Local Systems Meeting

These are a few opinions for the local systems meeting on Friday 22 July, which unfortunately I am not able to attend.

SAFE (the System for Admin in the Faculty of Engineering)

The features currently offered by SAFE include:

- **Single Signon** using the university's SSO system
- **Permissions** - a complex system of central and local roles, groups and restrictions
- **Site structure** - simplified URLs, e.g. /2010-1/COMS30500 for an academic year and unit
- **Admissions** - tracks central and local data about UG applicants, letters, visits, interviews
- **Editing** - e.g. the ability to create form letters to send to groups of applicants
- **Marks** - assessments, deadlines, penalties, anon student display, progress, timetables
- **Marksheets** - for examiners meetings, implements faculty rules, provides transcripts
- **Local data** - tutees, supervisors, tracking of tutorials, lab attendance
- **Searches** - for students etc., using a very wide range of criteria
- **Emails** - sends automatic or semi-automatic emails to groups, e.g. as reminders
- **Photos** - displayed from central source, opt-in for staff
- **PDF** - any web page which must be printed reliably can be converted to PDF

Principles

The principles which make SAFE successful can be summed up as:

- **Grass roots** - a system needs to be driven by those who use it, because people will simply hate or refuse to use a system which can't be made to do what they want

- **Quick prototyping** - people don't know exactly what they want, so it works best to produce something simple quickly, and use feedback to guide further development

- **Continual maintenance** - a system dies if it becomes static, it needs to be developed continuously in line with local feedback

- **Joined up** - apps must know who you are, what roles and permissions you have, what department you are in, what programmes and units you are involved with, and so on

- **Single source** - a lot of unnecessary admin work is caused by having too many copies of data, and keeping the copies consistent is an admin nightmare

- **Processes** - web based tools don't just provide a mechanism for getting things done, they also provide processes. For example, workflow, signatures, authorisations, tracking

- **Psychology** - online tools can also have a profound effect on psychology. Examples are deadline and penalty handling for students, and tools for use in staff meetings or processes

- **Devolution** - data is best maintained by those who have the greatest vested interest in it. E.g. local permissions within departments should be managed by department staff
Comments on Centralisation

These are thoughts on what I would do if asked to make SAFE go university-wide. For a start, although the SAFE system was designed with as few local Engineering faculty assumptions as possible, I would implement again from scratch, re-using the ideas and experience from SAFE and other local systems, rather than the code. Most of the issues below are, of course, about scale.

Hardware

I think this is a critical decision, affecting the overall look and feel of the system, the psychology of ownership, and the structure of the development team, not just how to get the system to work.

SAFE runs on a single computer (albeit a powerful one). That means we have put all our eggs in one basket, but we have done what we can to make the basket bombproof. This works for the faculty (we recorded over 900 users on the afternoon of the release of the UG results, without trouble). This is partly because we designed the system to keep all database data in memory as objects, directly accessible to read-only pages without memory turnover, and could afford to copy data from the datahub only overnight. Options for going university wide would seem to be:

- Use one computer. Unlikely to scale up.
- Use one computer per application. Only works if the applications are self-contained.
- Use one computer (or so) per faculty. Run identical software on each.
- Use an enterprise system. Share requests among identical multiple computers.
- Use cloud computing, e.g. the Google app system. A ready-made enterprise system.

I doubt if the first two options could be made to work well. The trouble with the last two is that every computer would need access to all the data, including all the locally added data in every faculty, school and department, which I guarantee would quickly become a huge bottleneck, with the slightest change made anywhere in the university needing to be propagated across the whole system. I think I favour the third option because it clearly spreads the load well, it spreads the local data well, (not needing to duplicate one faculty's local data across all the computers except slowly for backup, maybe), and it provides a possible focus for spreading the essential load of user support, data maintenance and development. It makes a positive feature of the potentially hideously difficult differences between faculties.

Software

I personally believe that conventional networked database services are not good enough. In fact, I don't think that even an embedded database is good enough. I think the data has to be directly available to the code (as in SAFE), with no memory turnover (as happens even in Google apps). The data in memory can be treated, as with SAFE, as a write-through cache, writing to any database system you like (e.g. Oracle, though I personally favour an embedded one, for reasons see https://wwwa.fen.bris.ac.uk/efac/db.html). It doesn't matter much what the programming language is, because the bottlenecks are certain to be the network and the database, but I personally wouldn't advise using a scripting language for a project this big.

Principles

Although the principles for SAFE might seem local, I think they should be adhered to in any central system, as far as is humanly possible. Outsourced systems or systems with insufficient local involvement develop a great deal of inconvenience, frustration, and downright hatred.