U.S. DEPARTMENT OF ENERGY
Office of Science

The Journey to STRUDEL:
How We Came to Embrace User Experience in Scientific Ecosystems

July 24, 2024

https://strudel.science

https://ux.lbl.gov

Lavanya Ramakrishnan

Lawrence Berkeley National Lab
LRamakrishnan@lbl.gov

Funded by Sloan Foundation, Liz Vu & Josh Greenberg Program Officers, Grants #10074 & #10572

ALFRED P. SLOAN FOUNDATION

UX.lbl.gov SciData

Scientific Data Division

BERKELEY LAB
Bringing Science Solutions to the World

Team

Lavanya Ramakrishnan
lramakrishnan@lbl.gov

Dan Gunter
dkgunter@lbl.gov

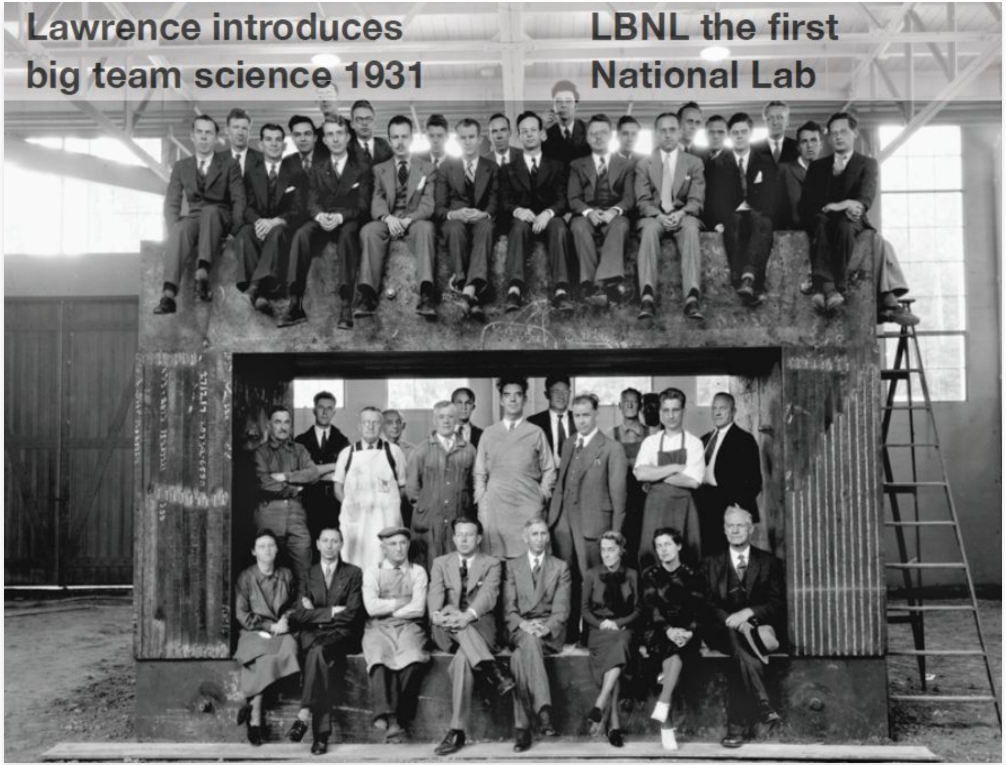
Sarah Poon
sspoon@lbl.gov

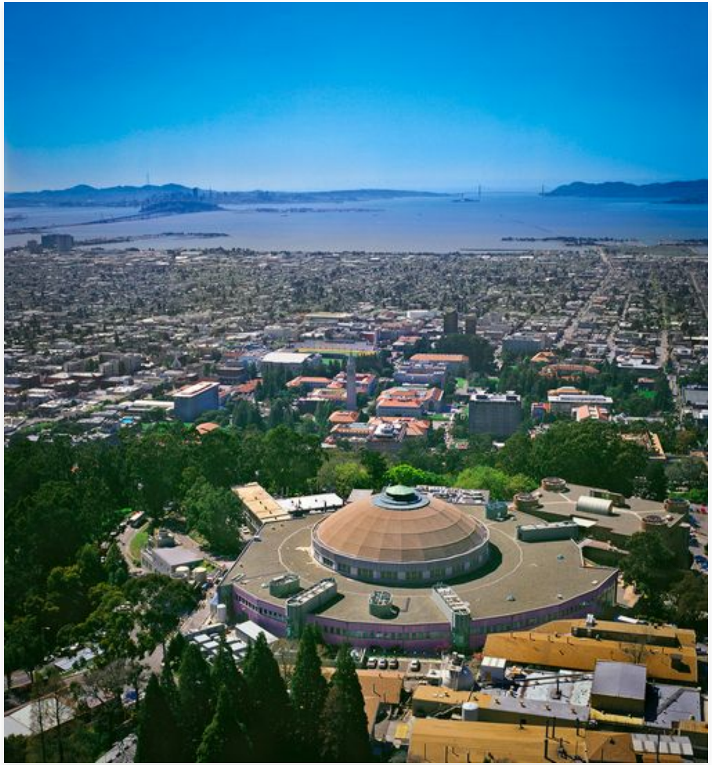
Rajshree Deshmukh
rajshreed@lbl.gov

Cody O’Donnell
ctodonnell@lbl.gov

Drew Paine
pained@lbl.gov

Team science is at the core of what we do at Berkeley Lab





from LBNL image archive

Workflows: How do we enable researchers to effectively and efficiently manage their computation and data?

ESnet

Molecular Foundry

ALS

JGI

NERSC

ESS-DIVE

AmeriFlux Network

NMDC

KBase

Workflow management

- data abstractions
- HPC and distributed
- resource management
- autonomous pipelines
- reproducibility

Data management

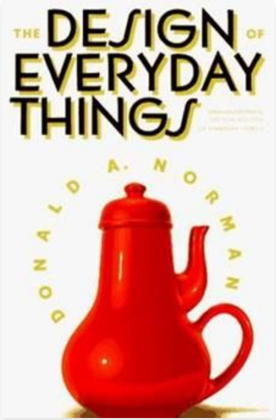
- search through AI-driven metadata extraction
- data change
- provenance

