Designing locally usable and meaningful technology

Magnus Li
Joe Cooper
DHIS2 Annual Conference 2019
Us

Magnus Li
B.Sc. & M.Sc. in Information Systems (UiO)
PhD research fellow (UiO)
DHIS2 Design Lab lead
magl@ifi.uio.no

Joe Cooper
DHIS2 core UX designer
joe@dhis2.org
Content

1. Fundamental concepts (use, user, usability, utility, user experience)
2. Understanding and involving users (aim, approaches, methods)
3. Design and localization of Generic Software
4. The generic-level design for end-use of DHIS2 (Joe)
5. Supporting implementation-level design of DHIS2
6. Group discussions
Fundamental concepts

• **Usability** = quality attribute that assesses how easy user interfaces are to use. (Learnability, Efficiency, Memorability, Errors, Satisfaction)

• **Utility** = whether the system is useful to certain users. That is, provides the functionality that they need / that are valuable to their work.

• **Meaningful** = usability + utility

• **User experience** = all aspects of the end-user's interaction with the company, its services, and its products.

• **Usability design** = the design process aimed at assuring or improving usability
‘Use’

• “Use” in our context refers to utilizing some kind of technology (such as DHIS2) to achieve some goal.

• Use of one tool / software always unfold in a network or ecosystem of tools, routines, people, communication mechanisms, etc..
User and end-user

- With large organizational systems, there are often a variety of user groups that use the system. For instance;
  - Maintenance / implementers (technical).
  - Health / program managers (on different levels).
  - Data entry clerks.
  - Doctors and nurses.
  - Community health workers.
  - Patients?

**Decreasing amount of:**
- Training attended?
- Interest in, and time for using and learning the system?
- Participation in requirement definition when system is developed?
Usability is a relational phenomenon

• Whether a system is usable depends on who’s using the system
• What’s usable for someone in a certain context may be unusable to other people or in other contexts.
Usability is a relational phenomenon

• Pending on, for instance:
  • Established practices
  • Domain terminologies
  • Technical knowledge
  • Other tools and technologies
  • Environment of use
    • In office on desktop computer
    • On smartphone as community health worker in rural areas
    • While interacting with patients under time pressure.
Matching the users’ mental models

Implementation Model
reflects technology

Represented Models
worse better

Mental Model
reflects user’s vision

(A. Cooper et. al, 2007)
Usability heuristics (Nielsen, 1998)

• **Visibility of system status**
  The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

• **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.

• **Consistency and standards**
  Users should not have to wonder whether different words, situations, or actions mean the same thing.
Usability heuristics (Nielsen, 1998)

• **Recognition rather than recall**
  Minimize the user's memory load by making 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.

• **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.

• **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.
Example of mismatch between system and real world

Periods implemented in DHIS2

Actual reporting periods
Example of mismatch between system and real world

• Data set
• Organizational unit
• Clear Cache
• Tracked entity
• ??
Understanding and involving end-users
Gap between designers and end-users

Understanding and involving users in the design process is crucial for making locally meaningful technology.
Classic methodologies for understanding or involving users

• **Participatory Design (PD) (Scandinavian tradition).** Emphasize highly participative design process for empowerment.

• **User/Human-centered design (UCD).** Emphasis on understanding users and evaluating prototypes with users to ensure relevant and usable systems.

• **Activity-centered design (ACD).** Emphasis on understanding the activities of the end-users to make software that fit into their established practices.

• **Scenario-based design (SCD).** Based on the development of problem and solution scenarios that are build through investigation and involvement of end-users. The scenarios are used to communicate between actors, and to inform design of prototypes.
Reasons for involving users seen in ICT4D projects

• Empowerment (PD)

• Advertisement (claiming PD, UCD etc.)

• Increase usability and utility for end-users (UCD, PD, ACD, SCD)

• Situated innovation (UCD, ACD, SCD)
Understanding and involving end-users in the design process

• In general, for ‘local’ usability and utility, the rationale is to create more usable and meaningful systems by:

  a) **Understanding** end-users, their established practices, activities, and context of use.

  and/or

  b) **Involving** end-users in the process of design to participate in decisions.

