

The user in the cultures of UX design and open source

Jan Dittrich

Design in Open
Source seems
hard

Different Values,
Different Practices

The User

Eric S. Raymond: The Cathedral and the Bazaar

Donald Norman: The Design of Everyday Things

The Cathedral and the Bazaar

User and Creator are
the same person

User and Creator are the same person

...from the start:

“1. Every good work of software starts by scratching a developer's personal itch”

User and Creator are the same person

...and with growth

“6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.”

User and Creator

In “The Cathedral And the Bazaar”

Are ideally the same person

Creating for their own need/use

Attracting other, similar people

In “The Design of Everyday things”

User and Creator must be different

...In their work, designers often
become expert with the device...
Users are often expert at the task...

p156

User and Creator must be different

“Design teams really need vocal advocates for the people who will ultimately use the interface”

Because

[Creators] “tend to simplify their own Lives.”

p156

User and Creator must be different

The only way to find out is to test the designs on users-people as similar to the eventual purchaser of the product as possible.

p156

User and Creator

In “The Cathedral And the Bazaar”

Are ideally the same person

Creating for their own need/use

Attracting other, similar people

In “The Design of Everyday things”

Have different expertise

Have different interests

Creators need thus to learn what helps users by empirical methods

User and Creator

In “The Cathedral And the Bazaar”

Are ideally the same person

Creating for their own need/use

Attracting other, similar people

Assumes a community of like-minded user/creators who can code

In “The Design of Everyday things”

Have different expertise

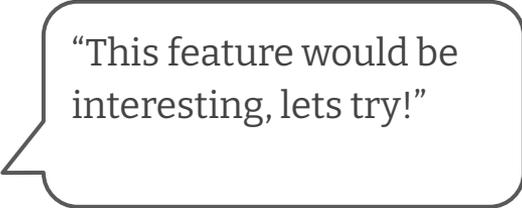
Have different interests

Creators need thus to learn what helps users by empirical methods

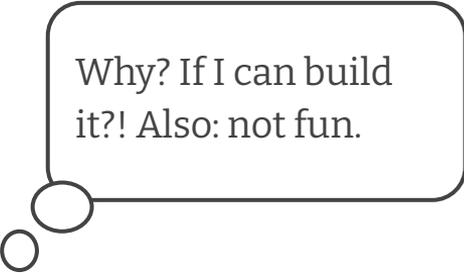
Assumes an organization with division of labor around creating products that are bought

User and Creator

Open Source

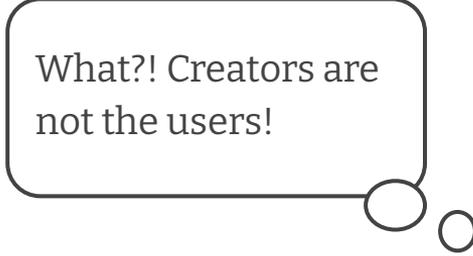


“This feature would be interesting, lets try!”



Why? If I can build it?! Also: not fun.

UX



What?! Creators are not the users!



“Let’s find out what users actually need!”

Easy to modularize, hard to use for beginners

```
jan@jan-Desktop:~$ █
```

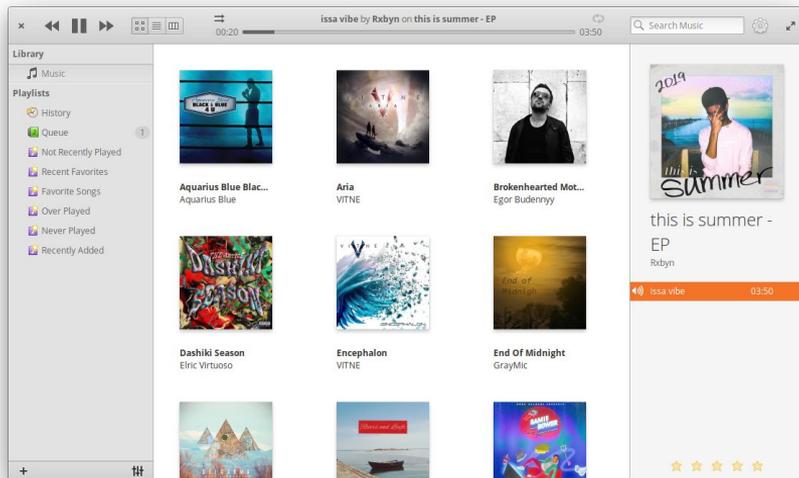
Add a parameter

Call another program

Pipe it

Everything is possible

Hard to modularize, easy use for beginners



One coherent UI

Works like the ones you know

But what if I WANT TO ADD A BUTTON for...

