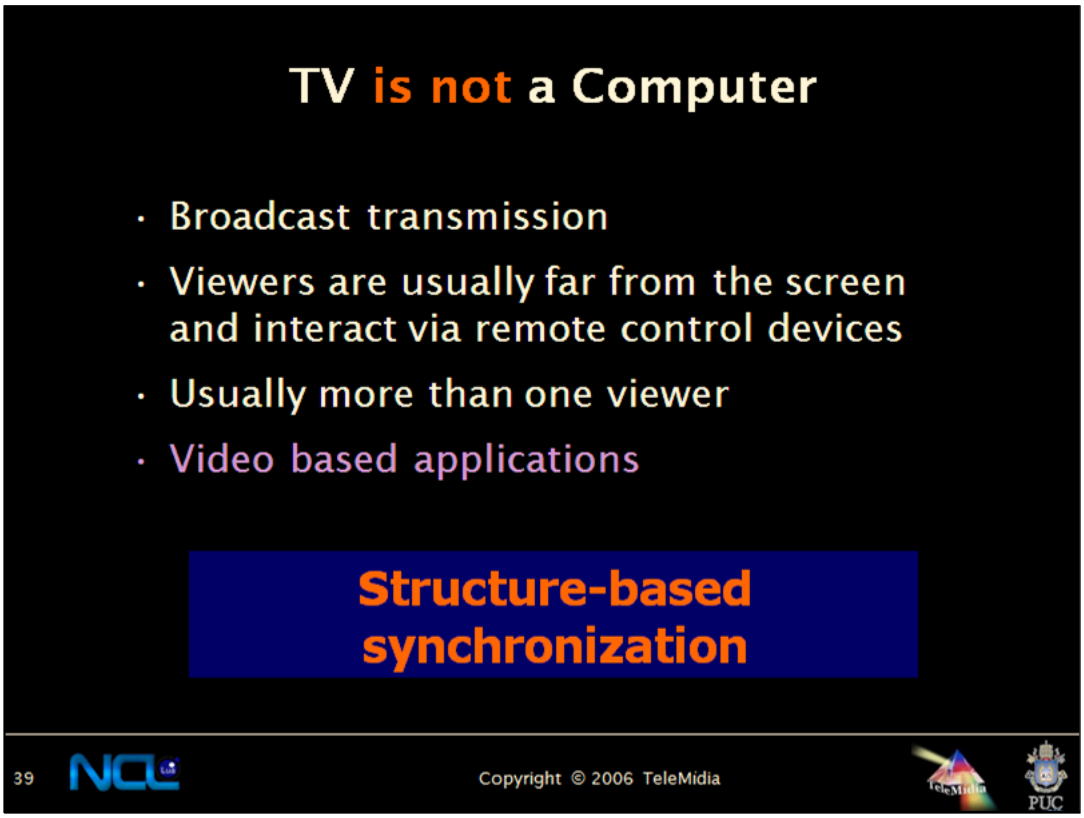


Since most contents will be transmitted by broadcast, for one place to many, they are not custom-made;

a viewer usually stands not so close to the screen and uses a poor interface device;

moreover, frequent interactions can annoy other viewers that are watching the same program at the same place.

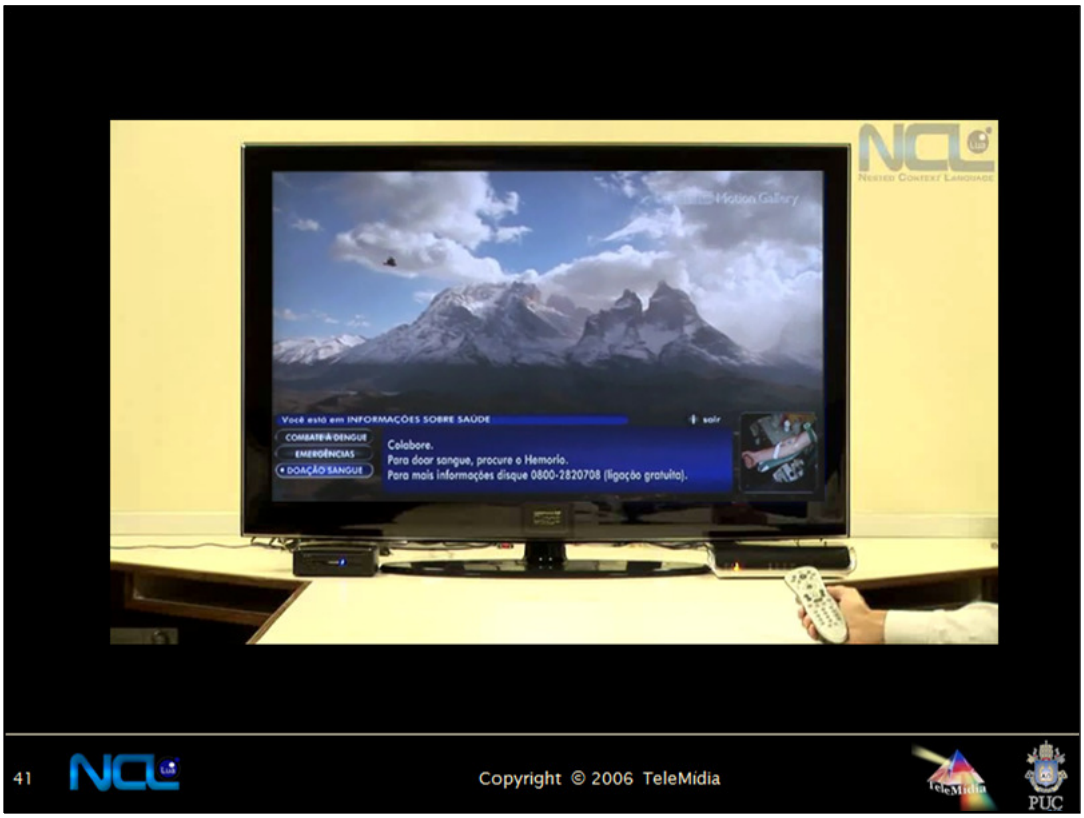
Thus, in digital TV applications, it is the temporal and spatial synchronization among media assets, in its broad sense, that should have a good support. Viewer interactions should be treated as just a particular case of temporal synchronization: the one that occurs at a time a viewer selects an object.



In addition, many digital TV applications will be based on the main video stream semantics, in contrast with Web navigations, which are based much more on text.

Therefore, the specification of synchronization relationships should not be embedded in media contents. Structure based instead of media based synchronization should be encouraged.

Let us now give a break to see some demos that illustrates some of the mentioned concepts.

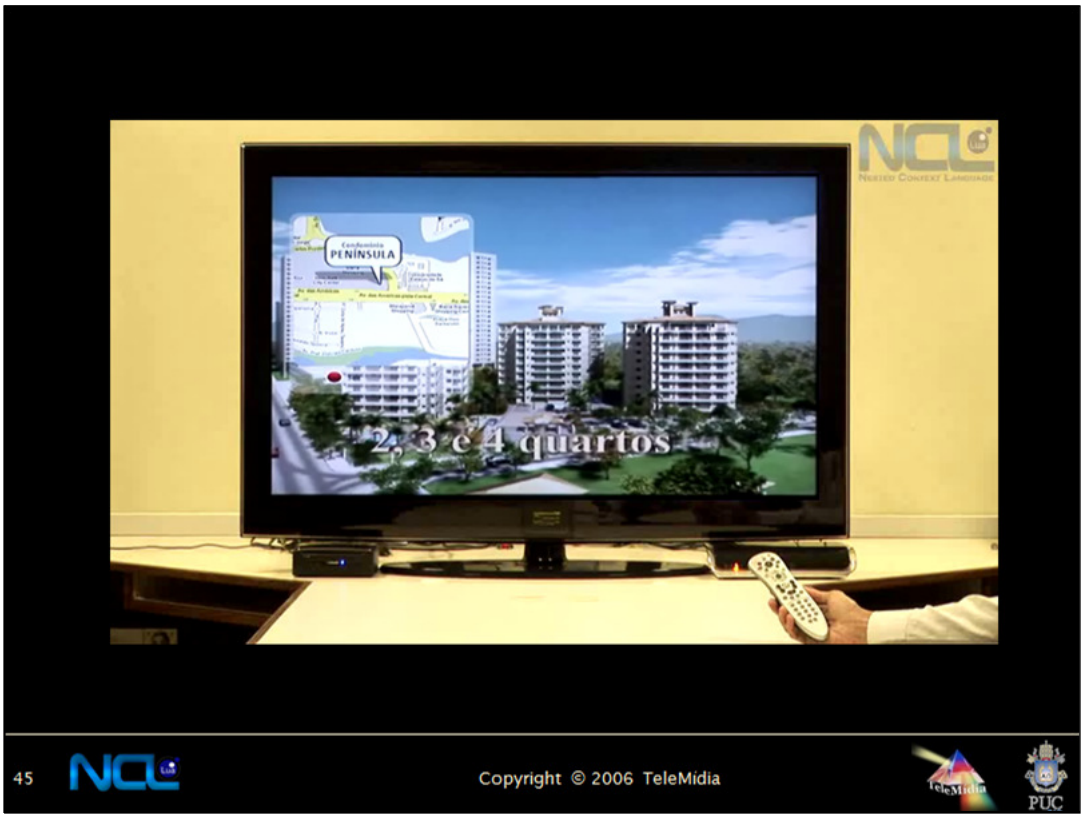


This example shows a health application that can be presented independly from the video program in exhibition.

This example shows a car race in which additional information can be obtained under viewer demand.

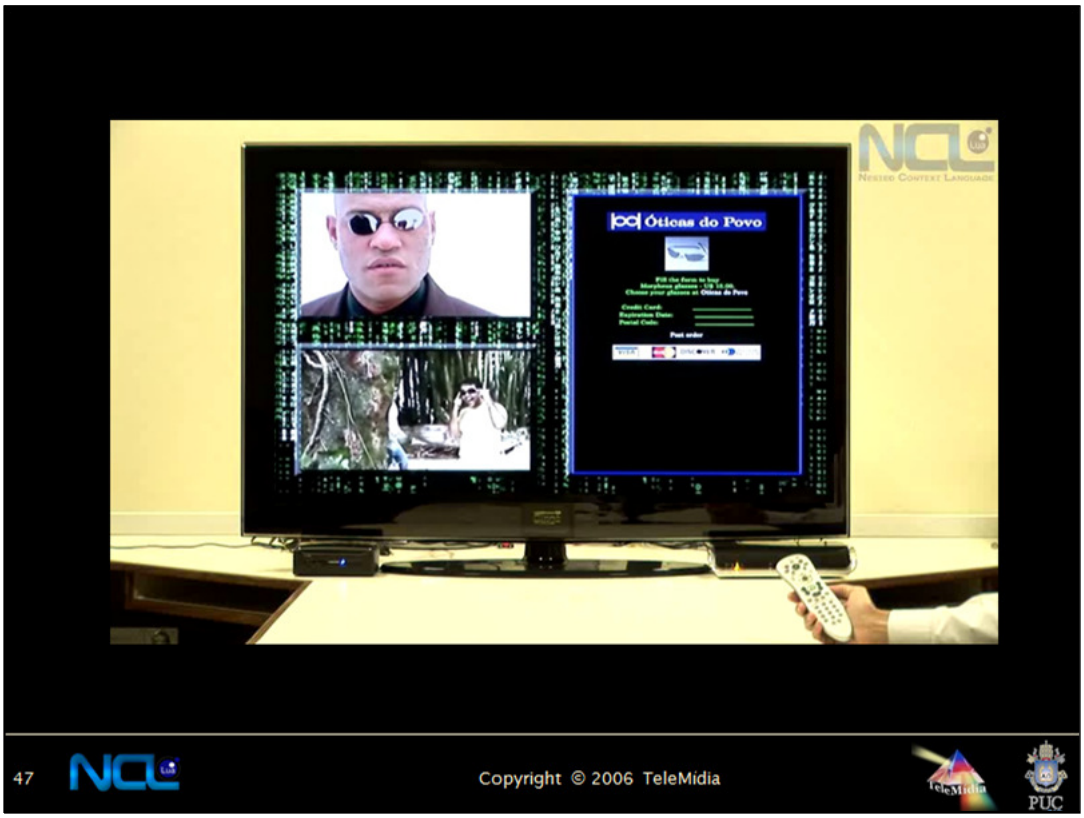
This example shows another health application. However, in this live show, viewer interventions are guided by the main video.

This example is part of the previous one (live show), in which data entered by viewers are used (computed) to give a personalized information return.



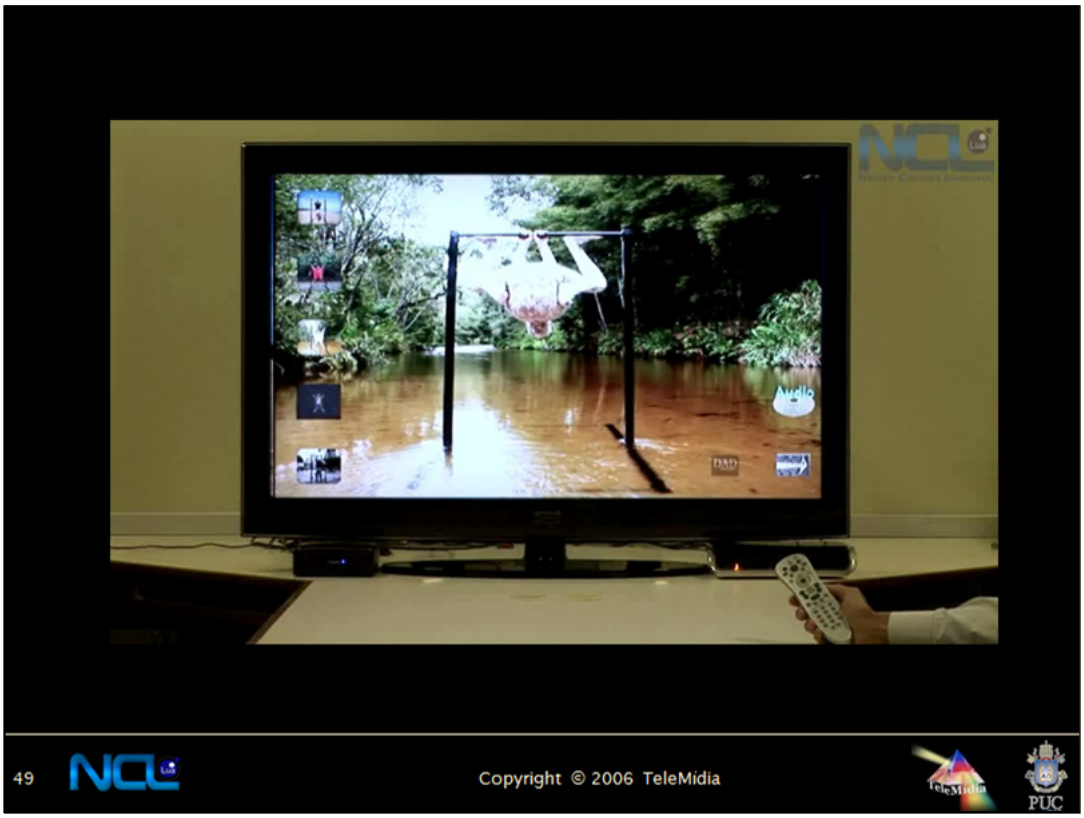
This example illustrates the interactivity in merchandizes.

This example illustrates the use of the return channel (another network) to download and upload viewer information.



Another example illustrating the use of the return channel. We will come back to this example later to illustrate the use of multiple exhibition devices.

This example explores the use of multicamera transmissions. In this case, one for each orchestra suit.

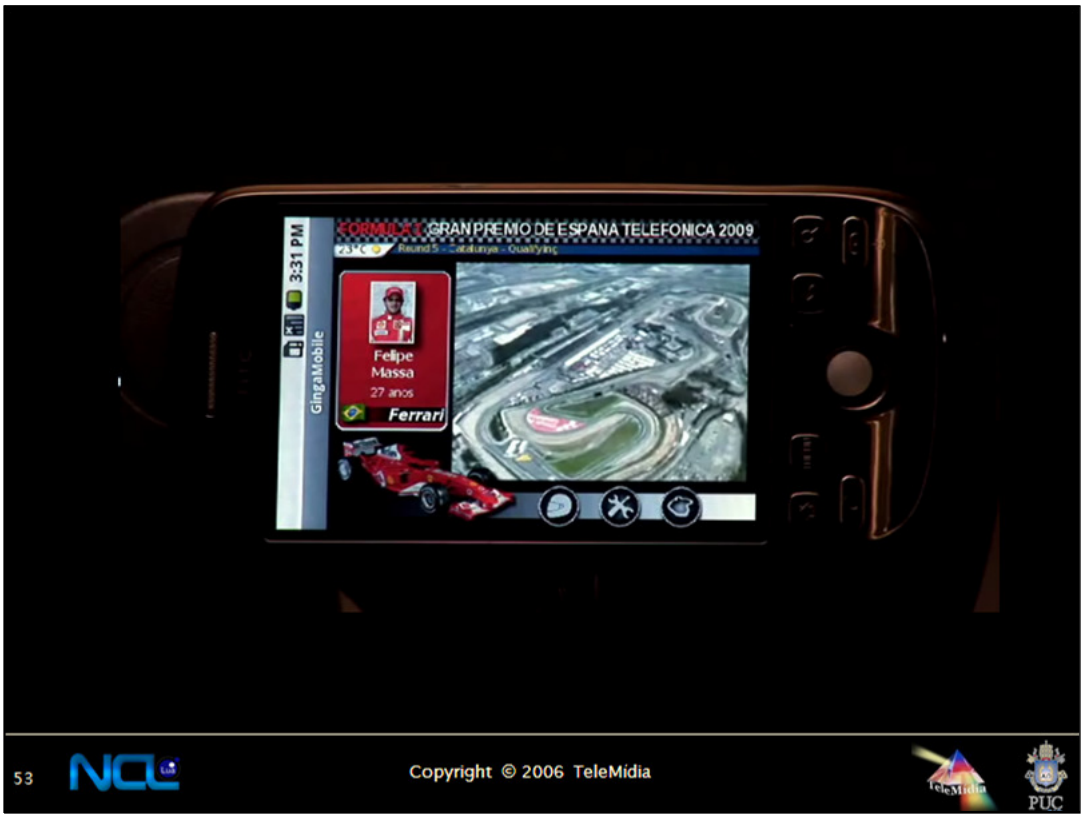


Another example of multicamera transmissions.

