The Design of Slow-Motion Feedback

Jo Vermeulen, Kris Luyten, Karin Coninx
Hasselt University – tUL – iMinds
Expertise Centre for Digital Media
Diepenbeek, Belgium
firstname.lastname@uhasselt.be

Nicolai Marquardt
University College London
UCL Interaction Centre
London, UK
n.marquardt@ucl.ac.uk

ABSTRACT
The misalignment between the timeframe of systems and that of their users can cause problems, especially when the system relies on implicit interaction. It makes it hard for users to understand what is happening and leaves them little chance to intervene. This paper introduces the design concept of slow-motion feedback, which can help to address this issue. A definition is provided, together with an overview of existing applications of this technique.

Author Keywords
Design; implicit interaction; slow-motion; feedback.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
There is a large potential for miscommunication between users and systems in *implicit interaction* — where the system takes actions based on implicit input that often occur beneath the user's radar [3,6]. These issues can lead to unintended actions, undesirable results and difficulties in detecting or correcting mistakes [6]. Indeed, before users can intervene, they must first understand what the system is trying to do [9].

Moreover, computers can take action in a fraction of a second, much faster than we humans are able to notice. This only further exacerbates the challenge of providing feedback in implicit interaction, as the user's implicit input could trigger thousands of system actions before they would even be aware of having interacted with the system. There is a thus a misalignment between the system's timeframe and the user's timeframe, as rightly argued by Bellotti et al. [3] in their paper on the challenges of sensing systems.

In this paper, we introduce the design concept of *slow-motion feedback*. Just as we would speak slowly when explaining something to a small child, computer systems could also slow down to make sure that their users understand them properly. By slowing down, users are also given time to notice what is happening and intervene, if necessary.

DIS '14, June 21 - 25 2014, Vancouver, BC, Canada
Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-2902-6/14/06…$15.00.
http://dx.doi.org/10.1145/2598510.2598604 ACM

A well-known example of slow-motion feedback is Gmail's 'undo send' feature that provides users with a configurable 5 to 30-second window to *undo* sending out an email (which is technically impossible). While Gmail shows feedback to the user informing them about the sent email, the actual sending of the email is delayed to allow users to cancel the action in progress. While slow-motion feedback is mainly used for safety reasons in this situation, it has several applications to improve awareness and prevent mistakes in implicit interaction, as we will discuss later.

Our contributions in this paper are twofold:

- We introduce a design space to reason about the time at which feedback is provided and use this framework to define slow-motion feedback;
- We give an overview of notable existing applications of slow-motion feedback and discuss its potential for improving awareness and providing opportunities for control in implicit interaction.

DESIGN SPACE
In order to come to a definition of slow-motion feedback, we present a design space that allows us to reason about the different possibilities about how and when *information about the result of an action* can be provided. The design space consists of two different dimensions: the *level of detail* about the result of an action, and the *time* at which the information is provided (Figure 1).

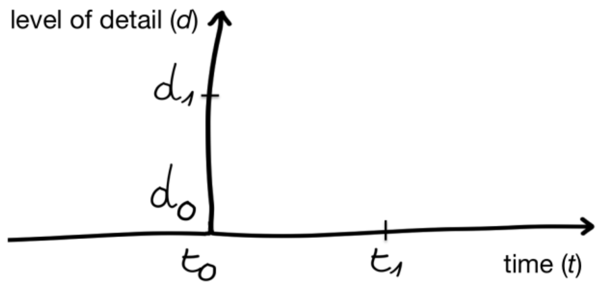

Figure 1: The design space for when and how information about the result of an action can be provided. These axes (*time* and *level of detail*) apply to the rest of the figures in the paper.

We consider both explicit actions initiated by the user, as well as actions initiated by the system that are planned routine tasks (e.g., auto-save in a word processor) and actions based on implicit input (e.g., a public display that reacts to a user's presence). There are two important events that we consider for actions: when they start, and when they are completed.

Time (t): The time dimension represents the time at which events occur, such as the start of an action, or when information is shown to the user. It consists of all moments in time that are relevant to the interaction. Again, we define two key moments: at time t_0, the action is started (either by the user or the system); and at time t_1, the action has been completed by the system.

Level of detail (d): This dimension represents how much information the user receives about the result of their action. While this dimension is hard to quantify and tends to be discrete, we define two important values for this dimension: d_0 and d_1. The level d_0 signifies the situation in which the user does not receive any information at all about the result of their action. At level d_1, on the contrary, the user receives complete information about the result of the action.

