Hidden Biases in Semiotic Engineering
Introducing Communicability Evaluation to Multi-Touch Interface Design

Jan Derboven

CUO | Social Spaces, KU Leuven-iMinds
Parkstraat 45 bus 3605
jan.derboven@soc.kuleuven.be

ABSTRACT
This paper describes the use of Semiotic Engineering methods in the context of multi-touch interaction. Focusing on the analysis of user testing data, we describe the transferability of Semiotic Engineering methods (specifically the Communicability Evaluation Method) across two interaction paradigms. While Communicability Evaluation is traditionally used for the analysis of graphical user interfaces (GUIs), we assess its applicability in multi-touch interaction through a case study.

ANALYZING ‘NEW’ INTERFACES WITH ‘OLD’ METHODS-RESOURCES
While multi-touch technology has been around for a few decades, the increased interest in this technology since the early 2000s has led to a lot of new research in interaction design for multi-touch applications. Commonly considered a ‘new’ interface paradigm, multi-touch technology continues to offer numerous challenges to optimize design for interaction using touch and gestures. The case study described in this paper gives an overview of the efforts we made to tackle multi-touch interaction design challenges using Semiotic Engineering methods. We first sketch the original application context of Semiotic Engineering methods. Afterwards, we describe the rationale for applying these methods to multi-touch interface design, and the experience we had with the process of the transfer to multi-touch.

Original Application Context
The foundations of De Souza’s Semiotic Engineering theory date back to 1993, with a paper on ‘The semiotic engineering of user interface languages’ [3]. In it, De Souza frames her work as ‘an attempt to give theoretical support to the elaboration of user interface languages’, and to ‘sketch the basis of a theoretic approach to user interface language design’. Since 1993, this approach has evolved from a basic theoretical sketch to a complete semiotic ontology of HCI, with its own distinct methods. The main novelty in the Semiotic Engineering approach to HCI is the focus on HCI as a specific type of designer-user communication: the actual communication ‘is between designer and person, where the technology is the medium’ [6], and not between user and system. The strength of Semiotic Engineering lies in this shift in perspective: it can generate a ‘new account of known problems’ [5], opening up new solutions, and possibilities for design and redesign.

The two main Semiotic Engineering methods, ‘epistemic tools’ that should not be used to give directly answers to design problems, but to increase the problem solver’s understanding of problems and alternative solutions’ [5], are the Semiotic Inspection Method (SIM) and the Communicability Evaluation Method (CEM):

- **SIM** allows the analyst-expert to analyze the way a specific system communicates the designer’s messages to the user. As such, SIM is an inspection method that aims at reconstructing the designers’ communication: it evaluates whether or not designers’ intent has been communicated effectively through the design choices and interactive content in the system.
- **CEM** focuses on the designers’ communication through user observation. Users’ interactions with a system are analyzed within a very specific analytic scope, resulting in a reconstruction of the actual designer-user communication that unfolds as the users interact with the system [5].

These methods can either be used on their own, or in combination with each other. The case study described in this paper, however, focuses on the transfer of CEM to multi-touch applications.

CEM is a specific qualitative method that analyses user interfaces based on a semiotic interpretation of user test sessions [5]. CEM primarily focuses on interaction problems, and enables researchers to identify and classify user problems in a fine-grained analysis that adds important nuances to the analysis that would otherwise risk being overlooked. For instance, user problems due to insufficient system feedback are broken down further by CEM into several subcategories, such as users misinterpreting the design solution presented by the interface, users completely unable to make sense of interface icons, etc. In this way, CEM presents an important added value compared to other evaluation methods, in that CEM’s detailed assessment of
user problems provides designers with a wealth of detailed information. This information can help in making informed decisions during an application redesign.

CEM offers a semiotic framework to process lower-level observations, and draw more generalised conclusions in order to determine an overall semiotic application profile of an application. This process, from low-level observations to high-level semiotic profile, consists of three stages: *tagging, interpretation and semiotic profiling*.

The first stage in the Communicability Evaluation Method is to *tag all problems users encounter* according to a predefined coding scheme based on semiotic principles. In order to identify all communicative breakdowns between the user and the designer (i.e., instances where users fail to receive the communication as it was intended by the designers), all test participant actions in a user test need to be recorded and tagged according to the 13 tags proposed by De Souza and Leitão [5]. These tags are expressions that users might utter when being faced with problems during the test.

