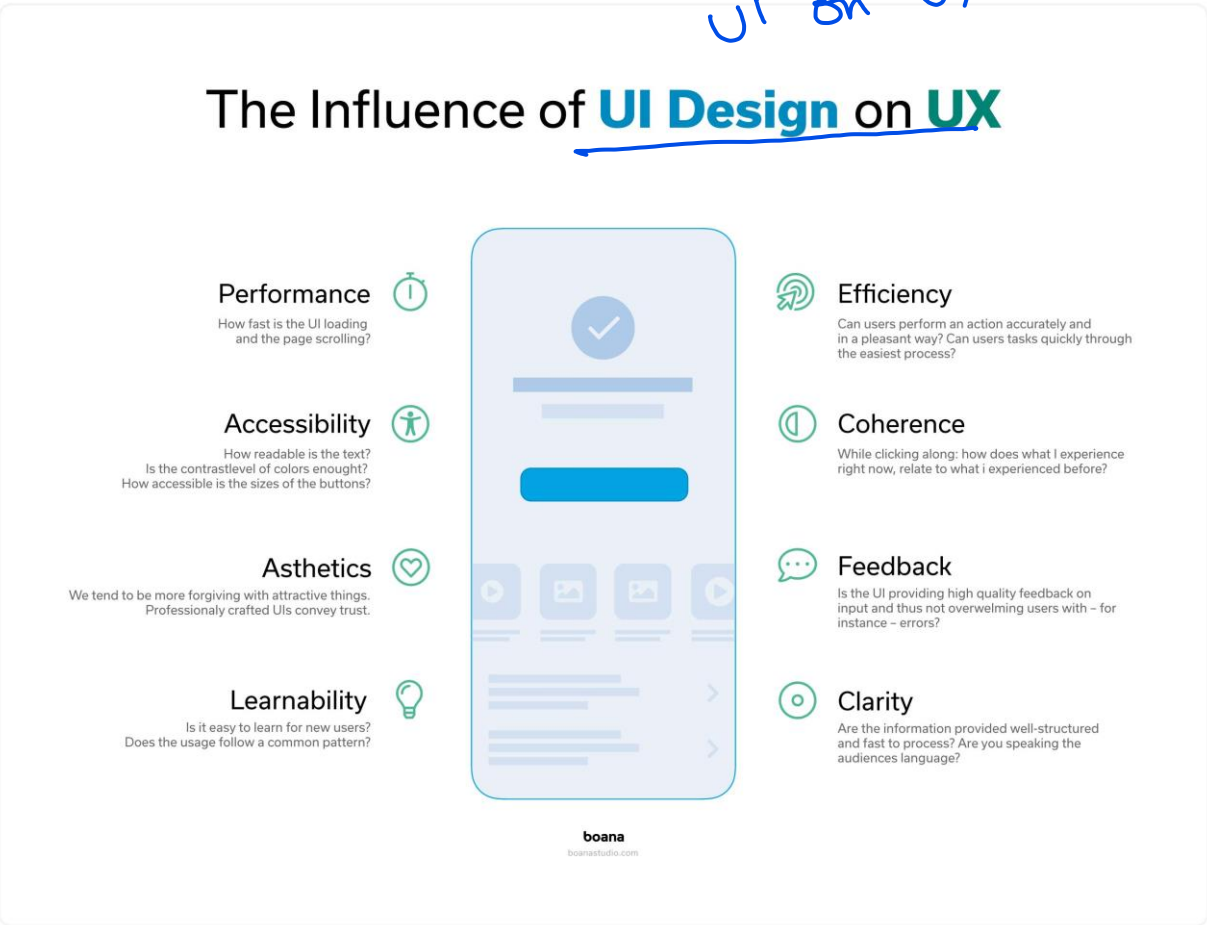
Chapter 4:

Principles for UI-Design by Shneiderman

Overview



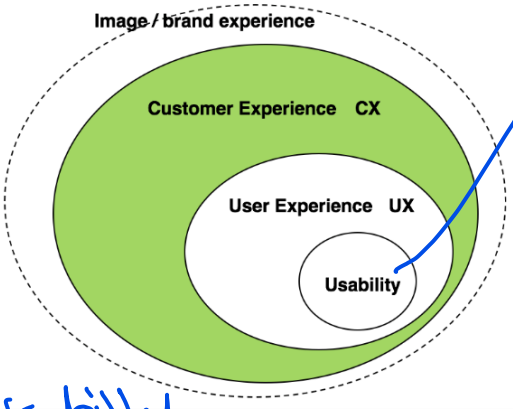
Excurse: User Interface Design (**UI**) vs. User Experience Design (**UX**)

User Interface Design (UI) and User Experience Design are both crucial to a product and work closely together. But despite their professional relationship, the roles themselves are quite different, referring to distinct aspects of the product development process and the design discipline.

User Interface Design	User Experience Design
This is how it looks like	This is how it feels like
User interface (UI) is anything a user may interact with to use a digital product or service. This includes everything from screens and touchscreens, keyboards, sounds, and even lights.	'User experience' encompasses all aspects of the end-user's interaction with the company, its services, and its products. (Don Norman, Jakob Nielsen)
UI is focused on how a product's surfaces look and function and develops the more tangible elements (of the application)	Is focused on the user's journey to solve a problem by looking at the users and the problems they encounter

User experience ≠ Usability

Usability is a quality attribute of the UI, covering whether the system is easy to learn, efficient to use, pleasant, and so forth.



Principle 1: Recognize User Diversity

Great user experience starts with a good understanding of your users. Not only do you want to know who they are, but you want to dive deeper into understanding their motivations, mentality, and behavior.

Accessibility and diversity ensure the usability of products by everyone. An inclusive UX design combines different perspectives to meet diverse user experiences. **There is no "average" user** and even one specific task will have various requirements based on who will want to perform it.

Inclusive UX ; No Avg User.

Example: consider an online travel agent

- This travel agent needs to perform and book many flights a day – everyday
- A teacher organizing a field trip (once a year) making bookings for a large group
- A businessperson changing bookings while travelling
- A family looking for a package holiday

You can structure the problem by:

a) Usage profiles
b) Task profiles

Usage profile
Task profile

To better describe your target user, you can use Persona profiles that characterize the typical user of your application (not the average!)[1].

Typical user vs Average User

User-Centered design ; immersion into end user needs

Persona: user-centered design and human-computer-interaction technique that promotes immersion into end-users' needs (Alan Cooper)

Creating software that is appropriate for a specific target group (e.g. 0,1% of the population) may still find a large user base (in Europe and the US this may be more than half a million people!).

What is the background of the user?

- Age
- Gender
- Education
- Cultural background

Know your user

What are their goals, motivation, personality?

What is their experience?

- Novice users
- Knowledgeable intermittent users
- Expert frequent users

Different people have different requirements for their interaction with computers.

User vs. Customer

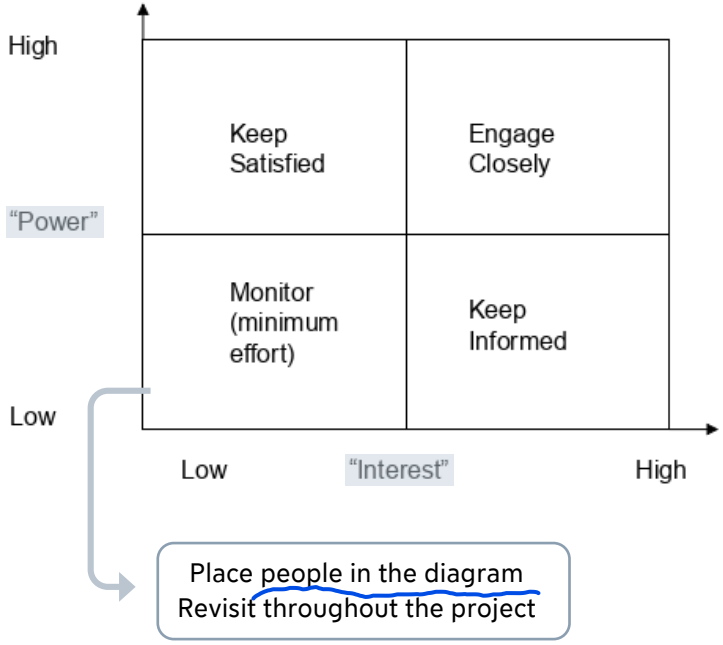
User != Customer.
Design for User ✓
Pleasing customer difficult

