# EVOLVING TASK ORIENTED SYSTEMS

*Paul Seaton and Tom Stewart*

System Concepts
Museum House
Museum Street
London, WC1A 1JT
tel +44 71 636 5912
email Tom_Stewart@Hicom.LUT.AC.UK

**ABSTRACT**

This paper describes an approach to developing systems which can be summarised as 'analyse top-down, design middle-out, and build bottom-up'. A case study is described in which this approach is used to develop a system to support staff who select new products for a major UK company. The novelty of the approach lies in its use of task analysis to define an appropriate domain for the system and then the use of a working prototype to grow a system from the bottom up. The project involved using simple development tools which allowed the users to start getting business benefit from the system right from the start. Their use could therefore develop as the system evolved.

**KEYWORDS:** Task Analysis, Prototyping, User Involvement, Design Methods, Evolutionary Design, Bottom-up Methods, Graphical Interfaces

## 1.0 INTRODUCTION

Our client is a major UK organisation with over 200 retail outlets in the UK and an increasing number worldwide. It is highly centralised with the London-based head office responsible for selecting and buying all merchandise. Within head office the Departments are organised according to groups of products, covering a very diverse range. Mainframe computer systems are a vital part of the buying process, with facilities for monitoring sales, estimating and placing orders with suppliers, but there is a growing recognition, as in so many organisations, that current systems development techniques are failing to deliver the business benefits expected. The systems which are developed are often late, over budget and difficult to use.



Traditional systems analysis techniques have been effective for specifying the functional requirements for new systems, but they have been criticised for placing insufficient emphasis on the usability of the resulting systems.

Other system development approaches have been tried which encourage greater user involvement in system specification and development. However, this has not always been successful. Insufficient analysis early-on in the development process has resulted in system specifications evolving in an ad hoc and sometimes uncontrolled way. Prototypes and demonstration systems have been developed to reflect the preferences of some users without sufficient attention to the technical feasibility or costs associated with full scale implementation.

Putting it another way, analysis-driven designs have tended to be unusable whereas user-driven designs have been unbuildable. To make matters worse, staff in head office User Departments are making increasing use of spreadsheets and other PC tools and are becoming critical of the limitations of the mainframe systems and the outdated style of interface. The scale of development required for mainframe systems prevents them from being sufficiently flexible to reflect the changing commercial priorities of the User Departments. Much of the information used by the User Departments is held on spreadsheets and other PC packages. Up until now these systems have been viewed by the Information Technology Department (IT) as distinctly separate from mainstream systems development.

What is required is a method for developing systems which combines the business benefits of increased usability with the efficiency benefits of more ordered development and better engineered systems.

### 1.1 The Need for a New Approach

The above problems are wasteful of resource and represent a major lost opportunity. Better usability could allow managers to use the systems directly instead of through intermediaries and assistants, saving staff costs. But experience in other organisations shows that allowing managers to drive systems themselves has significant benefits in helping them to understand the business better and make far more effective use of the information provided. However, this requires much more than just a 'friendly' or 'pretty' interface. It requires that the user interface as well as the functionality of the system is matched to the real business tasks of the users.

Tasks in this sense are defined in business terms not system operation terms. Thus 'estimate sales' is a business task whereas 'update an amend estimate screen' is not. It is simply a means to an end.

Ideally, therefore, a task oriented interface reflects an understanding of the steps, options and requirements for performing the business task. Using well-designed task-oriented software is not only efficient, it is also satisfying. At every stage, the choices available in menus or other navigation match the real decisions. The right options fall easily to hand. The system is intelligent and makes sensible assumptions, based on previous actions. The defaults are sensible guesses and more often right than not. Even when the task is novel, the interface allows the users to reach exactly the information or the choices they require, although perhaps not quite so quickly or so elegantly as in the frequent tasks.

However the task-oriented development approach is in its infancy and there are major risks in rushing into new and untried methods of working. We therefore decided to gain real experience of its use in a limited trial development of an exemplar system before considering scaling up the approach to other areas.