In the second, *interpretative stage*, the goal is to search for the higher-level problems users have in understanding the designers’ messages. In order to find these higher-order problems, all *tags are analyzed and organized* according to a number of different perspectives, such as the frequency and context of occurrence of each tag, and the existence of patterns in the sequences of tags [5]. CEM prescribes that the low-level tags should be grouped in a higher-level taxonomy of interaction and usability issues. While CEM proposes its own classification, the specific taxonomy used is left open [9] to other possibilities, such as the usability heuristics by Nielsen (see also [4]) or Shneiderman, or other, more domain-specific heuristics or guidelines.

In the final *semiotic profiling stage*, the goal is an in-depth characterization of the way the interactive product communicates with its users. Based on the previous stages, semiotic profiling can answer high-level questions on the way users interact and communicate with the computer system.

Although they have been applied primarily to point-and-click graphical user interfaces (GUIs) - examples include [8, 10] - the method descriptions offered by De Souza et al. (tools to increase the analysts’ understanding of user interface problems) suggest that Semiotic Engineering can be an invaluable theory in understanding and further developing innovative interface ‘languages’ or paradigms, such as multi-touch interface design.

**New Context: A Lot of Multi-Touch Experimentation, but Few Standards**

We considered multi-touch interfaces as being particularly interesting for a semiotic analysis, since they are often considered to be ‘Natural User Interfaces’ (NUIs) [15, 16]. The NUI paradigm is a user interface paradigm that goes beyond the point-and-click, metaphor-based interfaces of GUIs. The objective of NUIs, as described by its advocates, is to deliver intuitive, seamless and ‘unmediated’ experiences that unfold through natural human input [14].

The above characterization of NUI systems means one the one hand that the indirect input of graphical user interfaces using the WIMP paradigm are replaced by speech, touch and gesture — specifically touch and gesture in the case of multi-touch interfaces. On the other hand, the claim of ‘unmediated’ interaction seemed to imply that the content itself would serve as the interface: users interact in a direct way with the content, instead of needing a GUI with metaphors and icons to access the content [16]. However, the ‘unmediated interaction’ claim does not remain unchallenged: Saffer [11] argues that ‘metaphor will always play a role with our devices — it is impossible to use or understand them otherwise.’ In the same spirit, Microsoft detracts from their ‘unmediatedness’ claim by proposing that the NUI’s ‘icons’ are representations ‘in which a portion of the object stands for the object itself’ [16]. This phrase in itself describes a kind of mediation: it can be seen as a short, simple definition of metonymy.

Approaching NUI interaction as a distinct kind of mediated interaction, NUls can be evaluated using Semiotic Engineering. The interface, then, is seen as a communication channel between designer and user. We see several NUI design issues in which such semiotic analysis can be helpful. One of these issues is determining to what extent users can make sense of the system’s interface, and assess the need for explicit user assistance. User guidance, especially in multi-touch environments, is often regarded as unnecessary and merely a quick fix for a poorly designed application [13]. While exploring a system, users should be able to find out which functionality is available with as little user assistance as possible. This view is complementary to the idea that multi-touch interfaces should be ‘natural’, self-explanatory and intuitive. However, this idealistic view often contrasts with the way users actually explore and use multiple interfaces [14]. To learn more about issues such as this user guidance contradiction, we decided to use Semiotic Engineering, as a theory that is potentially well-positioned to shed a new light the way NUI interfaces communicate their functionality to the users.

---

1 Each tag in the method targets a specific interaction problem a user might encounter. Examples include *Why doesn’t it* (‘the user expects some sort of outcome but does not achieve it. The subsequent scenario is that he then insists on the same path, as if he were so sure that some function should do what he expects that he simply cannot accept the fact that it doesn’t’) and *I can’t do it this way:* ‘Sometimes the user follows some path of action and then realizes that it’s not leading him where he expected. He then cancels the sequence of actions and chooses a different path.’ [9].
**Stories of Transfer: Mixed Feelings**

When we were evaluating MuTable, the multi-touch tabletop project we use as a case study, in late 2009/early 2010, not many guidelines or resources were available for designing good multi-touch user interfaces, or for evaluating them. In fact, the field of multi-touch interaction still is quite new, with much room for additional research: the lack of a clear gestural UI language is brought forward by several authors [7, 15], Don Norman and Jakob Nielsen stating that ‘these interaction styles are still in their infancy, so it is only natural to expect that a great deal of exploration and study still needs to be done’ [7].

