Measuring the Learnability of Interactive Systems

<u>Andrea Marrella</u>, Tiziana Catarci

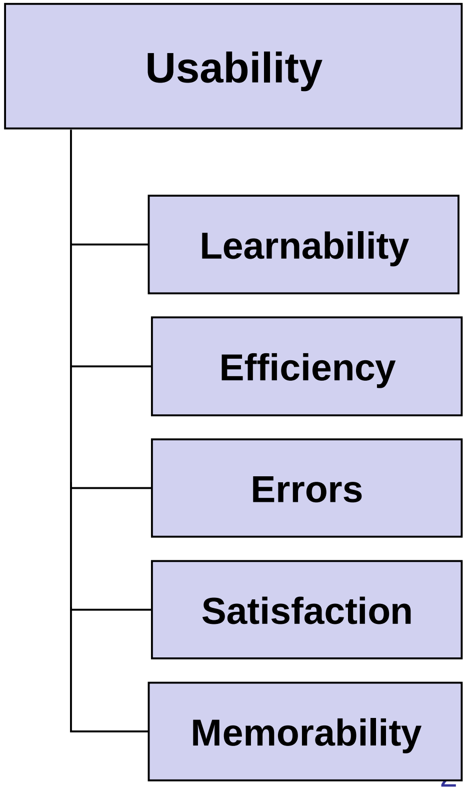
marrella@diag.uniroma1.it

18 May 2020

SAPIENZA
UNIVERSITÀ DI ROMA

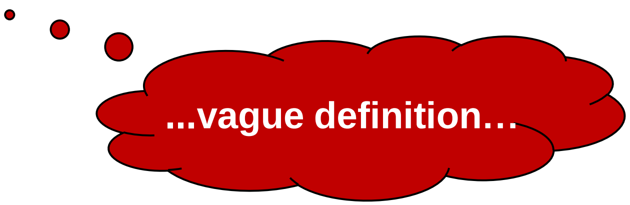
Definition of Usability

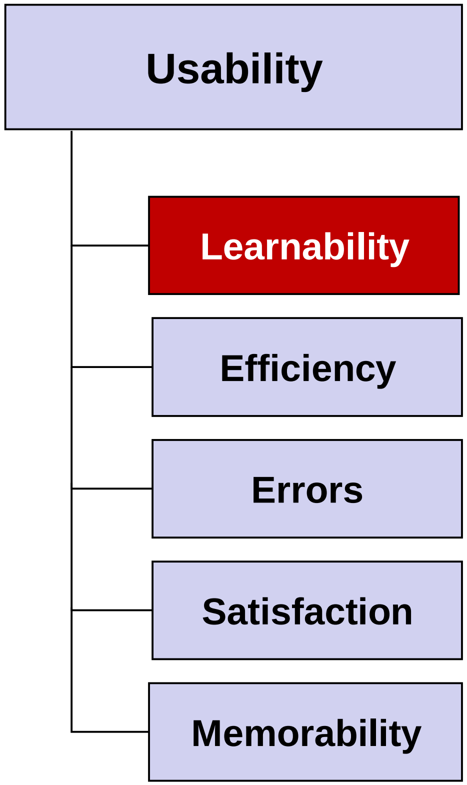
- **Usability** assesses how *easy* user interfaces are to use.
- ISO defines usability as *"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."*
- According to Nielsen [1], usability is defined by **5 quality components:**
  - **Learnability**: How easy is it for users to accomplish correctly basic tasks of a system after having executed it a few times in the past?
  - **Efficiency**: Once users have learned the design, how quickly can they perform tasks?
  - **Errors**: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
  - **Satisfaction**: How pleasant is it to use the design?
  - **Memorability**: When users return to the design after a period of not using it, how easily can they reestablish proficiency?



Learnability

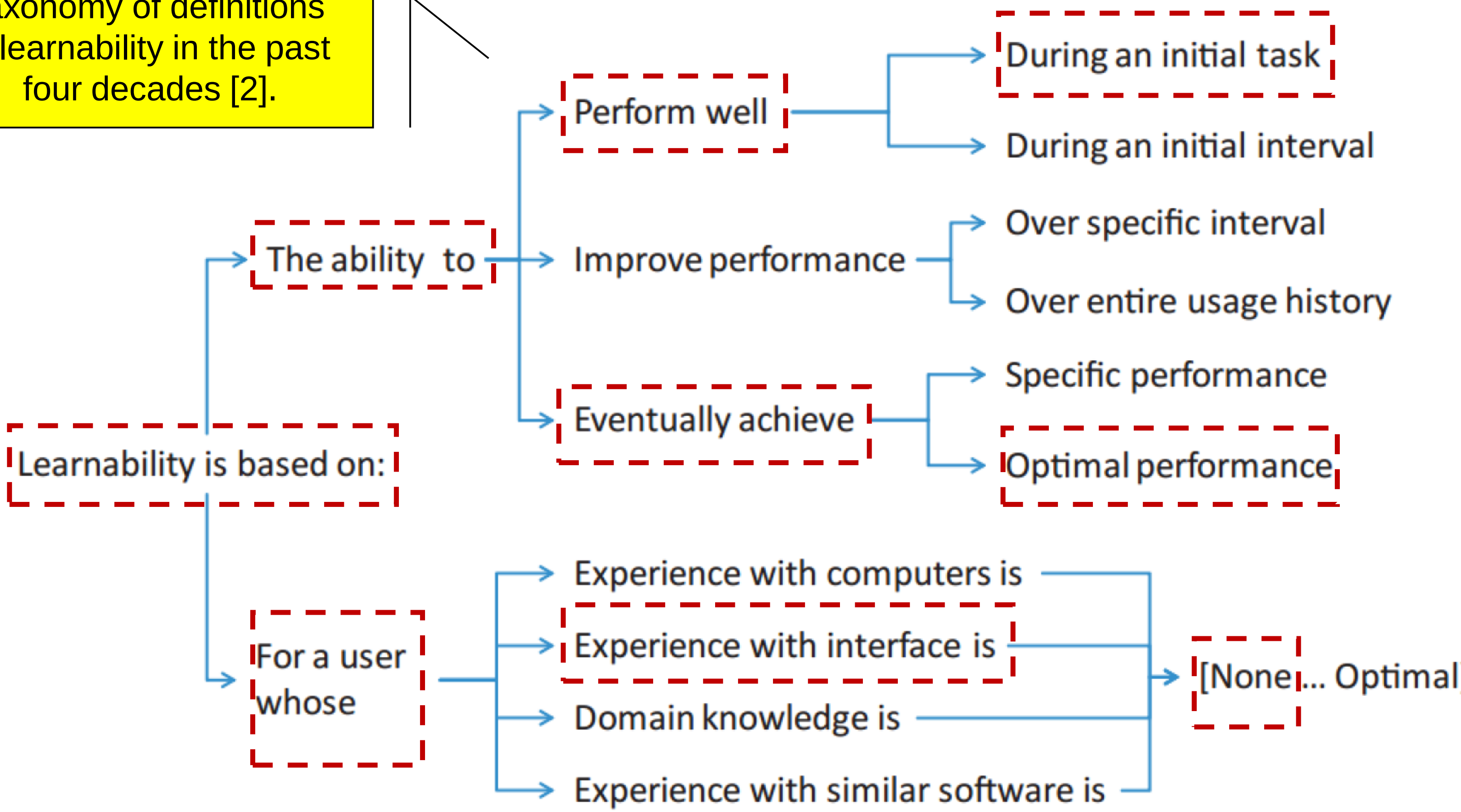
- In the HCI community, learnability is generally recognized as one of the most relevant components of usability [1].
- Learnability in ISO 9126-1
  - *“…the capability of the software product to enable the user to learn its application…”*
  - Learnability applies to the **nature of the performance change** of a user when interacting with a system to accomplish a specific task.





How to define learnability?

Taxonomy of definitions of learnability in the past four decades [2].



Dix-et-al [3] : *«Ease at which new users can begin effective interaction and achieve maximal performance»*

Some definitions of Learnability

- Nielsen advocates that an interactive system is highly learnable if it "*allows users to reach a reasonable level of usage proficiency within a short time*" [4].
- Holzinger defines learnability as "*allowing users to rapidly begin to work with the system*" [5].
- Santos recognizes learnability as "*the effort required for a typical user to be able to perform a set of tasks using an interactive system with a predefined level of proficiency*" [6].
- Shneiderman defines learnability as "*the time it takes members of the user community to learn how to use the commands relevant to a set of tasks*" [7].

The above definitions are focused on the initial learning experience of a user that interacts with a system (**initial learnability**).

Extended Learnability

Extended Learnability: The target is to measure the changes of user performance when interacting with a system.

- In 1980, Michelsen et al. provided one of the first definitions of extended learnability: "*a system should be easy to learn over time by the class of users for whom it is intended*" [8].
- Butler identifies learnability as "*user performance based on self instruction [allowing] experienced users to select an alternate model that involved fewer screens or keystrokes*" [9].
- Bevan and Macleod state that learnability is the "*quality of use for users over time*" [10].

Characteristics of Learnability

- 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.

How to measure Learnability?

- Several metrics exist, but they are limited to measure specific aspects of the interaction.
- **Qualitative metrics**
  - They measure the ***quality of the interaction*** by analyzing the user feedbacks after the interaction has happened [8,11].
- **Mental metrics**
  - They are used to understand ***which cognitive processes*** drive the user behavior during the interaction with the system [6,12].
- **Quantitative metrics**
  - They measure the ***performance of a user executing a relevant task through the system*** (e.g., completion times, error rates, etc.) [1,13,14].
- All the above measurements are performed in **controlled lab environments** under the guidance of an **external evaluator**.

