Two problems in large-scale / generic software development:

- "Generic" usability
- Working with local users in development

We will look at:

1. Usability.
2. User participation in design.
3. Our two problems.
4. Participation and scale. Four types of participatory design.
5. Architectures for participation and local adaptations.
6. IN5320 individual assignment award 2018!
Usability and participation
Usability

How well a system works for the user

Do the system allow the intended users to certain goals with

- Effectiveness (doing the right things)
- Efficiency (doing things right)
- Satisfaction
Usability - Nielsen’s 10 heuristics

- Visibility of system status
- **Match between system and the real world**
- User control and freedom
- Consistency and standards
- Error prevention
- **Recognition rather than recall**
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognize, diagnose, and recover from errors
- Help and documentation
Different mental models

System oriented versus real-world oriented language

DHIS2 concepts

Data Entry

- Organisation Unit: Sierra Leone
- Data Set: [Select data set]
- Period: [Select period]
Usability - Donald Normans 6 principles

- Affordances and signifiers
- Mapping
- Consistency
- Constraints
- Feedback
- Visibility
Hva betyr det da den gensern der bare står å blinker å vaskemaskina gir faen i alt? Får ikke opp døra eller noe..
Usability

Not just defined by qualities of the software, but dependent on a **specific set of users**
Mental models

To make systems intuitive and usable, designers must try to create interfaces between technology and the user that are close to their mental model.
Developers, designers, and end-users may have radically different understandings of the world.
Design - actuality gaps

Objectives and values
Processes
Technology
Information
Management systems and structures
Investment resources
Staffing and skills
Milieu

Organisational Change Design

Heeks 2002

Reality

Gap
Different mental models
Different mental models
Usability

Not just defined by qualities of the software, but dependent on a specific set of users in a certain context of use.
To ensure usable systems, we must understand the users and context that we design for.

A common way to do this is to:

1. Investigate the context
2. Involve end-users in the development process.
User participation in design

Several traditions. Two common are.

- User/human-centered design (UCD).
- Participatory design.
User participation in design

Different levels of participation.

**Informative**
Users provide and/or receive information

**Consultative**
Users comment on a predefined service or range of facilities

**Participative**
Users influence decisions relating to the whole system

*Figure 3-2 Forms of user involvement (Damodaran, 1996, p. 365)*
Challenge of scale
Challenges in large-scale systems development

Two problems in large-scale / generic software development:

1. “Generic” usability

2. Working with local users in development

What if we are developing software to be used by very different users in very different contexts?
Challenges of large-scale systems development

Example 1 - Standardized Patient Journal system

Implementing one common patient journal across different departments, hospitals, regions.
Challenges of large-scale systems development

Work practices, routines, language / semantics, culture, norms, legacy systems, dependencies etc.
Challenges of large-scale systems development

Not just defined by qualities of the software, but dependent on a specific set of users
Challenges in large-scale systems development

- Usability is dependent on the users
- Who is it usable for?
- Who to involve in design?
Example 2 (Rolland & Monteiro, 2002) - Large shipping survey company:

Implementing one common system to support surveying across 300 sites in over 100 countries.
# Challenges of large-scale systems development

## Example 4 - HISP / DHIS2

<table>
<thead>
<tr>
<th>Timeline</th>
<th>Stage</th>
<th>Use and Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994-1999</td>
<td>Pilot and national system</td>
<td>Users and collocated software developers, all in South Africa, network of users</td>
</tr>
<tr>
<td>2000-2004</td>
<td>Expansion</td>
<td>Multiple countries, core development isolated from local modifications</td>
</tr>
<tr>
<td>2004-2007</td>
<td>Technological transition</td>
<td>Two branches (v1 &amp; v2), infrastructure for sharing, but fragmented processes, isolated modifications</td>
</tr>
<tr>
<td>2008-07</td>
<td>Integration</td>
<td>Multiple local teams, travelling, local developments contributing to global software</td>
</tr>
</tbody>
</table>

Titlestad et al., 2009
Forms of participatory design (Roland et al., 2018)

Rolan et al., 2018 have identified four types of participatory design based on scale (number of heterogeneous users and settings)

- Singluar PD
- Serial PD
- Parallel PD
- Community PD
Singular PD

Singular - classic participatory design

Design technology in cooperation with small group of end-users.

Mutual learning

End-users can take part in fundamental decisions
Design of artifact used in multiple settings / organizations / groups of users

In cooperation with end-users at one site, then another, and so forth.

Titlestad et al., 2009
Parallel PD

Users are engaged at several sites in parallel to inform generic design.

Core developers make visits to sites in parallel.
Broader community negotiates generic features. Local customization without involvement from core developers.

Circulating use-cases and best practices. Workshops and online arenas for communication.
Meta-design and platforms
Architectures

- The technology must allow for local adaptations
- Flexibility for customization
- Modularization to innovate and make local adaptations.
- Open source software
Meta-design: Designing for future design (Andries Van Onck, Gerard Fischer e.g., 2008).

- Design continue during use.

Software developers create “spaces” so that the software can be shaped according to local use during or after implementation.

- Making functionality and interfaces customizable
- Enable development of plugins
- Open source software.

Mainly aimed at end-users as designers.
Meta-design

High design flexibility

Open source license

Modular apps

Built-in customization tools

Low design flexibility
Opening up the software architecture for the development of third-party apps could be one way of providing local implementers with flexibility.
Platform architectures to support PD

Roland et al., 2018
Architectures

- Technology is not enough.
- Also a “social architecture” is needed.

Need of

- Local competence
- Channels of communication
Enabling large-scale distributed design

Scaffolding - structure that supports design and implementation (Titlestad et al., 2009)

“for the duration of a particular human practice, actors draw on various artefacts, spaces, and infrastructures to conduct their activities” - Orlikowski 2006 p462
Enabling large-scale distributed design

Scaffolding - structure that supports design and implementation

HISP India
HISP Tanzania
HISP Bangladesh
HISP South-Africa
HISP Nigeria
HISP Oslo / UiO
DHIS2 core dev
jira
git
dhis2 academies
mailing lists
workshops
expert academies
Enabling large-scale distributed design

Boundary spanners

Figure 3-5 Implementers as boundary spanners (Titlestad et al., 2009, p. 18)
Architecture for design

Experiences are fed back for further generic development

Global development
Provides pre-designed but customizable software and local competence for utilization.

Local implementation
Possess competence to customize generic software and involve local users.
Example: Commodity ordering in Uganda

- Implemented DHIS2 to support commodity consumption reporting and ordering
- Hard to customize DHIS2 to this domain using built-in customization tools
- Platform core → too low use and design flexibility for this case
- Decided to build a third-party app (high design-flexibility)
- Enabled us to create a system tailored to the use-case.
- New tensions on the “scaffolding” and the boundary spanners