SyncSmarty: A Framework for Synchronizing Smart TV Applications with TV Programs

Cédrick Bamba N. and Cesar A.C. Teixeira
Department of Computer Science, Federal University of São Carlos, São Carlos, São Paulo, Brazil

Abstract - Smart TV applications are typically disconnected from the content of the tuned programming on the TV set. Some TV broadcasters offer specific applications for programs, however these applications are generally just loosely coupled or synchronized to the program. Furthermore, applications are fully under domain of TV broadcasters that have the information about the transmitted content. Based on the fact that synchronized information may arise from third-parties obtained by local content processing or offered directly by the viewers, new opportunities will be opened up for developing a bunch of new interesting applications aiming to promote the user interaction level in TV. Moreover, it will have new user interaction possibilities while using mobile devices and computers, furthermore a new business model will be emerged. In this paper, the SyncSmarty framework is presented and evaluated. The aforementioned framework offers several APIs that facilitate the development of Smart TV applications synchronized with TV program contents. The aim is to provide some facilities for developers to implement applications in this area in a clear approach without being concerned about low-level implementation details.

Keywords: Connected television, interactivity, smart TV, application integration, television programs, TV channels monitoring.

1 Introduction

The emerge of Digital TV and smart TV has been innovating the way people watch television and changing the passive paradigm where the user could only consume contents. In the Interactive digital TV environment, the user can interact with the TV contents because the TV broadcasters send common contents (i.e: novel, TV show, etcetera) along with their interactive application that can be executed on the TV processor.

However unlike the interactive scenario in iDTV, most of the time the audiovisual content of programs in the Connected TV does not communicate with the native TV applications and other TV platform features. The only basic interaction is the possibility to switch between the things are displaying on the TV screen either program or TV applications. This approach of connecting TV to the internet, where the audiovisual content is offered (without pre-defined mechanisms of communicating with additional TV-platform features) characterizes the difference between Smart TV and iDTV.

A study on the Smart TV markets, conducted by Strategy Analytics and BI Intelligence [3], assessed the sale of Smart TVs related to the other types of TVs / TV devices (Google's ChromeCast, Apple TV, etc.) predicted sale rate of the next three years. According to the survey whose results are shown in Fig. 1, the sale of Smart TVs is growing to dominate the overall TV market with 33% of global sales of TVs in 2013 and finished in 2014 with the rate of 44%. According to the survey, in 2017, Smart TVs will represent 73% of total sales of TVs throughout the world, 54% in 2015 and 63% 2016. Accordingly, it would suggest that the paradigm of Smart TVs should be popularized. Namely, however some studies have been done to explain why the Smart TV concept is not yet as popular as the Smartphone’s concept. In the following paragraphs we will discuss the result of some of these studies.

Bachelet [1] conducted a research in five European and North American countries in May 2013 and found out that only less than the half of 6115 Smart TV owners who were interviewed, connected their TVs to the Internet.

According to Bachelet [1], two elements are the reasons that why the Smart TV has not yet become popular and been widely accepted by consumers as much as Smartphones:

- The lack of content and interesting applications: although the majority of Smart TVs offer a wide range of content and applications, most of them are irrelevant and are not interesting to users.

- The Poor User Interface: a lack of rich user interface that can integrate TV applications with its audiovisual contents.

Schofield [7] mentioned that “if TVs are going to be truly smart they must do more than offer a wide variety of online video services. Instead they must add advanced functionality including voice control, motion control, advanced advertising, attractive user interfaces and two-way communications with other smart devices – so-called ‘second screens’– allowing these devices both to send video to the TV and know what is being watched. Manufacturers should focus less on adding more content and more on improving how users can interact with that content”.

Cédrick Bamba N. and Cesar A.C. Teixeira
Department of Computer Science, Federal University of São Carlos, São Carlos, São Paulo, Brazil
This suggests that the potential for interactivity of Connected TVs has not yet been properly utilized. Hence new synchronization mechanisms and interaction with TV environment can help to improve the user experience.