Why user experience (UX) matters for scientific software

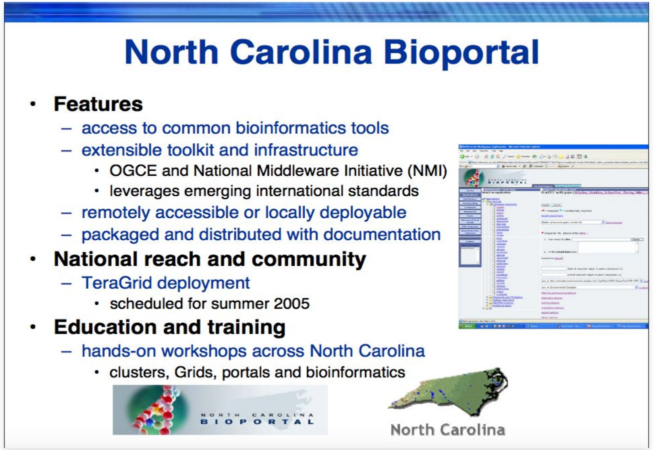
How our team views UX for scientific software development

How these experiences lead to STRUDEL as a way to provide open source tools to help teams build more usable scientific software

How did I get here …



~2001



~2005

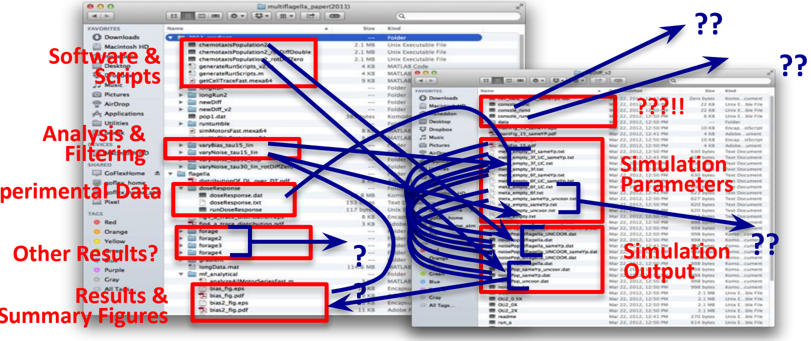
2012

Why is building usable scientific software challenging?

Menti poll — Question 1 & 2

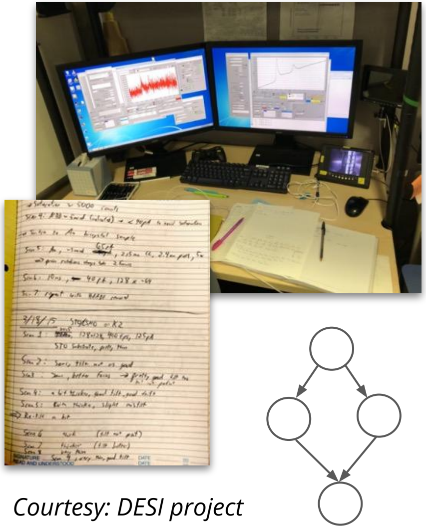
Realities of scientific work

Don't fit into nice graphs



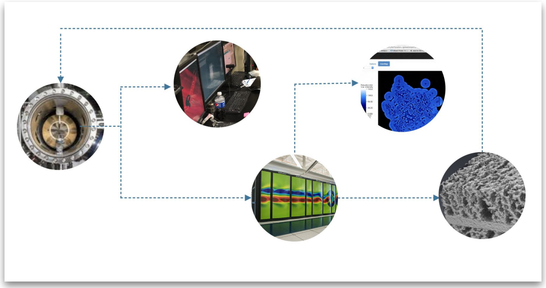
Courtesy: Paramvir Dehal, KBase team

Supporting artifacts and context are not captured



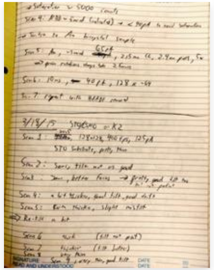
Courtesy: DESI project

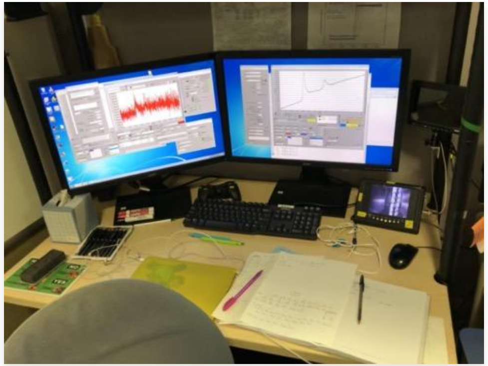
Collaborations have complex software stacks



New work practices that don’t fit into current work process will likely not get adopted.

How we see UX in scientific software development



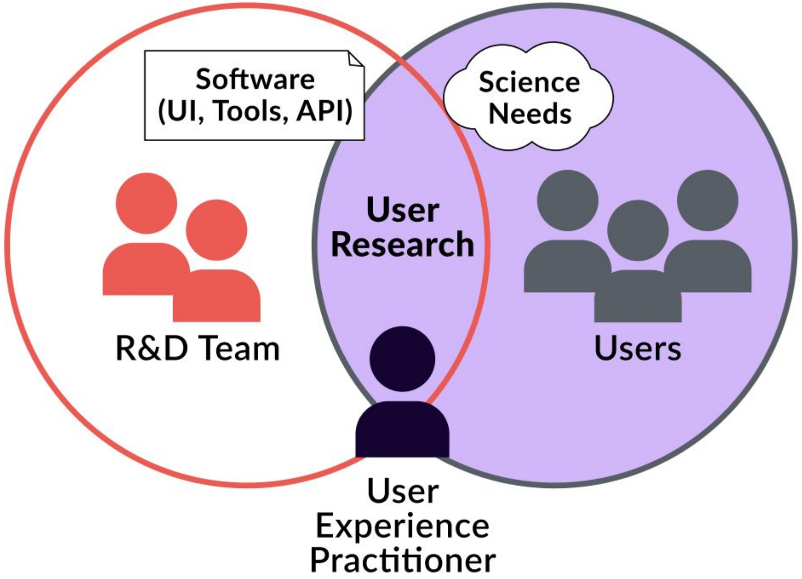


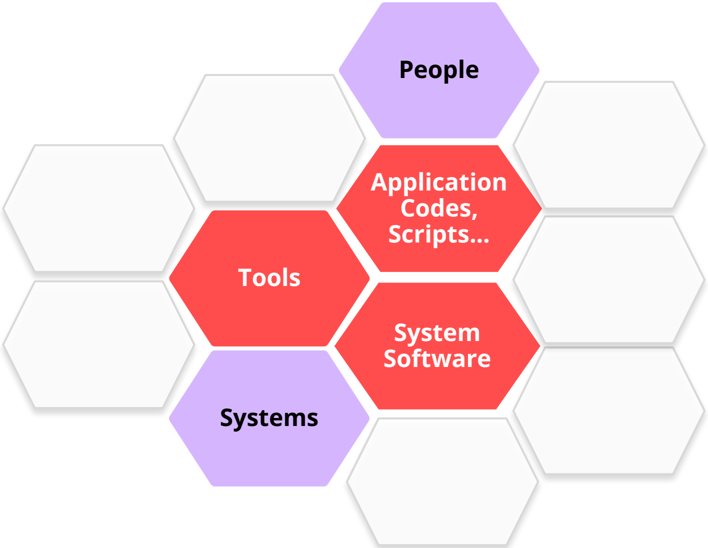
Scientific software projects involve art as much as science