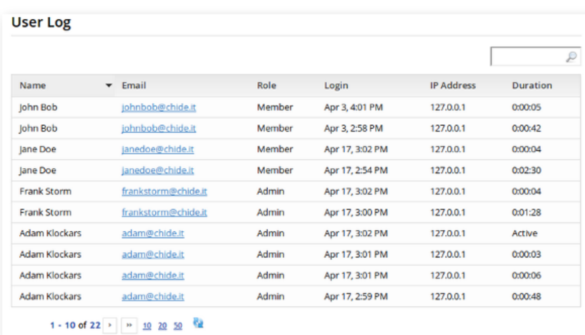
Evaluating Extended Learnability

- **Issue**: Lab measurements have been proven to be particularly suitable for measuring the initial learnability.
  - It represents (at best) a measure of the system’s intuitiveness.
  - Measuring intuitiveness is not necessarily an indicator of continued learning [1] over a longer period of use.

- Evaluating extended learnability traditionally requires **expensive** and **time-consuming techniques** for observing users over an extended period of time.
- Consequently, due to its prohibitive costs, evaluators have gradually refrained from performing extended learnability evaluation [6].

An approach to quantify learnability

- We propose an approach to **objectively quantify** the extended learnability of interactive systems **during their daily use**.
  1. For any relevant task of the system, we define an **interaction model** that represents the **expected way** of performing the task.
     - We use **Petri nets** for modelling human-computer dialogs.
  2. We record the **user's observed behavior** during the interaction with the system in a specific **user log**.
  3. 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.



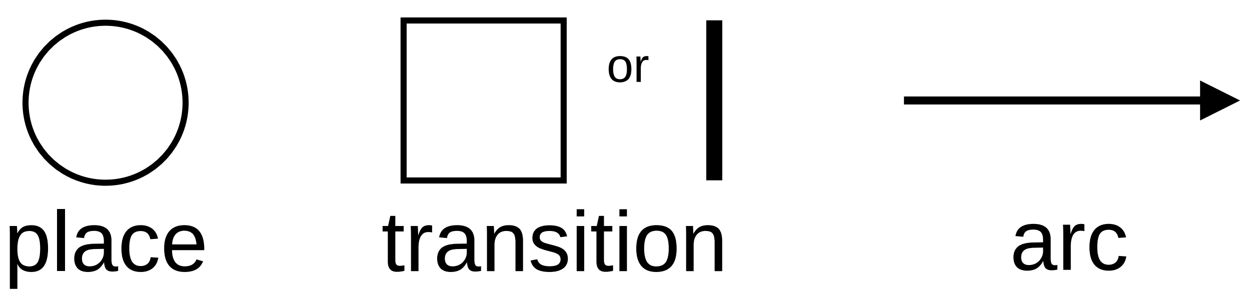
User Log

Name	Email	Role	Login	IP Address	Duration
John Bob	johnbob@chide.it	Member	Apr 3, 4:01 PM	127.0.0.1	0:00:05
John Bob	johnbob@chide.it	Member	Apr 3, 2:58 PM	127.0.0.1	0:00:42
Jane Doe	janedoe@chide.it	Member	Apr 17, 3:02 PM	127.0.0.1	0:00:04
Jane Doe	janedoe@chide.it	Member	Apr 17, 2:54 PM	127.0.0.1	0:02:30
Frank Storm	frankstorm@chide.it	Admin	Apr 17, 3:02 PM	127.0.0.1	0:00:04
Frank Storm	frankstorm@chide.it	Admin	Apr 17, 3:00 PM	127.0.0.1	0:01:28
Adam Klockars	adam@chide.it	Admin	Apr 17, 3:02 PM	127.0.0.1	Active
Adam Klockars	adam@chide.it	Admin	Apr 17, 3:01 PM	127.0.0.1	0:00:03
Adam Klockars	adam@chide.it	Admin	Apr 17, 3:01 PM	127.0.0.1	0:00:06
Adam Klockars	adam@chide.it	Admin	Apr 17, 2:59 PM	127.0.0.1	0:00:48

1 - 10 of 22 › » 10 20 50

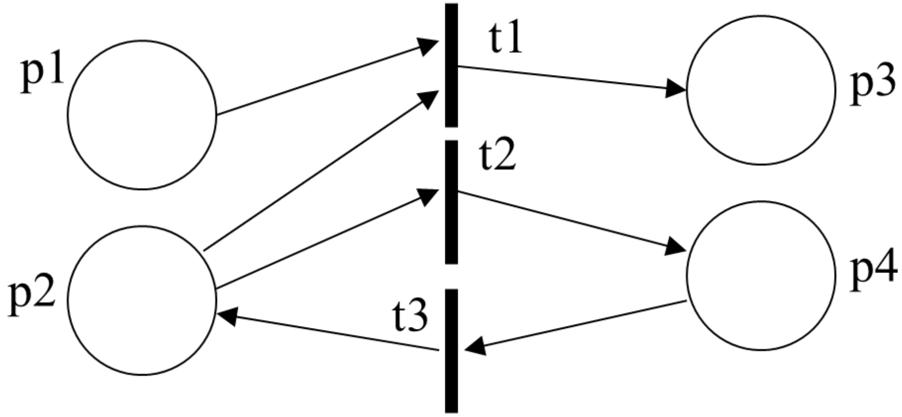
Petri Nets

- A Petri Net takes the form of a **directed bipartite graph** where the nodes are either *places* or *transitions.*
- **Places** represent **intermediate states** of a system (e.g., a *text editor*) that may exist during the interaction with it.
- Places can be input/output of **transitions**.
  - Transitions represent **user actions** (e.g., *windows opened, system commands executed, check boxes clicked, text entered/edited, etc*.) to achieve a **relevant task** (e.g., *copy and paste a document*).
- **Arcs** connect places and transitions in a way that <u>places can only be connected to transitions and vice-versa</u>.



Petri Nets: Definitions

- 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**



$P = \{p_1, p_2, p_3, p_4\}$

$T = \{t_1, t_2, t_3\}$

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

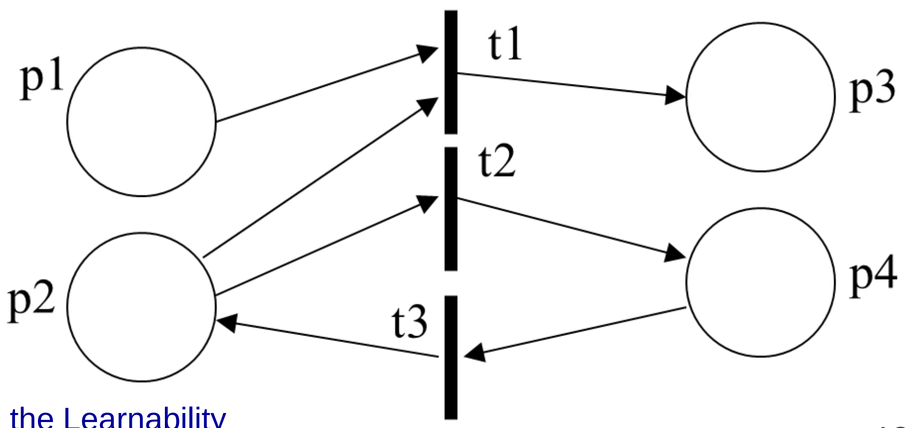
Petri Nets
Input/Output places and transitions

- 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 $\bullet\, p = \{t \in T \mid (t, p) \in F\}$

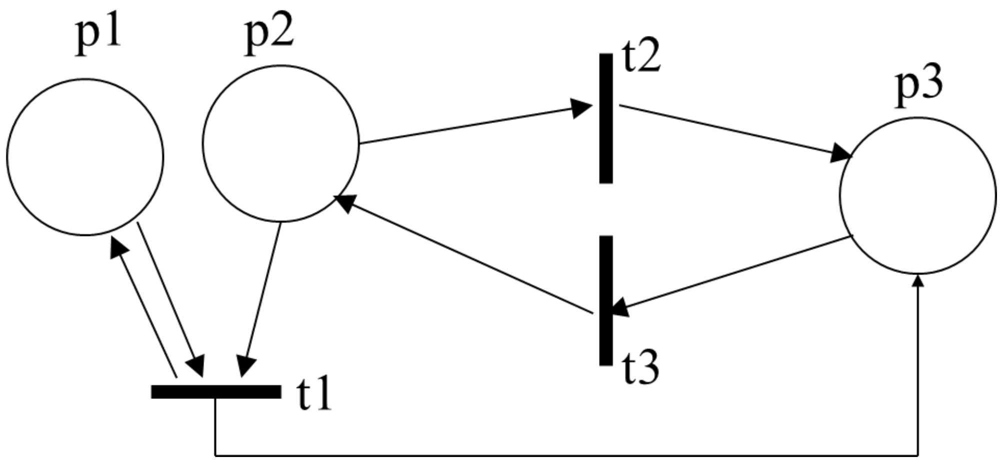
$t_1 \bullet = \{p_3\}; \bullet\, t_1 = \{p_1, p_2\};$

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

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



Exercise



P = ………………………………….

T = ………………………………….

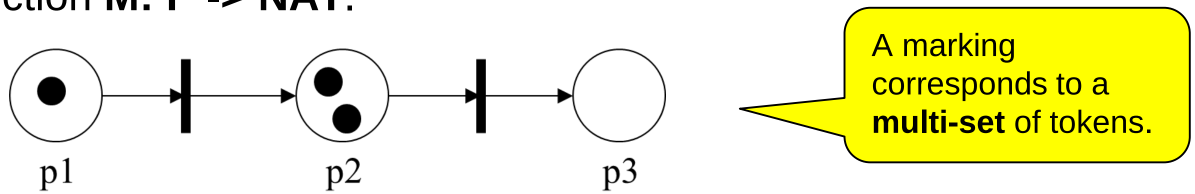
F = ………………………………….

$t_1 \bullet$ = …………………… ; $\bullet\, t_1$ = ………………………… ;

$\bullet\, p_2$ = ……………………; $p_2 \bullet$ = ………………………….;

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**.

