Dr. Craig Macdonald

# Week 1

## Lecture 1: basics

Data is facts and figures, information is data with context, and knowledge is conclusions based on information

Issues in managing data:

* Data storage
* Processing for concurrent users
* Accessibility (online)
* Access to the data
* Reliability and security

Features of managing data:

* Definition
* Entry (add new data)
* Editing (change existing data)
* Querying (extracting data)
* Persistence (reliable storage)

Database (DB) – an entity in which related data can be stored in a **persistent and** **structured manner**, with as **little redundancy** as possible

Features of DBs:

* Gives access to data, which can be entered, viewed or updated
* Viewable by many simultaneous users – **controlled concurrent access**

Relational DB – “a series of tables about related information”

DB parts:

* Header
* Name
* Rows
* Data

Redundancy:

* Ambiguity – same thing with different name in another file
* Inconsistency – if data changes in one place, it should also change in all others
* Wasted effort – data should be shared where possible to save time and effort

Controlled concurrent access happens through **transactions**, which makes sure the DB interactions are *independent* and *sequential*

Database Management System (DBMS) attributes:

* Sharing and integration of data
* Multiple views of data
* Controlled concurrent access
* Management of integrity and security

## Lecture 2: designing a DB

Forms of data access:

* Local DB (SQLite, Microsoft Access)
* Client/server (MySQL, Oracle)
* Web-based (PHP as a webpage)

Main components of DB:

* Users
* Applications
* DBMS
* Data (and DB)
* Host system

### Designing DB Systems

3-level (“schema”) architecture:

* External – user level
* Conceptual – how programmers model and implement DB
* Internal – DBMS storing data

DB design lifecycle:

1. **Requirements analysis** (user needs)
2. **Conceptual design** (high level description, Entity Relationship (ER) model)
3. **Logical design** (translate ER model into relational schema)
4. Schema refinement
5. Physical design/tuning

1. Requirements – what data do we care about,

Data model – a description of objects that could be represented by a computer sys and their properties and relationships

Schema – descr of how a DB can be designed

DB – an instance of a schema w/ corresponding data

People involved:

* Users
* DB designers
* (web) application devs
* DB admins
* Website (UX) designers

Requirements:

* Talk to client (CEO, chief)
* Talk to customers (end users, data viewers)
* Talk to diff levels of users (admins, programmers, technical staff)

Requirements HOW:

* Write down objects needed to store data about (uniquely identify the objects (accounts, customers))
* Take note of relationships between them

Data Modelling:

* Entities, attributes and relationships
* Data Model – an abstract representation of the data we wish to store

# Week 2

## Lecture 3: conceptual design for a DB

Entity Relationship Model: a conceptual data model

ER model:

* Entities
* Attributes
* Relationships

Entity – uniquely identifiable and independent object in the real world about which we wish to store data

Entities are grouped together into categories called **entity types** or **entity sets**

Entities are instances of given entity-types

Entity types as **rectangles** and can be:

* Physical objects
* Events
* Concepts

Attributes as **ovals** are properties that describe an entity (type) (connected to entity type with a **straight line**)

All entities of the same type will have the same attributes

Attributes:

* Simple
  + Indivisible value
  + E.g., age, gender
* Composite
  + Composed of a set of component values
  + E.g., address, date of birth
* Single-valued vs multi-valued (**double-lined oval**)
* Key – unique for all entity instances (**underlined**)
  + Sometimes several attributes together form a key

Subtype – entity type that inherits the properties of its parent type

(supertype)

* Disjoint – entities must belong to exactly one subtype
* Inclusive – entities may belong to either or both

d

Subtypes are used when some attributes differ between entity types

(subtype)

(subtype)

### Relationships

Represented as **diamonds**

Can have attributes

Types:

* Binary – 2 participating entity types
* N-ary – more than 2 entity types
  + E.g., seller, buyer and agent in a negotiating relationship

### Cardinality

Cardinality – specifies the number of entity instances that can participate from each side of the relationship

* One-to-one (1-1)
  + E.g., one project and one manager
* One-to-many (1-N)
  + E.g., one manager and many departments
* Many-to-many (M-N)
  + E.g., one employee can be employed by many departments

Totality (**double line**) – participation constraint (all entities in the entity type must participate in at least 1 relationship in the relationship set)

* E.g., every department must have a manager

Structural constraints:

* Cardinality
* Participation constraints

There may be different relationships between two entity types

Entity type may be in a relationship with itself

* Employees supervising other employees
* Name the roles of each side of the relationship (on the rel line)

Relationships can have attributes

### Weak entity types

* Do not have primary key (attributes) of their own
* Depend on other entities to guarantee uniqueness
* Do not have sufficient attributes to form a primary key of their own
* Must have a **total participation** in this identifying relationship
* Partial key (dotted underline)

### In short:

1. Identify entity types
2. Identify properties (attributes and keys)
3. Select primary keys
4. Determine relationships
5. Decide cardinality

## Tutorial 1

Abdul

Entity types:

* Courses
  + ID: 4 character string
  + Title
* Students
  + Student ID: 7 digit integer
  + Forename
  + Surname
  + Email address
* Lecturers
  + Start week
  + End week
  + Members of staff
* Assessments
* Staff
  + Staff number: 7 digit unsigned integer
  + Forename
  + Surname
  + Job-title (distinct: prof, sen lecturer, lecturer, uni teacher, associate lecturer, research assistant, research student, other)

## Lecture 4: recap

## Lecture 5: logical design

s/w: software

A table (**relation**) is constructed for each item of interest in a DB

* All relations have a Heading and a Body
* Rows (tuples) and columns (attributes)
* Relation variable (table name)

Body:

* Unordered
* **Number of tuples (rows): cardinality**

The heading:

* **Number of attributes (columns): degree**

A relation schema is a set of attributes

* Written R (A1, A2, A3, …)
  + STUDENT (name: Text, matric: Number, ex1: Number)

Each attribute in a relation schema has a **domain**: data type, permissible values of type

* Add, subtract, concatenate
* Domain: set of atomic values that can be assigned to an attribute
  + Meaning, e.g., matriculation number set
  + Format, e.g., integer in range 0…999
* (My)SQL:
  + Int, float/double, tinyint (instead of bool), char (fixed length of string), varchar (string length up to limit)

# Week 3

## Lecture 5 (cont.):

Keys:

* Primary
* Foreign: references to the primary key of another table, it’s a **referential constraint** between two tables (a value in a foreign key MUST exists in the referenced primary key)

Two ways of connecting two related pieces of data:

* Same tuple of the same table (same primary key)
* In tuples connected by a foreign key or a chain of foreign keys

Composite attributes: use super attribute as prefix for each individual attribute

Multi-valued attributes:

* Another entity type (weak)

### Relationships

1-to-many:

* The primary key on the “one” side of the relationship is added to the “many” side as a foreign key

1-to-1:

* Foreign key attributes may be added to either schema
* Choose one entity type to be the **subject** and one to be the **target** entity type
  + Total participatory entity type should be subject (main one)

Many-to-many:

* A new relation is produced which contains the primary keys from both sides of the relationships as foreign keys
* These attributes form a composite primary key

### ER to relational schema

1. Entities and their simple attributes
   1. For each *strong* entity type, create a relation with
      1. A column for each simple attribute
         1. Name
         2. Domain
      2. **A primary key**
   2. For each weak entity type:
      1. Create a new relation schema
      2. Add the key attributes of the related entity to the new schema as foreign key attributes
      3. Declare the primary key of the schema
      4. Add the simple attributes

### MySQL Workbench

Refer to slides.

## Tutorial 2: MySQL

Relations:

* Staff(StaffNo: Int, Name\_Forename: VARCHAR(50), Name\_Surname: VARCHAR(50), JobTitle: VARCHAR(50), ManagerOf: VARCHAR(50))
* Tutor(StaffID: Int, PartTime: Tinyint)
* Lecturer(StaffID: Int, StartWeek: VARCHAR(20), EndWeek: VARCHAR(20))
* Course(CourseID: CHAR(4), Title: VARCHAR(50))
* Tutorial Group(CourseID: CHAR(4), GroupNo.: Int, TutorialRoom: Int, LabRoom: Int)
* Student(MatricNo: Int, Name\_Forename: VARCHAR(50), Name\_Surname: VARCHAR(50), Email: VARCHAR(50))
* Takes(CourseID: CHAR(4), MatricNo: Int, ExamMark: Int, PracticalMark: Int)

## Lecture 6: Enforcing Integrity

ER to schema:

1. Strong entities
2. Weak entities
3. Relationships
   1. M:N : new table

Relation: a set of tuples (rows)

There are no multiples of sets

Every relation needs a primary key

Tuples are unordered

Attributes are also unordered

All values are atomic: values cannot be broken down into multiple values (no multi-valued attributes)

Relations represent:

* An entity type and its attributes
* A relationship
* A set of values

Referential integrity when updating:

* Restrict: ban any alterations to a primary key if there are foreign key references
* Cascade: cascade the effect to all relations in all tables that refer to it (usually used for updating PKs)
* Set to NULL: allow update in OG table, set all corresponding FK values to NULL (usually used for deleting primary keys)

# Week 4

## Lecture 7: Sets and Set Theory

Set Theory: branch of mathematics that studies sets

Sets: collections of objects

Usually all members of a set have similar properties

Objects: **elements**

A set **contains** elements.

Two sets are equal if they have the same elements:

* No order
* Repetition irrelevant

Set builder notation:

* O = {X | X is an odd integer less than 10}
* | - “such that”

Predicate: indicates set membership

* {x | F(x)}
* F(x): predicate (returns true or false)
* x: subject of the proposition

Empty set:

* {}

Sets can be represented as Venn diagrams

The power set P(S): all the possible subsets

* Every possible combination, including the empty set and the set itself
* If a set has n elements, the power set has elements
* It contains sets (which themselves have a basic data type)

Notation:

* Union: (all elements in either A, B, or both)
* Intersection: (all elements that are in both A and B)
* Difference
* Complement: e.g., not A
* Symmetric difference: (everything except their intersection)
* Cardinality (number of elements) of a set: |A|

Querying in 2 ways:

* Procedural: relational algebra, based on set theory
  + Sequence of operations
  + Output of each operation is the input to the next operation
* Declarative: SQL, internally implemented as RA operations
  + Describes the desired results (in terms of conditions)
  + The DBMS works out the operations

A relation is an ordered set of tuples (or n-tuples)

**Cartesian product**: A X B: the set of all ordered pairs <a,b> where aA and bB

A relation is a subset of a Cartesian product

## Lecture 8: Relations and Relational Algebra

Relation: subset of the cartesian product of the domains of the attributes

Foreign keys further restrict an attributes domain

### Relational Algebra

A query: performed when a subset of info is extracted from a database to answer a question.

