



University of Siegen
Chair of Information
Systems and New Media

THE BEST-RUN BUSINESSES RUN SAP™ 

End User Development: Approaches Towards A Flexible Software Design

Michael Spahn, Christian Dörner, Volker Wulf
Track: Design Theory, Research & Practice in Information Systems

ECIS 2008, 9th – 11th June 2008, Galway, Ireland



Chair of Information Systems and New Media
Prof. Dr. Volker Wulf



Questions to Answer

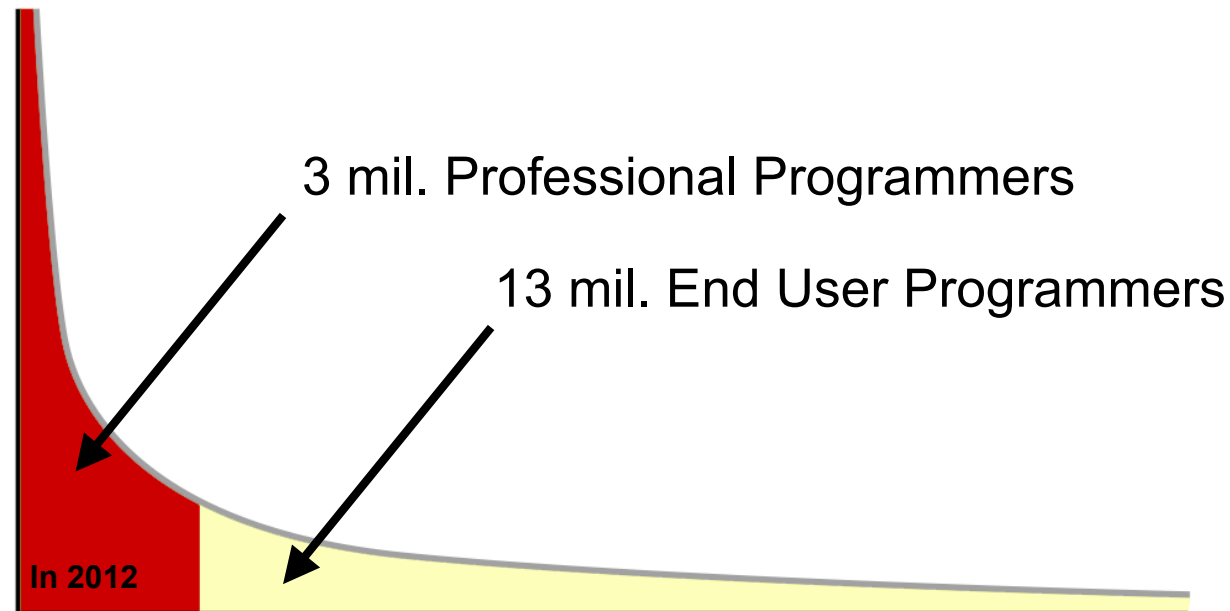
- Why End User Development?
- What is End User Development?
 - ◆ Definition
 - ◆ Classification
 - ◆ Approaches
- How to use it?



Why End User Development?

- Designing flexible software seems to be an unreachable goal
- Why should we make it even more complex, by involving end users?

➔ Design for change leads to cost reduction
The long tail of software development



C. Scaffidi, M. Shaw, and B. Myers, "Estimating the Numbers of End Users and End User Programmers," in *Proceedings of the VL/HCC'05*, 2005, pp. 207 - 214.

What is End User Development?

Lieberman et al.: *“End-User Development can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact.”*

 EUD seems to be everything, except the usage of software

Imagine, I am a Software Engineer, who wants to build a configurable, flexible system by using EUD techniques. What do I need to know?

 Overview of existing EUD approaches
A guideline, how to select appropriate approaches

H. Lieberman, F. Paternò, and V. Wulf, *End User Development*, 1 ed.: Springer, 2005.



What is End User Development?