Based on the finding that says innovative interactivity has not been promoted and used adequately in Connected TV platforms is a major reason for its limited use. Another reason for low adherence to the interactive resources of Smart TV is because of the lack of facilities that allow the integration of Smart TV applications with their own audiovisual content. The observation of these aspects motivated the current proposed paper.

2 Related Work

Several authors reported some results of building frameworks designed to support TV applications development. However just few of them focus on reuse in Smart TV applications, specifically those ones synchronized with television programming.

Group Share-TV [5] proposes a framework called share-tv used for the development of converged applications centered on TV for GoogleTV and Ginga-J platforms. The share-tv allows the development of TV applications that include a generic mobile application. Ever since this generic application get downloaded from a given available TV IP, installed and run on mobile device, the communication with the TV convergent application will be initiated automatically in order to register and receive shared objects. While the objects are being received, the device show them on the screen allowing interaction on them. Compared to the work reported in this paper, share-tv also provides communication services and reuse, however, this framework is limited to applications based on GoogleTV and Ginga-J platforms. The main difference is that share-tv is used to develop TV convergent applications while the work reported in this paper focuses on building Smart TV applications synchronized with the content of TV programs.

Samsung Smart TV [6] presents a framework called AppsFramework. This framework encapsulates reusable modules for scene management, video playback / music, and so on. This makes it easier for the developer of Smart TV applications to avoid performing complicated sequences of calls to the operating system in order to manage scenes (focusing, showing and hiding events) of an application, for instance. Some of these modules reused in the framework proposed in this paper.

Freitas and Teixeira [4] proposed an architecture for supporting the development of ubiquitous applications in home networks focusing on Digital TV. The proposed architecture consists of communication interface with home devices, a protocol layer for automatic service discovery, and so on. Although the architecture is designed to be implemented in iDTV middleware, some of its reusable artifacts such as the aforementioned ones were used and implemented in the framework proposed in this paper.

The framework for building synchronized smart tv applications with tv programs presented in this paper is different with all the aforementioned works from this scene that it is a reference framework that allows more efficiency and less cost in building multi-platform applications in this field.

3 Synchronization and Notification

To achieve the goal of providing integrated applications to television programming, it is essential to know what is being presented to the audience every moment. Then this information can be published to stakeholders and used to promote synchronization between program and application.

The following subsections are devoted to two issues, synchronization and notification. We discuss some aspects of synchronization that Smart TV applications should be concerned to promote integration with television programming and how the demands of notification services are generated.

3.1 Synchronization Aspects

Teixeira et al. [8] studied the synchronization in the context of multimedia applications and considered that synchronization is a mechanism to guarantee that actions can happen according to the defined time reference set by a clock or established by the occurrence of events. It considers the tolerances that vary according to the type of application. In case of television programming the characteristic of event is to happen according to the defined time reference set by a clock.

Three other relevant aspects for the integration of applications with television programming are: Source - refers to anyone who provides information about the occurrence of the event; Coupling - is related to the tolerance allowed by the application in deviations from the reference time; Exposition - categorizes two possible situations: explicit and implicit synchronization information.

3.2 Notification

To promote the used synchronization in the context of this work, we need to establish a communication protocol between the part that needs to receive synchronization signals (TV application) and the one who provides these signals (synchronization services). To do so it is necessary to create a
notification API that is an interface that allows applications to access the various offered notification features for registering such notifications in synchronization service and the last one in turn, will notify the applications about events (related to those notifications that previously were registered) occur in TV programming.

Notifications addressed in this work have various types such as start and end of trading blocs, start, pause and end of the TV programs, sex scenes, violence, crime and some notifications generated by the events triggered by the user. For example, a user can send notification related to the information about his current tuned channel to another user in TV social system.

4 SyncSmartv

SyncSmartv is a reference framework designed to facilitate the development of integrated Smart TV applications. The goal is to create mechanisms that allow developers to build applications in this field in a transparent manner, in a clear approach without being concerned about low-level implementation details. Additionally the framework aims to allow developers to think more about the business logic of their applications; hence, they can build integrated applications with lower cost and effort. In this work, when we talk about integrated applications, we mean those ones synchronized with the television programming.