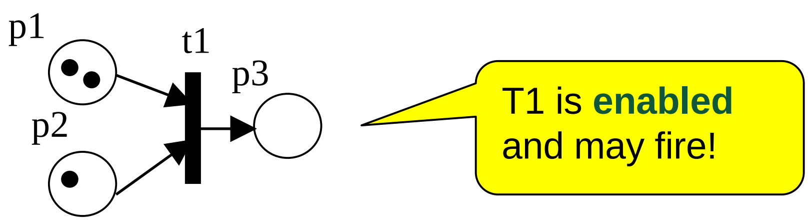


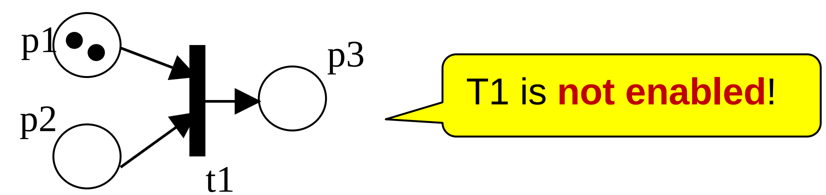
- The marking of a Petri net determines its **state**.
  - State of the example Petri net: $M = \{(p_1,1),(p_2,2),(p_3,0)\}$ or $M = [p_1,p_2^2]$.
- Petri nets must be associated with an **initial marking M_0** and with a set of possible **final markings**.

Firing Rules

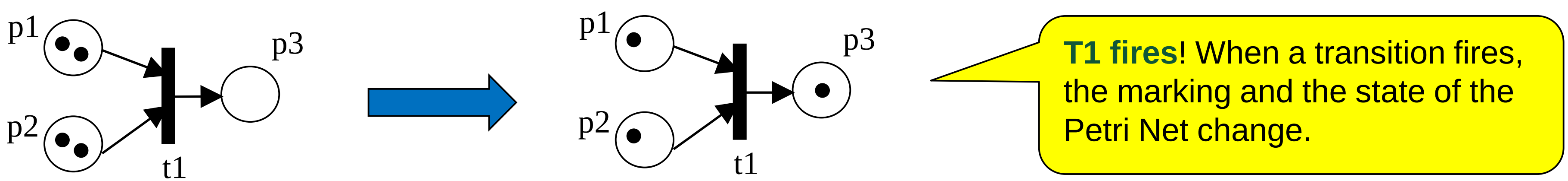
- The dynamic behavior of Petri nets is characterized by the notion of **firing** (*transition execution*). A transition can **fire** whenever there are one or more tokens in each of its input places.

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

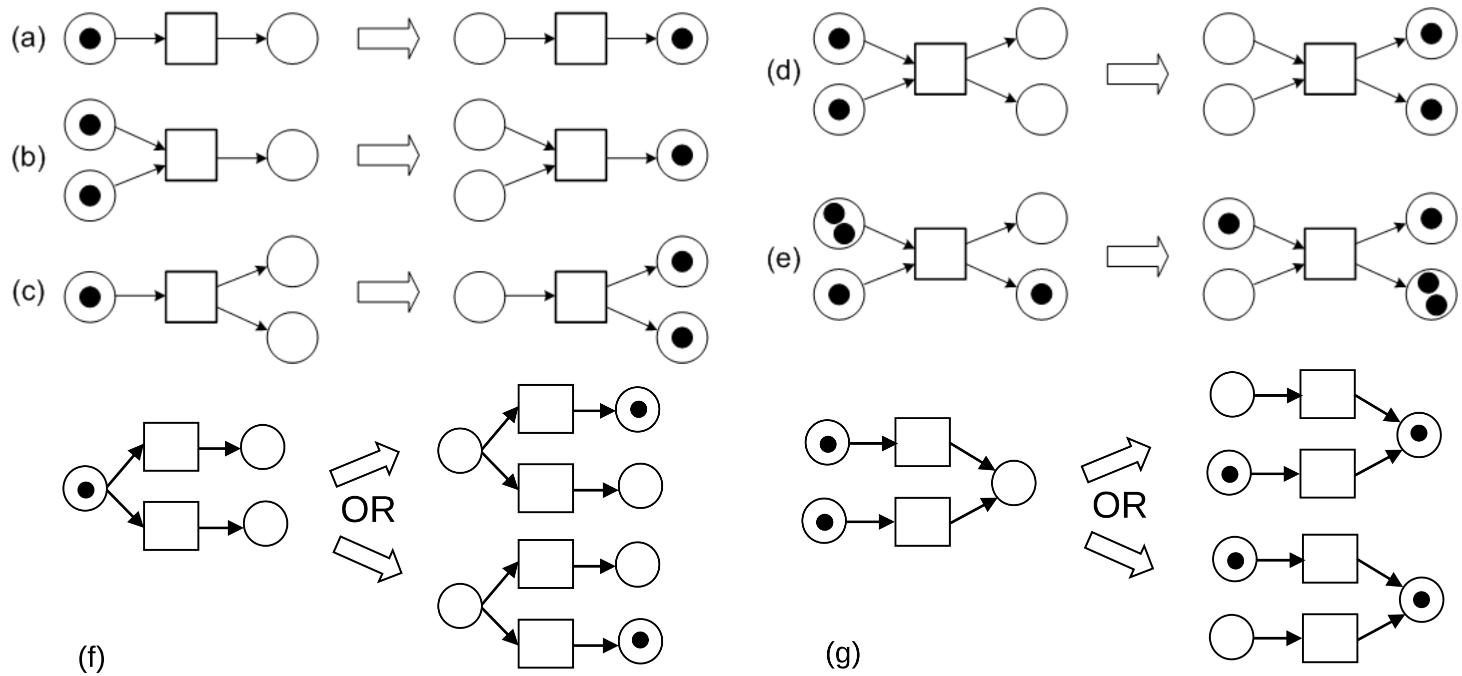




2. If transition t fires, then t **consumes one token** from each input place p of t and **produces one token** for each output place p of t.



Firing Transitions
Further Examples



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

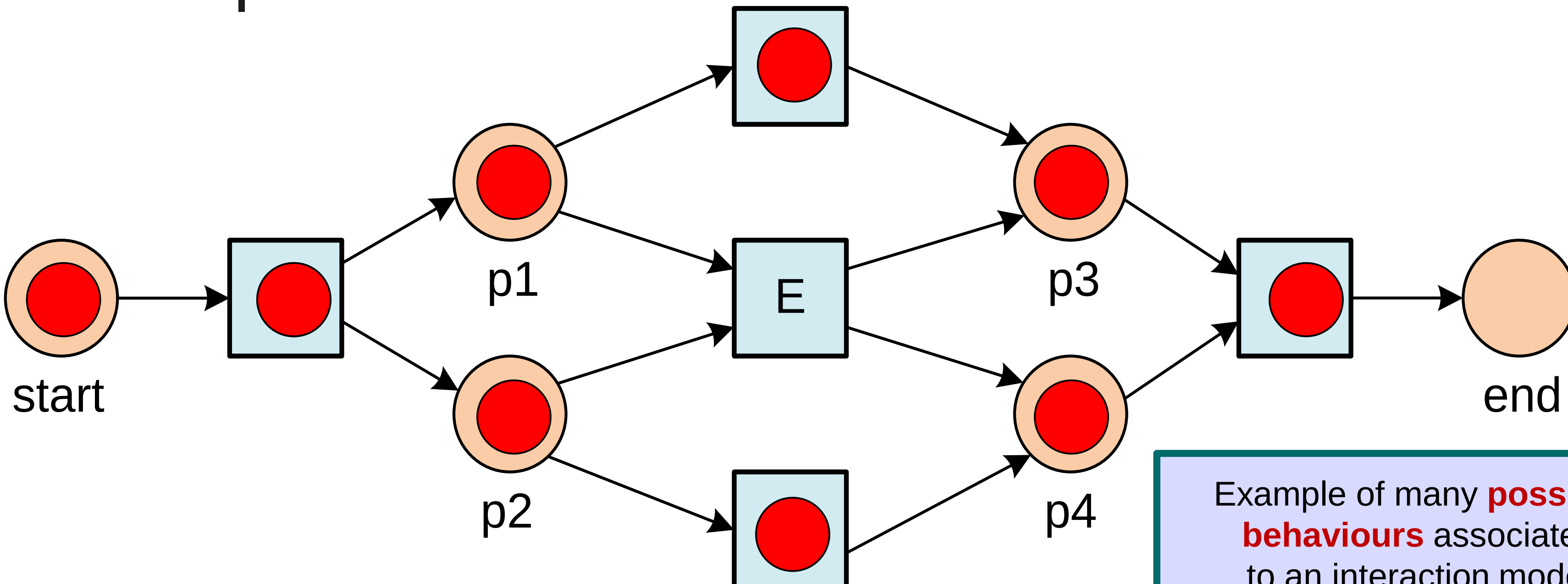
Boundness

- 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**.

Behaviors associated to a Petri Net



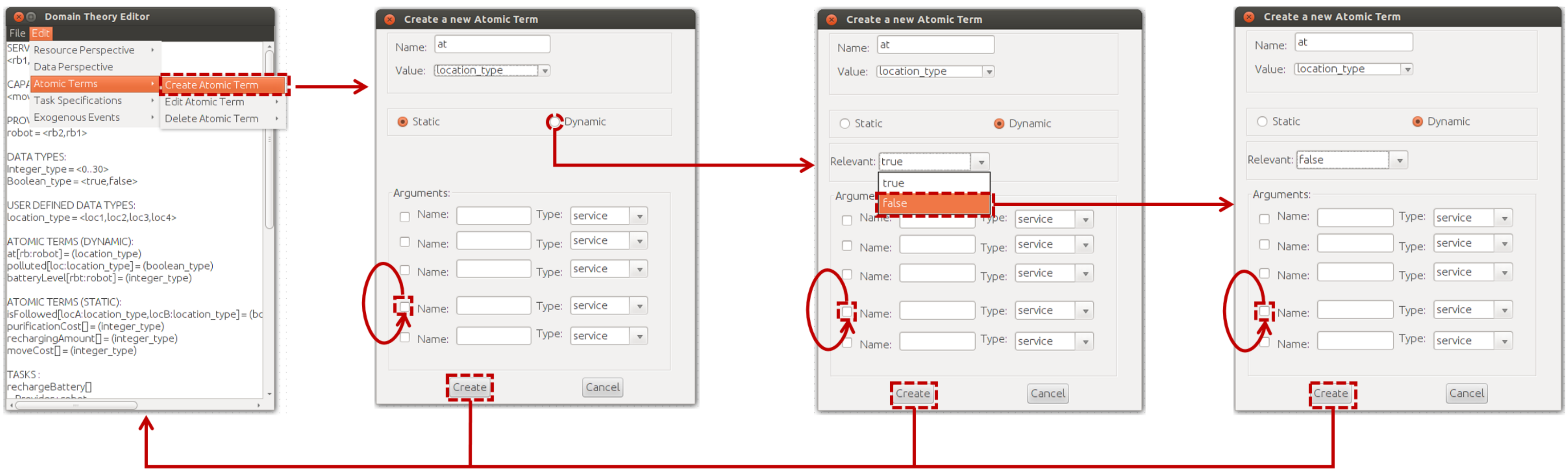
Example of many **possible behaviours** associated to an interaction model.