And then reflect their mental models and needs in the design of user interfaces and functionality
Understanding and involving end-users in the design process

Understanding practice

Observe, experience, understand the activities, practices and context of the user to inform design

Involving users

Let the users participate in decisions.
Typical elements of a design process

Understanding practice

Analyzing learnings

Building prototypes

Evaluating with users and actors

Working system
Understanding practice – some typical methods

• Focus groups and interviews
• Contextual inquiry
• Participatory observation
Understanding practice – Focus groups and interviews

• **Focus groups** involve gathering several users in a group focused around topics, such as:
  • How they work (activities, tools, processes)
  • Challenges they face
  • Collaborative prototyping / generating ideas for solutions

• **Interviews** involve talking to one or several users in an open-ended or planned session.
Understanding practice – Contextual inquiry

• **Contextual inquiry** involves sitting together with end-users, observing how they work while asking questions.

• For instance, ask them to perform a typical work activity and observe what they do. Ask when something is unclear or needs elaboration.

• Powerful approach to understand how things are done and issues faced.
Understanding practice – Participant observation

• **Participant observation** involves taking part in the end-users activities.

• Doing tasks and activities

• Testing technologies, tools and software.

• Gives in-depth experience of the future use-context of technology.
Typical elements of a design process

Understanding practice

Analyzing learnings

Building prototypes

Evaluating with users and actors

Working system

Making locally meaningful technology - the DHIS2 Design Lab
Analyzing learnings – some typical techniques

• Scenarios
• Personas
• Activity diagrams
• Hierarchical task analysis
Analyzing learnings – hierarchical task analysis

- Hierarchy of sub-tasks included in a task
- May help identify a) what people do, b) how a tool is related to others, c) where problems are, d) where to potentially improve something

1. Enter monthly report

1.1 Log in

1.2 Open data entry

1.2.1 search for app

1.2.2 click app

1.3 find form

1.3.1 select org.unit

1.3.2 select program

1.3.3 select period
Typical elements of a design process

Understanding practice

Analyzing learnings

Building prototypes

Evaluating with users and actors

Often a iterative process from low-fidelity to high-fidelity

Working system

Making locally meaningful technology - the DHIS2 Design Lab
Prototyping and evaluating

Prototyping

Evaluating
Typical elements of a design process

Understanding practice

Analyzing learnings

Building prototypes

Evaluating with users and actors

Often an iterative process from low-fidelity to high-fidelity → Through processes of evaluation with users

Working system
Design ‘never’ ends
Design and Localization of Generic Software
Generic software and usability

• *Generic* software is different from *bespoke* software.
• It is developed to serve different users in different organizations.
• Makes design for usability challenging

<table>
<thead>
<tr>
<th>Usability</th>
<th>Generic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitive to <strong>specific</strong> users practices and context of use</td>
<td>Emphasise variety of users in different organizational contexts</td>
</tr>
</tbody>
</table>
Dimensions of genericness and usability

Global

Audience

Community specific

Appliance

Purpose

Multi-purpose

(M. Li, 2019)
Dimensions of genericness and usability: DHIS2

![Diagram showing the dimensions of genericness and usability for DHIS2]

- **Global** vs. **Community specific**
- **Appliance** vs. **Multi-purpose**
Addressing usability in generic software

• A joint effort between the generic-level designers (i.e., the DHIS2 core developers), and the implementation-level designers (i.e., software implementers)

• Generic-level designers provide
  a) user interfaces and functionality for end-use on typical use-cases
  b) Adaption features in the software to facilitate localization during implementation

• Implementation-level designers utilize adaption features to localize the software based on specific organizational needs.
Addressing usability in generic software

Generic-level design
(DHIS2 ‘core’ developers)

Design for implementation-level design

Implementation-level design
(Implementers)

Design for end-use
generic UI for end-use

End-use

Design for end-use
localized UI for end-use

(Making locally meaningful technology - the DHIS2 Design Lab)

(Li & Nielsen, 2019)
Addressing usability in generic software

**Generic-level design**
(DHIS2 ‘core’ developers)

**Design for implementation-level design**
- Adaptable software
- Supporting artifacts
- Soft resources

**Implementation-level design**
(Implementers)

**Design for end-use**
- localized UI for end-use

**End-use**

**Design for end-use**
- generic UI for end-use
Generic-Level Design at DHIS2
Our process

1. Find out requirements/needs of users
2. Understand requirements, translate into user stories
3. Find the balance
4. Prototype ideas
5. Iterate
6. Remain open for feedback
1. Find out requirements

- Simple, fast data entry
- Multi-add forms
- Search controls
- Enrollment summary
- TEI notes
- Inline feedback
- Custom forms
- Log glass breakage
- Check for enrollment
- Batch editing
- Form/table input toggle
- Sort/filter log events
- Keyboard navigation
- Attach reason to log events
- Inline table row edit
# 2. Requirements → User Stories

