# Cross-Communicability: evaluating the meta-communication of cross-platform applications

Rodrigo de A. Maués and Simone Diniz Junqueira Barbosa

Informatics Department, PUC-Rio
Rua Marques de Sao Vicente, 225/410 RDC
Gavea, Rio de Janeiro, RJ, 22451-900, Brazil

{rmaues, simone}@inf.puc-rio.br

**Abstract.** Evaluating cross-platform systems is challenging due to the different constraints and capabilities of each platform. In this paper we extend the Semiotic Inspection Method (SIM), a Semiotic Engineering evaluation method, to evaluate cross-platform systems. We introduce the term cross-communicability to denote the quality of the meta-communication of the system as whole, taking into account the user traversal between the different platforms. To assess cross-communicability, we describe a novel approach to conduct the SIM, which introduces a contrastive analysis of the designer-to-user meta-communication messages of each platform, based on a semiotic framing of design changes initially proposed for End-User Development. The results from an analytical study indicate that this approach is capable of identifying and classifying several potential communication breakdowns particular to cross-platform systems, which in turn can inform the design or redesign of a cross-platform application.
**Keywords:** Cross-platform; user interface design; communicability; semiotic inspection method; semiotic engineering.

## 1 Introduction

People are increasingly using applications and services that span a wide variety of computing devices, such as PCs, smartphones, tablets and digital TVs [1, 2, 3, 4, 5]. However, developing such systems can be a challenge because it is necessary to know not only how but also when to incorporate features available in one platform into the others, since each platform may differ in its capabilities and constraints (*e.g.*, screen size and resolution, input mechanisms) [4, 5, 6, 7]. Hence, it can be difficult and costly to maintain the quality of the system across each platform.

Evaluating cross-platform systems is as challenging as designing them [1, 4]. Traditional methods for assessing quality attributes of systems need to be adapted for the cross-platform context. The evaluated quality of separate parts (*i.e.*, system versions on each platform) does not necessarily correspond to the quality of the whole cross-platform system [2, 4]. People alternately use the "same" application in different plat-

forms, frequently switching from one to the other. When traversing between system versions of a cross-platform system, a person should be able to reuse his or her knowledge of the available functions and of how tasks are performed. The usability aspects related to these traversals are introduced in [2] as inter-usability.

Recent works have investigated the development [6, 7, 8, 9] and the evaluation [2, 5, 10] of cross-platform systems by focusing mostly on their cognitive aspects, such as (inter) usability [2, 4, 10] and user experience [5]. In this paper, we shift the focus from evaluations based on cognitive theories to one based on Semiotics [11, 12]. Thus, instead of assessing the system's usability, we assess its communicability [12, 13], as described in the next section.

In this paper, we introduce the concept of cross-communicability as the system communicability across platforms. To evaluate cross-communicability, we use a semiotic framing originally proposed for End-User Development [15] to extend the application of the Semiotic Inspection Method (SIM) [12, 14], enriching the evaluation results and discussion.

This paper is organized as follows. We begin by presenting the theoretical background for this work. We then present the details of the application of the extended SIM to evaluate cross-platform systems and the concept of cross-communicability. Next, we describe our analytical study and present the results. Finally, we present our conclusions and future works.

## 2 Background

### 2.1 Semiotic Engineering Basics

Semiotic Engineering [11, 12] is a reflective and explanatory (as opposed to predictive) theory of human-computer interaction (HCI) that focuses on communicative rather than cognitive aspects of HCI analysis and design. It views the user interfaces of interactive systems as meta-communication artifacts, *i.e.*, through the user interface, designers[1] convey to the users their understanding of who the users are, what they know the users want or need to do, in which preferred ways, and why. Users then unfold and interpret this message as they interact with the system. This meta-communication message can be paraphrased as [12]:

*"Here is my understanding of who you are, what I have learned you want or need to do, in which preferred ways, and why. This is the system I have therefore designed for you, and this is the way you can or should use it in order to fulfill a range of purposes that fall within this vision." (p.25)*

The designer-to-user message is comprised of signs [11, 12, 16]. A sign is anything that stands for something else, to somebody in some respect or capacity [17]. Signs

[1] In this paper designers should be interpreted as whoever speaks for the design team of a given application.

compose one or more signification systems that arise from culturally (and, in the case of HCI, also artificially) encoded associations between content and expressions [11, 12, 15]. For example, words and images typically come from signification systems that exist in a culture outside the specific context of HCI, whereas mouse pointers belong to signification systems that are native to computer applications.

Semiotic Engineering classifies the signs in three different types [12, 16, 18]: static, dynamic and meta-linguistic. Static signs express and mean the system's state. They are motionless and persistent when no interaction is taking place (*e.g.*, icons, text areas, buttons at a given moment, menus). Dynamic signs express and mean the system behavior and emerge during interaction. They represent the transitions between the system states (*e.g.*, a save button becomes enabled after entering the name of a client in a registration form). Meta-linguistic signs refer to other interface signs and are used by the designer to explicitly communicate to users the meanings encoded in the user interface and how they can be used (*e.g.*, instructions and explanations, error and warning messages, hints). They are most commonly present in help systems, online/offline documentation, user manuals and promotional materials.

Based on this theoretical framework, the quality of a user interface is given by its communicability, which is "the system's property to convey effectively and efficiently to users its underlying design and interactive principles" [12, 13]. On the one hand, when a user can comprehend how the system works because the designer expressed himself properly through the user interface (communicability), it becomes easier to learn how to use it (usability) [11, 12]. On the other hand, when the user fails to understand the communication intended by the designer, a communication breakdown takes place that may hinder or even preclude the use of the system [11, 12]. Semiotic Engineering has two qualitative and interpretive methods to assess the communicability of a user interface [11, 12, 16]: the Communicability Evaluation Method (CEM) [12, 13], which involves user observation, and the Semiotic Inspection Method (SIM) [12, 14, 16, 18], which involves user interface inspection. While CEM focuses on the reception of the meta-communication message by the users, SIM focuses on its emission by the designer. Our paper extends SIM to cover the first part of the communication process – the emission of the message – in cross-platform systems.