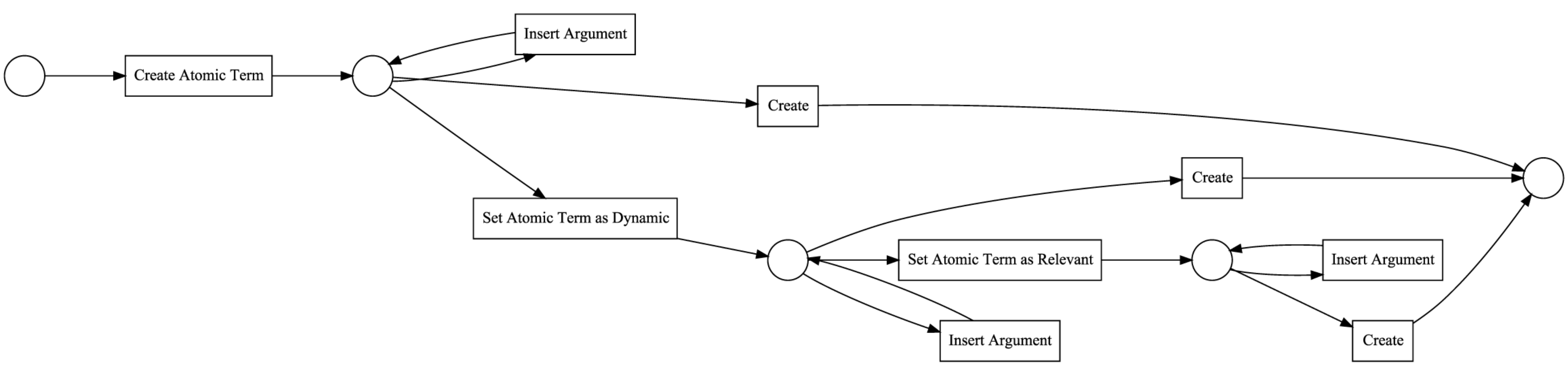
A B C D   A E D   A E D

A B C D   A C B D

A C B D   A E D   A C B D

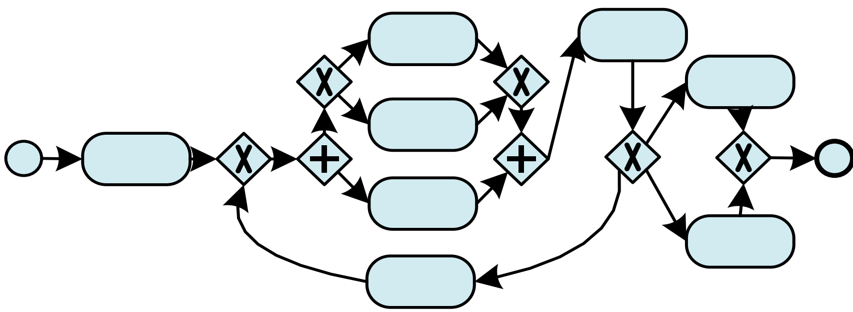
Petri Nets as Interaction Models





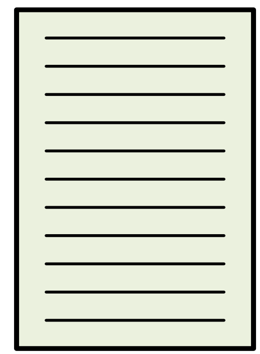
Generation of User Logs

Interaction models are not enforced by software systems. Traces can be **dirty**, with ***redundant*** or ***missing actions***



interaction model

Execution of relevant tasks



user log

Any execution of a relevant task produces a new **execution trace** recorded in a **user log**.

Example of a User Log

case id	event id	timestamp	activity	resource		
		properties				
1	35654423	30-12-2010:11.02	register request	Pete		
	35654424	31-12-2010:10.06	examine thoroughly	Sue		
	35654425	05-01-2011:15.12	check ticket	Mike		
	35654426	06-01-2011:11.18	decide	Sara		
	35654427	07-01-2011:14.24	reject request	Pete		
2	35654483	30-12-2010:11.32	register request	Mike		
	35654485	30-12-2010:12.12	check ticket	Mike		
	35654487	30-12-2010:14.16	examine casually	Pete		
	35654488	05-01-2011:11.22	decide	Sara		
	35654489	08-01-2011:12.05	pay compensation	Ellen		
3	35654521	30-12-2010:14.32	register request	Pete		
	35654522	30-12-2010:15.06	examine casually	Mike		
	35654524	30-12-2010:16.34	check ticket	Ellen		
	35654525	06-01-2011:09.18	decide	Sara		
	35654526	06-01-2011:12.18	reinitiate request	Sara		
	35654527	06-01-2011:13.06	examine thoroughly	Sean		
	35654530	08-01-2011:11.43	check ticket	Pete		
	35654531	09-01-2011:09.55	decide	Sara		
	35654533	15-01-2011:10.45	pay compensation	Ellen		
4	35654641	06-01-2011:15.02	register request	Pete	50	...
	35654643	07-01-2011:12.06	check ticket	Mike	100	...
	35654644	08-01-2011:14.43	examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	decide	Sara	200	...
	35654647	12-01-2011:15.44	reject request	Ellen	200	...
5	35654711	06-01-2011:09.02	register request	Ellen	50	...
	35654712	07-01-2011:10.16	examine casually	Mike	400	...
	35654714	08-01-2011:11.22	check ticket	Pete	100	...
	35654715	10-01-2011:13.28	decide	Sara	200	...
	35654716	11-01-2011:16.18	reinitiate request	Sara	200	...
	35654718	14-01-2011:14.33	check ticket	Ellen	100	...
	35654719	16-01-2011:15.50	examine casually	Mike	400	...
	35654720	19-01-2011:11.18	decide	Sara	200	...
	35654721	20-01-2011:12.48	reinitiate request	Sara	200	...
	35654722	21-01-2011:09.06	examine casually	Sue	400	...
	35654724	21-01-2011:11.34	check ticket	Pete	100	...
	35654725	23-01-2011:13.12	decide	Sara	200	...
	35654726	24-01-2011:14.56	reject request	Mike	200	...
6	35654871	06-01-2011:15.02	register request	Mike	50	...
	35654873	06-01-2011:16.06	examine casually	Ellen	400	...
	35654874	07-01-2011:16.22	check ticket	Mike	100	...
	35654875	07-01-2011:16.52	decide	Sara	200	...
	35654877	16-01-2011:11.47	pay compensation	Mike	200	...

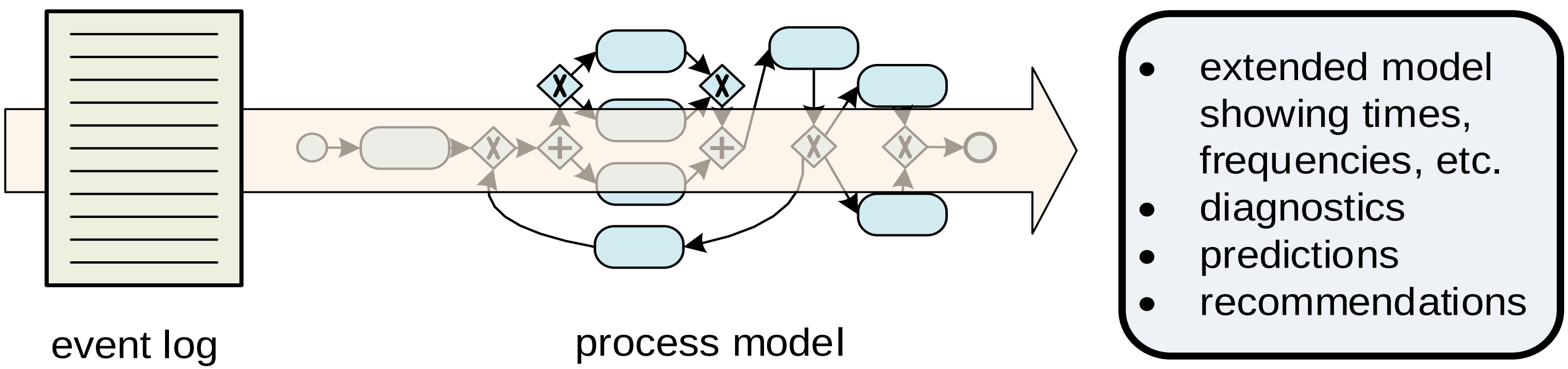
case id	trace
1	$\langle a,b,d,e,h \rangle$
2	$\langle a,d,c,e,g \rangle$
3	$\langle a,c,d,e,f,b,d,e,g \rangle$
4	$\langle a,d,b,e,h \rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h \rangle$
6	$\langle a,c,d,e,g \rangle$
...	...