### 1.2 Creating an Exemplar Task-Oriented System

In choosing the exemplar, we wished to ensure that the task area selected was important for User Departments. However, because we ultimately wanted to change the working practices and the attitudes of designers as well as users, it was important that we did not cut across existing IT activities. There already was a major system design activity adressing the problem of helping buyers (known as merchandisers) to estimate future sales

From early discussions with users in User Departments, the new product development activity emerged as a likely candidate. It is an important task area with little existing IT support. It is currently carried out by staff who are not heavily involved in other IT related activities.

We therefore proposed to develop an exemplar system to support these users in their new product development tasks, to demonstrate the business benefits of a more user centred approach to systems design.

In addition, we believed that:

. A successful system would provide the most powerful illustration of the benefits of a task-oriented interface.

. As a by-product of the development, we would be able to develop usable task analysis techniques and to create the tools necessary to support this approach.

. Since the process required close involvement with the staff in User Departments, our exemplar would demonstrate by example how to involve users in a disciplined way.

. The exemplar would also provide a sound starting point for linking into other areas of buying should the task oriented approach prove beneficial.

. Last, but not least, the exemplar would provide real business benefits to the chosen department, ideally far more than they expected and certainly more than they were promised.

### 1.3 Assumptions behind the approach

The most important requirement was that we would focus on real tasks. We intended to identify these through task analysis. However, we recognised that one of the dangers of our approach was that the new system would merely mimic the existing working practices. In order to overcome this constraint, we decided to implement a minimal working protoype system. Although this system supported the existing working practises of the users it was also sufficiently flexible to allow these practices to evolve to exploit the full capabilities of the system.

Such 'bottom-up' development helps to ensure that the initial system is useful as well as usable. But it does have the danger that it can be difficult to extend later. To avoid this, we took two important decisions.

First, the task analysis would be performed 'top down' from the departmental level. This would ensure that we could identify tasks which were capable of growth and

which were of sufficiently general applicability, ie they are tasks associated with meeting real business requirements rather than operating the existing systems.

Second, the system would be built from a limited set of standard modules to ensure that future growth could be achieved without major redesign. This standardisation included the overall 'look and feel' of the user interface. The modular approach also allowed us to be able to deliver 80% of the users' requirements quickly.

In addition we made it clear that we were not aiming for a 100% solution with the tools we had available. This helped us to manage the user's expectations to a realistic level which proved to be one of the most critical factors in the success of the approach.

We believed that the combination of 'top-down' analysis, middle out design and 'bottom-up' build would give us the best of all worlds.

## 2.0 ESTABLISHING THE INFRASTRUCTURE

There were two infrastructure components upon which our development depended:

- a user interface standard to serve as a model to guide the growth of the system and which could encompass all the functionality which the system might eventually develop

- a hardware and software platform suitable for immediate applications as well as providing future growth path opportunities

Our inital investigations into this area showed that Microsoft's Windows 3.0 was a suitable vehicle for the provision of a common interface on the IBM PS/2 workstations within User Departments. It is widely accepted across the industry and has an ever increasing portfolio of supporting products, ranging from simple spreadsheet packages to full development environments.

Using a product such as Windows provided us with a Graphical User Interface (GUI) which could potentially encompass all applications within the User Departments. It also provided the basis for a common user interface in that it has its own implied interface standards which are consistent across all its supporting products. Within the Windows framework it is then possible to use further GUIs to front end individual applications and again they adopt the common Windows interface standards. Such a feature allowed us to remain independent of products and suppliers, allowing new products to be 'plugged in' as and when they become available.

To support the approach a comprehensive in house user interface standard was developed independantly of Windows 3.0 and this ensured the approach could be extened to other technical platforms such as OS/2.

## 3.0 THE EVOLUTIONARY APPROACH

Some proponents of analysis methods seem to imply that a suitable design emerges almost as a by product of good analysis. We would argue that if this is the case, then it probably means that the analysis was conducted at the wrong level. In other words, the analysis probably focused on potential design solutions rather than on establishing requirements.

No matter how good the analysis method, there is still a vital design activity to create a system which matches these task requirements. If the analysis has been done properly, there will be several design options. The task analysis data may make it obvious that some of these options are more likley to succeed than others but they should still be valid options.

