12 Design Rules

12.1 Introduction

Design rules (or usability rules) are rules that a designer can follow in order to increase the usability of the system/product e.g., principles, standards, guidelines.
For the principles:

- Abstract and have high generality and low in authority
- Widely applicable and enduring

For the guidelines:

- Narrowly focused
- Can be too specific, incomplete and hard to apply but they are more general and lower in authority than Standards (like ISO and so on)

Design rules should be used early in the lifecycle. We will first look at abstract principles for supporting usability and later on, we will look at the most well used and well known sets of heuristics of golden rules, which tend to provide a succinct summary of the essential characteristics of good design.

12.2 Usability Principles

Usability principles:

- Learnability: the ease with which new users can begin effective interaction and achieve maximal performance.
  - Predictability: support for the user to determine the effect of future action based on past interaction history.
  - Synthesizability: support for the user to assess the effect of past operations on the currentstate.
  - Familiarity: the extent to which a user's knowledge and experience in other real-world or computer-based domains can be applied when interacting with a new system
  - Generalizability: support for the user to extend knowledge of specific interaction within and across applications to other similar situations.
  - Consistency: likeness in input-output behavior arising from similar situations or similar task objectives.
- Flexibility: the multiplicity of ways the user and system exchange information
  - Dialogue initiative: user freedom for artificial constraints on the inpur dialog imposed by the system.
  - Multithreading: the ability of the system to support user interaction for more than one task at a time.

  - Task migratability: the ability to transfer control for execution of tasks between the system and the user.
  - Substitutivity: the extent to which an application allows equivalent input and output values to be substituted for each other.
  - Customizability: the ability of the user or the system to modify the user interface.

- Robustness: the level of support provided to the user in determining successful achievement and assessment of goal-directed behavior.
  - Observability: the extent to which the user can evaluate the internal state of the system from the representation on the user interface.
  - Recoverability: the extent to which the user can reach the intended goal after recognizing an error in the previous interaction.
  - Responsiveness: a measure of the rate of communication between the user and the system.
  - Task conformance: the extent to which the system services support all the tasks the user would wish to perform and in the way the user would wish to perform.

12.3 Heuristics and Golden Rules

Here we list Jakob Nielsen's 10 Usability Heuristics:

1. Visibility of system status: the system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. Match between system and the real world: the system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom: users often choose system functions by mistake and will need a clearly marked ”emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. Consistency and standards: users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention: even better than good error messages is a careful design which prevents a problem from occurring in the first place.

6. Recognition rather than recall: make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use: accelerators –unseen by the novice user– may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design: dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors: error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation: even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Here instead, we list Ben Shneiderman's 8 Golden Rules:

1. Strive for consistency: layout, terminology, command usage, etc.

2. Cater for universal usability: recognize the requirements of diverse users and technology. For instance add features for novices eg explanations, support expert users eg shortcuts.

3. Offer informative feedback: for every user action, offer relevant feedback and information, keep the user appropriately informed, human-computer interaction.

4. Design dialogs to yield closure: help the user know when they have completed a task.

5. Offer error prevention and simple error handling: prevention and (clear and informative guidance to) recovery; error management.

6. Permit easy reversal of actions: to relieve anxiety and encourage exploration, because the user knows s/he can always go back to previous states.

7. Support internal locus of control: make the user feel that s/he is in control of the system, which reponds to his/her instructions/commands.

8. Reduce short-term memory load: make menus and UI elements/items visible, easily available/retrievable, ...

Here we list Norman's 7 principles:

1. Use both knowledge in the world and knowledge in the head.

2. Simplify the structure of tasks.

3. Make things visible: bridge the gulfs of Execution and Evaluation.

4. Get the mappings right.

5. Exploit the power of constraints, both natural and artificial.

6. Design for error.

7. When all else fails, standardize.

13 Evaluation Techniques

Evaluation tests usability and functionality of a system. It occurs in laboratory, field and/or in collaboration with users. Evaluation is done on both design and implementation and should be considered at all stages in the design lifecycle. The goals of evaluation are:

- assess extent of system functionality
- assess effect of interface on user
- identify specific problems

13.1 Evaluating Designs

There are three main evaluation techniques that we will examine in this chapter.

13.1.1 Cognitive Walkthrough

It is a technique that evaluates design on how well it supports user in learning task. It is usually performed by expert in cognitive psychology, expert that walks through design to identify potential problems using psychological principles. For each task, walkthrough considers:

- what impact will interaction have on user?
- what cognitive processes are required?
- what learning problems may occur?

