DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF TORONTO

CSC318S

THE DESIGN OF INTERACTIVE COMPUTATIONAL MEDIA

Lecture 5 — 26 January 1998

INTERACTIVE SYSTEM DESIGN METHODOLOGIES; DESIGN PRINCIPLES AND GUIDELINES

Ronald Baecker
Professor of Computer Science,
Electrical and Computer Engineering, and Management
University of Toronto

5.1 The Rubinstein & Hersh design methodology

We begin by describing one attractive design methodology, although there are also many other interesting ones

Start with iterative design process, which is diagrammed by R. Rubinstein and H. Hersh (*The Human Factor: Designing Computer Systems for People*, Digital Press, 1984) as follows...

Approach based on work in the human factors of computer systems done at Digital Equipment Corporation

Figure 5.1 Iterative development process (R & H, p. 15)

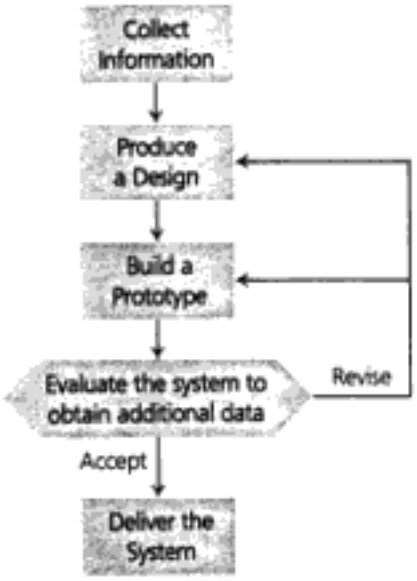

Figure 2-1. Conventional development process

Rubinstein & Hersh then elaborate the process as follows:

Figure 5.2 Elaborated development process (R & H, p. 18)

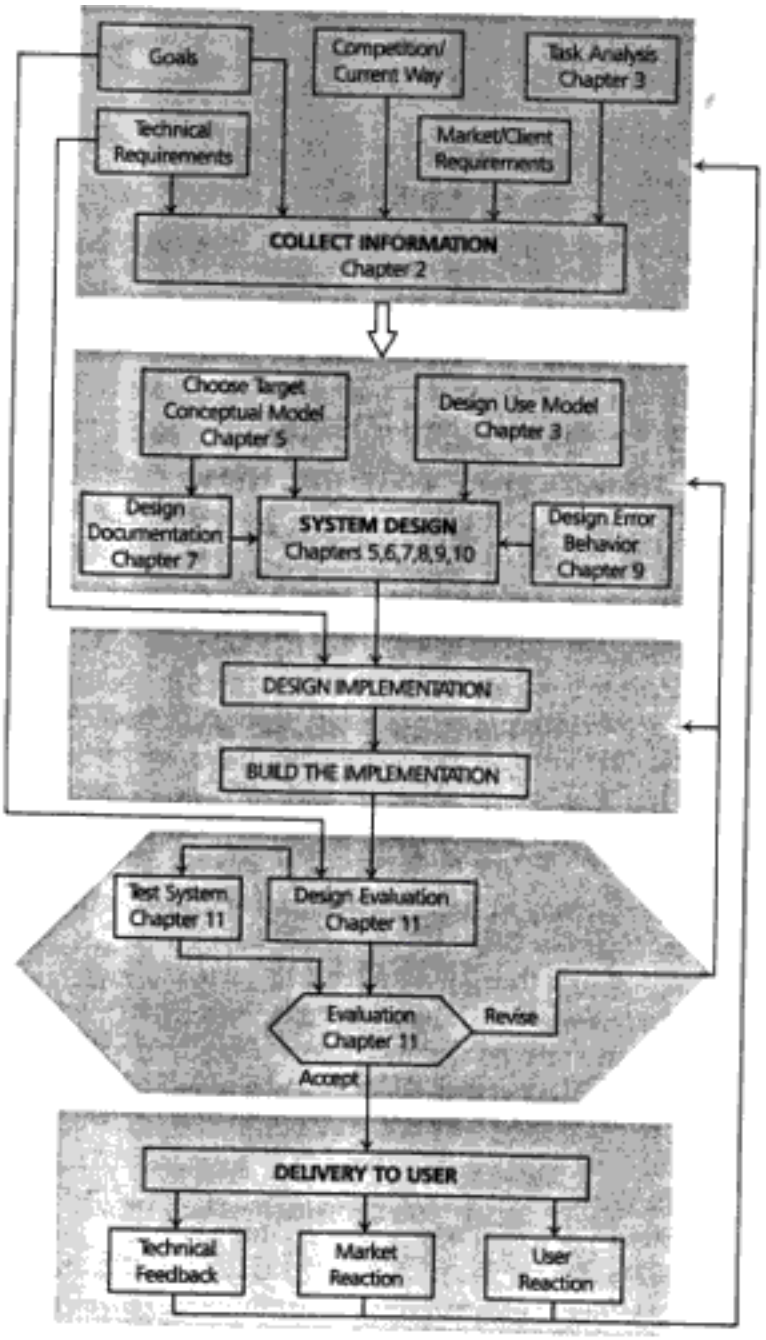

Figure 2-3. Elaborated development process

(User and) task analysis

Who the people are, what they do now, how they do it, and what a new technology or system will do for them

