Reflections on the Cross-Platform Semiotic Inspection Method

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 can be quite challenging. Unfortunately, despite the increasing number of such systems and therefore growing need for evaluation methods, little work has been done on the matter. We have extended the Semiotic Inspection Method (SIM), a Semiotic Engineering evaluation method, to evaluate cross-platform systems, producing the CP-SIM variant. However, despite its support in identifying and classifying several potential issues particular to cross-platform systems, the cross-platform aspects of the method (called 'horizontal analysis') were only briefly illustrated by an analytical study in the original work. This paper provides deeper reflection and a more detailed account of the horizontal analysis in order to support evaluators in using the method. It also situates CP-SIM among related work on cross-platform system evaluation.

Keywords: Cross-platform, user interface design, communicability, semiotic inspection method, semiotic engineering.

1 Introduction

Users increasingly expect to have access to the same applications and services with a multitude of computing devices (e.g., PCs, smartphones, tablets, digital TVs), which differ greatly in their capabilities and constraints (e.g., screen size and resolution, input mechanisms, etc.) [1, 2, 3, 4, 5]. Traditional evaluation methods are not entirely adequate to assess the quality of these cross-platform systems, since 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]. However, despite the increasing number of such systems and therefore growing need for specific evaluation methods [1], little work has been done on the matter.

This paper advances a line of research that evaluates cross-platform systems based on Semiotics rather than cognitive theories [2, 5, 6]. In a previous work [7], we extended the Semiotic Inspection Method (SIM) [8] in order to provide a systematic way to evaluate cross-platform systems, resulting in the Cross-Platform Semiotic Inspection Method (CP-SIM). However, we only briefly illustrated the cross-platform aspects of CP-SIM (called 'horizontal analysis') by an analytical study. This paper

M. Kurosu (Ed.): Human-Computer Interaction, Part I, HCII 2014, LNCS 8510, pp. 533–544, 2014.

provides deeper reflection on and a more detailed account of the horizontal analysis in order to support evaluators using the method. It also situates CP-SIM among related work on cross-platform system evaluation.

This paper is organized as follows. We begin by presenting the theoretical background and related work. We then take a deeper look at the horizontal analysis of CP-SIM, discussing further about each sign manipulation type and about other key concepts for CP-SIM, relating them with concepts found on cognitive theories literature. Finally, we present our conclusions and future work.

2 Theoretical Background

This section presents some basic semiotic engineering concepts, necessary to understand the Semiotic Inspection Method, also described here.

2.1 Semiotic Engineering Basics

Semiotic Engineering [9] 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, designers1 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 [9]:

"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 [9, 10]. A sign is anything that stands for something else, to somebody in some respect or capacity [11]. Signs compose one or more signification systems that arise from culturally (and, in the case of HCI, also artificially) encoded associations between content and expressions [9, 10, 12]. 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 [9]: static, dynamic and meta-linguistic. Static signs express and mean the system's state (*e.g.*, icons, text areas, buttons at a given moment, menus). Dynamic signs express and mean the system behaviour and emerge during interaction (*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

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

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 and tooltips).

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" [9]. 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) [9, 10]. 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 [10].

2.2 The Semiotic Inspection Method

The Semiotic Inspection Method (SIM) [8] 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 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.

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 execute the method properly, 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 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. In step (4) the evaluator contrasts the meta-messages generated in the previous steps and checks for potential ambiguities among them. Finally, in step (5), the evaluator qualitatively assesses the system communicability by unifying the meta-communication messages and then generates a report.

3 Related Work

Despite the proliferation of cross-platform systems, when it comes to evaluating such systems, it is not yet very clear which evaluation techniques should be used [1]. A great body of work has been done over the last years regarding the design and development of such systems [3, 4, 13, 14, 15, 16]. However, a rare few have focused on their evaluation, which is one of the main needs of practitioners [1].