Procedural querying:

* Uses relational algebra (RA)
* Since it’s a series of operations to instances of relations, the result is a relation
* Principal relation operations:
  + Select (rows)
    - Condition
  + Project (columns)
    - Attributes
  + Join (connect two relations (usually by FKs))
    - ⋈
    - Cartesian product:
      * Degree: attributes\_1 + attributes\_2
      * Cardinality: rows\_1 \* rows\_2
    - Selection with condition about an attribute from each relation (Cartesian product)
    - Equi-join: making a new relation from two based on the pairing of rows between those two relations
    - Natural join: if there are identically named attributes, they have the same value
* Set operations:
  + Union, intersection, etc
  + Removes duplicates
  + Union compatible: same number of attributes with the same domains
    - Union, intersection, difference
* Principal relation operations can be combined

# Week 5

## Lecture 9: Structured Query Language (SQL)

RA is used internally by SQL

RA is used directly in NoSQL, e.g., Pandas

* Stretch self-study exercise on Pandas on Moodle (not assessed)

SQL:

* Database language, which allows to
  + Create DBs and relation structures
  + Perform queries
* Non-procedural:
  + Just WHAT data you want, not how
* ISO standard
* Various DBMS products do not wholly conform to the standards

Basic syntax:

SELECT [DISTINCT] target list

FROM relation-list

WHERE qualification

* DISTINCT: no duplicates
* SELECT: project in RA
* FROM: table(s)
* WHERE: condition for selection
* UNION, INTERSECT, MINUS/EXCEPT
  + MySQL only has UNION
  + (SELECT FROM WHERE) UNION (SELECT FROM WHERE)
  + Union compatibility: same number of columns, with matching datatypes
* Cartesian Product
  + SELECT \* FROM Person, Animal;
  + person.houseNum: houseNum from Person table
  + Equi-join: SELECT Person.name FROM Person, Animal WHERE Person.houseNum = Animal.houseNum
  + Natural join
    - SELECT \* FROM Person NATURAL JOIN Animal
* AS (renaming (tables, attributes))
  + SELECT E.name, S.name FROM Employee AS E, Employee AS S, WHERE (E.supervisor = S.NI#)
    - Useful for querying across recursive relationships
    - Self-join
* Order matters: SFW: SELECT FROM WHERE

Semantics:

* Compute the product of relations
* Discard tuples that fail qualification
* Delete attributes not in target list
* If DISTINCT: delete duplicates

LIKE:

* ‘J%’ (names starting with J)
* ‘J\_ \_ \_’ (4 letter names starting with J)

AND, OR

Precedence:

* \* / before + -
* AND before OR

UNION = OR

INTESRSECT = AND

ORDER BY […]

* ORDER BY […] DESC, name
  + DESC: descending
  + Name: other attribute by which it’s ordered if tied

NULL in MySQL Workbench has dark background

## Lecture 10: Further SQL, Design Patterns

Different forms of queries:

SQL Design Patterns

* Unfinished but reusable design for commonly occurring problem types
* Good practice (good, minimal amount of code)

SQL Design Pattern types:

* Basic query
  + Used when there’s 1 table and simple static condition
  + SELECT FROM WHERE
* Equi-join
  + Data is in more than 1 table, rows distributed across more than 1 table
  + SELECT  
    FROM table1, table2  
    WHERE table1.column = table2.column  
    AND [CONDITION]
  + If columns have same name, NATURAL JOIN
* Self-join
  + Data is in one table, but rows are joined across
  + Recursive relationship
  + SELECT  
    FROM table AS t1, table AS t2  
    WHERE t1.column = t2.column
* Grouping
  + Used when looking for a description of a group of data and only 1 value per group is required
  + SELECT grouping-col, aggreg(col)  
    FROM table  
    GROUP BY grouping-col

If you have more than one relation listed in the FROM, you should be expressing a *join condition*

Aggregate Functions

* avg()
* count()
* sum(), max(), min()
  + Return 1 number
  + Do not count NULL
  + Do not count DISTINCT

Grouping

* GROUP BY
  + Produces as many answers as there are identified rows

Syntax:

SELECT  
 FROM  
 WHERE  
 GROUP BY  
 ORDER BY

Other types of Data Manipulation in SQL:

* Insert, update, delete

SQL commands:

* Data definition
* Data manipulation
* Data control

# Human-Computer Interaction

# Week 6

## Lecture 11: Intro to HCI

Lecturer: Dr Julie R. Williamson (Phd CS (UofG), BS Informatics (UofG))

### Policies

Office hours: 12-1 on Wednesdays (will try to change for CS1P)

Email:

### Assessment

No questions will be accepted 2 days before the assessment is due.