### 2.2 The Semiotic Inspection Method

The Semiotic Inspection Method (SIM) [12, 14, 16, 18] is a qualitative inspection method grounded in Semiotic Engineering that allows the evaluation of the communicability of a computer system through the analysis of its static, dynamic and meta-linguistic signs. The goal is to identify communication breakdowns that may take place during the user-system interaction and to reconstruct the designers' meta-message conveyed by the user interface. Like other inspection methods (e.g. Heuristic Evaluation [19] and Cognitive Walkthrough [20]), SIM does not involve the participation of users [11]; it is the evaluator who examines the interface in search of ambiguities and inconsistencies in the signs chosen by the designer. Thus, as in any theory-based method, the more knowledge the evaluator has, the better the evaluation results will be.

Heuristic evaluation and cognitive walkthroughs focus on identifying usability issues, mostly related to the ease of use and learning of the system operations, while the strategic aspects of the system (the logic underlying the design decisions) conveyed by the user interface are not covered. Instead of assuming users always adopt a plan-based problem-solving method, semiotic engineering is more aligned with Suchman's work on situated action [21], which suggests that users are constantly reevaluating their current circumstances and adapting their "plan sketches" via communicative actions performed at/through the user interface. By inspecting the user interface language, we are able to predict some of these breakdowns, which may lead to user interaction problems. Rather than relying on a cognitive plan-based analysis, our inspection draws on Linguistics and Semiotics, which provide us with clues on how people make sense of and negotiate meaning.

SIM requires a preparation phase, in which the evaluator defines the purpose of the inspection, performs an informal inspection by navigating through the system to define the focus of the evaluation, and finally elaborates the inspection scenarios. Next, the evaluator proceeds to the execution of the inspection. To properly execute the method, the evaluator must assume a "user advocate" position. The execution of the method is carried out in five distinct steps: (1) a meta-linguistic signs inspection; (2) a static signs inspection; (3) a dynamic signs inspection; (4) a contrastive comparison of designer-to-user meta-communications identified in steps (1), (2), (3); and, finally, (5) a conclusive appreciation of the overall system communicability. In the first three steps, the evaluator inspects the signs within the scope of the evaluation and reconstructs the meta-messages being conveyed by them at each level, filling out the template of the designer-to-user meta-communication and identifying potential communicability problems at each level. Because each sign level has distinct expressive possibilities, the generated messages may not be identical. Thus, in step (4) the evaluator contrasts the meta-messages generated in the previous steps and checks for real or potential inconsistencies or ambiguities among them. Finally, in step (5), the evaluator qualitatively assesses the system communicability by unifying the meta-communication messages and then generates an evaluation report.

### 2.3 Impermeability and Computer Manipulations of Signs

Before we proceed with the extended application of SIM to cross-platform systems, it is important to briefly present two semiotic concepts that will be used in our analysis: impermeability and computer manipulations of signs.

In [15] those concepts were used to provide a semiotic framing for End-User Development (EUD). Both are related to the identity of an application, which is an important matter for both EUD and cross-platform development. While for EUD the main concern is how to empower users to customize or extend a system without losing the designed system identity, the main concern of the designer of a cross-platform system is how to design a different version for each platform while preserving the system identity across them, so as not to hinder the alternate use of the various versions. By assuming that traversing between platforms bears resemblance to traversing between customized versions of a system, we propose to apply these concepts beyond

the scope of EUD for the first time, to help in the semiotic inspection of cross-platform systems. It should be made clear that the designer did not necessarily make these modifications, but based on the differences between supposedly equivalent signs we can assume which manipulations are capable of transforming a system version into another. From the nature of the manipulation, we can identify potential cross-communication breakdowns.

The concept of impermeability [15] is related to the encapsulation of signs, *i.e.*, the sign meaning is in an atomic capsule and therefore it cannot be altered. Thus, the originally encoded meaning of impermeable signs is always preserved. Impermeable signs can be essential or accidental. Essential impermeable signs can only be used in monotonic manipulations, *i.e.*, those that do not destroy the basic meanings (identity) of the application. Accidental impermeable signs may not be changed either, but they may be subtracted from the application by means of an operation we call *pruning*. For instance, let us consider a "skip to next track" button. It is an impermeable sign and thus its meaning of skipping to the next track on a playlist should not be changed, otherwise it may confuse the user (*e.g.*, if the next track button is used to skip to the next playlist instead). If this sign is essential (*i.e.*, part of the system identity), it should also be present in any other platform. However, if it is not part of the system identity (*i.e.*, it is an accidental sign), it may be pruned in any platform.

From a computer symbol-processing point of view, two different kinds of modifications can be made to signs [15]: meaning-preserving modifications (types I, II, and III) and meaning-changing ones (types IV through VII).

*Meaning-preserving manipulations* affect only impermeable signs and preserve the application identity. Type I changes correspond to *renaming and aliasing* operations that affect only lexical components. Changing the label of a button and changing an icon appearance are examples of renaming. Renaming should be limited to lexical items that are not part of the application's identity to avoid confusing the user about its meaning. Type II manipulations involve making changes only to syntactic components, such as *reordering*: changing the layout sequence of user interface elements or the order in which operations are performed. However, this reordering cannot change the meaning expressed by the individual reordered components. Finally, a type III manipulation (also known as *pruning*) corresponds to the possibility of selecting non-essential (accidental) components to be included in or excluded from the application. Pruning must preserve not only the identity of the application, but also the impermeability of the pruned components. Thus, an important design decision is to select which of the impermeable signs can be pruned if the application identity is to be preserved.