Our priority was to develop a working prototype which provided sufficient functionality and usability to get the users started. One of the key attributes of this approach is that the prototype works from the start ie it delivers value to the users immediately.

Often what are called prototypes are really just demonstrations of 'wouldn't it be nice if' features. These may mislead users about what is realistic and may focus attention on details of presentation or content which are less important than functionality at the early stages.

By providing users with something which works right from the start, we deliberately aimed to focus their attention on task related issues. That is not to say that we ignored their comments on the overall appearance and presentation of the system. First impressions are very important but too many changes can be confusing. Our strategy was to start a process whereby they could evolve a new method of working as the system evolved.

Paradoxically, this places more emphasis on the designers of the prototype understanding the users' real task requirements and anticipating future requirements. Thus the system has to be designed with a degree of 'future proofing' built-in. In practice, this involves adopting a

modular approach, paying additonal attention to the maintainability of the design and the code and recognising that some design effort will have to be thrown away as new ideas evolve.

There is no established methodology for developing task-oriented interfaces. Indeed, our own approach evolved during the course of the project. We believe this approach is repeatable and indeed we are currently planning to repeat it for a different application area. We describe it below in its evolved form, which although not final, is sufficiently mature to be useful.

The approach involves the following steps:

. Identify target users and define focus of future system

. Perform Task Analysis

. Agree Scope of working prototype

. Perform Task Modelling

. Perform Task Mapping

. Design working prototype

. Help users to learn through experience

. Manage Change

. Grow system within department

In the following sections, we discuss each of these steps in more detail.

### 3.1 Identify Target Users and Define Focus

Traditionally it is normal to define a boundary to the system being proposed. With a task oriented approach, this is not appropriate. Until the task analysis has been performed, it is not at all clear where the 'boundary' should be. Indeed, with front end tools such as Easel/Win (a PC based GUI for external databases especially those held on mainframes) it may be unnecessarily restrictive to think in terms of boundaries at all. We therefore identified a focus for our system and left the question of boundary to a much later date.

From our early investigations of User Departments, the New Product Selection Team emerged as a suitable target group for this approach. They play a key role in new product development and represent a combination of experienced and novice computer users. The staff in the chosen department already used spreadsheets to a limited extent to support their buying activities.

### 3.2 Perform Task Analysis

Although Task Analysis is an established technique in ergonomics, it has mainly been applied to tasks which have a large physical component eg operating machine tools. Tasks which are primarily mental are much more difficult to analyse not only because almost all the activity is hidden but also because the tasks tend to be ill-structured and involve subtle nuances.

A recent review of task analysis techniques suitable for user interface design carried out by System Concepts had identifed one technique - Hierarchical Task Analysis - as most promising. We therefore used that as our starting point but rapidly found that we had to evolve our own simplified version for it to be effective in a dynamic retail environment.

The main technique used to collect the data involved structured interviews with relevant staff in the User Departments. This enabled us to draw a hierarchy of tasks showing the linkages between the different levels. In this context, a task is an organised sequence of behaviour which is performed by a person or a group of people in order to achieve a business objective.

The first stage in the Task Analysis is to establish the highest level Departmental Tasks. These represent the main mission of the Department as a whole, eg Develop New Product. The next level is the Primary Task. These contribute directly to the business objectives of the Department eg Decide Overall Strategy (for developing a new product).

Below the Primary Task, the secondary task is a task which must be performed in order to perform the primary task eg Establish Lessons Learned (from previous seasons sales figures). Sub-tasks are the lowest level of detail and may be associated with document and screen based information sources and output.

Figure 1 shows the task hierarchy that was adopted throughout the project.

We found that it was important to conduct the analysis from Department Task down to Sub-Task level for the specified area. This helps to ensure that no false assumptions are made during the design phase and that all associated tasks are incorporated in the system.

