

make work make sense

Process, data and work

Process and data

It has become commonplace to distinguish between two overall approaches to system design – ‘process-oriented’ and ‘data-oriented’.

Process-oriented design focuses on system functionality: what the system has to do; what it has to display, produce etc. Data-oriented design focuses on the data which the system needs to deal with.

Traditional approaches to best practice in system development talk about process design and data design as the twin pillars of software engineering. They stress the importance of taking an iterative approach by which process design and data design feed and cross-check each other.

A currently popular synthesis between process-orientation and data-orientation is the encouragement of ‘declarative’ over ‘procedural’ design. ‘Declarative’ design essentially treats process as a special case of data.

This book takes a different view. ‘Declarative programming’ for example is certainly a laudable, theoretically sound and economically efficient way of building a computer system itself – when that computer system is considered in isolation from what it is being built *for*. I don’t mean just its ‘functionality’. A computer system is very often part of the machinery running a business – and very often an important part of that machinery. By extension, a major change in a computer system – including of course the extreme case of implementing a wholly new system solution – will almost always represent a major change in the machinery of the business. And it goes without saying that the business change itself will typically be the planned ‘end’ to which the system change is the ‘means’ – a new product, new method of distribution, new level of service, expansion of capacity, and so on.

This book sees the economic and logistical success of the planned business change as more important than the independent success or economic efficiency of the system development which is there to support that business change.

There should be nothing remarkable about that statement. It should be mathematically self-evident. The total planned

benefit of a business change (£B) should, generally speaking, be greater than the total estimated cost of that change (£C). The system development costs relating to the change (£D) should be, again generally speaking, only a proportion of the total estimated cost of the business change (£C). A system design methodology aimed at increasing the likelihood of achieving £B and/or increasing the size of £B should, other things being equal, be favoured over another system design methodology aimed at reducing the size of £D.

But we are jumping the gun. It is time for a four-letter word.

Work

I like work: it fascinates me. I can sit and look at it for hours.

Jerome K Jerome, *Three men in a boat*.

The approach in this book considers 'work' as a fundamental architectural component, one which cannot be reduced to either 'data' or 'process'.

Although 'work' can be analysed and expressed as data, this does not mean it is only a subset of data. And where 'process' means (as it very often does) 'what the *system* does', 'how the *system* behaves', rather than 'the work the business carries out', then 'work' is not reducible to 'process' either.

What is work? Work could be described in many ways. Something which perhaps processes objects into other objects? Or something which processes objects in an intentional, targeted way?

I am not attempting a universal definition. I want to look at a particular kind of work, but one which increasing numbers of people are involved in and feel the effect of. This is administration work – whether it is processing applications, granting approval, compiling statistics and so on. It could be in sales, financial services, central or local government, education, tourism, or one of hundreds of other walks of life.

There are some general – and very familiar – features of the sort of work I'm talking about.

It normally provides a service to an 'end-user' or customer. Implicitly or explicitly it involves carrying out a 'request' from, or on behalf of, that end-user or customer.

make work make sense

It is normally rule-governed. There are right ways and wrong ways of processing a tax return; booking a holiday; processing an insurance claim. Because of these rules, it is normally possible to distinguish between 'standard cases' and 'exceptions'. Rules also determine what has to happen in what order; and how you know when the work is complete, when the request has been carried out.

A third common feature of this kind of work is that it is becoming increasingly supported by computer systems. These systems are becoming increasingly sophisticated, as the services this sort of work provides are themselves becoming increasingly sophisticated.

The work itself is carried out by a mixture of people and machines. Fewer and fewer people carry out the basic functions. More and more people are managing (monitoring, changing, improving) the resulting processes and dealing with exceptions and special cases: making rules rather than following them.

These are the systems this book is about. It is about an approach to designing and implementing these increasingly sophisticated systems, and designing and implementing the changes in work and management practices which our increasingly sophisticated business world requires.

Business process

I now want to define the word 'process' in a specific way. I mean 'process' as in 'business process', not as in 'computer process'. 'Process' in this sense doesn't mean just what a computer system does or how it behaves, but the work that has to be done to carry out something for a customer, regardless of whether a computer system is involved or not.

It is 'process' in this sense which in the next chapter we will use to build up the concept of 'process modelling', which is crucial for the architectural approach this book is about.