<table>
<thead>
<tr>
<th>Data Entry</th>
<th>Person Management</th>
<th>Audit Management</th>
<th>Case Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple, fast data entry</td>
<td>Search controls</td>
<td>View audit log in modal</td>
<td>Form/table input toggle</td>
</tr>
<tr>
<td>Keyboard navigation</td>
<td>TEI notes</td>
<td>Log glass breakage</td>
<td>Inline table row edit</td>
</tr>
<tr>
<td>Custom forms</td>
<td>Enrollment summary</td>
<td>Attach reason to log events</td>
<td>Batch editing</td>
</tr>
<tr>
<td>Inline feedback</td>
<td>Check for enrollment</td>
<td>Sort/filter log events</td>
<td>Multi-add forms</td>
</tr>
</tbody>
</table>
3. Find the balance
4. Prototype ideas
5. Iterate
6. Remain open
Example: Organisation Unit Tree
Example: Organisation Unit Tree
Example: Organisation Unit Tree

Feature: Select-able organisation units

When enabled, users can select organisation units. Depending on the configuration, they can either select one organisation unit only or as many as they want.

Scenario: All organisation are selectable
  Given selecting all units is enabled
  When an organisation unit tree is rendered
  Then the label will contain a checkbox

Scenario: Selecting an organisation unit when all select-able
  Given selecting all units is enabled
  When the user selects an organisation unit
  Then the organisation unit's checkbox is checked

Scenario: Selecting a descendant organisation unit
  Given selecting all units is enabled
  And an organisation unit has descendants
  When selecting a descendant
  Then the organisation unit is marked as indeterminate

Scenario: Deselecting an organisation unit
  Given selecting all units is enabled
  And an organisation unit is selected
  When deselecting the organisation unit
  Then the checkbox is unchecked

Scenario: Deselecting a descendant
  Given selecting all units is enabled
  And an organisation unit has descendants
  When deselecting a descendant
  Then the checkbox is unchecked
Example: Organisation Unit Tree

Selected Organisation Units
- Clinics in Bombali
- Bonthé
- Chiefdoms in Bomali

Prototype controls

Filtering
- Filter with text "Kap" (results on 1 level)
- Filter with text "Bo" (results on multiple levels)
- Filter with text "Biriwa" (parent folder result)
- No filter results

Relative org. units
- Open relative org. unit selection menu
- Relative units selected (tree disabled)

Selection
- Open a group/level selection menu
- Open a children selection menu
- Select all clinics in Bombali
- Select all clinics, chiefdoms in Bombali and Bonthé as a single unit
- Recursive selection of all children
- Single selection tree
Example: Data Visualization Guidance
### Example: Data Visualization Guidance

**Table 3.** Frequency of dashboard problems

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context problem</td>
<td>61 (73.5%)</td>
</tr>
<tr>
<td>Dashboard layout problem</td>
<td>35 (42.2%)</td>
</tr>
<tr>
<td><strong>Visualization technique problem</strong></td>
<td><strong>69 (83.1%)</strong></td>
</tr>
<tr>
<td>Logical problem</td>
<td>47 (56.6%)</td>
</tr>
<tr>
<td>Data quality problem</td>
<td>28 (33.7%)</td>
</tr>
</tbody>
</table>

Assessing User-Designed Dashboards: A Case for Developing Data Visualization Competency, Chrysantina, Aprisa (et. al)
Example: Data Visualization Guidance
Example: Data Visualization Guidance
Example: Data Visualization Guidance

- Column
- Stacked column
- Bar
- Stacked Bar
- Line
- Area
- Pie
- Radar

*Good for comparing changes over time*
Example: Data Visualization Guidance

- March 2019
  - 89.7 (22.6 %)

- Is this the right chart type?
  
  A pie chart isn’t the best way to visualise time based comparisons. Try a column chart instead.

- Use column chart  Dismiss
Example: Tracker Capture
Example: Tracker Capture
Example: Tracker Capture
Example: Analytical Expressions
Example: Analytical Expressions

```
IF(#{MNG001.sum().period(-2, -1)} >= #{MNG002.sum().period(-2, -1)) AND (#{STK001.average().period(-6, -1)} != #{STK002.average().period(-6, -1))})
```

Diagram:
- Meningitis diagnoses
  - Sum
  - Periods: -2, -1
- Meningitis vaccinations
  - Sum
  - Periods: -2, -1
- AND
- Meningitis vaccine stock available
  - Average
  - Periods: -6, -1
- Meningitis vaccine stock missing
  - Average
  - Periods: -6, -1

DHIS2 branding
Example: Analytical Expressions

If the weeks between `imp` and `current_date` are less than 50 and `imp` has a value and `currentPregnancyOutcome` has a value.

