DYNA
ISSN: 0012-7353
ISSN: 2346-2183
Universidad Nacional de Colombia

Miranda, Mirtha; Casas, Sandra

Usable and interactive application generator for Digital TV

DYNA, vol. 86, no. 211, 2019, October-December, pp. 174-183
Universidad Nacional de Colombia

DOI: https://doi.org/10.15446/dyna.v86n211.75574

DYNA
ISSN: 0012-7353
ISSN: 2346-2183
Universidad Nacional de Colombia

# Usable and interactive application generator for Digital TV

**Miranda, Mirtha; Casas, Sandra**
Usable and interactive application generator for Digital TV
DYNA, vol. 86, no. 211, 2019
Universidad Nacional de Colombia
**Available in:** http://www.redalyc.org/articulo.oa?id=49663345020
**DOI:** 10.15446/dyna.v86n211.75574

# Usable and interactive application generator for Digital TV

Generador de aplicaciones interactivas y usables para TV Digital

Mirtha Miranda [a] mirfamir@gmail.com
*Universidad Nacional de la Patagonia Austral, Argentina*
Sandra Casas [a] scasas@unpa.edu.ar
*Universidad Nacional de la Patagonia Austral, Argentina*

DYNA, vol. 86, no. 211, 2019

Universidad Nacional de Colombia

Received: 15 October 2018
Revised document received: 19 September 2019
Accepted: 21 October 2019

**DOI:** 10.15446/dyna.v86n211.75574

**Abstract:** Interactive Digital TV (iDTV) enhances viewers' experience and participation by engaging them with an active role. The development of interactive software requires the employment of usability criteria to provide satisfactory experiences for users. In addition, the diverse characteristics of television programs require efficient processes for interactive software development. In Latin America, the process of implementing DTV is in progress, so the current stage of development of interactive applications is too incipient and immature to support industrial, quality-level development. This paper proposes combining software reuse strategies, specifically, software product lines (SPLs) and user-centered interaction design patterns, to improve productivity and quality. A generic feature model for the automatic generation of iDTV applications and an SPL-iDTV tool that supports the model are presented. The proposal is evaluated with two studies: an experiment that attempts to reproduce real application prototypes that were originally developed manually, through the use of patterns, and an evaluation of the quality of the SPL.

**Keywords:** interactivity, Digital TV, software product line, interaction design patterns, feature model, Ginga-NCL.

**Resumen:** La TV Digital interactiva (iDTV) mejora la experiencia y la participación de los espectadores. El desarrollo de software interactivo requiere el empleo de criterios de usabilidad para proporcionar experiencias satisfactorias a los usuarios. Además, las diversas características de los programas de televisión requieren procesos eficientes para el desarrollo de software interactivo. En América Latina, el proceso de implementación de TVD está en progreso, por lo que la etapa actual de desarrollo de aplicaciones interactivas es muy incipiente e inmadura como para afrontar el desarrollo a nivel industrial y de calidad. Este trabajo propone combinar estrategias de reutilización de software, específicamente, líneas de productos de software (SPL) y patrones de diseño de interacción centrados en el usuario, para mejorar la productividad y la calidad. Se presenta un modelo de características genérico para la generación automática de aplicaciones de iDTV y la herramienta SPL-iDTV que da soporte al modelo. La propuesta se evalúa con dos estudios: un experimento reproduce prototipos de aplicaciones reales que originalmente se desarrollaron manualmente, usando los patrones, y la evaluación de la calidad de la SPL.

**Palabras clave:** interactividad, TV Digital, líneas de producto de software, patrones de diseño de interacción, modelo de características, Ginga-NCL.

## 1. Introduction

Digital TV (DTV) is a set of technologies for the generation, transmission, and reception of optimal-quality images and sounds using digital signals. Digital signals allow data, video, audio files, software, and more to be sent over transmission channels. Television stations can also send several services over the same channel. Interactive DTV (iDTV) applications, which are based on the capabilities of DTV, are multimedia software that enables viewers to interact with linked content via remote control [1]. In addition to providing information, iDTV applications offer different mechanisms for viewer participation, for example, polls, debates, voting, and games. For this reason, interactive applications are very powerful, valuable content communication tools that can benefit society in general. Incorporation of the "interactivity" factor can lead actors in the television industry to experiment, create, and search for appropriate interactive applications [2].

Usability [3] is a key factor that must characterize interactive applications in order to provide satisfactory user experiences [4]. The design and subsequent implementation of the software must consider adequate criteria for the screen layout, buttons, menus, colors, and so forth, that will guarantee reasonable use of the interactivity and audiovisual content. However, if usability is implemented based on an evaluation of the product after it has been created, the required changes revealed by these software tests can delay the product delivery time, create higher costs, and require greater effort. Therefore, usability engineering [5] proposes that this evaluation be incorporated into the development process, which is more appropriate but has the drawback that developers must dominate this area or hire usability process engineers, which will increase the product cost.

The television industry is very diverse in terms of content (news, entertainment, magazines, debates, series, documentaries, etc.), frequency of broadcasts (daily, weekly), mode of production (live, recorded), audience (children, youth, etc.) and more. In addition, all combinations are possible; for example, a newscast can have several daily broadcasts and a documentary can have live performances but be based on recorded content. To address this heterogeneity, which is typical of television, highly efficient and productive interactive software development processes are necessary.

In Latin America, the transition from analogue to digital transmission (the "analog blackout") is in progress and will be completed by 2024. Seventy percent of Latin American countries have adopted the ISDB-Tb standard (Integrated Services Digital Broadcasting, Terrestrial, Brazilian version) that uses Ginga middleware [6] for the execution of iDTV applications, which must therefore be encoded in Nested Context Language (NCL; [6]) or Java. NCL code is basically an XML document that defines how media objects (videos, images, sounds, and text) are structured and related in time and space.

As the tasks to complete the technological infrastructure necessary for the implementation of DTV are being carried out, the analog and digital systems coexist. During this time, the following two problems arise from the development of iDTV applications in Latin America:

- the interactive application development processes are still too immature to support the construction of iDTV applications on an industrial scale, because the development is essentially manual and artisanal, reinforced by the distance between the software and television industries;
- effective methods to incorporate usability criteria in developed iDTV applications in a generalized manner are lacking.

Overcoming these shortcomings is an important challenge because, as in any productive process, it is necessary to reduce costs, effort, and development time as well as improve the quality of the final product. In short, this study seeks to address these challenges in order to facilitate the adoption of interactivity in Latin America and benefit the citizens.

The present study provides a solution to these challenges posed by the iDTV domain that combines two approaches to reusing software: It integrates the concepts of Software Product Lines (SPLs; [7]) and interaction design patterns for iDTV [8]. We present an SPL that can generate executable iDTV applications in Ginga middleware by automatically producing NCL code. The SPL-iDTV tool was built from a generic feature model (FM; [9]), based on patterns and design languages for iDTV [8]. The design patterns provide the tool with variability and reusability. We also present three case studies and an evaluation of the SPL quality.

The rest of this paper is organized as follows. Section 2 presents materials and methods used in this work. Section 3 discusses related work. Section 4 briefly describes the approaches and strategies used in this work. Section 5 presents an FM based on the design patterns. Section 6 discusses the SPL-iDTV tool that implements the model. Section 7 presents the evaluations. Section 8 presents the study's conclusions and future research directions.

## 2. Materials and methods

In this work, the approach called Design Science Research [10] was applied. It is generally used in research related to engineering, computer science and information systems. This approach proposes the production of artifacts, such as instances, constructs, models or methods. The steps that have been followed in this study are: (1) Identification of the problem for which a solution is sought, in which the background was elaborated, from the documentary research (presented in sections 3 and 4); (2) design and construction of the artifact that represents a solution to the problem. The Feature Model [8] approach was used for the construction of a generic model. The Java and FeatureIDE tools [11] with the AHEAD component [12] were used for the development of SPL-iDTV software; and (3) Artifact evaluation in which three complete case studies were developed to analyze the functionality of the SPL-iDTV software. It was completed with the evaluation of the model quality, using a specific method based on the SQuaRE standard.

## 3. Related works

A variety of tools have been proposed for the development of iDTV applications compatible with Ginga.

NCL Eclipse is a tool developed as a plugin for Eclipse. This plugin is intended for programmers; it is used to write the source code and correct syntax errors and has other very useful features for the NCL-Lua application developer. Basically, it is a code editor and compiler.

NCL Composer [14] is an author tool that supports several integrated views-structural, temporal, design, and textual-to create an NCL document. The views work synchronously, to provide integrated development. Minimal knowledge of the NCL language is required for using NCL Composer.

