Informatics 1B: Data and Analysis Unstructured Data

Contents

Unstructured Data

In cases ***where data is not normally structured we have to impose structure on it***.

For example, when designing a computer system, we need to collect data to specify the requirements of the system.
What data should we collect to inform this?
How do we collect it?

Also, when evaluating a system to see if it works:
What data do we collect and how?

Collecting, Analysing and Interpreting Data

1. What questions are we asking that we need data to answer?
2. What data would provide the answers to these questions?
3. What methods would enable us to collect this data?
4. How would we analyse the data?
5. How would we interpret it?

Collecting Data

In designing a system, or the interface to a system, ***the approach that we take guides the data that we collect***.

The approach we will take to design will be **task-oriented** and **user centered**.

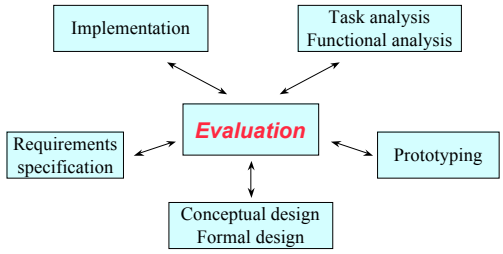
So we need to determine in detail ***what the task is*** that the system is intended to enable, and ***who the potential users are***.

System and Interface Requirements

Stages of system development

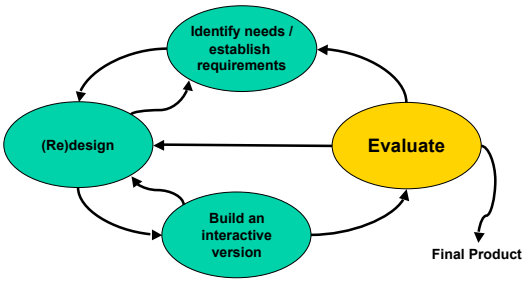
1. Task and requirements analysis
2. Design
3. Evaluating design
4. Prototyping
5. Re-design and iterate
6. Internal evaluation of content
7. Satisfaction of design requirements
8. Usability
9. Effectiveness
10. Conclusions

The "Star" view of system development



From Hix & Hartson (1992)

Waller (2004) summarises…



Task-Centered User Interface Design

Lewis and Rieman (1994)

a. figure out who's going to use the system to do what
b. choose representative tasks for task-centered design
c. plagiarize (from other systems)
d. rough out a design
e. think about it
f. create a mock-up or prototype
g. test it with users
h. iterate
i. build it
j. track it
k. change it

Methods?

Task analysis
Cognitive Walkthrough
Protocol analysis
Video Recording
Questionnaire
Sensitivity Analysis
Post-hoc analysis
Dialogue mark-up and analysis
Manipulation experiment
Self Report

Observation
Mock-ups
Wizard of Oz
Interview
Focus groups
Expert evaluation
Logging use

Sentient analysis

Requirements

Functional requirements: what are the task and sub-tasks?

Data requirements: what data need to be represented within the system?

Environmental requirements: physical, technical, social, organisational?

User requirements: who are they and what skills/constraints do they have?

Usability requirements: accessibility, efficiency of use?

Some (usability) heuristics
(Waller, 2004)

- Visibility of system status
- Match between system and real word
- User control and freedom
- Consistency and standards
- Help users recognise, diagnose and recover from errors
- Error prevention
- Recognition rather than recall
- Flexibility and ease of use
- Aesthetic and minimalist design
- Help and documentation

Joke Generation Example

Automated humour

JAPE (Binsted & Ritchie 1994, 1997) is capable of producing punning riddles such as:

What's the difference between leaves and a car?

What do you get when you cross a monkey and a peach?

What do you call a murderer with fibre?

It searches a general purpose dictionary to find words that fit pre-defined structures called *schemas* and *templates.*

Automated humour

JAPE (Binsted & Ritchie 1994, 1997) is capable of producing punning riddles such as:

What's the difference between leaves and a car?

One you brush and rake, the other you rush and brake.

What do you get when you cross a monkey and a peach?

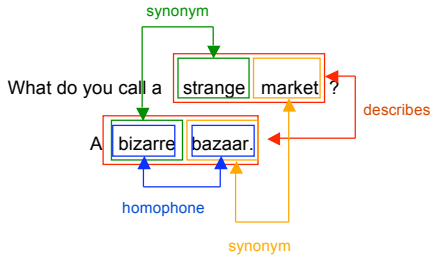
An ape-ricot.

What do you call a murderer with fibre?

A cereal killer.