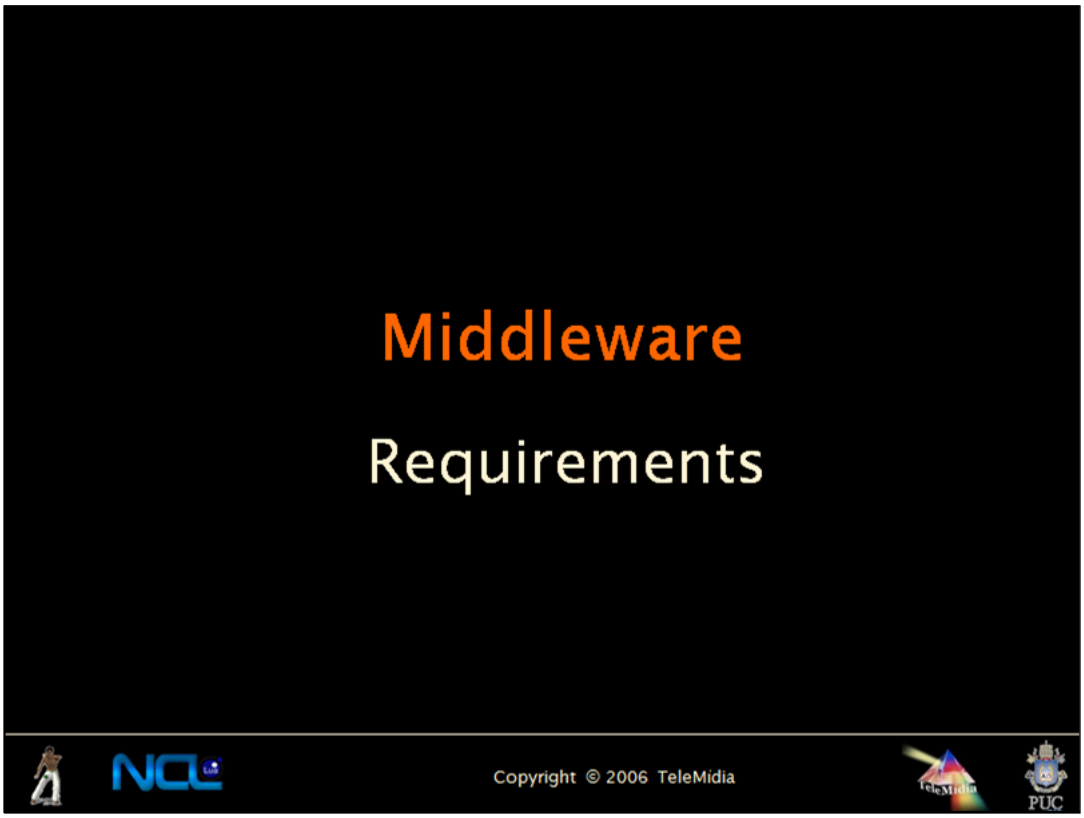
This example illustrates a game that is synchronized with the main video exhibition.

Another example of a game, but without any relationship with the video in exhibition.

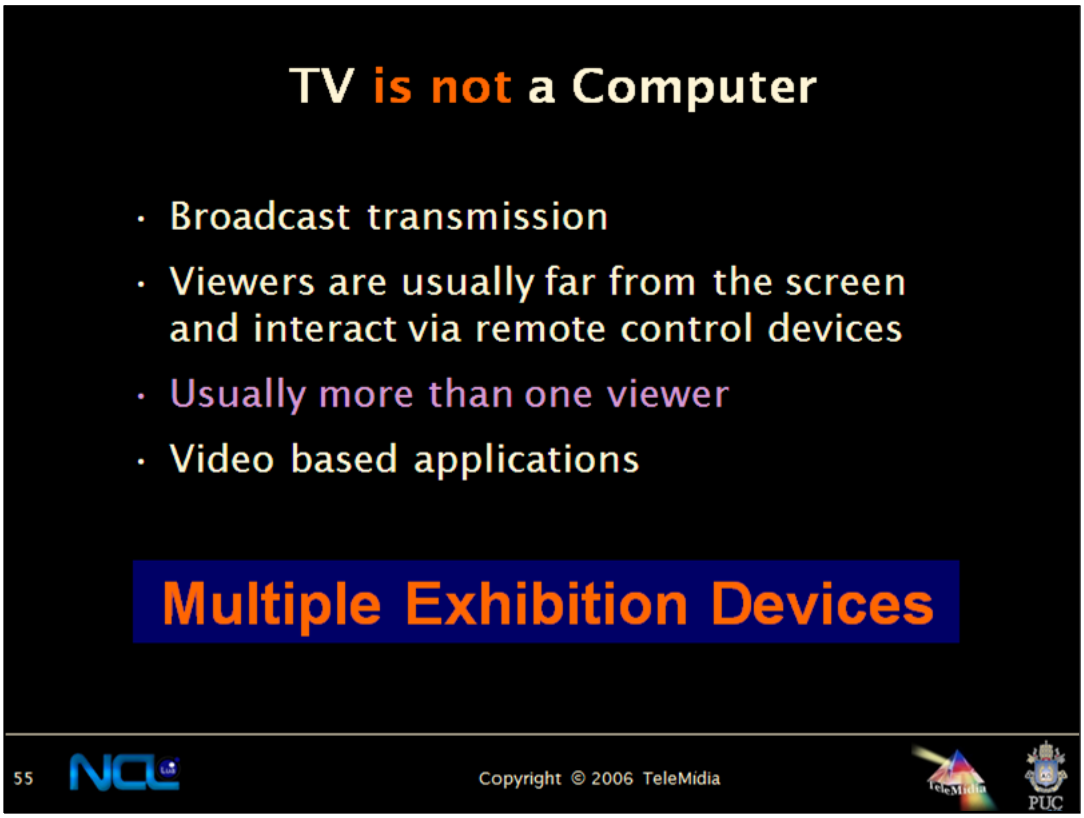
Some examples for portable devices.



One more example for portable device.



Turning back to the middleware requirements,



in order to avoid that a viewer interaction annoys other viewers watching the same program at the same place,

multiple exhibition devices should be used.

Let us take back our soccer game example. During the game, a controversial event can happen.

An icon can then appear to allow for reviewing the event.

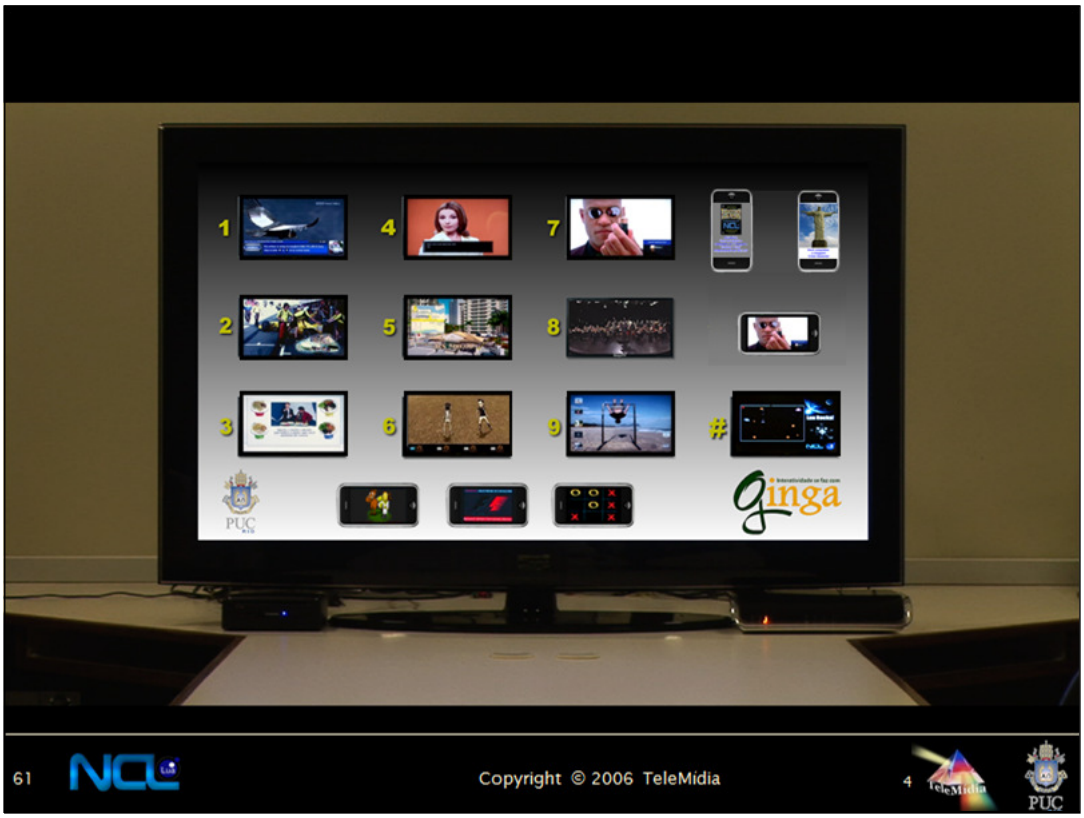
If this icon is selected,

the main video shrinks, the previous video event is repeated in another screen region, together with the graphical animation of the event, in order to allow for a viewer to check the occurrence.

The scene may be repeated as much time the viewer wants.

Let us take the same example, except that now a viewer can interact via a secondary exhibition device, a PDA or a remote control with a small screen

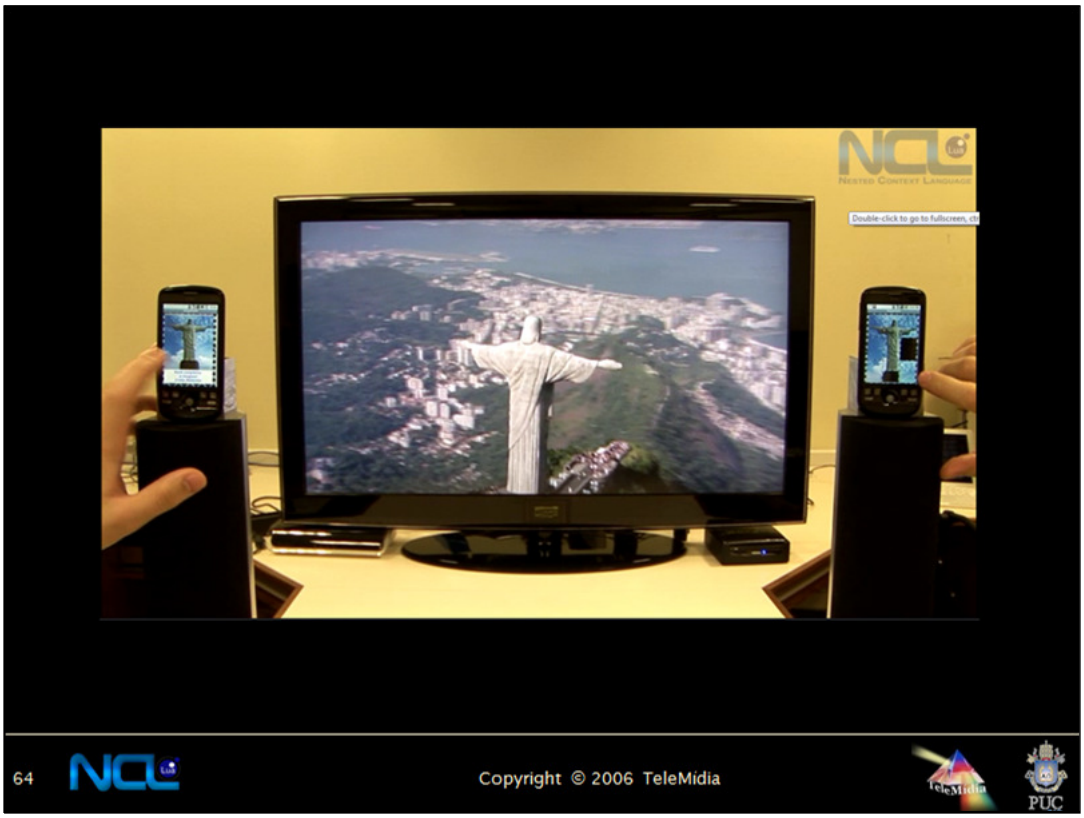
In this new scenario, several independent viewers can interact. The result of an interaction appears independently in each secondary device, without annoying other viewers.



Let us see some Demos to illustrate the exhibition concept on multiple devices.

This is the same example previously presented, but now with multiple devices.

This example shows extra information, about the car race, appearing on the secondary devices.



This example shows a puzzle game, in which each image step, appearing in the secondary device for playing, is synchronized with the main video on the TV set.

Another game example, synchronized with the horse race in exhibition on the TV.

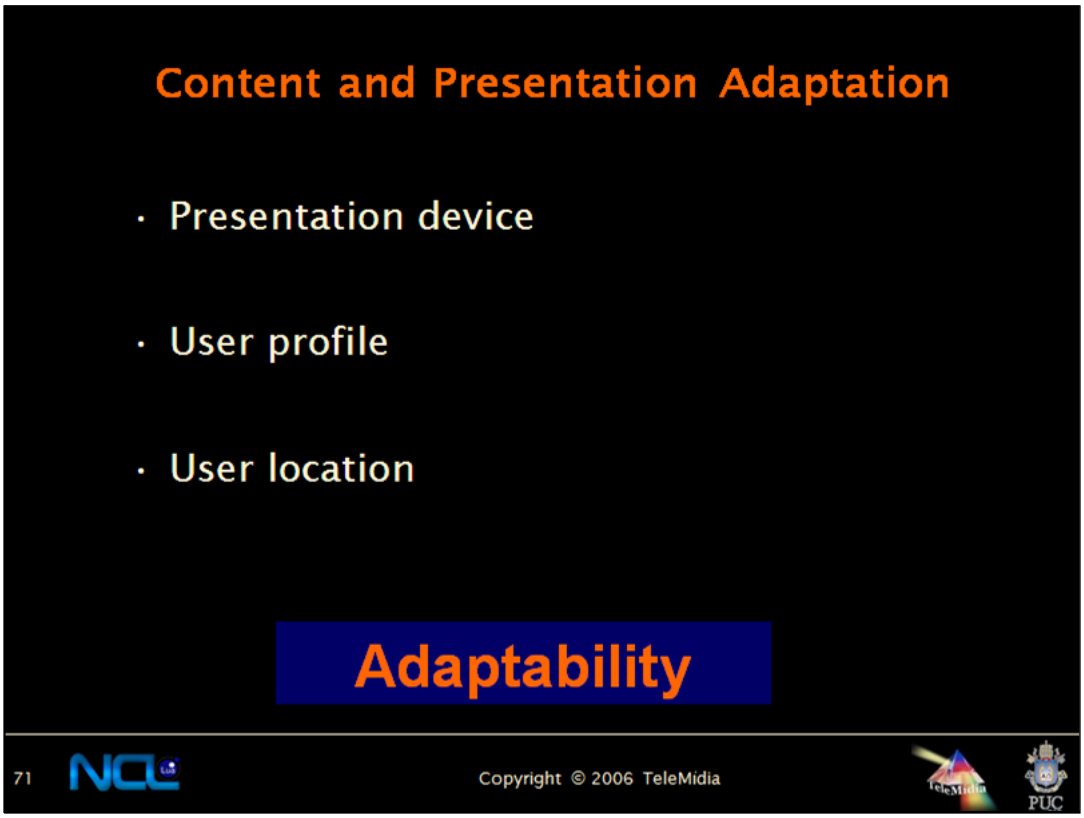
Individual receivers can also interact among themselves given rise to new social TV applications, as exemplified by this “tic tac toe” game.

As still another design requirement, note that several DTV applications will be written to adapt their contents or the way their contents will be presented.

In this slide, a different advertisement appears depending on the viewer location and age.

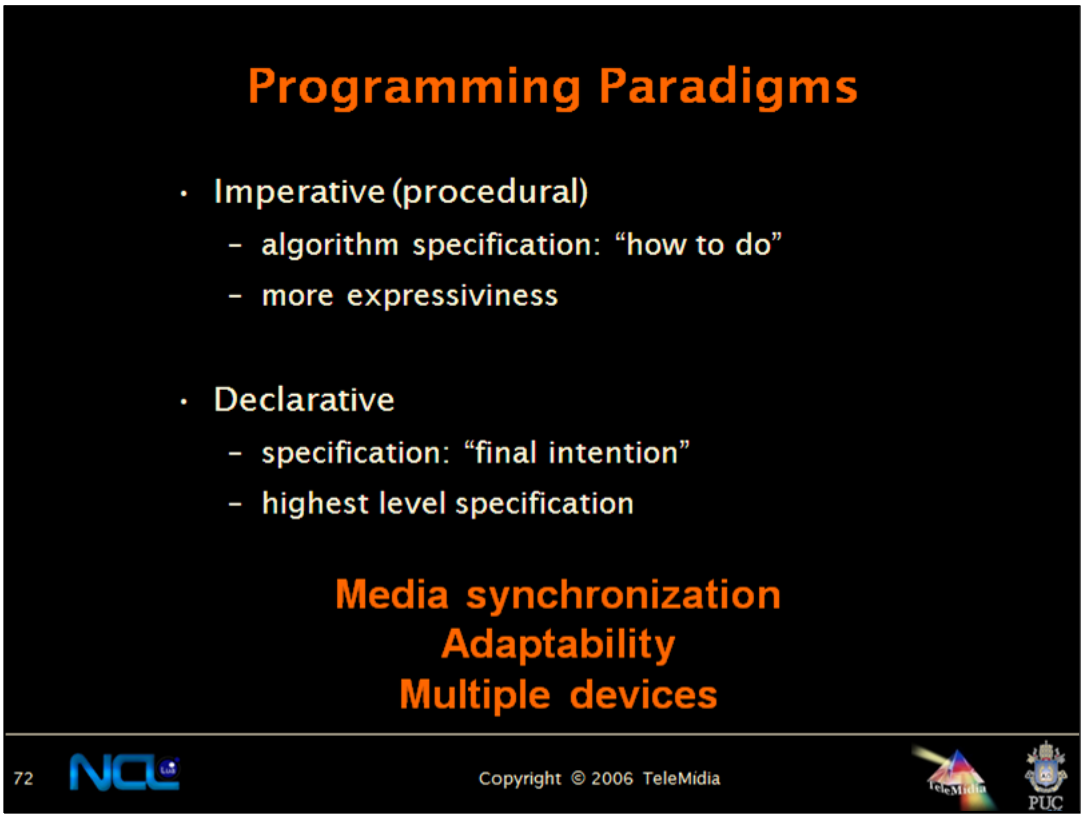
As, for example, an Argentine beer (Quilmes) merchandize, for an adult in Argentina.