Ginga Game [15] is a framework for the development of DTV games for Ginga. It offers different packages for reuse, interfaces (GingaGame), components (GameComponent), and specific classes of the platform (GingaGameJavaTV).

FrameIDTV [16] presents a framework created on the multimedia home platform (MHP) that offers components for the implementation of various kinds of services and allows generation of the following types of applications: electronic voting, e-mail, electronic program guides (EPGs), application portals, t-banking, and t-commerce.

NCL-Inspector [17] is an NCL code quality review tool that allows experienced programmers to define new rules to be validated. The specification of the rules is done in a Java or XSLT document. It also enables modifications for improving programming practices.

ATHUS [18] is a generic framework that aims to facilitate the development of games for Ginga. The authors present two versions, one based on Java for use in Ginga-J and the other based on Lua for use in Ginga-NCL.

ITV-Learning [19] is a tool for educational instructors that allow the creation of learning objects, which facilitates development of interactive digital materials by abstracting them from programming knowledge for the creation of iDTV applications.

Crea Digital TV [20] is a tool for creating NCL-Lua applications, aimed at content producers. It implements a graphic timeline model to represent the life of the application elements, the interactivity with the viewer, and the events that occur throughout the application. Users are not required to have knowledge of NCL.

SGAi [21] is a tool that automatically generates surveys for NCL-Lua applications, handling the return channel and preserving the result through a web application.

iTV Suite Tool [22] proposes a required hardware and software architecture for producing interactive applications. It also proposes a software development process methodology and applying design patterns to guarantee usability. The software used to program the interactive application is iTV Suite. An important feature of the iTV Suite tool is that the programming structure is by scenes.

NCL-Textual Data Mixer [23] is a tool that facilitates graphic design for creating media (images, scripts, and NCL code), by means of metadata types declared in the NCL documents that use web services, simple object access protocol (SOAP), repositories, and other technologies to access information for the creation of applications.

API TVD [24] is a wizart tool that consists of graphic templates. It handles the management of temporal synchronism between media and their lifespan. The purpose of this generator is to simplify the development process by allowing the user to engage with "what" the application should do rather than "how" it is done. To this end, the user does not require knowledge of the NCL language.

IT NEWs [25] is a tool that reuses previously created news templates to generate new applications. These templates consist of NCL code. By focusing on the elements of communicability and usability, this tool allows users to create journalistic applications without having to learn how to program.

Template Generator [26] is a tool that allows users to automatically create code for an interactive application by modifying the relevant fields of a pre-established template. The templates are based on usability parameters such as level, service, and type of interactivity.

Dr. Nau [27] is an NCL code (web) generator that uses some [8] patterns as its main abstraction. It is an assistant aimed at end users.

## 4. Strategies and approaches

### *4.1. Software product lines and feature models*

SPL [7, 28] is an approach to developing families of systems based on using reusable assets to improve software quality and reduce production costs and time to market. The products developed with an SPL are specified in terms of various features. A feature is defined as an increase in the functionality of the product, and an SPL can offer both common and variable features. An FM [9, 29-30] can be used to provide a compact representation of all the products of a product line in terms of features and relationships between them. FMs can specify which elements of the product family are similar or variable throughout the development life cycle. In addition, FMs can incorporate a set of rules in the form of logical expressions formed by features, logical connectives, and quantifiers. The reason for using predicate logic to express default rules is to avoid the ambiguities of natural language.

Several tools [31] support the development of feature-oriented software. Among these, we selected FeatureIDE [11] as the framework for the present study. Fig. 1 presents an FM represented in the FeatureIDE tool. The FM nodes represent features, and the lines indicate the relationships between them. The root node, A, represents the domain concept being modeled. The features of the model are classified as mandatory, optional, or alternative. Optional features, such as node C, are represented with an empty circle and may or may not be part of a product.

Mandatory features, such as nodes B, D, and E, are represented by a filled-in circle and will be part of all products that the SPL can generate. Alternative features can be exclusive (XOR) or nonexclusive (OR). XOR ("Alternative") indicates that only one subfeature can be selected, e.g., F or G; while OR ("Or") allows more than one option to be selected for a product, e.g., H or I or both H and I. In addition, the diagram includes abstract and concrete features. Feature A, which is abstract, represents an interface; and the rest, which are concrete, are the features that implement the functionalities. FMs can also be specified through textual notations [32].

### *4.2. Interactivity design patterns for DTV applications*

An interaction design pattern is a design pattern for the field of HCI [33]. It documents proven solutions to interaction design problems in a systematic and understandable way, which is why interaction design patterns are said to be centered on the user. Different formats for interaction design patterns have been proposed [34,35]; these vary in their elements, the number of elements, and the names and order of the elements.

For the domain of iDTV applications, Kunert [8] presents the most complete, richest, and widely accepted collection. This catalog of interactivity design patterns, centered on the DTV user, describes 41 patterns, classified into 10 groups. The catalog provides a template for each pattern that indicates its name, examples, the context of use, the problems it solves, the solutions it provides, citations, and related patterns. The groups and patterns are as follows:

*Page Layout Group:* Choosing the right page layout, Overlay, Full-screen with video, Full-screen without video

*Navigation Group:* Multiple ways to navigate, Menu, Video multi-screen, Index, Page numbers, Tabs.

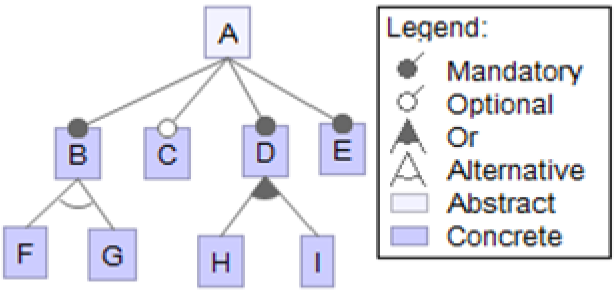


**Figure 1**
Feature Model Graphical Representation
Source. The Authors.

*Remote Control Keys Group:* Choosing the right keys, Arrow keys, OK-key, Colour keys, Number keys, Special keys.

*Basic Functions Group:* Initial call to action, Starting, Loading indication, Exiting, Hiding application, Going one level up.

*Content Presentation Group:* Text design, Content box, Paging, Scrolling, Switching between content items, Synchronized content.

*User Participation Group:* Multiple ways of user participation, Voting and multiple-choice question, Allocation of items, Text completion, Approval for connectivity.

*Text Input Group:* Multiple ways to input text, On-screen qwerty or alphabetical keyboard, Mobile phone keyboard.

*Help Group:* On-screen instruction, Help section.

*Accessibility & Personalization Group:* Accessibility, Personalization)

*Specific User Groups:* Children.

All the groups refer to specific types of content and are classified into three classes: tasks of generic users of iDTV, general content requirements, and general usability requirements. We concluded that the concepts in the patterns meet the usability levels required for iDTV, that they can be used effectively, efficiently, safely, and satisfactorily, and that they can be recast as features to provide variability.

## 5. Feature models for iDTV applications

Feature modeling consists of analyzing the domain of the product line, defining its scope, identifying the common elements and variable components of the product line, identifying the restrictions of the model, and designing the reusable devices that support the product line. The result is expressed in a feature diagram, graphic or textual, and a set of rules. We carried out the modeling in two stages: initial modeling and complete modeling.

### *5.1. Initial modeling*

This stage defined the scope of the product line as iDTV applications with local interactivity, which involves representing patterns of presentation and selecting additional information. The relevant patterns in the catalog are in the Page Layout, Navigation, Basic Functions, Content, Remote Control Keys, and Help groups, a subset that includes 27 of the 41 patterns in the catalog.

To obtain a first representation, we manually and carefully prepared a feature diagram for 32 iDTV applications using their corresponding patterns. These iDTV applications were taken from [8], and our strategy for defining the hierarchies of the diagrams was to place the iDTV application as the root of the diagram, the groups of patterns at the second level, and the patterns used from each group at the third level. Then we recorded the patterns used by each application in a matrix that grouped the applications into 19 families, in order to identify their similarities and differences.

An initial set of relationships was obtained from this analysis: (1) The Page Layout group is present in all applications. (2) Only one pattern of the Page Layout group (Overlay, Full-screen with video or Full-screen without video) is used in each application. (3) One or more patterns in the Navigation group (Menu, Video Multi-screen, Index, Page Numbers, and/or Tabs) are used in some applications. (4) One or more patterns of the Remote Control group (Arrow, OK, Color, Number, and/or Special) can be selected. (5) One or more patterns of the Basic Functions group (Initial call to action, Starting, Loading indication, Exiting, Hiding application, and/or Going one level up) can be selected. (6) One or more patterns of the Content Presentation group (Text design, Content Box, Paging, Scrolling, Switching between content items, and/or Synchronized content) can be selected. (7) Only one pattern from the Help group (On-screen instruction or help section) is used in an application.

