# An Experimental evaluation of sNCL compared to NCL

Jamile Santos[*]
Instituto Federal da Bahia
Rua Emídio Santos, s/n,
Barbalho,
Salvador, Bahia, Brazil
jamile.ssnts@gmail.com

Manoel C. M. Neto[†]
Instituto Federal da Bahia
Rua Emídio Santos, s/n,
Barbalho,
Salvador, Bahia, Brazil
manoelnetom@ifba.edu.br

Renato Novais[‡]
Instituto Federal da Bahia
Rua Emídio Santos, s/n,
Barbalho,
Salvador, Bahia, Brazil
renato@ifba.edu.br

## ABSTRACT

Nested Context Language (NCL) is the official declarative language for the Brazilian digital Television. It is widely used to create multimedia documents and interactive applications for digital Television. The NCL is known for its space-time synchronization between media objects, adaptability, and support for multiple devices. Furthermore, it is based on XML and NCM model (Nested Context Model). Although NCL is considered simple, the similarity to XML gives it more verbosity and it has been the topic of debate whether or not it is easy to use and understand. In order to simplify this language, reduce verbosity and ease the use sNCL was created. sNCL is a domain specific language centralized on decrease the verbosity in NCL documents. Moreover, its goal is to provide better use and productivity for developers. The objective of this work is to run a controlled experiment and through statistical analysis to evaluate if the sNCL is easier to learn, less verbose and simpler compared to NCL.

## Keywords

NCL, sNCL, digital television, experimental evaluation, usability, verbosity

## 1. INTRODUCTION

The digital television (DTV) in Brazil represents a important advance on telecomunication sector. It possibilites interactivity between users and application, in the communication process. It provides high quality sound and image. Moreover, it proposes new services not existing in conventional analog television.

[*]Student from the course of Analysis and Systems Development (ADS).

[†]Ph.D. in Computer Science, Professor and Researcher from the course of Analysis and Systems Development.

[‡]Ph.D. in Computer Science, Professor, Researcher and Advisor from the course of Analysis and Systems Development.

Some of the applications that provides these services and interactivity are developed in specific domain languages (DSL). DSL are based on the relevant concepts and characteristics of a given domain. Thus, they can provide an adapted notation to the family of applications of this domain and facilitate their creations. Therefore, DSLs are defined as small language, usually declarative, that describe domain applications [19, 20]. It makes them a simple type of language to use since much of the knowledge about the domain is implicit in its structure. A widely used DSL language for digital TV applications is NCL. Also, for the applications to work in the digital televisions a middleware to mediate the communication between hardware and software is necessary. The middleware used for the Brazilian television system is Ginga.

Ginga is an open source middleware being a intermediate layer that provides the interactions between the software applications and hardware. Moreover, it allows the development of interactive NCL applications for DTV independently of the hardware platform, and provides categories, Ginga-NCL and GingaJ, but in this paper we focus on Ginga-NCL.

NCL is designed to facilitate the modeling and authoring of Digital TV applications by content producers. It is the standard language of the Brazilian Digital Terrestrial TV System (SBTVD-T) [10] and ITU-T Recommendation H.761 [12]. NCL is a high level declarative and domain-specific language for specific based on XML and Nested Context Model (NCM) used to develop digital television applications development [8]. As an XML application, besides to other factors explored in this article, NCL introduces a high verbosity that is not quite useful. This verbosity offers the false idea that NCL is not as powerful as imperative languages such as Java, C, or Lua. As a result, it does not attract as many developers as the more traditional languages. The new DSL aims to be an easier language to use. To achieve this object, the implementation of sNCL was based on an analysis of the usability factors of NCL combined to the use of CDN. The results obtained from this analysis have a direct influence on the design decisions of sNCL.

This work presents an experimental evaluation about sNCL addressing the main aspects of the new DSL language comparing to NCL. The experiment is explained in details in section 3.

## 2. THEORETICAL FRAMEWORK

This paper focuses on the experimental evaluation of sNCL compared to NCL. Section 2 discusses about the Brazilian

Digital Television System, Ginga middleware and its architecture, NCL, and sNCL language. Section 3 presents the related work. Section 4 explains in details about the experimental evaluation, such as study hypothesis, the experimental object, participant's characterization, task design, experimental procedures and pilot study. Section 5 presents the results of the experiment along with the statistical tests. Section 6 discuss about the lessons learned from the experiment covering aspects such as productivity, correctness and verbosity. Section 7 validates the evaluation, and finally section 8 presents the conclusions about the study and its impacts.

## 2.1 Brazilian Digital Television System

Television in Brazil has an important role for Brazilian nation, being a source of information and entertainment for many people. According to [1] the television is already present in 94.8% of households. This fact indicates that the TV has a great importance for the Brazilian population.

The advent of digital television in Brazil marked a new phase of technological evolution that has generated major changes in various aspects of society. Digital TV has brought interactivity, advances in access to information, digital inclusion, improved quality of sound and image transmission. This technological advance allows the transmission of images in high definition and audio, providing a similar quality to that found in a movie theatre. But these improvements in the TV are not the only benefits that the digital television has to offer.

The major difference is the connectivity with the internet, a fact that introduces new uses for this device, which not only to watch audiovisual content. A range of interactive services became part of the functionalities of television associated with the internet, such as access to videos on demand, multiplayer games, government services, banking services, email accounts, access to shopping networks, social networks [5].

The transition from analogue to digital TV was projected by the Brazilian government since the beginning of the 2000s when it was already planned to suspend analogue signal in 2016 [5].

The first official broadcast of Brazilian digital Television took place in 2007, and was created based on ISDB-T (Integrated Services Digital Broadcasting Terrestrial). The ISDB-TB is an adaptation of the ISDB-T with added technologies developed by researchers from Brazilian universities [10].

The SBTVD (Brazilian Digital TV System) provides high-quality images and sounds without major interference compared to the analogical system. Besides theses improvements on the Brazilian TV System, the impact of digital TV is even greater because it allows flexibility and interactivity. This flexibility is related to the ability to expand its functions through applications that are built following a reference standard [10]. These applications, which are computer programs, should be independent and able to run in any hardware platform and operating system, hence to provide that, a middleware is needed. The middleware is a component that regulates the relationship between content production and manufacture of receivers, which makes it extremely important [10].

For the implementation of the DTV it was necessary to develop a middleware that would be able to encode the signals. Ginga is the middleware for the Brazilian digital system which was entirely developed in Brazil. It is not similar as others middlewares for terrestrial digital TV, the standard middleware of the Brazilian DTV (ISDTV-T) has its own environment inspired by XML application language.

As reported by [9], Ginga supports both declarative and procedural applications. The declarative applications should be based on a declarative language, that is, a language that emphasizes the declarative description of the problem [3], such as XML, HTML and NCL. On the other hand, the procedural contents is not based on declarative language, for instance Java.

The application environment is categorized in procedural and declarative as well. The declarative environment is called Ginga-NCL, and the procedural is Ginga-J.

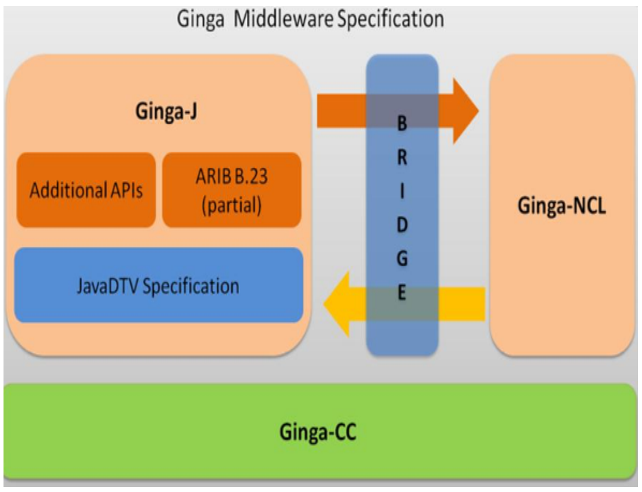


**Figure 1: Ginga Architecture**

The Ginga architecture is modular as shown in figure 2.1, and it can be applied to other television systems, such as satellite or cable [3]. Although Ginga is divided into two categories, the DTV (Digital TV) applications can be a hybrid, supporting both procedural and declarative. As stated by [9], Ginga-NCL is a logical subsystem of the Ginga System that runs NCL applications, and Ginga-J.

The main component of Ginga-NCL is the declarative content decoding engine [9]. Unlike Ginga-NCL, the important component on the Ginga-J is the engine execution of the procedural application formed by a JVM (Java Virtual Machine).

The Bridge component on the architecture is a way of communication between Ginga-J and Ginga-NCL. The Ginga-CC (Common Core) is composed of decoders and common procedures of content to get content transported in MPEG-2 transport streams (TS) and through a return channel [3].

## 2.2 NCL - The Nested Context Language

NCL is a declarative language for authoring of hypermedia documents based on NCL (Nested Context Model) [6]. The Nested Context Language was created by researchers from Pontifical Catholic University of Rio de Janeiro. The NCL language model is intended not only for declarative support for user interaction but for spatial and timing synchronization in its most general form, treating user interaction as a particular case.

NCL is a declarative domain specific language based on NCM model and XML language. It is the DSL standard language for the digital Brazilian system TV [2]. Although it is based on XML, NCL has a severe separation between content and structure [9]. An NCL document just defines the structure and relation of media objects, time and space.