Output: `valid`
Example: Analytical Expressions

(d2:weeksBetween(#{lmp}, V{current_date}) < 50 && !d2:hasValue('lmp') && d2:hasValue('currentPregnancyOutcome'))

- **Output**: If weeks between LMP and current date are less than 50 and LMP has a value and current pregnancy outcome has a value.

- **Functions**:
  - d2.ceil(<number>)
  - d2.floor(<number>)
  - d2.round(<number>)
  - d2.modulus(<number>, <number>)
  - d2.zing(<number>)
  - d2.oizp(<number>)
  - d2.daysBetween(<date>, <date>)
  - d2.weeksBetween(<date>, <date>)
  - d2.monthsBetween(<date>, <date>)
  - d2.hasValue(<sourcefield>)
Promoting usability with a system
## Promoting usability with a system

<table>
<thead>
<tr>
<th>Type</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>Basic</td>
</tr>
<tr>
<td>Primary</td>
<td>Primary</td>
</tr>
<tr>
<td>Secondary</td>
<td>Secondary</td>
</tr>
<tr>
<td>Destructive</td>
<td>Destructive</td>
</tr>
<tr>
<td>Dropdown</td>
<td>Dropdown +</td>
</tr>
</tbody>
</table>
Promoting usability with a system
Promoting usability with a system

Usage

Alert bars notify a user of some information. There are different types of alert bar for displaying different types of content. Use the alert bar type that matches your content type and importance. Note that alert bar can be ignored by the user, so they shouldn't be used for content that needs to block an app flow, use a modal instead.

Alert bars are always displayed at centered at the bottom of the screen. Some types of alert bar dismiss after a set time, others must be dismissed by the user.

To summarise the different types of alert bar available:

<table>
<thead>
<tr>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>Default. Inform the user of neutral content.</td>
</tr>
<tr>
<td>Success</td>
<td>Confirm a successful action.</td>
</tr>
<tr>
<td>Warning</td>
<td>Warn of a potential problem or obstruction, usually before it has happened.</td>
</tr>
<tr>
<td>Critical</td>
<td>Inform of a catastrophic error or problem, after it has occurred.</td>
</tr>
</tbody>
</table>

Using the correct type

It is important to use the correct type of alert bar. Users should not be overloaded with warnings that are not actually important, nor should they miss out on important information.

Info

“Antenatal Care Visit” dashboard updated.
Promoting usability with a system

DHIS2 Design System

This is the temporary home for the DHIS2 Design System while it is under development. The design system consists of principles to guide design of DHIS2 applications, as well as a library of flexible, reusable UI components.

Design Principles

- Typography
- Color
- Spacing, Alignment, Stacking
- Content & Communication
- Icons

https://github.com/dhis2/design-system
Future Collaboration
Future Collaboration

• Bad timing. Feedback coming from a visit right after we finish building something. Not aligned.

• Too specific. The implementation-level requirements are of course specific, sometimes that means adding too much complexity. By default we would not be generic anymore.

• Not enough feedback. We might make several different prototypes, but it can be hard to locate the people best placed to give feedback. Access to the end user is hard.
Future Collaboration

• Bad timing. Feedback coming from a visit right after we finish building something. Not aligned.

• Too specific. The implementation-level requirements are of course specific, sometimes that means adding too much complexity. By default we would not be generic anymore.

• Not enough feedback. We might make several different prototypes, but it can be hard to locate the people best placed to give feedback. Access to the end user is hard.
Future Collaboration

• Bad timing. Feedback coming from a visit right after we finish building something. Not aligned.

• Too specific. The implementation-level requirements are of course specific, sometimes that means adding too much complexity. By default we would not be generic anymore.

• Not enough feedback. We might make several different prototypes, but it can be hard to locate the people best placed to give feedback. Access to the end user is hard.
Future Collaboration

• Bad timing. Feedback coming from a visit right after we finish building something. Not aligned.

• Too specific. The implementation-level requirements are of course specific, sometimes that means adding too much complexity. By default we would not be generic anymore.

• Not enough feedback. We might make several different prototypes, but it can be hard to locate the people best placed to give feedback. Access to the end user is hard.
Usability as a priority
Supporting implementation-level design & the DHIS2 design lab
Addressing usability in generic software

• As an implementation-level designer, you are faced with end-user requirements on one hand, and the software to be ‘shaped’ on the other.