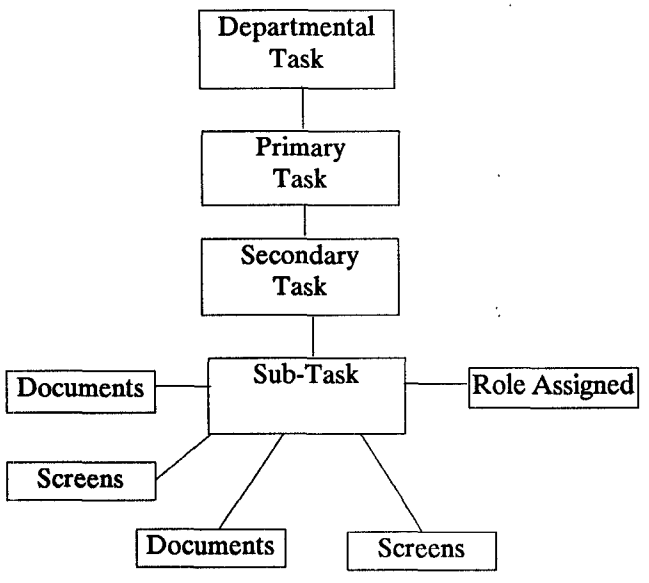


Figure 1 : Task Hierarchy

Figure 2 shows a detailed example of the break down of the Sub-Task : Update Developement Programme.

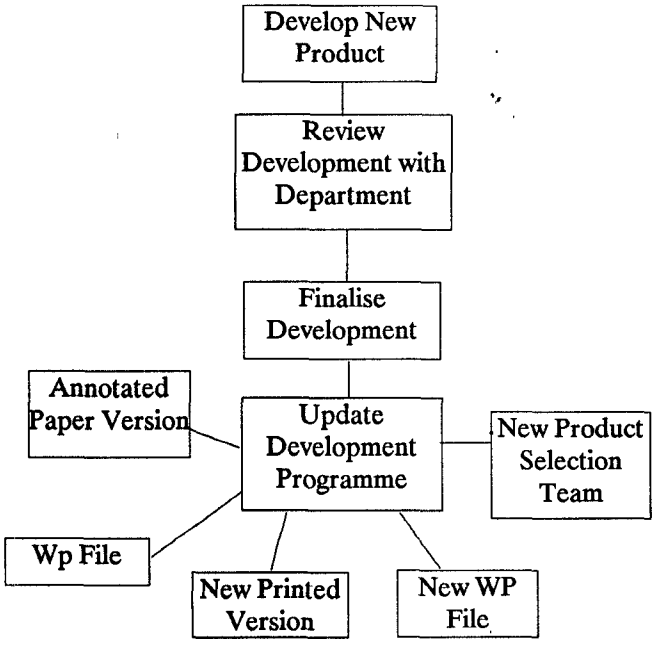


Figure 2 : Task Hierarchy for Sub-Task; Update Development Programme.

These tasks were then further reviewed in order to identify one which would be a suitable target for the working prototype. Ideally this should be a task which is important, which is amenable to computer aid and where improvements in current performance on the task would be important in achieving the departments overall business objectives.

The most suitable candidate was the sub task 'Update Development Programme'. This document formed the basis of the department's information about the characteristics and status of products under development.

### 3.3 Agree Scope of Working Prototype

The prototype was intended to simplify the maintenance and publishing of the two documents by the provision of a simple 'electronic list' to replace the current manual method. Such a list could form the basis of a different way of working which could help them keep better track of new products and communicate progress within the department.

Each feature of the system was negotiated with the users in terms of timescale to build and the ability of the technology to deliver. This concept was central to our approach in that it allowed us tto quickly provide the users with a working 'prototype/system' that demonstrated its benefit immediately. We therefore had to very carefully manage their expectations of what the system would deliver. By doing this effectively we were able to provide them with the basic system quickly and then use their experiences of using it to 'evolve' the system to better reflect the true user need.

### 3.4 Perform Task Modelling

Once the Task Analysis had been completed to the Sub-Task level, and the scope of the prototype agreed with the users, then the next stage in the process was to Model the Sub-Task in terms of how it would logically be performed. This process was achieved by the use of a Task Model. This model reflects the Actions and Objects relating to the Sub-Task and should be independent of any pre-conceived view of how the system will function.

Figure 3 shows the Task Model for the Sub-Task Update Development Programme. Subsequent Sub-Tasks can either be incorporated into an existing Task Model or a new Model developed.