The basic NCL structure is the following elements: <ncl>, <head>, and <body>. The <ncl> tag is the parent of the head and body elements.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Generated by NCL Eclipse -->
<ncl id="new_ncl_file" xmlns="http://www.
    ncl.org.br/NCL3.0/EDTVProfile">
  <head>
    <regionBase>
      <region width="1060" height="1029" id
        ="rgtv">
        <region left="448" width="1024" top
          ="156" id="rgvideo">
        </region>
      </region>
    </regionBase>

    <descriptorBase>
      <descriptor id="dvideo" region="
        rgvideo">
      </descriptor>
    </descriptorBase>

  </head>

  <body>
    <port id="pbegin" component="video"/>
    <media descriptor="dvideo" src="media/
      video.mpeg" id="video">
    </media>
  </body>
</ncl>
```

**Listing 1: Example of NCL code**

In the <head> element are the elements related to the code reuse, that are defined as bases. For instance: <descriptorBase> and <descriptor>. These structures allows the code reuse of its children elements [9].

The <body> element has the structure that specifies the kind of content application, the medias and the relationship between them [13]. Its children elements are: <port>, <attribute>, <media>, <context>, <switch> and <link> [9]. The media element is responsible for define the type of content and its the location.

Some of these elements are responsible for the presentation, and other for the relationship between medias. For instance <region>, <transition>, <descriptor>, <context> and <media> is in charge of media presentation.

The <connector> and <link> relates the medias, they connect the different types of medias. In NCL language, the <link> and <connector> elements are responsible for synchronizing the medias. Hence, they are common elements in an NCL document. The <connector> provides reusable models for the causal relation between elements and its functions. These functions follow a conditional and actional model, which means that the conditions are established to be a model.

Other elements that are useful for reuse purposes are three types that precede the <media> element: <region>, <descriptor>, and <transition>. The <region> element defines the screen area where the media will be shown. The <descriptor> defines a model for the transition effect, there is a chain of elements, and it is a referred region by many descriptors, which are referred my many medias. However, the first two elements are in the body while the media is on document's body. this increases the error probability because the media can refer to a not declared descriptor.

The developer also can declare only the <media> element, defining all properties within it. However it increases its diffusion, and the application loses the reuse properties. The segregation between region and descriptor increases the reuse although it creates a hidden sequence of dependencies. For instance, the <region> may be referred by many <descriptor> and the descriptors by many media, which decreases the visibility because it is required the utilization of two or more elements to become a visible media.

There are many studies about the NCL verbosity, and how to reduce it. Between these studies, there are two approaches based on template, TAL (template Authoring Language) and Luar. These works propose a language for development of multimedia applications, and divide its authoring application into two steps, the templates authoring, which is an incomplete document that describes the application defining the used media's objects.

## 2.3 Simpler NCL

sNCL (simpler NCL) is a domain-specific language (DSL), whose purpose is to solve or minimize problems that NCL has, introducing a simpler way to develop multimedia applications for the Ginga-NCL middleware [9] and to provide to professionals, especially programmers, a new approach, which is not based on XML. It aims to be less verbose, more compact and it does not require a lot of writing from the author to express an idea. The syntax is inspired in Lua, which is used as script language and extension for NCL, such as NCLua, that allows using Lua code as objects in NCL application [13].

The similarity between sNCL with Lua language, as mentioned, also it is used for digital TV application, provides a familiarity for developers of this new language [9]. The sNCL authors state that for not being based on XML the verbosity decreases, and for those developers who is familiar with imperative languages the understanding is easier than NCL.

The NCL elements are used on sNCL, such as media, area, context and its attributes like the reserved words, that determine the kind of the data, as int, float [2]. The figure 2, shows the declaration in sNCL.

```
region beginReg
  width = "100%"
  height = "100%"
  zIndex = "10"
end
media photo1
  right = "10%"
  source = "media/foto01"
  area aphoto1
    begin = "5s"
    end = "10s"
  end
```