Öquist and coauthors discuss a method to assess usability of different interfaces by taking into account the different environments or contexts where different devices and interfaces are used, and by identifying some environmental variables that affect the usability of different devices and interfaces in those contexts of use [17]. However, this evaluation is not sufficient to guarantee the quality of cross-platform systems, since, as argued by Denis and Karsenty, the evaluated quality of each separate platform does not necessarily correspond to the quality of the whole cross-platform system [2]. People alternately use the "same" application in different platforms, 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 to perform tasks. Based on that, Denis and Karsenty introduced the concept of inter-usability to designate "the ease with which users can reuse their knowledge and skills for a given functionality when switching to other devices" [2]. They also introduced a conceptual framework for achieving inter-usability, which proposes design principles addressing inter-device consistency[2], transparency and adaptability. They focus on knowledge and task continuity, and on how these can be better supported through design.

Huang and Strawderman [6] also acknowledge that the knowledge gained from the previous platform may greatly affect users' performance in the following platforms, but they believe that a lack of theoretical support weakened the generalization of the approaches proposed in [2]. Hence, they proposed the Usability Paradigm for Multiple Device System (UPMDS), a conceptual framework that relies on the area of transfer of learning [18], which embraces many theoretical and empirical studies rooted in behavioral and cognitive psychology. In their framework, usability is composed of two attributes: transferability and user perception, with transferability further divided into effectiveness, efficiency and user perception further divided into satisfaction and attractiveness. They define transferability as "the ease with which users switch between using different interfaces". Their work is still in its initial stages, and more studies are needed to empirically validate the framework.

Instead of focusing on the usability as the aforementioned studies, Wäljas et al. investigated the key elements that characterize the user experience when users exploit web-based applications through different computing platforms [5]. They conducted their analysis focusing on three key themes: composition, continuity and consistency. Based on their findings, they proposed an initial conceptual framework of cross-platform user experience, in which the central elements include: fit for cross-contextual activities; flow of interactions and content; and perceived service coherence.

Maués and Barbosa [7] have shifted the focus from cognitive theories to semiotic engineering [9] and provided a systematic way to evaluate cross-platform systems: the cross-platform semiotic inspection method (CP-SIM). Having a method specific for evaluating cross-platform system, instead of only conceptual frameworks, meets the practical needs of practitioners. We discuss CP-SIM in detail next.

[2] In this paper we follow a definition of consistency similar to the one adopted in [2]: striving for uniformity regarding the system's appearance, presentation and offered functionalities.

4 CP-SIM

When evaluating each platform of a cross-platform system separately, each one may have high communicability. However, when traversing between platforms, the user brings his or her understanding of one version that may not be applicable to another, creating or enhancing conflicts and communication breakdowns. Hence, the designer-to-user messages within a cross-platform system should not be conflicting; instead, they should complement each other.

With that in mind, we introduced the quality attribute named *cross-communicability* [7]: the cross-platform system property of each of its platforms being able to convey effectively and efficiently to users not only its individual underlying design and interactive principles, but those of the cross-platform system as a whole. Just like high communicability consequently leads to a better usability [9], it is expected that high cross-communicability will result in a better inter-usability.

To assess the cross-communicability of a cross-platform system, we proposed CP-SIM [7], an extension of SIM [8]. After the SIM's traditional preparation, the execution of the method is involves two phases: vertical (within-platform) analysis and horizontal (between-platform) analysis.

We designed the vertical analysis 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 version of the system).

After completing the vertical analysis, the horizontal analysis consists of contrasting the meta-communication messages of each system version in order to assess the system's cross-communicability. This analysis is based on a semiotic framing of design changes initially proposed for End-User Development (EUD) [12], where the differences between the system versions are related to possible manipulations made to signs on each interface. The horizontal analysis is composed of three steps: (1) to identify the sign manipulations in each pairwise combination of the evaluated platforms; (2) to 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) to 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. After going through the three analysis steps, the evaluator generates an evaluation report.

5 Reflections on the Horizontal Analysis

There already is a good body of work explaining and exemplifying how to properly apply the traditional SIM [9, 10], and therefore it is safe to assume that it would not be difficult to conduct the vertical analysis. However, the same cannot be said about

the horizontal analysis, which is a significant part of the proposed method. As mentioned before, the horizontal analysis relies on concepts taken from [12], namely, impermeability, semiotic dimensions and computer manipulation of signs. However, in [7], we could barely mention the concept of semiotic dimensions and we could only briefly discuss about the manipulation types that were found in their analytical study. In order to support evaluators to better identify and analyze the manipulations of signs, in this paper we discuss the aforementioned concepts in more details later in this section, and we contrast them with concepts found on related work to provide a better understanding and to avoid ambiguity.