13.1.2 Heuristic Evaluation

Usability criteria (heuristics) are identified and design is examined by experts to see if these are violated. Heuristic evaluation debugs design highlighting the problems we should solve to achieve better results. The list of criteria is well defined and standardized.

13.1.3 Review-based evaluation

Results from the literature used to support or refuse parts of design. Care is needed to ensure results are transferable to new design. This kind of evaluation is model-based, in fact cognitive models are used to filter design options.

13.2 Evaluating through user participation

From the title we can easily understand what kind of evaluation is done.

13.2.1 Laboratory Studies

Advantages are:

- specialist equipment available
- uninterrupted environment

Disadvantages:

- lack of content
- difficult to observe several users cooperating

It is an appropriate method if system location is dangerous or impractical for constrained single user systems to allow controlled manipulation of use.

13.2.2 Field Studies

Advantages are:

- natural environment
- context retained (though observation may alter it)
- longitudinal studies possible

Disadvantages:

- distractions
- noise

It is an appropriate method where context is crucial for longitudinal studies.

13.3 Evaluating Implementations

In order to evaluate implementations, we need prototypes of an app, or the whole app itself. *Experimental evaluation* is a controlled evaluation of specific aspects of interactive behavior. Evaluator chooses hypothesis to be tested and a number of experimenta conditions are considered which differ only in the value of some controlled variable. Let's see experimental factors:

- **Subjects**
- **Variables**, things to modify and measure
- **Hypothesis**, what you'd like to show
- **Experimental design**, how you are going to do it

Variables are divided in IV (independent), like interface style, number of menu items and DV, like time taken and number of errors.
Hypothesis is the prediction of an outcome framed in terms of variables, e.g. "error rate will increase as font size decreases". Null hypothesis states no difference between conditions and the aim is to disprove this.
Experimental design can be within groups (each subject performing experiments under each condition, transfer of learning possible) and between groups (each subject performs under only one condition, no transfer of learning).
Analysis of data is something we always have to do. We have to look at data and save the original data before doing any kind of statistics. The choice of statistical technique depends on type of data (discrete or continuous) and information required. Tests can be parametric (assume normal distribution of data, robust, powerful) or non-parametric (do not assume normal distribution, less powerful, more reliable). What information is required? Is there a difference? How big is it? How accurate is the estimate? Parametric and non-parametric tests can answer first of these questions.
Experimental studies on groups are more difficult than single-user experiments. In fact, we have problems with subject groups, choice of task, data gathering and analysis.

- Subject groups, larger number of subjects and longer time to settle down, difficult to timetable
- The task must encourage cooperation and perhaps involve multiple channels
- Data gathering is done through several video cameras. Problems are synchronization and sheer volme. A solution could be recording from each perspective.
- Analysis is useful because there is a vast variation between groups. Solutions can be within groups experiments, micro-analysis and anecdotal and qualitative analysis.

13.4 Observational methods

We will now analyze different types of observational methods.

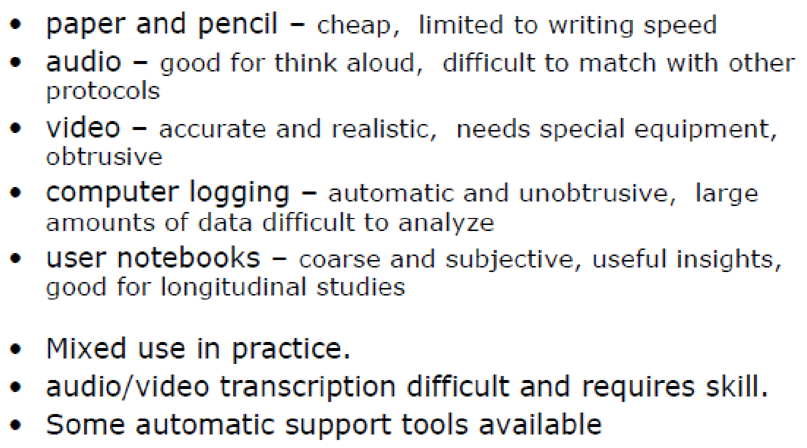
13.4.1 Think Aloud

Think aloud happens when user is observed while performing a task. User is asked to describe what he is doing and why, what the think is happening etc. Advantages are simplicity, providing of useful insights and showing how system is actually used. Disadvantages consist of subjectivity, selectivity and act of describing that may alter performance.