The user actions belonging to a relevant task are ordered and form a **trace**, which can be seen as one complete execution of a task.

a = register request,
b = examine thoroughly,
c = examine casually,
d = check ticket,
e = decide,
f = reinitiate request,
g = pay compensation,
and h = reject request

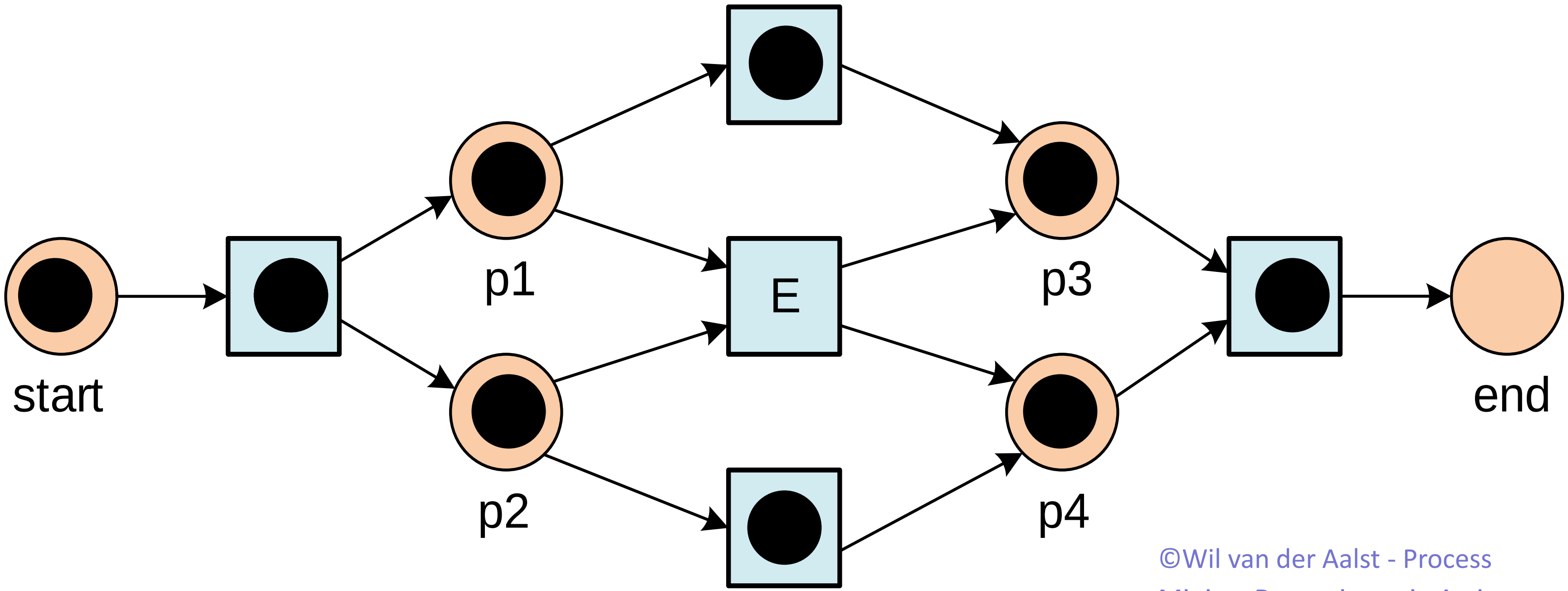
Replay

- **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***.



Replay

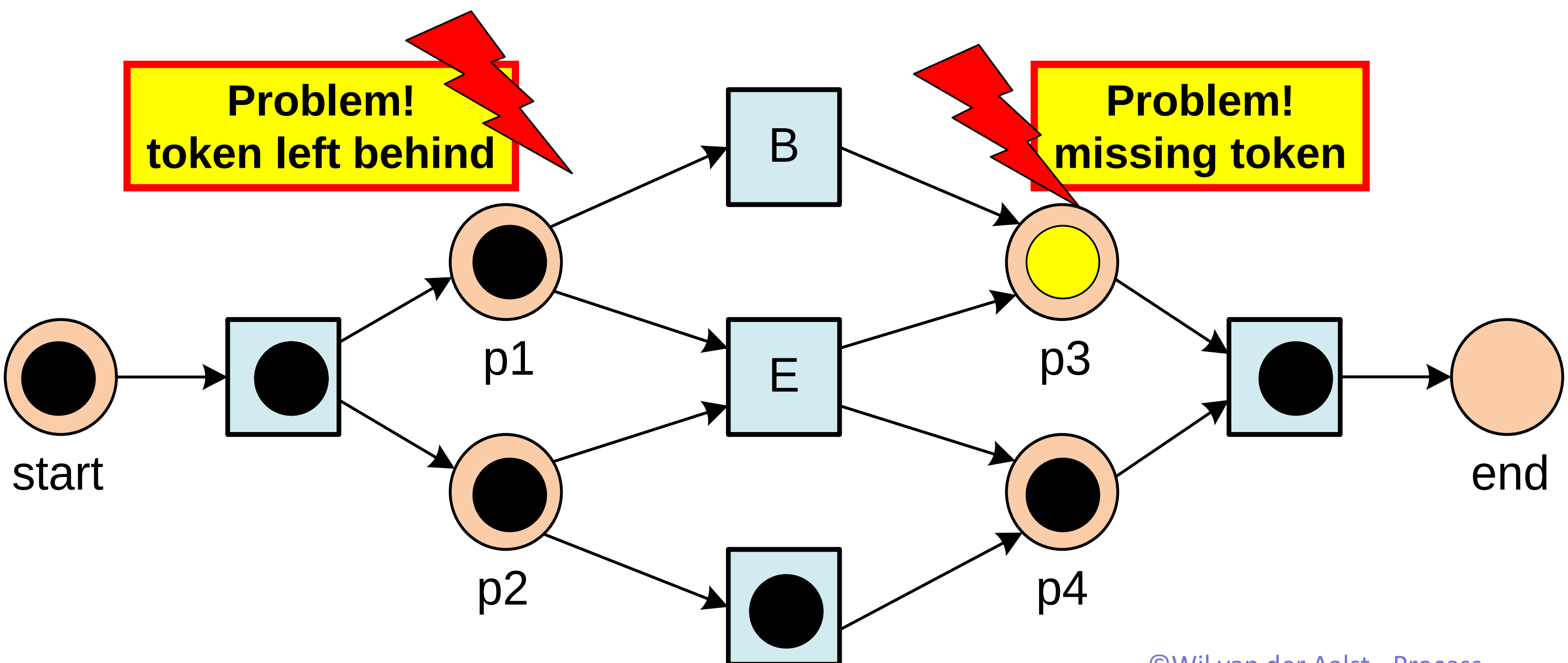
A B C D



©Wil van der Aalst - Process Mining: Data science in Action

Replay

A C D

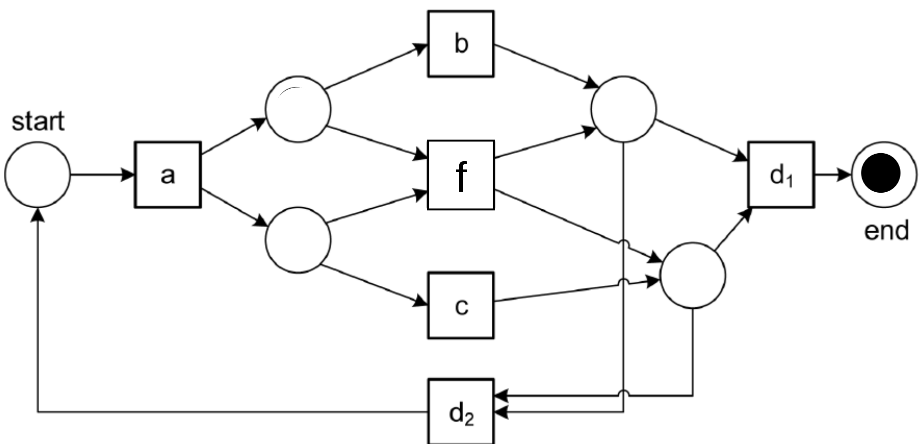


©Wil van der Aalst - Process Mining: Data science in Action

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)

