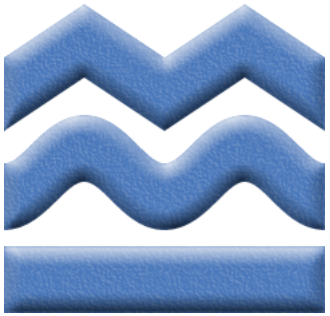
Solving scaling problems with the modern GUI

Pete Bagnall pete@surfaceeffect.com



The Problem…

What is the problem?

Modern software has more functionality than the GUI can effectively present.

A large software vendor carried out a focus group exercise, to find what extra functions users wanted.

70% of the features requested already existed.

Moore's law is exceeded only by programmers ability to consume those resources.

Reaching the Limit

If software is going to become yet more powerful and useful it must find an alternative way of presenting the increased capability.

Some functions are presented in different ways by different vendors making the burden of learning even greater for the user. This needs to be addressed too.

Scaling Limits in GUIs

Why are there limits on the number of functions an application should have?

Limited Display Space

Navigation Difficulties

Humans Don't Scale

Programmers time appears not to be a limiting factor!

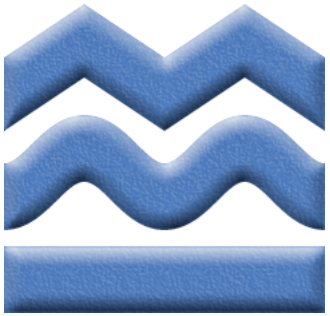
Levels of Complexity

Most computers have about 100 applications installed.

Of these it is rare for a user to frequently use more than about 10. Many users use only 2 or 3.

Most commonly used are either general purpose applications, or in some domains, software written specially for the job.

General purpose applications typically have a few hundred functions.



Modern Software…

Monolithic Applications

A major feature of modern GUI OS's is the application.

The application is a presentation of a more fundamental paradigm, the process.

The concept of a process evolved from the old batch processing systems, where each process was isolated and insulated from errors in the others.

Now information is used more fluidly this seems inappropriate. Applications need to share better.

Types of Applications

- Document Processors
  - General Purpose
  - Specialised
- Information Repositories
- Accessories

Document Processors

Any application that works with documents

- e.g. Word Processors, Spreadsheets, Photo Retouching
- Relies on the OS filesystem to organise and retrieve documents

Specialised

- Timesheet Program, Interface Builders

General Purpose

- MS Word, MS Excel, Adobe Photoshop

Specialised Document Editors

Can be very highly tuned.

Are used in a relatively constrained way.

May be associated with a business workflow, and therefore have a very prescribed path through the software.

General Purpose Editors

Used for a wide variety of tasks.

Has many different modes of use.

Often has extremely rich functionality.

Frequently suffer from “creeping featuritis”.

Information Repositories

Applications that store data in their own data structures, and provide their own navigation systems to that data

e.g. Email, iTunes & iPhoto from Apple

Includes most enterprise systems and corporate database applications, such as OSS systems

Accessories

Applications that do not create or store state for long periods.

e.g. Calculators, Image Viewers, DVD players,
File Browsers

The Feature Myth

With car safety there is now a European standardised test, and a simple scale for expressing how safe a car is.

There is no equivalent, simple system, to express how easy a piece of software is to use.

We are still at the beginning of learning how to design software that is genuinely easy to use.

The Feature Myth

Most vendors are still mostly concerned with capability, not desirability.

A program is capable of doing something if it has a “feature” to do it, even if it is tortuous.

A program is desirable if it does the things that are needed effectively, and easily.