And, alternatively, as a Brazilian soft drink (guarana) merchandize, for a child in Brazil.



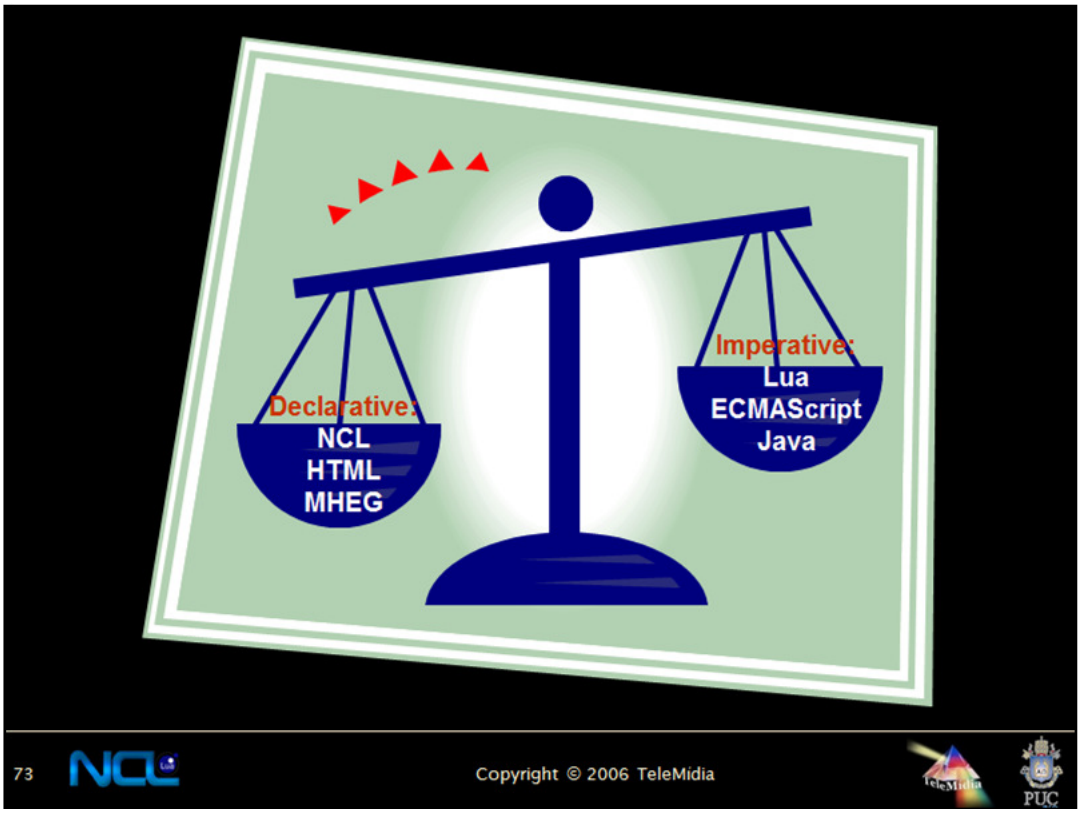
Content and content-presentation adaptation may depend on the type of the exhibition device; on the user profile; and on the user location.

In short, it is also important to provide a good support for adaptability.

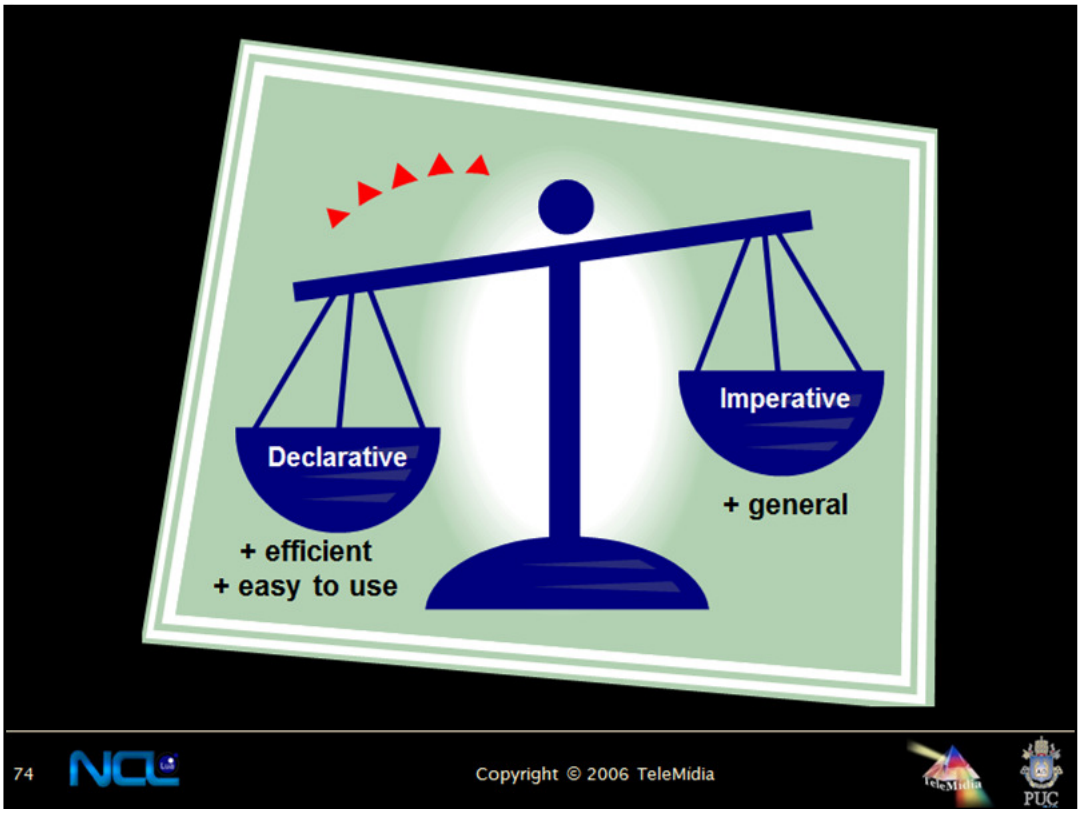


Now we can answer the question about which should be the focuses of a declarative language supported by a middleware.

From our "middleware requirements" discussion, the focus on media synchronization, in its general aspect, on adaptability, and on multiple exhibition device support is probably the answer.



Since declarative languages are usually not general purpose, in some way, every middleware gives support both to declarative and to imperative specifications.

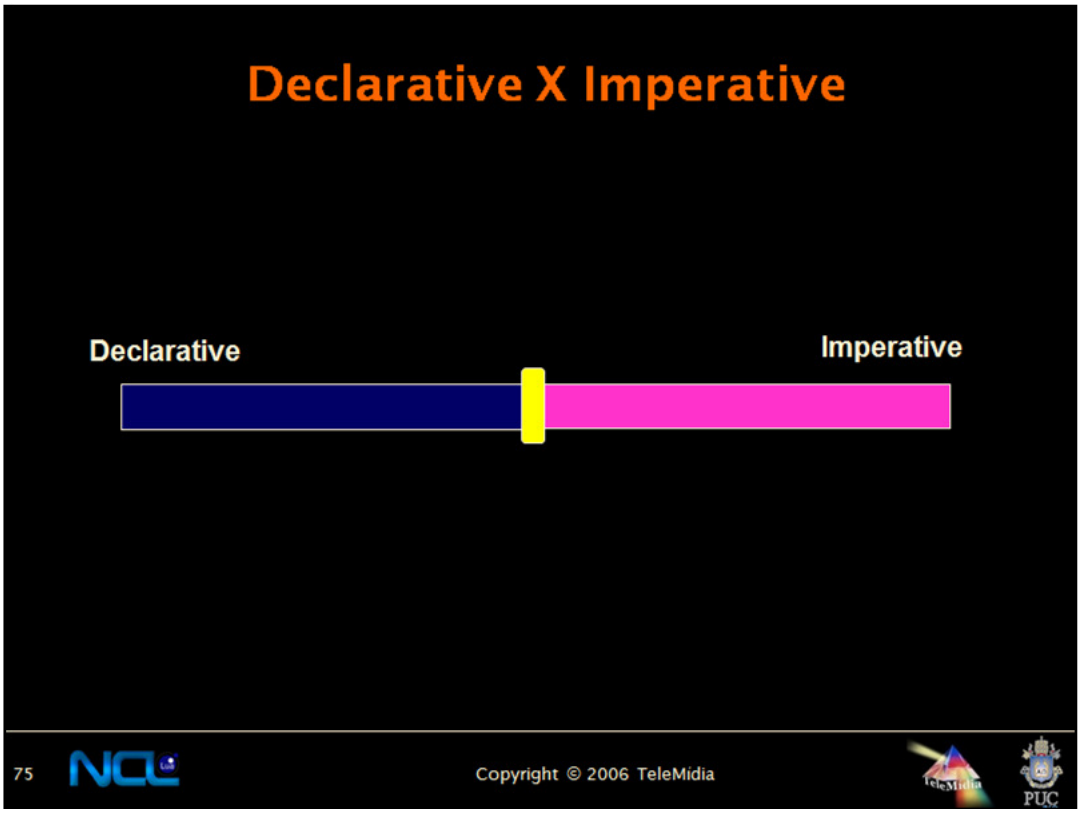


Each paradigm has its advantage, as we have already discussed.

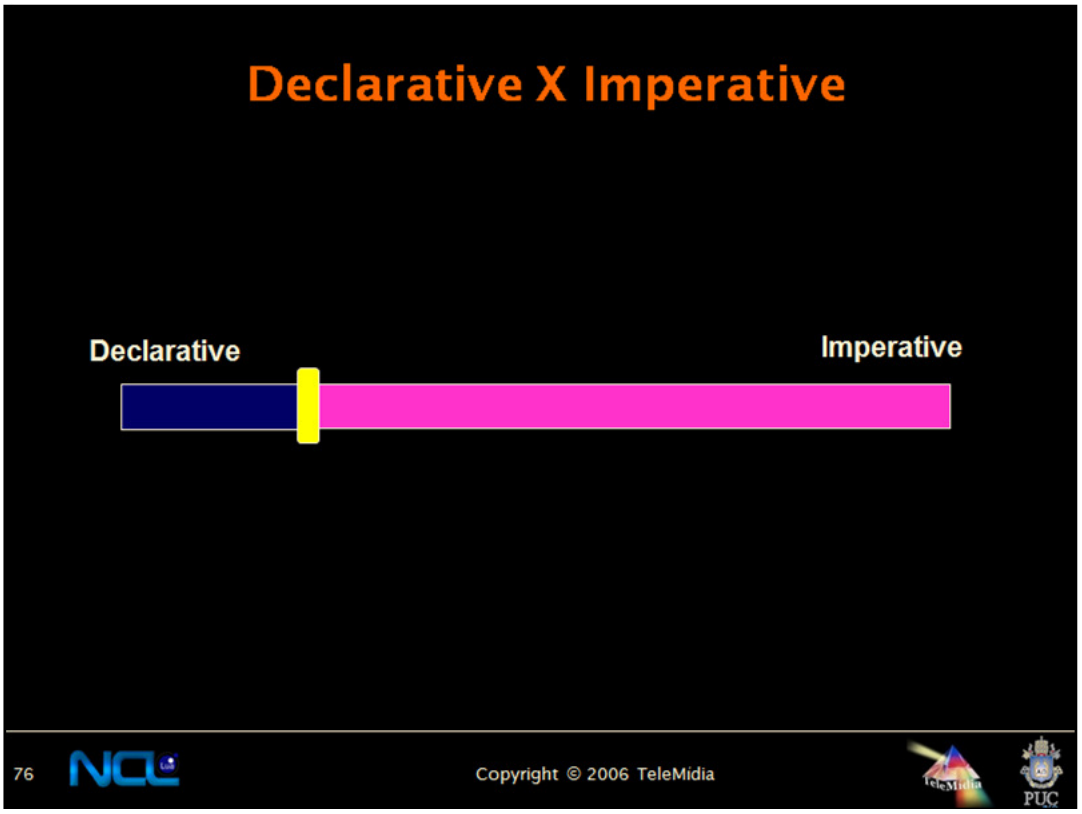
The declarative paradigm is in general more efficient, less error-prone, and easier to program, but only when the task can be solved declaratively. However, some tasks can be difficult or impossible to be solved using a declarative domain specific language.

In short: as more tasks can be accomplished declaratively it is better, but not everything can be done declaratively.

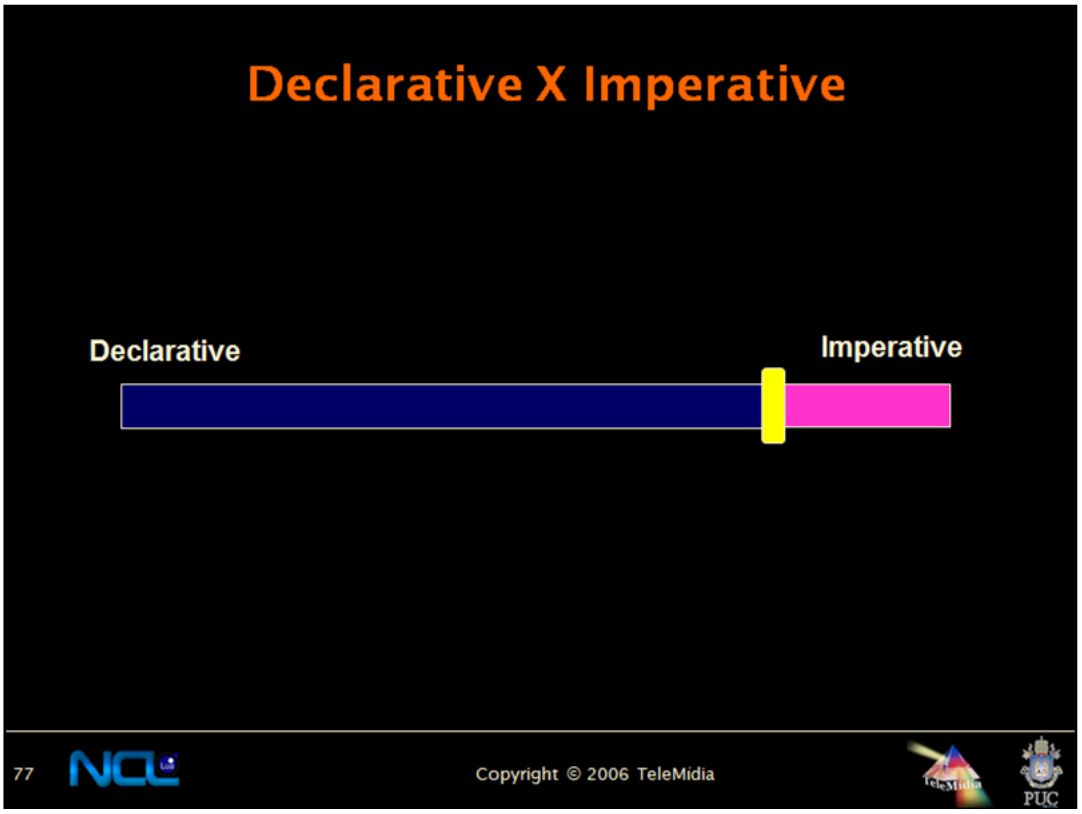
So, **the declarative DSL language should be as much expressive as possible**.



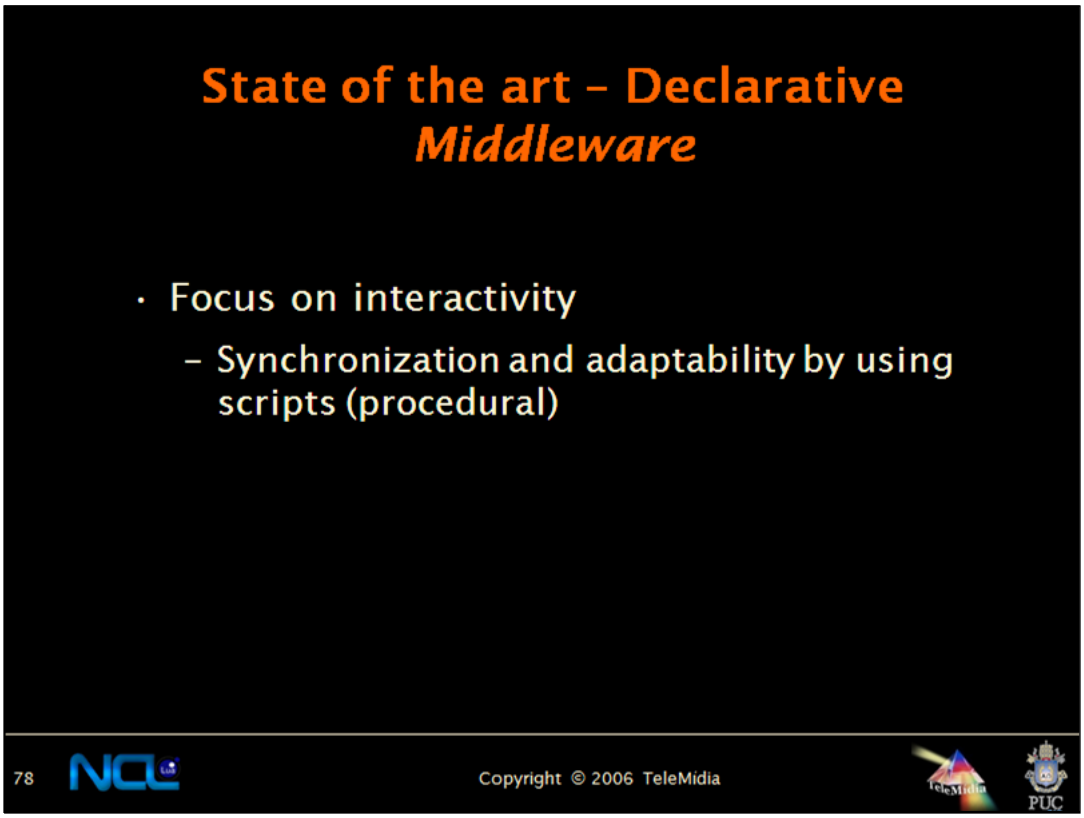
Therefore, in the ruler of tasks that can be declaratively done,