```
end
```

**Listing 2: Example of NCL code**

It is an important highlight that the applications written in sNCL are compiled as NCL, and then the NCL document is run. There is no a specific player for sNCL yet, and it does not plan substitute NCL, however, it aims to work as an intermediate tool.

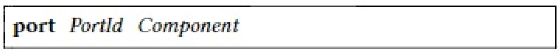
In NCL, the elements <head> and <body> provides reuse of elements however it introduces a problem of hidden dependencies both on presentation area and relationship part. In sNCL, these elements aforementioned are removed as well the <descriptorBase>, <connectorBase> and its children.

In the analyzed applications, descriptor elements mostly are used to call a media and its region. Hence, with its removal, the properties become defined within medias.

The <region> element is kept because it has many uses. Medias can refer to the same region, but there is no need for the declaration of its parent <regionBase>. Besides that, there is a <importBase> that may be used to import documents with .ncl and .sncl extensions.

In the relationship area, the <connector> element was discarded from the language. Once the separation of elements relation affects the Viscosity Dimensions, Visibility and Propensity to errors, as seen earlier. Only the link element is kept allowing the author to define the relationship between medias.

The sNCL syntax resembles a imperative language, where the elements become tokens (reserved words) of the language. Because of that, once an element is declared, it is used as reserved word followed by its attributes. The <port> element has the id attribute and component. Figure 3 shows the syntax of the port element. The elements follow a syntax standard, expect the <link> element. It is easy to notice that the sNCL structure differs from the NCL elements. The <link> element loses its xconnector and id attributes.



**port** *PortId Component*

**Figure 2: sNCL of port element**

Considering that, the first is used in NCL to specify which <connector> the <link> is being used as relationship definition, and the second one is used to reuse purposes of the element. However, no element in NCL can refer to <link>, therefore the attribute id becomes a useless attribute. The xconnector was removed because in sNCL there is no more than one <connector> element to be referred.

The elements follow a syntax standard, expect the <link> element. It is easy to notice that the sNCL structure differs from the NCL elements. The <link> element loses its xconnector and id attributes. Considering that, the first is used in NCL to specify which <connector> the <link> is being used as relationship definition, and the second one is used to reuse purposes of the element. However, no element in NCL can refer to <link>, therefore the attribute id becomes a useless attribute. The xconnector was removed because in sNCL there is no more than one <connector> element to be referred.

Besides the reason aforementioned, the reason for removal of <connector> the element is that for being a DSL, the domain knowledge is implicit in the language structure. When the link element is declared, the sNCL compiler infers on what is being a descriptor, therefore removing the need to declare the connector.

The syntactic analysis of sNCL document is done using the LPeg (Lua Parsing Expression Grammar) library, that allows the definition of rules and analyze the input document as a syntactic tree for development of multimedia applications for the terrestrial Brazilian digital TV system. The sNCL authors aim not only to reduce the final document verbosity but also simplify the application development. The development process of sNCL follows the development techniques test oriented. For the learning process of sNCL, it is recommended to start with simple examples and then increment adding new elements. Those who are not familiar with XML language. Its imperative structure allows the user to understand better the commands.

## 3. RELATED WORK

There are several papers that investigate specific domain languages. Some of these articles perform empirical studies about aspects of language, such as [4]. Some of these works perform empirical studies about aspects of language, such as [4], that presents an empirical research on the use of DSL on an industry. Another example of empirical study is [11], however, in this study, the authors focus on specific DSL language, the NCL.

On the first study cited [4], the authors investigate some aspects of DSL, such as usability, reliability, and learnability through a survey with 18 users. The questionnaire focus on to measure the success factors about DSL language.

This work is related to our study because it performs an experiment with survey that investigates aspects like usability and learnability of DSL. It gave us the base to create our questionnaire.

Moreover, [11] is also an empirical study about NCL which is specific type of DSL language, and its goal was to get usability indicators of NCL from users. Qualitative and quantitative questions were made for 220 students from introductory courses about application development for digital television. The focus of the courses was production of digital television content.

The students completed two forms, one referring to their profile and another with questions related to the course. The conclusions of this study show that for students with technical training in computing, NCL is effective for developing applications for digital television, but presents some challenges related to usability that must be improved to become an efficient language.

According to the study performed by [11], students found that although it is a verbose language, NCL is not difficult to learn. Hence, the authors suggest the creation of editing tools for NCL, or creation of mechanisms in the language to reduce verbosity.

This work is relevant to the present study because it addresses aspects such as verbosity and usability, just as our experiment expects to measure.

## 4. EXPERIMENTAL EVALUATION

The main objective of this experiment is to evaluate quantitatively and qualitatively the verbosity, productivity and learnability of the sNCL language compared to NCL. The

**Table 1: STUDY HYPOTHESES**

| H | Description |
|---|---|
| H1 | sNCL provides to the programmers a superior productivity than NCL. |
| H2 | sNCL is less verbose than NCL. |
| H3 | sNCL is easier to learn because it is simpler than NCL. |

goal is to analyze if the sNCL language offers advantages with its use. The experiment is motivated by a need to understand the differences between NCL and sNCL in terms aforementioned.

We performed an experiment using the evaluation techniques based on [14] with both programming languages following a set of designed tasks that will be described throughout the following sections. The goal is to investigate if the sNCL meet its purpose as stated by [2, 13].

The perspective is to know if there are substantial differences in development performance, learnability and verbosity between those who use sNCL compared to NCL users. In this section, we describe in details the experimental evaluation.

## 4.1 Study Hypothesis

The authors of sNCL defend that the language is simpler and its main objective is to minimize the problems presented by NCL. According to [13], the main goal is to introduce a simpler way of developing multimedia applications for the Ginga-NCL middleware.

Like NCL, sNCL is a domain-specific language (DSL), however, it is based on Lua, an imperative language, whereas NCL is based on XML, which makes NCL more verbose. The article [13] argues that because sNCL is less verbose, users use less code to express an idea using sNCL. Also, it is more succinct. It is worth mentioning that its authors did not create the language with the objective of replacing NCL, but rather, offering a complementary tool for use in the development of multimedia applications.

The first hypothesis of this work (H1) says that sNCL provides to the programmer higher productivity compared to NCL. The second hypothesis (H2) states that sNCL is less verbose than NCL. The last hypothesis (H3) says that sNCL is easier to learn because it is simpler than NCL. The null hypotheses are the opposite of the above-mentioned ones, as shown on Table 1.

## 4.2 Experimental Object

The object of this experimentation is sNCL, a DSL programming language for development of multimedia applications. Also, it is analyzed its the qualities compared to the NCL language.

Some aspects were chosen to examine of this language in order to compare to NCL and verify whether the hypotheses are proven or refuted. We aim to analyze the following points: (i) productivity; (ii) verbosity, and finally (iii) learnability.

## 4.3 Participants

The experiment was performed with 29 participants, who agreed to participate voluntarily. The participants are undergraduate students from the Federal Institute of Bahia (IFBA) primarily from the course Systems Analysis and Development. The students have different backgrounds or experience related to computing and programming languages. They also have different levels of knowledge in programming languages.

For this study, all the information about the experiment was provided, such as the purpose of the experiment. However, details about the hypotheses stated by this work were not provided to the participants. They were advised about the possibility of quitting their participation at any time without penalty. Moreover, all the data collected from the participants, such as personal information, opinions, and answers were treated as confidential.

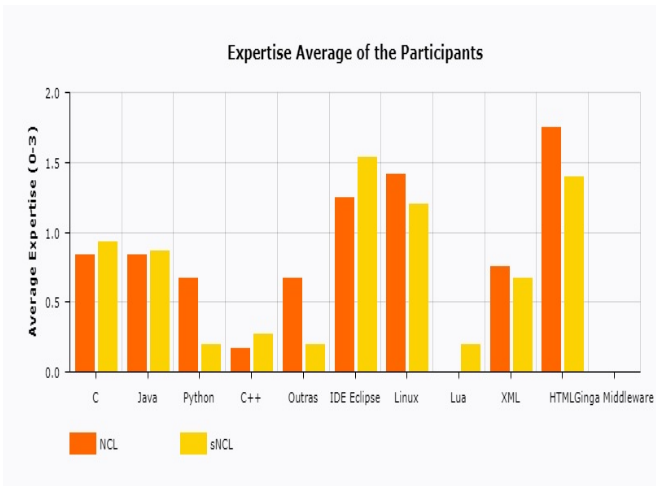


**Figure 3: Technical Knowledge**

We provided three sessions of training for each language. During the planning of these training sessions, the knowledge levels of the participants were taken into account. The participants were separated by the level of expertise so they could attend to the appropriate training session. Those with a lower level participated in the training with the class compatible with their level. This strategy was adopted, in order to levelling the participants.

We divided the participants into groups, where each group received a training about either NCL or sNCL. During the selection we tried to balance the knowledge of the group so that the participants with similar level of expertise could be together. It is worth to mention that the participants did not attend more than one session. Each student could participate only once, also they were asked to not comment on the experiment with others students in order to avoid prior knowledge about the study.