Screenshot from <https://github.com/elementary/music>

What is modular about GUIs?

Icons!



Libre Office Themes



Elementary



Colibre

Yaru

So much Open Source Design is icon design since changing/adding icons

it does not clash with typical open source development concerns – or actually fits it well

Coherence

The model of the “Bazaar” tolerates incoherence. Or needs it. The best ways will win in the end.

Incoherence is also no problem, as the competent user/creator will have no problem with it

The “Design of Everyday Things” needs coherence. With the existing skills of users, within the product

Incoherence is a problem, as the user might know nothing about the products technology.

Coherence

Incoherence in the UI is also no problem, as the competent user/creator will have no problem with it.

Coherence in the UI is a problem, as it makes changes hard and conflict and politics-prone.

Problems should be solved by modularity and experimentation

Incoherence in the UI is a problem, as the user might know nothing about the products technology.

Coherence is great, it will create ease of use

Problems should be solved by finding needs and principles

Combining the needs?

Modularity

Experimentation

Being competent creator and
user

Coherence

Empiricism

NOT being the user but learning about
them

Combining the two?

Hard

But here are some helps for small wins

Boundary Objects for collaboration

Boundary Objects for collaboration

Interface Design Guidelines

Design Systems

(GUI) Extension APIs

Interface guidelines

Back to the 90s!

When to use which element and why

Gnome Interface Guidelines Screenshot,
Copyright © 2005–2014 The GNOME Project



Whether you are a developer or a designer, these guidelines contain everything you need to design effective applications using GTK. They cover design principles for GNOME 3, common guidelines such as how to write text and use images and icons, as well as a library of design patterns which you can use in your application.

While the HIG places an emphasis on designing for GNOME 3, it can also be used to create cross-platform applications, as well as for applications that have previously followed the GNOME 2 Human Interface Guidelines. The [compatibility guidelines](#) contain more information on this.

Core material

Patterns and user interface elements form the core of the HIG. Together, they are the building blocks for application design. If you are new to the HIG, it is recommended that you start with the page on design principles and then browse the patterns, before going on to other material.

Design principles	General design guidelines and advice.
Patterns	Essential and optional design components.
User interface elements	Guidelines on common elements, such as buttons, progress bars and popovers.

Common guidelines

These guidelines apply to the full range of design elements. It is recommended that you familiarize yourself with them.

Application basics	Basic application behavior and characteristics.
Compatibility	Using the HIG for cross-platform or GNOME 2 style applications.
Visual layout	Arranging elements within user interfaces.
Writing style	Writing text for your user interface, including capitalization rules.
Icons and artwork	Guidelines on selecting and creating icons.
Typography	Advice on font sizes, weights and styles, as well as special characters.
Pointer and touch input	Mouse, touchpad and touchscreen interaction.
Keyboard input	Keyboard navigation, access and shortcut keys.

Interface guidelines

<https://docs.microsoft.com/en-us/windows/win32/uxguide/guidelines>

Design systems

UI Elements...

Input Box

Box Label

Button

... and how to combine them

Name

Create

[Atomic Design](#) by Brad Frost

Manage a Design System in [Storybook](#), Dev focus

Design systems

Inputs

Dropdown

Dropdown Input

Value

Dropdown Input

Value

First Entry

Selected Text

purposefully not
on grid

Following Point

Text input

Text Input Base

Value

Text Input Focussed

Value

Text Input Error

Value



Please check the foobar, it seems to be incorrect. Without a correct foobar, we



[Spenden](#)

[Mitglied werden](#)

[Häufige Fragen](#)

[Mittelverwendung](#)

Die Wikimedia Fördergesellschaft ist eine unabhängige gemeinnützige Organisation, die in Deutschland Spenden für Wikipedia und andere Wikimedia-Projekte sammelt.

Sie spenden: halbjährlich 75,00 Euro via Lastschrift

[Zahldaten ändern](#)

Wie lauten Ihre IBAN und BIC?

IBAN (oder Kontonummer)

z. B. DE12345678909876543210

Spenden als...

Privatperson

Verschlüsse

Frage zum :

Rufen Sie ur

[Kontaktform](#)

030 / 21 91 :

Extendable UIs

Coherent mechanism
for UI extension and an
API of what such an
extension is allowed to
do (without messing up
other functions)

Extendable UIs

Might be a mess in the managed-by-extension space BUT at least not messing with the rest of the application.