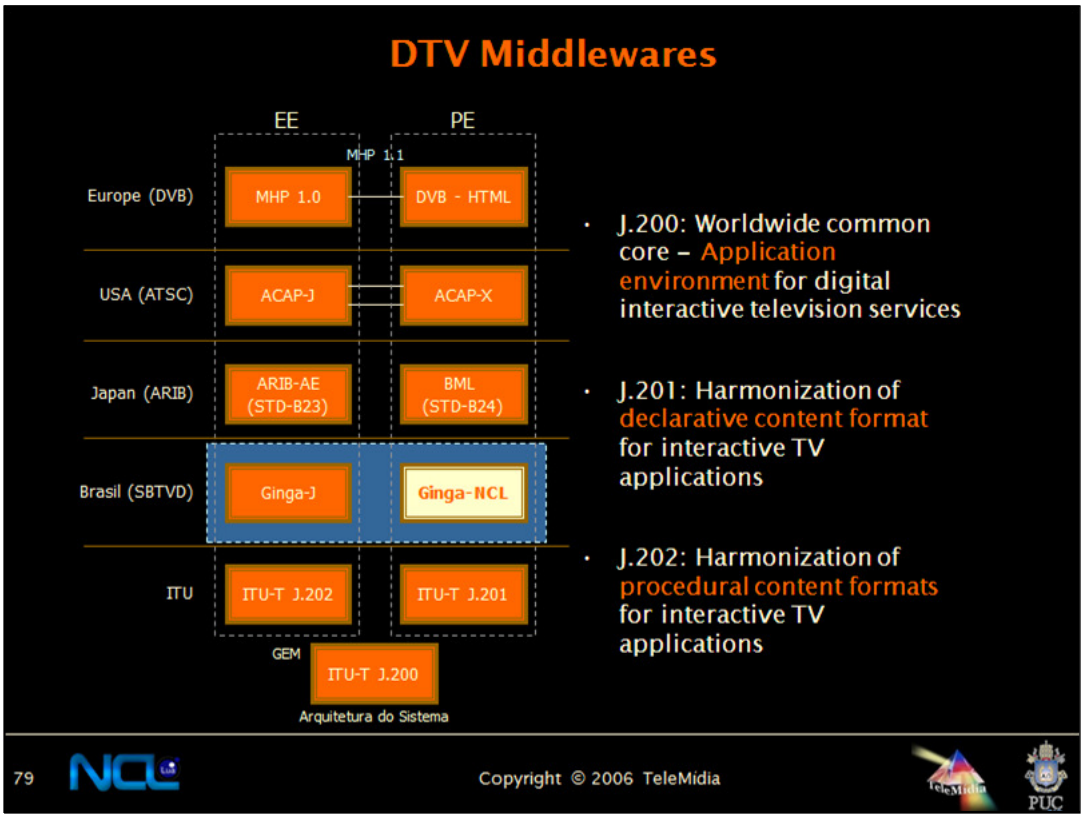
the best is to have a declarative language



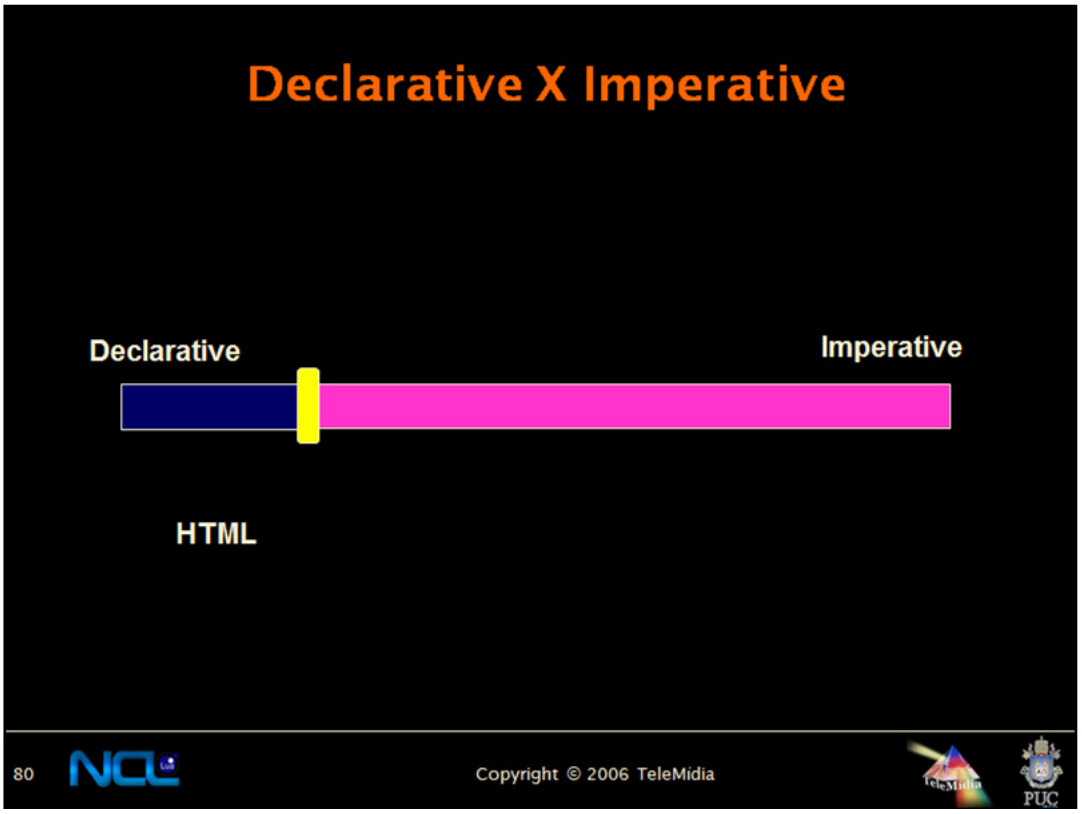
that can solve as more tasks as possible, as shown in the slide.



How can the usual DTV declarative middleware be positioned in this ruler?



The ACAP-X, HBBTV, DVB-HTML, LIME and BML proposals are all based on the XHTML language.

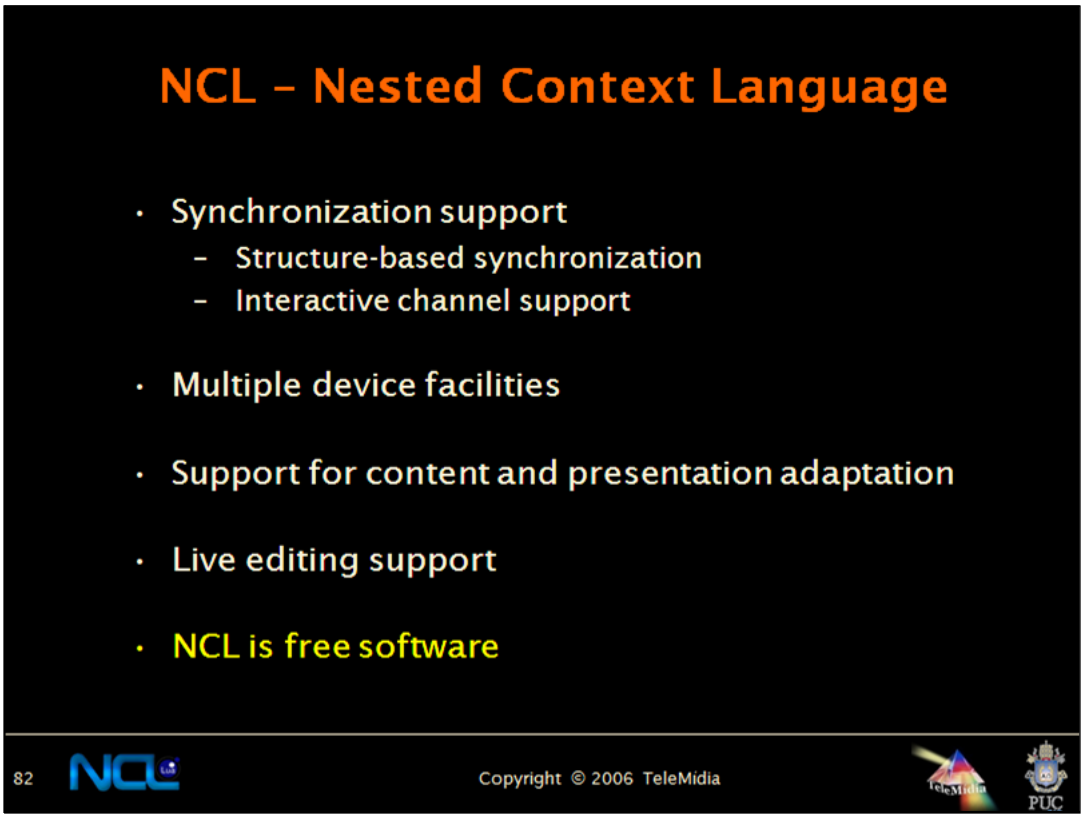


XHTML is not that expressive.

XHTML carries a legacy from previous technologies developed for text navigation and formatting, and it has lots of add-ons created to overcome its limitations in the DTV domain.

XHTML is focused on user-interaction declarative support as a means of synchronizing media assets' presentations. This narrow declarative scope forces application authors to solve spatiotemporal synchronization that goes beyond simple user interactions, as well as to solve content and presentation adaptations, and other issues usually found in DTV applications, by using imperative objects, usually written in ECMAScript.

With all middleware requirements in mind, we can answer the question why the ISDB-T and ITU-T IPTV middleware has adopted NCL as their declarative language.



The answer is because NCL is the single declarative language that offers declarative support for:

Temporal and spatial synchronization;

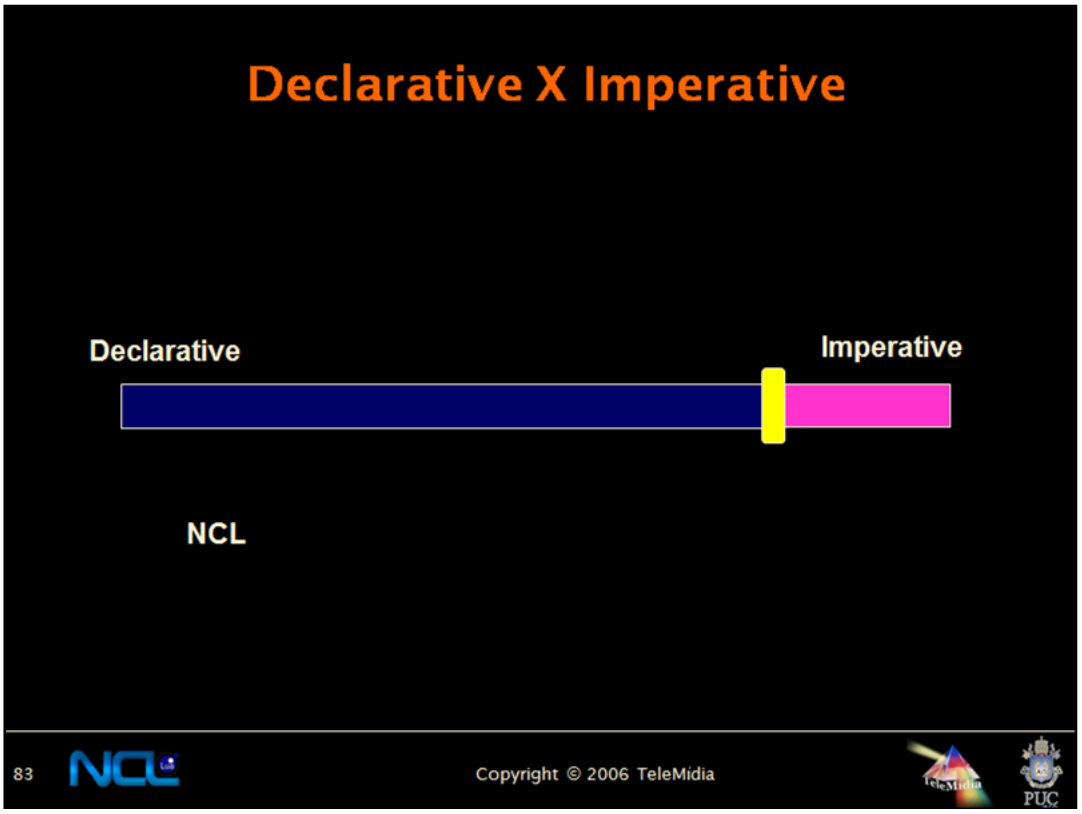
Exhibition on multiple devices;

and for

Content and presentation adaptation.

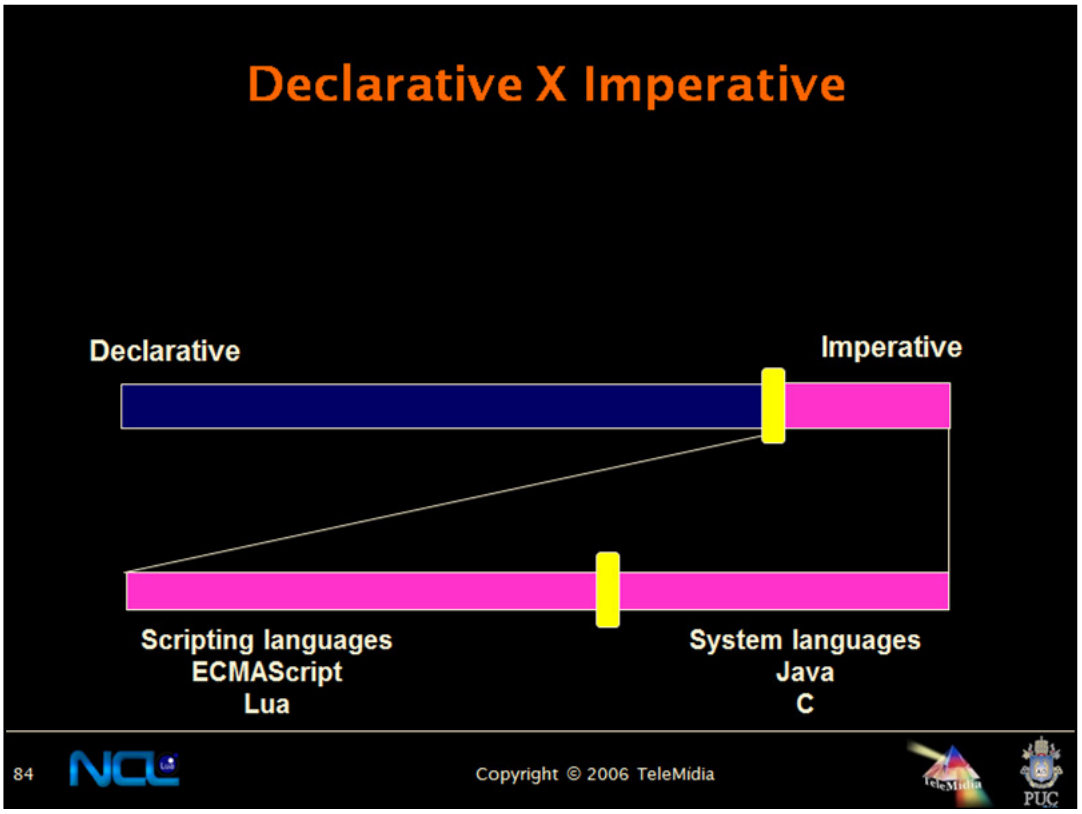
Besides this, NCL offers support for live editing non-linear programs.

And very important …, NCL is free software



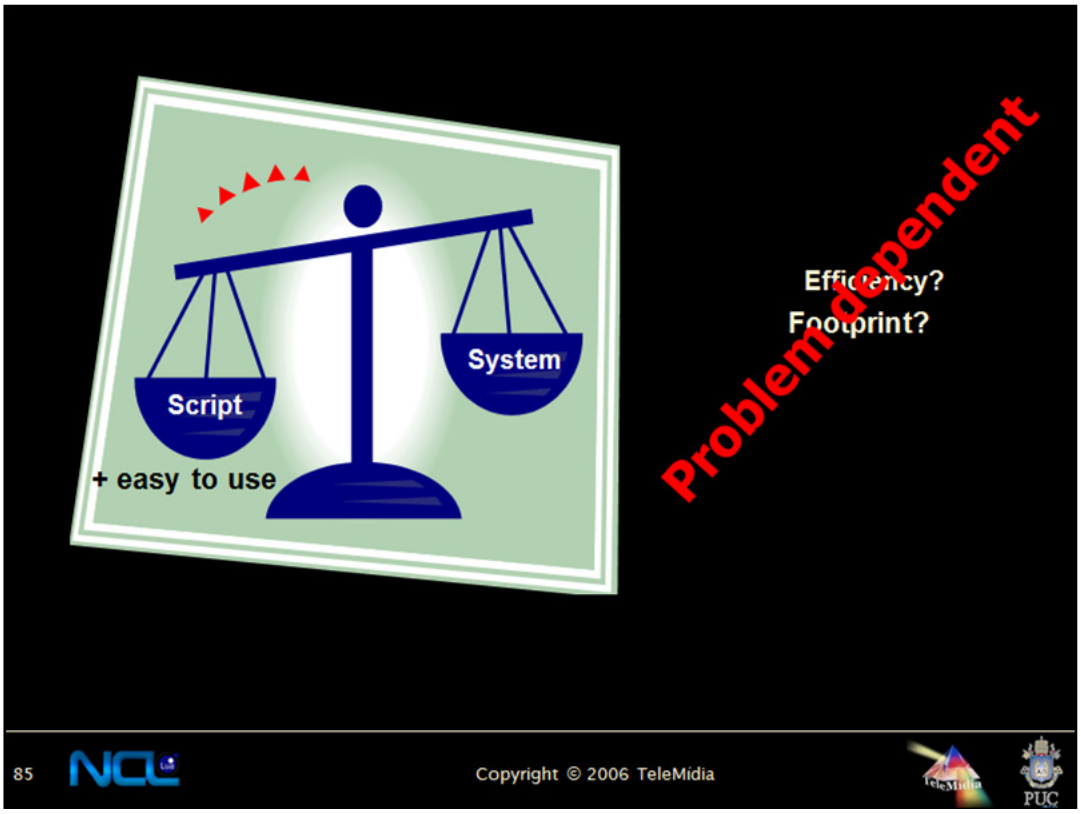
Expressiveness is the great advantage of NCL, which, as a glue language, declaratively includes HTML as one type of its objects, and, thus, is in harmony with all X-HTML based middleware, in particular BML and LIME.

NCL not only encloses but extends the HTML facilities, increasing the declarative expressiveness.