**Figure 1. Main MuTable functionality. The functionality includes navigation elements and several widgets: 1. central ball menu (collapsed 1a. and expanded 1b.), 2. a submenu, 3. a typewriter tool, 4. a presentation creation tool (with a slide opened), 5. a file browser tool, 6. a piece of content (picture) opened.**

Due to a particular mix of characteristics (e.g., a focus on productivity in a school context, see figure 1) and a lack of a clear gestural UI language, the design of the tabletop included a number of novel design solutions, sometimes inspired by other multi-touch developments, but not familiar to the general public. With an interface design based on much experimentation and little standards, an evaluation method capable of providing detailed feedback about non-standard user interfaces was needed. We chose for the Communicability Evaluation Method as a semiotic way into the analysis of multi-touch interfaces.

In the end, the application of CEM to the MuTable multi-touch platform was successful (it was published in a special issue on ‘Developing, evaluating and deploying multi-touch systems’ [2]). However, the cost at arriving at this outcome was unanticipatedly high. Although the method is not GUI-specific at first glance, a lot of time has been devoted to tailoring the method to the new multi-touch context. De Souza and Leitão [5] state that the Semiotic Engineering methods can be used in both ‘technical and scientific contexts’. However, while the process of transfer proved to be worthwhile in the scientific perspective of the HCI research group I am working in, such an effort would probably have been considered a terrific waste of time and resources in an industrial setting, where the study focus would have been more on evaluation results. In other words, the time needed to do the study described in this paper was well-spent because its main outcome was the application of a research method to a new domain [12], rather than the specific evaluation results of the study.

**THE TRANSFER FROM GUI TO NUI IN DETAIL**

Developed before the boom in multi-touch application development, it is not realistic to expect the Semiotic Engineering Methods to be tailored to multi-touch situations. Moreover, Prates et al. [9] explicitly acknowledge that the ‘method applies basically to single-user interfaces’, leaving the possibility of introducing new tags and categories open for e.g. multi-user and artificial intelligence applications.

While we were expecting to add some extensions to CEM based on Prates et al. [9], the issues are more profound than making a few tweaks to the method. The changes were absolutely crucial for the application of the method to succeed. To be sure, the analytic power of the CEM framework depends heavily on the appropriateness of the low-level tags, and the higher-level interpretation taxonomy. We distinguish between two main difficulties: augmenting the method with useful, new tags and categories on the one hand, and dealing with the ‘hidden WIMP bias’ (windows, icons, menus, pointers) of the existing tags.

**Creating New CEM Tags and Categories**

Introducing new tags and categories for the use of CEM in new contexts is not always self-evident, despite the fact that Prates et al. [9] almost casually mention this possibility. In new contexts, it is difficult to anticipate beforehand which user problems ‘are likely to occur’, and need a new, separate tag, especially when applying the framework to new interface paradigms. This requires at least some experience with the technology and the test user group before one can make informed extensions to the CEM framework. Less appropriate tags and interpretation taxonomies will significantly reduce the framework’s analytic power, by failing to point out the important ‘communicative breakdowns’ between designer and user.

Apart from the anticipation problem, it is a challenge to decide on what level to introduce new elements: as described above, CEM includes a low ‘tag’ level, and a higher ‘category’ level. In other words, after deciding to add an element, it is an equally important step to decide whether to introduce a series of new, ‘dedicated’ lower-level tags, or to re-use the existing tags, but add a new higher-level category to attribute them to. In situations where this decision is not clear-cut, the options need to be considered carefully, as they have important consequences for the resulting analysis.