From the nature of the manipulation, we can identify potential cross-communication breakdowns [7]. Regardless whether the signs were manipulated or conserved across platforms, the evaluator should intentionally explore the possibility of assigning plausible contradictory or ambiguous meanings to the signs that constitute the messages in the evaluated platforms. We derived some questions from our previous analytical study [7] that can serve as scaffolds for the evaluators' analysis:

1. Would the user plausibly be able to interpret this sign differently in other platform? How? Why?
2. Would the misinterpretation of this sign propagate and affect the interpretation of the same or other signs in other platforms? How? Why?
3. Would the user plausibly be able to interpret this sign consistently in this particular platform regardless of the amount and order of platforms that he previously interacted with? Why?

These questions are not the only ones that the evaluator can or should ask, but they provide useful guidance and input for conducting a productive horizontal analysis, which is especially useful for less experienced evaluators. Next we discuss in detail and in the context of cross-platform systems the aforementioned concepts of impermeability, semiotic dimensions and computer manipulation of signs.

5.1 Impermeability of Signs

The concept of impermeability [12] is related to the encapsulation of signs, i.e., when 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. For instance, essential impermeable signs cannot be removed from the application. Accidental impermeable signs may not be changed either, but they may be subtracted from the application by means of a manipulation called *pruning* (a type III manipulation, as seen in the next section). For instance, let us consider a "skip to the next track in the playlist" button in a music player. 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.

5.2 Semiotic Dimensions and Manipulation of Signs

According to [12], the underlying signification system (i.e., the computer languages to which users and programmers have access) has three semiotic dimensions: lexical, syntactic and semantic. Manipulations of these three dimensions effect different symbolic transformations (numbered from I to VII; see Fig. 1). The designer of a cross-platform system did not necessarily consider making 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 [7].

The lexical and the syntactic dimensions of computer languages are related to the surface features of the user interface, i.e., the look or appearance of a user interface. These dimensions are addressed in the perceptual or terminology and visual appearance inter-device consistency level in [2]. However, these two are different from the perceptual (look and feel) consistency dimension mentioned in [5] because they are not related to the feel (the behavior associated to interface elements).

The lexical dimension encompasses the color, shape and typefaces of labels and icons signs in the user interface, which corresponds to the lexical consistency level in [2] and to the semantic consistency level in [5]. The syntactic semiotic dimension relates to the layout or spatial organization of graphical signs. It encompasses the navigation and the arrangement of steps in a procedure or the ordering of operations in order to accomplish a task. In [2] this latter aspect is addressed in an inter-device consistency level called syntactical consistency, and in [5] it is called syntactic or interaction logic consistency.

The semantic dimension relates to the feel of the user interface, which corresponds to the semantic or partition of data and functions inter-device consistency level in [2]. This dimension is related to the meanings expressed by signs in the system and it allow us to identify two different subsets of manipulations to signs (Fig. 1): meaning-preserving manipulations (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. Every application requires that users learn a new and unique signification system used in HCI [12]. 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. This is especially important when it comes to cross-platform systems, since you have many versions of the same system and they cannot or should not be entirely consistent in every case [2]. A coherent user experience is the ultimate target of cross-platform service design [5]. Meaning-changing manipulations, however, can threaten the application's identity (and consequently the system's perceived coherence), and therefore should be avoided whenever possible in order to minimize more serious conflicts between system versions.

