# Interactive Digital Television for Developing Countries: a software/hardware perspective

Antônio Augusto Fröhlich
Federal University of Santa Catarina
Software/Hardware Integration Lab
88040-900 Florianópolis - SC - Brazil
E-mail: guto@lisha.ufsc.br
http://www.lisha.ufsc.br/∼guto

Valter Roesler
Federal University of Rio Grande do Sul
Informatics Institute
91501-970 Porto Alegre - RS - Brazil
E-mail: roesler@inf.ufrgs.br
http://www.inf.ufrgs.br/∼roesler

**Abstract**

*Interactive Digital Television* is mostly a matter of entertainment for developed countries, but bears great potential to narrow the digital divide in developing countries. The television's return channel might become the family's gate to the Information Society. For this to come true, however, digital receivers and return channel technologies must reach cost levels that enable mass penetration. In this essay, we discuss design decisions around the receiver that can drastically impact the success of interactive digital television in the developing world.

**Keywords:** Interactive Digital Television, Multimedia, Middleware e Set-top box

## 1 Introduction

When an author assumes the risk of entitling a paper with provocative terms, he has no better way to begin that paper than by explaining the provocation. After all, most readers might think that there is no significant difference between interactive digital television in developing and in developed countries—specially as regards the software/hardware design of associated gadgets. Furthermore, even if there were significant differences, it is a fact the most of the previous attempts to promote digital-inclusion by means of extremely low-cost gadgets

ended overcame by mass production of ordinary devices. Nevertheless, before going further with this reasoning, it is perhaps important to clearly define the context in which the term "interactive digital television" will be used in this paper.

Getting involved in more than one controversy in a single paper would not be wise, so no attempt to define "interactive digital television" will be made here. However, two facts about the technology surrounding iDTV seems to be evident:

- Interactive digital television, despite being digital and interactive, remains *television*. Users are now very well familiarized with their TV sets and any technological enhancement must preserve that relationship. Rebooting a TV set while watching a program, for instance, is an unthinkable situation.
- It will take some more time until television experts devise the possibilities embodied in the new technology and we get a broader notion of how "interaction" will take place. In the near horizon, interactive digital television seems to point out to a merger of standard television plus that what we now call the *Web*.

Indeed, there is no need to go deeper into the discussion about what interactive digital television is, or will be, in order to make the point about the very distinctive cases for developed and developing countries, if we accept the idea that, for now, interactive digital television is being built on standard television content plus services derived from those currently available on the *Web*:

In **developed** countries, the microcomputer/Internet revolution took place mostly in the 90s. Nowadays, much of the population of those countries has plain access to the *Web* at home, workplace and public places (e.g. schools, libraries, etc). Because it was born earlier, the *Information Society* in developed countries evolved relatively independent from the digital television scenario. Indeed, even personal communication and Internet access developed as two distinct industries that only now are showing signals of true convergence. In this scenario of independent media (i.e. TV) and communication (i.e. *Web*) services, developed countries underwent the digital television race looking for high added-value services, such as high definition video and audio, intelligent video recorders, and a large set of portal-like applications. This makes sense, since this industry targets customers who have already acquired, at least, a TV set, a microcomputer and a broadband access to the Internet [Duncan(2006)]. Aggregating value to the television is a logical way to attain mass marketing conditions.

On the other hand, the reality of **developing** countries is quite different: television is mostly ubiquitous, but the population of these countries still have limited access to the Internet [Crampton(2006), Monte(2006)]. Indeed, thinking in terms of iDTV, the *return channel* is by itself a big challenge for countries with deficient telephony infrastructure. In this regard, investments are being made in power line, satellite, and cell communication, which might provide answers to localized questions.

Nonetheless, supposing that a return channel infrastructure is on the way for developing countries, iDTV becomes the most concrete digital-inclusion mechanism available at short-term. A low-cost, web-enabled set-top box could be the pivot of a plan to catapult developing countries directly into the digital society of today, bypassing the microcomputer+modem lap. Such a gadget, besides acting as the traditional TV gateway, could be the gateway to e-government, e-learning, e-health, and other e-services to come.