Use model

Description of how people will use a proposed system, what problems it will solve for them, and how it will integrate with their environment (physical, organizational, social)

A task analysis assuming a proposed system, assuming user characteristics do not change, but assuming user's life will change

Can use “day in the life” scenarios to represent and visualize the use model

This helps us make decisions about the system functionality and user tasks to be supported

Conceptual model:

Also known as *users' model* or *mental model*
How the user thinks about the system
Terminology, ideas, operations, relationships, . . .
Metaphors

Documentation

Error behaviour

Market and user reaction

More on these concepts later

5.2 The Gould, Boies, & Lewis design methodology

Based on the user interface and system design principles of John Gould and his colleagues at IBM Yorktown Heights Research Lab (Fig. 5.3) (Gould paper, BGBG)

"Early Focus On Users. Designers should have direct contact with intended or actual users—via interviews, observations, surveys, and participatory design. The aim is to understand users' cognitive, behavioral, attitudinal, anthropometric characteristics—and the characteristics of the jobs they will be doing.

Early — And Continual — User Testing. The only presently feasible approach to successful design is an empirical one, requiring observation and measurement of user behavior, careful evaluation of feedback, insightful solutions to existing problems, and strong motivation to make design changes.

Iterative Design. A system under development must be modified based upon the results of behavioral tests of functions, user interface, help system, documentation, training approach. This process of implementation, testing, feedback, evaluation, and change must be repeated to iteratively improve the system.

Integrated Design. All aspects of usability (e.g., user interface, help system, training plan, documentation) should evolve in parallel, rather than be defined sequentially, and should be under one management."

Figure 5.3. Four user interface and system design principles (Gould, Boies, and Lewis, 1991, p. 75)

Gould then enumerates a number of methods to apply each of these principles (Fig. 5.4)

“Methods to Carry Out Early Focus on Users
Talk with users
Visit customer locations
Observe users working
Videotape users working
Learn about the work organization
Thinking aloud
Try it yourself
Participative design
Expert on design team
Task analysis
Surveys and questionnaires
Testable behavioral target goals

Methods to Carry Out Early — and Continual— User Testing
Printed or video scenarios
Early user manuals
Mock-ups
Simulations
Early prototyping
Early demonstrations
Thinking aloud
Make videotapes
Hallway and storefront methodology
Computer bulletin boards ... and conferencing
Formal prototype test
Try-to-destroy-it contests
Field studies
Follow-up studies

Methods to Carry Out Iterative Design
Software tools
System development work organization

Methods to Carry Out Integrated Design
Consider all aspects of usability in the initial design
One person or group has responsibility for all aspects of usability.”

Fig. 5.4. Methods to apply the design principles of Fig. 5.3 (Gould, 1988, reprinted in BGBG)

5.3 The BGBG design methodology

Realizes iterative, user-centred design philosophy (Lecture 4)
Not intended as a rigid formula, but as an illustration of a philosophy and as examples of how to proceed

Design --> Prototype --> Analyze/evaluate
--> Redesign --> Implement --> Analyze/evaluate
--> Redesign --> Revise implementation -->
Analyze/evaluate-->etc.

	Design-->	Implement-->	Evaluate-->
Information Collection	Questionnaires Interviews with discipline specialists Characterizations of users and tasks	"Day in the life" scenarios	e.g., interviews with users to get reactions to "Day in the life" scenarios
Concept design	Initial design concepts	Design mockups	e.g., interviews with users to get reactions to design mockups
Functionality and interface design	Design of system functionality and look-and-feel	Implementation of "smoke and mirrors" prototype	e.g., usability tests
Prototype implementation	Design of "critical mass" of system	Implementation of partial working system	e.g., heuristic evaluation
Deliverable system implementation	Design and modification of deliverable system	Implementation and installation of this system	e.g., usability tests
System enhancement and evolution	Design of monitoring and feedback system	Implementation of this system	e.g., interviews & questionnaires

5.4 Design principles (guidelines)

Statements which advise a designer on how to proceed

Example (Hansen, 1971)

- Know thy user
- Minimize memorization
- Optimize operations
- Engineer for errors