Origin and shape of the curve: We define the origin of the graph as (t_0, d_0); when the action starts and no information is provided yet. The curve's shape illustrates how information is revealed (non-continuously or continuously), how much information is provided and when, and whether the user has time to intervene.

From these basic definitions, we can derive a number of key regions:

- $t \geq t_1$: the time period *after* the action
- $t \leq t_0$: the time period *before* the action
- $t_0 \leq t \leq t_1$: the time period *during* the action
- $t_1 - t_0$: the amount of time available to the user to intervene, e.g., to cancel an unwanted action or correct the system. Note that the user must be aware that the action is taking place ($d > d_0$).

Feedback Patterns Covered by the Design Space

We illustrate the different regions in the design space with a number of common feedback patterns.

After the Action: Only Feedback

In graphical user interfaces (GUIs), information about the result of an action is typically only provided after the action has been completed, or in other words when $t \geq t_1$, as shown in Figure 2 (left). In this specific situation, information is provided in full detail ($d = d_1$).

Even though the action might take some time to complete (i.e., $t_1 - t_0 > 0$), users can only intervene when they have information about what is happening. If no information is provided before the action has been completed ($d = d_0$), users have no way to prevent the action from occurring. A typical solution offered by GUIs is to allow users to revert back to a previous state, e.g., via an *undo* command. This strategy is only suitable when users explicitly trigger actions. It would be cumbersome in implicit interaction, where users might get frustrated of repeatedly attempting to correct unwanted behavior.

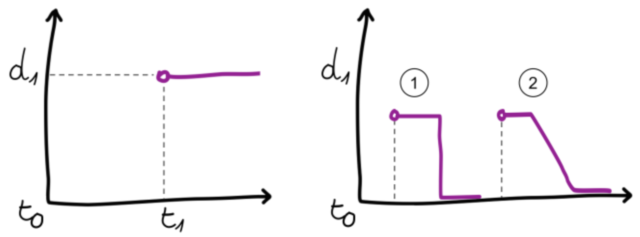

Figure 2: Feedback (*left*) and different options for the duration of feedback (*right*).

There are different options regarding the duration of feedback, as shown in Figure 2 (right). Feedback can be an inherent part of the user's ongoing task, and will then remain visible (e.g., text that has been added to a document), which is the case in Figure 2 (left). Other kinds of feedback might be temporary and disappear quite quickly, such as subtle notifications when a word processor has auto-corrected a word. Note also that in this case, the provided information is not complete ($d_0 \leq d \leq d_1$), but can be sufficient for the user. Not all notifications are temporary though, some might also be important enough to remain visible until the user deals with them (e.g., notifications about software security updates).

During the Action: Intermediate Feedback

Another typical pattern is showing information during the execution of the action ($t_0 \leq t \leq t_1$). This allows users to keep an eye on what is happening and to intervene if necessary. This type of feedback is commonly used for long-running tasks to inform users about the current state of the system. While several curve shapes are possible, the level of detail is commonly increased incrementally, as shown in Figure 3 (left).

Typical examples of incremental feedback are applications that allow users to preview results while they are being processed. When loading a webpage, for example, users can already see partial content coming in (e.g., the general layout, text, and images) before the webpage is fully loaded. Should the user suddenly realize that this was not the webpage they were looking for, they can easily go back,

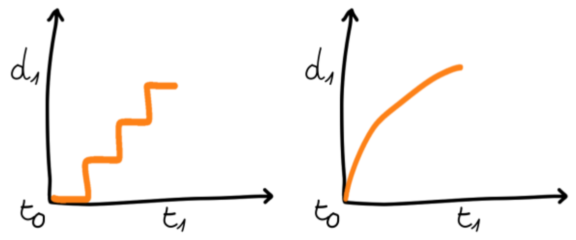

Figure 3: Intermediate feedback: incremental (*left*) and continuous (*right*).

without having to wait for the entire page to be loaded. An example of continuous intermediate feedback, is the OctoPocus gesture guide [1]. When users perform a gesture, it continually updates the possible remaining gesture paths.

Before the Action: Previews

Information about the result of an action can also be provided before the action has been started ($t < t_0$), as shown in Figure 4. Users are essentially given a glimpse into the future, which can be useful to help them in choosing the right action [8]. This information could remain visible or disappear once the action is being executed.