When porting the system to a more constrained platform, the designer of cross-platform systems usually has to leave features out (type III – *pruning*) or adjust them according to the space available in the screen (type II – *reordering*). Also due to space constraints or specific platform patterns, the designer may have to change the appearance of some components (type I – *renaming and aliasing*).

Every application requires that users learn a new and unique signification system that is used in HCI [15]. Therefore, the application's identity and the notion of impermeability are important to keep the user's semiosis sufficiently tied to the designer's vision, so that productive interpretations can be motivated, and unproductive ones

discouraged [15]. This is especially important when it comes to cross-platform systems, since you may have many different versions of the same system.

*Meaning-changing manipulations*, however, can threaten the application identity, and therefore should be avoided whenever possible in order to minimize conflicts between system versions. A type IV manipulation, for example, involves using existing signs to mean something different from what they were designed to mean (*e.g.*, the same button triggers different actions on each platform). Hence, when transitioning from one version to another, users may feel frustrated because of this conflict of meanings. A type V manipulation makes changes to meanings and lexical items but not to grammatical structures. Type VI may involve reordering components or even eliminating components that change the conveyed meaning. And finally, type VII makes changes to meanings, grammatical structures and lexical items, consisting of the biggest threat to the application identity.

In our analytical study, we give examples of some of the manipulations we found and discuss about their potential consequences concerning cross-platform systems.

## 3 Using SIM to Evaluate Cross-Communicability

When evaluating each version of a cross-platform system separately, each one may have high communicability, *i.e.*, it may efficiently and effectively convey to the users its underlying design intent and interactive principles. However, when traversing between platforms, the user carry understandings from one version that may not be applicable to another, creating conflicts and communication breakdowns. Besides, the meta-communication message of each version should also convey the composition of the cross-platform system, *i.e.*, how functionalities are organized in and across platforms [5]. The quality of the meta-communication conveyed by the cross-platform system as a whole, including all versions, is what we call *cross-communicability*.

The traditional SIM alone is not suitable for evaluating cross-communicability, since we must also systematically contrast the meta-communication messages across system versions. Hence, after the preparation phase required to the traditional SIM described previously, we divide the execution of the method applied in two phases: vertical (within-platform) analysis and horizontal (between-platform) analysis.

### 3.1 Vertical Analysis

The first phase, called vertical analysis, is designed to evaluate the communicability of the system in each platform individually. Hence, it consists of conducting the same five steps of the traditional SIM, with one difference: the evaluator should also highlight the signs that denote any compositional aspects of the cross-platform system (*e.g.*, a sign that acknowledges the existence of another system version).

Because the inspections must be conducted individually, different evaluators may conduct the inspection of each version. There are three main benefits of this approach: it distributes and reduces the cost of the inspection; it avoids contaminating the inspection of one version with another; and it allows designating each system version to

the most appropriate evaluator, a professional with solid technical HCI knowledge in that platform.

### 3.2 Horizontal Analysis

As stated before, when evaluating a cross-platform system it is important to contrast the meta-communication messages of each system version. Thus, after completing the communicability assessment of each version, the next phase consists of performing a horizontal analysis, which is divided in three steps: (1) identification and collation of sign manipulations between the evaluated platforms; (2) an analysis of the impact of each manipulation on the meta-communication of each version; and finally (3) a conclusive appreciation of the quality of the overall system cross-communicability.

The signs collated in the vertical analysis may undergo modifications across the evaluated system versions due to the constraints, advantages or patterns of each platform. Thus, the evaluator should (1) identify the meaning-preserving and meaning-changing manipulations in each pairwise combination of the evaluated platforms; (2) examine the manipulations collated and categorized in the previous step to assess how they could negatively affect the horizontal meta-communication messages by intensifying already identified vertical communication breakdowns (e.g., causing ambiguities) or even by creating new ones; and (3) qualitatively assess the system cross-communicability by unifying the meta-communication message obtained in each previous step, judging the costs and benefits of the identified manipulations made between platforms, and then he generates an evaluation report.

## 4 Analytical Study

For the application of SIM on a cross-platform system, we chose an Internet radio and social networking website named 8tracks,[2] which revolves around the concept of streaming user-curated playlists composed of at least 8 tracks. Besides being a popular cross-platform system, 8tracks was chosen because it has different versions available on different devices (desktop and mobile) and also has different versions available on the same device (mobile native apps[3] and mobile web app), therefore covering a broad range of combinations to analyze in our study.

### 4.1 Preparation Phase

The goal of this analysis is to identify different communication breakdowns that may occur individually on each platform version of a cross-platform system and their relation to other breakdowns that arise when these versions are put together. We defined the context of use for the evaluation through the following scenario:

[2] http://8tracks.com/

[3] 8tracks has two different mobile native apps available, an iOS and an Android version, but only the Android version was evaluated in our study.

*"Rodrigo will throw a party and needs to set a good playlist, but has no idea how to start. Gabriel, Rodrigo's friend, suggested he use 8tracks system, where he can discover and listen to several different mixes. Rodrigo had already created an account in the past, but he has never really used the system. He thinks this might be the perfect opportunity to give it a try. He then searches 8tracks for mixes with the latest hits. He saves his favorite mixes, to be able to play them later at the party, if he so wants. He also creates his own mix only with the favorite songs from the mixes he has played. So as not to forget which ones they are, he will use another feature that allows saving only individual favorite tracks to be bought later, instead of saving full mixes."*

Although it was observed by a previous informal inspection that not all the functionalities in the scenario would be available in all the evaluated system versions, we decided to let it be this way to evaluate how each version conveys information about the system composition as well. Next, we briefly present the main results from the vertical and horizontal analyses.

