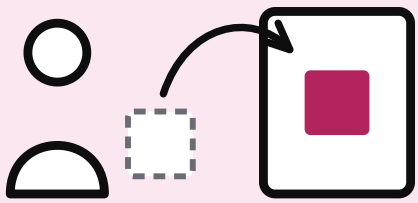
Concept

Tesler's law

Every application has a share of complexity that cannot be removed, only moved. The question is who deals with it: the person or the team that builds it.



In Spanish

Ley de Tesler

Origin

Larry Tesler (around 1984, by his own account); spread by Tognazzini (1998) and Saffer (2010)

What problem it addresses

Explains why simplifying an interface does not make the work disappear: someone has to do it, and it is better if it is not the person using it.

An example

An illustrative example: entering a shipping address.

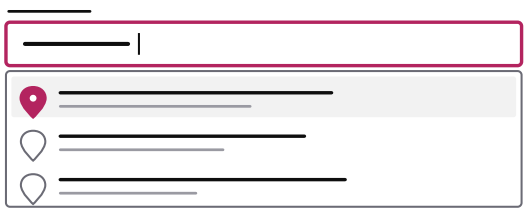
Complexity left to the person



Five fields: the work falls on the person

Five fields to fill in and a format to guess. **Each person does the work.**

Complexity taken on by the system



One field: the system fills in the rest

A single field that suggests the full address as you type. **The system does the work.**

The address is just as complex: only who deals with it has changed.

Why it happens

1 

Every task has a share of complexity that cannot be taken away.

2 

If the system does not deal with it, the person has to.

3 

Better for the team to solve it once than for each person every time they use the product.

Applications in design

Use it like this

Before asking for a piece of data or a decision, ask yourself whether the system could handle it for the person.

Avoid

Calling something simplified when it has only been hidden and the person will have to deal with it later.

Don't fall for it

Moving complexity is not taking away control: if the system decides for the person, let them see and correct what it did.

Around 1984

What the research says

Larry Tesler formulated the law around 1984: every application has complexity that cannot be reduced, and the question is whether the user, the application developer or the platform developer takes it on (Tesler, n.d.).

● ○ ○ **Single study.** No studies measure it: it is a principle from practice, known through Tesler himself and those who cited him. Tognazzini (1998) adds a nuance: when something gets simpler, people take on more complex tasks.

Connections

Used in

- Interaction design

Related to

- Cognitive load
- Progressive disclosure
- Default effect

Authors and works

around 1984	**Larry Tesler**	Formulated the law of conservation of complexity; he did not publish it in a paper at the time.
1998	**Bruce Tognazzini**	Cited it in his column “The Complexity Paradox” and added that, when things get simpler, people take on harder tasks.
2010	**Dan Saffer**	Included in Designing for Interaction an interview in which Tesler explains the law and argues that complexity should be borne by those who program, not by those who use.
2020	**Jon Yablonski**	Gathered it with other user experience laws in the book Laws of UX.

Sources

Original sources

- Tesler, L. (n.d.). The law of conservation of complexity (Tesler's law). Larry Tesler (sitio web mantenido por su familia). View source (https://www.nomodes.com/computing-history)

Secondary sources

- Tognazzini, B. (1998, septiembre). The complexity paradox. AskTog. View source (https://asktog.com/columns/011complexity.html)
- Saffer, D. (2010). Designing for interaction: Creating innovative applications and devices (2nd ed.). New Riders.
- Yablonski, J. (2020). Laws of UX: Using psychology to design better products & services. O'Reilly Media. View source (https://www.oreilly.com/library/view/laws-of-ux/9781492055303/)

Last reviewed: October 26, 2026

Atlas público de diseño, de Adriana Ojeda. Texto e ilustraciones bajo licencia CC BY 4.0.