4.1 Design Considerations

Bosch et al. [2] report that framework development is different from a common application development. This is because of that framework’s design needs to cover all relevant features of a particular domain and not just those ones of specific application. This is why it is important to consider the following when developing framework for Smart TV integrated applications. Generally, there are six issues to consider. First, How to synchronize the TV content with Smart TV applications. Second, how mobile devices (smartphones, tablets, and others), which are in the same space with TV can interact with it. Third, how connected mobile devices in the TV environment can detect the presence of available TV service for use. Forth, how Smart TV applications can act over the TV controls such as changing channels, controlling volume and so on. Fifth, how ticker applications must share the remote control with the TV. Finally how a Smart TV application can identify the channel, which is being watched by the user.

In order to build the reference framework to support the issues that was discussed above we considered the set of artifacts in Fig. 2 for extracting the common and reusable modules existing between them. The extracted modules are a part of the SyncSmartv components.

In the right side there are some built applications (shown in gray rectangles) that have some typical characteristics of one or more applications or components located in the left (represented in the form of ellipse). These applications are simple that are not directly interesting for end users. These applications built in order to explore features of Smart TV. An application of the right side presents similar characteristics to a component of application on the left hand while there is a line that joins each rectangles in right hand to the corresponding application in the right and passes over the ellipses.

The set of artifacts considered in Fig. 2 is composed of nine main applications: the TickerApp used to manage the use of remote control between applications and TV; the VideoList is in charge of receiving synchronized notifications generated by the viewer and then starts to playback the pre-defined video list; the MobileSync uses HTTP protocol to allow two-ways communication and sharing media content among the TV and all the existing devices in the same environment; the SharedApp allows the viewer to use the Smart TV as second screen in a synchronized way with the TV programming; the KaeptorInt implements some protocols of the third-services providers aiming to facilitate TV applications to receive synchronized notifications; the BackApp is responsible for switching channels automatically after that a synchronization event (starting scene crime, for example) is triggered; the DescriptionApp allows user to access more details of the product which is in focus in the current scene; the TVNewsApp allows user (on a given tuned channel) to be provided with informations related to the status of the broadcasting TV news and the TVMonitor, which contains most of the functionalities of the aforementioned applications that is one of the completed and integrated applications developed by the author of this paper as a proof of concept of the framework.

4.2 Development

Below, we illustrate only the domain analysis of the first problem (how to synchronize the TV content with Smart TV applications), that is, how some appropriate artifacts were extracted and used to resolve this problem.

First, the common features are identified while the applications of Fig. 2 were comparing and after a simplified design (can be a textual description) of artifacts needed to implement each feature, was made. Then a feasibility evaluation was done on the artifacts found by analyzing, for example the estimated effort in construction and reuse level, that means how many applications would take advantage of this artifact. Finally, we decided whether it is beneficial to implement them or not.

After identifying the first functional requirement (1. Means to receive synchronization information) for the first problem that already mentioned, two artifacts were drafted: 1. a login
service and connection establishment with the TV channel monitoring service 2. A registration and communication service with the registered channel. In Fig. 3, the use cases in yellow are functionalities of the artifact 1 (login service and connection establishment with the TV channel monitoring service) and in blue are part of the artifact 2 (registration service and communication with the registered channel).

4.2.1 SyncSmartv Architecture

The following figure depicts the architecture of the SyncSmartv. In a quick view this architecture is based on the AppFramework architecture [6] where the Framework layer in gray and Adaption Layer in blue were added. The SyncSmartv integrates a set of tools and components that enable a software engineer to quickly design, develop and deploy new Smart TV integrated application. The main SyncSmartv functionality is offered by using its API. Using the API specification a developer may build various applications such as monitoring applications that detect crime or sex scene and start a playback of some music from the viewer mobile device.