### 4.2 Vertical Analysis

A single evaluator conducted the study. Due to space constraints, in this paper, we do not describe the whole vertical analysis of the 8tracks versions, which consists of a typical application of SIM. Instead we present synthesized versions of the meta-message, focusing on some communication issues found when analyzing the system communicability on each platform. Although the vertical analysis was conducted separately for each platform, some references will be made to similar characteristics across the versions to avoid repetition and also due to space constraints.

**Desktop Web App.** Besides the hints and tips throughout the system, there are many internal pages (*e.g.*, about, licensing, FAQ) and a blog available where some meta-linguistic signs can be found. This documentation mainly explains the system purpose and its underlying concepts, instead of how to use it. Therefore, it is implied that the system is in general thought to be easy to use and to learn. The available explanations describe many aspects of the legal rules that the service must follow to allow the free streaming of music. These rules limit some functionalities of the system, but expecting that users will probably not want to read the documentation before using the system, the main rules are only briefly explained through dynamic messages to let users become aware of them by trial and error when interacting with the system.

Some aspects of trial and error, however, can be frustrating due to the lack of appropriate communication. When attempting to skip to a track one too many times, for instance, the following message is shown: “Apologies for the inconvenience, but our music license requires us to limit the number of tracks you may skip each hour.” Since there is no static or dynamic sign indicating how many tracks you have already skipped or may still skip, or how long it has been since you exceeded the number of “skips,” and also because the skip button is always enabled, the user may often try to skip a track, fail, and then receive the infamous message over and over again.

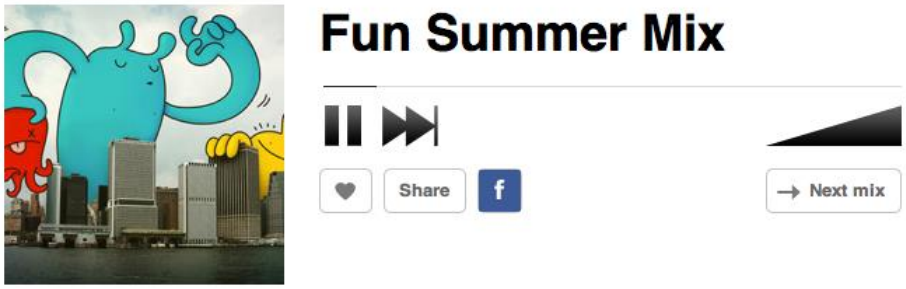


**Fig. 1.** Part of the desktop web version's mix page.

The meta-communication message derived from the documentation and help content tells us that the service is designed for music-lovers that want to create, share and discover new mixes in a radio-style way, *i.e.*, like a non-interactive music player. Static signs on the mix page (Figure 1) appropriately convey this radio-style music discovery experience depicted in the meta-linguistic signs: the music progress is represented by a very thin and discreet line, indicating that it is not so important and that users should not worry about it, just like on the radio; there is no bar over this thin line to control the music progress; and the only control buttons available are a button to play/pause the music, a button to skip to the next track or to the next mix.

Regarding the search for mixes, a communication breakdown was found. Once a search has been done, the user may sort the results by hot, new or popular. Although it is easy to infer the outcome of sorting by newest mix, the same cannot be said about the other two criteria, and there is no explanation on the system about the difference between popular and hot mixes.

Meta-linguistic signs on some of the internal web pages of the system and on the 8tracks blog almost exclusively express the information regarding the system composition. References are made to both the mobile web and the mobile native versions. There is also a static sign at the bottom of each system page, a button, referring to the mobile native version.

**Mobile Web App**. Besides the fact that the meta-linguistic signs available on documentation of the desktop web version discussed previously are also available to any other system version, in the mobile web app there is a direct link to the desktop web app's "FAQ" and "About" pages, as if they were its own. Because of this and the fact that very little information about the mobile web version can be found in the system documentation, it can be assumed that this version has the same features of the desktop version. However, during the analysis of the static signs it became clear that not all the functions depicted in the meta-linguistic signs are available in this version. Hence, if a user consults the documentation he may be misled and frustrated after searching the whole app for a function that does not even exist in it.

Although it is harder to present hints on a mobile web system because it is not possible to hover over user interface components, in general there are really few messages and instructions in this platform, possibly meaning that the designer believes that this interface is also fairly easy to use and no further explanation is necessary.

In addition to the same meta-linguistic signs available at the desktop version, in this version two static signs can be found on the bottom of the dashboard page referring to both the mobile native and the desktop web versions.

Another communication breakdown was found on the system search mechanism. Although there is a tip on the kind of text the user should type to perform a search, in the results page there is no indication of which criteria were used to search the mixes. Hence, the user does not know whether the text he typed was matched with the mixes tags, with the mixes names, with another criterion or with all of them. Another potential breakdown found on the results page is the lack of static or meta-linguistic signs evidencing what happens when the user selects one out of the three options (most recent, hot this week or popular this month) on the search results page. Even with the aid of the dynamic signs, there is no clear evidence as to whether you are sorting or filtering the search results by one of these criteria.

Users can also directly search mixes by selecting tags in the dashboard. Although the static signs indicate that you can search only by one tag at a time, it is actually also possible to search by a combination of tags if you perform a long press on each desired tag at a time, and then you press a go button that will appear next to the tags (Figure 2). However, since this is not explained to users, it is a communicability breakdown, and users may never perceive this search possibility even if they intensely interact with the system through trial and error.