We would like to offer the example of user problems related to gestural interaction. In specific, consider the
situation in which a user performs a gesture in such a way that it is not recognized by the system. The system does not recognize the gesture and remains inactive, and the user does not understand why his actions have no effect. One way of approaching this issue is to create a new tag for this type of gestural issue, as this interaction issue is a ‘new’ one, specifically relating to the use and availability of gestures. On the other hand, this gestural issue can be considered a slightly different version of an issue already present in the taxonomy. The original Why doesn’t it? tag refers to a user struggling to make sense of an interface, because a specific part does not react as expected. In a GUI environment, this is typically the case when there is a breakdown in the user’s understanding of the system, the user not understanding why the system is unable to complete the request. While the end result is the same in the gestural example (the user struggling to make sense why the system doesn’t), the gestural issue is different: the request itself may be valid, but the user’s execution of the request is not. CEM’s original classification taxonomy does not allow for this type of issue, where the user’s intentions are correct, but the user’s actions are not recognized by the system.

For the gestural issue described above, we decided to go with the second option. Instead of adding a series of gesture-specific low-level categories, a higher-level category was added to the interpretation taxonomy. The existing low-level tags were interpreted more broadly to include gestural problems. The new Gestures category distinguishes between issues on a higher level: issues concerning communication about input methods are differentiated from those related to e.g. meaning assignment (interpretation of interface elements) and navigation. During analysis, this high-level category allowed to attribute a diverse set of breakdowns to the communication about gestures the designers had embedded in the user interface.

From the perspective of Woolrych et al.’s [17] description of methods as a set of resources, the adjustments to CEM (adding new tags and categories) were made based on local knowledge resources. Application-specific knowledge about multi-touch gestures was used to create the adjustments. However, the adjustments also needed to fit in the specific semiotic orientation of the method, focusing on designer-user communication. Therefore, the adjustments to CEM had to be considered carefully, balancing the local knowledge resources from the project with the theoretical specific orientation of the method.

**WIMP Bias**

Apart from the necessity to revise CEM for multi-touch purposes, the method’s appropriateness was also decreased by another, more covert ‘WIMP bias’ of sorts. While none of the tags or categories explicitly refer to windows, icons, menus or pointers, applying the method to a new domain such as multi-touch showed that the tags implicitly target common GUI problems. I will offer a few examples of this bias.

One example of a WIMP bias can be found in the need to create the Gestures category described above. Semiotic engineering, developed roughly between 1993 and 2005, seems to take the interaction modality for granted. Mice and keyboards were the dominant input mechanisms at the time: these input mechanisms, and the way the input is transferred to, and visualized in the system is not taken up in the analysis. However, with a lack of gestural interaction standards in multi-touch interaction, the interaction modality is foregrounded, and analytic devices (tags or higher-level categories) targeting input mechanisms need to be added.

Another examples of the WIMP bias are the tags created for breakdowns due to multi-step navigation paths (implying deep menu structures, and embedded functionalities: the I can’t do it this way tag), or on the modality of user interfaces (applications offering separate modes in which specific functionality can be available or not: the Where am I? tag). During a first inspection of the CEM tags and categories, the user-centered wording of the tags themselves (referring to the user in the first person) does not reveal a bias, and the method seems broadly applicable. It is only when applying it that the WIMP bias becomes apparent. In multi-touch multi-user systems, these WIMP-styled modality and menu structure-oriented tags are often not very relevant. Deep menu structures are generally avoided in multi-touch applications, and while every single screen in a multi-touch app can be seen as a separate mode [1], these modes are generally quite distinct, and offer only a limited set of functionality, reducing the risk of confusion. These tags were not removed from the taxonomy, to avoid overlooking these issues during analysis. However, in the analysis of the MuTable results, their occurrence was marginal.

In sum, specific elements in the method appeared to be tailored to GUI interactions, while the method needed extensions to cater for specific non-GUI interactions. These observations lead to the conclusion that CEM implicitly seems to be tailored to GUI evaluation. The creators of CEM do acknowledge that for specific applications, extensions to the method are needed [9]. As such, they explicitly present multi-user applications and artificial intelligence systems as outside the scope of the original CEM method. However, they do not mention interaction modality (keyboard and mouse vs. touch) when discussing the scope of the method. In this sense, CEM’s scoping resources [17] are not entirely adequate.