Of course, for these vague ideas to become a concrete plan, many other variables must be considered besides the return channel. For instance, there are concerns that the low-resolution TV sets now in use will not be able to display the *Web* as we know today[1]. The most critical variable, however, seems to be bound to the cost of the set-top boxes that will have to be installed in millions of houses. And this is exactly the point in which iDTV strategies for developed and developing countries differ the most: while developed countries are envisioning complex (and expensive) services to be the promoters of the new media, developing countries must seek for solutions that are extremely affordable.

Based on previous experiments, it is also reasonable to suppose that any heterodox proposal, like closed standards, incompatible content formats, specific marketing conditions, will not help in building an universal information society—at most they will create digital islands. In this context, this paper brings a reasoning about two technical approaches that could fulfil the requirements of iDTV for both developed and developing countries.

The reminder of this paper presents a reasoning about an API for iDTV, discusses the currently available technologies and finally presents the authors perspectives about the discussed matters.

## 2 An API for iDTV

As stated before, interactive digital television is yet in its infancy and a comprehensive characterization of its applications will not be available soon. By the time iDTV applications become commonplace, much of the hardware and software limitations of today will probably be overcame, so they should not be allowed to dictate any major guideline of application development. The significant points in delineating an Application Program Interface (API) for iDTV are what we now know about such applications and also the concern not to prevent foreseeable innovations. Some of the facts about iDTV applications we can list today are:

- They will be sent to set-top boxes along with TV programs, multiplexed in video and audio streams, but somehow confined as not to allow applications to impact the stability of ordinary television (watching will still be the priority for most users);

[1] Standard TV sets support resolutions of about 500 pixels, while the *Web* is now at roughly double that resolution.

- They will be executed in set-top boxes of different architectures and functionalities;
- Applications, at least initially, will have the traditional remote control as the main human interaction interface. Otherwise, typical devices of ordinary PCs, like mouse and keyboard, will be used;
- Some applications will require data to be sent back through a return channel in a way very similar to the *Web* applications of today;
- Some will explore concepts such as Personal Video Recorders (PVR) and will require large and fast storage;
- A few will challenge the establishment, possibly revolutionising the way we watch TV today.

Although limited, this small set of remarks about applications gives rise to some important issues for the design of a tentative API for iDTV. Extensibility, for instance, comes forth as one of the main requisites, so that the operating system (or middleware) implementing the API will be able to evolve by improving its services and also by adding new services. The interface of such a *run-time support system* for interactive digital television could be seen as a kind of *Interactive Digital Television Language* (iDTV-L), and thus a first consideration about its design could be whether that language should be a *imperative* language or a *declarative* language[2].

Having our iDTV-L to be a **imperative language**, such as those we currently use to develop software, would allow iDTV application developers to "program" the set-top box in order to execute tasks it was not initially devised for. In this way, application developers could simply ignore the high-level services provided by the API and implement their own services, without any aid from the original set-top box manufacturer. Not even an operating system upgrade would be necessary. Nonetheless, a imperative language of this nature would require a complex interpreter/compiler to be part of the native set of services provided by the API. This might be a major disadvantage as it probably prevents the implementation of set-top boxes based on the premises of embedded systems, guiding them towards the realm of ordinary personal computers. The impact in cost such a design decision would have would probably hinder the use of the API in developing countries.

Alternatively, our iDTV-L could be a **declarative language** that would allow us to specify "what" is to be done by the STB without being concerned about "how" it will take place. The big advantage of a language of this nature comes from high-level elements such as: `TV Program` (to be handled by the receiving-decoding-rendering subsystem); `URL` (to be handled by a web-browser);

[2]It is not a goal of this paper to approach the rich taxonomic discussion about formal languages. The term *declarative language* is being used here to designate *markup* languages such as `HTML` e `XML`, while *imperative languages* designate programming languages like C++ and JAVA.

`Program Tag` (to be handled by a PVR for recording); etc. A linguistic set of elements of this kind would probably allow for a slimmer API implementation, since each OS could implement it considering the particularities of the set-top box architecture. Families of set-top boxes could implement specific subsets of the API, ranging from simple DTV receivers to full-fledged iDTV devices. However, any eventual modification or inclusion of services would require a system upgrade and would have to be carried out in agreement with manufacturers.