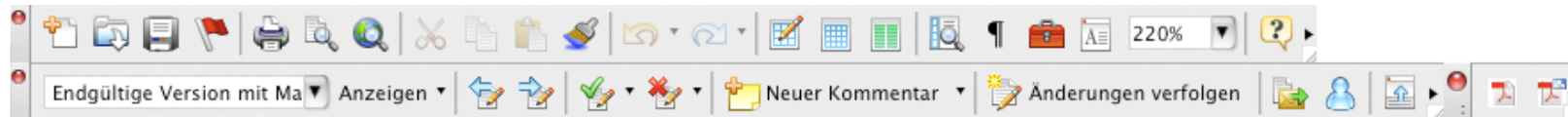
- In contradiction to other overviews, like the one of Germonprez, our classification differentiates the approaches by the two dimensions:
 - ◆ **Complexity:** Technical knowledge, which is required to apply the principle
 - ◆ **Adaptation power:** The level or extend of adaptations that can be realised

EUD Approaches				Supportive EUD Approaches
Complexity / Adaptation Power	Customisation	Integration	Extension	
Programmers			Programming	Testing: Question-based testing; WYSIWYT; Integrity checks; Exploration environments Community aspects: Configuration files Appropriation support
Local Developers		Component swapping at runtime; Separated tailoring interfaces	Natural Programming; Scripting	
Non-Programmers	Interface customisation; Parameterisation	Programming by demonstration (PBD); Accountants paradigm; Integrated tailoring interfaces		

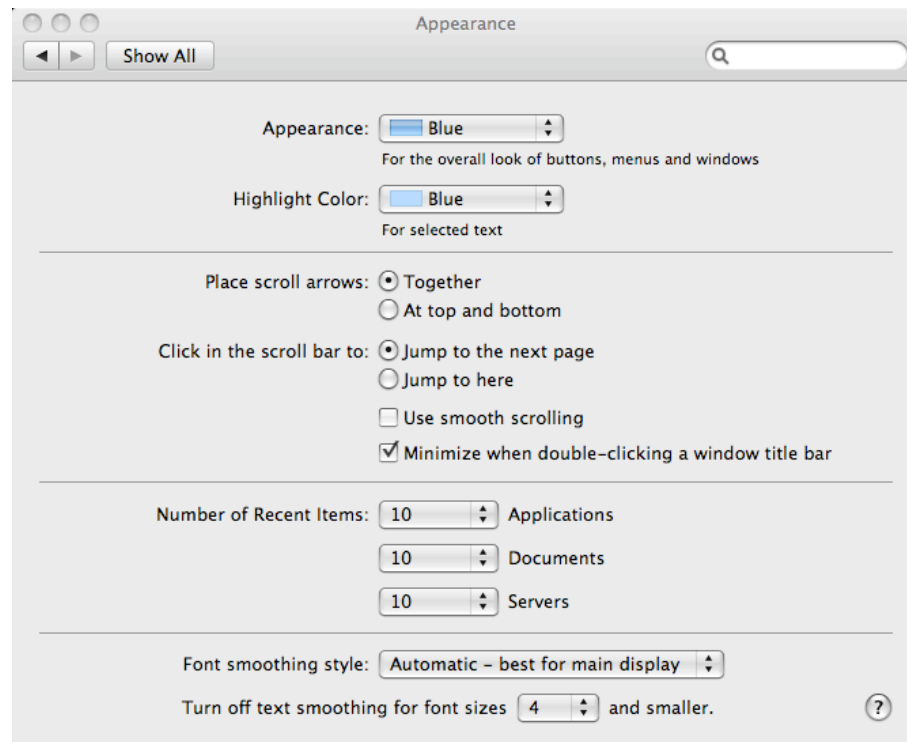
M. Germonprez, D. Hovorka, and F. Collopy, "A Theory of Tailorable Technology Design," *JAIIS*, vol. 8, pp. 351-367, 2007.