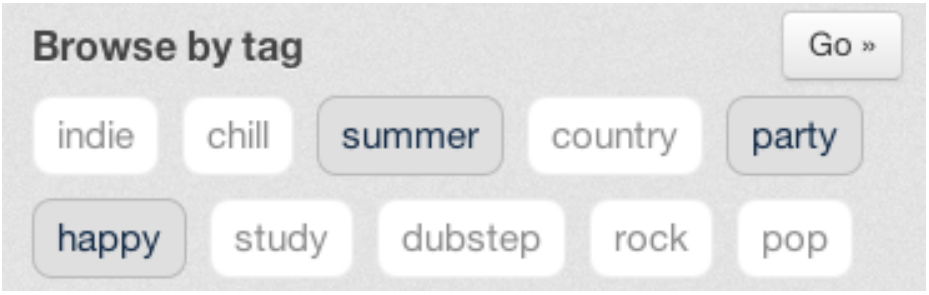


**Fig. 2.** Example of the mobile web version of a search where the user combined multiple tags as the search criteria.

**Mobile Native App.** Although there is no static sign referring to the other versions, this version contains an interesting new meta-linguistic sign regarding the system composition on its own internal FAQ: it clearly states that, if you want to create a mix or preview the songs you have added to your favorites, you need to use the desktop version. Hence, this sign not only recognizes the existence of another version but it also indicates the system distribution of functionalities, revealing more clearly the redundancy across these system versions. Nevertheless, there is no reference about the mobile web version on this platform.

There are additional meta-linguistic signs available on the app's Google Play page and also on an Update Log page internal to the app, clearly explaining the purpose, functionalities and limitations of this version. Both pages also display the changes

made to each release of the app, which is important because it helps the user to interpret new features or modifications made to previously known signs.

The static signs indicate that the user can filter the mix search results by tags, but since this label is also next to the hot, new and popular buttons, the user may also wrongly infer that these buttons can filter the results (Figure 3). However, when inspecting the dynamic signs we notice that the result count is not affected; therefore these buttons only sort the results. Still, a distracted user may never notice this dynamic sign, since it is quite subtle.

This version also introduces a new sign: a hidden left-hand menu, used to hide navigation options until someone taps a button to expose them (Figure 3). This off-canvas pattern is very common and useful in mobile applications, since it avoids occupying the small screen of the device all the time. However, the way this menu is accessed in this platform introduces a temporary failure [12]: it uses the sign of a home button, which naturally would make the user infer that he is going to a home or dashboard page of the system, which does not exist in this version. Some other problems previously discussed on the inspection of the desktop version were also found on this version, such as the hot/popular ambiguity.

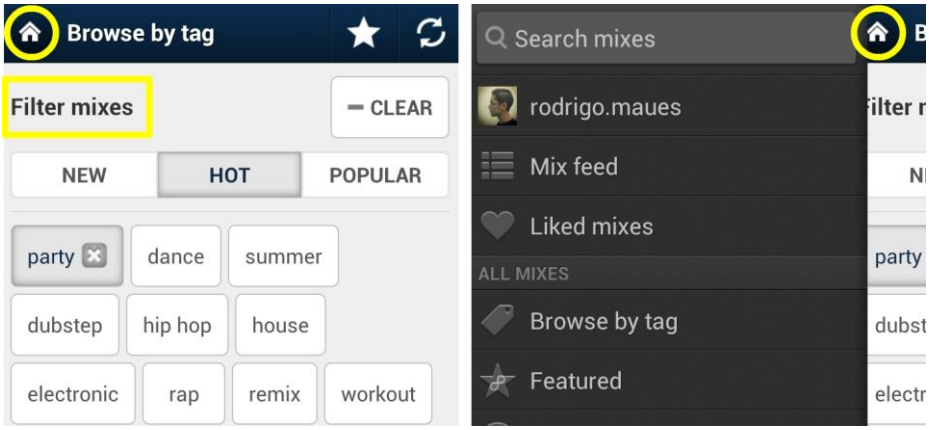


**Fig. 3.** Search results filtering page before (on the left) and after (on the right) pressing the home button, which actually triggers the hidden left-hand menu to appear on the screen. The highlighted signs are responsible for different vertical and horizontal breakdowns.

### 4.3 Horizontal Analysis

Having concluded the vertical analysis, we conducted the horizontal analysis to assess 8tracks's cross-communicability. First, we contrasted each evaluated version with the two others, resulting in six tables containing the modifications that could have been made at the user interface signs in going from one version to the other, classified by the nature of the manipulation. Manipulations from type I to V were identified, but most of them were meaning-preserving ones. Due to space constraints, only a sample

of them is presented, classified by type to discuss the main communication breakdowns found across versions and to illustrate the benefits of this analysis.

It is worth mentioning that, in a cross-platform system such as 8tracks, *i.e.*, a system whose functionality and content is available in multiple devices, many of the unchanged characteristics found across the platforms may indicate the identity of the system and some of its impermeable signs. For instance, the control buttons on the mix page, the buttons for liking the mix and for sharing the mix (Figure 1) are present with their meanings unchanged in all the platforms. This indicates that they probably are some of the essential impermeable signs and part of the system's identity. However, these relations are still not so clear.

**Type I Manipulations**. Only a few type I (*renaming*) manipulations were found: the label "Created" on the mobile versions is written "Published" on the desktop web version (Figure 4); there is a slight difference between the desktop and the mobile versions of the mix feed icons (Figure 5); and the label "recent" in the mobile web version is written "new" in the other versions (Figure 6). Although they may not lead the user to a failure, those unnecessary inconsistencies should be avoided.