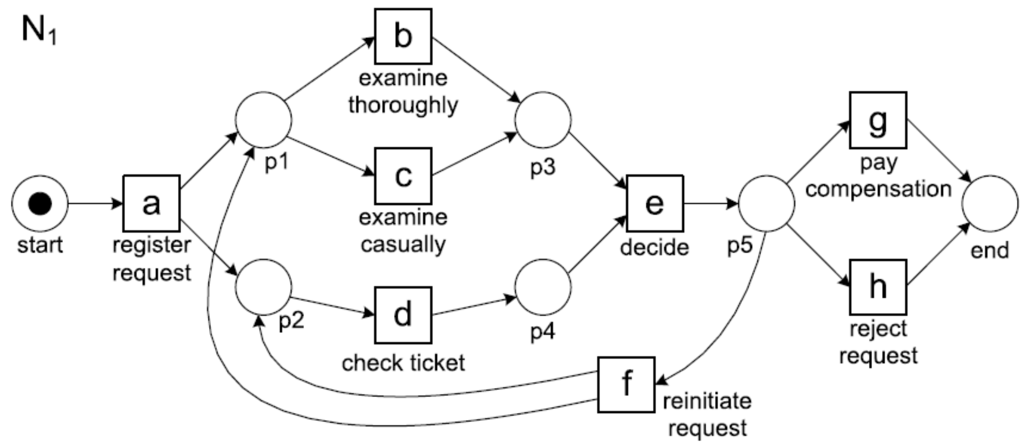
a	d	b	c	>
a	>	b	c	d

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

Moves have costs

- Standard cost function:
  - c(x,») = 1

…	a	…
…	»	…

  - c(»,y) = 1

…	»	…
…	a	…

  - c(x,y) = 0, if x=y

…	a	…
…	a	…

  - c(x,y) = ∞, if x≠y

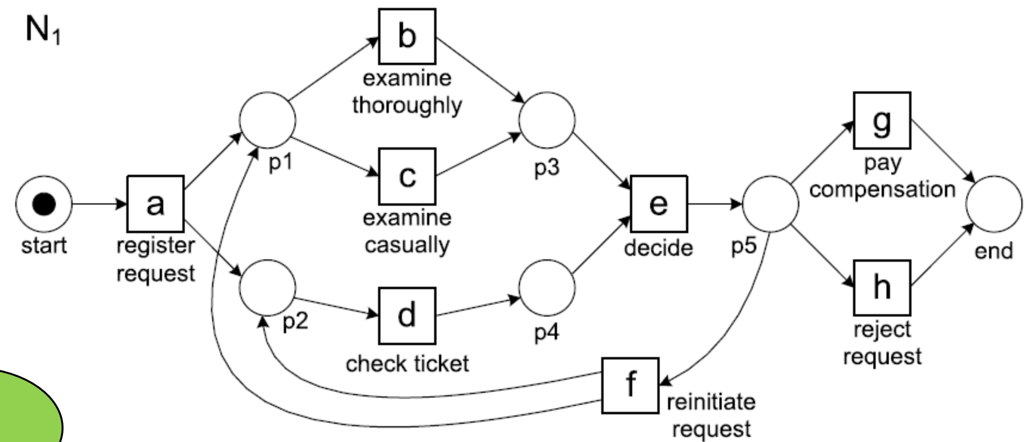
…	b	…
…	a	…

OPTIMAL ALIGNMENT
alignment with minimum deviation cost

Any cost structure is possible!

Optimal alignments

<abdeg>



optimal

a	b	d	e	g
a	b	d	e	g

0

a	b	»	d	e	g
a	»	c	d	e	g

2

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

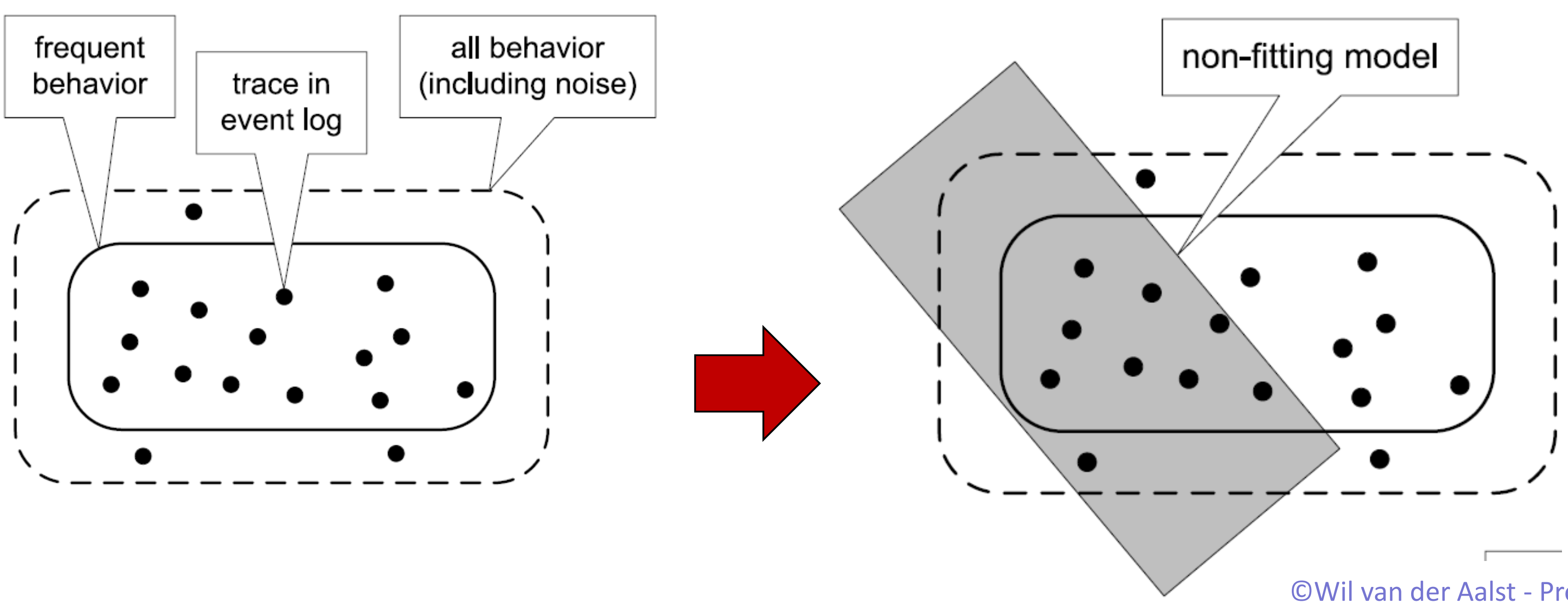
10

Learnability via fitness values

- **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 task is the **fitness value**, i.e., *how much the log adheres to a model of the interaction*.
  - The fitness value 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: 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.



©Wil van der Aalst - Process Mining: Data science in Action

Fitness based on alignments

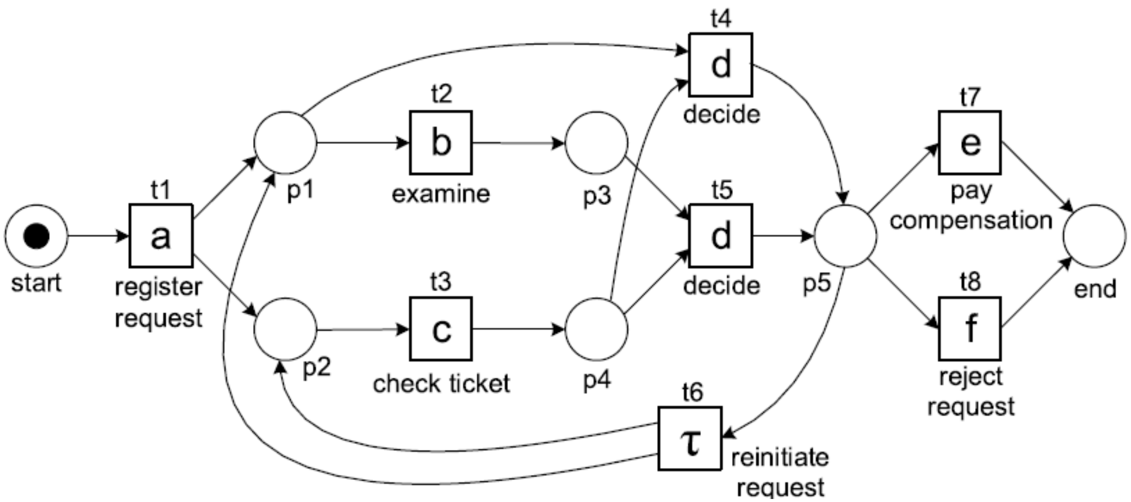
Cost of the **optimal alignment** of the trace σ

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

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

Example



$$\gamma_{5,2a} = \begin{array}{|c|c|c|c|c|} a & b & \gg & d & f \\ \hline a & b & c & d & f \end{array}$$

Cost of optimal alignment: 1

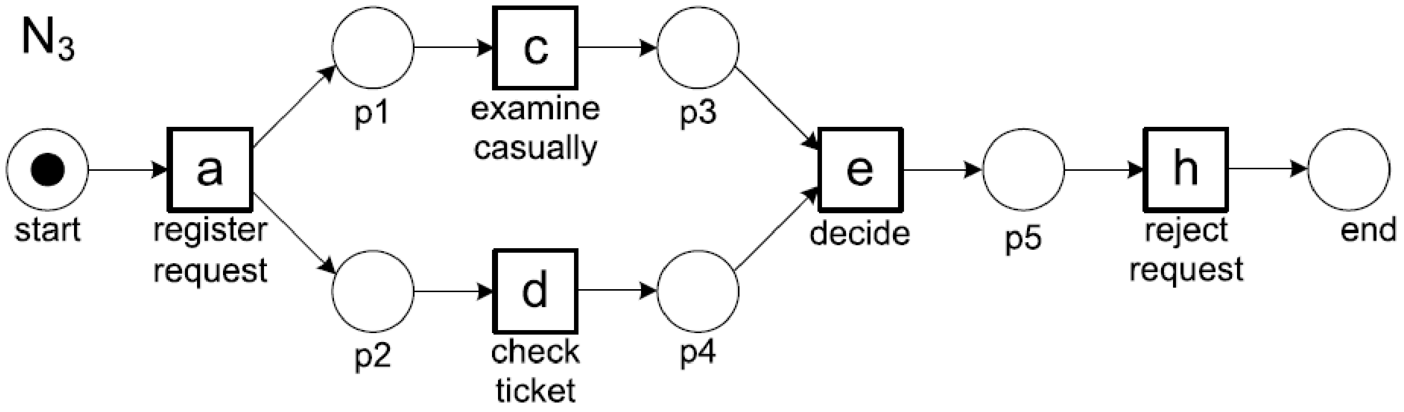
Cost of worst-case alignment: 8

$$\gamma_{5,2w} = \begin{array}{|c|c|c|c|c|c|c|c|} a & b & d & f & \gg & \gg & \gg & \gg \\ \hline \gg & \gg & \gg & \gg & a & c & d & f \end{array}$$

$$fitness(\sigma_2, N_5) = 1 - \frac{1}{8} = 0.875$$

Exercise

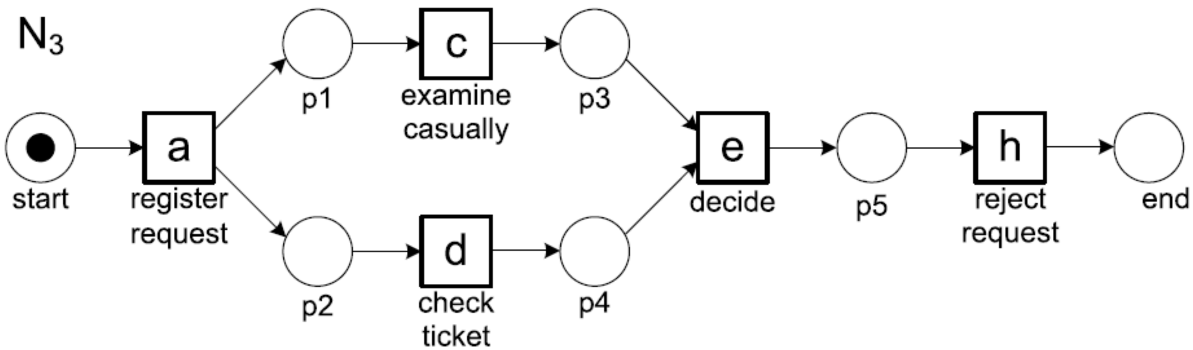
- Calculate the fitness of the trace <a,d,b,e,h> with respect to the model N_3