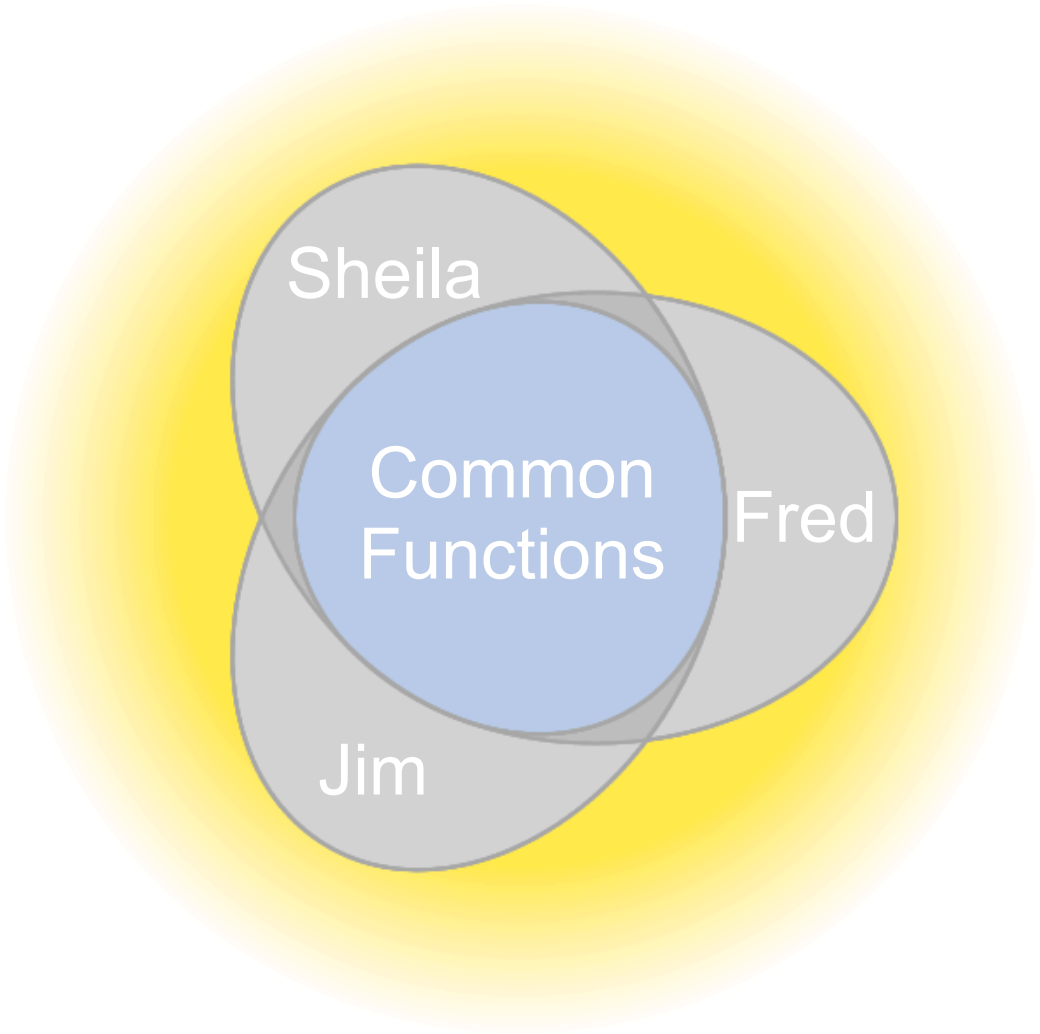
So a feature should only be present if it addresses a need, and it should be “well designed”.

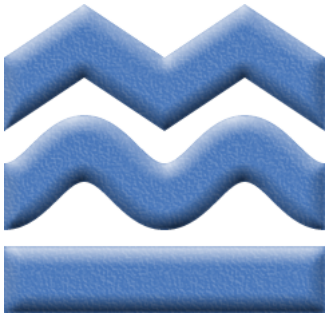
Software needs to be specific

" I don't know the key to success, but the key to failure is to try to please everyone."

Bill Cosby

Feature Use





A Solution…

Wasted Features

“ I know half the money I spend on advertising is wasted, but I can never find out which half. ”

Alfred Dupont

Wasted Features

“I know most of the features in modern software are never used, I just don’t know which ones… ”

Importance of Design

To make software that is easy to use, great care must be taken with the interaction design.

It must be aimed at a specific audience.

The designer needs to focus on users goals, rather than the interface.

Most software today flouts these rules.

Finding Goals

Understand the people the software will be used by, and the circumstances it will be used in.

e.g. A phone book for a receptionist.

- Cope well with interruptions
- Get the number needed quickly
- Be able to find **Marc Smyth** without embarrassment

This leads to knowing which features, and behaviours are needed.

Specialised Software

Specialised software has fewer functions, and so puts less stress on the scaling of the UI.

So replace general purpose software with more pieces of specialised software.

Consistency between these specialised applications becomes more important.

The End of Applications

Traditional applications are too heavyweight for giving this kind of flexibility. The development effort is too high.

Finer grained software, modules that can be installed into a framework, can allow this sort of capability.

Previous attempts have so far failed, not just for technical, but also for commercial and “political” reasons. E.g. OpenDoc.

The answer is not OpenDoc

OpenDoc and other similar systems allowed any document to become anything. It is the ultimate general purpose software.

As an editor included more parts to allow inclusion of more types of content it lost focus.

OpenDoc tried to please everyone with a single editing system…

Software needs to be specific

" I don't know the key to success, but the key to failure is to try to please everyone."

Bill Cosby

Functional Templates

A functional template would define which software modules were available when doing different types of work.

Created by software designers who have a deep understanding of the domain being served.

These designers can rapidly assemble specialised, well tuned applications to serve real needs.

Functional Templates

Specificity reduces the number of functions in the interface to those that are important, and hence reduces the scaling problem.

Since the same components would be reused wherever they were relevant consistency would be enhanced.

This reduces the cognitive burden on users.

Functional Templates

Being specific allows Templates to include unusual features that would not work in general purpose software.

Imagine a Novel Authoring Template.

It could have tools for recording characters details, chapter plots, etc. These functions are only appropriate to a small audience, and currently would be hard to make and sell.

Packaging

Packaging companies, akin to the Linux Distributors, could make specialised software from components bought from a wide variety of vendors.

Some modules would be of general use, some very specific.

Template definitions however would always be specific to a community.

All applications could ultimately be replaced this way.

Technology Wish List

Dynamic Linking
(Java, Smalltalk, DLL, .so)

Open Data Formats
(XML, PNG, SVG)

Architecture Neutral Code
(Java, Tcl, JavaScript)

Network Based Software Delivery
(Java Web Start, linux updates)

Standard, Evolving, Abstract Interfaces

Any Questions?

Pete Bagnall pete@surfaceeffect.com