Just like pastry making… such as strudel

UX involves a combination of science (well developed methods and tools) and art (intuition and adaptation in scientific contexts)

Our UX approach to addressing challenges in scientific workflows





User research gives you a **process to verify/validate your "intuition about what the user needs" (hypothesis) and convert into action**

User research processes can significantly improve the research and software outcomes

Discover Explore
- Interviews
- Contextual Inquiry, observations
- Competitive Analysis

Synthesis
- Journey Maps
- Scenarios
- Design Constraints or Considerations

Design
- Wireframes
- Detailed Mockups
- Prototypes

Usability Tests
- Interfaces, APIs

- Increased productivity for end Users
- Decreased development costs and time
- Increased adoption
- Better and/or succinct documentation and training
- Fewer errors/bugs, lower Costs

How do we define User Experience (UX)?

User experience (UX) is the **practice** of developing services & products that provide ***consistent, relevant, productive, & joyful*** experiences for users.

Misconception: UX is purely focused on graphical user interfaces.

Best Practice: UX practices are employed to shape *everything* from internal organizational processes to all varieties of user interfaces (UIs) & interactions among systems & users.

Ten Principles for Creating Usable Scientific Systems

1. Solve the right problem first
2. Understand user motivations
3. Understand the context of use
4. Validate and verify what you have heard
5. Test before building; test after building
6. Clean interfaces can't make up for bad design
7. Build for the right user (i.e., computer engineers vs scientists)
8. Understand the user's metrics (usually not performance)
9. Cost/benefit for the science team is different from the development team
10. Be willing to iterate (early and often)

#1 Source: Dula Parkinson

#2 Source: Ameriflux project

[More details at https://berkeleydata.github.io/tenprinciples/ - Ten Principles for Creating Usable Scientific Systems, Paper at eScience 2017]

Our Experiences

Time is a key factor in our optimization strategy …

Machine Time
[Optimize Perf]

Human time
[Optimize Productivity]

Allocation

Compile / Move data

Preparation

Get code running /
Manage data / Submit jobs

Jobs wait in queue

Submit to Queue

Wait for jobs

Schedule and run jobs

Execution

Wait for jobs and debug results

Move data

Post Processing

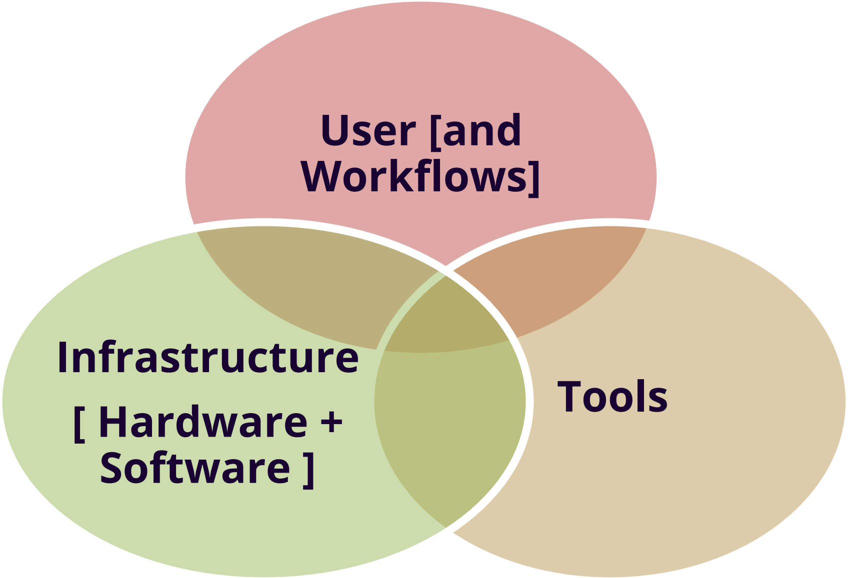
Manual data management

Considering Time in Designing Large-Scale Systems for Scientific Computing, CSCW 2016
https://dl.acm.org/doi/10.1145/2818048.2819988

User perceptions on wall clock time and queue wait time

We can't solve next-generation scientific ecosystem problems till we talk about the metric disconnect

Metric: Science result



Metrics: Performance/Efficiency

UX research highlighted how incorporating open source software in HPC environments requires strategic adaptations

Qualitative UX research in 2019-2020 investigated experiences with Jupyter on NERSC HPC systems

UXR surfaced *joyful* and *frustrating* user experiences, showed challenges & opportunities HPC environments face incorporating common open source tools

😀Streamlined JupyterLab setup makes accessing HPC resources easier & users happier

🤩Adaptations facility provided for pre-configured Jupyter kernels & python environments made for productive experience

🗃️Customized JupyterLab file system browser was small but significant improvement for users

😡Facility maintenance windows induce frustration

😠Customization of a shared Jupyter instance is tricky

😩Real time collaboration not simple or easy to accomplish

Follow on R&D work tackled these challenge!