- Customer is often not the user!
- Creating awareness with the customer for the user
- Design for the user not to please the customer (this is a difficult one)
- Clear assessment of target group and personas will help to convince the customer

Convince customer

In order to identify your users customers and other involved parties you need to take into account, you can conduct a **Stakeholder analysis**.

How do you react to stakeholders of the product?



Place people in the diagram
Revisit throughout the project

Identify stakeholders

Categorize stakeholders

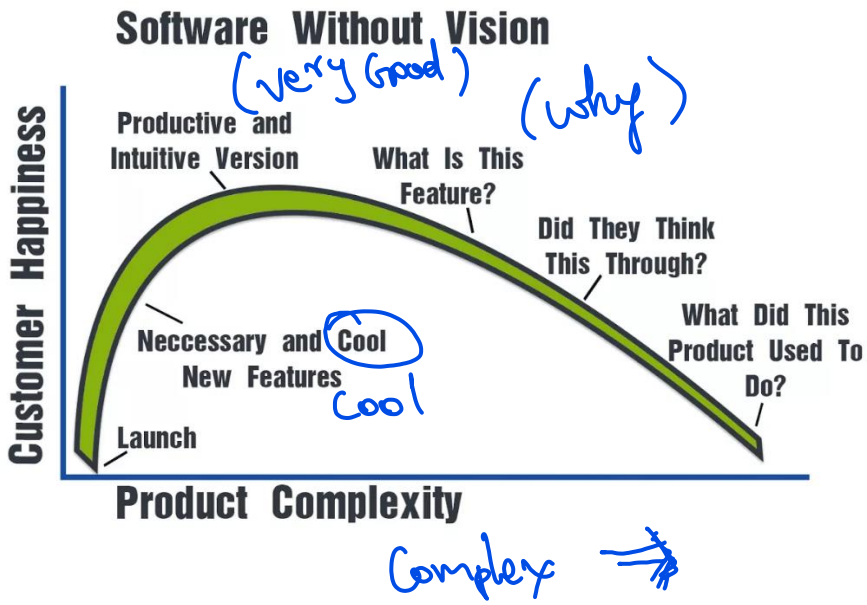
- **Interest** in the project
- Influence on the team/project (**Power**)
- Attitude (positive / negative)
- Reasons for attitude

See also:
http://www.mindtools.com/pages/article/newPPM_07.htm

Make sure to focus on what is **really** necessary!