13.4.2 Cooperative evaluation

This is a variation of think aloud. User collaborates in evaluation and both user and evaluator can ask each other questions throughout. Additional advantages are that it is less constrained and easier to use, user is encouraged to criticize system and clarification is possible.

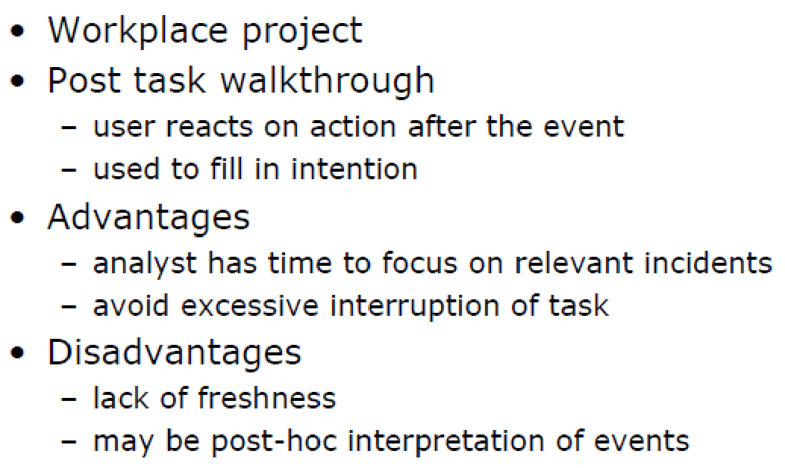
13.4.3 Protocol analysis



- paper and pencil – cheap, limited to writing speed
- audio – good for think aloud, difficult to match with other protocols
- video – accurate and realistic, needs special equipment, obtrusive
- computer logging – automatic and unobtrusive, large amounts of data difficult to analyze
- user notebooks – coarse and subjective, useful insights, good for longitudinal studies

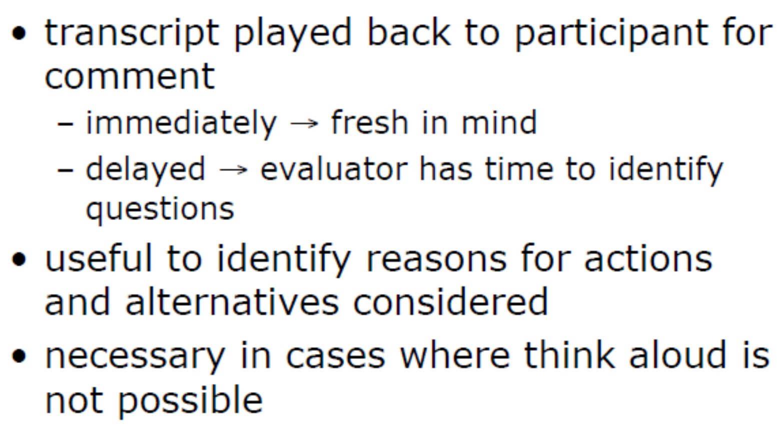
- Mixed use in practice.
- audio/video transcription difficult and requires skill.
- Some automatic support tools available

13.4.4 Automated analysis



- Workplace project
- Post task walkthrough
  - user reacts on action after the event
  - used to fill in intention
- Advantages
  - analyst has time to focus on relevant incidents
  - avoid excessive interruption of task
- Disadvantages
  - lack of freshness
  - may be post-hoc interpretation of events

13.4.5 Post-task walkthroughs



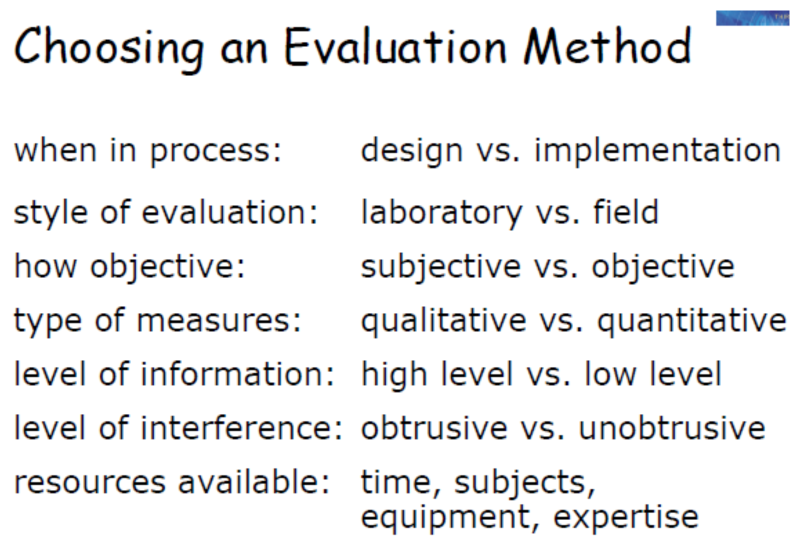
- transcript played back to participant for comment
  - immediately → fresh in mind
  - delayed → evaluator has time to identify questions
- useful to identify reasons for actions and alternatives considered
- necessary in cases where think aloud is not possible