EUD Approaches - Customization

Interface Customisation

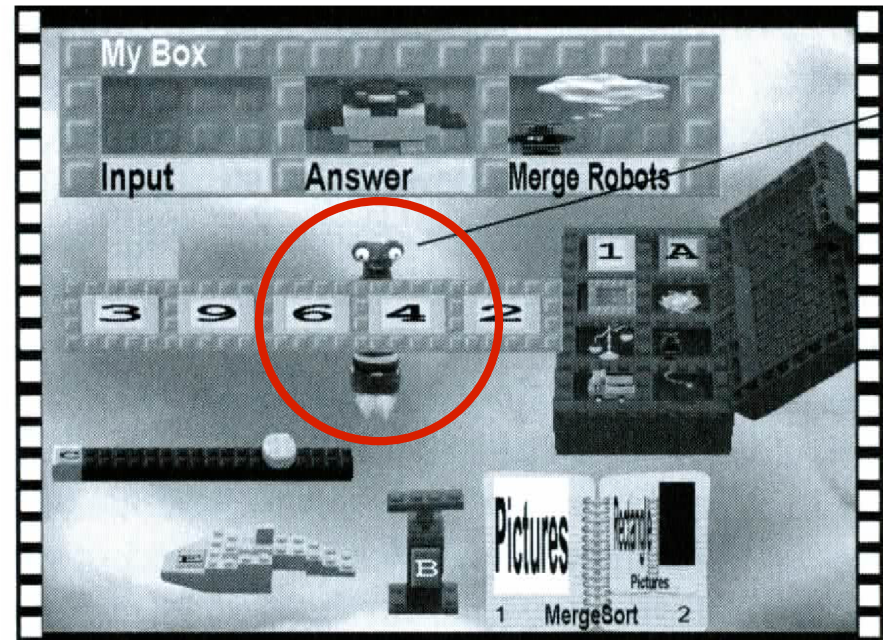
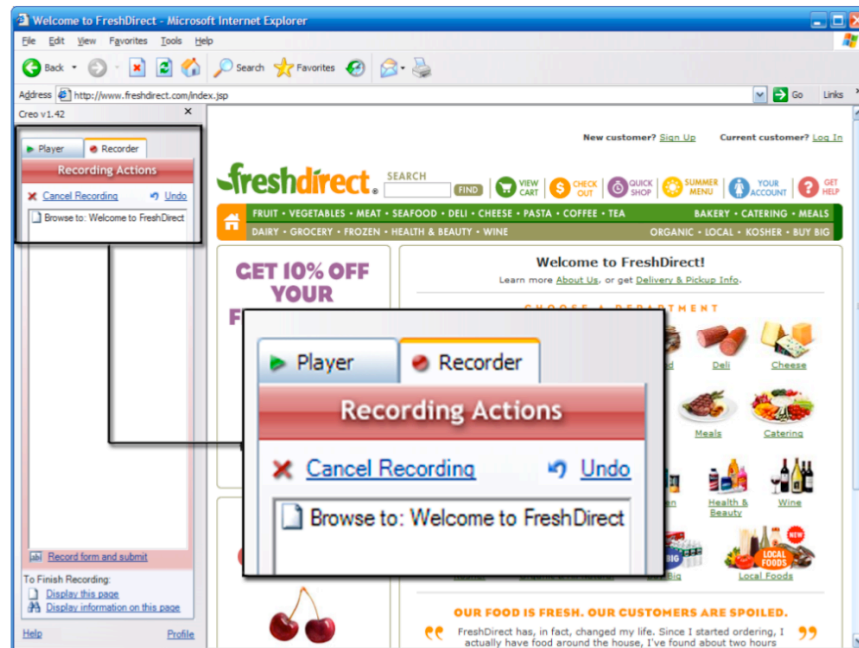