**Extra: What About Other Semiotic Engineering Methods?**

After the above issues in applying CEM to multi-touch, we further explored the other main Semiotic Engineering method, the Semiotic Inspection Method (SIM), to see whether or not this method has similar issues associated
with it. SIM was not used to evaluate the MuTable interface, as SIM is targeted at reconstructing the designer’s intended message from the interface. In this case, the technology was evaluated by the application’s designers themselves. Therefore, as the ‘designer’s intent’ was well-known to the evaluators, a SIM analysis would have lost at least some of its explanatory power.

A short, informal application of the method to the Windows 8 ‘Modern UI’ (a ‘touch first’ interface) showed a similar WIMP bias as CEM. This bias can be found in the first three steps of the method: the analysis of ‘metalinguistic signs’, of ‘static signs’, and of ‘dynamic signs’. These steps are aimed at deconstructing the metacommunication message, allowing the researcher to inspect in great detail what and how the designer communicates with each type of sign [5].

In typical GUI interfaces, a clear separation can be made between the three sign types described above. De Souza and Leitao offer the example of MS Word, in which metalinguistic signs (help texts, either within the application or online support material on Microsoft websites) can clearly be separated from static signs (menu items, toolbars, ...) and dynamic signs (‘what you see is what you get’ WYSIWYG on-the-fly changes that happen in response to the users’ actions). However, in contemporary (multi-touch) interfaces, this sign type division as a starting point for analysis becomes problematic:

- **Metalinguistic signs.** Increasing numbers of smartphone and tablet apps offer very little or no explicit user guidance, thereby all but eradicating the metalinguistic sign type.
- **Static signs.** In smartphone and tablet interfaces, ‘static’ interface elements are often few and far between. While mail apps typically do have a static menu bar in some screens, there are probably few apps that have static elements that recur across all screens. As stated above, this creates a situation in which every single screen in a multi-touch app can be seen as a separate mode [1]. The Windows 8 operating system, even without a user performing any action, constantly refreshes mail messages, pictures, etc. (see figure 2), reducing the static signs to a screen title, and fixed, empty boxes that are filled up with dynamic content.
- **Dynamic signs.** The SIM analysis of ‘dynamic signs’ is targeted at investigating interaction [5]. Dynamic signs are seen primarily as system feedback to the users’ actions, as in the WYSIWYG example above. However, in contemporary interfaces, the system often refreshes and changes information without user action (see figure 2). This implies that the dynamic sign category needs further refinement into dynamic signs as a result of user interaction, and dynamic signs as a result of system-driven updates.

**Figure 2. Windows 8 Modern UI.**

The above analysis shows that SIM’s analysis of contemporary touch interfaces can be expected to be very heavy on ‘dynamic signs’, with hardly any metalinguistic and static signs. While this observation in itself can be seen as merely a characteristic of touch interfaces, it becomes more problematic in the fourth step of the method (steps 1-3 being a separate analysis of all three sign types). Step four is based on the first three steps, and ‘collate[s] and compare[s] the results of segmented metacommunication analysis. The aim is to detect inconsistencies and/or consistent relationships and patterns between elements collected in segmented analysis’ [5]. While SIM treats the three sign types as equally important in the analysis, it is clear that this position is no longer tenable. The relative importance of the sign types needs to be reconsidered. Especially the dynamic sign type needs to figure more prominently in the analysis; the category also needs further refinement in subcategories to allow for an adequate analysis. Without such modifications, the semiotic inspection methods loses its much of its value due to a mismatch between the interface design under analysis, and the methodological tools.

**CONCLUSION**

The Semiotic Engineering case study presented in this paper shows that the transfer of a method from one context to another should not be taken lightly. In the Communicability Evaluation Method transfer, the changes to the method did more than ‘augment’ its applicability. For instance, the addition of a Gestures category to the method to cater for the specifics of multi-touch interaction was absolutely crucial for the research to succeed. As for the Semiotic Inspection Method, the short exploration presented above showed that a change in context to multi-touch applications also deeply influences the applicability of the method.

While the case study presented here describes a transfer of methods that are probably somewhat less familiar to the majority of HCI researchers, the results do indicate that the transfer of a method to a new context needs to be done with care. Even when no specific issues seem to pop up at first inspection, one needs to be watchful: hidden biases and other difficulties are not at all impossible.
REFERENCES