* Professional Issues Assessment
  + Debate about the ethical/moral responsibilities of developers doing data mining on Facebook
  + 500 word position statement
    - Due Nov 27th at 4:30
  + Debate participation using YACRS
    - During Nov 28th Lecture
* HCI Assessment
  + Individual report on the implementation of signal processing for user input
    - Worth 8%
    - Due December 13th at 4:30
    - Implementation submitted as a PDF of Jupyter Notebook
    - 4 week cumulative lab assessment (8-11)
      * Week 8: design a new input technique for a flashlight

**How to pass:**

* Do the readings
* Attend the labs

### Why HCI is important:

* HCI sits at the interface of many disciplines
  + Noisy data streams
* HCI can help people
  + Google Crisis Response
    - Making critical info more accessible in times of disaster
* HCI advances Computing in real-world contexts
  + Touch screen
* HCI values inclusiveness and accessibility

Overall:

* HCI principles
* Understanding human perception and control
  + Signal processing
* Quantitative evaluation techniques
* Qualitative evaluation techniques
* Professional issues

HCI:

* Visual layout is a small part of it
* Requires strong technical and creative skills
* Shouldn’t be boring

YACRS at the start of every lecture for reading comprehension (NOT ASSESSED)

## Lecture 12: Usability Heuristics

**KNOW THEM BY HEART**

### The 10:

1. **Visibility of system status**
   1. The system should always keep users informed about what is going on, through **appropriate** feedback within reasonable **time**.
   2. Without feedback, there’s no control
2. **Match between system and the real world**
   1. The system should speak the users' language, with words, phrases and concepts **familiar to the user**, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. **User control and freedom**
   1. Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support **undo** and **redo**.
   2. Ex: edit social media posts
4. **Consistency and standards**
   1. Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow **platform conventions**.
   2. Ex: Android back button acting consistently
5. **Error prevention**
   1. Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
   2. Ex: Email service reading email, noticing “I attached” and no file attachments verify sending
6. **Recognition rather than recall**
   1. Minimize the user's memory load by making objects, actions, and options **visible**. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
   2. Ex: “MyCampus bad”: deep levels of headings, have to remember a lot on your own
7. **Flexibility and efficiency of use**
   1. Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
   2. Ex: Blender: keyboard shortcuts for expert users, complex commands hidden from novice users
8. **Aesthetic and minimalist design**
   1. Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
   2. Ex: Google Search (logo, search in middle, blank space around, some buttons in corners)
9. **Help users recognize, diagnose, and recover from errors**
   1. Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
   2. Ex: exception codes, no explanation
10. **Help and documentation**
    1. Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.
    2. Ex: Tutorial for chat application MeWe

Why this is important:

* Foundational principles of usability
* Apply in many contexts
* Giving tools to evaluate and improve interfaces

### Heuristic Evaluation:

* Evaluators find different problems
  + On average, a single evaluator finds only 35% of usability identified by a group (10-15 evaluators)
  + Each evaluator finds as many problems as possible alone
  + Problems are aggregated

Running a heuristic evaluation:

* One evaluator works alone and completes a report
* Two evaluators work together where 1 speaks aloud during interaction and one takes notes
* Errors identified in terms of heuristics

Number of evaluators:

* Diminishing returns when numerous evaluators are used
* Nielsen recommends 5, but this should be considered with the cost/benefit of the specific context

Successful evaluator: one who finds many errors (unsuccessful for otherwise)

Graph:

* Evaluators: unsuccessful-successful
* Usability problems: hard-easy
* Priority can be understood

# Week 7

## Lecture 13: Sketching

Sketching important:

* A quick way to generate ideas and test ideas quickly
* Another way to express and communicate your ideas
* Often drawing a quick example or diagram is the best way to express a complex concept

Techniques:

* Angle
  + 45 degrees give good control, but limited range of motion
  + 20 degrees give wider range of motion
* Pivoting and anchoring
  + Pivoting from the wrist is familiar, but pivoting from the elbow gives wider range of motion
  + Draw **away** from the body
* Dynamic grip
  + Change grip depending on darkness, strength of lines

Design inspiration:

* User Personas
  + Describing archetypal users and their behaviours into a profile that can inspire design
    - Can’t design for *everyone*
  + Roughly a page or shorter
    - Name
    - Photo/sketch
    - Narrative
      * Key relevant features of life, goals, and behaviours
* Scenarios
  + Identify potential users, their common tasks and the context where interaction will occur
  + Identify users’ goals and the steps needed to achieve the goal
  + Identify steps to be completed by the user and the system
  + Develop a range of scenarios to cover common and uncommon usages
  + Include easy and hard goals
  + Write out a short story with visuals and sketches to add detail

## Lecture 14: User Research

User research: the systematic study of the goals, needs, and capabilities of users so as to specify design, construction, or improvement of tools to benefit how users work and live.

Usability testing data importance:

* Usability testing remains a central way of determining whether users are accomplishing their goals
* Usability testing uses many of the same metric as other user research techniques, e.g., completion rates can be found just about everywhere

Usability: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.