13.5 Query Techniques

Query techniques are used to gain information from possible users of the app. *Interviews* happen when analyst questions user on one-to-one basis usually based on prepared questions. It is informal, subjective and relatively cheap. Advantages include that it can be varied to suit context, issues can be explored more fully and it can elicit user views and identify unanticipated problems. *Questionnaires* are a set of fixed questions given to user and advantages are that they are quick and reaches large user groups and can be analyzed more rigorously. Disadvantages are less flexibility and less probing.

13.6 Physiological methods

Eye tracking is one of such methods where head or desk mounted equipment tracks the position of the eye. Eye movement reflects the amount of cognitive processing a display requires. Measurement include fixations (eye maintains stable position, number and duration indicate level of difficulty with display), saccades (rapid eye movement from one point of interest to another) and scan paths (moving straight to a target with a short fixation at the target is optimal).
Physiological measurement is an emotional response linked to physical changes. These may help determine a user's reaction to an interface. Measurement include heart activity, swat glands activity, muscle and brain electrical activity... There is some difficulty in interpreting this data, research is required.



Choosing an Evaluation Method

when in process:	design vs. implementation
style of evaluation:	laboratory vs. field
how objective:	subjective vs. objective
type of measures:	qualitative vs. quantitative
level of information:	high level vs. low level
level of interference:	obtrusive vs. unobtrusive
resources available:	time, subjects, equipment, expertise

14 Measuring learnability

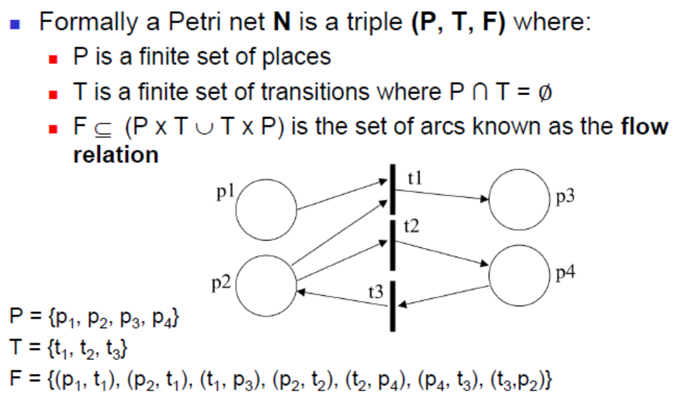
Usability assesses how easy UIs are to use. ISO defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use". It consists of five criteria, but we will examine learnability only.
Learnability is the capability of the software product to enable the user to learn its application. However, there is no consistent agreement on how the concept of learnability should be defined. Some common features that are related to it can be identified:

- learnability is a function of user's experience it depends on the type of user for which the learning occurs (experience vs novice users)
- learnability can be evaluated on a single usage period initial learnability or after several usages (extended learnability)
- learnability measures the performance of a user interaction in terms of completion times, error and success rates or percentage of functionality understood

This lack of consensus has naturally led to a lack of well accepted metrics for measuring learnability. Some metrics exist, but they are limited to measure specific aspects of the interaction. Qualitative, mental and quantitative metrics are the ones we just said. All these measurements are performed in controlled lab environments under the guidance of an external evaluator. However, these measurements are good for initial learnability only. Evaluating extended learnability traditionally requires expensive and time-consuming techniques for observing users over an extended period of time.
In order to objectively quantify the extended learnability of interactive systems during their daily use, we define an interaction model that represents the expected way of performing the task (Petri nets), then we record the user's observed behavior during the interaction with the system in a specific user log, and then we introduce the concept of alignment to check if the observed behavior as recorded in the user log matches the expected behavior as represented in the interaction model.