In order to differentiate these two alternatives, perhaps it is necessary to go one step deeper in our decision tree. On the **imperative language** branch of the tree, we will soon find a decision of whether the language should be compiled to each STB architecture or whether it should be interpreted, thus bringing about a virtual machine. Translating our iDTV-L to **real machine** code would show a series of advantages, specially as regards resource utilization, and consequently would foster the design of receivers in the realm of embedded systems. Nonetheless, this decision would also imply in TV programs being recompiled for each STB architecture. Considering the diversity of architectures commonly deployed in embedded systems, this branch is likely to be ruled-out. On the other hand, translating iDTV-L to **virtual machine** code would eliminate this problem, but not without bringing about additional costs. A virtual machine demands bringing a good fraction of the compilation system to run-time.

Thinking about a imperative language for a virtual machine without remembering JAVA is virtually impossible nowadays. But could JAVA itself be our iDTV-L? Could it be the API on which the future generation of TV programs will be developed? The fact is that, like JAVA or not, few people would give us credit if we started a discussion about a "new" imperative language at the level of elementary computations for a virtual machine. On the other hand, a higher-level imperative language would suffer from the same deficiencies of the declarative language approach: modifications or additions would require coordinated upgrades. Therefore, the decision tree about the nature of iDTV-L we have been building meets a crucial choice: the imperative language JAVA or a novel declarative language.

Apropos, JAVA is already deeply entangled in the development of some current DTV commercial initiatives, such as MHP [Institute(2003)] and DASE [Committee(2003)]. Indeed, this might be the appropriate choice for developed countries, because, while accepting the higher costs associated to the run-time support system of JAVA, they also bring about all the advantages of adopting a widespread imperative language, including easy access to skilled human resources. Adopting JAVA as it is today for the iDTV-L, however, could put the costs of iDTV at prohibitive levels for developing countries. As an example, consider a video-on-demand (VoD) application written in JAVA and executed in a standard PC: besides the application itself, we will find a huge operating system kernel, a virtual machine (JVM), a set of run-time support libraries (*classpath*) and a media framework (e.g. JMF). The overhead resulting from this architecture is so high that only high-end PCs are able to satisfactorily execute the application. Non-functional requirements, such as response time and quality of service can

hardly be met in such a scenario.

Going back to the context of declarative languages, the same VoD application could be easily described by a *complex element* that would include information about servers, content and protocols. The description would subsequently be parsed by a sort of server that would invoke native OS services accordingly. This approach would certainly incur in far less overhead than the previous JAVA stack and perhaps could be implemented as a low-cost embedded system. It is important to remember, however, that this approach would constrain iDTV programs to that what is made available through the API. Any extension would imply in upgrading the OS on the STB, thus binding content developers with equipment manufacturers in a delicate matter.

A declarative language could handle this limitation by offering lower level elements, such as `TCP Connection`, `File`, `Graphic Window`, and others. This would considerably increase the complexity of the language interpretation engine, but this, when compared to JAVA, would probably not be enough to exclude the alternative[3]. More important, perhaps, is the impact such an approach would have on security. A low-level API would give access to basic OS services, including storage management and communication via the return-channel. If not addressed consistently, this could open the way to a whole new era of cybernetic crimes, bringing the actual chaos of personal computing into the realm of television: viruses, interruptions, lost of stored programs, privacy corruption, etc. One could argue that JAVA, for being a low-level imperative language (if compared to our high-level declarative language, not to C or assembly) is exposed to the same problems. This might be true in theory, but in practice, JAVA security issues have been consequently explored by Web application developers in the last decade, so that it would not be fair to suppose that both strategies would be exposed to the same level of security flaws.

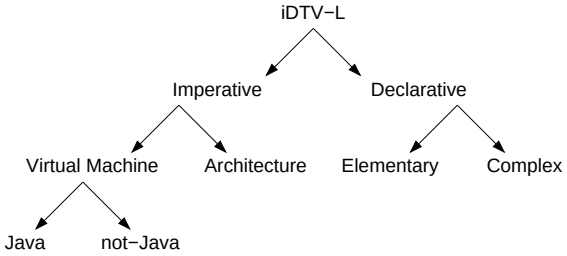


Figure 1: Decision tree for an hypothetical iDTV API.