Also, the participants received a material containing the basics concepts about the language and it was available during the experiment. It is noteworthy that no previous knowledge about any programming language was a requirement to join this experiment. Because one of the objectives is to observe how the participants learned the sNCL language.

Before starting the experiment, we asked each student to fill-out a characterization form, where they could attest their experience and level of technical expertise on the programming languages and concepts. This was used to grouping them as mentioned.

It is important to emphasize that the participants were not

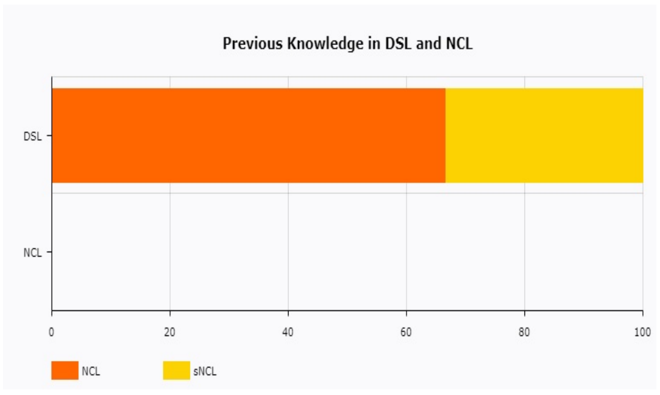


**Figure 4: Previous Knowledge**

familiar with the experimental object, as shown in Figure 4. However, this was not a requirement to join the experiment. The average level of expertise in different technologies of each group is shown on Figure 3. It is possible to note that there is not a large difference between the two groups. As shown in Figure 5, the participants of each group have similar expertise mean. The expertise level of each participant ranged from 0 (no knowledge) to 3 (Expert) on a certain topic, being 1 is equivalent to low and 2 to moderate. We extracted the results of knowledge based on the sum total from the range above mentioned.

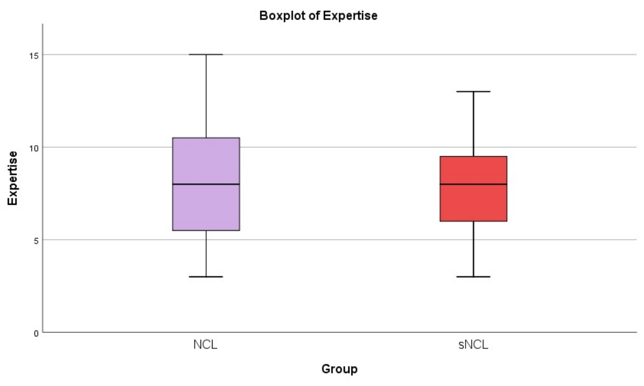


**Figure 5: Boxplot Expertise by group**

## 4.4 Pilot Study

A pilot study was conducted prior to the experiment with the purpose of identifying certain issues in its procedures, tools used, or in the development environment. The intention was to reduce the possible problems that could happen during the experiment as much as possible. Also, the pilot study help us to set a estimated time for the training sessions.

Two participants were chosen to perform the pilot study. Each of them was assigned to attend either NCL or sNCL. As the participants in the final experiment, they also filled out the consent and characterization form so that we could analyze their expertise level. It is worthy to emphasize that these two participants did not take part of the final experiment. The pilot study allowed us to enhance the assignments' description, objectives, and level of difficulty. Moreover, it was also essential to tune the experiment timing in 100 minutes.

## 4.5 Task Design

We tried to determine a comprehensive task for the experiment where the groups could answer applying the concepts based on the session training and the provided material about the language. The task definition was based on basic concepts of both NCL and sNCL languages. The goal was verify the learnability, productivity and verbosity from both language.

The task asked to the participants was to develop a simple application using few concepts of the languages. The simplicity of the task was chose because the time limited time during the sessions. Through this task, it is possible to introduce the basic concepts about the NCL and sNCL's equivalency. It is worth mention that the proposed task can be done in both languages.

The task had this design because: (i) the students are not familiar with the programming languages; (ii) it was set up a time limit for the experiment in order to avoid participant fatigue; and (iii) finally, it does not favor either programming languages for the execution of the tasks.

As we only proposed one task, we measure the results based on each element correctly declared and if it meets its purpose according to the task. The participant could get a total or partial point for each element.

The maximum number of possible points is 12 points, and we considered all participant's answers that filled out the correct start time and end time information asked during the experiment.

## 4.6 Experimental Procedures

The experiment was performed in six training sessions, three for each group. To avoid any problems during the experiment all sessions were supervised.

The experiment occurred in two different moments. At first moment a group of participants received training about the language. After the the sessions, online surveys were given asking qualitative questions about both languages.

The necessary tools were installed and tested previously to the experiment, such as IDE, plugins, virtual machine. Eclipse IDE with NCL plugin was used for the NCL group, VMware Workstation Player, a virtual Ubuntu server machine with Ginga Set up box already configured. For sNCL training session, the software used was Lua and LuaRock and sNCL library. We chose Linux as operating system to setup environment to run the experiment. The version used was Kubuntu 17.10.1, and in order to reduce the setup time in the preparation of the environment, a custom image of Linux system was made. This custom system has been installed on USB drives for each participant.

The participants received all the information about the experimentation and its methods, and signed up a form consenting the participation. The experiment itself was run in university laboratories where the entire environment was previously set up.

The training session for the experiment last 100 minutes. This time was defined according to the pilot study (see Section 4.4). The experimental procedures were:

1. The NCL Eclipse plugin tools used to help on the de-

velopment environment. Hence, all the students could have the same IDE and tools during the experiment;

2. The students filled out the characterization and consent form and were divided based on the information that they provided. The criteria to separate them into two group was the background and previous knowledge;

3. A training session of 50 minutes was organized in order to present the basic concepts of both languages and to show how both tools work. Additionally, we provided the purpose and goals of the study. After the explanation about the language concepts, we gave an example using the language and the tools, in order to make the students feel comfortable with the software used in the experiment. They were encouraged to address any questions or concerns during the training session. Thereafter, we gave to the participants an assignment where they need to use the concepts learned and use the tools. Moreover they were asked to measured to time of begin and end of the task;

4. The experiment itself was run in university laboratories where the entire environment was set up. The time was measured by considering the difference between the final and initial time of each task. The time of any interruption during the task execution was deducted from the total execution time;

5. We analyzed the results. Correctness was measured by comparing the correct answer model with each student's answer. We converted the responses into quantitative values according to 10. The entire assignment worth 12 points, and each correct part worth a set of points. For each correct answer the point for the section was give, however the wrong answer did not count negatively;

6. Finally, we applied a feedback questionnaire. It questionnaire was applied on-line in order to guarantee a better experience for the participants, and to avoid discourage from them. In the questionnaire they were asked about the training session, language, opinion about the understanding and the task;

## 5. RESULTS

This section discusses the experimental results. The statistical tests used are described throughout this section. On subsection 5.1 we discuss about the times spent on the experiment as well its results and implications. On subsection 5.3 we discusses about the correctness of the experiment and its relation to the time spent.

### 5.1 Statistical test

We performed statistical tests to accept or reject the hypotheses aforementioned Table 1. The tests were conducted in three phases. First of all, the descriptive statistics analysis was performed with the calculation of mean, minimum, maximum and standard deviation value. Also we analyzed the data to verify if the samples distribution were normal and with equal variances. To ascertain these information, we applied the Shapiro-Wilk and Levene tests. After verifying the normality and distribution of the data, we chose two tests, t-test and Mann-Whitney to be applied relying on the fact whether the data were parametric or not. The t-test was chosen because it is a parametric method used to test whether a set of samples comes from normalized distributions. It is used to test the null hypothesis of that all populations have equal distribution functions against the alternative hypothesis that at least two of the populations have different distribution functions. In order to avoid cumbersome fixes and possible errors, improving the validation process of the statistical test, we use IBM SPSS Statistics and R. The variables measured were time, correctness (points), verbosity (length) and expertise, productivity. Thus, we used the parametric t-test to analyze the hypotheses H2, H3. The sample distribution of productivity was not normalized, consequently we select the non-parametric Mann-Whitney test to analyze the hypothesis H1. The confidence level used on this study was of 95% ($\alpha = 0.05$). The third phase was to evaluate the hypotheses based on the tests previously done.

**Table 2: Statistical hypothesis testing**

| | t-test p-value | Levene | Mann Whitney p-value |
|---|---|---|---|
| Time (minutes) | 0.775973 | 0.224 | |
| Correctness (points) | 0.45135 | 0.675 | |
| Verbosity (length) | 0.000033 | 0.019 | |
| Verbosity (lines) | 0.000106 | 0.004 | |
| Productivity (length/time) | | 0.756 | 0.009 |

#### *5.1.1 H1: Productivity*

The productivity of both languages was measured from the ratio of the size of the code by the time spent during the experiment. As we can see in the Table 3, the productivity is higher on the sNCL group. Analyzing the time spent separately, we can see that the participants in the sNCL group had a slightly lower mean than the participants in the NCL group as shown in Figure 3. However, as the productivity is measured by the ration of length by time, the mean's verbosity of NCL code affected directly the productivity.

Although the mean's time spent was lower, the variance calculated with the Levene test shows that there is no significant variance between the samples (p-value = 0.2244). To test H1 we used the Mann-Whitney test, because for this attribute, from the statistical calculations we can observe that the sample is not normalized as we can see in Table 3. According to the result, the p-value=0.0097 can not reject the null-hypotheses H1, indicating that sNCL does not present higher productivity than the NCL.

