design for change

Yvonne Dittrich
IT-University in Copenhagen
Design and Use of IT

Roadmap

• what is it I am talking about
• some theoretic base
• a research project
• implications for the design, development and use of software
• how that relates to a research project at ITU.

what do they all have in common?

ERP system
your mobile phone
sorter in an e-mail system
CAD systems
spread sheet
work flow management system
word editors and typesetting systems

what do they all have in common?
Programs life Cycles and Laws of Software Evolution

... at the very least, any program is

a model
of a model
within a theory
of a model
of an abstraction
of some portion of the world or of some universe of discourse.

deploying the symbolic of the 'symbolic machine'

- understanding computer applications as auto-operational forms
- operational form: description of a set of operations that - if carried out - result in an anticipated result.
  Examples:
- auto-operational form: a description of a set of operations that can be carried out independent of a human actor.
- the important 'side effect' is that the description has a causal relation to the behaviour of the program.
The clue:
let the user edit the model

- present parts of the computational model to the user, and allow him/her to edit it.
- you need to present the things to manipulate and the 'language' that is constraining the manipulation/assembly.
- the user might need support/guidance to do that.

This is called end-user tailoring or end-user development

Different levels of end-user tailoring

- customisation
- integration
- extension

Design in Use

- participants: a telecommunication provider, a small software developer and the university
- case: a business application to administrate contracts and compute payments based on the contracts and triggered by certain events
- context: rapidly changing business practices require a flexible adaptable system
- tools: a flexible meta-modelling database system allowing to manipulate the data model during operation
specificities of our project:

- a special purpose system
- adaptation to a changing business practice
- the structure of the common object of work has to be adapted
- dependency on interfaces to other systems
- the system controls money

technical experiments

- using a database implementation allowing for changing the database scheme on the run
- using a meta object protocol approach to implement a program where you could change the model through a user interface
- designing an application that allows end-users to redefine the interaction between programs that are part of an infrastructure.

flexibility light

- a good design changes are easily done
- some meta techniques
  - sparsely populated database tables
  - rule based combination and prioritisation between different payments triggered by the same event
  - parts of the program controlled by meta tables
  - feed back to be able to test the adaptation
- tailoring interface for the system administrator
Use: moving Design to Use

- how to present functionality and its possible manipulation to the user?
- 'Gentle slope of complexity'
- support for tailors: annotations and test environments
- users, super users, gurus and local designers
- deliberating change or 'the organisational function of a rigid system'
- how do our methods change when applied by users?

product: design and architectures

- opening up the model to the user
- 'vertical objects' in a layered design (Kasper Ø.)
- meta-techniques: paradigm/notation specific
- interpretation vs compilation
- designing ‘languages’ to manipulate aspects of an application that are understandable by non-users
- it's not only about requirements: integrating requirements from different contexts.

process: 'a small matter of programming'

- blurring the boundaries between use and design
- from software to infrastructure development: co-developing business, work practices, and technological infrastructure
- software engineering for end-user developers?
- interface of usage and different design and development activities
- flexibilisation of software development
Designing Evolvable Software Products

- software products should be applicable in a variety of organisations and survive the necessary changes
- they are often designed in layers (kernel, application, customisation, customer specific data)
- the interfaces between the layers should allow for independent development
- what is fitted into which layer/part of the system
- how are the customisations treated, if the software is upgraded
- how is the communication and coordination between the different design activities organised