Drew Paine, Lavanya Ramakrishnan, "Understanding Interactive and Reproducible Computing With Jupyter Tools at Facilities", LBNL Technical Report, October 31, 2020. LBNL-2001355. https://escholarship.org/uc/item/9n11k2zm

User view on abstractions and new technologies …

- **Abstractions may or may not improve usability**
  - middleware tools frequently hides the complexities
  - upon breakdown of their workflow users want to be able to see inside.
  - transparency upon demand should be a key design goal

- **Users are perpetually learning how HPC systems function**
  - changes in hardware & software configurations lead to uncertainty
  - building relationships to align scientific & computing/data worldviews is necessary to enable productive use of an HPC system

- **Adoption of new technologies**
  - differences in timelines between systems and scientific projects make it hard to leverage novel features easily
  - worked needed to adjust code/workflows is often a roadblock

Drew Paine, Sarah Poon, and Lavanya Ramakrishnan, Investigating User Experiences with Data Abstractions on High Performance Computing Systems, LBNL Tech Report, 2021

Tigres: Impact of usability study on workflow API

TABLE I: Impact of the process on the Tigres API with severity ratings [25]. The issues that were fixed during the user-centered design phase are marked as Fixed. The issues are rated as 0 -Don't think it is a usability problem, 1 - Cosmetic usability problem, 2 - Minor usability problem, 3 - Major usability problem,, needs to be fixed, 4-Catastrophic usability problem, needs to be fixed. Other issues were fixed in our first implementation.

Tigres API after usability testing	**Individual changes**	**Group-level changes**
InputTypes (name, types[ ])	Initially was called parameter_list **(3-Fixed)**	Make name optional **(1)**, Support language arrays **(2)**, Unsure how implicit data parallelism will work **(0)**, Unsure if user needs to specify O/P/s **(0)**
InputValues (name, values[ ])	Initially was called data_list **(3-Fixed)**	
InputArray (name, input_values[ ])	Initially started with set and renamed to arrays **(3-Fixed)**	
Task (name, type, impl_name, input_types, env)	Confusion over impl_name **(1)**	Make name optional **(1)**, Use of language-supported arrays rather than a new type**(2)**
TaskArray (name, task[ ])		
Sequence (name, task_array, input_array)	Allow users to not specify dependency when it is a simple sequence **(2)**	Dual syntax for dependency **(3)**
Parallel (name, task_array, input_array)	Was initially called DataParallel and it was not clear if it would handle dissimilar tasks **(1-Fixed)**	
Split (split_task, split_input_values, task_array, task_array_in)	The difference between task and task array was striking here **(1)**	
Merge (task_array, input_array, merge_task, merge_input_values)	Started with calling it Synchronization **(2-Fixed)**	

Building a Usable CLI Tool: The STRUDEL CLI

The Problem: STRUDEL has useful UI templates but accessing them requires GitHub, knowledge of the frontend architecture, and a series of error-prone copy and paste commands.

The Solution: A command line tool for generating the exact templates a user needs for their project.

Challenges

- Making the tool **easily accessible**
- Developing commands that are **easy to learn and easy to remember**
- Helping users **recover from errors**
- **Communicating** next steps to users

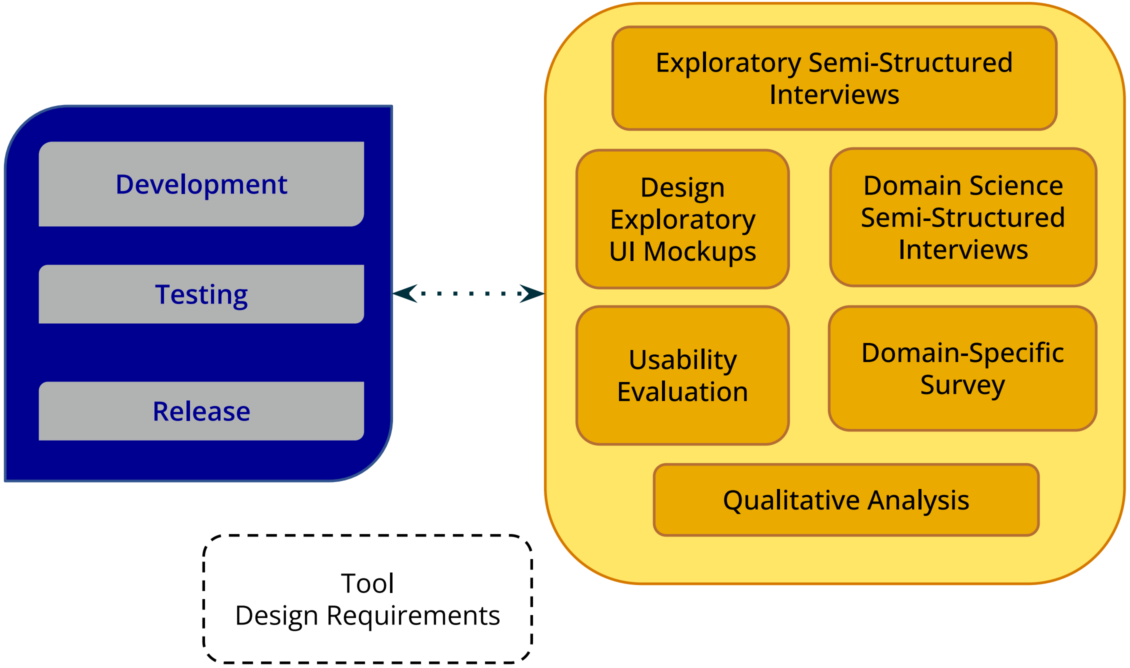
Five Key Questions

1. Where should the tool be distributed?
2. What are the commands and options available to users?
3. Do users have ways to get in-context help?
4. Is the tool robustly documented?
5. **How will people use the tool?**

We held a hackathon to observe how people used our tools on their own.

We observed that users relied heavily on default options and preferred to tinker first, configure later. This lead to key changes in our CLI tool commands.

User research methods can weave closely with the R&D process to produce better results for the project and users.



Paine, D., Ghoshal, D., & Ramakrishnan, L. (2020). Experiences with a Flexible User Research Process to Build Data Change Tools. Journal of Open Research Software, 8(1), 18. DOI: http://doi.org/10.5334/jors.284