The obtained results allowed us to design the feature diagram shown in Fig. 2, in which the groups of patterns are located at the second level as abstract features and the patterns are represented at the third level as concrete features. The diagram also indicates the classifications corresponding to mandatory, optional, and alternative features.

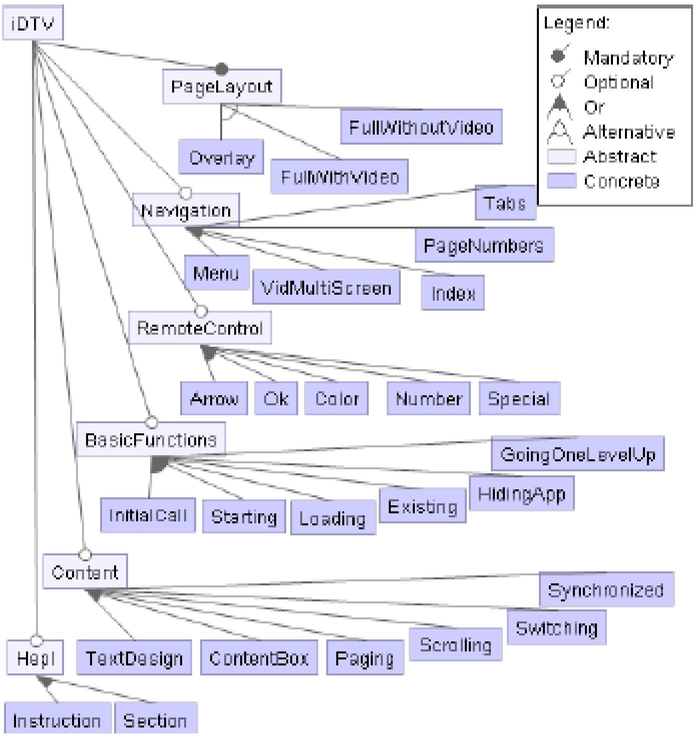


**Figure 2**
iDTV Feature Diagram
Source. The Authors.

Next, we added a set of rules to the model. These specify restrictions that must be met by the configurations that are derived from the model for the products (applications). In addition, these rules allow debugging valid products and in turn contribute to the traceability of the created model. The rules are obtained from the same catalog, which defines the relationships between groups of patterns and patterns, and their importance lies in the fact that they allow compliance with the imposed usability guidelines. Because the patterns define complementary and/ or exclusionary relationships with other patterns, these guidelines are indicated by rules in the model. Given that they arise at the level of pattern groups, we considered the following initial rules:

*Navigation => PageLayout*
*Remote Control => PageLayout # Navigation*
*BasicFunction=>PageLayout#Navigation # RemoteControl*
*Content => PageLayout # Navigation # RemoteControl*
*Help => PageLayout # Navigation # RemoteControl*

These rules indicate, for example, that to use the patterns of the Remote Control group, patterns of the Page Layout and Navigation group are required, and that to use the patterns of the Basic Functions group, at least one pattern from each of the following groups is required: Page Layout, Navigation, and Remote Control.

Using the presented model, configurations can be derived and valid products can be generated. The configuration of a product is its specification or general instantiation. A configuration of an FM is the result of selecting certain model elements and eliminating others. Through the model presented, a number of configurations or instances of new products can be obtained, such as the following examples:

*Product1 = {Overlay, Menu, Color, TextDesign}*
*Product2={FullWithVideo,Menu,Color,InitialCall,TextDesign}*
*Product3 = {FullWithoutVideo, Menu, Color, Starting, Loading, Exiting, TextDesign}*

### *5.2 Final Modeling*

The complete FM for iDTV applications, graphically represented, and the corresponding set of rules can be found in this link[1].The diagram incorporates a fourth level of features, which represent the attributes or elements of each pattern. The elements of a pattern can be mandatory, optional, or alternative. Similarly, these new features impose their own restrictions, so that rules to handle them are also incorporated.

Fig. 3 is a cutout of the complete diagram, presenting the graphical representation of the Page Layout group of patterns, which is mandatory and whose specification requires a total of 33 features.

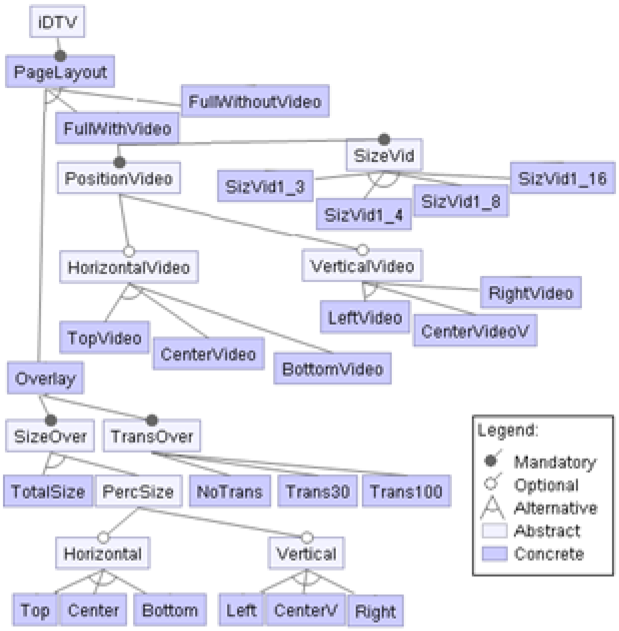


**Figure 3**

Feature Diagram for the Page Layout Pattern Group

Source. The Authors.

The three patterns of the group (Overlay, Full-screen with video and Full-screen without video) are optional and concrete features. All the elements of these patterns are represented at the fourth level. For example, Overlay consists of two obligatory features, SizeOver (the size of the area that the video occupies on the screen) and TransOver (transparency over the video) that is, whenever Overlay is chosen, the SizeOver and TransOver elements must be specified. SizeOver, in turn, has two options, TotalSize (the video occupies 100% of the screen) or PercSize (the percentage of the screen the video occupies). The latter also offers the options Horizontal or Vertical, which in turn each present three options for the location of the video: Top, Center, or Bottom in the case of Horizontal, and Left, CenterV, or Right in the case of Vertical. TransOver only offers three options, without transparency (NoTrans), Trans30 (30% transparency), or Trans100 (100% transparency).

Thus, the groups of features that are located at the fourth level of the diagram comprise the constituent elements of the patterns, indicating the precise details of the interface. Some patterns do not require more features for their configuration, as is the case with Full-screen without video.

As previously mentioned, the model also includes a set of rules that, when applied to the FM, allow the usability properties specified in the pattern catalog to be met. We defined and applied a set of 50 rules in total for the FM, which can be categorized as rules to control certain features of the same group of patterns and rules to restrict features of different groups of patterns. Examples of rules include:

*R#1: TopVideo # LeftVideo # TopMenu => ¬LeftMenu*

*R#2: LeftVideo # LeftMenu => ¬ LeftCont*

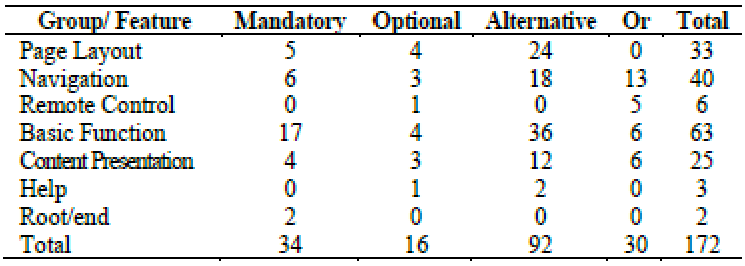
R#1 is a restriction that allows placing the video in the upper left while prohibiting placement of the menu in the same position on the screen;

this rule is related to the Full-screen with video and Menu patterns, which correspond to different groups. R#2 restricts the position of elements on the screen: if the video and the menu will be on the left side, the content should not be on the left; this rule is related to the Full-screen with video, Menu, and Content patterns. R#1 and R#2 are rules that control the validation of different groups of patterns: R#1 concerns patterns of the Page Layout and Navigation groups, while R#2 concerns patterns of the Page Layout, Navigation, and Content groups.

In summary, to model the 27 interaction design patterns for the design of an SPL, a total of 172 features were required, as detailed in Table 1.