[Firefox Web Extensions](#)

[Chrome Browser Extensions](#)



Extendable UIs

Best if build upon a design system and/or with the standard GUI elements

[Jamovi Statistics](#)

The screenshot displays the Jamovi Statistics software interface. The top navigation bar includes 'Data', 'Analyses', and 'Edit' tabs. Below the navigation bar is a toolbar with icons for Exploration, T-Tests, ANOVA, Regression, Frequencies, and Factor. The main workspace is divided into two panels: the configuration panel on the left and the results panel on the right.

One-Way ANOVA Configuration Panel:

- Dependent Variables:** A list box containing variables A, B, and C.
- Grouping Variable:** A text input field.
- Variations:**
 - Don't assume equal (Welch's)
 - Assume equal (Fisher's)
- Missing Values:**
 - Exclude cases analysis by analysis
 - Exclude cases listwise
- Additional Statistics:**
 - Descriptives table
 - Descriptives plots
- Assumption Checks:**
 - Homogeneity test
 - Normality test
 - Q-Q Plot
- Post-Hoc Tests:** A dropdown menu currently showing '> | Post-Hoc Tests'.

Results Panel:

- Reliability Analysis:** Two sections showing 'Scale Reliability Statistics' with 'Cronbach's α ' and 'scale' values, and a significance level of [.].
- Proportion Test (2 Outcomes):** A section for 'Binomial Test' with a table structure:

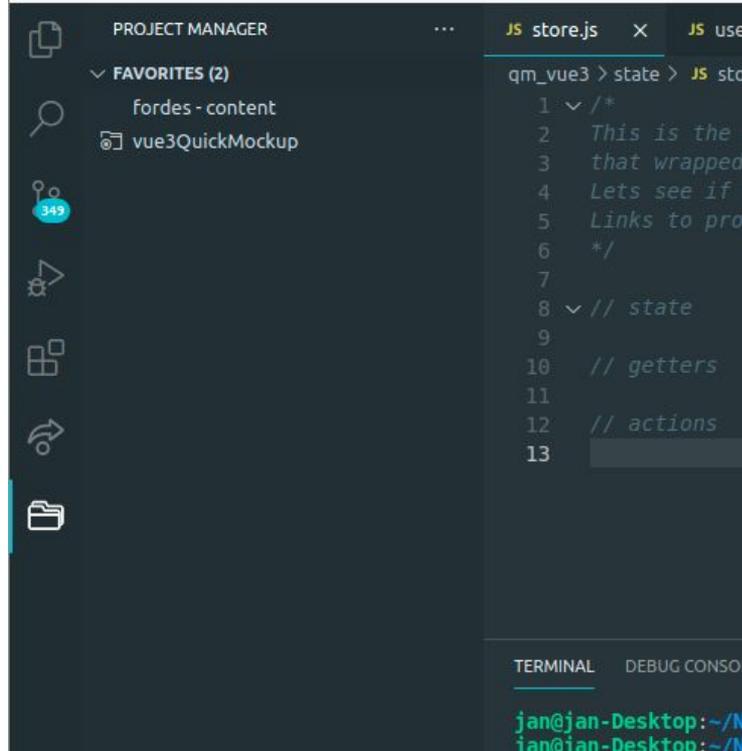
Level	Count	Total	Proportion	p
- One-Way ANOVA:** A section for 'One-Way ANOVA (Welch's)' with a table structure:

	F	df1	df2	p

Extendable UIs

Best if build upon a design system and/or with the standard GUI elements

[VS Code](#)



```
PROJECT MANAGER
...
JS store.js x JS use
qm_vue3 > state > JS sto
1  ✓ /*
2  This is the
3  that wrapped
4  Lets see if
5  Links to pro
6  */
7
8  ✓ // state
9
10 // getters
11
12 // actions
13

TERMINAL  DEBUG CONSOLE
jan@jan-Desktop:~/N
ian@ian-Desktop:~/N
```

Recap

Very different views on what a “good” user is and how to work with them

Open Source: Experimenting user/creator in a community similar to them

UX Design: Designing for others using empirical research.

Boundary Objects shared by designers and devs:

- Interface Guidelines
- Design systems

Create APIs for changing UIs

Thank You

Jan Dittrich

Works at Wikimedia Deutschland e.V
as user researcher

@simulo

d_jan@ymail.com