Most usability tests contain:

* Completion rates
* Errors
* Task times
* Task-level satisfaction
* Test-level satisfaction
* Help access
* Lists of usability problems
  + Frequency
  + Severity

Two types of usability tests:

* Formative tests
  + Finding and fixing usability problems
* Summative tests
  + Describing the usability of an application using metrics
  + Benchmark summative tests
    - Describe how usable an application is relative to a set of benchmark goals (provides input on what to fix, essential baseline for comparison)
  + Comparative summative tests
    - Compares app to previous version/competing apps
    - Within-subjects design: Each user completes all conditions
    - Between-subjects design: participants complete only one condition

Sample sizes don’t have to be big, statistics can be practically applied even with 2-5 users.

Representativeness is important, not sample size.

Complete randomness is very difficult to attain because of willingness to participate.

Representativeness > randomness

Data collection:

* Remote sessions
  + Moderated (by moderator)
  + Unmoderated (software collecting clicks, duration of stay, etc)

Completion rates (Success rates):

* Binary measure of task success (1) or failure (0). Calculated by
* Can be not binary, but binary completion rates are easier to quantify

### Usability problems:

* If it’s connected to a user interface, it’s a **UI problem**
  + Name
  + Description
  + Severity rating
    - Frequency (users with problems divided by all users)
    - Impact on user
      * Whether it prevents task completion
      * Whether it causes a significant delay or frustration
      * Whether it has a relatively minor effect on task performance
      * Whether it is just a suggestion
* To prioritise data, it has to be combined
  + Arithmetically: assign percentage results to integers (for different types) and sum everything up
  + Multiplication
* Prioritisation
  + Associate UI problems to users who encountered them
* Knowing the probability with which users will encounter a problem at each phase of development can become a key metric for measuring usability activity impact and return on investment (ROI)

Task Time: how long a user spends on an activity

* Task completion time
* Time until failure
* Total time on task

Errors: unintended action, slip, mistake, or omission a user makes while attempting a task

Satisfaction ratings

* Post-task questionnaires
* Post-test questionnaires
* Standardised

Combined scores

* Easier instead of thinking about many metrics (which not always correlate)

A/B testing (split-half testing): users randomly work with one of two deployed design alternatives

* Popularised by Amazon
* Collecting clicks and page views is often used for determining conversion rates, purchase rates, or feature usage

### Survey Data

Rating scales

* Closed-ended response options
* Likert-type items: agree/disagree to statements
* Can be quantified into distributions with mean, standard deviation, confidence intervals

Net Promoter scores®:

* The question: “How likely is it that you’ll recommend this product to a friend or colleague?” (0-10)
* Promoters: 9-10
* Passives: 7-8
* Detractors: 0-6
* Calculated:

Open-ended data:

* Different forms
  + Reasons why customers are promoters/detractors
  + Customer insights from field studies
  + Complaints
  + Reasons for difficulty of a task
* Can be turned into categories

Requirements gathering:

* There are methods for analysing customer behaviours that reveal unmet needs

KEY POINTS FROM THE CHAPTER

* User research is a broad term that encompasses many methodologies that generate quantifiable outcomes, including usability testing, surveys, questionnaires, and site visits.
* Usability testing is a central activity in user research and typically generates the metrics of completion rates, task times, errors, satisfaction data, and user interface problems.
* Binary completion rates are both a fundamental usability metric and a metric applied to all areas of scientific research.
* You can quantify data from small sample sizes and use statistics to draw conclusions.
* Even open-ended comments and problem descriptions can be categorised and quantified.

### Lecture

User Research importance:

* Data is a crucial part of everyday life
* CS students have great power and responsibility in the age of “Big Data”
* Understand how data is collected and analysed will give you a better understanding of the world around you

Continuous vs Discrete Data

* **Continuous**: varies on a spectrum (e.g., reaction time)
* **Discrete**: segmented type of data, e.g., no. of errors
* Collecting a range of data sources will help form the best picture of usability

Data collection:

* Data can be collected **actively or passively**
  + **Actively**: prompts/questionnaires
  + **Passively**: automatic logging
    - E.g., pedestrian tracking, software tracking

How to analyse quantitative data

* Understand mean and standard deviation
* Perform statistical tests
  + Identify the features of data that would tell one what test to run

Population – everyone

* E.g., average height of CS1Q class

Sample – representative group

* E.g., average height of students in class now

Independent vs dependent variables

* **Independent**: what is manipulated
* **Dependent**: what is measured

Parametric and Non-parametric data:

* Data is not always normally distributed
* Standard deviation can vary

# Week 8

## Lecture 15: Quantitative & Qualitative Evaluation Techniques

State Machines:

* Diagrams which represent different states of product/device
* Analysing complexity of interfaces

Why evaluate:

* Meaningful answers to questions about the system
  + Inferring causal relationships
* Ethical practice

Running an experiment:

* Key stages
  + Provide info to participants about the goals of the experiment and what is involved
    - Actions, risk
  + Gather informed consent from participants
  + Complete a series of tasks to collect data from participants
  + Finish the experiment and provide de-briefing
* Ethics
  + Participants must provide **informed** (not T&S) consent
  + Experimenters must not be in a position of authority over participants
  + Participants understand they can withdraw at any time
  + Participants are given the contact info for experimenters