It searches a general purpose dictionary to find words that fit pre-defined structures called *schemas* and *templates.*

JAPE: example of structure

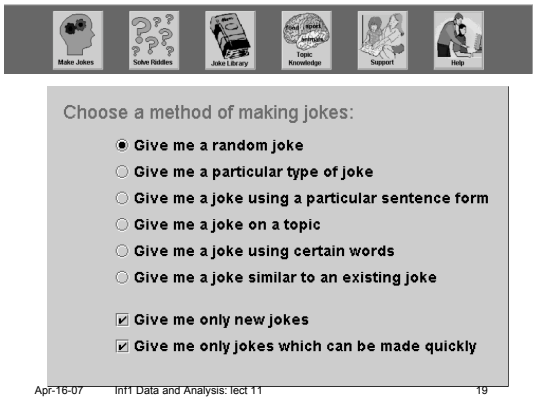


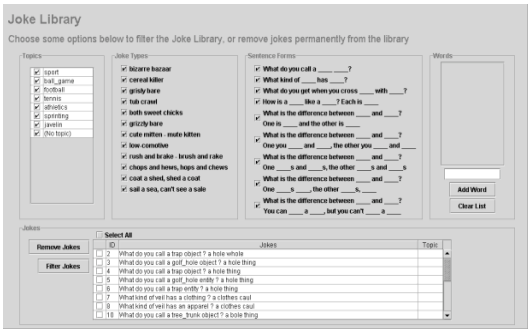
Extensions to Jape

Low (2003) developed a graphical user interface (GUI) for JAPE, with added functionality:

- Creating jokes
- Riddle-solving
- Joke library
- Topic database
- Lexical support

Still slow, interface fairly complex





Further extension

Trujillo-Dennis (2003) developed UI for
- children with speech impairments
- (possibly) motor disabilities
- no cognitive disabilities

Explores several aspects of the user interface:
- Simple language and visual layout
- Adaptable colour schemes
- Speech output
- Single switch scanning interface

Not tested on actual target users

Accessibility (scanning interface)

Following Jape: *Standup*

- To build a tool that helps Language Impaired Children (LIC) to create jokes and play with language
- Interactive: speed, efficiency
- Customizable: extensible
- User-centred design for LIC-specific interface
- Appropriateness (e.g. not unknown vocab):

  What do you get when you cross a vitellus and a saddlery?

  A yolk yoke.

Functional Requirements

Be able to generate jokes:
1. Based on a **topic**
   - Sport
   - Football
2. From **keyword**(s)
   - Using boots and smelly
3. From **templates**
   - Cereal killer: What kind of ___ has ___?

Data Requirements

Templates, schema and lexicon to generate joke

Lexical information on word frequency *(to avoid uncommon words)*

Lexicon related to topic, or method of classifying as such

Environmental Requirements

Will have to include a **scanning device** for interface access for some users

Use at **home or school**

Speech access (generation, not recognition)

Assume parent, teacher or **helper to set up**

User Requirements

Group 1: LIC *(harder to access)*
- Impaired language use
- Not impaired intelligence
- Literacy level below expected for age
- Possible physical impairment (e.g. cerebral palsy)

Group 2: non-LIC
- No language impairment
- Expected literacy level

Experts:
Teachers, parents, speech therapists, carers
Plus **LI Adults** as expert users

Usability Requirements

Not too many key presses

Easy to go back if make unintended selection

Different levels of access to manage language skills and possible progressions:
- ***Vocabulary*** (*measured by word frequency)*
- ***Task difficulty*** *(keyboard input harder than simple selection)*
- ***Joke type*** *(partial word matching harder then homophone substitution)*

Accessible to all users

Initial Methods

Interview + Mock-ups with ***speech therapists:***
- suggests too much reliance on text
- need picture language interface
- suggests various ways to use

Task analysis

Mock-ups

More detail in next lecture…

Task Analysis Exercise

Mobile Phone: task analysis

For each of these tasks:

1. **how many decisions did you have to make**
2. **how many key presses**

- Phone a friend (don't actually do it)
- Who sent you the most recent text
- Add a new person to your phone book called Johnny

One person do the task, and count key presses, the other to observe and count decision points

References

Dix, A., Finlay, J., Abowd, R. and Beale, R. (2004) *Human-Computer Interaction.* Prentice Hall

Lewis, C. and Rieman, J. (1994) *Task-Centered User Interface Design.* Shareware web publication, available at: http://hcibib.org/tcuid/

Preece, J., Rogers, Y., Sharp, H., Benyon, D. Holland, S. and Carey, T. (1994). *Human-Computer Interaction.* Addison-Wesley

STANDUP related references: see http://groups.inf.ed.ac.uk/standup/

Binsted, K. and Ritchie, G. (1994) An Implemented Model of Punning Riddles. Pp. 633-638 in *Proceedings of the Twelfth National Conference on Artificial Intelligence*/Sixth Conference on Innovative Applications of Artificial Intelligence (AAAI-94).

Binsted, K. and Ritchie, G. (1997). Computational rules for punning riddles. *HUMOR*,10 (1), pp.25-76

Low, A. (2003). *Software Support for Joke Creation.* 4th year project report, School of Informatics, University of Edinburgh, Edinburgh, UK.

Trujillo-Dennis, L. (2003). *An Accessible Interface for a Joke Creation Tool*, 4th year project report, School of Informatics, University of Edinburgh, Edinburgh, UK.