**Table 1**

Details of Features in the Model



| Group/ Feature | Mandatory | Optional | Alternative | Or | Total |
|---|---|---|---|---|---|
| Page Layout | 5 | 4 | 24 | 0 | 33 |
| Navigation | 6 | 3 | 18 | 13 | 40 |
| Remote Control | 0 | 1 | 0 | 5 | 6 |
| Basic Function | 17 | 4 | 36 | 6 | 63 |
| Content Presentation | 4 | 3 | 12 | 6 | 25 |
| Help | 0 | 1 | 2 | 0 | 3 |
| Root/end | 2 | 0 | 0 | 0 | 2 |
| Total | 34 | 16 | 92 | 30 | 172 |

Source. The Authors.

## 6. SPL-IDTV Tool

### *6.1. Design and implementation*

The SPL-iDTV tool implements the abstract, generic conceptual model described above. SPL-iDTV was implemented in Java and FeatureIDE [11] with the AHEAD component [12]. This last component allows applying feature-oriented programming [9] techniques. The central idea of the implementation is to modularize iDTV design patterns as features and automatically generate the required sections of code in an NCL document.

The class diagram in Fig. 4 illustrates the most important classes that make up the tool. These classes have been divided into two groups. The classes on the left (App, WinNewProduct, Main, Product, WritelnFile and Connector) are the classes that control and execute the entire process of specifying, configuring, and generating code, regardless of the patterns that are chosen. For example, the App class is the main module that controls all of the execution, offers the functionality of creating a configuration, and allows generating an NCL application, and the Main class realizes the invocation of the classes and methods that implement the patterns involved in a given configuration and controls the entire generation of a final product (the NCL code) through invocation of the WinNewProduct class.

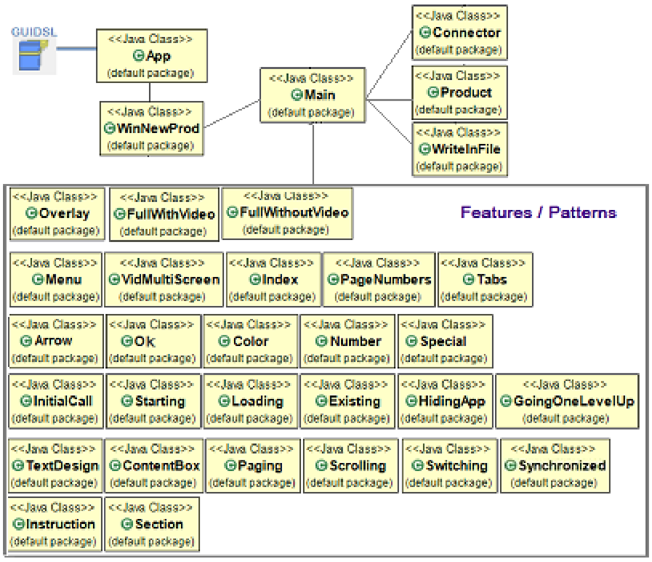

GUIDSL
<<Java Class>> App (default package)
<<Java Class>> WinNewProd (default package)
<<Java Class>> Main (default package)
<<Java Class>> Connector (default package)
<<Java Class>> Product (default package)
<<Java Class>> WriteInFile (default package)
Features / Patterns
<<Java Class>> Overlay (default package)
<<Java Class>> FullWithVideo (default package)
<<Java Class>> FullWithoutVideo (default package)
<<Java Class>> Menu (default package)
<<Java Class>> VidMultiScreen (default package)
<<Java Class>> Index (default package)
<<Java Class>> PageNumbers (default package)
<<Java Class>> Tabs (default package)
<<Java Class>> Arrow (default package)
<<Java Class>> Ok (default package)
<<Java Class>> Color (default package)
<<Java Class>> Number (default package)
<<Java Class>> Special (default package)
<<Java Class>> InitialCall (default package)
<<Java Class>> Starting (default package)
<<Java Class>> Loading (default package)
<<Java Class>> Existing (default package)
<<Java Class>> HidingApp (default package)
<<Java Class>> GoingOneLevelUp (default package)
<<Java Class>> TextDesign (default package)
<<Java Class>> ContentBox (default package)
<<Java Class>> Paging (default package)
<<Java Class>> Scrolling (default package)
<<Java Class>> Switching (default package)
<<Java Class>> Synchronized (default package)
<<Java Class>> Instruction (default package)
<<Java Class>> Section (default package)

**Figure 4**

Class Diagram for SPL-iDTV Tool

Source. The Authors.

The classes on the right correspond to each of the specific features of the FM and interaction design patterns (Overlay, FullWithVideo, FullWithoutVideo, Menu, VidMultiScreen, Index, PageNumbers, Tabs, Arrow, Ok, Color, Number, Special, InitialCall, Starting, Loading, Exiting, HidingApp, GoingOneLevelUp, TextDesign, ContentBox, Paging, Scrolling, Switching, Synchronized, Instruction, and Section).

The classes that correspond to the concrete features of the FM (patterns and pattern elements) contain all the logic needed to implement a pattern in an NCL document. For this reason, some attributes are concordant with NCL tags, such as region, descriptor, media, and link, which are required in NCL. But each class also specifies its own data for the implementation of the pattern in question.

### *6.2. Instrumentation*

The SPL-iDTV tool generates an application in three steps:

(1) Configuration design: The first step for the generation of a product is to indicate the product's features (patterns). To do this, the user must specify which interaction elements the application will incorporate (images, text, menus, buttons, etc.) from the design patterns. For this purpose, SPL-iDTV offers a menu of characteristics with all the patterns. At this point, each group with its set of patterns is identified and ordered. An SPL product is specified by selecting or canceling the selection of features according to the user's needs. A configuration is obtained as the result.

(2) Media selection: Once the configuration is created, the user must specify the media (image, video, text) required for the design. For this, SPL-iDTV offers an interface that allows searching for the media files from a browser window. Once the media are selected, the system assigns them to the corresponding configuration elements.

> (3) Generation of the product: The user selects a created configuration and the set of associated media and orders the automatic generation of the NCL code for the new product, which implements the functionality of the application.

These three simple steps allow users to automatically generate iDTV applications. The tool is very economical for developing different configurations using the same media until the product that best suits the needs of the TV producer is obtained. Similarly, once the optimal configuration that corresponds to the style of a TV program is found, updating the media is even simpler. The user repeats the task from step 2, that is, updating the media, according to the different broadcasts of a program. This last capability of the tool works as a template generator.

## 7. Evaluations

### *7.1. Prototyping, Experiments, and Case Studies*

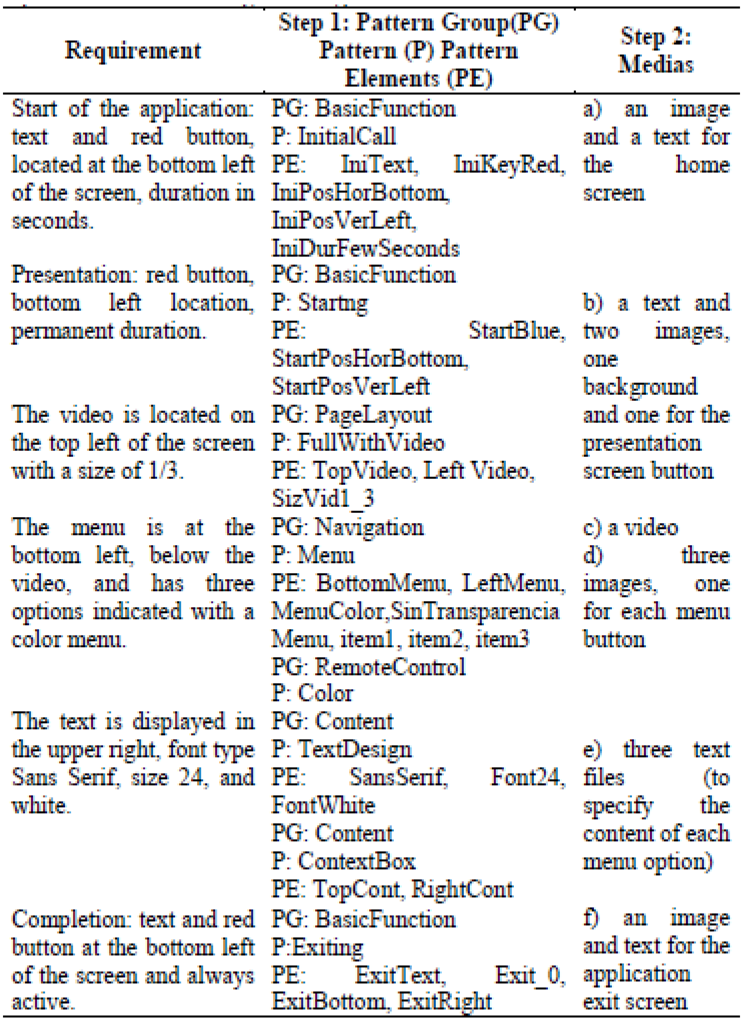
This part of the study consisted in using the SPL-iDTV tool to reproduce three iDTV applications, which were later executed using a Ginga emulator. The case study experimentation had several objectives: to prove the correct functionality of the SPL-iDTV tool, to verify the design of different configurations, to verify that the restrictions that fit the model are correct and flexible, to verify that the generated code is valid, and primarily to verify that the tool can be used to develop applications at the same level of complexity as manually developed application prototypes. For this last reason, the experimentation consisted of replicating manually developed prototypes that make intensive use of the patterns.