Menti poll — Question 3 & 4

STRUDEL

STRUDEL builds on our experiences incorporating UX in many scientific projects

Providing UX as consultants, typically design or some usability research

IDAES Institute for the Design of Advanced Energy Systems

ESS-DIVE Deep Insight for Earth Science Data

NERSC

KBase PREDICTIVE BIOLOGY

Incorporating UX as key part of <u>our</u> R&D Projects

Deduce

DUA

SCIRA

Tigres

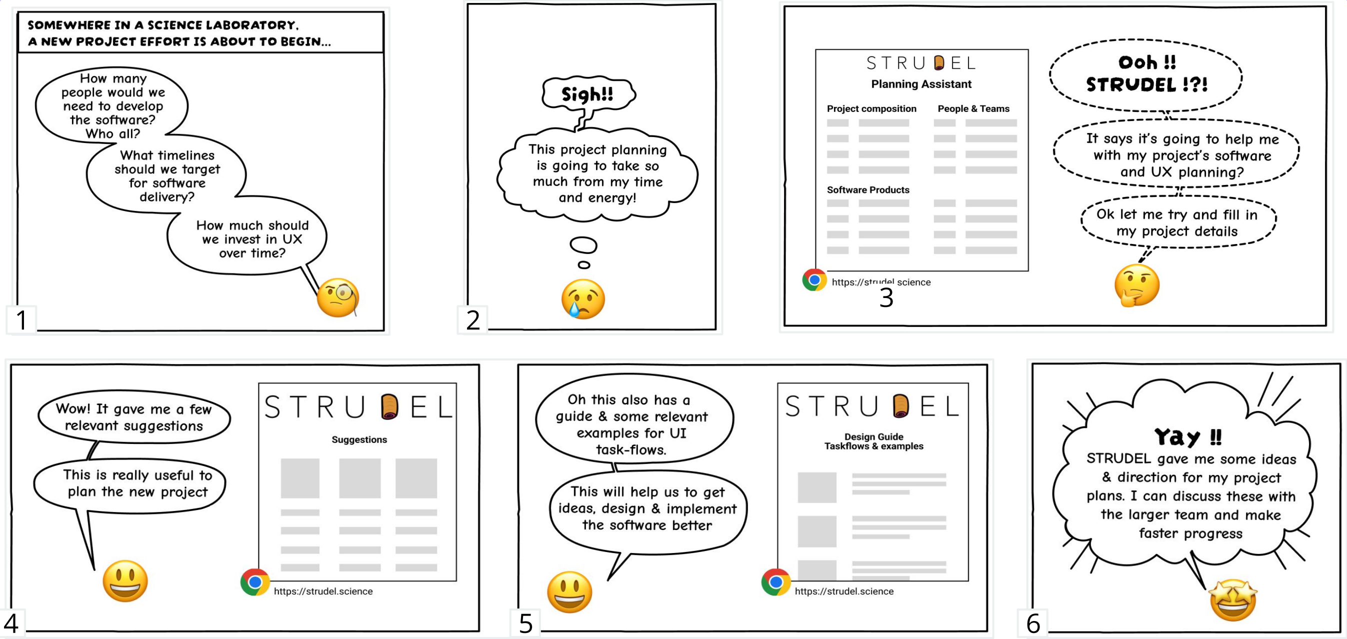
Science Capsule

Systematically expanding & abstracting insights from this repeated work

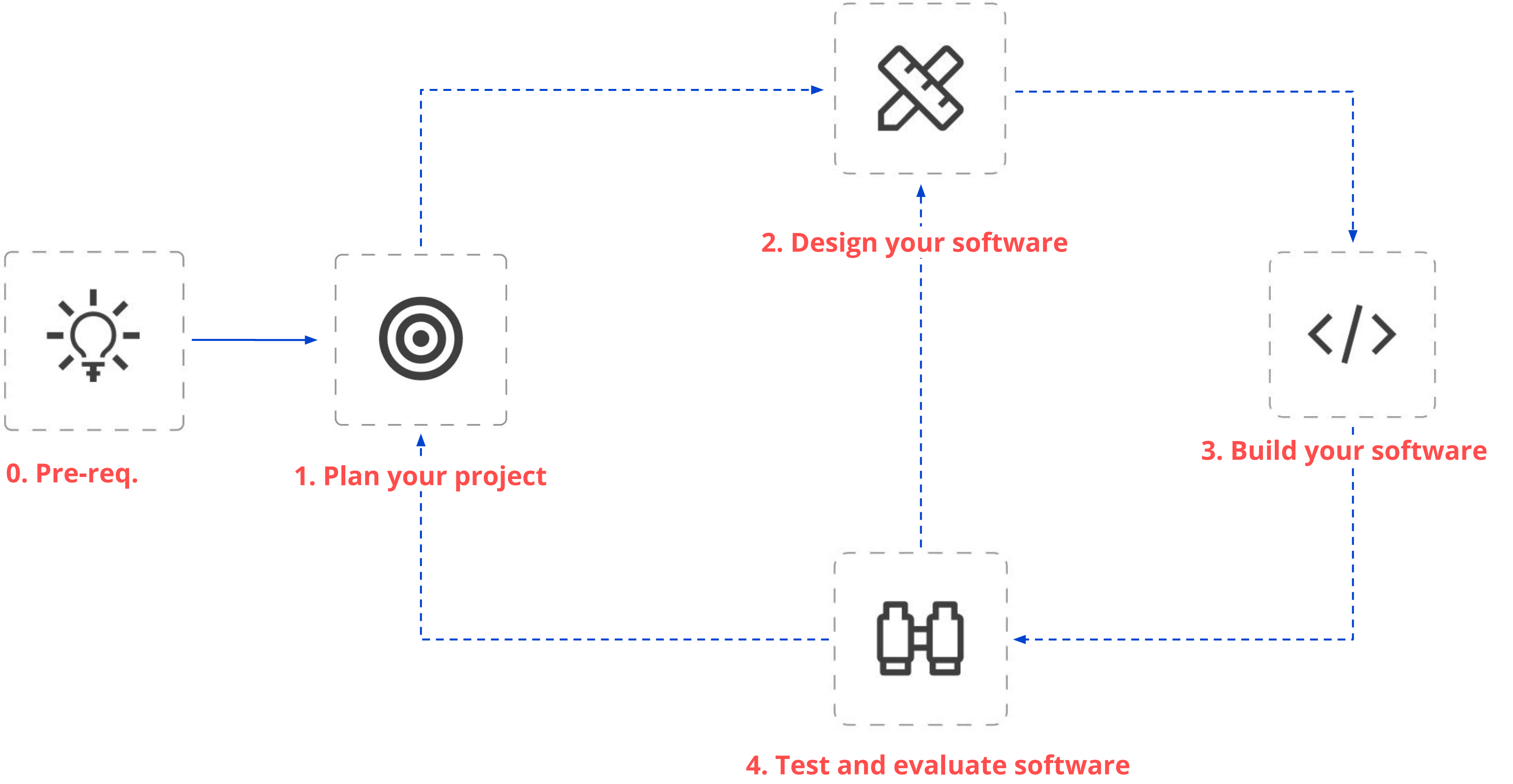
STRUDEL

The long-term STRUDEL vision

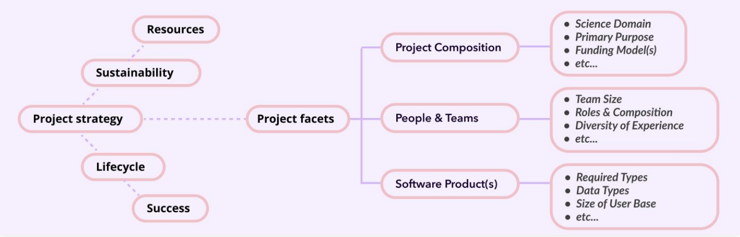
Our aim is to develop products that help scientific software teams simplify adoption of UX approaches to enable more usable, sustainable software.