14.1 Petri nets

A *Petri net* takes the form of a directed bipartite graph where the nodes are either places (circles) or transitions (squares or vertical lines). Arcs connect these states.

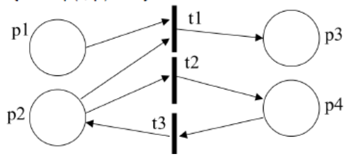


- Formally a Petri net **N** is a triple **(P, T, F)** where:
  - P is a finite set of places
  - T is a finite set of transitions where $P \cap T = \emptyset$
  - $F \subseteq (P \times T \cup T \times P)$ is the set of arcs known as the **flow relation**

$P = \{p_1, p_2, p_3, p_4\}$
$T = \{t_1, t_2, t_3\}$
$F = \{(p_1, t_1), (p_2, t_1), (t_1, p_3), (p_2, t_2), (t_2, p_4), (p_4, t_3), (t_3, p_2)\}$

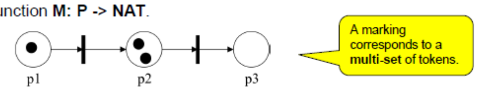
- A directed arc from a place **p** to a transition **t** indicates that **p** is an **input place** of **t**. Formally:
  - $\bullet t = \{p \in P \mid (p, t) \in F\}$
- A directed arc from a transition **t** to a place **p** indicates that **p** is an **output place** of **t**. Formally:
  - $t \bullet = \{p \in P \mid (t, p) \in F\}$
- With an analogous meaning, we can define **input/output transitions** of a place. Formally:
  - $p \bullet = \{t \in T \mid (p, t) \in F\}$ and $\bullet p = \{t \in T \mid (t, p) \in F\}$

$t_1 \bullet = \{p_3\}$; $\bullet t_1 = \{p_1, p_2\}$;
$\bullet p_2 = \{t_3\}$; $\bullet p_1 = \varnothing$;
$p_2 \bullet = \{t_1, t_2\}$; $p_1 \bullet = \{t_1\}$



Marking

- The operational semantics of a Petri Net N = (P,T,F) is described in terms of particular marks called **tokens** (represented as black dots).
- Places in Petri Nets can contain <u>any number of tokens</u>.
- Any distribution of tokens across all of the places is called a **marking**.
  - A marking is a function **M: P -> NAT**.

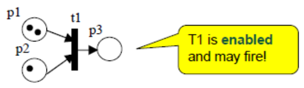


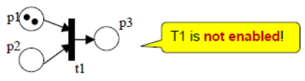
- The marking of a Petri net determines its **state**.
  - State of the example Petri net: $M = \{(p_1,1),(p_2,2),(p_3,0)\}$ or $M = [p_1,p_2^2]$.
- Petri nets must be associated with an **initial marking M_0** and with a set of possible **final markings**.

Firing Rules

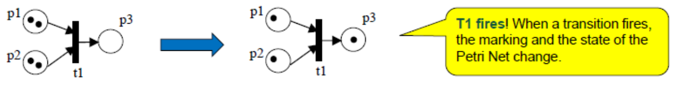
- The dynamic behavior of Petri nets is characterized by the notion of **firing** (*transition execution*). A transition can **fire** whenever there are one or more tokens in each of its input places.

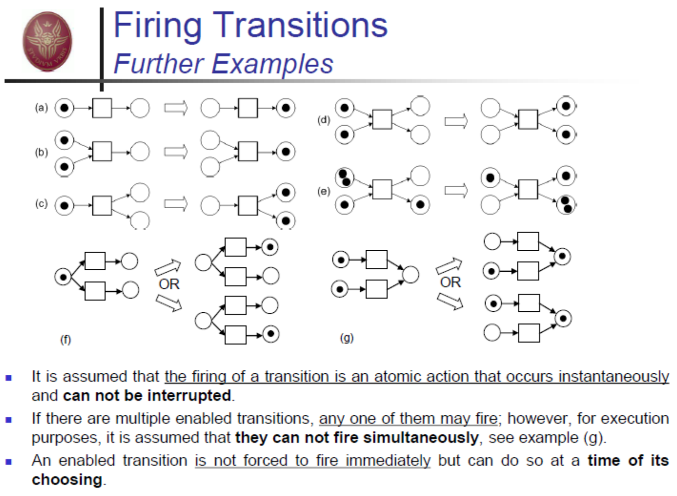
1. A transition t is said to be **enabled** if and only if each input place p of t contains at least one token. <u>Only enabled transitions may fire</u>.
   - A transition t is enabled in a marking M iff for each p, with p ∈•t, M(p) > 0.