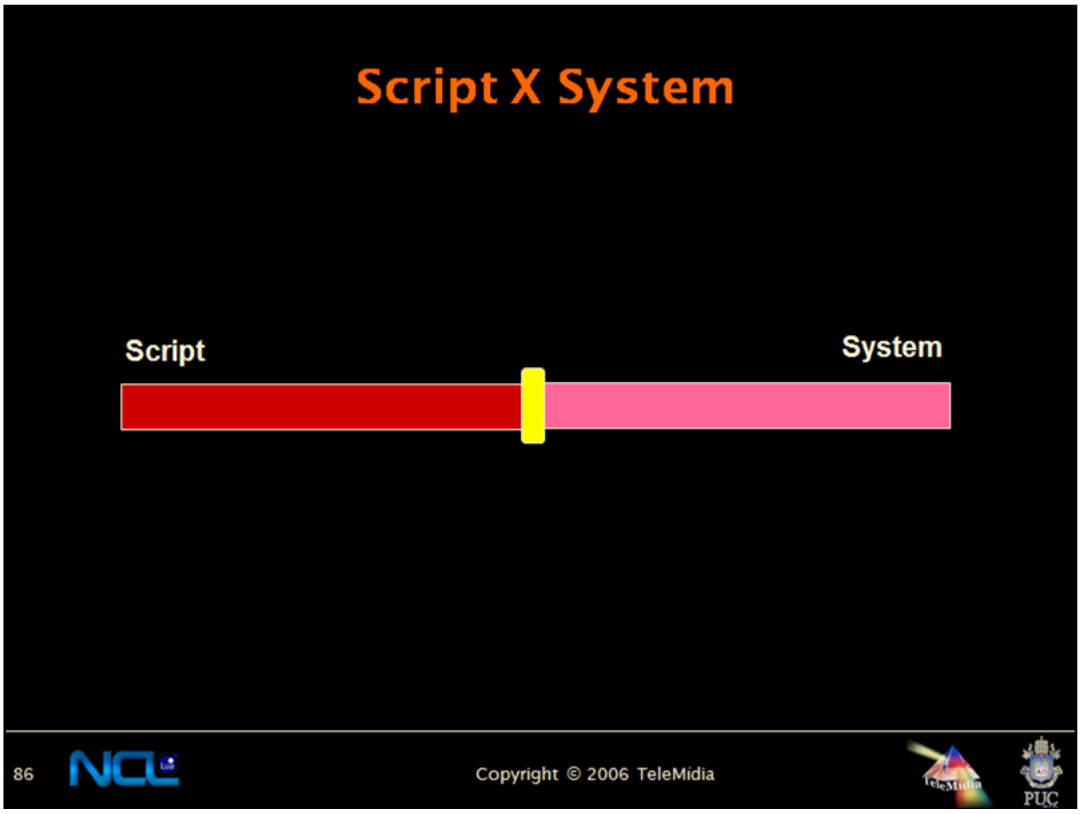
There remains the imperative middleware environment.

There are two types of imperative languages: scripting languages and system languages.

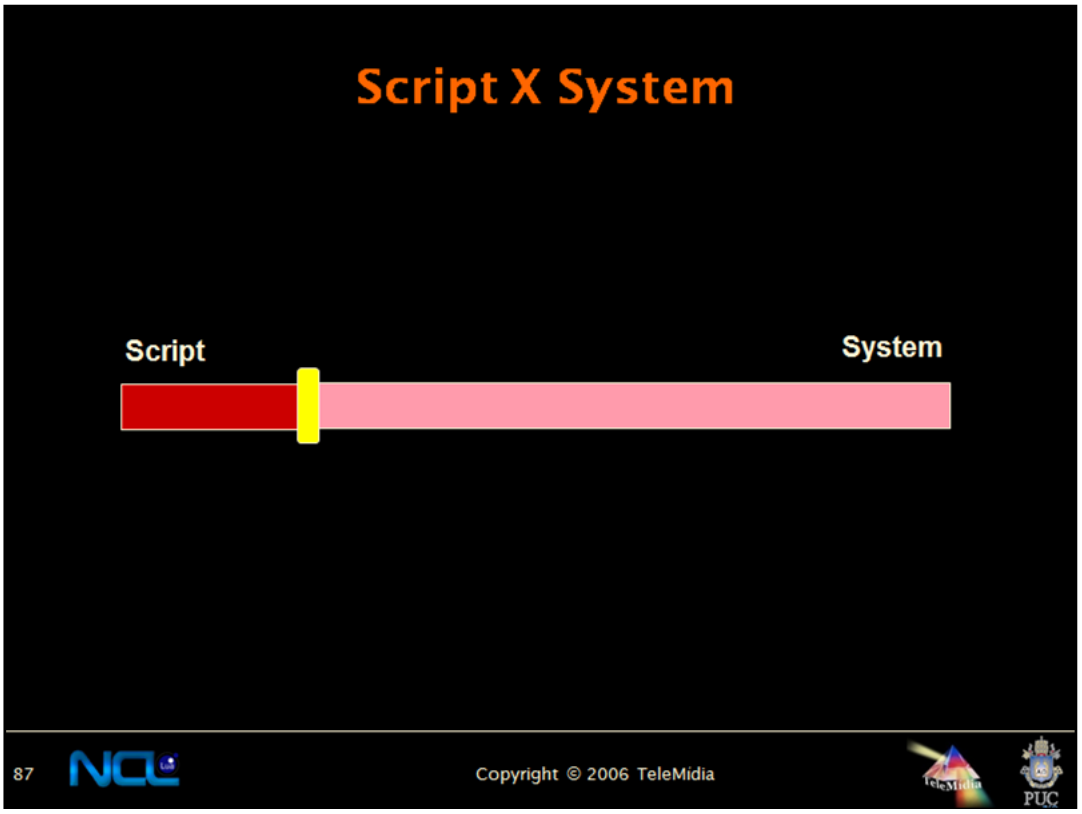


Again we have a choice to make. However, the choice is more difficult than before, because we do not have a solution that is always better.

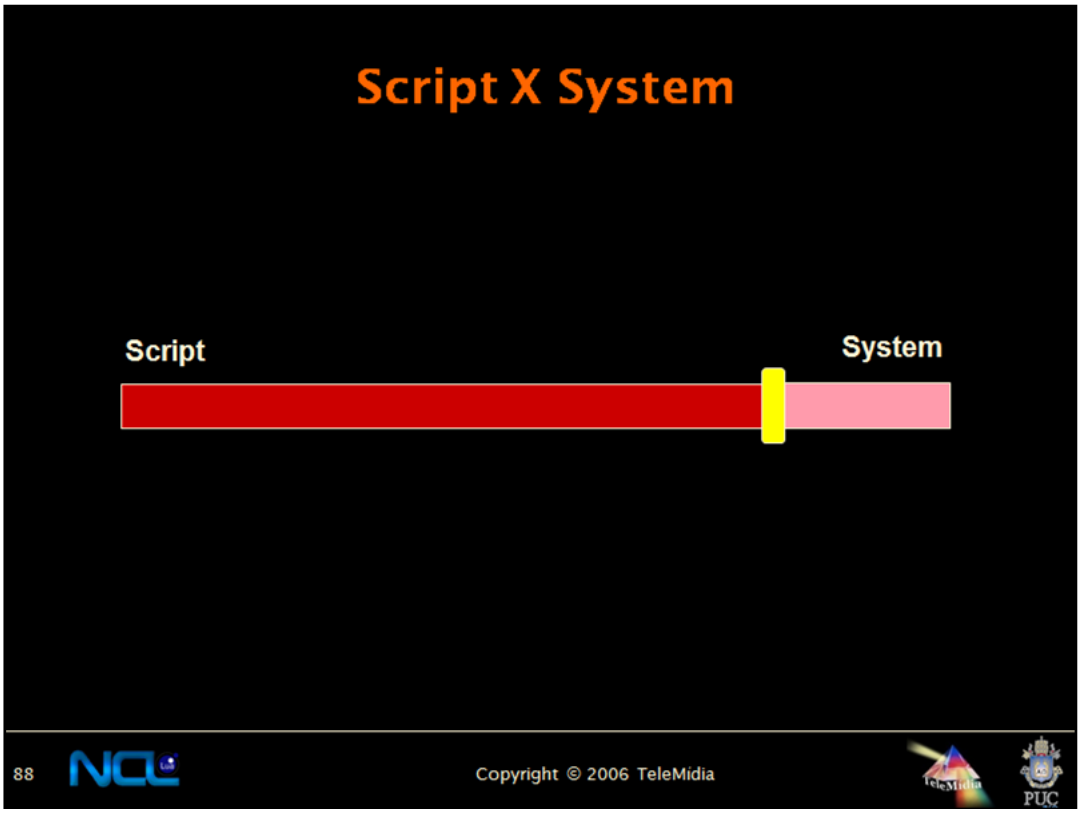
The only thing that can be said is that, usually, scripting languages are easier to use than system languages.



So, in our ruler



we must look for a scripting language that



solves as much tasks as possible, in a efficient way.

The ISDB-T system have chosen Lua as its scripting language and Java as its system language.

For portable receiver, ISDB-T only requires the Lua language support.

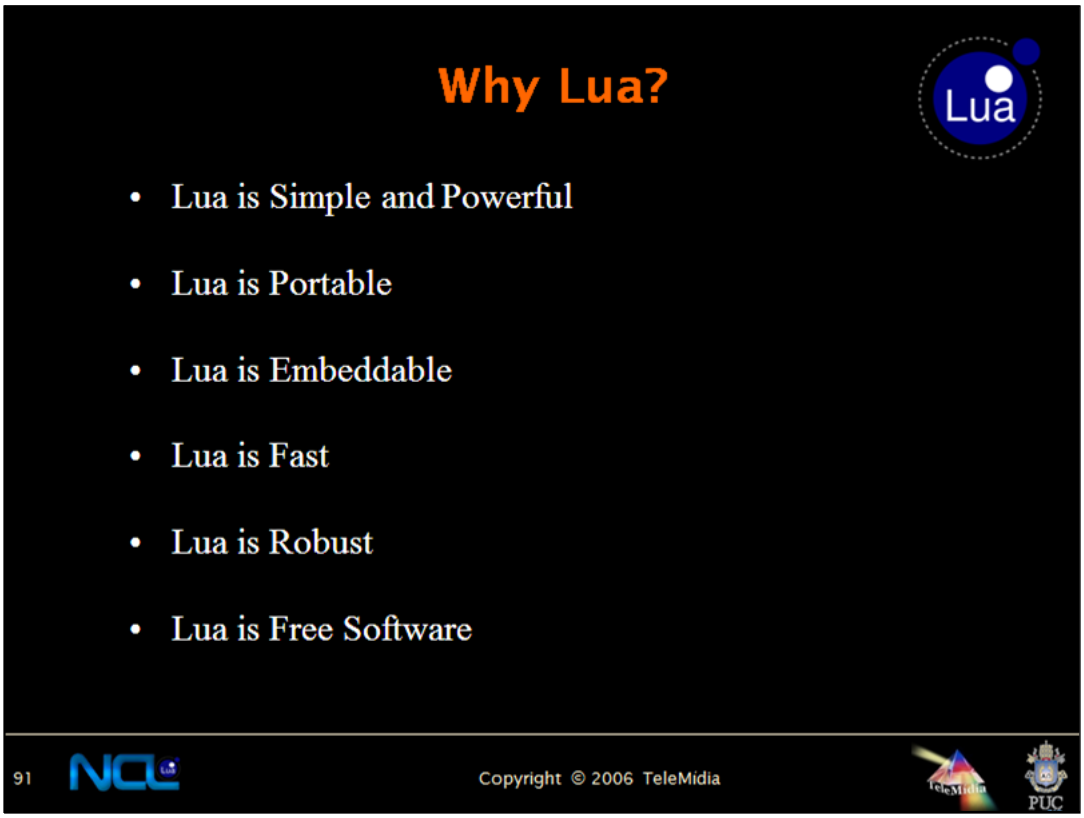
It should be strongly stressed that **Lua and Java has the same expressiveness**. What can be done using one language can be done using the other, however, with different complexity and efficiency.

The ITU-T Recommendation for IPTV services **only requires the Lua** support either.

Thus, in order to extend the NCL language basic model adding decision-making features that deserves the imperative paradigm, Lua objects are part of the Ginga-NCL specification.

Lua is the scripting language of NCL.

Why Lua was chosen?



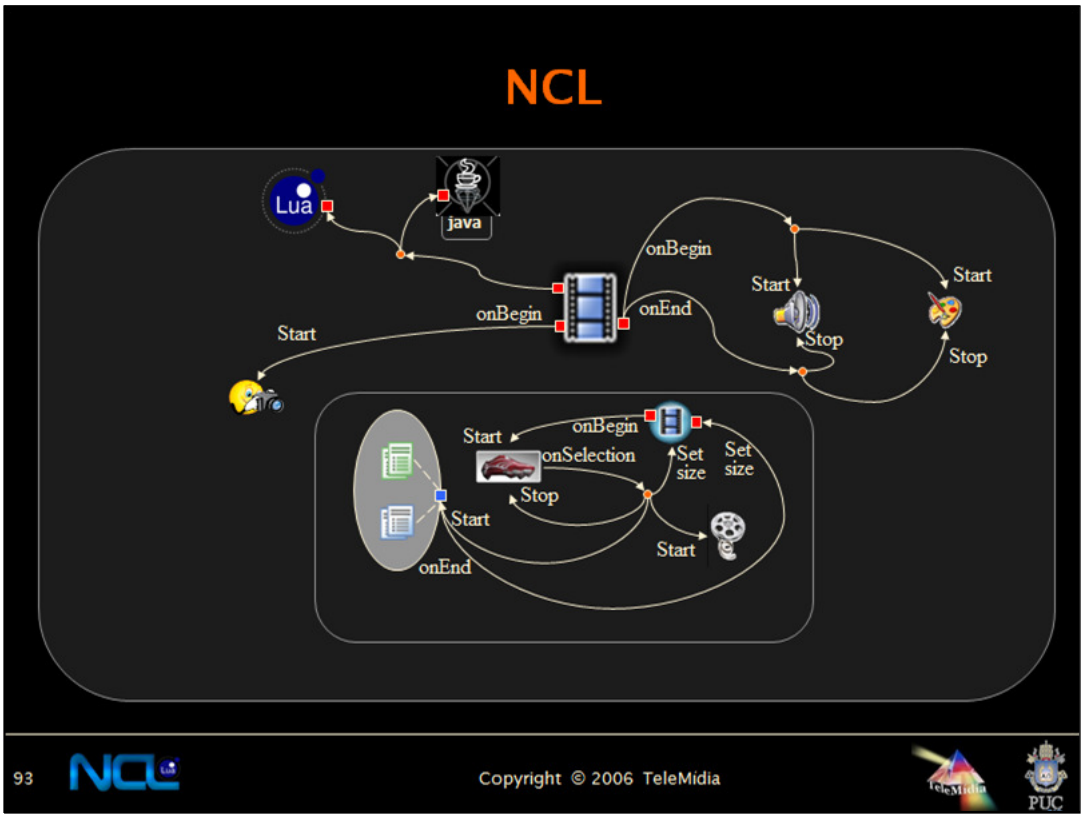
Among all these Lua's characteristics and advantages as a scripting language, I would like to stress its efficiency.

Lua is one of the most efficient dynamic languages (interpreted, dynamic typing)

-faster than Perl and Python; much faster than Tcl and JavaScript

This slide shows a comparison with the Javascript implementation used in the Mozila (FireFox).

In average, Lua is 7 times faster with a memory footprint 40 times lesser.



Now we can summarize the NCL-Lua support.

In short, NCL is a Glue language that relates objects in time and space, independent from their types. Therefore, we can have perceptual objects like: videos, audio, images, texts …

Objects can be organized in sets, like a drama is organized in chapters, which are organized in scenes, which are organized in shots, which are organized in takes, and so on.

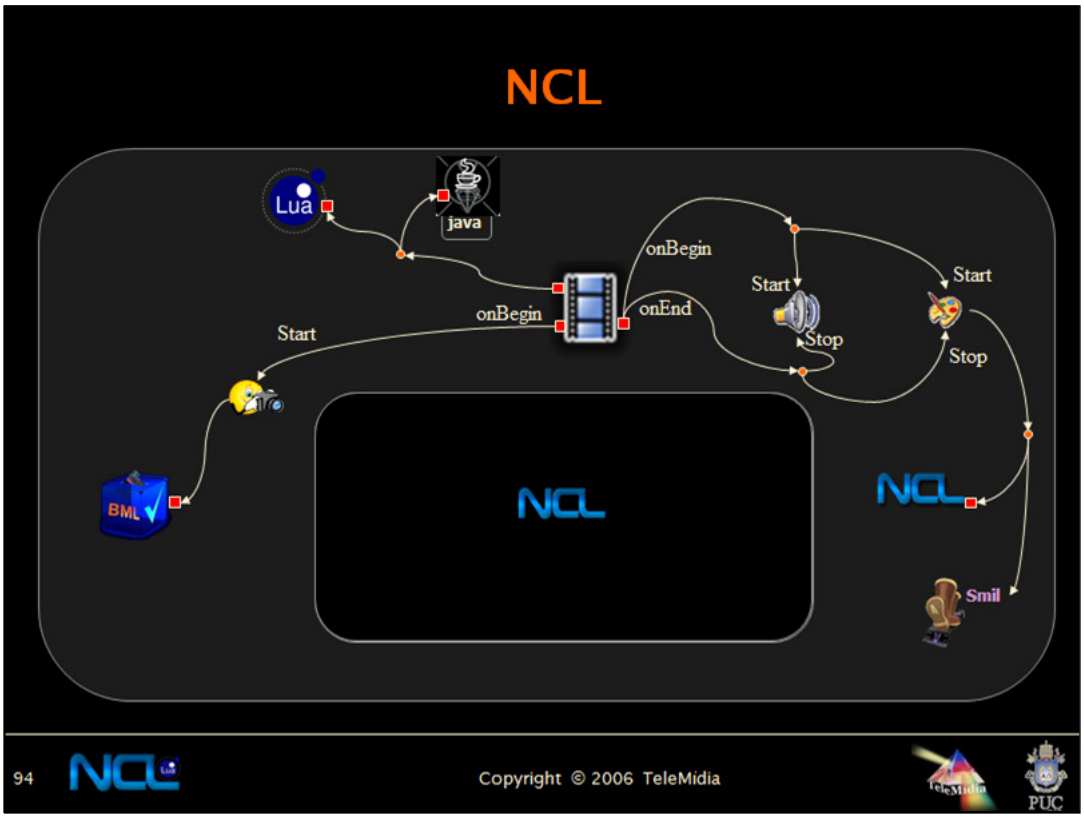
Sets of objects can be nested, and the document itself is considered a set by NCL.