Functionality should only be added if identified to help solving tasks

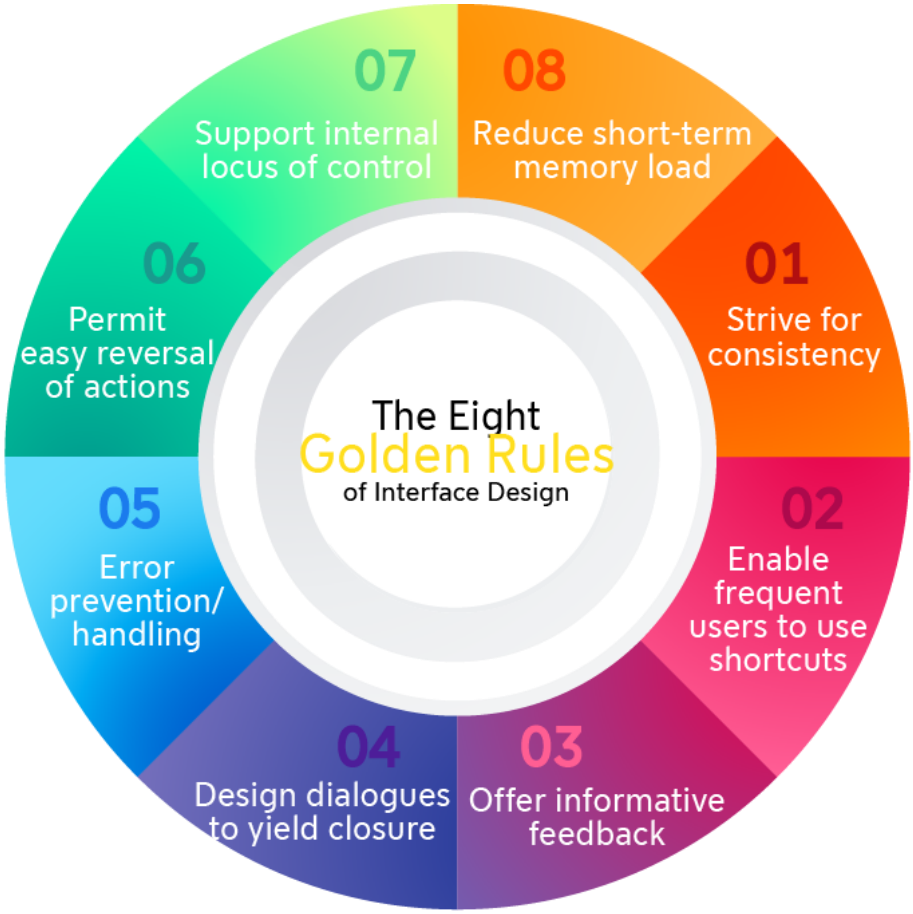
Temptation: If additional functionality is cheap to include it is often done – this can seriously compromise the user interface concept (and potentially the whole software system)!



See also: https://www.onboardmeetings.com/blog/software-with-vision/

Principle 2: Follow the Eight Golden Rules

The Eight Golden Rules have been introduced by Ben Shneiderman and summarize what needs to be considered when creating user interfaces. The Book 'Designing the User Interface: Strategies for Effective Human-Computer Interaction:' [2] is a must-read for everyone who wants to learn more about the foundation of Human-Computer Interaction and how to apply it to everyday work.



01: Strive for consistency

Consistency means using the same design patterns and the same sequences of action for similar situations – this includes a color scheme, typography, and terminology.

Consistency plays an important role by helping users become familiar with the digital landscape of your product so they can achieve their goals more easily.

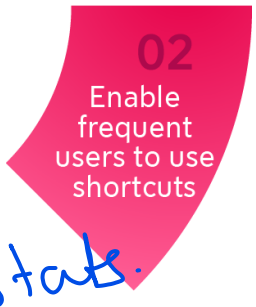
In a specific environment it is defined by guidelines (e.g., for GNOME, for KDE, for Mac OSX, for Win XP, for JAVA Swing).

Note: In the WWW it gets pretty hard:

- No real guidelines and no authority
- How are links represented?
- Where is the navigation?
- Styles and "fashion"
- change quickly...

Consistency is divided into the following levels:

- **Lexical** ("is found in the dictionary"), which means for example coding consistent with common usage (e.g., red = bad, green = good), using consistent abbreviation rules or character delete key is always the same
- **Syntactic** ("is spelled correctly"), which means that for example error messages are placed at the same (logical) place, commands are all given either first or last or, menu items can be found at the same place
- **Semantic** ("means the same"), which includes those global commands that are always available (Help, Abort, Undo), operations are valid on all reasonable objects
  Note: Semantic consistency is applicable to command line, user interfaces, to keyboard short cuts, to speech interfaces, to tool bars, to menus, to selection operation, to gestures



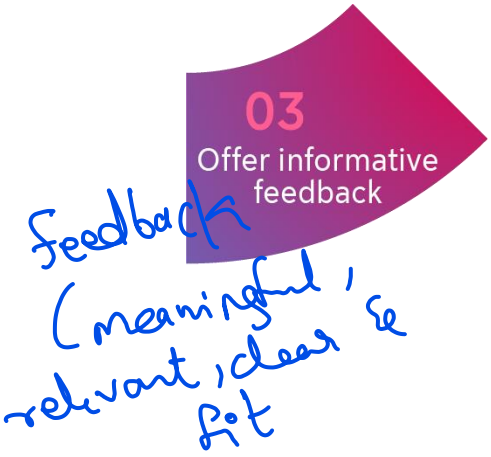
02: Enable frequent users to use shortcuts

Shortcuts will be especially beneficial for tasks, that need to be completed very frequently.