Rubinstein and Hersh (1984)

- .....
- 2. Separate design from implementation
- .....
- 4. Develop an explicit use model
- .....
- 87. Articulate the evaluation goals
- .....
- 93. Videotape real users

Smith and Mosier: 679 (!!?) guidelines (1984)

Macintosh Human Interface Guidelines (Apple Computer, Inc., Addison-Wesley, 1992) "...describes the way to create products that optimize the interaction between people and Macintosh computers"

Ch. 1: Human Interface Principles
Ch. 2: General Design Considerations
Ch. 3: Human Interface Design and the Development Process
Ch. 4: Menus
Ch. 5: Windows
Ch. 6: Dialog Boxes
Ch. 7: Controls
Ch. 8: Icons
Ch. 9: Colour
Ch. 10: Behaviours
Ch. 11: Language

Tog On Interface (Bruce "Tog" Tognazzini, Addison-Wesley, 1992), in answering specific questions about user interface design for the Macintosh computer, presents roughly 200 guidelines dealing with, for example:

The Design Process
Positively Determining System Behaviour
Positively Influencing User Perceptions and User Behaviour
Promoting Consistency
Making the Interface "Visible"
Reducing or Eliminating Navigation
Dealing with Conceptual Models and the System Image
Dealing with Human-Computer Conversation, Vocabulary
Dealing with Screen Objects, Menus, Icons, Fonts, Error Messages
User Testing
Minimizing Impact of New Releases on Old Users

Pros of guidelines

- Stimulate ideas and insights
- Good checklists giving helpful advice
- Use in *heuristic evaluation*

Cons of guidelines

- Occasionally incorrect
- Usually vague
- Sometimes contradictory (need for tradeoffs)
- Very often not at the appropriate level of specificity
- Often difficult to apply to real design problems

5.5 Heckel's design elements (guidelines)

See P. Heckel, *The Elements of Friendly Software Design*, The New Edition, Sybex, 1991

1. Know your subject.

   Lever your background and your knowledge.

2. Know your audience.

    Who is the typical user?
    Who are all the users, and what are they like?

3. Maintain the user's interest.

    System must be articulate.
    System must allow articulate expression.

4. Communicate visually.

    Not just text, ... but tables, charts, graphs, sketches, diagrams, animation, video, sound!, ...

5. Lever the user's knowledge.

    Increasingly, we know who are users are, and what they know, and that they know... (“user modelling” and “adaptive systems”)

6. Speak the user's language.

    The user's jargon, not computer jargon.

7. Communicate with metaphors.

    Metaphors have strength.
    Metaphors have limits, but that's OK.

8. Focus the user's attention.

    The tracking symbol.
    Getting the user's attention.

9. Anticipate problems in the user's perception.

    Different levels of user knowledge, interest, awareness.
    “Errors” must be anticipated and if possible prevented.

10. If you can't communicate it, don't do it.

    Keep it simple... if you add a feature, throw one out.
    But, unfortunate commercial realities.

11. Reduce the user's defensiveness.

    Communicate descriptively, not judgmentally.
    Communicate with empathy, not with scorn.
    Allow the user to undo and to backtrack.

12. Give the user control.

    Let the user drive.
    But provide support.

13. Support the problem-solving process.

    Provide power tools.
    Speak the user's language.
    Support user learning and user error recovery.

14. Avoid frustrating the user.

    Speed *and* predictability of response.
    Give the user control and provide assistance.

15. Help the user cope.

    Training, documentation, error messages, help.

16. Respond to the user's actions.

    Rapid feedback.
    Comprehensible feedback.
    Minimize hidden system state.

17. Don't let the user focus on mechanics.

    The task, not the tool, should be visible.

18. Help the user to crystallize his thoughts.

    The role of menus.
    The role of examples.

19. Involve the user.

    Emotional involvement.
    Intellectual involvement.

20. Communicate in specifics, not generalities.

    User specification by example instead of by programming.

21. Orient the user in the world.

    Where am I?
    Where have I been?
    Where can I go?

22. Structure the user's interface.

    A frame of reference, a mental map.
    Consistency wherever possible. No surprises.

23. Make your product reliable.

    If it doesn't work, the interface won't save it.

24. Serve both the novice and the experienced user.

    And the transition from novice to expert.
    Layers of functionality.

25. Develop and maintain user rapport.

    Talking the user's language, helping the user cope, consistency, etc.

26. Consider the first impression.

    First impressions last.
    System installation guide, tutorial guide, guided tours.

27. Build a model in the user's mind.

    Begin with the metaphor.

28. Make your design simple...

    Elegance aids both the user and the implementor.

29. But not too simple.

    ...don't lie to communicate the illusion of simplicity.

30. You need vision!!!