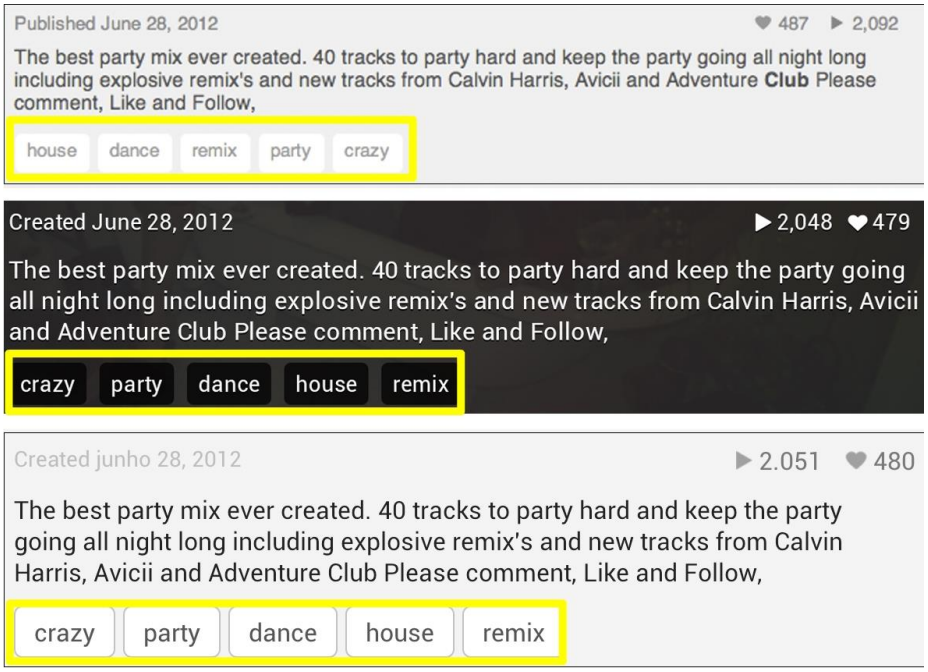


**Fig. 4.** From top to bottom, the mix description appearance on the desktop web, mobile web and mobile native versions, respectively. The highlighted tags are not clickable only in the mobile web version.

**Fig. 5.** Example of type I manipulation and unnecessary inconsistency: difference in the appearance of the mix feed icon used in the mobile versions (on the left) and in the desktop web version (on the right).

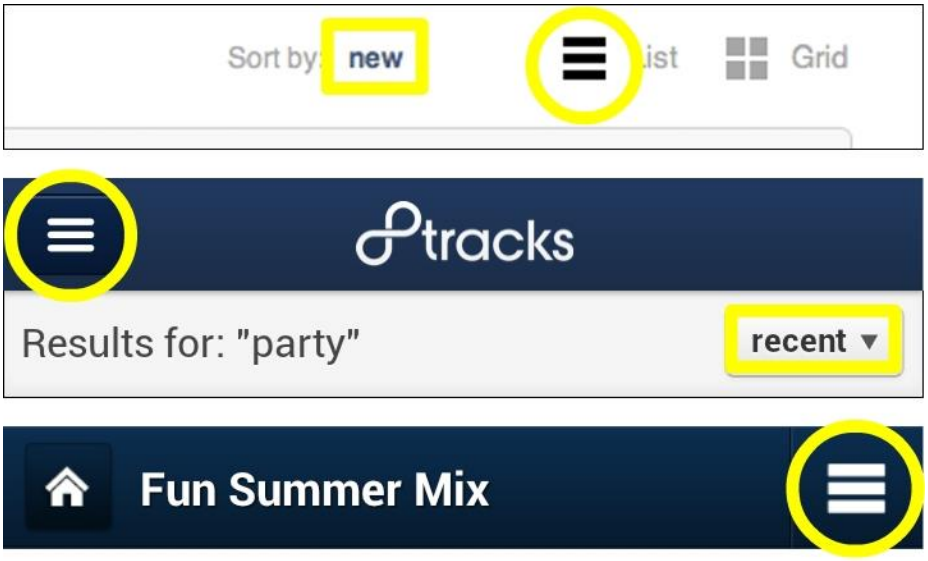


**Fig. 6.** From top to bottom, part of the search results page on the desktop and mobile web versions and part of a mix page on the mobile native version, respectively. The highlighted elements reveal type I (rectangle) and type V manipulations (circle).

**Type II Manipulations.** Type II manipulations (*reordering*) are most common and inevitable when it comes to cross-platform systems working in different devices. This happens because the display size and resolution of each device force designers to reorganize the layout to fit all the interface elements, or even to separate elements into two or more screens to avoid pruning some essential impermeable signs. Hence, it is inevitable to break the interface layout consistency to fit everything.

However, it is possible to find on 8tracks some type II manipulations that resulted in unnecessary inconsistencies. For instance, in the mix description (Figure 4), the labels for the mix like and play counts are displayed in one order in the desktop version and in the reverse order in the mobile versions, even though there are no space constraints. Although the latter example may not constitute a serious communication breakdown, it may negatively affect some cognitive aspects. For instance, a person who is already used to the order of the labels for mix like and play counts in a plat-

form will have to learn the order of the elements again in the other platforms. If instead of labels the interface elements were buttons, for example, the same reordering would represent a bigger issue, since a distracted person might press the wrong button because the order (that the person was already used to) has changed.

Another reordering problem found was that the same groups of mixes are arranged in the desktop web in a different order from the mobile web. Since the order of elements may also convey a relation of importance between them, there is an ambiguity on what the designer thought was more important to the user: the desktop web prioritizes the users' mixes (*i.e.*, their own mixes, liked mixes and mix feeds from whom he follows) while the mobile web prioritizes general featured and hot mixes.

**Type III Manipulations.** After the vertical analysis, the evaluator already has a clear vision of which features and components are present or not in each version, which makes it easy to identify type III (*pruning*) manipulations. Based on the collated type III manipulations, it became evident that the versions are complementary, *i.e.*, although they share a set of data and functions, some of them provide access to data or functions that are not available in the others.

Some of the observed pruning operations from the desktop to the mobile versions may be justified by either reduced platform capabilities or little utility. For instance, the functionality of creating mixes is not available in the mobile versions, maybe because of either platform limitations (slower internet connection when using 3G to upload each track and limited processing power in comparison to a desktop); or little utility, since the user will be using the system mostly on the go and therefore will not be interested or even capable of spending too much time looking at the device screen. Thus, the 8track's designer may have intended that the users should only listen in their mobile devices to what had been produced on their desktop computers.