The figure consists of four layers, the first called Application Manager is where applications are managed and built using languages such as JavaScript, HTML, CSS and so on. Usually once the application is built, it is executed on the AppEngine Smart TV Platform layer. If the developer wants to build an integrated application, it can take advantage of the available features in the framework layer. Then this application uses the Adaption Layer to implement generic artifacts specified in the framework according to the need of each Smart TV platform used for application execution. Finally, the application is run in the Smart TV Platform layer. The next paragraphs provide an overview of SyncSmartv modules that meet the functional requirements listed in Subsection 4.1.

4.2.2 Modules Overview

The SyncSmartv set of modules we present in this section is a set of JavaScript classes and interfaces, bind with synchronization services, which aim to add a level of integration between the Smart TV, the application and for enhancing TV social experience. The Sync Event Manager module provides an interface for Smart TV applications for receiving the notifications of the occurring events in television programming. The Communication Manager provides the features needed to establish a two-way connection between Smart TV applications and mobile devices (smartphones, tablets, etc.) found in the same environment. The User Device Discovery module provides necessary mechanisms for the connected devices such as smartphones, tablets, printers, etc can discover the existence of some available TV services for use. The TV Control Manager module provides functionalities for Smart TV applications that act on TV controls. The Input Manager provides mechanisms to manage the use of remote control buttons between applications and the TV. The TV Channel ID provides mechanisms to detect the channel being watched by the user, through a clear approach.

4.2.3 Framework Instantiation

As a proof of concept of the framework, we developed a set of tools to implement test with developers. This set of tools were mainly developed using JavaScript. The Fig. 5 depicts an overview of the set.

The set is composed of two main parts: the Back-end used to store the information of users and devices and to allow communication and sharing of media content among different components of the set; and the Front-end, which contains applications executed on mobile devices and Smart TV platforms.
Back-end

The **Back-end** of this work provides web services for Smart TV discovery services and for managing the users of social TV systems. The aforementioned services were developed using the Grails framework (Fig. 5(a)). Moreover, the **Back-end** allows communication and sharing the media content among the different applications respectively using Apache ActiveMQ message broker (Fig. 5(b)) and PHP server (Fig. 5(c)).

Front-end

The **Front-end** is based on client-side (PC, Smart TV, Tablet, Smart Phone, etc.) that was developed using Apache Cordova + HTML5 + JQuery (Fig. 5(d)) and JavaScript-based SyncSmartv API’s.

5 Framework Validation

5.1 Developing Experimentation

To validate the framework proposed here we used the experimental method proposed by Wohlin et al. [9]. The experiment consisted of a comparative study of the development processes of two versions of TVMonitor application, one built with the reuse approach using the proposed framework and other built without this approach.

The experimental phases performed in the study will be expatiated in the following subsections.

5.1.1 Definition of the Experiment

The objective of this study was:

- To analyse the use of the proposed framework in the construction of Smart TV applications synchronized with television programing;
- With the purpose of evaluation
- Regarding the efficiency in terms of time spent and productivity;
- From a point of view of software developers;
- In the context of undergraduate and graduated in computer science and computer engineering. It is important to point out that in this experiment twelve (12) developers and one object were considered (TVMonitor).

5.1.2 Design of the Experiment

Efficiency was selected as dependent variable for this experiment. Independent variables were the development method, the developed application, the development environment and development technologies.

Among the four independent variables, three were kept constant during the study:

- Application = TVMonitor ;
- Development environment = Eclipse IDE ;
- Development Technologies = HTML, CSS, JavaScript and JQuery.

Formulation of Hypotheses

M1: First Mobile Application (MobileSync)
M2: Second Mobile Application (AdminSync)
Sm: Smart TV Application (TVMonitor)
P: PHP Server
D: Discovery

S: Synchronization
C: Communication
K: Keymanagement
Ch: Channel identification
T: TV

Fig. 5. Implementation Overview.
For the formulation of the three hypotheses, the following metrics were considered:

- \( \tau \) - Total time spent by the team for developing Smart TV application synchronized with the TV program;
- \( \Pi \) - Team Productivity in terms of produced lines of code (LOC) per unit time (\( \Pi = \text{LOC} / \tau \));
- \( \mu_{\tau} \) - Average of the spent time by the teams for developing Smart TV application synchronized with the TV program;
- \( \mu_{\Pi} \) - Average productiveness of the teams in the development of Smart TV application synchronized with the TV program.