2. If transition t fires, then t **consumes one token** from each input place p of t and **produces one token** for each output place p of t.

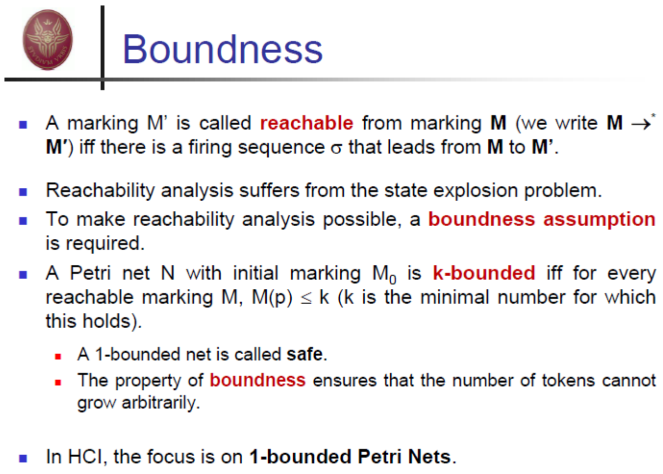

p1
p3
p2
t1
p1
p3
p2
t1
T1 fires! When a transition fires, the marking and the state of the Petri Net change.


(a)
(b)
(c)
(d)
(e)
(f)
(g)
OR
OR

Firing Transitions

Further Examples

- It is assumed that the firing of a transition is an atomic action that occurs instantaneously and **can not be interrupted**.
- If there are multiple enabled transitions, any one of them may fire; however, for execution purposes, it is assumed that **they can not fire simultaneously**, see example (g).
- An enabled transition is not forced to fire immediately but can do so at a **time of its choosing**.



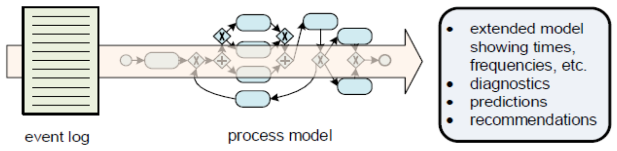
Boundness

- A marking M' is called **reachable** from marking **M** (we write $\mathbf{M} \rightarrow^{*} \mathbf{M'}$) iff there is a firing sequence σ that leads from **M** to **M'**.
- Reachability analysis suffers from the state explosion problem.
- To make reachability analysis possible, a **boundness assumption** is required.
- A Petri net N with initial marking M_0 is **k-bounded** iff for every reachable marking M, $M(p) \leq k$ (k is the minimal number for which this holds).
  - A 1-bounded net is called **safe**.
  - The property of **boundness** ensures that the number of tokens cannot grow arbitrarily.
- In HCI, the focus is on **1-bounded Petri Nets**.

Interaction models like these nets are not enforced by software systems. Traces can be dirty, with redundant or missing actions. Any execution of a relevant task produces a new execution trace recorded in a user log.

Replay

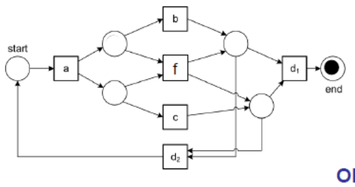
- **Replay** approaches use a ***user log*** and an ***interaction model*** (that may have been constructed by *hand* or *discovered*) as input. The user log is ***replayed*** on top of the interaction model.
- In this way, **discrepancies** between the log (observed behavior) and the model (expected behavior) can be ***detected*** and ***quantified***.



Trace alignment

- Investigate relations between **moves in the log** and **moves in the model** to establish an alignment between the model and a trace.
- **Replay**: If a move in the log cannot be mimicked by the model and vice-versa, such "no moves" are denoted by > (and may have a **cost**).