Solution

- Calculate the fitness of the trace <a,d,b,e,h> with respect to the model N_3

N_3



a	»	d	b	e	h
a	c	d	»	e	h

Cost of optimal alignment: 2

- Fitness: $1 - \frac{2}{10} = 0{,}8$

a	d	b	e	h	»	»	»	»	»
»	»	»	»	»	a	c	d	e	h

Cost of worst-case alignment: 10

Fitness for the entire log

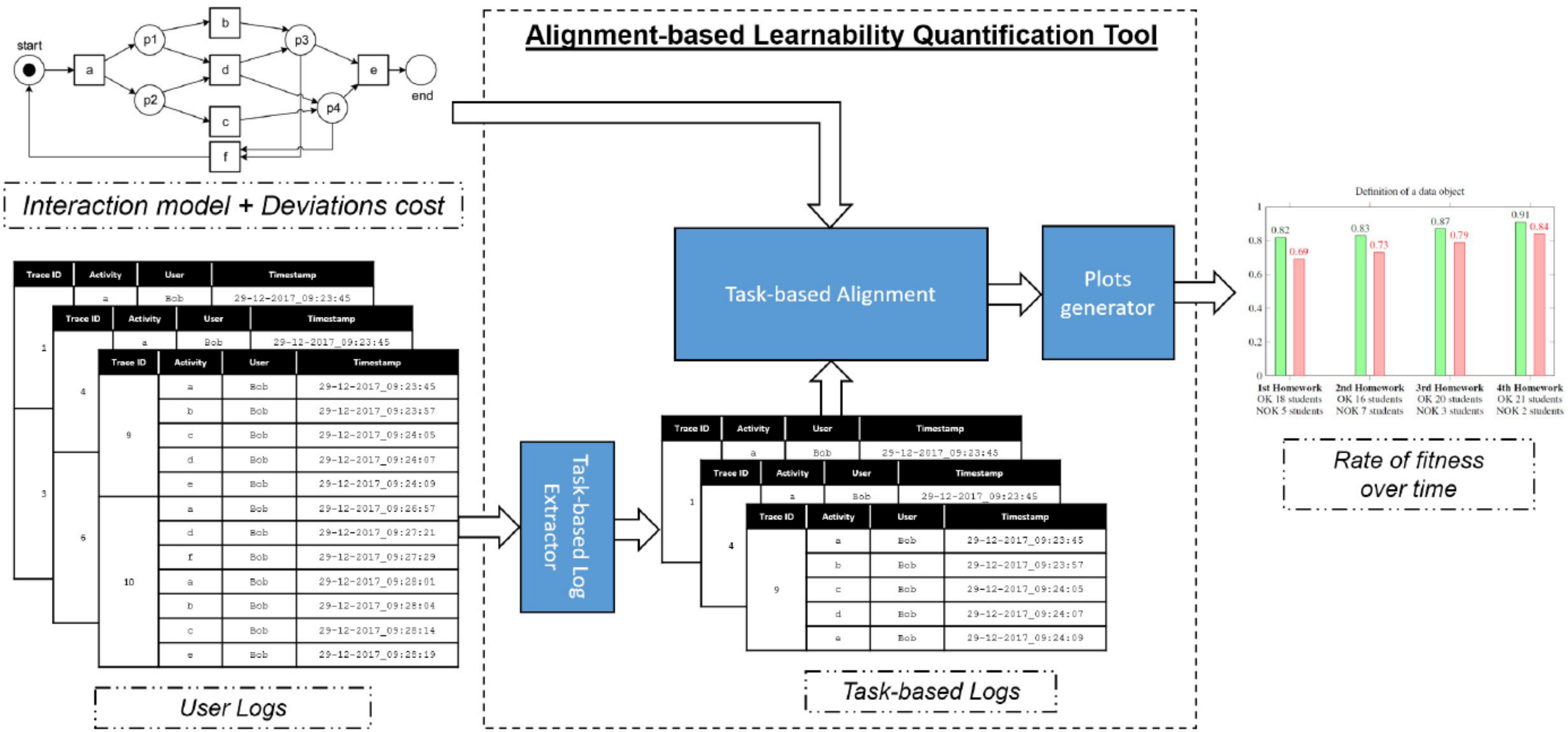
This is the sum of all costs when replaying the entire event log using optimal alignments

Number of occurrences of a specific trace in the log (e.g., if a trace σ appears 200 times in the log, L(σ) will be equal to 200)

$$fitness(L, N) = 1 - \frac{\sum_{\sigma \in L} L(\sigma) \times \delta(\lambda_{opt}^{N}(\sigma))}{\sum_{\sigma \in L} L(\sigma) \times \delta(\lambda_{worst}^{N}(\sigma))}$$

It is divided by the sum of the cost of all worst-case scenarios to obtain a normalized fitness value

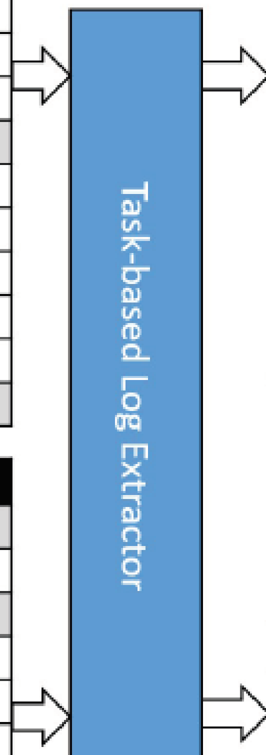
The tool for quantifying learnability [15]



Extraction of task-based log

Trace ID	Activity	User	Timestamp
1	a	Bob	29-12-2017_09:23:45
	b	Bob	29-12-2017_09:23:57
	c	Bob	29-12-2017_09:24:05
	d	Bob	29-12-2017_09:24:07
	e	Bob	29-12-2017_09:24:09
2	h	Bob	29-12-2017_09:25:19
	j	Bob	29-12-2017_09:25:54
	p	Bob	29-12-2017_09:26:17
3	a	Bob	29-12-2017_09:26:57
	d	Bob	29-12-2017_09:27:21
	f	Bob	29-12-2017_09:27:29
	a	Bob	29-12-2017_09:28:01
	b	Bob	29-12-2017_09:28:04
	c	Bob	29-12-2017_09:28:14
	e	Bob	29-12-2017_09:28:19

Trace ID	Activity	User	Timestamp
1	a	Bob	30-12-2017_17:13:45
	d	Bob	30-12-2017_17:13:57
	e	Bob	30-12-2017_17:14:09
2	h	Tom	30-12-2017_17:15:19
	j	Tom	30-12-2017_17:15:54
	p	Tom	30-12-2017_17:16:17
3	a	Tom	30-12-2017_19:56:57
	d	Tom	30-12-2017_19:57:21
	e	Tom	30-12-2017_19:57:29

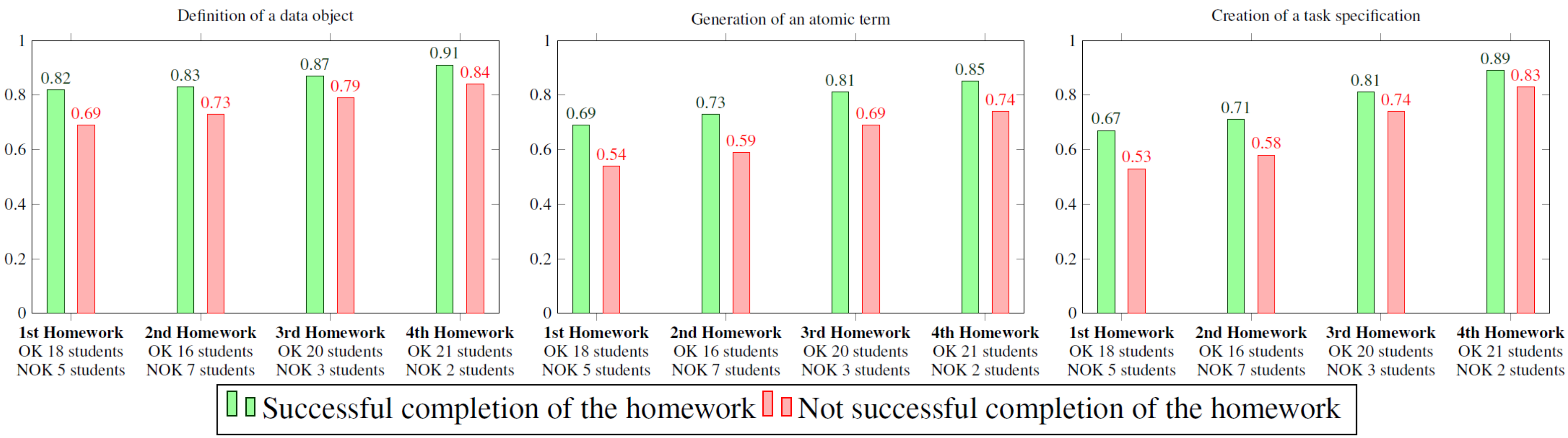


Trace ID	Activity	User	Timestamp
1	a	Bob	29-12-2017_09:23:45
	b	Bob	29-12-2017_09:23:57
	c	Bob	29-12-2017_09:24:05
	d	Bob	29-12-2017_09:24:07
	e	Bob	29-12-2017_09:24:09
3	a	Bob	29-12-2017_09:26:57
	d	Bob	29-12-2017_09:27:21
	f	Bob	29-12-2017_09:27:29
	a	Bob	29-12-2017_09:28:01
	b	Bob	29-12-2017_09:28:04
	c	Bob	29-12-2017_09:28:14
	e	Bob	29-12-2017_09:28:19

Trace ID	Activity	User	Timestamp
1	a	Bob	30-12-2017_17:13:45
	d	Bob	30-12-2017_17:13:57
	e	Bob	30-12-2017_17:14:09

Preliminary Experiments

- We performed a preliminary **longitudinal study** with 23 Master students against a GUI-based tool developed to support the design activity of business processes. <u>We recorded all the user actions in specific user logs</u>.
  - We assigned 4 different homeworks of growing complexity in 4 consecutive weeks.
  - The students were requested to complete the homework assigned to them in a specific week within the end of the week itself.
  - Before the assignment of a new homework, we shown to the students the minimal optimal solution to perform correctly the previous homework.
- We replayed user logs over three interaction models of the system describing the expected ways to achieve three relevant tasks.