#### *5.1.2 H2: Verbosity*

To measure the verbosity of the codes we adopted two metrics. The first is calculated by the number of lines of the code, and the second is the amount of characters. The results is presented on Table 3 as Verbosity (lines) and Verbosity (length) respectively. Both the mean of the lines and the length are significantly larger in the code's participant from NCL group. As we can see on Table 2, Levene test shows that variance the significance equal 0.004 so that we can reject the null hypothesis, showing that there is a significant variance between the lines from both samples. The variance for the length had a similar result, with a p-value=0.019. The results on Table 2 shows that there is a statistical difference between the verbosity of NCL and sNCL. Also, regardless the correctness the verbosity of NCL code is indeed higher.

#### *5.1.3 H3: Correctness*

The score of each participant was extracted from a previously determined task performed during the experiment as mentioned throughout the Section 4. If the participant had answered partially or entirely correct, part or all the points were computed, otherwise no point were attributed. The points were computed per corrected element, and we adopted integer values in order to facilitate point calculations. Although there is a difference between the means of the two groups as shown in Figure 3. The variance of the calculated samples is significantly equal with p-value=0.6753, and the mean from NCL is higher than sNCL, as well the measure of Minimum and maximum (Figure 3) this might be related to the fact that, the NCL group used the NCL plugin. Moreover, during the experiment was noticed that the participants from sNCL group had more compilation errors, as shown in the Figure 9, it is because the way that the sNCL code is compiled to NCL. Through the t-test we found the p-value=0.45135 so the $p > 0.05$ the difference between means is not statistically significant.

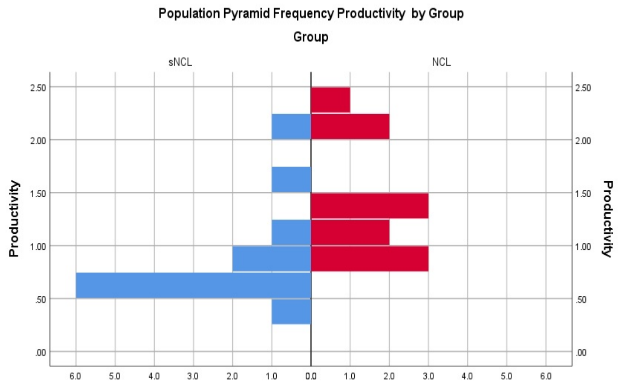
## 6. LESSONS LEARNED

This section shows lessons learned from the experiment. Moreover, we discuss about the productivity, verbosity, correctness of the sNCL compared to NCL and the user's point of view.

### 6.1 Productivity, Correctness and verbosity

Regardless of the correctness of the participants, through the data collected it was possible to verify that the productivity is greater with the NCL language. Nonetheless, it is due to the fact that the NCL is more verbose than sNCL as we can verify in the chart of figure 6, 7 and 8.

**Figure 6: Productivity by Group**



Furthermore, the tool (NCL Eclipse) used during the experiment in the NCL training session helped participants in coding. Because the plugin helps to highlight syntax errors and auto-completion.

The verbosity measures used were both lines of code, which is represented by the box-plot on figure 8 and the length, which is the number of characters presents on the code, that is represented by the box-plot on Figure 7.

**Figure 7: Box-plot of Length**

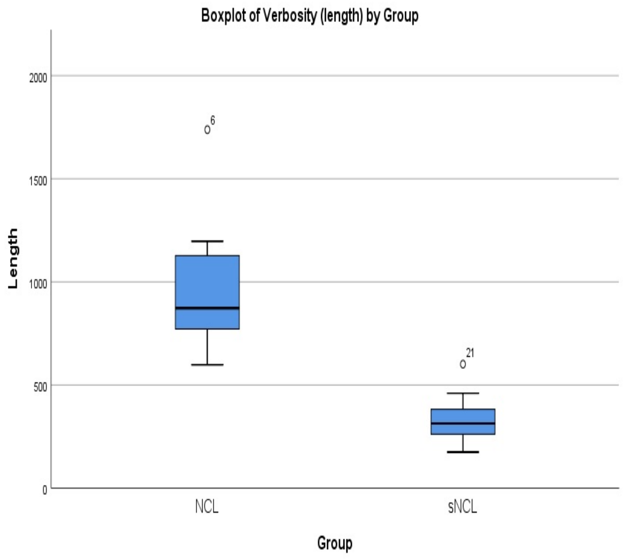
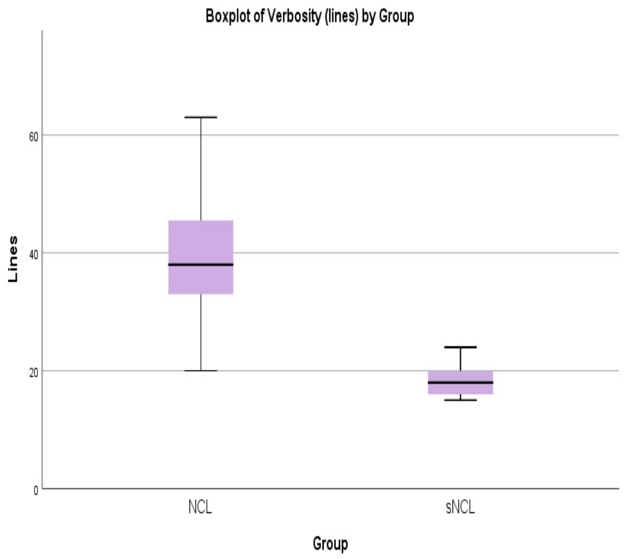


**Figure 8: Box-plot of Lines**



As can be noticed in box-plots (Figure 7 and 8), both the measurements of lines and the length of the code are much larger in the samples from the NCL group than in the sNCL group.

Another aspect it is worth to highlight, is that during the experiment the participants had some compile errors. How-

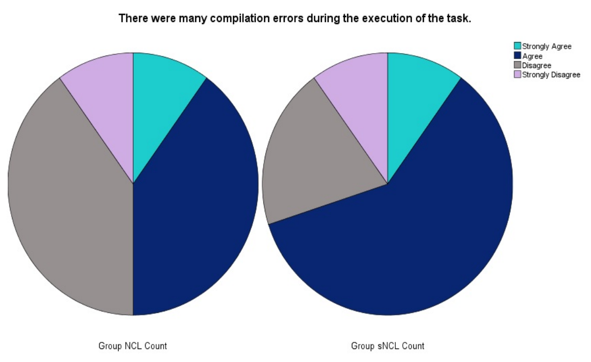
**Table 3: Descriptive Statistics of the experiment Results**

| | Group | Mean | Minimum | Maximum | Standard Deviation | Shapiro-Wilk p-value |
|---|---|---|---|---|---|---|
| Time (minutes) | NCL | 29.36 | 16 | 37 | 6.741 | 0.098 |
| | sNCL | 28.25 | 10 | 50 | 11.054 | 0.924 |
| Verbosity (length) | NCL | 971.36 | 598 | 1738 | 314.906 | 0.054 |
| | sNCL | 335.58 | 175 | 601 | 115.404 | 0.534 |
| Verbosity (lines) | NCL | 39.82 | 20 | 63 | 11.762 | 0.993 |
| | sNCL | 18.41 | 15 | 24 | 2.644 | 0.342 |
| Correctness (points) | NCL | 6.18 | 4 | 9 | 1.662 | 0.249 |
| | sNCL | 5.66 | 3 | 8 | 1.556 | 0.051 |
| Productivity (length/time) | NCL | 1.42 | 0.84 | 2.37 | 0.536 | 0.087 |
| | sNCL | 0.86 | 0.34 | 2.13 | 0.522 | 0.008 |

ever, participants from sNCL group had more difficult to deal with these errors and we believe that is due the tool used to compile the code. Because unlike the tool used for the NCL group, the sNCL compilation is done through command line so that does not provide further information about the syntax errors. On the survey applied after the experiment we asked to participants if they faced errors during the resolution of the task. The method used for the answers was Likert scale, and the results can be seen on figure 9. The bar chart, shows that, the majority participants from sNCL group answered that had compilation errors during the execution of the assignment.

During the experiment we could notice this kind of problem, also the sNCL group did not use a IDE as Eclipse, they use Kate editor.

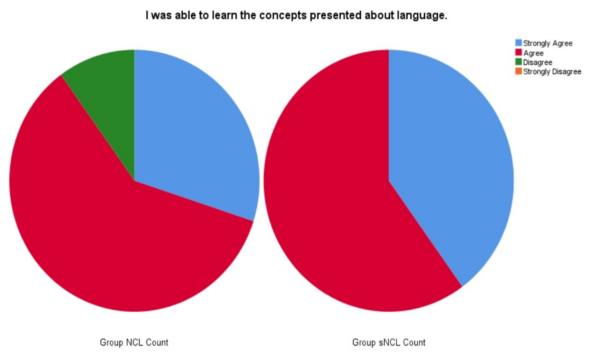
**Figure 9: Pie chart of errors during the experiment**



## 6.2 User's Point of View

In order to analyze the opinion of the participants about the experiment and about the languages of the study. We asked to the participants questions about the characteristics of both languages as well questions related to the understanding from the training session. These questions are qualitative using the Likert scale. The questions were about the ability to understand the concepts of the languages through the training section. The questionnaires were equal for both group. The objective of it was to analyze the participant's opinion, also to have a complementary qualitative data about the object of this study. From 29 participants, 20 answered the questions, 10 from each group. The results are described in Table 4, the questions follow the scale aforementioned and are numbered from 1 to 7.

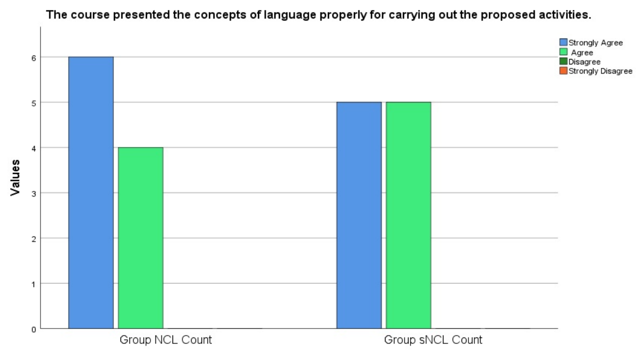
**Figure 10: Pie chart about the training session**



The first question was related to the training session, if the concepts taught were learned. The goal was to identify if even with the limitation of the time the concepts were assimilated by the participants. As we can see on Figure 10, the majority of both group agreed that could learn the concepts, although the answers from sNCL group were more positive. These results presents that according to their opinion they could apply the knowledge on the task proposed. The second question asked if the training sessions were appropriate according to the given task. We can notice on Figure 11 that all the participants had a positive opinion about the lecture, even though they did know about both language prior the experiment. This question was useful to measure the perception of the participants regarding the content taught, as well the quality of the training.

The third question asked if the participants understood all the elements presented, the percentage of the group is described on Table 4. It important to highlight that the referred elements are the one presented in the sessions, not

**Figure 11: Bar chart**



all the elements available on both languages. The results shows that both groups felt they understood.

On the fourth question, they were asked about the learnability of the language. Although the result of the previous question has shown that most students understood the elements in question 4, the result in Table 4 shows 50% of those who learned agreed that NCL is easy while 30% disagree with the statement. In the sNCL group everyone agreed with the statement.

The fifth question is related to the ease of implementing the language code. All participants agreed with the statement, however 70% of those in the sNCL group strongly agreed with the statement. Related to the previous question, the issue 6 states that languages have simple syntax. As well as the previous question, the result was higher for participants in the sNCL group. The final question was related to the errors that occurred during the session. In this topic most participants agreed that there were many compilation errors. This may be related to the fact that during the experiment, the participants learned the languages for the first time, as well as some tools used.

## 7. VALIDITY EVALUATION

This section show the threats to validity of this experiment and how they lessen during this study.

**Conclusion Validity.** We noted three threats in this conception: (i) *design of the task*: the designed task could have been too challenging or biased to a specific language. In order to reduce this threat, we extracted a set of basic equivalent concepts from both languages, and designed an assignment that could be performed by the participants. The difficulty of the task was analyzed in the pilot study where the participants were able to perform the assignment in both languages; (ii) *time restrictions* : the period of time reserved to the experiment could have impacted the participants' answers. The pilot study was used to estimate the appropriate average time to perform the task in both tools, so that we could avoid this risk; and (iii) *heterogeneity of the participants*: it refers to the choice of the participants involved, in the experiment. To decrease this threat, we selected novice and experienced participants in programming languages and tools from the undergraduate courses of IFBA. Although this heterogeneity can still be seen as a conclusion threat, it contributes, on the other hand, to minimize the external threats of the study.

**Table 4: Results of the Questionnaire**

| # Question | Scale | NCL(%) | sNCL(%) |
|---|---|---|---|
| 1 | Agree | 60 | 60 |
| | Strongly Agree | 30 | 40 |
| | Disagree | 10 | 0 |
| | Strongly Disagree | 0 | 0 |
| 2 | Agree | 40 | 50 |
| | Strongly Agree | 60 | 50 |
| | Disagree | 0 | 0 |
| | Strongly Disagree | 0 | 0 |
| 3 | Agree | 70 | 80 |
| | Strongly Agree | 20 | 20 |
| | Disagree | 0 | 0 |
| | Strongly Disagree | 10 | 0 |
| 4 | Agree | 50 | 40 |
| | Strongly Agree | 20 | 60 |
| | Disagree | 30 | 0 |
| | Strongly Disagree | 0 | 0 |
| 5 | Agree | 80 | 30 |
| | Strongly Agree | 20 | 70 |
| | Disagree | 0 | 0 |
| | Strongly Disagree | 0 | 0 |
| 6 | Agree | 70 | 30 |
| | Strongly Agree | 20 | 70 |
| | Disagree | 10 | 0 |
| | Strongly Disagree | 0 | 0 |
| 7 | Agree | 40 | 60 |
| | Strongly Agree | 40 | 10 |
| | Disagree | 10 | 20 |
| | Strongly Disagree | 10 | 10 |