We have a null hypothesis and its two corresponding alternatives:

- **Null Hypothesis (H0):** There is no difference between teams who used the proposed framework and teams that did not use while developing TVMonitor application regarding the efficiency (\( \varepsilon \)) of the team.

  \[ H_0: \varepsilon_{\text{framework}} = \varepsilon_{\text{without framework}} \Rightarrow \mu_{\text{framework}} = \mu_{\text{without framework}} \]

- **Alternative Hypothesis (H1):** Teams who use the proposed framework for building TVMonitor application are generally more efficient than those ones who developed without the use of framework.

  \[ H_1: \varepsilon_{\text{framework}} > \varepsilon_{\text{without framework}} \Rightarrow \mu_{\text{framework}} < \mu_{\text{without framework}} \]

- **Alternative Hypothesis (H2):** Teams using the approach "without framework" for building TVMonitor application are generally more efficient than those developing with the use of framework.

  \[ H_2: \varepsilon_{\text{framework}} < \varepsilon_{\text{without framework}} \Rightarrow \mu_{\text{framework}} > \mu_{\text{without framework}} \]

### 5.1.3 Implementing the Experiment

In this step we prepared effectively the material needed to support the experiment, that means, the set of objects manipulated during the experimentation and which are defined in 4.2.3. In addition, were prepared documents that allowed the experimenter to exchange information with the participants.

The organization of the data in Fig. 6 is done according to the two development approaches used in this experiment: development with and without the use of the framework reported in this paper.

### 5.1.4 Analysis and Interpretation of Results

An initial analysis was done on the data collected in Fig. 6. It is important to note that the distribution efforts of the groups in the design and test phases of TVMonitor development was constant. However, there is a great discrepancy of development efforts for the groups that used the proposed framework during the implementation of TVMonitor. While groups which have not used the proposed framework, in average they spent 4 hours and 59 minutes but those who used the framework spent 2 hours and 08 minutes (a decrease of 57, 2%).

### Table 1: Data Collected

<table>
<thead>
<tr>
<th>Group</th>
<th>StartTPro</th>
<th>FinTPro</th>
<th>StartTImp</th>
<th>FinTImp</th>
<th>StartTTTest</th>
<th>FinTTTest</th>
<th>LCG Auto</th>
<th>LCG Man</th>
<th>TLOC</th>
<th>TT((\alpha))</th>
<th>TPrd((\alpha))</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Framework</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G3</td>
<td>14:30</td>
<td>14:47</td>
<td>15:00</td>
<td>16:10</td>
<td>16:20</td>
<td>16:30</td>
<td>2480</td>
<td>56</td>
<td>2536</td>
<td>1:37</td>
<td>1569</td>
</tr>
<tr>
<td>Average</td>
<td>00:20</td>
<td>2:08</td>
<td>00:53</td>
<td>3:01</td>
<td>1080</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without Framework</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G5</td>
<td>8:30</td>
<td>9:10</td>
<td>9:20</td>
<td>15:17</td>
<td>15:25</td>
<td>16:12</td>
<td>1882</td>
<td>427</td>
<td>2309</td>
<td>7:24</td>
<td>312</td>
</tr>
<tr>
<td>G6</td>
<td>8:30</td>
<td>9:02</td>
<td>9:10</td>
<td>14:48</td>
<td>15:00</td>
<td>15:33</td>
<td>1882</td>
<td>392</td>
<td>2274</td>
<td>6:43</td>
<td>339</td>
</tr>
<tr>
<td>Average</td>
<td>00:34</td>
<td>4:59</td>
<td>00:52</td>
<td>6:05</td>
<td>398</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**StartTPro:** The Start Time of the Project;
**FinTPro:** The Finish Time of the Project;
**StartTImp:** The Start Time of the Implementation;
**FinTImp:** The Finish Time of the Implementation;
**StartTTTest:** The Start Time of the Test;
**FinTTTest:** The Finish Time of the Test;
**LCG Auto:** Lines of Code Generated Automatically;
**LCG Man:** Lines of Code Generate Manually;
**TLOC:** Total Lines Of Code;
**TT:** Total Time;
**TPrd:** Total Productivity.