Set of alternatives may also be defined for object presentations. For example, a text in English or in Portuguese, depending on the viewers language.

Objects may be related through its interface: a track in a video or audio, a region in a image, etc.

Objects may be reused in different sets, with different interfaces.

Relationships may be defined among objects.



We may have not only perceptual objects, but also objects with imperative code, like Lua code and Java code.

This allows to extend the language basic model adding decision-making features that were otherwise not possible.

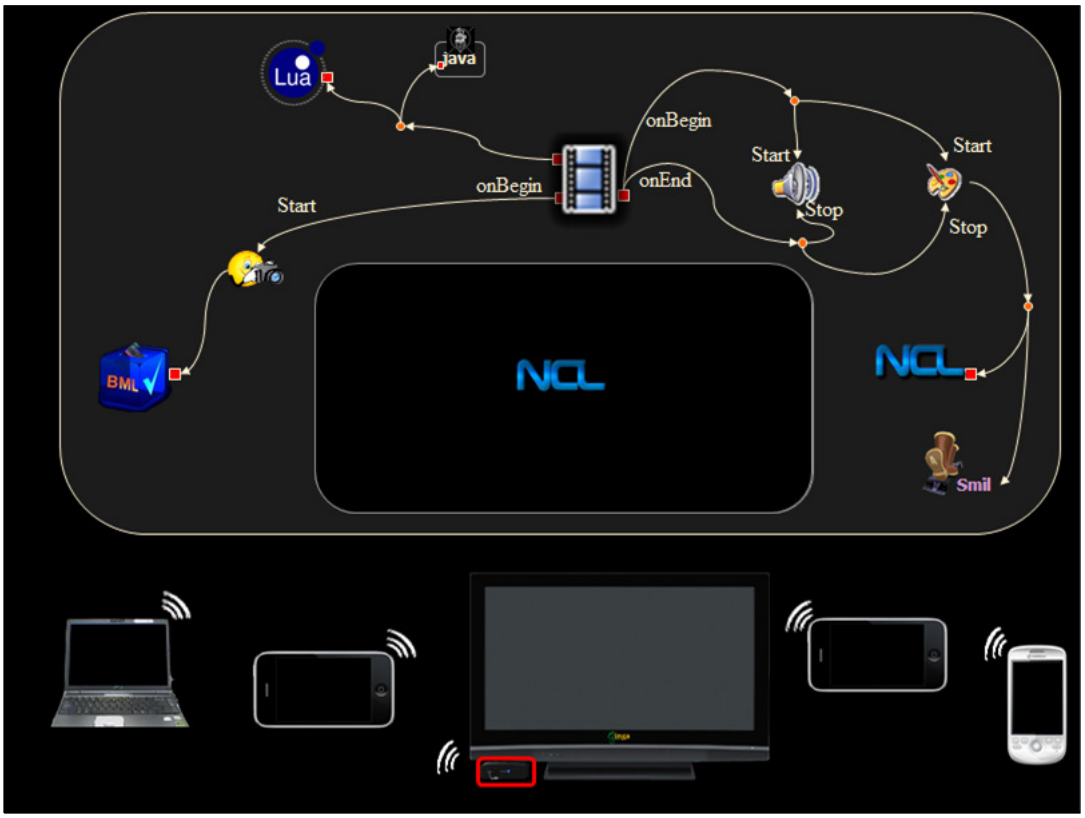
We may also have objects with other declarative codes, like SMIL objects, other NCL embedded documents, HTML-based objects, etc.

NCL is a glue language that does not restrict or prescribe any media-object content type. Which are the media objects supported depends on the media players embedded in the NCL engine (the NCL formatter). Therefore, Ginga can run any HTML application developed for ISDB, DVB and ATSC, depending only on the XHTML player implementation, as for example, a BML or a LIME application.

Note thus that NCL does not substitute but embeds HTML-based objects.

Interfaces may also be defined for imperative objects' function and methods; and for hypermedia declarative objects' anchors.

Imperative and declarative objects may also be related as any other object.

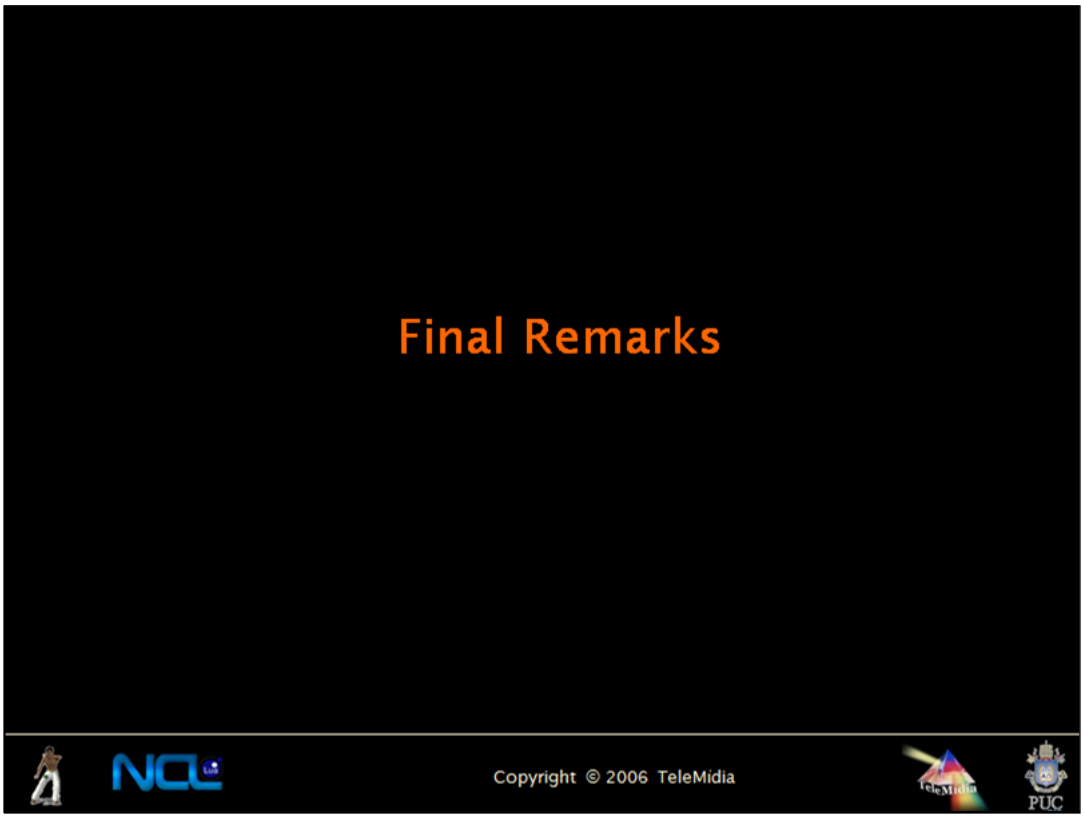


NCL was designed for multiple device environments, as in a home network.

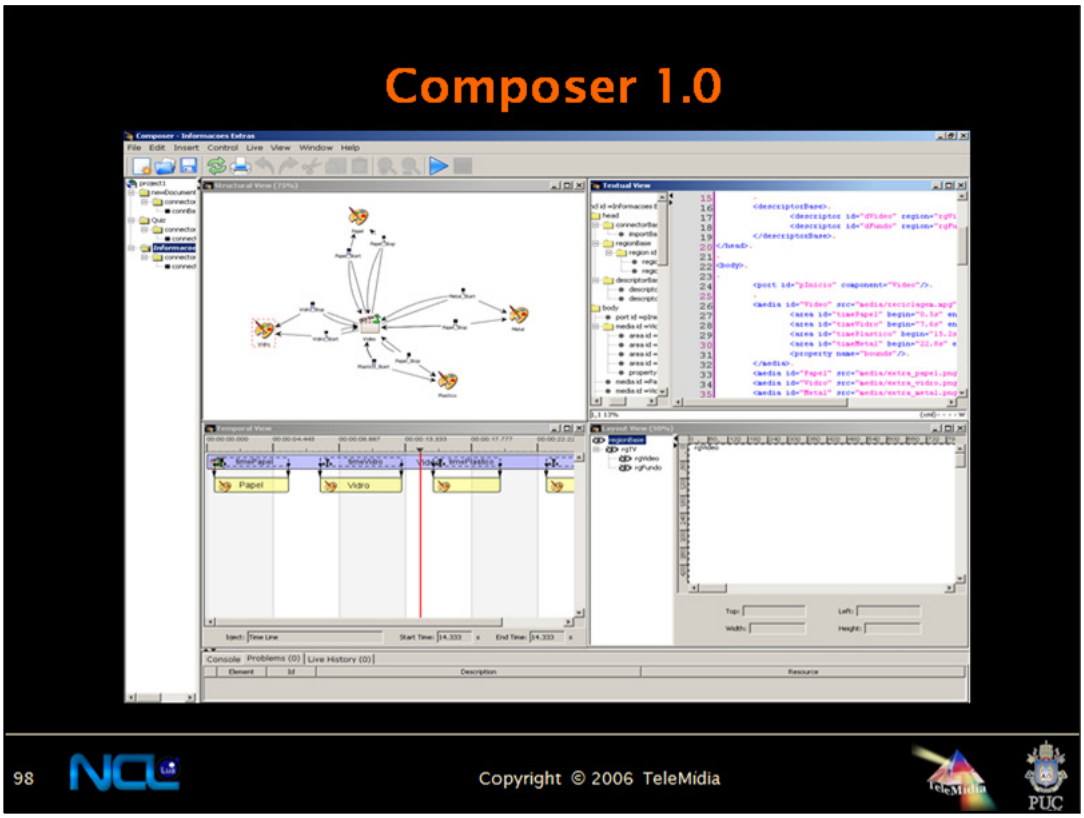
NCL allows specifying where and when each object is presented.

Some objects can be placed on the TV set to be exhibited to the whole audience. An embedded Java application can be sent to a device with a Java virtual machine. A SMIL document to another specific device with a SMIL player. A BML application to a device that support the Japanese standard. An NCL widget to a secondary device that implements Ginga-NCL, etc. All this happening with the control of a media center.

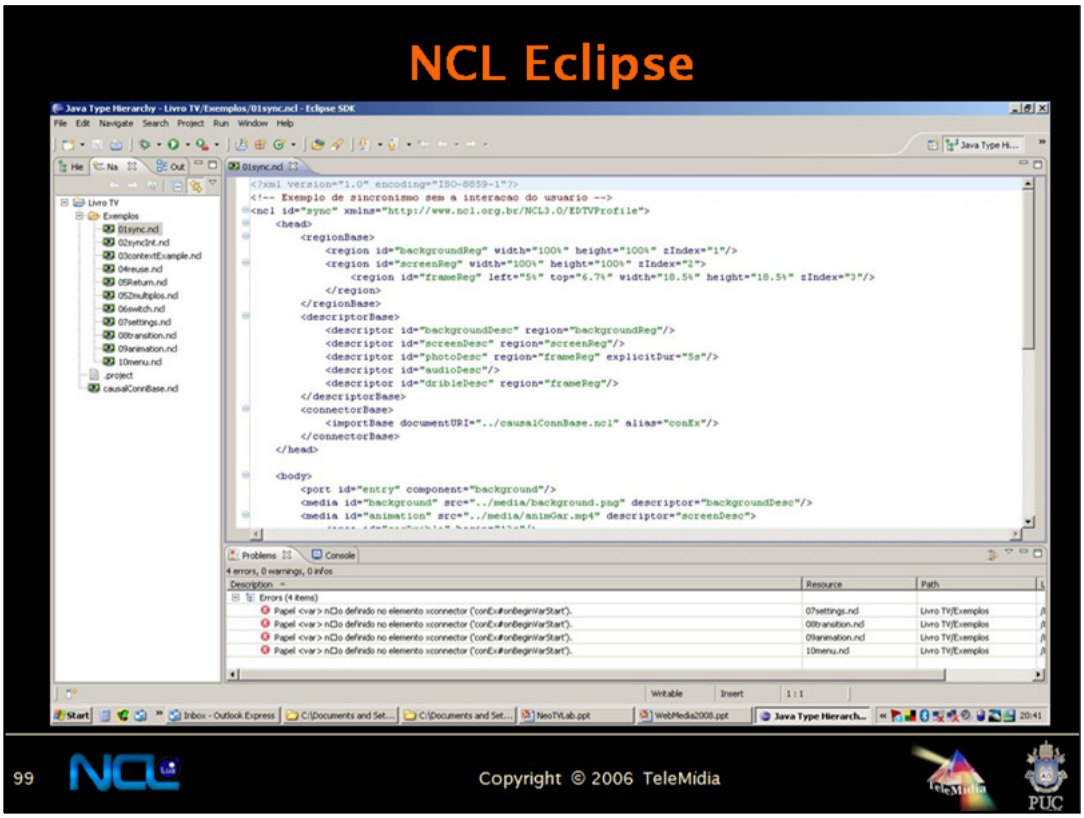
Before moving to the last topic of this presentation, let us see one more demo illustrating how NCL acts as a glue language.



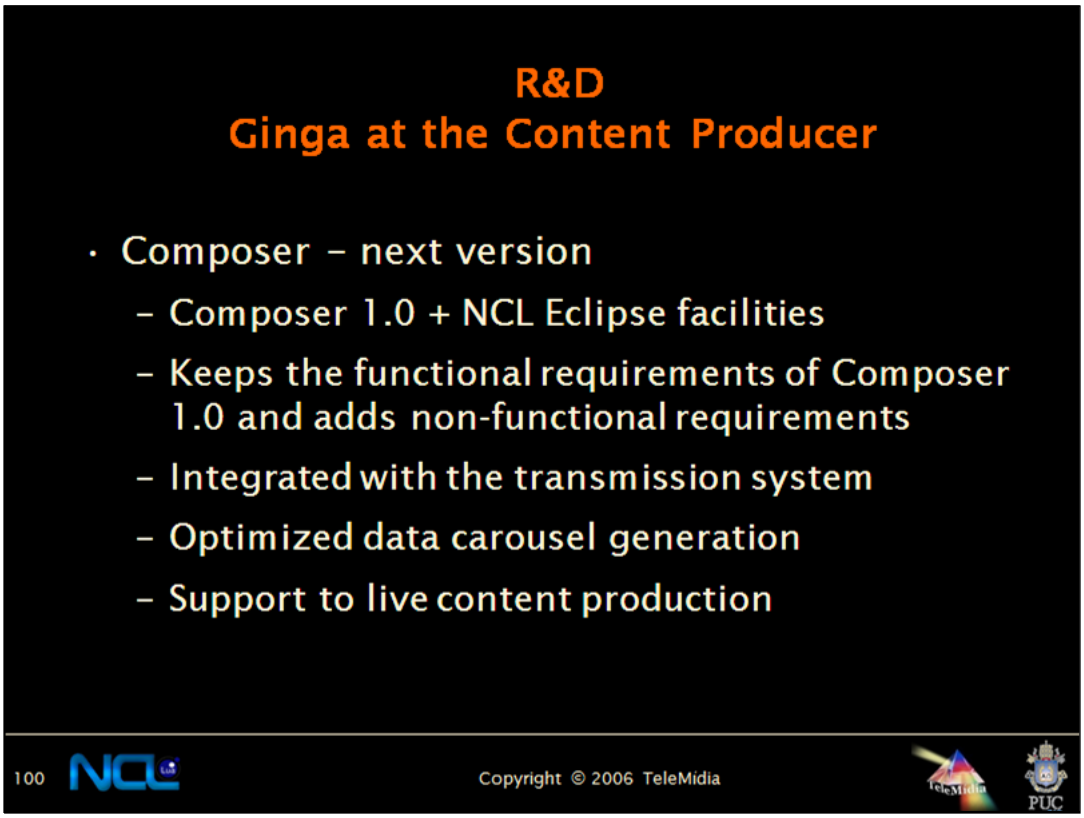
Let us now move on to the last topic of this presentation. Future research directions of the TeleMidia Lab, where Ginga-NCL has been conceived.



There are several authoring tools for NCL applications, like NCL composer and



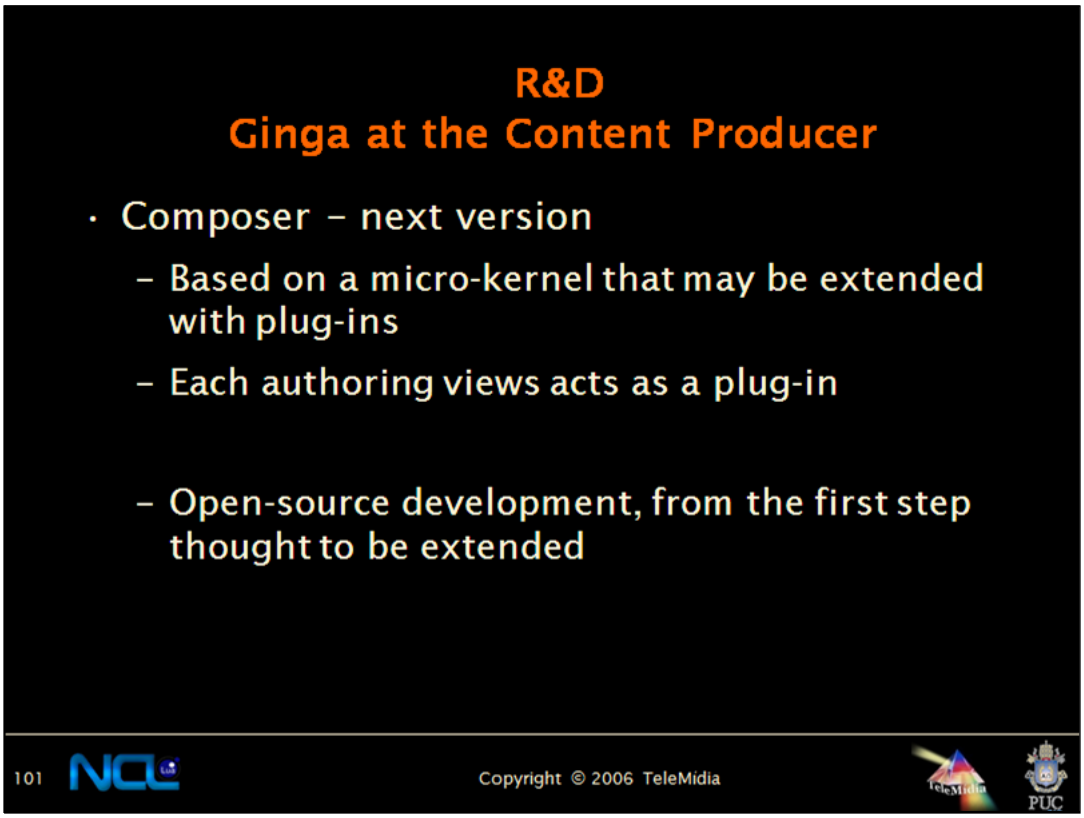
NCL Eclipse.