**Construct Validity.** In this category, we observed two threats: (i) *operational procedures of the experiment*: the participants could not understand the experiment guidelines properly. In order to reduce this threat, we performed a training session explaining the basic concepts from both languages and the experimental execution process. During the task, an example assignment was performed in the training session with the goal to present how to use the tools for each language. We also elaborated a tutorial material for the participants so that they could have more information about both language; (ii) *confounding constructs and levels of constructs*: it relates mostly to the expertise level of the participants. To lessen this threat, we chose students with distinct levels of knowledge.

**Internal Validity.** Concerning the internal threat, we noted one threat, which refers to the division of participants to perform the task from the treatments. It is related to the partition of the participants in the groups (NCL and sNCL). In order to avoid this threat, each participant filled out a form about their knowledge in programming languages and tools related to the experiment. The groups were divided based on the participant's answers so that taking into consideration their answers in order to make the partition as impartial as possible.

**External Validity.** The threat in this category concerns the generalization of the experiment result to other participants compared to the ones on the experiment. To reduce this threat, any participant who had a previous knowledge in programming language regardless their level could join the experiment.

We can note that exist threats to the construct, internal and external validity. Nevertheless, these were mitigate in order to have valid results.

## 8. CONCLUSIONS

This paper has presented a sNCL language and its features comparing to NCL on application development. To analyze these characteristics, it was conducted a controlled experimental evaluation comparing sNCL to NCL in order to understand the learnability, productivity and verbosity and its impacts. The data obtained from the experiment revealed positive results related to the verbosity of sNCL. sNCL code provides a simpler and less verbose code. Also we could learn that learnability from sNCL was not significantly higher, as lesson learned we find that this may be related to aspects such as: not exist a development plugin for IDE yet for sNCL, the way to compile the code and error output is not as clear as the NCL. The results regarding the productivity present a higher mean related to the NCL, however it is due the fact that NCL is more verbose, so it produce more code per time. In addition to the quantitative data, through the questionnaire applied, we could identify the opinion of the participants with respect to languages. According to the questions, most students in the sNCL group found the language to have simple syntax and it is easy to learn.

In this study, we selected undergraduate students with a with varied background and applied the experiment in-vitro in a controlled environment, and chose a simple task using the basic concepts of the languages. The data obtained from the experiment revealed statistically significant results related to verbosity of sNCL. It is worthwhile to run new sessions of the experiment to perform the statistical calculations based on a larger sample to have a more accurate results. All the written material about the procedures needed is provided on [7].

After the conclusion of the experiment and the collected data, we may suggest some aspects of improvements such as: (i) Creation of a plugin for use in Eclipse or another widely known IDE, in which it is possible to highlight syntax and code suggestions; (ii) Improvement in the code compilation process, so that a detecting of an error is faster; (iii) The continuous evolution of the language so that it can be widely used by developers.

## 9. REFERENCES

[1] S. de Indicadores. Pesquisa nacional por amostra de domicílios. 2009.

[2] L. de Macedo Terças, D. d. S. Moraes, and C. d. S. S. Neto. Specifying a domain specific language for simplifying the authoring of digital tv applications.

[3] A. D. M. GINGA. Suporte para desenvolvimento de apli-cações multiusuário e multidispositivo para tv digital com ginga. 2007.

[4] F. Hermans, M. Pinzger, and A. Van Deursen. Domain-specific languages in practice: A user study on the success factors. In *International Conference on Model Driven Engineering Languages and Systems*, pages 423–437. Springer, 2009.

[5] A. MÉDOLA. Televisão digital brasileira e os novos processos de produção de conteúdos-os desafios para o comunicador. *Revista da Associação Nacional dos Programas de Pós-Graduação em Comunicação, E-Compós, Brasília-DF*, 12(3):1–12, 2009.

[6] P. Pessoa. Desenvolvimento de jogos para a plataforma ginga utilizando nclua.

[7] J. Santos. Experimental evaluation, https://github.com/jamilessnts/experimental-evaluation.git.

[8] L. F. G. Soares, M. F. Moreno, C. D. S. S. Neto, and M. F. Moreno. Ginga-ncl: declarative middleware for multimedia iptv services. *IEEE Communications Magazine*, 48(6), 2010.

[9] L. F. G. Soares, R. F. Rodrigues, and M. F. Moreno. Ginga-ncl: the declarative environment of the brazilian digital tv system. *Journal of the Brazilian Computer Society*, 12(4):37–46, 2007.

[10] L. F. G. S. Soares. *Programando em NCL 3.0: desenvolvimento de aplicaçoes para middleware Ginga: TV digital e Web*. Elsevier, 2009.

[11] C. Soares Neto, C. Souza, and L. Soares. Linguagens computacionais como interfaces: um estudo com nested context language. *Simpósio Brasileiro de fatores humanos em sistemas computacionais, Porto Alegre, RS*, 2008.

[12] G. L. d. Souza Filho, L. E. C. Leite, and C. E. C. F. Batista. Ginga-j: The procedural middleware for the brazilian digital tv system. *Journal of the Brazilian Computer Society*, 12(4):47–56, 2007.

[13] L. d. M. Terças, D. d. S. Moraes, D. d. S. Ribeiro, M. C. M. Neto, and C. d. S. S. Neto. Usability-based language for authoring ncl documents. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, pages 101–108. ACM, 2017.

[14] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

## 10. APPENDIX

### 10.1 Characterization Form

**SUBJECT CHARACTERIZATION**

**A) PERSONAL DATA**

1. Name: ______________________
   Gender: ( )Male ( )Female
2. Course: ____________ Semester: ____
3. Email: ______________________

**B)TECHNICAL INFORMATION**

1. Which language do you program?
   ( ) C ( ) Java ( ) Python ( )C ++ Other: ____
2. How do you rate your knowledge on the following topics:
3. IDE Eclipse
   ( ) None ( ) Low ( ) Moderate ( )Expert
4. Linux
   ( ) None ( ) Low ( ) Moderate ( )Expert

5. Lua Language
   ( ) None ( ) Low ( ) Moderate (
6. XML
   ( ) None ( ) Low ( ) Moderate ( )Expert
7. HTML
   ( ) None ( ) Low ( ) Moderate (
8. Ginga Middleware
   ( ) None ( ) Low ( ) Moderate ( )Expert
9. Do you know any DSL language?
   ( )Yes ( )No
10. Have you worked with NCL before?
   ( )Yes ( )No

## 10.2 Consent Form

Name: ______________________________

The information contained in this form intended to enter into a written agreement, whereby the subject authorizes its participation in the NCL and sNCL experiment. The subject has full knowledge of the nature of the procedures which he has to follow to be a participant. The subject is free to give up to be a participant at any time without coercion. This participation is voluntary and the subject of this experiment is free to withdraw your consent at any time and fails to participate in the study without prejudice to any service that is being or will be submitted.

1. **EXPERIMENTAL STUDY TITLE**
   An Experimental Evaluation of sNCL compared to NCL.
2. **TOPIC**
   Experimental Evaluation.
3. **STUDY PURPOSE**
   To evaluate the differences between NCL and sNCL.
4. **RESPONSIBLE INSTITUTION**
   Federal Institute of Bahia
5. **RESPONSIBLE RESEARCHERS**
   • Janile Samtos
   • Manoel Neto
   • Renata Novais
6. **INFORMED CONTENT**

I, _________________, certify that, having read the above information, and sufficiently informed of all the items, I fully agree with the experiment. So, I authorize the execution of the research above.

_________________ , ____/____/_____ .

Subject signature:

________________________________

## 10.3 Survey

Name: ________________
Programming language: ________________
Date: ____/____/_____
Location: ______________

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

1. I was able to learn the concepts presented about language.

2. The course presented the concepts of language properly for carrying out the proposed activity.

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

3. I understood the elements of language.

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

4. I understood the elements of language.

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

5. I found the language easy to learn.

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

6. The language is easy to implement.

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

7. I found the language syntax simple.

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

8. Many compilation errors occurred during the execution of the task.

| | |
|---|---|
| Strongly Agree | |
| Agree | |
| Disagree | |
| Strongly Disagree | |

9. What is your experience with logic programming and software development?

| Time (years) | Less than 1 | 1 to 2 | 2 to 3 | More than 3 |
|---|---|---|---|---|
| | | | | |

## 10.4 NCL Tutorial

This tutorial aims to introduce the concepts of NCL (Nested Context Language).

### *Part I*

NCL is a declarative programming language based on XML for developing applications for Brazilian digital television. We use NCL to build multimedia documents.

### But what is multimedia document?

It is a document that have more than one type of media, for example: image, audio, video.To build these types of documents, we need to define: what we want to play, where (where on the screen) and when.

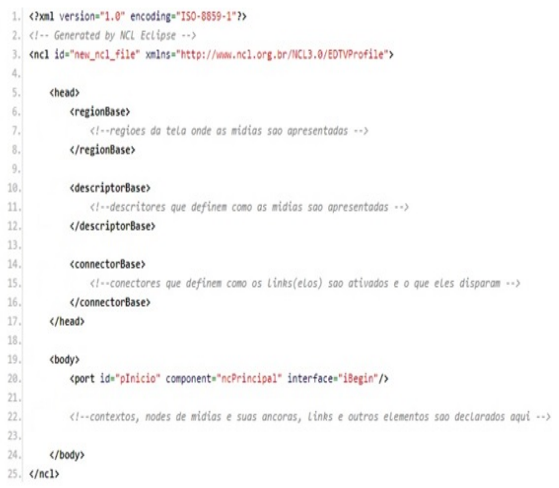
### NCL document's structure

An NCL document is a XML-based file that contains:

• Head of the file
• Head of the program: where is defined regions, descriptors, connectors and the usage rules.
• Body: where is defined the contexts, media nodes, links and other elements that defines the content and structure of the program.
• Port: indicates where starts the exhibition of the program
• Conclusion: end of the document.

On the previous figure is presented the basic structure of NCL document.

1. Head of the file
2. Head
3. body
4. port
5. contexts
6. medias, links



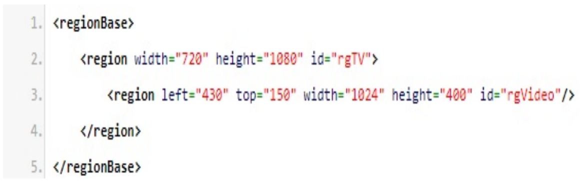
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Generated by NCL Eclipse -->
<ncl id="new_ncl_file" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

    <head>
        <regionBase>
            <!--regioes da tela onde as midias sao apresentadas -->
        </regionBase>

        <descriptorBase>
            <!--descritores que definem como as midias sao apresentadas -->
        </descriptorBase>

        <connectorBase>
            <!--conectores que definem como os links(elos) sao ativados e o que eles disparam -->
        </connectorBase>
    </head>

    <body>
        <port id="pInicio" component="ncPrincipal" interface="iBegin"/>

        <!--contextos, nodes de midias e suas ancoras, links e outros elementos sao declarados aqui -->

    </body>
</ncl>
```