The prototypes were extracted from [8]. The Anke Late Night prototype is an entertainment and interview program, produced and issued by the Ilmenau Technology University (Germany) in 2004. The Music prototype is a musical program, also produced and broadcast by the same german university in 2004. The Sport prototype is a sports program, produced and broadcast by the UK's BBC since 1997.

After the prototypes were selected, the input sets (pattern information and media) necessary for steps 1 and 2 of the development using SPL-iDTV, as indicated in Section 4.2, were determined. Table 2 specifies the input for the Anke Late Night prototype, Table 3 for the Music prototype, and Table 4 for the Sport prototype.

**Table 2**

Input for the Anke Late Night Prototype.

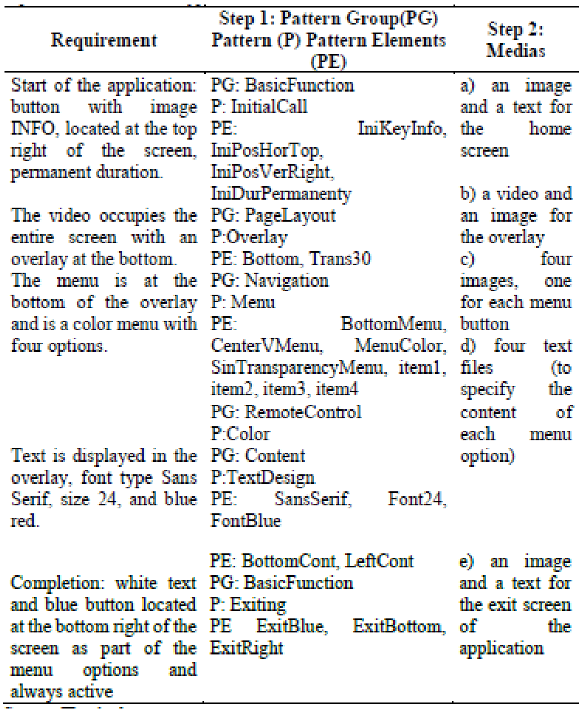


| Requirement | Step 1: Pattern Group(PG) Pattern (P) Pattern Elements (PE) | Step 2: Medias |
|---|---|---|
| Start of the application: text and red button, located at the bottom left of the screen, duration in seconds. | PG: BasicFunction<br>P: InitialCall<br>PE: IniText, IniKeyRed, IniPosHorBottom, IniPosVerLeft, IniDurFewSeconds | a) an image and a text for the home screen |
| Presentation: red button, bottom left location, permanent duration. | PG: BasicFunction<br>P: Startng<br>PE: StartBlue, StartPosHorBottom, StartPosVerLeft | b) a text and two images, one background and one for the presentation screen button |
| The video is located on the top left of the screen with a size of 1/3. | PG: PageLayout<br>P: FullWithVideo<br>PE: TopVideo, Left Video, SizVid1_3 | |
| The menu is at the bottom left, below the video, and has three options indicated with a color menu. | PG: Navigation<br>P: Menu<br>PE: BottomMenu, LeftMenu, MenuColor,SinTransparencia Menu, item1, item2, item3<br>PG: RemoteControl<br>P: Color | c) a video<br>d) three images, one for each menu button |
| The text is displayed in the upper right, font type Sans Serif, size 24, and white. | PG: Content<br>P: TextDesign<br>PE: SansSerif, Font24, FontWhite<br>PG: Content<br>P: ContextBox<br>PE: TopCont, RightCont | e) three text files (to specify the content of each menu option) |
| Completion: text and red button at the bottom left of the screen and always active. | PG: BasicFunction<br>P:Exiting<br>PE: ExitText, Exit_0, ExitBottom, ExitRight | f) an image and text for the application exit screen |

Source. The Authors.

**Table 3**

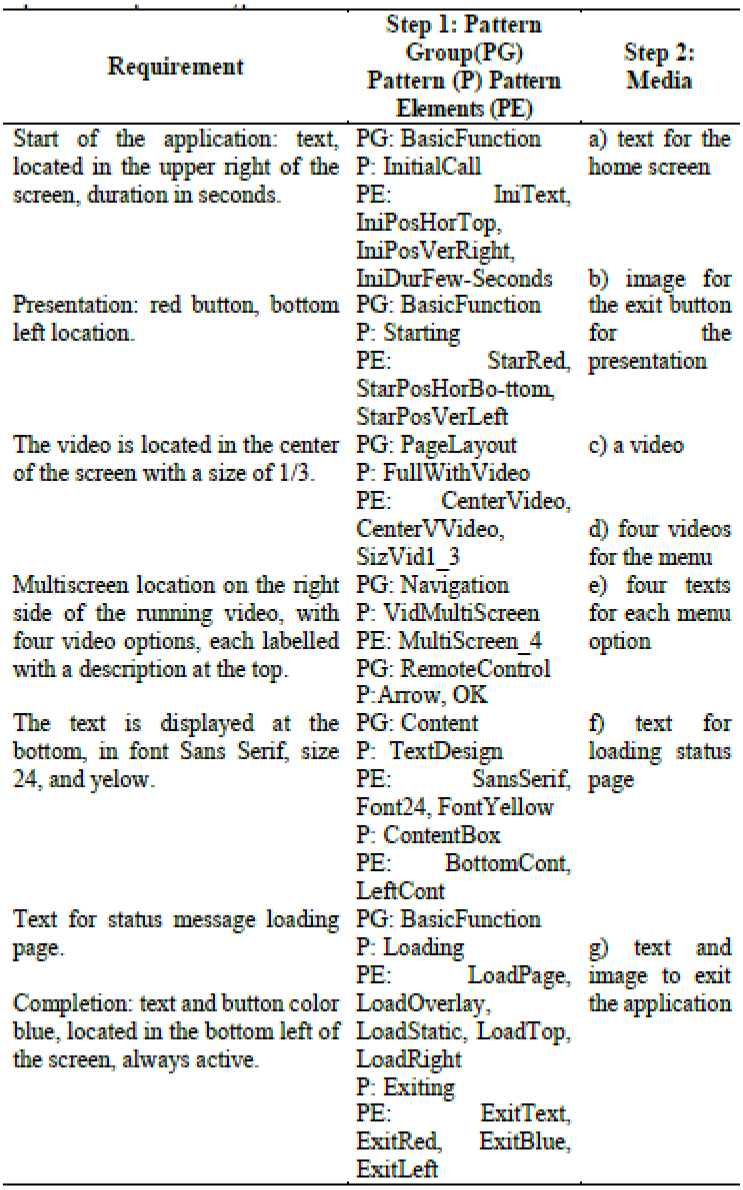
Input for the Music Prototype.



| Requirement | Step 1: Pattern Group(PG) Pattern (P) Pattern Elements (PE) | Step 2: Medias |
|---|---|---|
| Start of the application: button with image INFO, located at the top right of the screen, permanent duration. | PG: BasicFunction<br>P: InitialCall<br>PE: IniKeyInfo, IniPosHorTop, IniPosVerRight, IniDurPermanenty | a) an image and a text for the home screen |
| The video occupies the entire screen with an overlay at the bottom. | PG: PageLayout<br>P:Overlay<br>PE: Bottom, Trans30 | b) a video and an image for the overlay |
| The menu is at the bottom of the overlay and is a color menu with four options. | PG: Navigation<br>P: Menu<br>PE: BottomMenu, CenterVMenu, MenuColor, SinTransparencyMenu, item1, item2, item3, item4<br>PG: RemoteControl<br>P:Color | c) four images, one for each menu button<br>d) four text files (to specify the content of each menu option) |
| Text is displayed in the overlay, font type Sans Serif, size 24, and blue red. | PG: Content<br>P:TextDesign<br>PE: SansSerif, Font24, FontBlue<br>PE: BottomCont, LeftCont | |
| Completion: white text and blue button located at the bottom right of the screen as part of the menu options and always active | PG: BasicFunction<br>P: Exiting<br>PE ExitBlue, ExitBottom, ExitRight | e) an image and a text for the exit screen of the application |