Expert users might find the following features helpful:

- Abbreviations
- Function keys
- Hidden commands
- Macro facilities

However, do not forget the novice users and give explanations or more information to the functionalities (e.g., help menu for shortcuts)

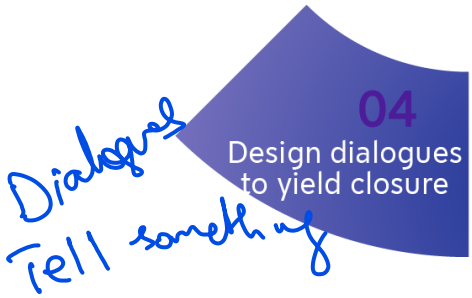


03: Offer informative feedback

Users need to be informed of what is happening at **every** stage of the process. Make sure, that the feedback is meaningful, relevant, clear, and fit the context.

For frequent actions it should be modest, peripheral For infrequent action it should be more substantial

A good example of applying this would be to indicate to the user where they are at in the process when working through a multi-page questionnaire. A bad example we often see is when an error message shows an error-code instead of a human-readable and meaningful message.



04: Design dialogues to yield closure

Just like a good story, sequences of action need to have a beginning, middle and end. Don't keep your users wondering! Tell them the outcome of their actions and if the task has been completed.

This feedback should be considered in all possible levels:

E.g.: in the large: Web shop - it should be clear when I am in the shop, and when I have successfully check-out
Or: cash dispenser - forget your bank card?
Or: in the small: a progress bar shows the status of the action

05: Error prevention/handling

A good interface needs to be designed to avoid errors as much as possible. But when errors do happen, your system needs to make it easy for the user to understand the issue and know how to solve it. Simple ways to handle errors include displaying clear error notifications along with descriptive hints to solve the problem.

Murphy's law:
'Anything that can go wrong, will go wrong.' [3]

Thus, you need to consider:

- If someone can interpret the interface wrong, they will
- If something can be used incorrectly, eventually it will
- If something is designed ideal user, then it's not designed for users

You are the expert of your system; therefore, you know much more than your users and some components might seem trivial. They are not.

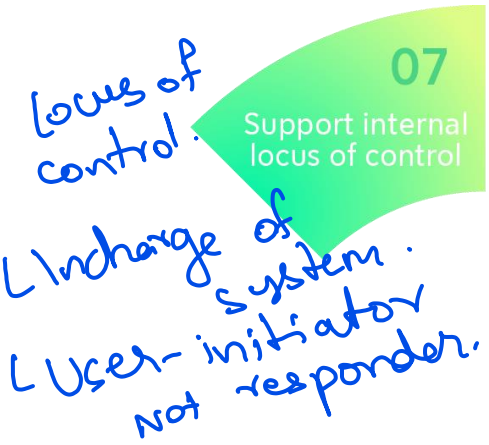
06: Permit easy reversal of actions

Users need to have an 'undo' option after a mistake is made. This will permit to feel less anxious and more likely to explore options if they know there's an easy way to reverse any accidents.

This rule can be applied to any action, group of actions, or data entry. It can range from a simple button to a whole history of actions.

Note: Undo is not trivial if user is not going sequential
→ E.g., write a text, copy it into the clipboard, undo the writing. The text is still in the clipboard!