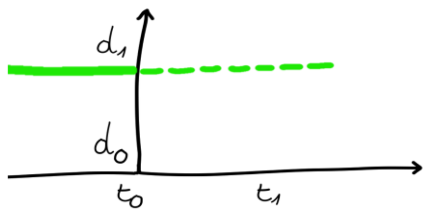

Figure 4: Previews show information before t_0.

A typical example of this is a *preview*. Microsoft Word, for example, shows the result of changing the font color of a selection when the user hovers over the different color buttons. Although one can argue over when an action actually starts in this case (when the user hovers over a button or when she clicks it?), the user still has to confirm the action before it is executed by the system. Any information provided before executing the action, is considered to be on the left side of the time axis ($t < t_0$).

DEFINING SLOW-MOTION FEEDBACK

We can define slow-motion feedback using the previous design space. It is clear that slow-motion feedback does not make sense for long running tasks, i.e., when the duration of an action is long enough so that the user has the opportunity to react (e.g., loading a webpage over a slow connection).

Slow-motion feedback amplifies the time difference between t_1 and t_0 ($t_1 – t_0$) or the duration of an action in the user's timeframe, as shown in Figure 5. The system's response to the user's action is postponed by delaying t_1 to t_2 ($t_1 \rightarrow t_2$) so that users have ample time to notice that the action is occurring and the option to intervene. The available time to notice that the action is happening is $t_2 – t_x$ for a certain time $t_0 \leq t_x \leq t_2$ where information is being provided to the user (in other words: ($t_2 – t_x$) for (t_x, d_y) where $t_0 \leq t_x \leq t_2$ and $d_y > d_0$). Designers could rely on animations [4] to transition between t_x and t_2, such as slow-in/slow-out to improve motion predictability [5]. Note that $t_x = t_0$ in Figure 5.

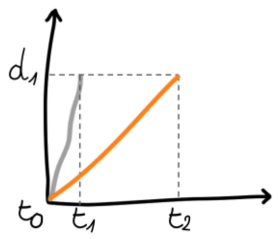

Figure 5: Slow-motion feedback amplifies the time to intervene by showing feedback until t_2 instead of t_1.

APPLICATIONS OF SLOW-MOTION FEEDBACK

In what follows, we demonstrate a number of notable successful applications of slow-motion feedback.

Emphasizing Change

Phosphor by Baudisch et al. [2] visually augments GUI widgets to emphasize changes in the interface and leave a trail to show users (in retrospect) what just happened. Phosphor increases the already existing feedback's level of detail (an increase in *d*) and the time that it is shown to the user (an increase in *t*), as illustrated in Figure 6.

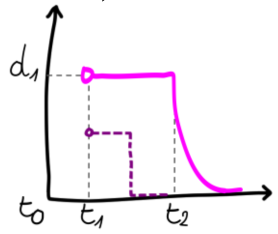

Figure 6: Phosphor increases both the level of detail and time.

System Demonstration

Ju et al.'s electronic whiteboard [6] uses slow-motion feedback to transition between an ambient display mode and a whiteboard based on the user's distance to the display. It shows an animation moving all content from the center of the board to the sides when a user steps closer. This happens slowly enough so that users both notice it, and have the time to react if it was not what they wanted (Figure 7). Users can override the automatic action of making space by grabbing content and pulling it back to the center. When users see an action being executed slowly (in this case: content that is moving to the side of the display), users can already predict what will happen and override the system's action before they have complete information at their disposal.

Progressive Feedback

Marquardt et al.'s gradual engagement design pattern for proxemic interactions [7] uses proximity to facilitate information exchange between devices. It is comprised of three stages in which more information is shown as the user's engagement with the system increases (e.g., by approaching a device). Marquardt et al. [7] assume that users will approach or orient themselves towards a device when they are interested in interacting with it.

An interesting feature here is that users control the speed at which information is revealed. The faster users approach a device, the faster information is shown, which realigns the system's timeframe with their own (Figure 7). In this case, the natural hesitation of novices and the rapid approach of experts might have exactly the intended consequences.

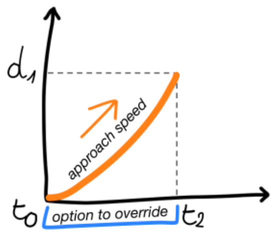

Figure 7: System demonstration uses slow-motion feedback to allow users to intervene. Progressive feedback gives users control over the speed at which information is revealed.