**Table 4**

Input for the Sport Prototype.



| Requirement | Step 1: Pattern Group(PG) Pattern (P) Pattern Elements (PE) | Step 2: Media |
|---|---|---|
| Start of the application: text, located in the upper right of the screen, duration in seconds. | PG: BasicFunction<br>P: InitialCall<br>PE: IniText, IniPosHorTop, IniPosVerRight, IniDurFew-Seconds | a) text for the home screen<br>b) image for the exit button for the presentation |
| Presentation: red button, bottom left location. | PG: BasicFunction<br>P: Starting<br>PE: StarRed, StarPosHorBo-ttom, StarPosVerLeft | |
| The video is located in the center of the screen with a size of 1/3. | PG: PageLayout<br>P: FullWithVideo<br>PE: CenterVideo, CenterVVideo, SizVid1_3 | c) a video<br>d) four videos for the menu |
| Multiscreen location on the right side of the running video, with four video options, each labelled with a description at the top. | PG: Navigation<br>P: VidMultiScreen<br>PE: MultiScreen_4<br>PG: RemoteControl<br>P:Arrow, OK | e) four texts for each menu option |
| The text is displayed at the bottom, in font Sans Serif, size 24, and yelow. | PG: Content<br>P: TextDesign<br>PE: SansSerif, Font24, FontYellow<br>P: ContentBox<br>PE: BottomCont, LeftCont | f) text for loading status page |
| Text for status message loading page.<br>Completion: text and button color blue, located in the bottom left of the screen, always active. | PG: BasicFunction<br>P: Loading<br>PE: LoadPage, LoadOverlay, LoadStatic, LoadTop, LoadRight<br>P: Exiting<br>PE: ExitText, ExitRed, ExitBlue, ExitLeft | g) text and image to exit the application |

Source. The Authors.

The entries for each case study were entered into SPL-iDTV. Fig. 5 shows the SPL-iDTV configuration interface for the Anke Late Night prototype. The chosen options are highlighted with different shapes: rectangle without corner for the groups of patterns, rectangle for the patterns, and ellipse for the elements of each pattern.

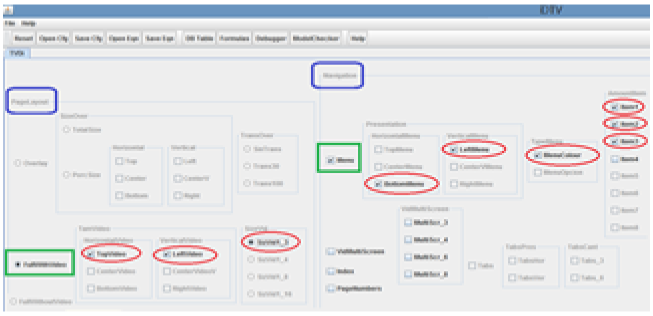
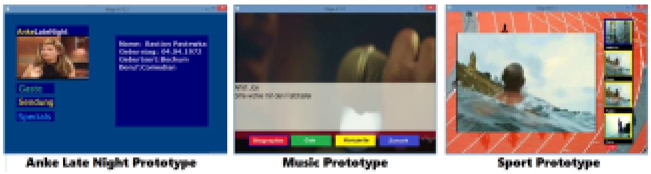


**Figure 5**
Configuration Design for Anke Late Night prototype
Source. The Authors.

After creating the three configurations and associating the corresponding media, the code for the applications was generated. The applications were then executed in a Ginga emulator. Fig. 6 presents images from these executions.



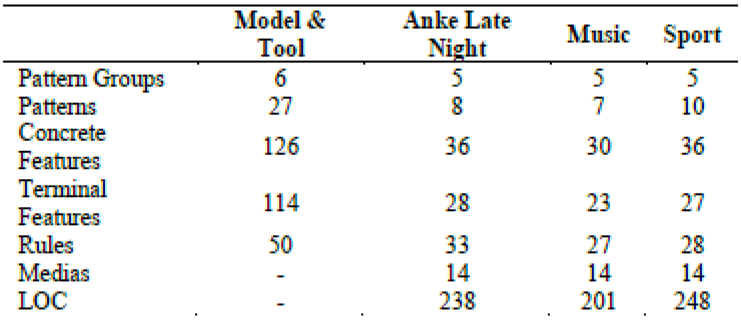
Anke Late Night Prototype Music Prototype Sport Prototype

**Figure 6**
Images of prototypes generated by SPL-iDTV
Source. The Authors.

The reproductions of the prototypes verify that the tool functions correctly, as it was possible to design diverse configurations that include different sets of patterns consisting of various elements and a considerable amount and diversity of medias. The code generated by the tool is correct and executed correctly. The prototypes developed with the tool are not trivial examples, and it was possible to automatically reproduce the manually designed and coded iDTV applications.

To complete the analysis of the prototypes developed in the experimentation, some results are listed in Table 5. The first column indicates the numbers of pattern groups, patterns, concrete and terminal features, and rules represented by the model and implemented by SPL-iDTV. The last three columns indicate the quantities for each prototype. The bottom two rows indicate the amount of media (images, videos, texts) and the number of automatically generated lines of code (LOC) for each prototype.

**Table 5**
Results.



| | Model & Tool | Anke Late Night | Music | Sport |
|---|---|---|---|---|
| Pattern Groups | 6 | 5 | 5 | 5 |
| Patterns | 27 | 8 | 7 | 10 |
| Concrete Features | 126 | 36 | 30 | 36 |
| Terminal Features | 114 | 28 | 23 | 27 |
| Rules | 50 | 33 | 27 | 28 |
| Medias | - | 14 | 14 | 14 |
| LOC | - | 238 | 201 | 248 |

Source. The Authors.

### *7.2. Evaluation of the SPL-iDTV*

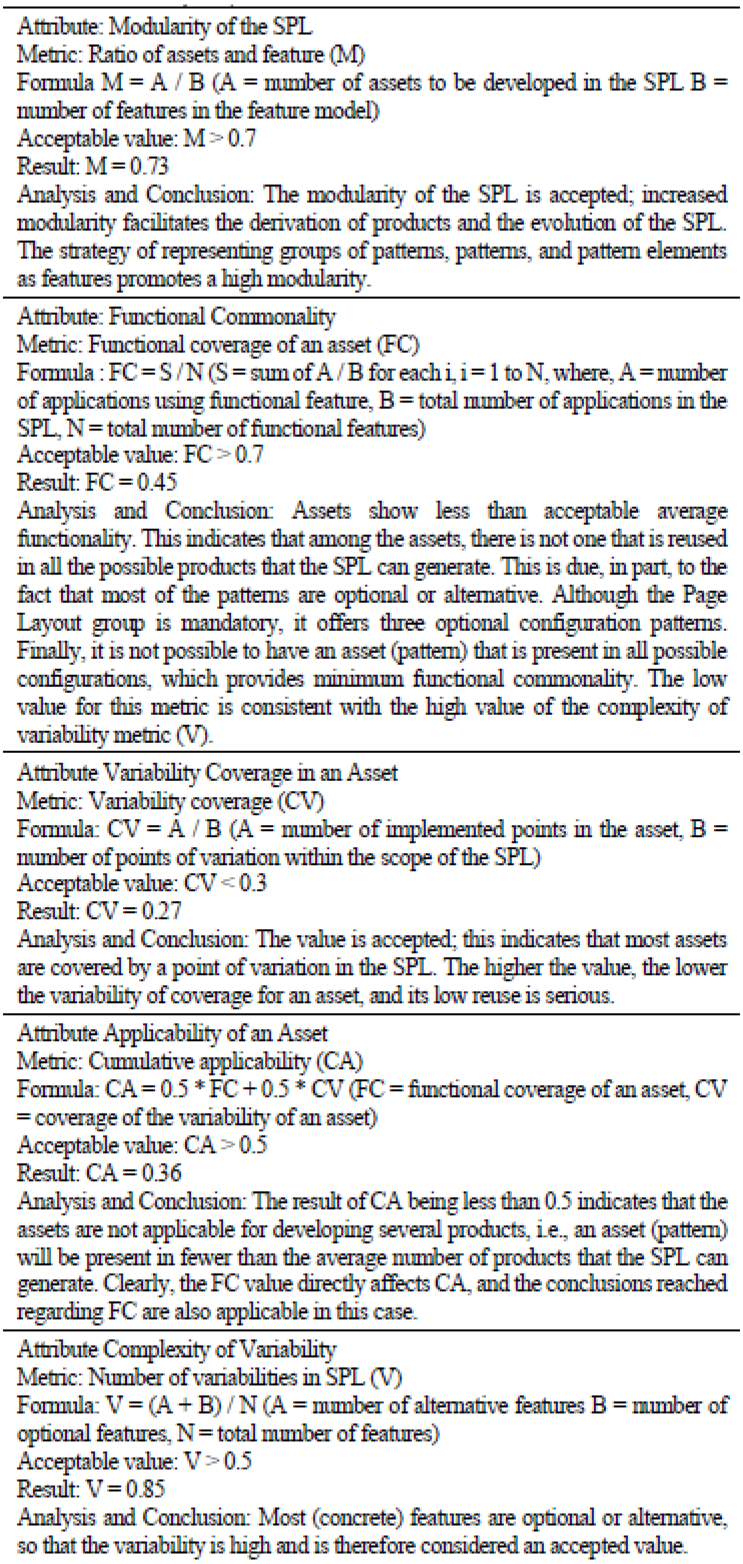
We conducted an evaluation of the quality of the SPL developed with SPL-iDTV, using the method proposed in [13]. This method is based on the quality standard SQuaRE. The main advantage of this method is that it is general and can be applied to any developed SPL. In addition, it is flexible insofar as it allows the independent evaluation of activities at any stage of the SPL development. Although the quality assessment process always involves the same activities at each stage of the SPL's life cycle, these should be adapted for the conditions of each phase.