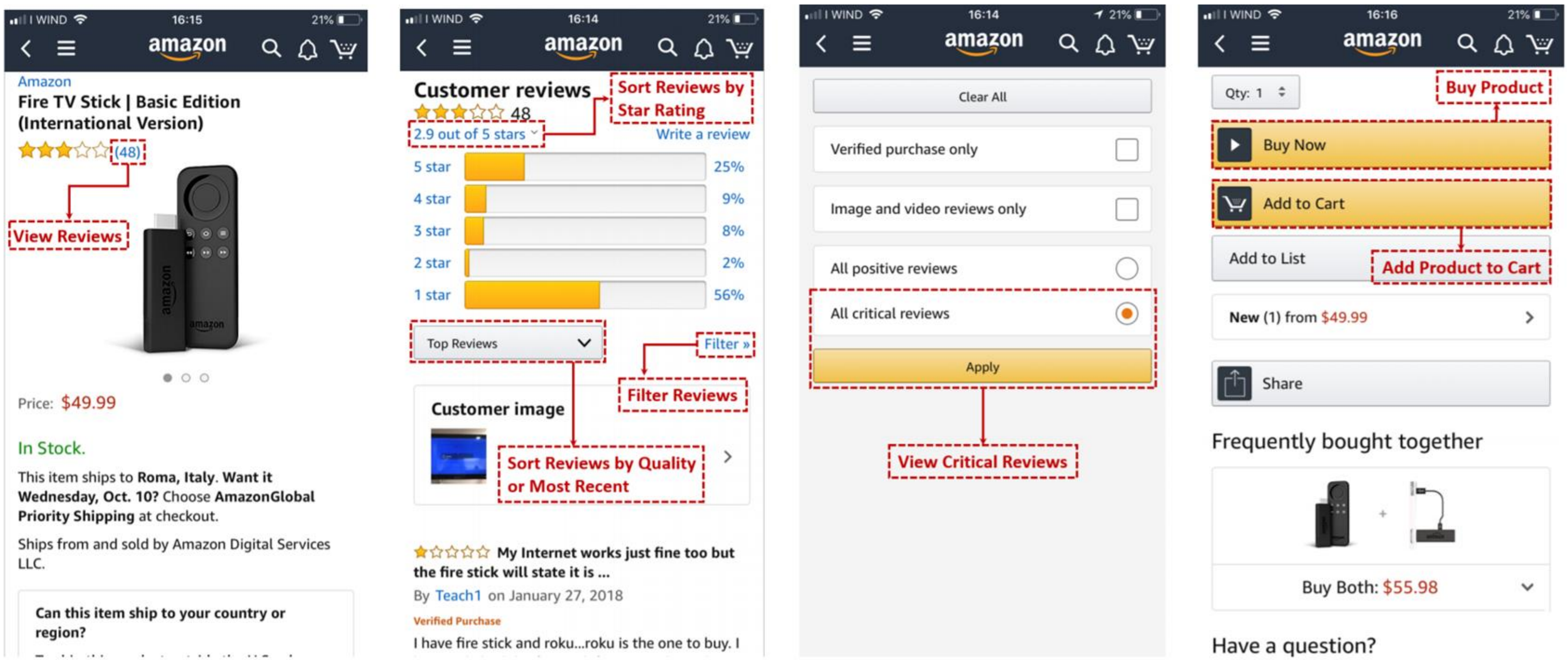


From Petri Nets to Declare

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	✓ acbacb	× abcab

Capturing user actions with Declare



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

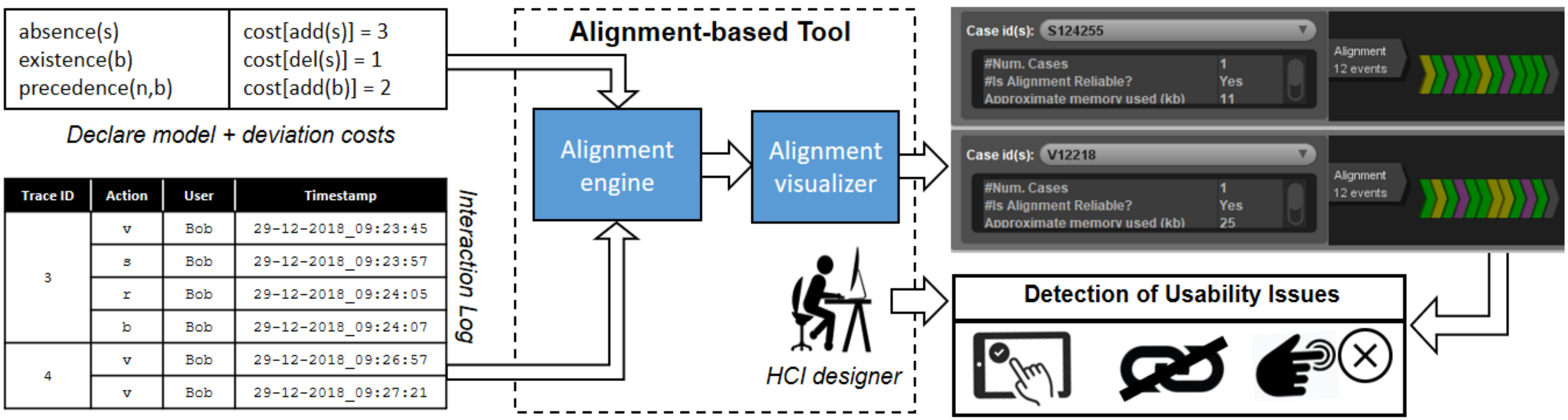
Some screenshots of the UI of Amazon shopping mobile app. In this example, we focus only on a subset of possible user actions such as the user reading only the critical reviews associated to a Fire TV stick, and then purchasing it.

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]



Trace ID	Action	User	Timestamp
3	v	Bob	29-12-2018_09:23:45
	s	Bob	29-12-2018_09:23:57
	r	Bob	29-12-2018_09:24:05
	b	Bob	29-12-2018_09:24:07
4	v	Bob	29-12-2018_09:26:57
	v	Bob	29-12-2018_09:27:21

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.

References

[1] Jakob Nielsen. 1994. Usability Engineering. Elsevier.

[2] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. 2009. A Survey of Software Learnability: Metrics, Methodologies and Guidelines. In Proc. SIGCHI Conf. Human Factors in Comp. Sys. ACM.

[3] Alan Dix. 1990. Design issues for reliable time-critical systems. Interacting with Computers 4, 3 (1990).

[4] Jakob Nielsen. 1994. Usability Engineering. Elsevier.

[5] Andreas Holzinger. 2005. Usability engineering methods for software developers. Commun. ACM 48, 1 (2005).

[6] Paulo J Santos and Albert Badre. 1995. Discount learnability evaluation. Technical Report. Georgia Institute of Technology.

[7] Ben Shneiderman. 2010. Designing the user interface: strategies for effective human-computer interaction. Pearson Education India.

[8] Christie D Michelsen, Wayne D Dominick, and Joseph E Urban. 1980. A methodology for the objective evaluation of the user/system interfaces of the MADAM system using software engineering principles. In Proc. of the 18th annual Southeast regional conference. ACM, 103–109.

[9] Keith A Butler. 1985. Connecting theory and practice: a case study of achieving usability goals. In ACM SIGCHI Bulletin, Vol. 16. ACM, 85–88.

References

[10] Nigel Bevan and Miles Macleod. 1994. Usability measurement in context. Behaviour & information technology 13, 1-2 (1994), 132–145.

[11] Han X Lin, Yee-Yin Choong, and Gavriel Salvendy. 1997. A proposed index of usability: a method for comparing the relative usability of different software systems. Behaviour & information tech. 16, 4-5 (1997), 267–277.

[12] Christian Stickel, Josef Fink, and Andreas Holzinger. 2007. Enhancing universal access–EEG based learnability assessment. Universal Access in Human-Computer Interaction. Applications and Services (2007), 813–822.

[13] Sid Davis and Susan Wiedenbeck. 1998. The effect of interaction style and training method on end user learning of software packages. Interacting with Computers 11, 2 (1998), 147–172.

[14] Deborah A Mitta and Sherrill J Packebush. 1995. Improving interface quality: an investigation of human-computer interaction task learning. Ergonomics 38, 7 (1995), 1307–1325.

[15] A. Marrella, T. Catarci. Measuring the Learnability of Interactive Systems Using a Petri Net Based Approach. 2018 Int. Conf. on Designing Interactive Systems (DIS '18)

[16] A. Marrella, L.S. Ferro, T. Catarci. An Approach to Identifying What Has Gone Wrong in a User Interaction. IFIP Conference on Human-Computer Interaction, 361-370