Parameterisation



EUD Approaches - Integration

Programming by Demonstration



Robot about to be trained to drop the box on blank number pad to determine number of holes

A. Faaborg and H. Lieberman, "A Goal-Oriented Web Browser," in *Proceedings of the CHI '06*, 2006, pp. 51 - 760.

H. Lieberman, *Your Wish Is My Command: Programming by Example*: Morgan Kaufmann, 2001.

EUD Approaches - Integration

- Accountants paradigm

- ◆ Well-known depiction of data in a table
- ◆ Programming by
 - Formulas



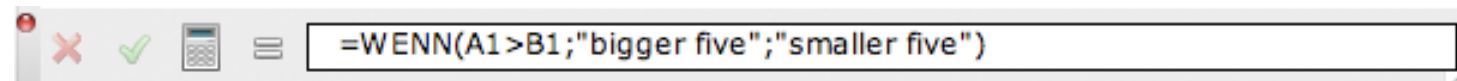
A screenshot of an Excel formula bar. On the left, there are icons for error (red X), success (green checkmark), and a calculator icon. The formula bar contains the text `=EXP(5)`.

- Relations between cells / tables



A screenshot of an Excel formula bar. On the left, there are icons for error (red X), success (green checkmark), and a calculator icon. The formula bar contains the text `=A1+B1+Tabelle2!A1`.

- Logical constructs

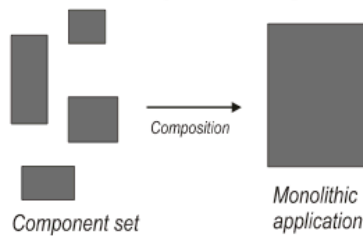


A screenshot of an Excel formula bar. On the left, there are icons for error (red X), success (green checkmark), and a calculator icon. The formula bar contains the text `=WENN(A1>B1;"bigger five";"smaller five")`.

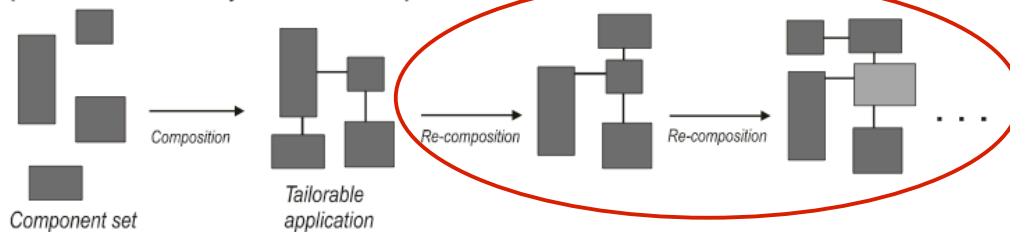
EUD Approaches - Integration

Component swapping at runtime

a) Traditional use of components during development

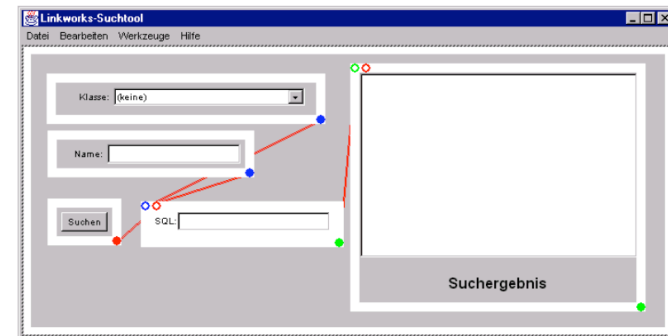
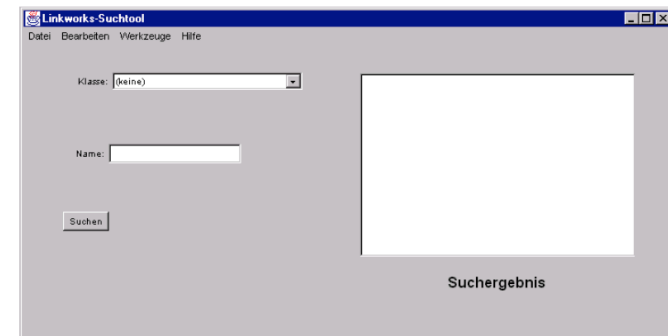