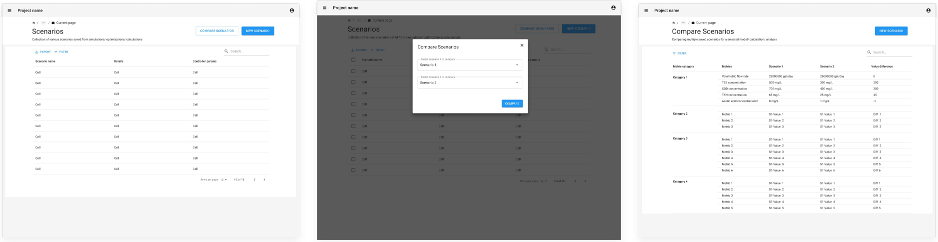


Scientific software design life cycle



STRUDEL: Open source project with two key products





Typology of Scientific Software
informing a strategic
Planning Framework

Design System
with
Task Flows

Supporting the scientific software development life cycle

0. Pre-reqs.
1. Plan your project
2. Design your software
3. Build your software
4. Test and evaluate software

Categorizing Patterns in Scientific (Software) Work

Relevant Stages

0. Pre-reqs.

1. Plan your project

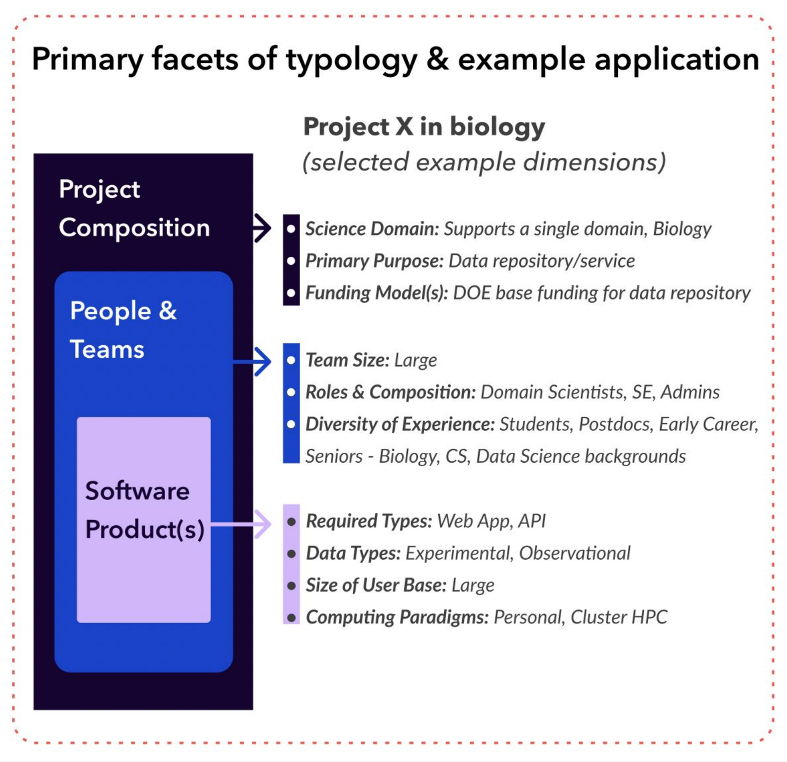
2. Design your software

Today

Typology is a first attempt to categorize questions & concerns we have seen repeatedly across projects, environments, etc.

Tomorrow

Crafting a strategic **Planning Framework** from this categorization & resources to enable better project planning & software design



Design system

A design system is a set of reusable components and patterns for designing and building UIs as well as guidelines on when and how to use them.

What is unique about the STRUDEL design system?

Designed specifically for scientific UIs.

Enables building UIs applicable across different scientific domains

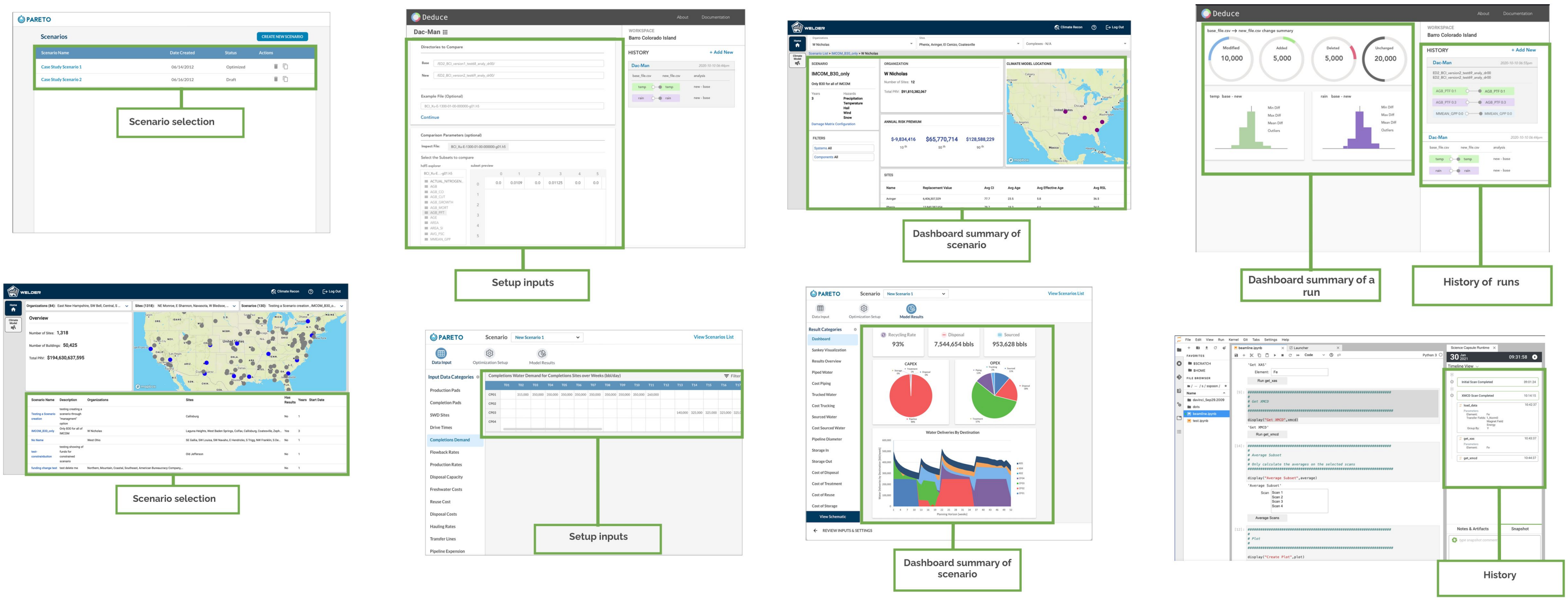
Focuses on the larger flow & function of UI