Visualizing Causality

Slow-motion feedback can also be used to improve the understanding of causality. Vermeulen et al. [9] visualize the inner workings of different sensors and devices in a smart environment by projecting animated lines into the environment that connect them when events and actions are triggered. In one of their scenarios, a motion sensor causes a light to be turned on. The system's action (turning on the lights) is slowed down here and timed to exactly coincide with the moment in time at which the animated line that started from the motion sensor reaches the light (Figure 8 left). Again, this allows users to override the system action.

Postponed Feedback

Ju et al. [6] have also applied another strategy for slow-motion feedback. Their electronic whiteboard performs automatic stroke clustering in the background while the user is drawing. The system provides feedback about the clusters by surrounding strokes in dotted light-gray bounding boxes. However, to avoid interrupting users while they are drawing, this feedback is only shown when the user steps back. When users notice a misclustering, they can override the system's action by redrawing the outline.

The interesting aspect of this approach is that the action and feedback cycle is shifted into the future (Figure 8 right). Instead of increasing the time between start and end of the action, users are only made aware of the action when they can be interrupted (t_2), similar to attentive interfaces [10].

DISCUSSION

Designing systems that rely on implicit interaction remains challenging. We feel that slow-motion feedback is a promising technique to increase awareness of system actions and to provide users with more opportunities for control.

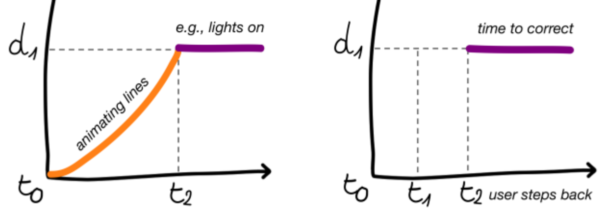

Figure 8: *Left*: Using slow-motion feedback to visualize causality. *Right*: Postponed feedback, which is only shown after t_2, even though the action was already completed at t_1.

However, there are also a number of implications of applying this technique. An open issue is how slow-motion feedback can be applied to time-critical tasks, as it might have a negative effect on the overall task completion time. While this will be negligible in most cases, when applied to several sequential micro-interactions, the cumulative effect over time might be too large to ignore.

In addition, more work is needed to take into account diverse groups of users. If the speed of slow-motion feedback is fixed for all users, there will be situations in which the provisioning of feedback will be either too slow (e.g., for experts) or too fast (e.g., for novice users). We see the biggest potential in approaches that allow the user to control the speed at which information is provided.

ACKNOWLEDGEMENTS

This paper was inspired by discussions at the Dagstuhl Seminar on Proxemics in HCI (#13452).

REFERENCES

1. Bau, O. and Mackay, W.E. OctoPocus: a dynamic guide for learning gesture-based command sets. *Proc. UIST '08*, ACM (2008), 37–46.
2. Baudisch, P., Tan, D., Collomb, M., et al. Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects. *Proc. UIST '06*, ACM (2006), 169–178.
3. Bellotti, V., Back, M., Edwards, W.K., Grinter, R.E., Henderson, A., and Lopes, C. Making sense of sensing systems: five questions for designers and researchers. *Proc. CHI '02*, ACM (2002), 415–422.
4. Chang, B.-W. and Ungar, D. Animation: From Cartoons to the User Interface. *Proc. UIST '93*, ACM (1993), 45–55.
5. Dragicevic, P., Bezerianos, A., Javed, W., Elmqvist, N., and Fekete, J.-D. Temporal Distortion for Animated Transitions. *Proc. CHI '11*, ACM (2011), 2009–2018.
6. Ju, Lee, and Klemmer. Range: exploring implicit interaction through electronic whiteboard design. *Proc. CSCW '08*, ACM (2008), 17–26.
7. Marquardt, N., Ballendat, T., Boring, S., Greenberg, S., and Hinckley, K. Gradual Engagement: Facilitating Information Exchange Between Digital Devices As a Function of Proximity. *Proc. ITS '12*, ACM (2012), 31–40.
8. Vermeulen, J., Luyten, K., van den Hoven, E., and Coninx, K. Crossing the Bridge over Norman's Gulf of Execution: Revealing Feedforward's True Identity. *Proc. CHI '13*, ACM (2013), 1931–1940.
9. Vermeulen, J., Slenders, J., Luyten, K., and Coninx, K. I Bet You Look Good on the Wall: Making the Invisible Computer Visible. *Proc. AmI '09*, Springer-Verlag (2009), 196–205.
10. Vertegaal, R. Introduction to the Special Issue on Attentive User Interfaces. *Commun. ACM 46*, 3 (2003), 30–33.