b) Component-based tailorability after initial development



Development phase

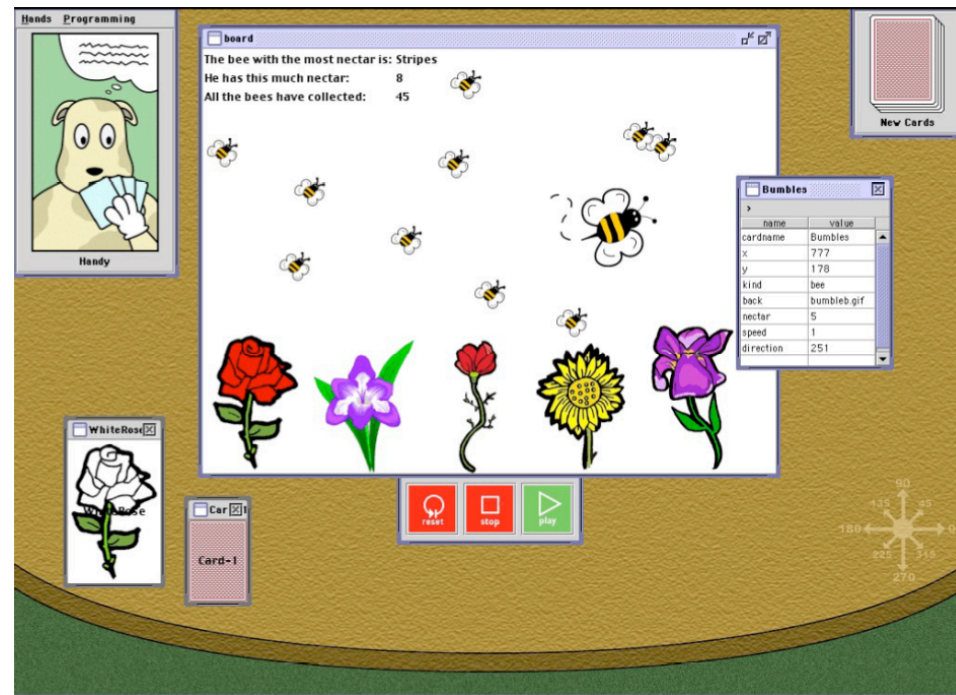
Use and tailoring phase



O. Stiemerling, "Component-Based Tailorability," in *Mathematisch-Naturwissenschaftliche Fakultät*. vol. PhD Bonn, Germany: Universität Bonn, 2000, p. 180.