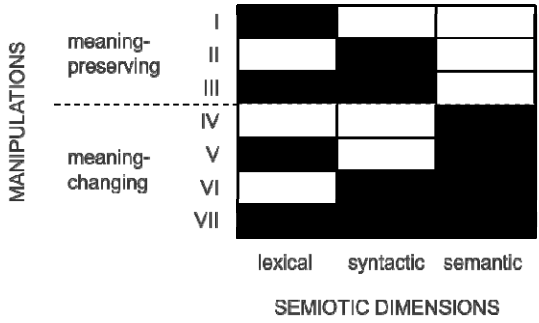

Fig. 1. Semiotic manipulation possibilities of symbols and signs from a computer symbol-processing perspective

Type I Manipulations. Type I changes correspond to *renaming* and *aliasing* operations that affect only the lexical component. This manipulation is partially related to the simplification and magnification content adaptation strategies mentioned in [19] and to the transducing adaption approach in [20], because it modifies the appearance of some piece of information (signs) present in the user interface. Changing the label of a button and changing an icon appearance are both examples of renaming. Renaming should be limited to lexical items that are not part of the application identity [12]. Besides, it should be used only when imposed by the device's constraints (e.g., summarizing a description or using synonyms for a label in order to fit in the screen) or the platform patterns (e.g., changing the appearance of a component or icon from the Android platform to accommodate the patterns in the iOS platform and vice-versa) to avoid unnecessary inconsistencies.

Whenever a sign is renamed between platforms (i.e., receives different names in each one), the user must follow a reasoning process to establish whether the object has the same function (meaning) as its instance in another version of the application [2]. Depending on the degree of the lexical inconsistency, the user might fail to associate an object (sign) with its function (expected meaning), causing a communication breakdown and a continuity problem. Resizing a user interface element is usually not a problem since users still have enough visual clues to judge whether the different sized objects are similar [2, 19]. Changing the color of an element, however, can lead to some issues since a color is by itself a sign and therefore express its own meanings (e.g., the red color is commonly associated to cancel or removing operations).

Type II Manipulations. Type II manipulations involve making changes only to syntactic components, i.e. *reordering* items: changing the layout placement of user interface elements (the adaptation strategy called rearrangement in [19]); or changing the navigation or the order in which operations are performed.

Type II manipulations are most common and inevitable when it comes to cross-platform systems. Resizing elements is often not enough to adapt the application interface when the display sizes and resolutions are different, which forces the designer to reorganize the layout to fit all the interface elements (or even to separate elements into two or more screens) to avoid pruning essential impermeable signs.

However, is worth noticing that any reordering in the layout, when not imposed by the device's space or resolution constraints, is unnecessary and counterproductive, negatively affecting the continuity or inter-usability of the system [2]. Keep in mind that, whenever this sort of manipulation occurs, users will have to make an effort to locate the object. At best, this will increase their workload (e.g., a person who is already used to the order of some labels in a platform will have to learn the order of the such elements again in the other platforms); at worst, if they can't locate the object quickly, they could conclude that the related function is unavailable on the new device [20]. Also, if the reordered interface elements were buttons, a distracted person might press the wrong button and perform some undesired action because the order with which this person was already familiar has changed. Moreover, from a semiotic perspective, the order of elements may also convey a relation of importance between them, and therefore such inconsistencies may cause an ambiguity when communicating to the user what the designer thought was more important. Finally, as shown in the analytical study in [7], a misplacement of even a simple element as a label may mislead the user and hinder the use of the system.

Type III Manipulations. Rearranging (type II) or changing the appearance (type I) of some components is often not enough. 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 (which corresponds respectively to the increase and reduction adaptation strategies mentioned in [19]). Not only must pruning preserve 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 are essential and thus cannot be pruned i.e., which signs constitute the application identity [12].

As in type II manipulations, one of the reasons why type III manipulations are likely to occur arises from differences between platform capabilities. However, even when technologically feasible, some functionalities and signs should still be distributed across platforms according to their utility (i.e., if they will be useful considering the platform or device context and purpose of use) in order to reduce the complexity on each application version [5]. However, if the composition of applications and devices and the way they are combined is not in line with the user's activity or needs, it may essentially hinder the resulting user experience [5]. From an inter-usability perspective this happens because the lack of some expected features hinders both knowledge and task continuity [2]. From a semiotic perspective, this mismatch relates with poorly defining the application's identity.