*Fig. 6: Data Collected.*

### Testing Hypotheses

The purpose of the t-test (Montgomery, 2000) is to verify that a variable differs between two independent samples, based on the arithmetic average and considering variability of its data items. Then with some degree of significance (\( \alpha \)), reject the null hypothesis (H0) and choose one of the alternative hypothesis (H1 or H2). The t-test formula is given by equation (1), where \( Sx^2 \) and \( Sy^2 \) are
the variances of each sample; $Sp$ is the dispersion; and $n$ and $m$ are the numbers of data items that each sample contains. In equation (2), $n+m-2$, typically noted by $gl$ is called the degree of test’s freedom.

$$t_0 = \frac{x_1 - x_2}{\sqrt{\frac{Sp_1^2}{n} + \frac{Sp_2^2}{m}}}$$  \hspace{1cm} (1)

$$Sp = \sqrt{\frac{(n-1)Sp_1^2 + (m-1)Sp_2^2}{n+m-2}}$$  \hspace{1cm} (2)

Once you have calculated $t_0$, $\alpha$ and $gl$, you can check the value of the standard $t$ in t-test distribution to see if $t_0$ is so significant.

If $|t_0| > t_{0.1000, 4}$ → REJECT $H_0$.
Otherwise, → $H_0$ NOT REJECTED and no conclusion is drawn from the experiment.

As the dependent variable of the experiment (efficiency of teams) has two treatments (total time (a) productivity (b)), the application of t-test was performed in two steps too. During this process, we calculated the variance using the following equation:

$$Sx^2 = \frac{\sum x_i - (\frac{\sum x}{n})^2}{n-1}$$  \hspace{1cm} (3), with $n = 3$

### Step 1: t-test (Total Time)

After calculating the variance of each group, we have:

$Sx^2$ (without framework) = 2,97723  \hspace{1cm}  $Sx^2$ (with framework) = 1,5325
$\alpha = 0,2$  \hspace{1cm}  $Sp = 1,501620791012165$
$|t_{0.1000, 4}| = 2,1318$  \hspace{1cm}  $t_0 = -2,498498775014366$

Then we have $|t_0| > t_{0.1000, 4}$ REJECT the null hypothesis $H_0$ with 20% of significance.

### Step 2: t-test (Total Productivity)

After calculating the variance of each group, we have:

$Sx^2$ (without framework) = 15951  \hspace{1cm}  $Sx^2$ (with framework) = 179347
$\alpha = 0,2$  \hspace{1cm}  $Sp = 130,1691207621838$
$|t_{0.01, 4}| = 4,6041$  \hspace{1cm}  $t_0 = 5,475964797075994$

Then we have $|t_0| > t_{0.01, 4}$ REJECT the null hypothesis $H_0$ with 2% of significance.

### 6 Final Remarks

SyncSmartTv can be offered to developers in form of a semi-complete source code skeleton that integrates synchronization, notification and TV controls functions. A set of tools that were developed in 4.2.3 with the purpose of a proof of concept of the framework can provide to a developer more facilities in the process of building Smart TV integrated applications. In addition, all functional requirements that were established while planning the development of this framework were implemented and used during the instantiation of the SyncSmartTv. In addition, the experiment conducted in this paper could prove statistically that the SyncSmartTv can be considered as an important tool to support the developers of applications in this field.

Regarding future work, we plan to: (a) add mechanisms of synchronization through local audio/video processing of TV content to framework; (b) explore and add adjustment mechanisms of synchronization with the purpose of minimizing the delay difference that exists among various forms of television content transmission (radio broadcasting, cable, satellite, etc); and (c) offer more notification API’s for supporting the development of social TV systems with the purpose of improving the user experience quality in Smart TV environment.

### 7 References