**Figure 12: Struture of NCL**

1. Head of the file:
   **<?xml version="1.0" encoding="ISO-8859-1">** is the first code line where is defined the version and type of codification. In this example is ISO-8859-1. **<ncl id="new-ncl-file">** this tag defines the beginning of NCL program. The tag has few attributes such as id which defines a unique identifier that can be used as a reference by other NCL elements.
2. Head:
   The header is delimited by the <head> and </ head> tags equal to the HTML. Within the head tag we define three basic elements of an NCL program, which are described below:

### • Region Base

It is the region It is the region where we define where the media will be displayed on the screen.



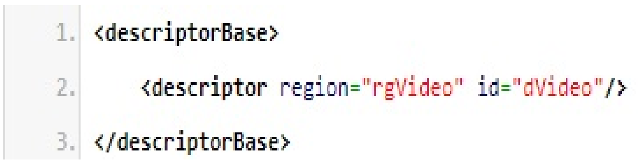
```
<regionBase>
    <region width="720" height="1080" id="rgTV">
        <region left="430" top="150" width="1024" height="400" id="rgVideo"/>
    </region>
</regionBase>
```

**Figure 13: Region Base**

On the previous example is defined two regions: rgTV with the attributes width, height and id which specify the width of the region in pixels, the height of the region and the identifier of the region, respectively. The "rgVideo" region has aside the attributes already mentioned two others: left and top, which define the position of the region on the screen, in relation to the left and top, respectively.

### • Descriptor Base

Descriptors define how and where media will be displayed. In the example below a descriptor with the attribute id equal to "dVideo" is created. Also defined is the "region" attribute that refers to a previously created region, called rgVideo. The descriptors govern the behavior of a "media node", for example defining the region where the media will be displayed. In this example, the media would be displayed in the rgVideo region.

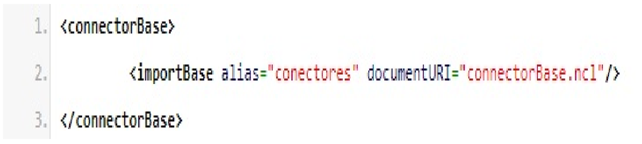


```
<descriptorBase>
    <descriptor region="rgVideo" id="dVideo"/>
</descriptorBase>
```

**Figure 14: Descriptor Base**

## • Connector Base

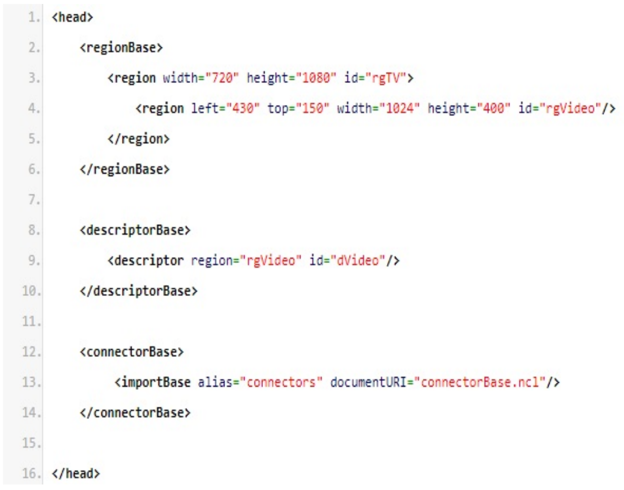
Connectors define how links are activated and what they trigger. Usually, the connectors are defined in a file outside the main file. It is possible to keep the code of the connectors in the main file, however, it is recommended to use an external file. This keeps the code more organized, allows the reuse of the connectors and makes the work easier.



```
<connectorBase>
        <importBase alias="conectores" documentURI="connectorBase.ncl"/>
</connectorBase>
```

**Figure 15: Connector Base**

In the previous one, the "documentURI" attribute of the file "connectorBase.ncl", which is in the same directory as the NCL program, contains dozens of connectors ready. The "alias" attribute is used to identify to be loaded. This "identification" will be used by the links to refer to the loaded base.



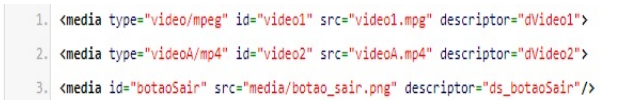
```
<head>
    <regionBase>
        <region width="720" height="1080" id="rgTV">
            <region left="430" top="150" width="1024" height="400" id="rgVideo"/>
        </region>
    </regionBase>

    <descriptorBase>
        <descriptor region="rgVideo" id="dVideo"/>
    </descriptorBase>

    <connectorBase>
         <importBase alias="connectors" documentURI="connectorBase.ncl"/>
    </connectorBase>

</head>
```

**Figure 16: Example of a complete <head>**

Body:
In the body are inserted media (nodes), ports, links, and anchors.

## • Media

They are associated with media types (txt, HTML, jpeg, mpeg etc).



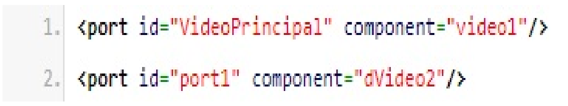
```
<media type="video/mpeg" id="video1" src="video1.mpg" descriptor="dVideo1">
<media type="videoA/mp4" id="video2" src="videoA.mp4" descriptor="dVideo2">
<media id="botaoSair" src="media/botao_sair.png" descriptor="ds_botaoSair"/>
```

**Figure 17: Media example**

In the picture, we have 3 nodes, where the media are referenced through the "src" attribute. Note that the "type" attribute of the media is optional, it defines the type of media, whether it is video, text, image, etc. The most common types are: image / gif, image / png, image / bmp, video.mp4, text / plain, audio.mp3.

## • Ports

Through the ports, we can access the contents of a context. That is, for a link to point to a node that is internal to the context, it must have a port that directs to the internal media.



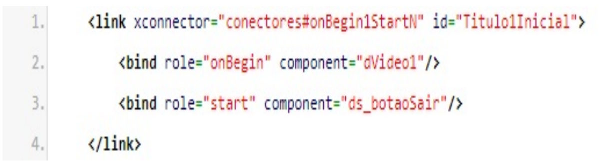
```
<port id="VideoPrincipal" component="video1"/>
<port id="port1" component="dVideo2"/>
```

**Figure 18: Port example**

The attribute "id" defines the port and the "component" defines which media will be accessed in a given context. In the example, component = "video1" will activate the id = "video1" media.

## • Links

With the links we can synchronize events in an NCL program. For instance, the links help to start running one media simultaneously with another, it also defines its term.



```
<link xconnector="conectores#onBegin1StartN" id="Titulo1Inicial">
    <bind role="onBegin" component="dVideo1"/>
    <bind role="start" component="ds_botaoSair"/>
</link>
```

**Figure 19: Link example**

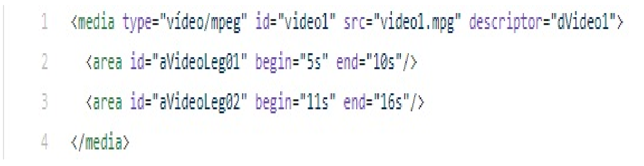
This synchronization is possible because of the connectors, as already mentioned above can be created in a separate file or inside the main NCL file.Note that in the example, the "xconnector" attribute references the base connectors that will be used. In xconnector = "connectors "onBegin1StartN", note that # separates 2 values in the attribute, the first part refers to the connector base, which was the alias we gave in the declaration of the connectors, and the second part indicates that the connector will be used. The <bind> tag determines which form and which media will activate the link. On <bind> the media is being called by “dVideo1” and “ds-botaoSair” will be displayed simultaneously.

- **Anchors**

Anchors are entry points for the media nodes or contexts. The purpose of using anchors is to use segments of a media node or context, either as the source or destination of links. In other words, the area represents an excerpt in the time or space of the media to which it belongs. There are two types of anchors: content and attribute.

- **Content**

Defines a segment of the media, a range of time, or region of the screen that can be used as the trigger point for links. The content anchor is defined by the <area> tag within the <media> tag.

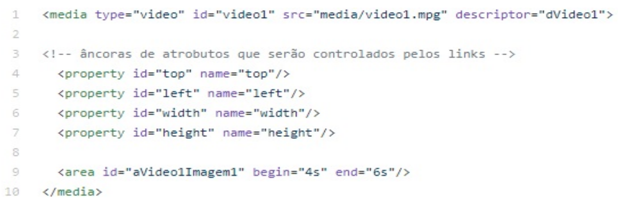


```
<media type="video/mpeg" id="video1" src="video1.mpg" descriptor="dVideo1">
  <area id="aVideoLeg01" begin="5s" end="10s"/>
  <area id="aVideoLeg02" begin="11s" end="16s"/>
</media>
```

**Figure 20: Content anchor**

- **Attributes**

Refers to the ownership of a source or destination media, which can be manipulated by the links. For example media volume (video).



```
<media type="video" id="video1" src="media/video1.mpg" descriptor="dVideo1">

<!-- âncoras de atrobutos que serão controlados pelos links -->
  <property id="top" name="top"/>
  <property id="left" name="left"/>
  <property id="width" name="width"/>
  <property id="height" name="height"/>

  <area id="aVideo1Imagem1" begin="4s" end="6s"/>
</media>
```

**Figure 21: Attribute anchor**

## 10.5 sNCL Tutorial

This tutorial aims to introduce the concepts of sNCL (Simpler Nested Context Language).