EUD Approaches - Extension

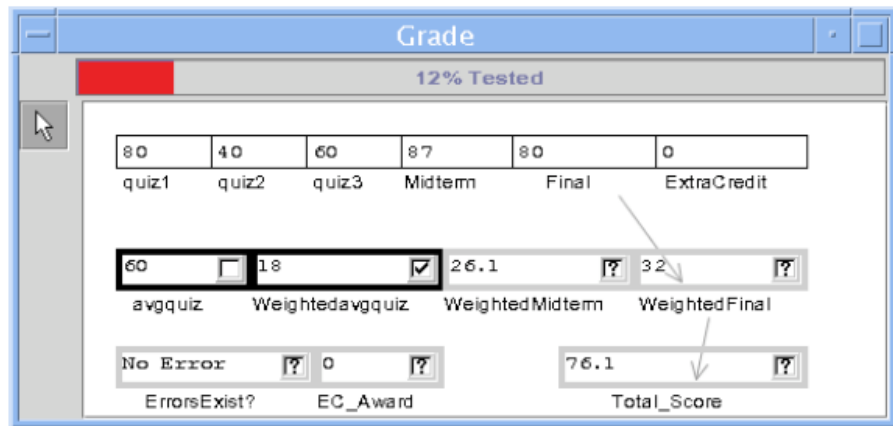
Natural Programming



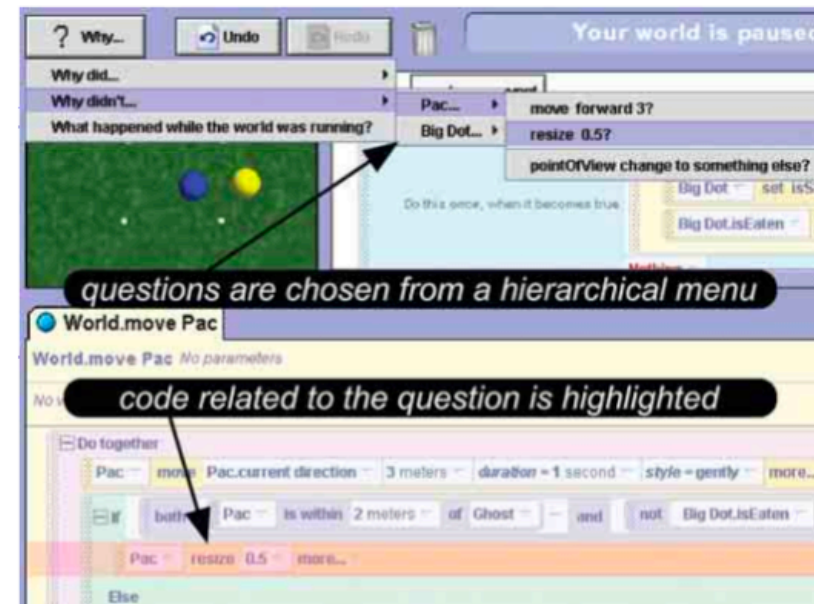
J. F. Pane and B. A. Myers, "More Natural Programming Languages and Environments," in *End User Development*, 1 ed, H. Lieberman, F. Paternò, and V. Wulf, Eds. Berlin: Springer Netherlands, 2005, pp. 39-58.