Gives you a jump start to think about entire UI flow rather than starting from scratch

Designed by experts for experts.

Informed by over a decade of collective UX experience in the sciences and democratizes good UX practices

Identifying Task Flows From Common UI Needs



Scenario Selection

Select Inputs

Dashboard summary of results

History

Task Flows

Relevant Stages

2. Design your software

3. Build your software

Task Flow: series of steps represented by screens which helps user to accomplish particular task in the scientific software's user interface

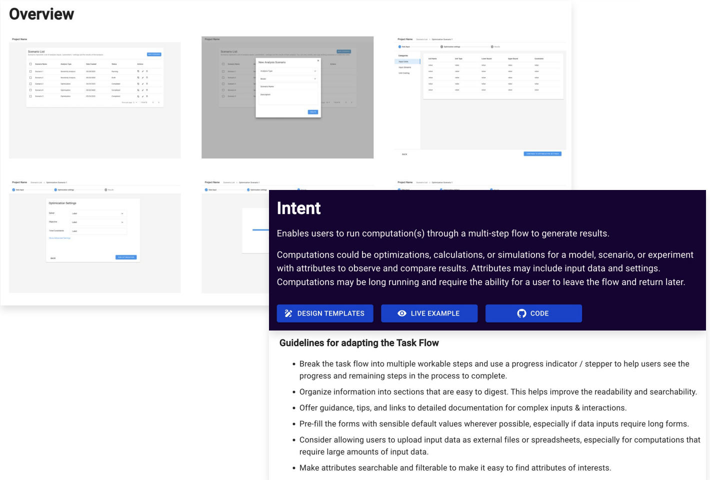
Similar Task Flows exist across various types of scientific software.

Analysis	Data	Exploration	Community Contributions
Run Computation	Explore Data	Monitor Activity	
Run Interactive Computation	Explore Data Repositories	Track State	
Compare Data	Contribute Data	Manage Account	

Task Flow Resources

Design templates & guidelines for the series of steps involved in the Task Flow.

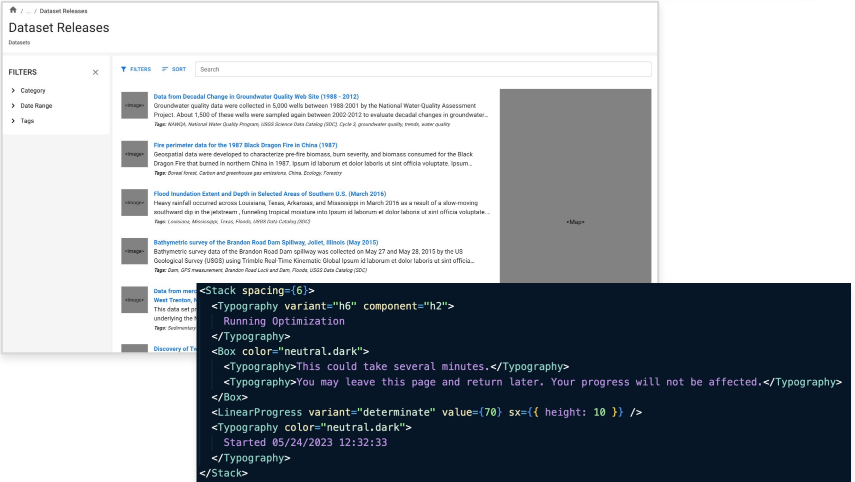
These templates are available as images and as design files on **Figma community** for customizing designs.



strudel-kit

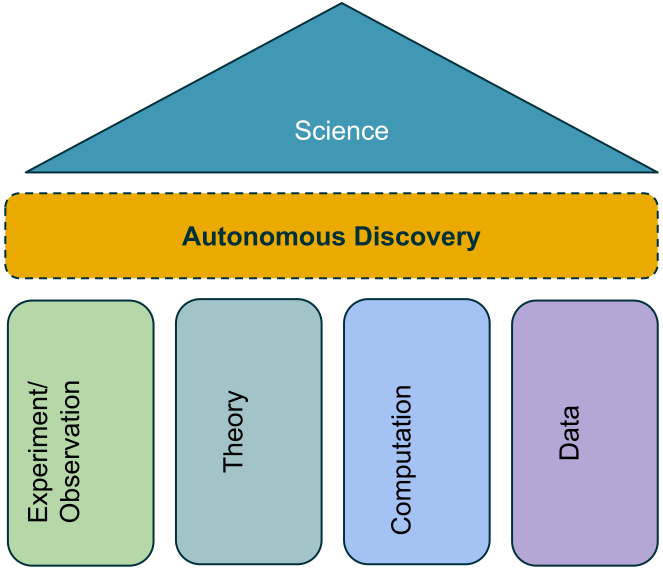
Web interactive templates and coded UI library for high level components & task flows from our design system.

Uses **React javascript framework** and is built on top of the popular Material UI (MUI) components library



Looking Forward

Software is ubiquitous and critical to scientific research.



Consortium for the Advancement of Scientific Software

Fostering collaboration across a diverse collection of Software Stewardship Organizations (SSOs)

CASS

Software requires ongoing usability and user experience (UX) improvements in order to be a reliable, sustainable resource for user communities.

Planning, design & stewardship of scientific software often *tumultuous, even chaotic*

Individuals often fulfill roles that are varied, multifaceted

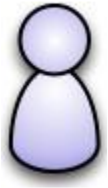
Never enough resources (time, $$ people)

Management & planning can be ad hoc responding to emerging scientific demands and needs

UX often an afterthought at best

Uncommon is an industry-like *Product Management* role who stewards vision, user engagement, etc.