In the desktop version, since some mix arts may contain inappropriate content to be shown during the service use in a public or work environment, the user may flag some arts as Not Safe For Work (NSFW) and configure them not to be shown. This feature is not available on the mobile versions, probably because Smartphones are highly personal; therefore there is little utility in avoiding displaying any type of inappropriate content, since only the user would see it on his device.

Type III manipulations can indicate some interesting communication breakdowns across platforms, because the lack of some features between certain versions may represent an ambiguity for the user. In our study, one example is the feature of adding a track to the favorites. To do so, the user must press the track's star button placed either next to where the current track is being shown or on the tracklist, which shows only the mix tracks already played. The vertical analysis of the mobile web version revealed that, on the tracklist page (Figure 7), there is only a button to buy each track and you can neither see the tracks you have added to your favorites nor add them; therefore you can only add a track to your favorite (or know that it has already been added to your favorite tracks) while it is still playing. When contrasted to the desktop and the mobile native versions, this issue is amplified, because the user in the desktop version can buy the track or add it to the favorites using buttons on the tracklist, and in the mobile native the user can only add but not buy the track in the tracklist (Figure 7). Therefore a person used to these versions may get frustrated when trying to add a

track to the favorites on the mobile web version after he has listened to the whole song, only to discover that he cannot do it. Since in 8tracks the user cannot go back to a previous track and the tracklist is shuffled each time the user listens again to a mix from the beginning, he would be forced to go to another version to see the full tracklist so far and then add the track to his favorites. Since pruning the button to buy a track from the tracklist does not represent a problem, but removing the star button (to add a track to the favorites) does, it is possible to infer that the tracklist and the star button together make up an impermeable sign, and therefore should not be separated.


**Fig. 7.** From top to bottom, a part of the tracklist page on the desktop web, mobile web and mobile native versions, respectively. The highlighted signs are responsible for a horizontal communication issue.

Still regarding pruning manipulations, removing the dashboard page on the mobile native version per se would not lead to a communication problem across platforms. However, confusion arises because there is a similar home button that shows the left-hand menu (Figure 3) instead of leading the user to a dashboard page (Type IV manipulation – changing the meaning of a sign), which happens in the other platforms.

Pruning manipulations can also be observed on the 8tracks meta-linguistic signs. Adding a new FAQ to the mobile native version results in interesting observations. First, its benefit to the mobile native version is clear, once it avoids the same communication breakdown as the mobile web version regarding the confusion about which functionalities are available or not. Second, the mobile native's FAQ contains a piece of information about the purpose of the favorite tracks feature (*i.e.*, save tracks to buy them later), which is missing from the meta-linguistic signs in the desktop version. Hence, if a user learns about this feature from the native app, he may understand it better than another user who inferred its behavior by its name on the desktop version.

A similar situation can be observed regarding the feature of sorting search results by hot, new or popular. As mentioned before, a common breakdown in the desktop and mobile native versions is the lack of clarity on the difference between the hot and popular criteria. However, if a person has used this feature on the mobile web version, he can eliminate this doubt, since the labels there are more detailed: "hot this week" and "popular this month". In the mobile web version the breakdown regards not knowing whether a sorting or a filtering operation was applied to the results after

selecting one of the criteria. This problem can be either avoided or transformed into another issue, depending on which version was used prior to the mobile web one. If the user has already used the desktop version, he will know the results are sorted by one of those criteria. If the user has had previous experience only with the mobile native, he may mistakenly assume a similar behavior, i.e., that the results were filtered (Figure 3). Finally, contrasting the desktop and mobile versions altogether may lead to another breakdown, since the user may no longer be sure whether each version is behaving differently or similarly, and in case they are indeed doing the same operation, they might not know whether it is sorting or filtering the results.

The latter breakdown is probably due to one of two reasons: an unfortunate type I manipulation, because as stated before you should not change a label if you are changing an impermeable sign, and therefore you mislead the user into thinking that something means something else (sort *versus* filter); or a type II manipulation that went wrong, *i.e.*, the label in the mobile native app was referring only to the tags but was misplaced next to the sorting options as well.

**Type IV Manipulations.** Regarding meaning-changing manipulations, one type IV manipulation (only the meaning of a sign is changed) is responsible for a potential issue when the user selects a mix to go to its page: in the desktop and mobile web apps, when entering the mix page, the mix does not start playing right away (the user needs to press a play button), but the same is not true in the mobile native app. The fact that the mix will start playing once selected in the latter version may frustrate a person used to the other meaning of selecting a mix on the web versions of the system, and vice-versa.

**Type V Manipulations.** On the mobile web version, a button (Figure 6) on the header of the search results page (which triggers the interface to go to the home or dashboard page) can cause a breakdown because it does not resemble a home button and therefore its meaning cannot be easily interpreted or even learned by the user. When contrasting this version with the mobile native and desktop web versions, this button can cause additional breakdowns, because a very similar sign is used in those versions (Figure 6) with different meanings. Thus, we have the occurrence of a type V manipulation (meanings and lexical items are changed).

In the mobile native the meaning of this sign is to display the mix tracklist, while in the desktop web the meaning is to display the search results in a list (instead of in a grid). The same sign did not create a vertical communication problem in the desktop web and mobile native versions; it created a vertical communication problem in the mobile web version; and it also created a horizontal communication breakdown, since the user may get confused about which action is associated to the button on each platform, leading to temporary failures.

Another type V manipulation is responsible not for a new communication issue, but for amplifying a breakdown identified during the mobile version's vertical analysis: the tags on the mix description (Figure 4) resemble buttons, but they are not actually clickable. Since in the desktop web and mobile native versions the same tag buttons (although different in color) are clickable (triggering a search by a tag when clicked), this manipulation may lead the user to frustration, since he is likely try to

click on the tags on the mobile web version if he is already used with the other system versions.