• The localization-related resources provided by the generic-level designers are thus important.
Implementation-level design

**Generic-level designers**

‘DHIS2 core developers’

- Change requests (e.g., Jira), or support

**Adaption approaches**

- Configuration
- Customization
- Extension

**Client**

- Project managers
- Users

**Implementation-level designers**

E.g., HISP Tanzania

**DHIS2**

Negotiating requirements

“Good cop, Bad cop”

Making locally meaningful technology - the DHIS2 Design Lab
Implementation-level design

• Typical design-methods are not aligned with the adaption capabilities of generic software

• Also, to be usable (and sustainable beyond research projects), they need to be time, resource and competence effective.

• We often see that design methods are unattractive to developers, implementers and client project managers.

• Further, when we try to apply them, we often end up with «fighting the software constraints»
Difficult to solve usability problems

Periods implemented in DHIS2

Actual reporting periods
Implementation-level design: adaption features

- **Configuration** (default options in the software, e.g., org.units, data sets, data elements, form builder)
- **Customization** (changing the core source code)
- **Extension** (building add-ons or third-party apps through the APIs)
Implementation-level design: **adaptation approaches**

<table>
<thead>
<tr>
<th></th>
<th>Configuration</th>
<th>Customization</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time and competence</strong></td>
<td>Fast, easy</td>
<td>Competence intensive (fast or slow)</td>
<td>Competence and time intensive</td>
</tr>
<tr>
<td><strong>Upgrades</strong></td>
<td>Few upgrade problems</td>
<td>Always upgrade problems</td>
<td>Limited to API changes</td>
</tr>
<tr>
<td><strong>Design flexibility</strong></td>
<td>Limited</td>
<td>High (ish)</td>
<td>High</td>
</tr>
</tbody>
</table>
The DHIS2 Design Lab

• The new initiative, DHIS2 Design Lab aims to strengthen the resources provided to implementation-level designers, to better support usability-related design on the level of implementation.

• We refer to these resources as a «design infrastructure»

• The *infrastructure* consist of
  • Adaption features in the DHIS2 software
  • Other artifacts that supports implementation-level design
  • Soft resources such as guidelines, documentation and tutorials.
The DHIS2 Design Lab

- Learning from and contributing to
- Dynamic process
- Design and use of

Theory & research

Design lab

Generic-level design

Design-infrastructure

Implementation-level design
(some of) or current projects

<table>
<thead>
<tr>
<th>Adaption</th>
<th>Tools</th>
<th>Design methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>- DHIS2 UI library</td>
<td>- DHIS2 prototyping tool</td>
<td>- Resource page with tested design-methods</td>
</tr>
<tr>
<td>- Shared web-app component library</td>
<td></td>
<td>- Aligned with adaption features of the software</td>
</tr>
<tr>
<td>- Terminology translator</td>
<td></td>
<td>- Frugal in terms of time, resources, competence</td>
</tr>
<tr>
<td>- Drag and drop form builder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- DHIS2 dashboard widget resources</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Making locally meaningful technology - the DHIS2 Design Lab
Readings
Further readings

Books (usability-related design in general)

• The Design of Everyday Things (Norman, 2013)
• Interaction Design (Sharp, Preece, and Rogers, 2015)
• Usability Engineering - Scenario-based development of human-computer interaction. (Rosson and Carroll, 2002)

Articles (on supporting and conducting implementation-level design within generic software / DHIS2)

• Making Usable Generic Software – A Matter of Global or Local Design? (Li and Nielsen, 2019)
• Making Usable Generic Software – The Platform Appliances Approach (Li, 2019)
• A Design Approach to Addressing Usability in Generic Software (Li, 2019)
Group discussions
Group discussions

• We have prepared some questions

• We want you to discuss in smaller groups (3-5)

• For each topic, we want you to discuss for 15 minutes, and then present your main points to us all.
Group discussion (15 min)

1. Have you experienced usability issues in any implementation-project?
2. What are your practices related to design and usability during DHIS2 (or other software) implementation?
   A. Understanding end-users’ practices
   B. Involving end-users
   C. Analyzing experiences, prototyping and evaluation
3. Are you (or your colleagues) using any of the methods and techniques discussed in this session? Why/why not, and what are the potential issues with applying them?
4. What kinds of approaches do you find working and what does not work? Why?
Group discussion (15 min)

1. What experiences do you have with the design infrastructure of DHIS2?

That is:

- Adaption features and approaches (configuration, customization, extension)
- Supporting resources (documentation, guidelines, etc.)

2. What is lacking and what could help strengthen the implementation-level design process? (to strengthen local usability and utility for end-users)