The next Composer version integrates the facilities of the composer version 1.0 and the NCL Eclipse, and also integrates transmission system protocols.

Several related work addresses functional requirements for authoring hypermedia applications. However, no one addresses non-functional requirements like: reliability, maintainability, adaptability, performance and extensibility.

In the next version of Composer not only the functional requirements of Composer 1.0 and NCL Eclipse are kept, but non-functional requirements are added.



The next version of Composed is based on a micro-kernel that may be extended with plug-ins.

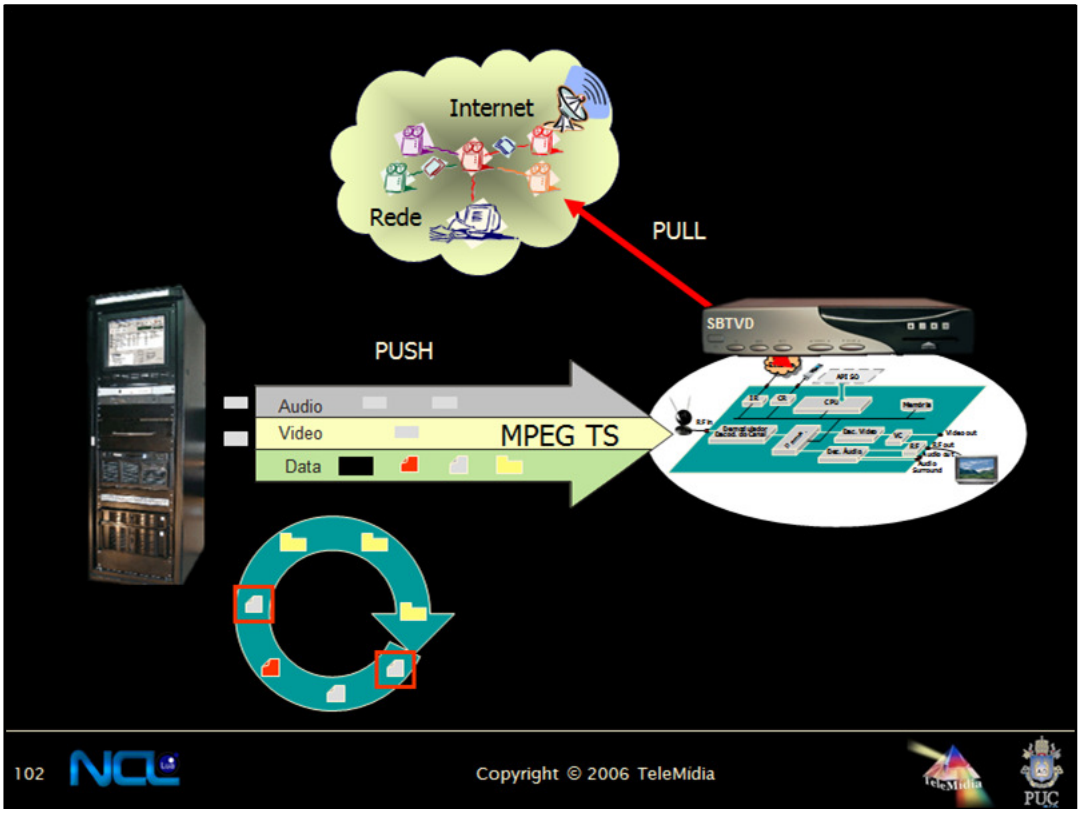
Each authoring view acts as a plug-in.

The advantages are:

* the fast adding of authoring views; and
* the fast adding of functionalities not predicted yet.

It must also be stressed that Composer is an open-source development, from the first step thought to be extended.

As regarding the composer transmission plug-ins, several QoS problems is being addressed.



Applications can be pre-produced or generated on-the-fly; applications can be transmitted on demand (as in IPTV) or as unsolicited data (in a data carousel, transmitted time after time, as in T-DTV).

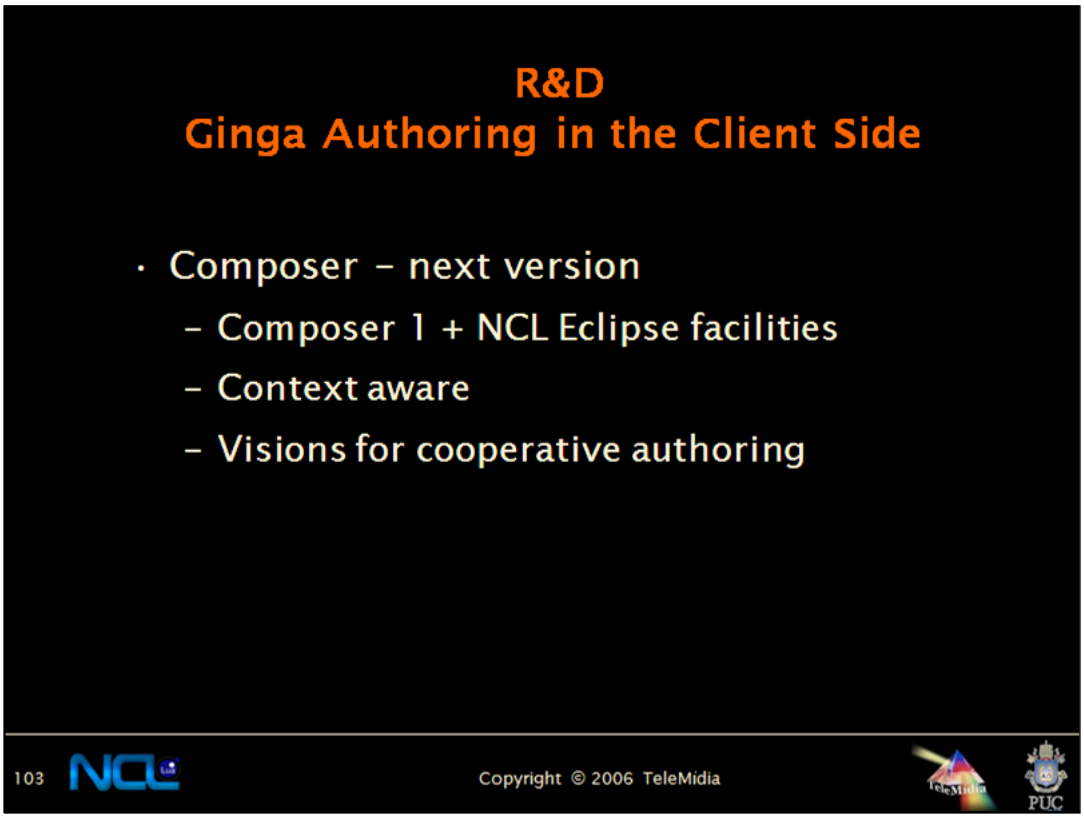
How many times an object is placed inside a carousel and in which place gives the maximum delay for that object and also the maximum error rate. However, note that if a carousel increases, the delay also increases for other objects. Increased delays can cause synchronization mismatches.

The amount of carousel bandwidth, the main video and the main audio bandwidths is limited, and must be less than 6 MHz, in the case of terrestrial DTV in Brazil. If we have a bigger carousel we will loose video and audio quality.

So, we have a very interesting carousel management optimization problem, comprising bandwidths, delays and other QoS parameters, all these guided by a temporal graph that can be extract in advance from the application specification.

The same data structure can also guide the QoS negotiation process in downloading pulled data. QoS in advance techniques proposed for resource reservation in mobile networks can be a very interesting starting point for the solution.

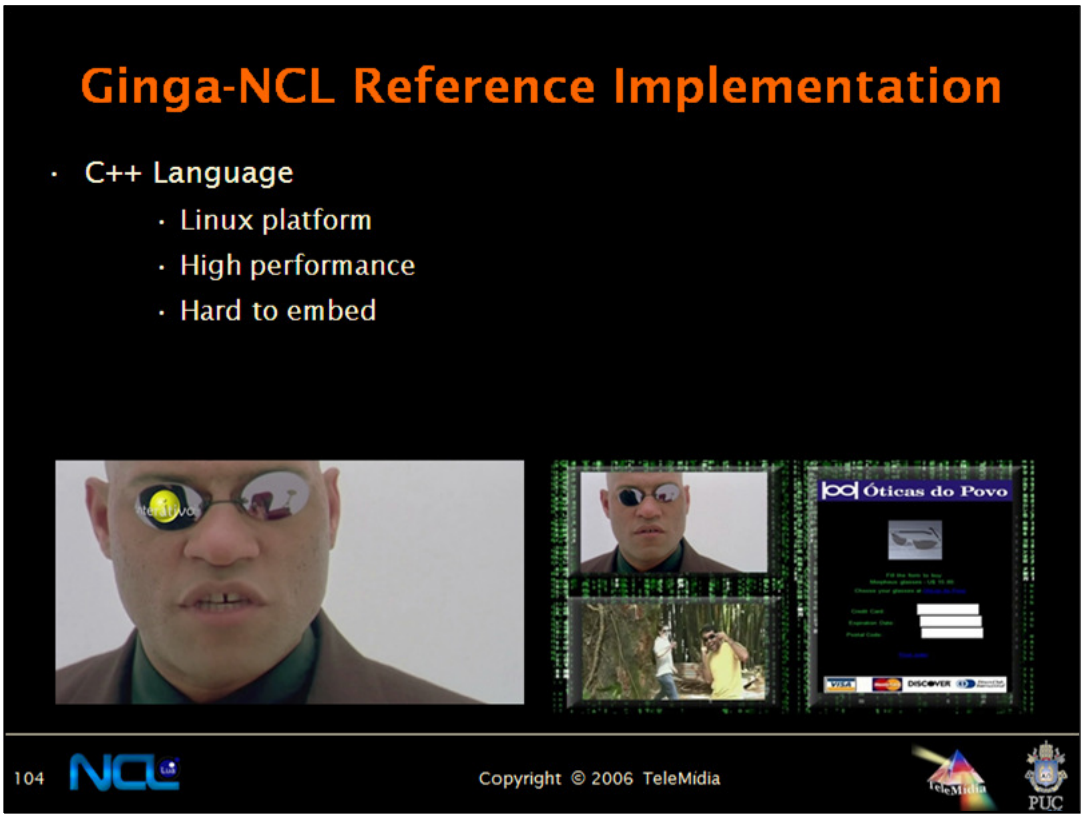
In order to maintain the required synchronization, content adaptation can also be necessary for the carousel and interactive channel management.



Authoring tools, and so Composer in its next version, must also be adapted to be used in the client side, giving rise to several social TV applications.

NCL allows specifying content and presentation adaptations, depending on the type of the exhibition device; on the user profile; on the user location; and on network conditions.

The NCL engine must perform necessary adaptations based on variables collected and controlled by a context manager. An authoring tool must also provide support to context aware applications.



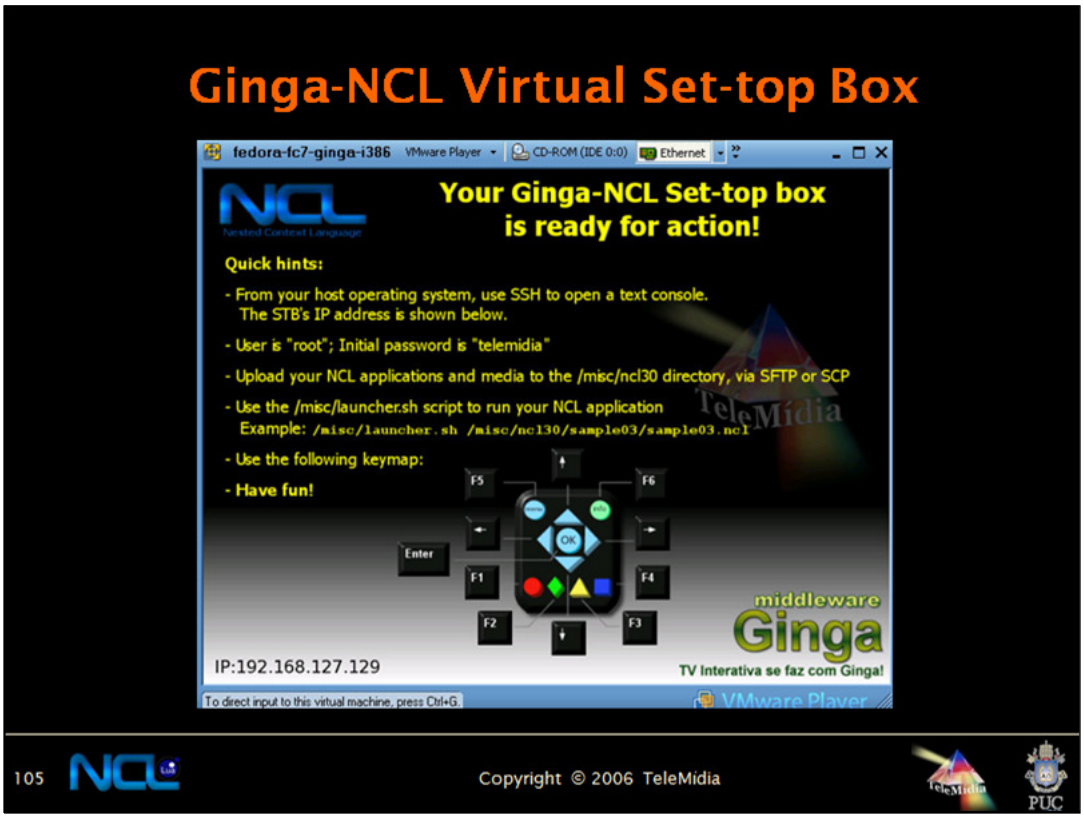
As regarding the client side middleware,

there are several implementations for Ginga-NCL:

as the reference implementation in C/C++ for Linux platform.

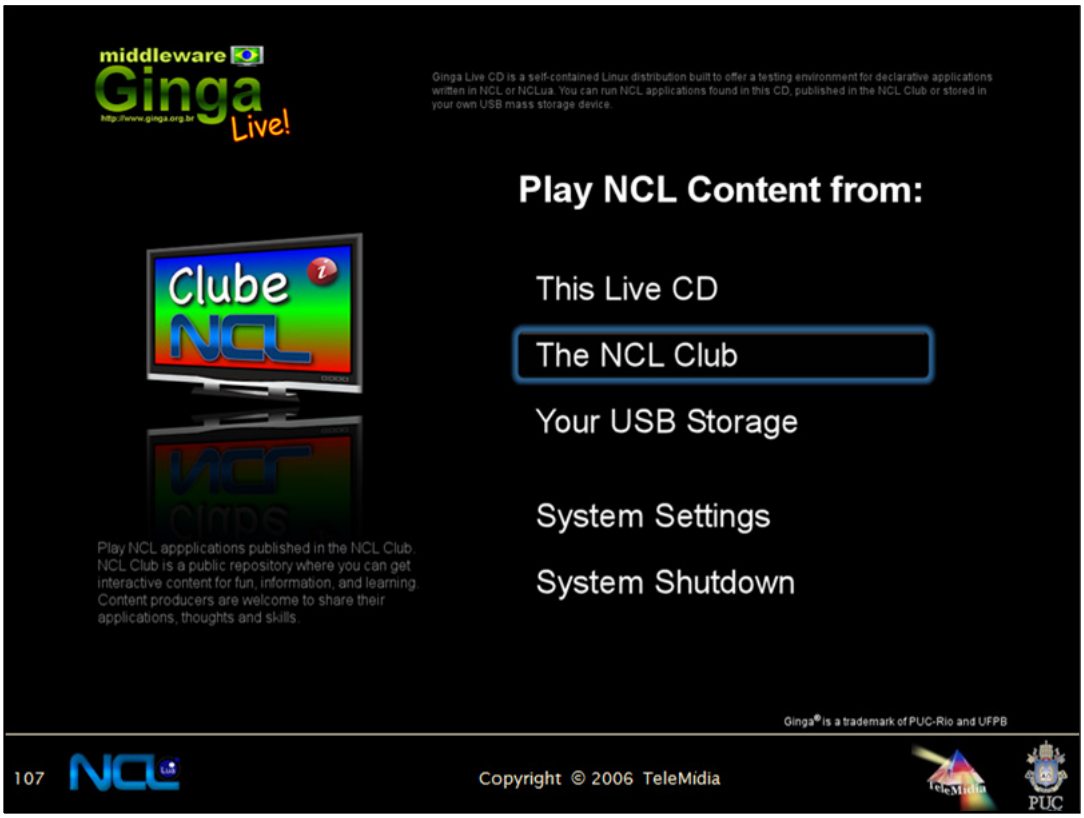
The current release of the reference implementation is component-based.

The component-based release allows easy on-the-fly component update , and has a much better resource management, thus reducing hardware requirements.



The Ginga-NCL Virtual Set-top Box is a virtual Ginga-NCL machine for VMware products.

The live CD is a complete Ginga-NCL bootable CD that transforms a lap-tot or desktop into a Ginga-NCL set-top box for application development and test.