In certain settings processes and basic physical laws prevent reversal of actions. Here an interaction layer (buffering user interaction) may be possible – but not always (e.g., breaks, emergency stop)

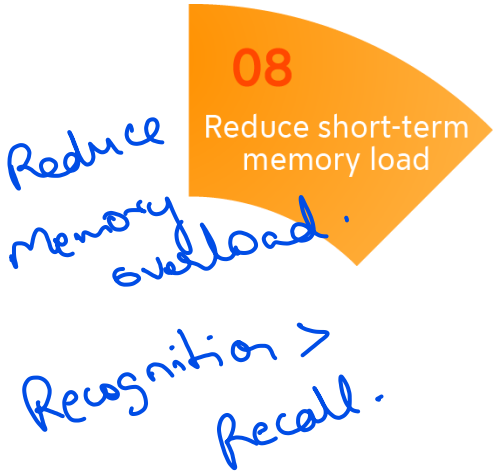


07: Support internal locus of control

"In personality psychology, locus of control is the degree to which people believe that they have control over the outcome of events" [4]

Users need to feel in charge of the system, not the other way round. Avoid surprises, interruptions, or anything that hasn't be prompted by the users.

The elements on an UI need to show, that users are the initiators of the actions rather than the responders.



08: Reduce short-term memory load

Human attention is limited, and we are only capable of maintaining around five items in our short-term memory at one time.

As Nielsen already stated:

Recognition is easier than recall. [5]

So, if you keep your interfaces simple and consistent, obeying to patterns, standards, and conventions, you are already contributing to better recognition and ease of use.

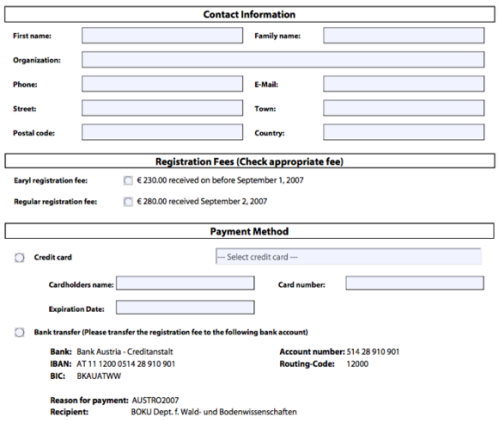
Like all rules of thumb or principles, the eight golden rules of interface design help with decisions and provide a basis for argumentation. The art lies in interpreting them to the problem at hand. And it is always particularly interesting when you must weigh up individual points against each other.

Principle 3: Prevent Errors

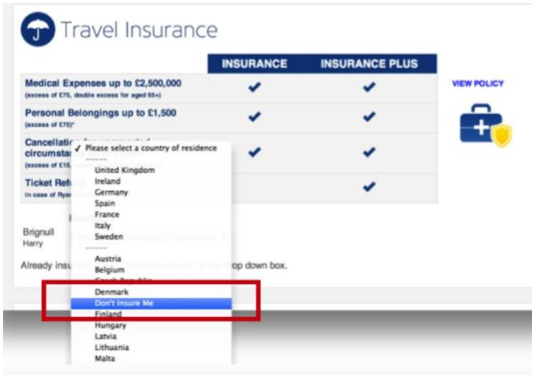
> *"Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action".*
> From: Nielsen's usability heuristics[5]

Essentially, it involves alerting a user when they're making an error, with the intention to make it easy for them to do whatever it is they are doing without making a mistake. The main reason this principle of error prevention is important is that we humans are prone to- and will always make mistakes. Reasons for mistake might differ from misinterpretations of the system, cognitive overload and lacking attention, distraction, intentional manipulation of the system and so forth.

A good user interface design should prevent and limit the possible user errors and take according action.

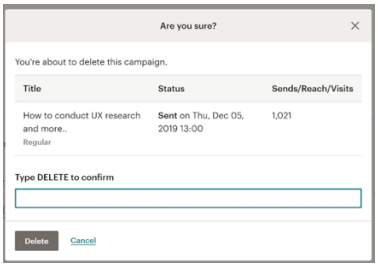


People can and will type in different formats of the information, e.g., prefix of phone number, date format, etc.



The choices in a dropdown menu should strive for consistency in their options and give clear communication about possible consequences.