At this point, the decision tree about an API for iDTV has only two strong branches: JAVA and a novel declarative language with high-level elements. JAVA is a well-know imperative language, what makes additional explanations unnecessary, but what about this hypothetical declarative language? What would it look like? Well, considering the Web of today, we could think about this lan-

[3]In a JAVA run-time environment, the interpreter (i.e. bytecode parser) is not one of the most significant causes of overhead [Rayside et al(1999)Rayside, Mamas, and Hons].

guage as a collection of XML DTDs and Schemas. An iDTV program would thus be described by an element whose attributes would include, for instance, available audio and subtitle languages. The OS would parse that tag and react to it invoking services to notify the user about the multilingual features of the program. Other tags could be multiplexed along with the program's audio and video streams, for instance, to indicate that there is a chat room for the program, to supply program-related URLs, to setup a poll, etc. Whenever such a high-level tag is parsed by the OS, one or more native services are invoked.

In this scenario, HTML would certainly be one of the constituent DTDs of our iDTV-L. Furthermore, interaction events, like a mouse click, could be handled by script languages embedded into the iDTV-L program in a way similar to JAVA-SCRIPT embedded in HTML. On the broadcaster side, the analogy would take PHP instead. It is important to notice that this level of Web accessibility is now available in a variety of complex embedded systems, such as cell phones and PDAs. A consistent re-engineering of those systems is likely to succeed in bridging the gap to low-cost iDTV terminals.

# 3 Currently Existing Approaches

The first approaches to realize an DTV API resulted in proprietary middlewares such as OPENTV, NDS, CANAL+, POWERTV and MICROSOFTTV that shaped a vertical market, in which the televiewer depended on the same company to provide the service and the set-top box. Recently, open standards started to attract the attention of the industry and are now reshaping the market towards a more flexible scenery. This evolution is represented in the time-line in figure 2.

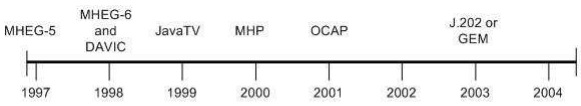


Figure 2: Evolution of DTV middleware standards.

Since the publication of MHEG-5 standard in 1997, many other open standards followed, like MHEG-6 and Digital Audio and Video Council (DAVIC) in 1998 and JavaTV in 1999. These last two plus Havi formed the basis of DVB Multimedia Home Platform (MHP) [Institute(2003)] released in 2000. After that, OpenCable Application Platform (OCAP) was developed based on MHP and, in 2001, all these standards started to converge toward Globally Executable MHP (GEM) [Institute(2005)], which was also standardized as J.200 by ITU-T in 2003 [Union(2005)].

The first open standards, like for instance MHEG-5, followed the declarative approach. Today, however, iDTV middlewares tend to combine both approaches, including support for declarative and imperative subsets. HMP 1.1,

for instance, supports HTML in its Internet Access profile, which is known as DVB-HTML.

Another middleware that supports both approaches is the brazilian Ginga [PUC-Rio(2007)], which is divided in GINGA-J (imperative approach), based on JAVA, and GINGA-NCL (declarative approach), based on the Nested Context Language (NCL) [PUC-Rio(2006)].

The official Chinese middleware for digital TV is designed around five layers [Hong-Guang and Shi-Bao(2004)], including a system layer with three blocks: API packages (creates an abstraction layer to the applications), JAVA-VM (run imperative applications) and web engine (run declarative applications).

Pablo Cesar proposes an extension to the GEM standard in order to support a declarative approach, namely XHTML, XForms, CSS, Synchronized Multimedia Integration Language (SMIL) and 3D graphics, thus improving the support for applications [Cesar(2005)].

In all this initiatives, the declarative portion of the API deliveries ready-to-use services that can be efficiently implemented on the native architecture of the set-top box; while the imperative portion of the API maintains a flexibility compromise with application developers. Nonetheless, even this kind of hybrid middleware, as far as the authors are concerned, are reported to have been implemented only in high-end gadgets with PC-class processors (i.e. 32-bit with MMU) and large storage capabilities.