Democratization of skills is critical for future software

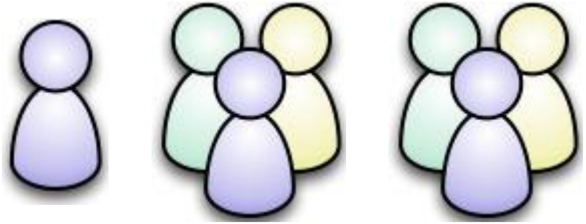


Single person teams

Small teams

Medium teams

Scientific teams are often resource-constrained and people's roles don't always match their training



Large teams

Software Design Styles

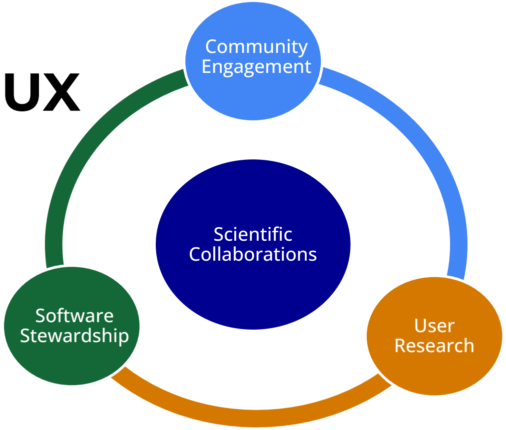
- Unintended Design
- Self Design
- Genius Design
- Activity Focused Design

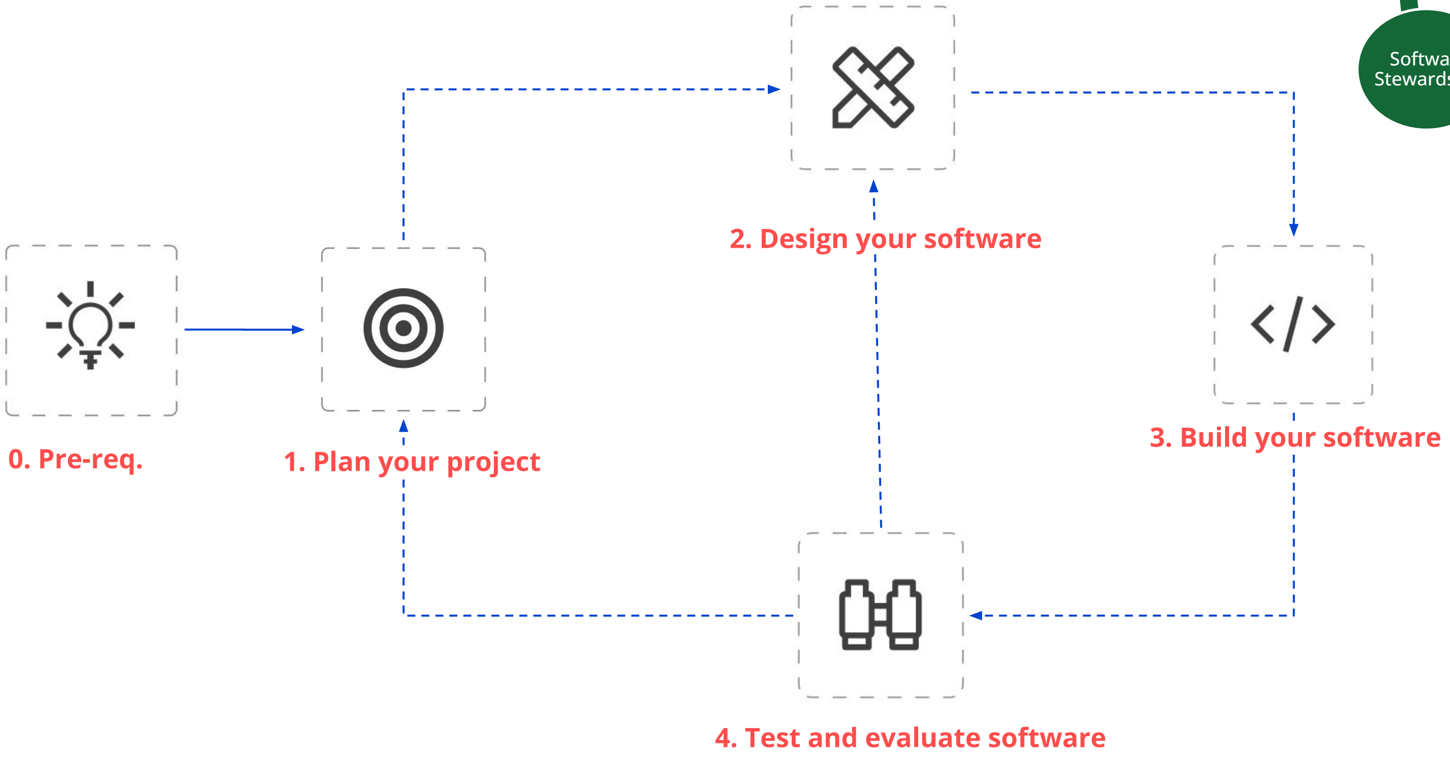
User Focused Design

Need different styles for different projects
but experience in user focused design helps

Source of classification: https://articles.uie.com/five_design_decision_styles/

Software Stewardship will greatly benefit from UX





Some future questions

- How is the right framing that will let us think about software sustainability?
- How do we democratize user research and software sustainability principles?
- How do we measure the success of software sustainability and user experience research?
- How do we organize and structure teams to ensure great software outcomes?
- How do we build community?
- How do we scale up UX efforts from in depth single qualitative studies to quantitative macro studies?

Key Takeaways

User experience and software sustainability are closely tied to ensure successful software

User research processes can significantly improve the research and software outcomes

Get Involved!
Join the STRUDEL Community

Visit our website to learn more & use our products!

https://strudel.science

Have comments?
Start a conversation on our GitHub
https://go.lbl.gov/strudel-discussion

Join our mailing list to keep up to date & contribute to the community!

strudel-community+subscribe@lbl.gov

Join the US-RSE User Experience working group to connect with the larger community of practice!

#wg-ux on the **US-RSE Slack**
https://go.lbl.gov/usrse-uxwg

Visit our website to learn more

https://ux.lbl.gov

Thank you!