Always give clear communication on the user's options and next possible actions. There is a lack of information about what will happen if the user clicks "Done".



Good example if important data is to be deleted. To prevent user's from deleting something by accident, a confirmation step is included in the dialog.

Human Error may also be a starting point to look for design problems
→ Design implications

- Assume all possible errors will be made
- Minimize the chance to make errors (constraints)
- Minimize the effect that errors have (is difficult!)
- Include mechanism to detect errors
- Attempt to make actions reversible
- Forcing function (interlock, lockins, lockouts)

Errors are routinely made. Communication and language are used between people to clarify – more often than one imagines. Common understanding of goals and intentions between people helps to overcome errors.

Users are often distracted from the task at hand, so prevent unconscious errors by offering suggestions, utilizing constraints, and being flexible.

There are two fundamental categories of errors:

1. **Mistakes:** are made when users have goals that are inappropriate for the current problem or task; even if they take the right steps to complete their goals, the steps will result in an error.
2. **Slips:** occur when users intend to perform one action but end up doing another (often similar) action. For example, typing an "i" instead of an "o" count as a slip; accidentally putting liquid hand soap on one's toothbrush instead of toothpaste is also a slip. Slips are typically made when users are on autopilot, and when they do not fully devote their attention resources to the task at hand.

What are the types of Slips users can make?

Capture errors	Two actions with common start point, the more familiar one captures the unusual (driving to work on Saturday instead of the supermarket)
Description errors	Performing an action that is close to the action that one wanted to perform (putting the cutlery in the bin instead of the sink)
Data driven errors	When users return to the design after a period of not using it, how easily can they reestablish proficiency?

Associate action errors	You think of something and that influences your action. (e.g. saying come in after picking up the phone)
Loss-of-Activation error ~ forgetting	In each environment you decided to do something but when leaving then you forgot what you wanted to do. Going back to the start place you remember.
Mode error	You forget that you are in a mode that does not allow a certain action or where an action has a different effect

If something goes wrong, we attempt corrections on the lowest level

References

1. Cooper A. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*. 1 edition ed. Sams - Pearson Education; 1999.

2. Shneiderman B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 3rd ed. Addison-Wesley Longman Publishing Co., Inc.; 1997.

3. Roe A. The Making of a Scientist. :127.

4. Rotter JB. Generalized expectancies for internal versus external control of reinforcement. *Psychol Monogr Gen Appl*. 1966;80(1):1-28. doi:10.1037/h0092976

5. Nielsen J, Molich R. Heuristic evaluation of user interfaces. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Empowering People - CHI '90*. ACM Press; 1990:249-256. doi:10.1145/97243.97281

Further ressources for information:

MacKenzie, I. S., Sellen, A., & Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. Proceedings of the CHI `91 Conference on Human Factors in Computing Systems, pp. 161-166. New York: ACM.

From: John, Bonnie and Kieras, David E., The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast, ACM Transactions on Computer-Human Interaction 3,4 (December 1996b), 320-351

Harold Thimbleby. User Interface Design With Matrix Algebra. ACM Transactions on Computer-Human Interaction, Vol. 11, No. 2, June 2004, Pages 181–236

Card, S. K., Moran, T. P., and Newell, A. 1980. The keystroke-level model for user performance time with interactive systems. Commun. ACM 23, 7 (Jul. 1980), 396-410.

Kay, A. User Interface: A Personal View. In Brenda Laurel (ed.), The Art of Human-Computer Interface Design. New York: Addision-Wesley, 1990, pp.191-207. Cited accroding to Virtual Reality and Abstract Data: Virtualizing Information. By Michael B. Spring and Michael C. Jennings. University of Pittsburgh. (http://www2.sis.pitt.edu/~spring/papers/abstdat_vr.pdf)

Urban Stress: Experiments on Noise and Social Stressors. DC Glass, JE Singer - 1972 - Academic Press Wilfred J. Hansen, User Engineering Principles for Interactive Systems, 1971 Jef Raskin, The Humane Interface, ACM Press 2000