# 4 Final Considerations

Interactive Digital Television seems to be following quite different tracks in developed and in developing countries: while it is more of a technological enhancement of ordinary TV for the first category—where most citizens already have Internet access, the second sees it as a concrete possibility to bring millions of digitally excluded citizens into the information society. Nonetheless, it is known from past experiences that isolated developing countries initiatives are unlikely to succeed, specially considering the obvious pressure from the global content industry. From the technological point of view, the solution would be an iDTV that has all the features demanded by consumers from developed countries while being affordable to consumers in developing ones.

In this paper, we presented a reasoning about the implementation of Interactive Digital Television in developing countries focused on the API that must be implemented in every set-top box in order to support iDTV programs. This reasoning guided us towards two major alternatives: adopting JAVA as a substratum for higher-level iDTV frameworks, or developing a new declarative language based on XML. The JAVA approach has its main advantage on the flexibility to develop new iDTV programs independently from the original API implementation, after all JAVA is an object-oriented language and higher-level frameworks can always be bypassed. The main disadvantage is the complexity of the associated run-time environment, which implies in higher-cost set-top boxes. From the XML approach, the main advantage comes from the possibility of a leaner implementation of the iDTV API, what could enable the production

of relatively low-cost terminals. The main disadvantage, however, is that such implementation would have to be done almost from scratch and would require explicit upgrades in order to support additional features.

The hybrid approach of combining declarative and imperative languages to build the iDTV API, which seems to be the direction for current open standards, is a recognition of the gap between flexibility and set-top box cost. However, initiatives to develop more efficient run-time support systems for JAVA, which could make it more comfortably fit in the embedded systems scenery are under way withing several research groups. For instance, the Kilo Virtual Machine (KVM), which is the reference implementation of J2ME by Sun Microsystems, requires a 32-bit processor and about 1/2 Mbyte of RAM to run an ordinary application, what is far too much for the desired scenario, but is already a fantastic advance if compared to the counterpart PC with WINDOWS/LINUX and J2RE[Rayside et al(1999)Rayside, Mamas, and Hons]. These approaches might give us the answer to many problems in the path of iDTV to digital inclusion in the developing world.

# References


[Cesar(2005)] Cesar P (2005) A graphics software architecture for high-end interactive tv terminals. PhD thesis, Helsinky University of Technology

[Committee(2003)] Committee ATS (2003) ATSC: DTV Application Software Environment Level 1 - Part 1: Introduction, Architecture, and Common Facilities

[Crampton(2006)] Crampton T (2006) In the developing world, broadband spans the digital divide. On-line, [http://www.iht.com/articles/2006/12/03/technology/btdivide.php] (Jan 30, 2007)

[Duncan(2006)] Duncan G (2006) Dsl dominates global broadband. On-line, [http://news.digitaltrends.com/article11921.html] (Jan 30, 2007)

[Hong-Guang and Shi-Bao(2004)] Hong-Guang Z, Shi-Bao Z (2004) An extensible digital television middleware architecture based on hardware abstraction layer. In: Proceedings of the IEEE International Conference on Multimedia and Expo, Taipei, Taiwan, vol 1, pp 711–714

[Institute(2003)] Institute ETS (2003) Digital Vídeo Broadcasting: Multimedia Home Platform Specification 1.1.1. France, v.1.1.2. rev.1. edn

[Institute(2005)] Institute ETS (2005) Digital Vídeo Broadcasting: Globaly Executable MHP Specification 1.0.2. France

[Monte(2006)] Monte F (2006) Brasil tem 4,7 mil conexões de banda larga, mas taxa de penetração é de 2,2%. On-line,

[http://wnews.uol.com.br/site/noticias/materia.php?id_secao=4&id_conteudo=5853] (Jan 30, 2007)

[PUC-Rio(2006)] PUC-Rio LT (2006) Main Profile Specification of NCL 2.3

[PUC-Rio(2007)] PUC-Rio LT (2007) Middleware ginga. On-line, [http://www.ginga.org.br/] (Jan 30, 2007)

[Rayside et al(1999)Rayside, Mamas, and Hons] Rayside D, Mamas E, Hons E (1999) Compact java binaries for embedded systems. In: Proceedings of the 9th NRC/IBM Centre for Advanced Studies Conference, Toronto, Canada, pp 1–14

[Union(2005)] Union IT (2005) Harmonization of procedural content formats for interactive TV applica tions