Alignment of t = <adbc>



Move in the model (only)

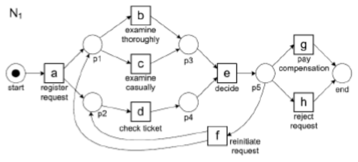
a	d	b	c	>
a	>	b	c	d

OPTIMAL ALIGNMENT

Move in the log (only)

Several possible alignments

<abdeg>



a	b	d	e	g
a	b	d	e	g

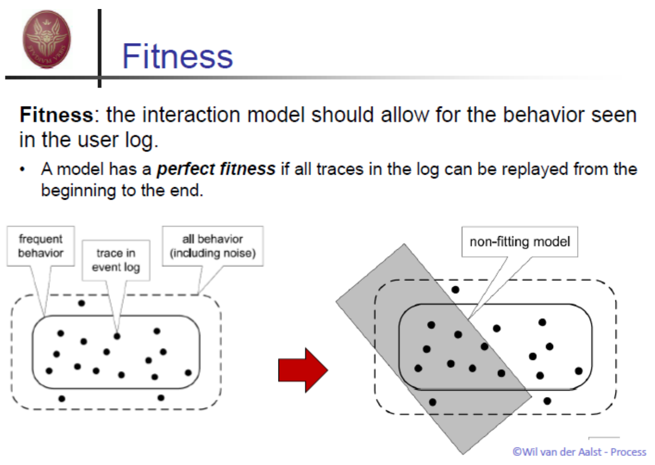
a	b	»	d	e	g
a	»	c	d	e	g

a	b	d	e	g	»	»	»	»	»
»	»	»	»	»	a	c	d	e	g

Trace alignment allows to:

- verify if a trace is compliant with an interaction model
- identify the root and the severity of each deviation

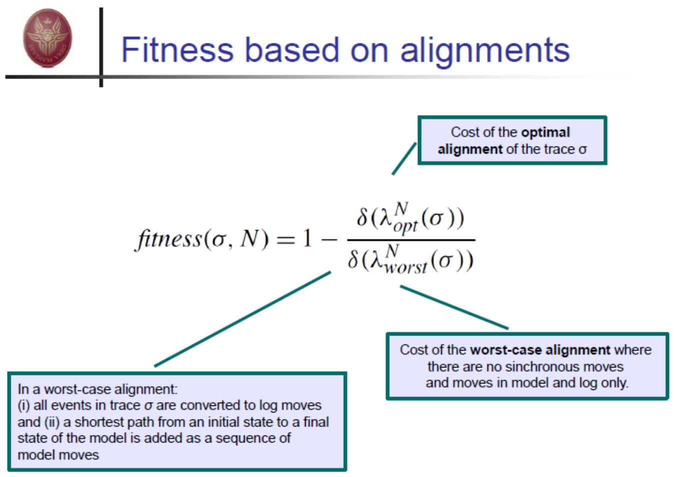
The result of the alignment is the fitness value, i.e. how much the log adheres to a model of interaction (it can vary from 0 to 1).
To measure the learnability of a system we can analyze the rate of the fitness values corresponding to subsequent executions of the system over time. An increasing rate will correspond to a system that is easy to be learnt with respect to its relevant tasks. Given an initial low fitness, a not increasing or stable rate may indicate the presence of some learning issues that need to be fixed.



Fitness

Fitness: the interaction model should allow for the behavior seen in the user log.

- A model has a ***perfect fitness*** if all traces in the log can be replayed from the beginning to the end.



Fitness based on alignments

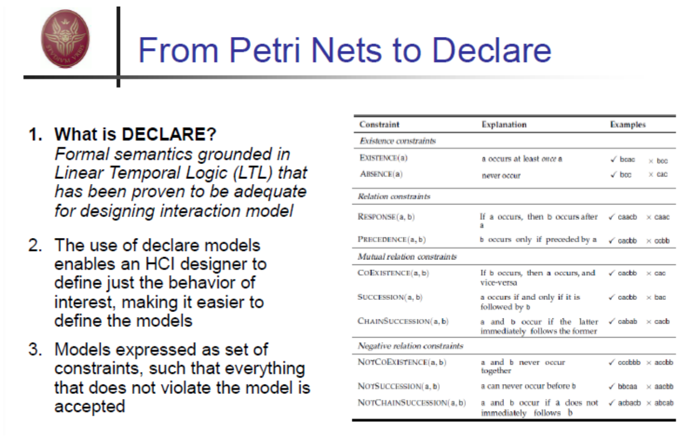
$$fitness(\sigma, N) = 1 - \frac{\delta(\lambda^N_{opt}(\sigma))}{\delta(\lambda^N_{worst}(\sigma))}$$

Cost of the **optimal alignment** of the trace σ

Cost of the **worst-case alignment** where there are no sinchronous moves and moves in model and log only.