The negative effect of such inconsistencies may be mitigated by an appropriate degree of system transparency [2, 5]. The system should help the user to understand the potential and limitations of distinct technologies, and the different useful ways to

combine the system versions altogether. We go even further to say that whenever an appropriate degree of transparency is not reached, the user might end up thinking that some pruned feature is actually present in some other part of the system (type II manipulation) and therefore he will eventually get frustrated after endlessly looking for it. Alternatively, also due to a transparency problem, the user might think that some feature is not present (since it was not present in the other version) while it actually is. Although being unaware of an extra feature may not seem to lead to a problem, it will in case this feature performs some automatic operation without the user being aware of it. For instance, when first launched, most Android users were not aware of Instant Upload, a feature that allows you to upload photos and videos taken from your device automatically to a private album on Google+. These users ended up uploading undesired pictures and videos to their profiles, often consuming their 3G data allowance (when no Wi-Fi was available) without even noticing it.

Type IV Manipulations. A type IV manipulation affects the semantic (meaning), but not the lexical and syntactic dimensions. Hence, it involves using existing signs to mean something different from what they mean in another platform (e.g. the same button triggers different actions on each platform). As a result, when transitioning from one version to another, users may feel frustrated because of this conflict of meanings. This manipulation should be avoided since, to ensure continuity, the effect or result of the operations (i.e. their meanings) should be as similar as possible across devices [20]. Besides, if the user does not notice the functional inconsistency the failures may be even more frustrating and harder to recover from. For instance, the user might expect that, when pressing a button, an e-mail will be sent to the trash can just like when pressing a similar button in another version of the system. However, it might actually delete the e-mail permanently.

Type V Manipulations. A type V manipulation makes changes to meanings and lexical items but not to grammatical structures. In this case, it is important to mention that only because signs in different platforms have the same grammar structure (syntactical base), but different lexical and semantic bases, it does not necessarily mean that they represent the same sign. We can consider, for instance, that instead of a Type V manipulation, two type III manipulations took place: a sign was pruned and another one was added. Therefore, when the lexical bases of two signs are different enough from each other, such Type V inconsistencies may impair the use of a cross-platform system only as much as a type III inconsistency would. However, if the lexical base is somewhat similar, the user may not expect that some sign will have a different meaning across platforms, which will lead him to failures such as the ones that arise from a type IV manipulation.

Type VI Manipulations. Type VI affects both syntactic and semantic dimensions, and therefore it may involve reordering components or even eliminating components that change the conveyed meaning. As in type IV manipulations, since the lexical base does not change, the user may not understand or perceive at first that there are different grammatical structures associated with same sign across platforms, which

will lead to different effects. For instance, when closing an application in a platform there might be an intermediate step asking the user whether he wants to save or discard unsaved changes to a document, while in another version this step might be eliminated, leading the user to lose this data, a failure from which he may not be able to recover. Regarding rearranging graphical elements instead of operations, users will hardly assume that the same element (sign) will have a different meaning and purpose only because it is in a different place of the user interface.

Type VII Manipulations. Type VII makes changes to meanings, grammatical structures and lexical items. This manipulation can freely affect the inside of any sign capsule and thus it can potentially override the limits of the application identity. As discussed before, the application cannot be entirely consistent across platforms, but it should at least be coherent, otherwise they user will not be able to reuse any of the knowledge acquired when transitioning between platforms (i.e., knowledge continuity). Moreover, it may even seem to the user that such versions are not part of the same system. The concept of application identity helps to draw the boundary of how inconsistent, in the worst-case scenario, a cross-platform system might be: at least the signs that constitute the application identity should be should never be pruned or changed. Therefore, such manipulations should be avoided at all costs.

6 Conclusion and Future Work

This paper discusses the Cross-Platform Semiotic Inspection Method, CP-SIM, an extension of the Semiotic Inspection Method designed to evaluate cross-platform systems. In order to support evaluators using this method, we presented a more detailed account of the horizontal analysis of the method. In this paper, we contextualized (in the cross-platform scenario) the concepts used in the method better and also related these concepts to the ones found in literature about cross-platform systems evaluation based on cognitive theories. We believe this link is extremely helpful especially for evaluators that are not so familiar with Semiotics or Semiotic Engineering. Finally, we discussed in detail each type of sign manipulation, providing a better perspective on their actual and potential impact on the quality of cross-platform systems.