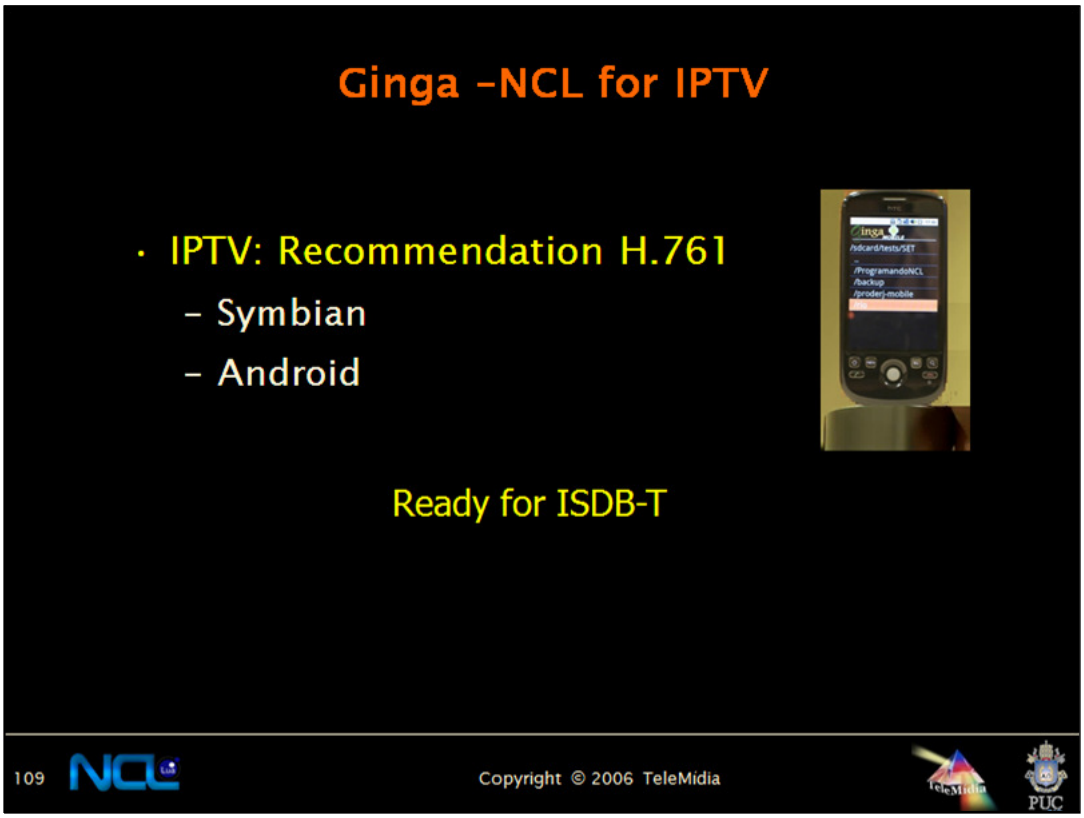


Applications that can come from a pen-drive or from the NCL Club.

NCL club is a public repository of NCL applications and NCL-Lua Widgets.

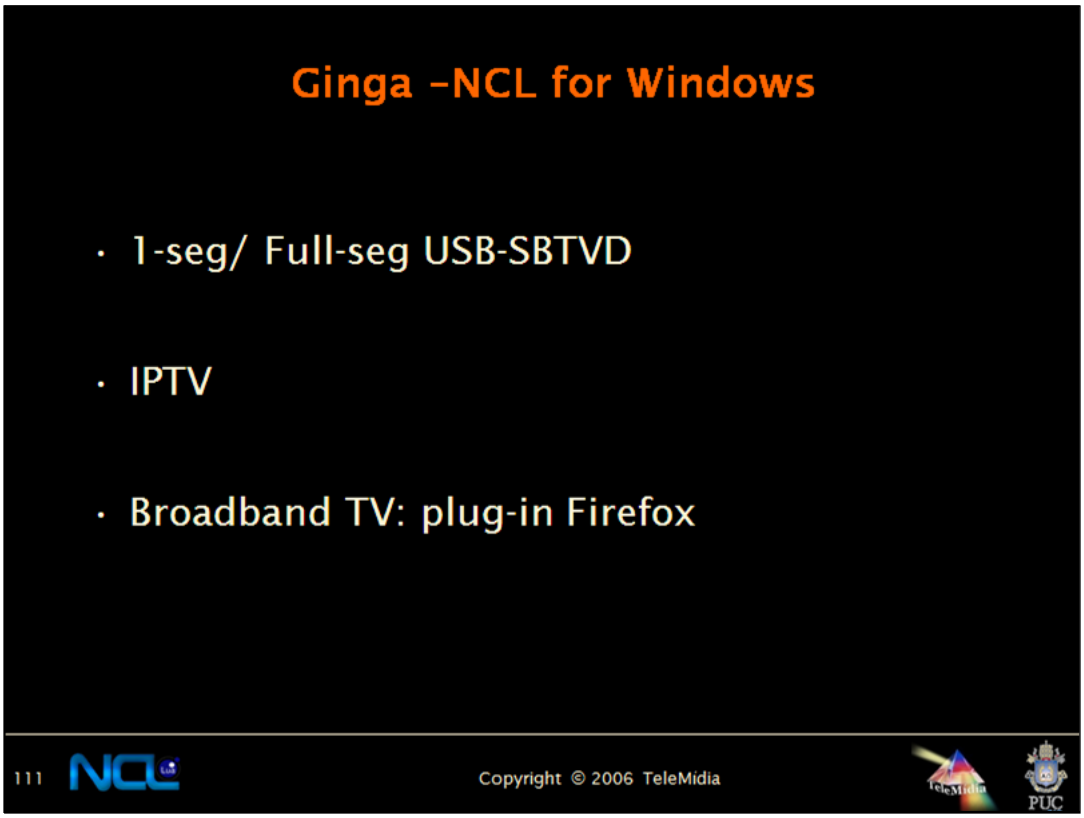
SAGGA is a project for automatic generation of NCL application, based on video and other NCL applications available on the OVERMUNDO, CLUB.NCL and ZAPPIENS sites.

Overmundo and Zappiens are video sites following creative common licences, like the Club.ncl

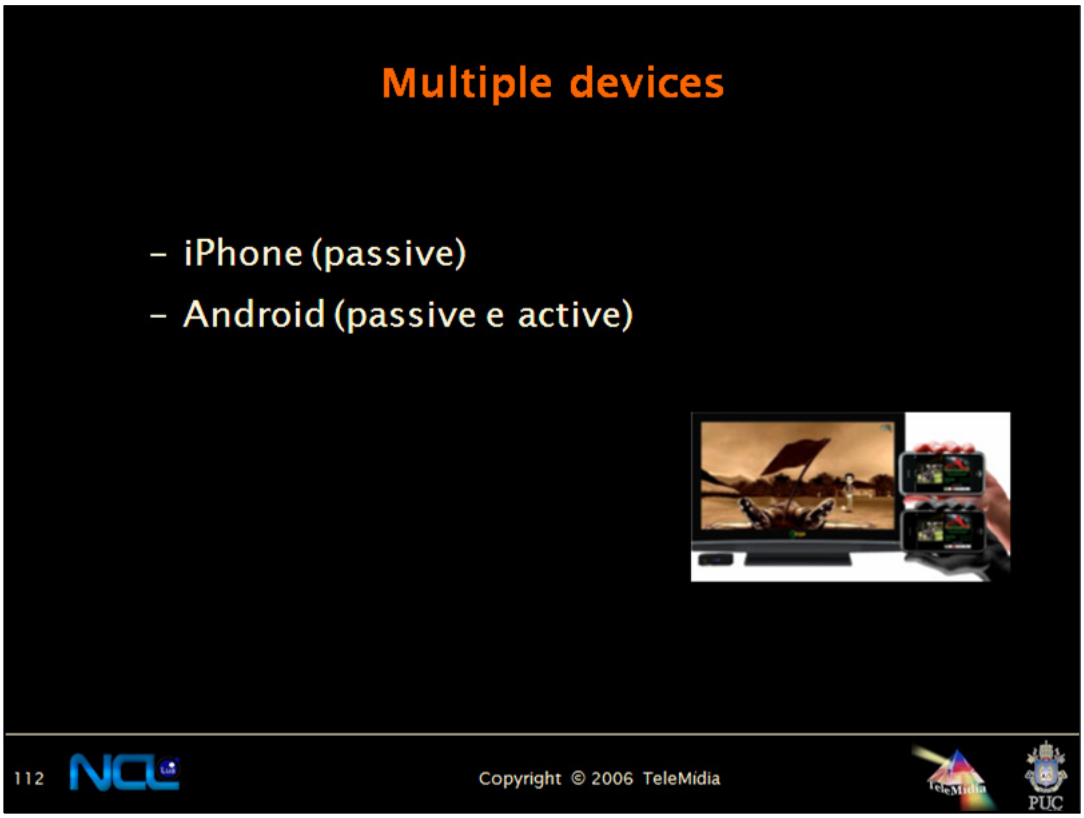


There are also Ginga-NCL implementations for portable devices, aiming at IPTV services, but that can be extended to ISDB-T terrestrial DTV.

More recently, a Ginga-NCL implementation was launched, both for Windows and Linux platform, for PC with USB-ISDBT receivers.



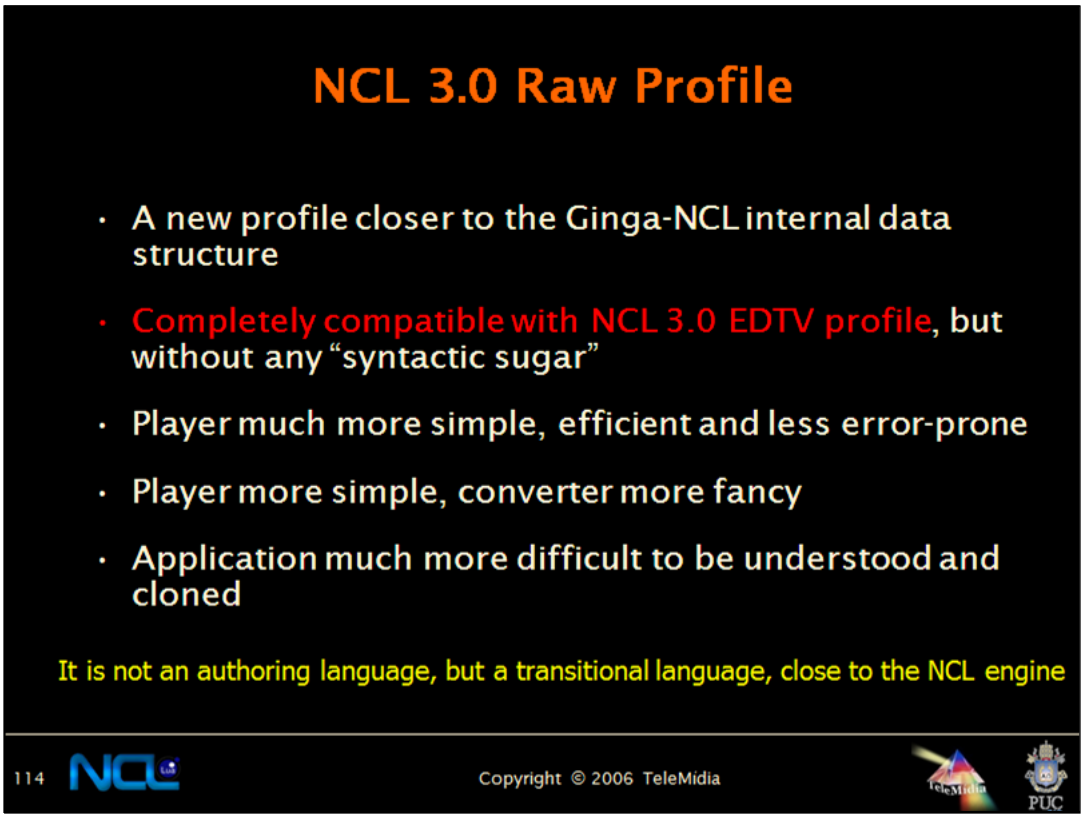
The open-source code for windows is available, together with a version to run as a Firefox plug-in, addressing **broadband TV needs.**



The multiple device implementation is available for iPhone (class 1) and Android platform (class 2).

How a set-top box manages its multiple secondary devices is a problem to be solved by each manufacturer. However, everything points to use the DLNA architecture. A standard proposal on how to register and manage Ginga-NCL secondary devices is under study.

As regarding NCL next generations,



a new profile of NCL, closer to the Ginga-NCL internal data structure is being developed.

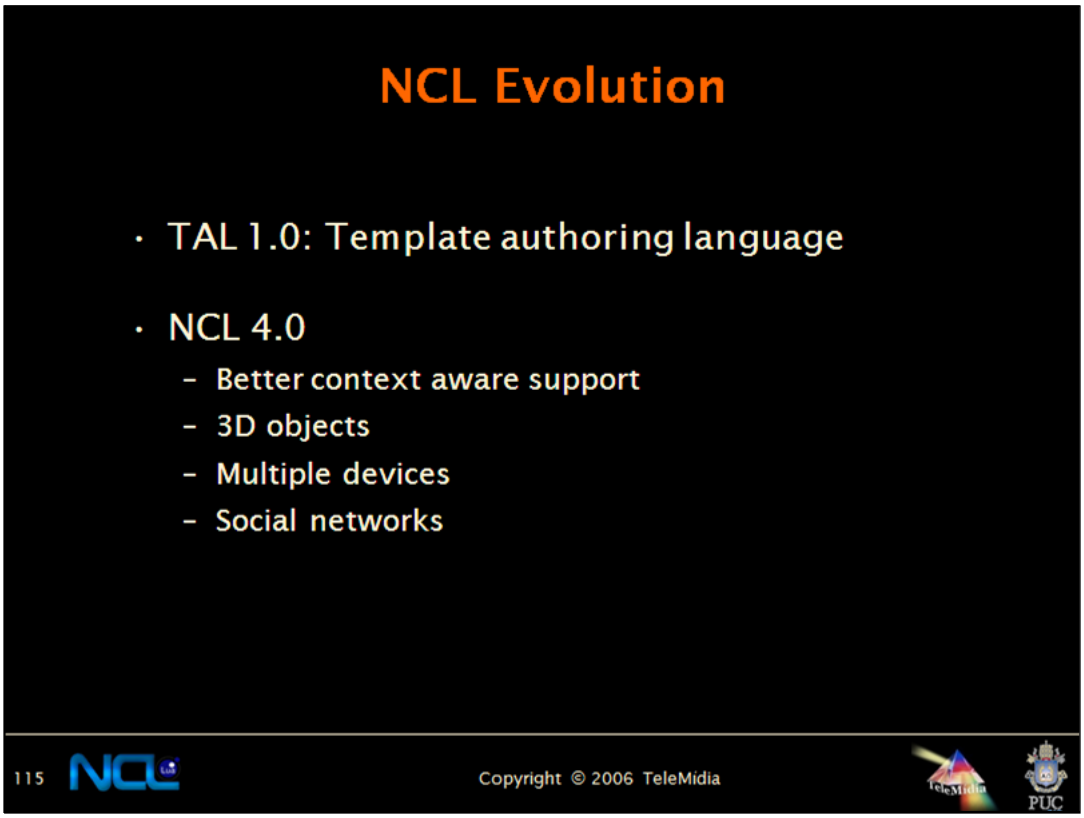
The profile is completely compatible with NCL 3.0 EDTV profile, but its is clean, without any "syntactic sugar".

However, the profile is not for an authoring language, but it is a transitional language (probably a transfer language), close to the NCL engine,
allowing a player implementation much more simple, efficient and less error-prone.

NCL applications will still be developed following the EDTV profile and then converted to this **raw profile**.
Note that applications translated to the raw profile is much more difficult to be understood and cloned.

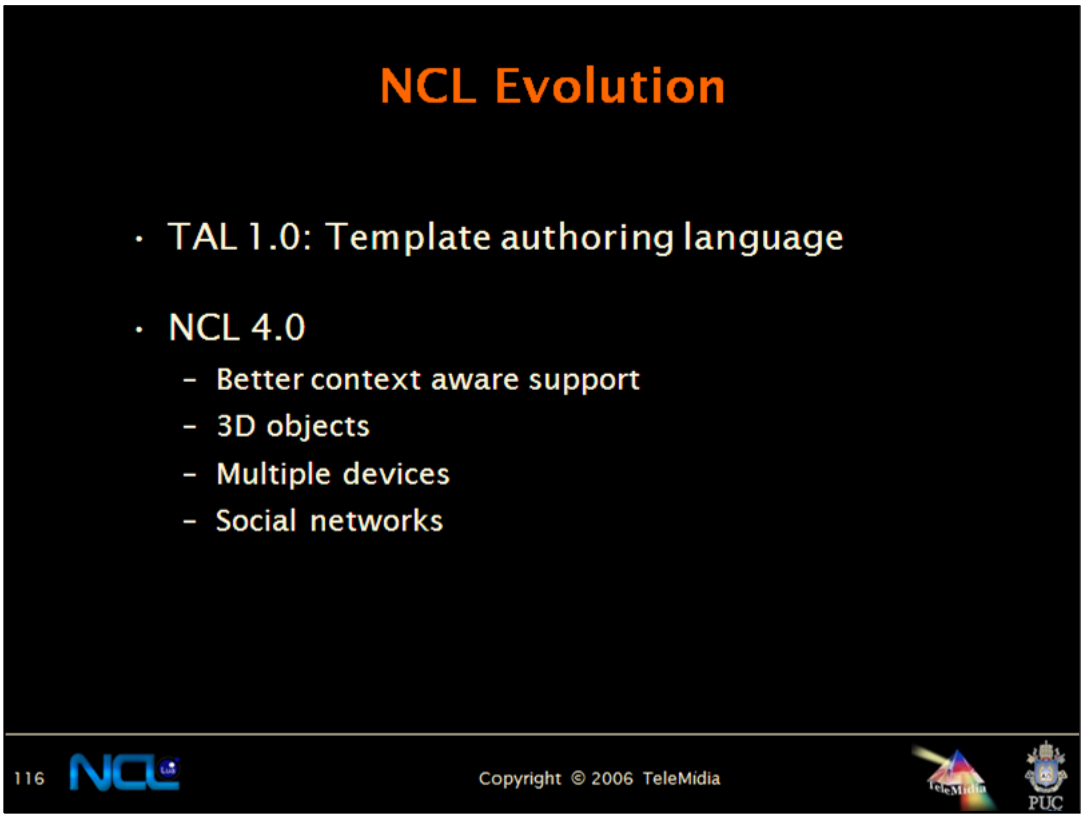
The raw profiles focus only on having an easier player implementation. However, a player more simple implies on a converter more fancy.

This is not a problem, even because the conversion can be done in the server side (broadcast side).



Several issues are under consideration for the next generation of the NCL language (EDTV profile).

A semiotic analysis of NCL in its several cognitive dimensions is being done in order to guide the conception of a new Template language (the TAL language) for NCL, and also to guide the next version of NCL.



Context aware applications need a higher level data structure than the one currently supported by NCL. The language should be extended to support metadata following an appropriate ontology to describe application, user and platform profiles.

Today NCL only allows media object exhibition on two dimensional rectangular regions.

The next step is to allow defining media object presentation on 3D surface.

As a glue language NCL should also be able to embed 3D objects, specified in another 3D language, like X3D.

This 3D objects should be able to relate with other 2D and 3D objects, in the same or different 3D world.

Moreover, the presentation should take profit of the multiple device facility of NCL in a true virtual environment, moving social applications one step ahead.

Well,

This brings us to the end of the presentation.

In this slide you can see some sites where additional information can be obtained and all mentioned open source implementation for Ginga-NCL and NCL authoring tools can be downloaded.

Thank you very much for your attention.