In a worst-case alignment:
(i) all events in trace σ are converted to log moves and (ii) a shortest path from an initial state to a final state of the model is added as a sequence of model moves

14.2 DECLARE

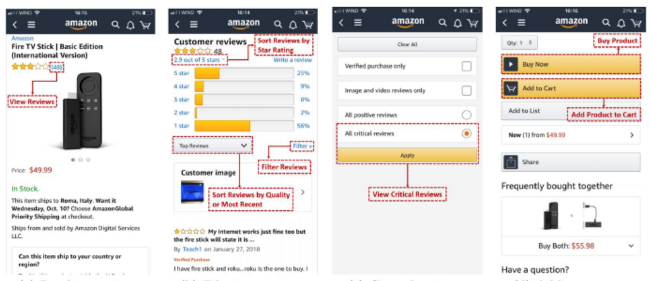


From Petri Nets to Declare

1. **What is DECLARE?**
   Formal semantics grounded in Linear Temporal Logic (LTL) that has been proven to be adequate for designing interaction model
2. The use of declare models enables an HCI designer to define just the behavior of interest, making it easier to define the models
3. Models expressed as set of constraints, such that everything that does not violate the model is accepted

Constraint	Explanation	Examples
Existence constraints		
EXISTENCE(a)	a occurs at least *once* a	✓ bcac × bcc
ABSENCE(a)	never occur	✓ bcc × cac
Relation constraints		
RESPONSE(a, b)	If a occurs, then b occurs after a	✓ caacb × caac
PRECEDENCE(a, b)	b occurs only if preceded by a	✓ cacbb × ccbb
Mutual relation constraints		
COEXISTENCE(a, b)	If b occurs, then a occurs, and vice-versa	✓ cacbb × cac
SUCCESSION(a, b)	a occurs if and only if it is followed by b	✓ cacbb × bac
CHAINSUCCESSION(a, b)	a and b occur if the latter immediately follows the former	✓ cabab × cacb
Negative relation constraints		
NOTCOEXISTENCE(a, b)	a and b never occur together	✓ cccbbb × accbb
NOTSUCCESSION(a, b)	a can never occur before b	✓ bbcaa × aacbb
NOTCHAINSUCCESSION(a, b)	a and b occur if a does not immediately follows b	✓ adbacb × abcab

Capturing user actions with Declare



(a) Reading reviews. (b) Filtering reviews. (c) Critical reviews. (d) Adding to cart.

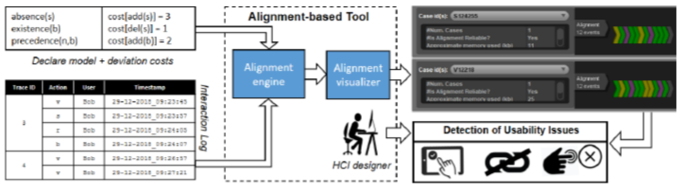
Some screenshots of the UI of Amazon shopping mobile app. In this example, we focus only on a subset of possible user actions such as the user reading only the critical reviews associated to a Fire TV stick, and then purchasing it.

Interaction models via Declare

If we consider our running example, we can specify the interaction model that describes the expected behavior underlying the relevant task, such as the user only reading the critical reviews associated to a Fire TV stick, and then proceeding to buy it, as the set consisting of the following declare constraints:

- **absence(s)** means that action s = Sort Reviews by Quality or Most Recent cannot ever be performed.
- **existence(b)** means that action b = Buy Product must be executed at a certain point of the interaction.
- **precedence(n; b)** forces n = View Critical Reviews to precede b = Buy Product.

The declarative approach [16]



Conclusion and Future Works

- Our approach couples interaction models represented as Petri nets with the notion of alignment wrt. user logs to obtain a more precise measurement of the learnability of interactive systems.
- STRENGTHS
  - The approach can be enacted while the system is used in **real user contexts**.
  - The ability of associating **different weights** to the deviations identified during an alignment.
  - For a specific system's task, **different interaction models can be developed** to represent the different interaction strategies of novice and experienced users
- LIMITATIONS
  - Only few interactive systems provide a structured recording of user logs.
  - There should be one or more identifiable entry/exit points in the user log for extracting the specific traces associated to the task.
- FUTURE WORKS
  - Further robust longitudinal studies to be performed in longer time frames.
  - Identification of threshold values for the fitness.