As future work, we plan to conduct several analytical studies to investigate further and to better characterize each type of sign manipulation in the context of cross-platform systems. We also plan to investigate the learning curve of the method in order to decide how to improve it.

Acknowledgments. Simone Barbosa thanks CNPq (process #308490/2012-6) for the support to her research work.

References

1. Antila, V., Lui, A.: Challenges in designing inter-usable systems. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011, Part I. LNCS, vol. 6946, pp. 396–403. Springer, Heidelberg (2011)

2. Denis, C., Karsenty, L.: Inter-usability of multi-device systems - a conceptual framework. In: 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. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007, Part II. LNCS, vol. 4663, pp. 702–703. Springer, Heidelberg (2007)
4. Seffah, A., Forbrig, P., Javahery, H.: Multi-devices "Multiple" user interfaces: development models and research opportunities. Journal of Systems and Software 73(2), 287–300 (2004)
5. Wäljas, M., Segerståhl, K., Väänänen-Vainio-Mattila, K., Oinas-Kukkonen, H.: Cross-platform service user experience: a field study and an initial framework. In: Proc. MobileHCI 2010, pp. 219–228. ACM (2010)
6. Huang, Y., Strawderman, L.: Introducing a New Usability Framework for Analyzing Usability in a Multiple-device System. In: Proc. Human Factors and Ergonomics Society Annual Meeting, vol. 55(1), pp. 1696–1700. SAGE Publications (2011)
7. de A. Maués, R., Barbosa, S.D.J.: Cross-communicability: Evaluating the meta-communication of cross-platform applications. In: Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) INTERACT 2013, Part III. LNCS, vol. 8119, pp. 241–258. Springer, Heidelberg (2013)
8. de Souza, C.S., Leitão, C.F., Prates, R.O., da Silva, E.J.: The Semiotic Inspection Method. In: Proc. IHC 2006, vol. 1, pp. 148–157. SBC (2006)
9. de Souza, C.S.: The Semiotic Engineering of Human-Computer Interaction. The MIT Press (2005)
10. Barbosa, S.D.J., Silva, B.S.: Interação Humano-Computador. Campus-Elsevier (2010)
11. Peirce, C.S.: The essential Peirce. In: Houser, N., Kloesel, C. (eds.), vols. 1&2. Indiana University Press (1992)
12. de Souza, C.S., Barbosa, S.D.J.: A semiotic framing for end-user development. In: Lieberman, H., Paternò, F., Wulf, V. (org.) End User Development: People to Flexibly Employ Advanced Information and Communication Technology, pp. 401–426. Springer (2006)
13. Florins, M., Vanderdonckt, J.: Graceful degradation of user interfaces as a design method for multiplatform systems. In: Proc. IUI 2004. ACM (2004)
14. Lin, J., Landay, J.A.: Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In: Proc. CHI 2008, pp. 1313–1322. ACM (2008)
15. Ghiani, G., Paternò, F., Santoro, C.: On-demand cross-device interface components migration. In: Proc. MobileHCI 2010, pp. 299–308. ACM (2010)
16. Paternò, F., Zichittella, G.: Desktop-to-mobile web adaptation through customizable two-dimensional semantic redesign. In: Forbrig, P. (ed.) HCSE 2010. LNCS, vol. 6409, pp. 79–94. Springer, Heidelberg (2010)
17. Öquist, G., Goldstein, M., Chincholle, D.: Assessing usability across multiple user interfaces. In: Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces, pp. 327–349 (2004)
18. Haskell, R.E.: Transfer of learning: cognition and instruction. Academic Press (2000)
19. Berti, S., Paternó, F., Santoro, C.: A Taxonomy for Migratory User Interfaces. In: Gilroy, S.W., Harrison, M.D. (eds.) DSV-IS 2005. LNCS, vol. 3941, pp. 149–160. Springer, Heidelberg (2006)
20. Paternò, F., Santoro, C.: A logical framework for multi-device user interfaces. In: Proc. EICS 2012, pp. 45–50. ACM (2012)