In this study, we applied this method to evaluate the maintainability of the SPL. A high degree of maintainability indicates that a product is structured in a way that makes it is easy to modify. This property is one of the most evaluated properties for SPLs, since it considers their modularity and reusability. Modularity analyzes the active proportion by feature, and reusability employs four metrics to independently mediate commonality and variability. Commonality is evaluated by taking into account aspects related to the reuse of assets, because these are common parts of different products, and variability is evaluated based on the SPL's richness and complexity. The attributes and metrics for commonality involve functional commonality, the functional coverage of an asset, and its cumulative applicability. The attributes and metrics for variability include the variability in an asset and the complexity of the variability of the SPL, captured by measuring the number of variabilities in the SPL.

The results of the evaluation are presented in Table 6. This table also provides the formulas used for the calculations, the acceptable values, the bases, the derived metrics, and the dependencies between them. The possible values of the metrics are continuous values between 0 and 1. Since these are subjective values, we consider each evaluated attribute as accepted or not accepted for an independent reason. The values for the calculation of the metrics were taken from the statistics provided by FeatureIDE.

**Table 6**

Evaluation of SPL quality.



| |
|---|
| Attribute: Modularity of the SPL<br>Metric: Ratio of assets and feature (M)<br>Formula M = A / B (A = number of assets to be developed in the SPL B = number of features in the feature model)<br>Acceptable value: M > 0.7<br>Result: M = 0.73<br>Analysis and Conclusion: The modularity of the SPL is accepted; increased modularity facilitates the derivation of products and the evolution of the SPL. The strategy of representing groups of patterns, patterns, and pattern elements as features promotes a high modularity. |
| Attribute: Functional Commonality<br>Metric: Functional coverage of an asset (FC)<br>Formula : FC = S / N (S = sum of A / B for each i, i = 1 to N, where, A = number of applications using functional feature, B = total number of applications in the SPL, N = total number of functional features)<br>Acceptable value: FC > 0.7<br>Result: FC = 0.45<br>Analysis and Conclusion: Assets show less than acceptable average functionality. This indicates that among the assets, there is not one that is reused in all the possible products that the SPL can generate. This is due, in part, to the fact that most of the patterns are optional or alternative. Although the Page Layout group is mandatory, it offers three optional configuration patterns. Finally, it is not possible to have an asset (pattern) that is present in all possible configurations, which provides minimum functional commonality. The low value for this metric is consistent with the high value of the complexity of variability metric (V). |
| Attribute Variability Coverage in an Asset<br>Metric: Variability coverage (CV)<br>Formula: CV = A / B (A = number of implemented points in the asset, B = number of points of variation within the scope of the SPL)<br>Acceptable value: CV < 0.3<br>Result: CV = 0.27<br>Analysis and Conclusion: The value is accepted; this indicates that most assets are covered by a point of variation in the SPL. The higher the value, the lower the variability of coverage for an asset, and its low reuse is serious. |
| Attribute Applicability of an Asset<br>Metric: Cumulative applicability (CA)<br>Formula: CA = 0.5 * FC + 0.5 * CV (FC = functional coverage of an asset, CV = coverage of the variability of an asset)<br>Acceptable value: CA > 0.5<br>Result: CA = 0.36<br>Analysis and Conclusion: The result of CA being less than 0.5 indicates that the assets are not applicable for developing several products, i.e., an asset (pattern) will be present in fewer than the average number of products that the SPL can generate. Clearly, the FC value directly affects CA, and the conclusions reached regarding FC are also applicable in this case. |
| Attribute Complexity of Variability<br>Metric: Number of variabilities in SPL (V)<br>Formula: V = (A + B) / N (A = number of alternative features B = number of optional features, N = total number of features)<br>Acceptable value: V > 0.5<br>Result: V = 0.85<br>Analysis and Conclusion: Most (concrete) features are optional or alternative, so that the variability is high and is therefore considered an accepted value. |

Source. The Authors.

The FM defined for the creation of the proposed SPL presents an acceptable degree of maintainability. Each asset is defined by a feature, which facilitates the modularity of the model. The level of reuse of an asset covered by the functional features of the model does not reach the average possible value, while the coverage of variability is favorable. The predominance of optional and alternative features (variability) controls this tension, giving the model a high level of flexibility.

## 8. Conclusion

In Section 3, we describe some tools to develop iDTV applications for Ginga middleware. A group of these tools clearly aimed at developers [14-19,23], since they require knowledge of programming and software design. These tools offer different levels of reuse, placing the frameworks of specific domains [15,18] on a more prominent level than general-purpose text editors. The generators [20,21,24-26] are intended for TV producers, allowing applications to be made mainly using predesigned template, of general purpose or specific domains. [27] an assistant to generate NCL applications with a certain level of usability for end users. Here SPL-iDTV presents an advantage, since it not only generates the code of a certain configuration, but also allows designing a configuration that fulfills the function of a template. Thus, a user could design configurations that are then used as templates or style guides, in specific domains. Among the generators mentioned, the possibilities that the generated applications have usability criteria as SPL-iDTV does, are very limited, this marks an important difference. The applications generated with SPL-iDTV do not require subsequent evaluations of usability. Furthermore there is no need to spend time in the design of iDTV applications that use usability guidelines, which means lower costs.

This study offers two main contributions. The first is a model that enables building an SPL to generate families of iDTV applications. Three important advantages of this model are as follows: It is an abstract model because it is independent of the DTV standard, middleware, and interactivity programming language used; it is defined and structured based on proven usability patterns; and it is extensible, as it can easily incorporate new patterns (features). The second contribution is a tool that supports deployment of the SPL for the design and creation of configurations (templates) and automatic generation of Ginga-NCL code. These capabilities allow development of iDTV applications with reduced time, cost, and effort.

The evaluations we conducted indicate that the products generated with this approach are at a similar level to applications that were manually produced, even though the tool is not yet a professional version. The proposed approach promotes the automatic reuse of design and code, the modular design facilitates the derivation of products and the evolution of the product line, and the level of supported variability is high, which enhances the tool's ability to design flexible configurations.

Our planned objectives for future work are (1) to incorporate more patterns in the model and in the tool, in order to implement the entire catalog of patterns, (2) to add a module to the tool that will allow configuring different target languages for the generated code, and (3) to optimize the code generated by the tool.

## References

[1] Rodrigues, R. e Soares, L., Produção de conteúdo declarativo para TV Digital. Proceedings of SemiSH-XXXIII Seminario integrado de software e hardware. Campo Grande, Brazil. pp. 287-300, 2006.

[2] Fernández, F. and Goldenberg, S., Aplicaciones interactivas para la Televisión Digital en Chile. Cuadernos de Información, 22, pp. 6-17. 2008. DOI: 10.7764/cdi.22.86

[3] Nielsen, J., Usability engineering. Morgan Kaufmannm, Ed. San Francisco, CA, USA, 1993.

[4] Solano, A., Rusu, C., Collazos, C., Roncagliolo, V., Arciniegas J. and Rusu, V., Usability heuristics for interactive digital television. Proceedings of the 3rd International Conference on Advances in Future Internet. IARA, Nice, France, 2011.