Evaluation example:

* Scenario: tourism app with navigation
* Questions
  + Do users discover **more points of interest** using the audio navigation?
  + Do users have a **better experience** using audio navigation?
  + Are users less likely to make **navigation errors** using audio navigation?
* Scenario: online banking interface with text notifications
  + No. of text notification setup
  + No. of failed attempts to setup notifications
  + Time spent setting it up
  + User feedback
  + Mouse movement logs
  + Gaze position

## Lecture 16: Qualitative Research

### Data collection

* A research journal
  + A record of all activities during the research process (e.g., appointments, summaries of discussions, proposal writing, problems, dates, committee and review board decisions, reasons for the decisions)
  + Allows researcher to become more self-aware

### Sources of data

* + Examples: interviews, observations, videos, documents, drawings, diaries, group meetings, memoirs, newspapers, historical documents, biographies, etc)
  + Quality matters
  + Interviews
    - Record and Transcribe to complete analysis (an interview will happen at the end of user study usually)
    - Unstructured
      * Provide the richest source of data for theory building
      * Participants able to talk more freely
      * Participants have more control
      * Researchers influence interview with reactions
      * Ability to follow up with additional interviews with questions that arose from the previous one
      * Longer to analyse
    - Semi-structured
      * Some topics are chosen previously
      * Ability to ask follow-up questions
      * Participants might not add important info because they did not think the researchers were interested in that topic
    - Structured
      * Each interviewee is given the same set of questions
      * Consistent
      * Ability to make adjustments during data collection based on analysis of previous interviews is missing
      * Participants have almost no control (large researcher bias in choice of topics)
    - Issues
      * Beginner researchers filling in silences with questions: interrupts participant’s thought process
      * Taping interviews makes participants shy/less talkative/more guarded
        + When the recorder is turned off, participants may be more willing (and able, after the deeper insight into the thoughts of the interviewee) to share more details
        + Bring pen and paper
      * Have to build a rapport with interviewees, a skill to be developed
  + Observations
    - More difficult
    - More time consuming
    - Benefits
      * Reality of participants’ behaviour can only be gleamed with observations because participants might be wrong about their actions
    - Issues
      * Researcher’s interpretation of participants’ behaviour might be wrong
      * Sticking to observational guides is bad because they don’t foster discovery

### Considerations beforehand

* + Committees and review boards
    - First show proposal to research committee members
    - Then show proposal to institutional review boards (IRBs)
    - Proposal can include
      * Title
      * Research question
      * Research design and method to be used
      * Target population
      * Data collection methods
      * Proposed number and types of participants
      * Promise for no financial compensation
      * Rights of participants – how
      * Consent form
      * Potential risk to participants
  + Interview and observational guides
    - Interviewees can withdraw consent at any time in the research process (even after)
    - Acts only as an intro to interview
  + Informed consent
    - Even reluctance should be accepted as withdrawal
    - Participants sign consent form (duplicate) but can withdraw at any time
  + Confidentiality and anonymity
    - IRBs and committees try to ensure these
    - Some participants may want to reveal their names purposely
  + Researcher responsibilities
    - Follow through procedures indicated in the protocol
    - Treat participants as you would like to be treated
    - Remove info that is requested to remove
    - Report potentially harmful behaviour to appropriate authorities

Strategies for controlling intrusion of Perspectives, Biases, and Assumptions:

* Keeping a journal
  + Be aware
  + Record not only participants, but also researcher response to data, responses
* Using the Method
  + Compare data

### Lecture

Why use qualitative data:

* User experience is more complex to capture than usability
* Qualitative data is one of the few ways to understand **reasons** for behaviour
* Rigorous analysis techniques can ensure that results are trustworthy

User experience: all aspects of the end-user’s interaction with the company, its services, and its products

Both user experience and usability have to come together to create immersive experiences.

**Qualitative data**: involves interpretation by a researcher, thus making the researcher as much a part of the results as the participant and the data they provide

**Grounded Theory**:

* Research method where theory is derived from the data collection
  + Theoretical concepts are not used to guide data collection, and initial research questions can be very open-ended
* Analysis is done through a 3-part process
  + Open, Axial, and Selective Coding

Sociological theories (to understand behaviour):

* Ethnomethodology
  + Study of how peoples’ actions and words serve to maintain social order
    - Harold Garfinkle: The art of everyday life through expression and action
* Performance of Self
  + Use theatre and performance studies as a metaphor for how people behave in public spaces
    - Goffman: All actions can be understood as performances of self
  + Every action performed with audience in mind, optimising presentation of self, public or private space

# Week 9

## Lecture 17: Ethnomethodology, Qualitative Data

Leading questions and researcher assumptions:

* Bias can be introduced by researching based on the questions they ask and how they phrase questions
* Neutral questions

Ethnomethodology:

* What are the actions and words people use to maintain order in everyday life?
  + In contrast to traditional sociology, which is concerned with large scale interaction, ethnomethodology is concerned with the fine-grained detail of individual interactions
  + Talk is a large part of that
* Breaching experiments: How do we reveal these everyday actions and common sense decisions? By breaking them
  + Bargain for prices in a store
  + Act as a guest in your family home
  + Invade personal space
  + In HCI:
    - Used when developing new interactions that don’t have existing practice
      * Asking people to play in unusual settings
      * Asking people to approach unfamiliar technologies
      * Asking people to collaborate and share in new contexts
* Conversation analysis
  + Tone is important
* Embodied interaction
  + Inspired by a focus in sociology on “micro-interactions”
    - Go beyond the real world as a *metaphor* for interaction and using it as a *medium* for interaction
  + Where the Action is (Paul Dourish (Glaswegian))

How to analyse Qualitative Data:

* Observe and interpret

1. Open Coding (adding labels)
   1. For each quote, concept, or important utterance, add a code (tag)
      1. Do this as one per post-it note or with a list
2. Axial Coding (getting themes)
   1. Determine relationships between open codes
      1. Typically by grouping the codes into meaningful groups
         1. Post-it notes help there to see all the codes and move the groupings around
3. Selective Coding
   1. From the groups identified, give each a code which represents the key concept for each group
      1. Use these codes to present results
      2. Re-code the transcript with these codes

## Lecture 18: Perceptual Control Theory, Finite State Machines

Controlling behaviour is unique to life

Controlling:

* Action
  + Controlling vs. affecting (obv)
  + Much of controlling requires producing an action that affects the environment
  + Often can’t be planned beforehand
  + No everyday action is exactly the same because of different environmental variables (water temp in pipes, different TV show episodes on Netflix, etc)
  + Mostly consequences of actions repeat, not actions themselves, e.g., bath temp is warm after getting a new ratio of cold to hot water
* Perception
  + Sensory signals
  + Looking at relationships, e.g., front of car to position on road, temperature for bath water
  + “Perception tells us the current status of whatever it is we’re trying to control.”
* Comparison
  + Comparing perception and preferred perception (**reference perception / intention**)
    - Reference perception: not prediction, but target/goal
  + Error – a difference between actual perception and reference perception

### Lecture

Stripping back the UI:

* Model interface functionalities
* No layout, visual styling
  + They’re important, but affect aesthetics, ergonomics, rather
* Separate: peel away representation

FSMs are helpful:

* Precise: no vagueness
* Implementable: can be transformed into code
* Abstract: strips away irrelevant details (appearances can’t be deceiving)
* Analytical: useful properties, like depth or rudeness can be worked out

Structure of FSMs:

* **States**: a continuing state of the world
  + Ellipses
* **Action**: a transition/event that moves between states
  + Arrows with labels
* .
* System moves from state to another when an action occurs
* States persist over period of time
* Actions are instantaneous
* Every next state must be specified for every valid action
* .
* **Self-transitions** are possible
* **Terminal state**: a state which has no leaving
* **Unreachable state**: no action can get to that state
* **Pointless state**
  + A state that cannot change an outcome
  + A state with only one incoming and one outgoing action is likely to be a waiting state or pointless

Timing: a common type of transition that occurs after a certain time has elapsed. The timeout is **reset** every time the state is entered.

Rail-roading:

* A state machine with few backward transitions
  + Users forced down a predefined path
* Relying on a **reset** to jump to the start of the interaction is clumsy

Bridge & hinge:

* Bridge: an event that links two otherwise disconnected parts
* Hinge: a state that is the only connection between two disconnected parts
* They’re bad because they’re **bottlenecks**

Deep hierarchies: confusing

Randomly wandering user:

Rudeness:

* A state sequence you can move through, but leads to a state that cannot be useful
* A good interface “fails early”, and doesn’t allow the user to wander the state machine with no possible “winning option”

Modes: pitfalls

* Allow much more functionality with a limited set of controls
* But it requires user memorisation of modes (the same actions having different effects)
  + For example, Air France Flight 447, Royal Majesty accident

# Week 10

## Lecture 19: Control + States

Views on interaction:

* Commands: sequences of characters that actuate functionality
* Dialogue
  + OK Google, Siri, Alexa
* Direct manipulation
  + Mouse, keyboard
* Pattern recognition
  + Gestures

Interaction structure

* Formally: driving a system into a state compatible with user intentions
  + Intention:

Signals becoming actions:

* Computers sense the physical world with sensors
* People change the state of the world by moving their bodies

Signals: what we measure about the world, some of which is changes related to a user’s intention

* Corrupted by noise and irrelevant observation

Mental space -> Physical space -> Sensor space -> Control space

Role of feedback

* Control requires feedback
* A system without feedback is uncontrollable (and unusable)
* Feedback provides stability under disturbances
* The state that is fed back must offer opportunities for action
  + It must be possible to use the feedback to make useful things happen
* And it must represent an internal state of the system being controlled
  + a common “view on the world”
  + **You don’t control your mouse – you control your cursor**

Control: closed loop

* Human-Computer
* Intention-State

Sensors:

* Mouse
  + Uses cursor to mediate control
  + The cursor is the state which represents belief about what you are going to do
  + Feedback is presented
  + Acceleration curves
    - Each “speed” setting is really a curve
    - Control is easy

Keyboard

* Continuous control all happens in hardware (the physical mechanics)
  + This is why real keyboards are usually easier
* Visual feedback
* Common state