## 5 Conclusion and Future Work

This paper described a first effort to apply Semiotic Engineering and its methods to the evaluation of the quality of cross-platform systems, which we defined as cross-communicability. We presented an analytical study using a new approach to SIM to assess how well the designer-to-user meta-communication message gets across to users through each system version separately and also through the cross-platform system as a whole, taking into account the possible ambiguities and communicative breakdowns when traversing between versions.

The proposed evaluation of the cross-communicability of a system consists of two analyses: a vertical analysis, which is mostly an application of the SIM to each system version; and a horizontal analysis, in which the signs and meta-messages from the previous analyses are contrasted and the differences are classified according to the type of sign manipulations that would be necessary to transform one version into another. This classification is based on a semiotic framing originally proposed for EUD.

We could identify through our analytical study some interesting facts when contrasting the meta-messages of the different system versions: ambiguities and new communication breakdowns may arise in one or more system versions; individual breakdowns may be transmitted from one version to another; individual breakdowns in one version may be amplified by individual characteristics or issues of other versions; and also some individual breakdowns in one version may be overcome due to signs present on the other versions.

Thus, we believe that by looking at the results of a SIM evaluation informed by our extended approach, the designer of cross-platform systems may become more aware of several communicative breakdowns regarding composition, continuity (*i.e.*, how tasks state and actions are migrated across platforms) and consistency aspects, which can help them to better design or redesign each system version. The proposed sign organization by manipulations may also help to improve existing cross-platform development processes, as in the graceful degradation [6] method, for instance. Results from a horizontal analysis could serve as a basis to refine the set of transformational rules applied during the graceful degradation approach, as well as to validate the priority ordering between the rules in the method.

As future work, we plan to triangulate the results obtained with endogenous and exogenous sources. We also plan to conduct a study comparing the extended SIM with more traditional inspection approaches, such as heuristic evaluation and cognitive walkthrough. Also we intend to investigate more closely the relation between the system's identity and the quality of a cross-platform system, as well as how we can properly define which signs are impermeable and essential to one or more platforms, and to the system as a whole. We also plan to formulate a set of heuristics as a result of consistently applying SIM to multiple cross-platform applications. And finally, we plan to extend CEM to cross-platform systems as well.

## References


1. Antila, V., Lui, A. Challenges in designing inter-usable systems. Proc. INTERACT 2011, Vol. Part I. Springer-Verlag (2011), 396-403.
2. Denis, C., Karsenty, L. Inter-usability of multi-device systems - a conceptual framework. Seffah, A., Javahery, H. (eds.) Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces. Wiley & Sons (2004).
3. Paternò, F. Designing multi-device user interfaces: how to adapt to the changing device. Proc. INTERACT 2007, Vol. Part II. Springer-Verlag (2007), 702-703.
4. Seffah, A., Forbrig, P., Javahery, H. Multi-devices "Multiple" user interfaces: development models and research opportunities. Journal of Systems and Software (2004), Volume 73, Issue 2, 287-300.
5. Wäljas, M., Segerståhl, K., Väänänen-Vainio-Mattila, K., Oinas-Kukkonen, H. Cross-platform service user experience: a field study and an initial framework. Proc. MobileHCI 2010. ACM (2010), 219-228.
6. Florins, M., Vanderdonckt, J. Graceful degradation of user interfaces as a design method for multiplatform systems, Proc. IUI 2004. ACM (2004).
7. Lin, J., Landay, J. A. 2008. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. Proc. CHI 2008. ACM (2008), 1313-1322.
8. Ghiani, G., Paternò, F., Santoro, C. On-demand cross-device interface components migration. Proc. MobileHCI 2010. ACM (2010), 299-308.
9. Paternò, F., Zichittella, G. Desktop-to-mobile web adaptation through customizable two-dimensional semantic redesign. Proc. HCSE 2010. Springer-Verlag (2010), 79-94.
10. Serna, A., Calvary, G., Scapin, D. L. How assessing plasticity design choices can improve UI quality: a case study. Proc. EICS 2010. ACM (2010), 29-34.
11. Barbosa, S.D.J.; Silva, B.S. Interação Humano-Computador. Campus-Elsevier (2010).
12. de Souza, C. S. The Semiotic Engineering of Human-Computer Interaction. The MIT Press (2005).
13. Prates, R. O., de Souza, C. S., Barbosa, S. D. J. Methods and tools: a method for evaluating the communicability of user interfaces. ACM Interactions (2000), 7, 1, 31–38.
14. de Souza, C. S., Leitão, C. F., Prates, R. O., da Silva, E. J. The Semiotic Inspection Method. Proc. IHC 2006. SBC (2006), 1, 148-157.
15. de Souza, C. S.; Barbosa, S. D. J. A semiotic framing for end-user development. Lieberman, H., Paternò, F., Wulf, V. (Org.) End User Development: people to flexibly employ advanced information and communication technology. Springer (2006), 401-426.
16. de Souza, C. S.; Leitão, C. F. Semiotic Engineering Methods for Scientific Research in HCI. Morgan & Claypool (2009).
17. Peirce, C. S. The essential Peirce (Vols. I and II). Houser, N., Kloesel, C. (eds.). Indiana University Press (1992).
18. de Souza, C. S.; Leitão, C. F.; Prates, R. O.; Bim, S.A.; da Silva, E.J. Can inspection methods generate valid new knowledge in HCI? The case of semiotic inspection. International Journal of Human-Computer Studies (2010), 68 (1-2), 22-40.
19. Nielsen, J. Heuristic Evaluation. Mack, R. & Nielsen, J. (eds.) Usability Inspection Methods. John Wiley & Sons (1994), 25-62.
20. Wharton, C., Rieman, J., Lewis, C., Polson, P. The Cognitive Walkthrough Method: a Practitioner's Guide (1994).
21. Suchman, L. Plans and situated actions: The Problem of Human-Machine Communication. Cambridge University Press (1987).