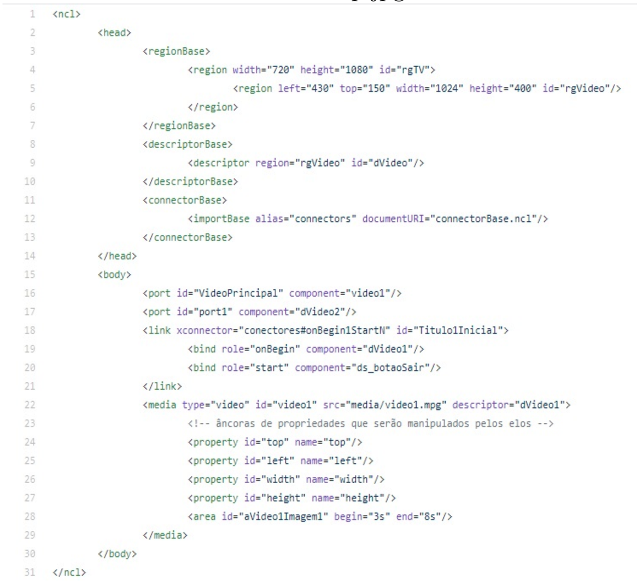
### *Part I*

sNCL it is a domain-specific declarative language (DSL) that was developed with the goal of ease the authorship of multimedia applications that compile for NCL.The syntax is similar to the imperative language Lua.

## But what is multimedia document?

It is a document that have more than one type of media, for example: image, audio, video.To build these types of documents, we need to define: what we want to play, where (where on the screen) and when.

basico comp.jpg



```
<ncl>
        <head>
                <regionBase>
                        <region width="720" height="1080" id="rgTV">
                                <region left="430" top="150" width="1024" height="400" id="rgVideo"/>
                        </region>
                </regionBase>
                <descriptorBase>
                        <descriptor region="rgVideo" id="dVideo"/>
                </descriptorBase>
                <connectorBase>
                        <importBase alias="connectors" documentURI="connectorBase.ncl"/>
                </connectorBase>
        </head>
        <body>
                <port id="VideoPrincipal" component="video1"/>
                <port id="port1" component="dVideo2"/>
                <link xconnector="conectores#onBegin1StartN" id="Titulo1Inicial">
                        <bind role="onBegin" component="dVideo1"/>
                        <bind role="start" component="ds_botaoSair"/>
                </link>
                <media type="video" id="video1" src="media/video1.mpg" descriptor="dVideo1">
                        <!-- âncoras de propriedades que serão manipulados pelos elos -->
                        <property id="top" name="top"/>
                        <property id="left" name="left"/>
                        <property id="width" name="width"/>
                        <property id="height" name="height"/>
                        <area id="aVideo1Imagem1" begin="3s" end="8s"/>
                </media>
        </body>
</ncl>
```

**Figure 22: An example of a full basic NCL code**

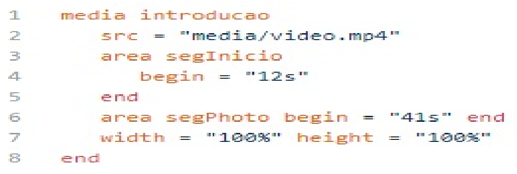
## Elements of an sNCL document

- Media
- Port
- Links
- Region
- Context

### Media

It is the word reserved to represent a media, which can be: photo, video, audio.

media.jpg



```
media introducao
    src = "media/video.mp4"
    area segInicio
       begin = "12s"
    end
    area segPhoto begin = "41s" end
    width = "100%" height = "100%"
end
```

**Figure 23: Media declaration**

The figure above illustrates the state of a media in sNCL. Notice that the reserved word media is followed by the word introduction, which is the identifier (id) of the media. This identifier must be unique and serves for the media to be referenced elsewhere in the code.The reserved word **src** is where we indicate where the media is.

The media can contain attributes that define its size and position as width, height, left, right, top, bottom. Note that there is also an area declaration in the example, this area is an anchor that is an entry point for the media nodes or contexts. The purpose of using anchors is to use segments of a media node or context, either as the source or destination of links.

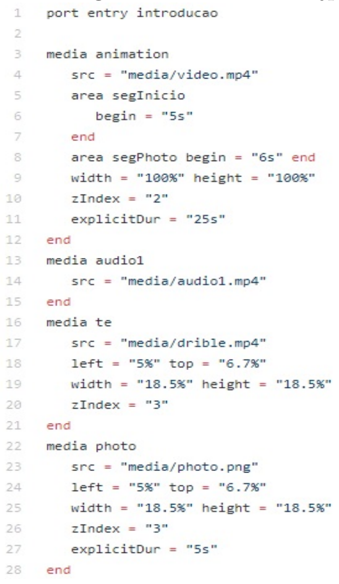
In other words, the area represents an excerpt in the time or space of the media to which it belongs.In the example shown we can see that we have two areas, one with all the information in one row and the second one is made the declaration in an indented way and in separate lines.

It is worth noting that indented code is not mandatory, but in this way, it transforms the readable code for understanding.

## Port

At launching an application, the player needs to know which media will be played at the beginning, the port has that purpose, and then the media will be executed at the beginning. The port can be declared anywhere in the code, as long as it is not the child of any element. It must be declared independently, as shown in the following example:

e codigo inicio basic sNCL.jpg



```
port entry introducao

media animation
   src = "media/video.mp4"
   area segInicio
      begin = "5s"
   end
   area segPhoto begin = "6s" end
   width = "100%" height = "100%"
   zIndex = "2"
   explicitDur = "25s"
end
media audio1
   src = "media/audio1.mp4"
end
media te
   src = "media/drible.mp4"
   left = "5%" top = "6.7%"
   width = "18.5%" height = "18.5%"
   zIndex = "3"
end
media photo
   src = "media/photo.png"
   left = "5%" top = "6.7%"
   width = "18.5%" height = "18.5%"
   zIndex = "3"
   explicitDur = "5s"
end
```

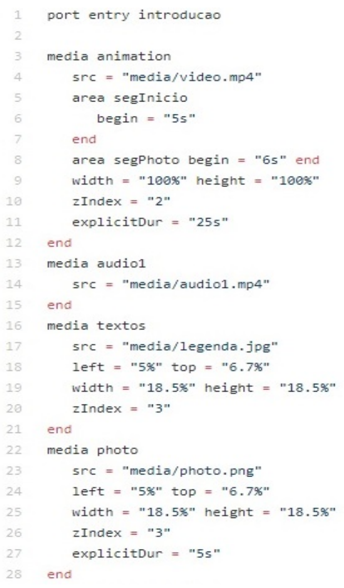
**Figure 24: sNCL basic example**

Notice that there is a new attribute in the medias animation and photo, the attribute is explicitDur, it specifies the duration of each media explicitly.

## Link

Through the links, we can synchronize events into a program. For instance, the links help to start running one media simultaneously with another, it also defines its term.

The conditions for the execution of the media is made by the links, these conditions are blocks, which begin with the word defining the action, and the media that receives the action, and ending with the word "end", within which the parameters of the action. As showed on the following picture:



```
port entry introducao

media animation
   src = "media/video.mp4"
   area segInicio
      begin = "5s"
   end
   area segPhoto begin = "6s" end
   width = "100%" height = "100%"
   zIndex = "2"
   explicitDur = "25s"
end
media audio1
   src = "media/audio1.mp4"
end
media textos
   src = "media/legenda.jpg"
   left = "5%" top = "6.7%"
   width = "18.5%" height = "18.5%"
   zIndex = "3"
end
media photo
   src = "media/photo.png"
   left = "5%" top = "6.7%"
   width = "18.5%" height = "18.5%"
   zIndex = "3"
   explicitDur = "5s"
end
```

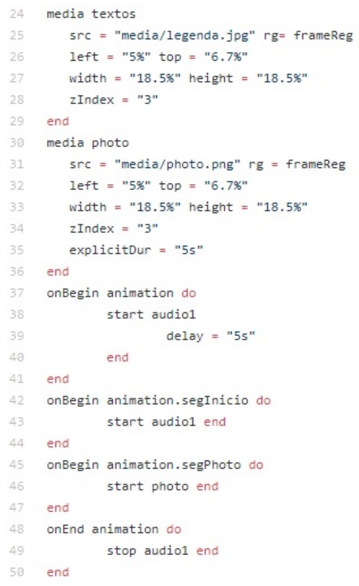
**Figure 25: sNCL Link example**

## Region

The region element is important because it defines a region on the screen, and is widely used since several media can reference the same region.

The region has attributes as well as the media element, among its attributes are: width, height, zIndex, rg.The zIndex attribute specifies how to overlap regions. A region of greater value for zIndex overlaps with that of lower value.

The rg attribute is used in the media to reference a region.



```
media textos
   src = "media/legenda.jpg" rg= frameReg
   left = "5%" top = "6.7%"
   width = "18.5%" height = "18.5%"
   zIndex = "3"
end
media photo
   src = "media/photo.png" rg = frameReg
   left = "5%" top = "6.7%"
   width = "18.5%" height = "18.5%"
   zIndex = "3"
   explicitDur = "5s"
end
onBegin animation do
        start audio1
                delay = "5s"
        end
end
onBegin animation.segInicio do
        start audio1 end
end
onBegin animation.segPhoto do
        start photo end
end
onEnd animation do
        stop audio1 end
end
```

**Figure 26: sNCL region example**

## Context

The context element can be used to structure an application. In the following example, we will illustrate the use of contexts, grouping all the elements of the animation. This will allow, in addition to the greater structure of the program, the reuse of the entire structure.