Touchscreen

* An array of capacitive sensors
* Direct mapping: position of finger
* Control is over visual finger position relative to display
* Position recovered from “image”

Compatibility of dynamics:

* For a human to control smth, the internal (controlled) state needs to be compatible with their **dynamics**: the timing of the motion of their limbs and their perceptual capabilities
* Interaction is (almost) universally carried out by limb movement

Smoothing:

* Eliminates fluctuations (noise) which are impossible to make intentionally

Order of control: rates of change

* There have to be mappings of signals which correspond to what people can control
* Sometimes raw measurements do not capture this
* Time derivative: rate of change

Phase space:

* Representing both time derivatives (velocity) and original values (position) as points in space

Thresholding:

* Most UI tasks are discrete actions
  + print a doc, submit a form, delete a file
* Signals are continuous
* **Threshold** continuous values to trigger actions
  + **All interaction comes down to thresholding**

States:

* Discrete actions
  + Debouncing: translating continuous to discrete without unintentional consequences
    - Hysteresis: a system depends on its history
      * Bouncing can be removed by applying hysteresis
      * e.g., waiting time
    - Deadline

## Lecture 20: Ethics of Research

BCS Code of Conduct

* Public Interest
* Professional Competence and Integrity
* Duty to Relevant Authority
* Duty to the Profession

ACM (Association of Computing Machinery) Code of Ethics

* Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing
* Avoid harm
* Be honest and trustworthy
* Be fair and take action not to discriminate
* Respect the work required to produce new ideas, inventions, creative works, and computing artefacts
* Respect privacy
* Honour confidentiality

# Week 11

## Lecture 21: HCI Revision

User Interface Design:

* Usability Heuristics
  + Not just name, but explanation and examples
* Heuristic Evaluation
  + The chart
* Personas
  + Develop it and explain how they might use product
* Scenarios
* Sketching Techniques

Quantitative Evaluation Techniques

* Quantitative data definition
  + Task times, error rates
* Continuous vs discrete data
* Methods for collecting and analysing data
  + Descriptive statistics for analysing
* Populations vs Samples
* Independent and dependent variables
* Within and Between Subjects
  + Within: every participant sees every condition (both sides of A/B testing)
  + Between: participants are divided into groups and see part of the conditions
* Normal distribution
  + Percentage of std
* How to Run an Experiment

Qualitative Evaluation Techniques

* Usability vs User experience
* Qualitative data definition
* Methods for collecting and analysing data
  + Interviews
  + Observations
* Sources of Bias
  + Removal of leading questions
* **Open, Axial, and Selective coding**

Implementing User interfaces

* What is interaction
* Signals and human input
  + Control
  + Thresholding
* Feedback
* Dynamics
  + Detectable and Performable
* Finite State Machines
  + “Sketching”
  + Rudeness, bridges, hinges

Ethics

* Ethical procedures for running evaluations
  + Data privacy, ability to withdraw at any time (even after)
* Informed Consent

Exam advice:

* Pen
  + Including sketching
* Write only on the front of each page
* Do not write in the margin

### Mock Exam

***This question is about quantitative evaluation techniques.***

**You have recently completed an evaluation of a student record management system that allows users to input and manage student records. You have collected a dataset that includes the task times for entering data into the system for twenty participants.**

**You selected your participants from a pool of administrators who are likely to use the system. What is this kind of selection called? [2]**

Sample

**How would you describe the quantitative data you have collected? [2]**

**Your analysis shows that the data is normally distributed. With respect to the mean and standard**

**deviation, what does this say about your data? [4]**

***This question is about implementing user interfaces.***

**Being able to control interfaces is crucial for successful interaction. What is required to enable control, and why is this important? [4]**

**Continuous actions by a user need to be translated into discrete actions. How can this be achieved? [2]**

**Give an example of an interface that translates continuous action into discrete action, including the actions performed by a user, the signal that is generated, and how the signal is analysed. [5]**

***This question is about qualitative evaluation techniques.***

**How would you complete a semi-structured interview as part of a user study? [2]**

**Describe how you would analyse the results of a semi-structured interview using qualitative analysis techniques. [4]**

## Lecture 22: IM Revision

Revision tips:

* Summarise lectures
* Write definitions
* Practice practical questions (rel. algebra, SQL)
* Use other resources
  + recommended books
  + the Internet
    - refs on Moodle
  + Quizzes on Moodle
  + YACRS quizzes
  + The library
  + Past papers
    - CS1F on Moodle
    - CS1Q & CF2 on uni past paper system
    - Style and structure is the same as the past ones
    - Week 7 IM tutorial solutions

Exam advice:

* Answer each section on separate answer books
  + Easier for Craig and Julie
* Read ALL questions before starting to write
* 90 minutes, 50 marks -> pace yourself
* 1 mark for each distinct aspect
* Use diagrams and examples if you think they might help the answer
  + Sometimes they are ASKED FOR, then GIVE BOTH
* Writing
  + Write in pen
    - Black or blue
  + Plenty of space -> don’t write too small
  + Don’t write in the margins
  + You can do rough workings in the left side
  + **Write only in the right-hand side**
  + Leave gaps between answers
* Types of questions
  + Worked examples
    - ER diagrams, queries
  + Critical questions
    - Compare, assess