[5] Dix, A., Finlay, J., Abowd, G. and Beale, R., Human-computer interaction, 3rd ed., Pearson, Edinburgh, UK, 2003.

[6] Soares, L., Rodrigues, R. and Moreno, M., Ginga-NCL: the declarative environment of the Brazilian digital TV system. Journal of the Brazilian Computer Society, 12, pp. 37-46. 2007. DOI: 10.1007/BF03192400

[7] Clements, P. and Northrop, L., Software product lines: practices and patterns. Addison-Wesley Professional, Boston, MA, USA, 2002.

[8] Kunert, T., User-centered interaction design patterns for interactive digital television applications. Springer-Verlag, London, UK, 2009. DOI: 10.1007/978-1-84882-275-7

[9] Apel, S. and Kästner, C., An overview of feature-oriented software development. Journal of Object Technology, 8(5), pp. 49-84. 2009. DOI: 10.5381/jot.2009.8.5.c5

[10] Hevner, A. and Chatterjee, S., Design research in information systems, theory and practice. Springer, Boston, MA, USA, 2010. DOI: 10.1007/978-1-4419-5653-8.

[11] Kästner, C., Thüm, T., Saake, G., Feigenspan, J., Leich, T., Wielgorz, F. and Apel, S., Feature IDE: a tool framework for feature-oriented software development. Proceedings of the International Conference on Software Engineering, IEEE Computer Society, Vancouver, Canada, 2009, pp. 611-614, DOI: 10.1109/ICSE.2009.5070568

[12] Batory, D., AHEAD tool suite. [online]. 2017. Available at: https://www.cs.utexas.edu/~schwartz/ATS/fopdocs.

[13] Montagud, G., Un método para la evaluación de la calidad de líneas de productos software basado en SQuaRE. MSc. Tesis. [en línea]. 2009. RiUnet repository UPV Disponible en: Http://hdl.handle.net/10251/11923

[14] Guimarães, R., de Resende-Costa, R. and Soares, L., Composer: authoring tool for iTV programs. European Conference on Interactive Television, Springer Berlin Heidelberg, pp. 61-71, 2008.

[15] Barboza, D. and Clua, E., Ginga Game: a framework for game development for the interactive digital television. Proceedings of the VIII Brazilian

Symposium on Games and Digital Entertainment. IEEE Computer Society, Rio de Janeiro, Brazil, 2009, pp. 162-167. DOI: 10.1109/SBGAMES.26

[16] Pequeno, H., Gomes, G. and Castro, M., FrameIDTV: a framework for developing interactive applications on digital television environments. International Journal on Network and Computer Applications, 33(9), pp. 503-511, 2010. DOI: 10.1016/j.jnca.2010.01.002.

[17] Honorato, G. and Barbosa, S., NCL-inspector: towards improving NCL code. Proceedings of the ACM Symposium on Applied Computing, ACM. Sierre, Switzerland, 2010, pp. 1946-1947. DOI: 10.1145/1774088.1774500

[18] Segundo, R., da Silva, J. and Tavares, T., ATHUS: a generic framework for game development on Ginga middleware. Proceedings of Brazilian Symposium on Games and Digital Entertainment. IEEE, Florianopolis, Brazil, 2010, pp. 89-96. DOI: 10.1109/SBGAMES.2010.28

[19] Neto, S., Bezerra, P. and Dias, D., ITV-Learning: a prototype for construction of learning objects for interactive digital television. Proceedings of the International Conference on the Future of Education, Pixel. Florence, Italy, 2012, pp. 486-490.

[20] Arroyo, M., Schwartz, S., Cardozo, S. and Tardivo, L., CreaTVDigital: Composición Visual de Aplicaciones Interactivas para TV Digital. Proceedings of the 41st Jornadas Argentinas de Informática, SADIO, La Plata, Argentina, 2012, pp. 305-321.

[21] Bernal, I., Cabezas, G. and Quezada, M., Sistema de generación de aplicaciones interactivas para TV Digital para la evaluación de servicios masivos. Revista Politécnica, 32(2), pp. 11-22, 2013

[22] Gutiérrez-Duarte, S., Guía para el desarrollo de aplicaciones interactivas en TDT para Colombia. Bdigital Repository. Universidad Nacional de Colombia, Bogota, 2013.

[23] Cevallos-Salas, D., Cevallos-Salas, F., Bernal-Carrillo, I. and Mejía-Navarrete, R., Generación automática de contenido para aplicaciones interactivas de Televisión Digital con Ginga-NCL. Proceedings of the III Jornadas iberoamericanas de difusión y capacitación sobre aplicaciones y usabilidad de la TVD, SBC, João Pessoa, Brazil, 2014, pp. 114-121.

[24] Oyarzo, F., Herrera, F. and Casas, S., API TVD: a wizart for interactive applications for Digital TV. Proceedings of the XL Latin American Computer Conference, IEEE. Montevideo, Uruguay, 2014, pp. 1-8.

[25] De Souza, V., Galabo, A., Fernándes-Pinto, R., Araujo, F. e De Salles Sores, C., Plataforma online orientada a templates para a criação de aplicativos de telejornalismo. Proceedings of JAUTI, RedAUTI 2014, Palma de Mallorca, Spain, 2014, pp. 102-108.

[26] Ochoa, S., Pillajo, A., Acosta, F. and Olmedo, G., Template generator: software para la generación de aplicaciones interactivas para la televisión digital terrestre a partir de plantilla Ginga y LUA. Proceedings of III Jornadas iberoamericanas de difusión y capacitación sobre aplicaciones y usabilidad de la TVDi, Palma de Mallorca, Spain, 2014, pp. 109-113.

[27] Casas S., Herrera F., Oyarzo F. and Trinidad F., Dr. Nau, a Web generator of interactive applications for Digital TV. In: Abásolo, M., Silva, T. and González, N., Eds., Applications and usability of interactive

TV. Communications in Computer and Information Science, vol. 1004. Springer, 2019. DOI: 10.1007/978-3-030-23862-9_6

[28] Pohl, K., Böckle, G. and van der Linden, F., Software product line engineering: foundations, principles and techniques. Springer-Verlag, New York, USA, 2005. DOI: 10.1007/3-540-28901-1

[29] Batory, D., Feature models, grammars, and propositional formulas. NAMES (Eds.), LNCS: Vol. 3714. Proceedings of Software Product Lines, 9th International Conference, Springer. Rennes, France, 2005, pp. 7-20. DOI: 10.1007/11554844_3

[30] Kang, K., Cohen, S., Hess, J., Novak, W. and Peterson, A., Feature-oriented domain analysis (FODA) feasibility study (Technical Report CMU/SEI-90-TR-021, SEI/CMU). [online]. 1990. SEI Digital Library. Available at: https://resources.sei.cmu.edu/asset_files/technicalreport/1990_005_001_15872.pdf. 1990.

[31] Dammagh, M. and Troyer, O., Feature modeling tools: evaluation and lessons learned. In: De Troyer, O., Bauzer-Medeiros, C., Billen, R., Hallot, P., Simitsis, A. and Van Mingroot, H., Eds., LNCS, Vol. 6999. Advances in conceptual modeling: recent developments and new directions: ER 2011, Springer, Berlin, Germany: 2011, pp.120-129. DOI: 10.1007/978-3-642-24574-9_17

[32] Capilla, R., Bosch, J., Trinidad, P., Ruiz-Cortés, A. and Hincheyd, M., An overview of Dynamic Software Product Line architectures and techniques: observations from research and industry. Journal of Systems and Software, 91, pp. 3-23, 2014. DOI: 10.1016/j.jss.2013.12.038

[33] Borchers, J., A pattern approach to interaction design. Wiley Eds., New York, USA, 2001.

[34] Tidwell, J., Common ground: a pattern language for human-computer interface design. [online]. 1999. Available at: http://www.mit.edu/~jtidwell/common_ground.html

[35] van Welie, M., A pattern library for interaction design. [online]. 2000. Available at: http://www.welie.com/patterns.

## Notes

**M. Miranda,** is an assistant at the Institute of Applied Technology of the National University of Southern Patagonia, Argentina. She received her MSc. from the same university at 2018. Her research interests include software development for Digital TV applications. ORCID: 0000-0001-7240-167X.

**S. Casas,** is an associate professor at the Institute of Applied Technology of the National University of Southern Patagonia, Argentina. She received his PhD from the University of Vigo of Spain at 2008. Her research interests include the study and application of software development improvement techniques. ORCID: 0000-0002-8289-6132

**How to cite:** Miranda, M. and Casas, S, Usable and Interactive Application Generator for Digital TV. DYNA, 86(211), pp. 174-183, October - December, 2019.