Supportive EUD Approaches

WYSIWYT



Question-based testing

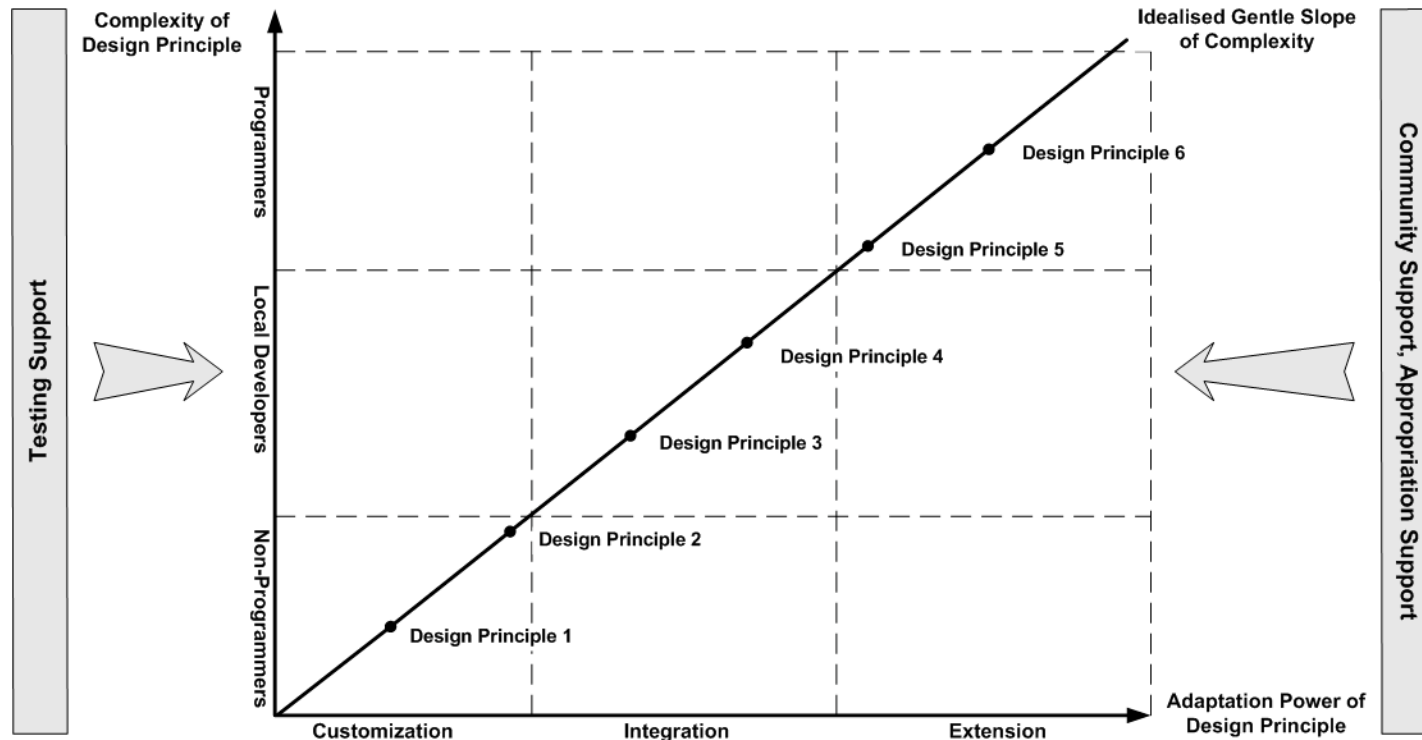


M. Burnett, G. Rothermel, and C. Cook, "An Integrated Software Engineering Approach for End-User Programmers," in *End User Development*, Springer, 2005, pp. 99-126.

B. A. Myers, J. F. Pane, and A. Ko, "Natural programming languages and environments," *Commun. ACM*, vol. 47, pp. 47 - 52, 2004.

How to apply EUD? - A Guideline?

- First, make sure that you have sufficient information about
 - ◆ The users (e.g. who are they, how they are educated)
 - ◆ The domain / field (e.g. which kind of organisation is it)
 - ◆ The context (e.g. work processes, used information systems)



How to use it? - A Guideline?

- Second, fill the created complexity distribution with EUD approaches
- Keep these recommendations in mind
 - ◆ EUD environment of your system should be as natural as possible
 - ◆ The complexity slope should be gentle, if you want to address more than one user group
 - ◆ If users should be able to create and integrate extensions, alternative programming approaches, like programming by example, are required
 - ◆ To get users started quickly, runtime and design-time should be similar
 - ◆ The environment should provide results instantly
 - ◆ The environment should offer support mechanisms
 - Debugging should be natural (users could ask why...?, why didn't...? questions)
 - Debugging tools should help to correct errors
 - Save exploration of adaptations should be possible
 - The user community should be supported (e.g. by exchangeable configuration files)



Conclusion

- EUD can help to increase the flexibility of a software, as it supports the “long tail” of end user developers
- IS developers should think how they can address the different end user developers of their system
- There are many EUD approaches, which may be suitable
- Our guideline can help to select the “right” ones in a structured way

- Future work
 - ◆ The guideline has so far not been evaluated in practice
 - ◆ We work on prototypes with a “gentle complexity distribution”, which combines several EUD approaches to support different user groups



Contact

Christian Dörner
Research Associate

University of Siegen
F: +49 - 271 740 40 70
M: christian.doerner@uni-siegen.de

<http://www.wineme.uni-siegen.de/>

eud
end user development

<http://www.enduserdevelopment.org>