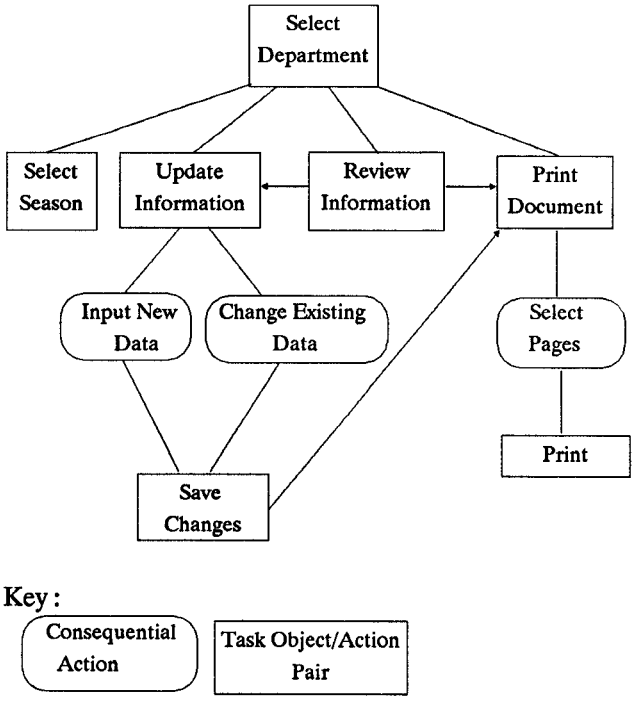


Figure 3 : Task Model for Sub-Task Update Development Programme. The arrows indicate the sequence in which the Actions are performed. The bottom of each 'column' represents a potential exit point from the dialog.

Both the functionality and the design of the interface are centered around the task model produced. This helps to ensure that the system reflects the tasks performed by the users.

### 3.5 Perform Task Mapping

The next stage was to map the Task Objects and Actions onto interface objects eg Dialogue Box or Action Bar etc. This required a working knowledge of the In-house User Interface Standards.

Each Task Action/Object pair is taken in turn and the mapping procedure applied to it. The procedure involves establishing the options and rules, (determined from the task analysis and any limitations of the technology being used) and then mapping the Task Action/Objects onto Interface Objects and Controls. Such a process ensures that maximum task orientation is achieved in the specification of the user interface and that the usability factor is maximised.

### 3.6 Build Working Prototype

Through consultation with the users it was established that the minimum requirement was the provision of a simple 'electronic list' which would allow easy maintainance of the data and automatic generation the required documents. From the tools at our disposal at that time we considered the Excel spreadsheet package as the most appropriate choice with which to build the application. Excel is a very powerful package however, we considered it complicated to learn and intimidating to users who are not familiar with spreadsheets.

The majority of our users had never used the PC before and had difficulty even switching it on. In any case our users had neither the time or the inclination to become sufficintly familiar with the package to allow them to use it for themselves. It was for these reasons that we decided not to offer them the Excel package to use directly but decided to customise the Excel interface to provide the users with only features they required. This allowed the spreadsheet itself to be hidden from the users along with the majority of the Excel functionality. The documents were formated by Excel macros extracting the required information from the various databases contained on the spreadsheet itself. The update of the database was through an Excel dataform, this had the added advantage of offering the users online reference to the data and it is hoped that with time they will begin to use this facility more often and only produce hard copies when absolutly necessary.

### 3.7 Help Users to Learn by Experience

As already stated the prototype was introduced to the users as quickly as possible. We found that this had the effect of gaining the users interest and getting a more involved commitment to the project from them. It also had the effect that they began requesting features that they had not at first seen as being necessary. By managing thier expectations we were able to keep the prototype functional and add additional features once they had been agreed as feasible (in terms of technology and timescales). Once the users realised that certain features would mean a more lengthy development they soon became less vital and they began to accept small drawbacks in favour of having a working system quickly. To enable a quick implementation of the prototype and to gain the commitment of the users we had to 'hold the hands' of the users in the initial stages.

Individual training sessions were held for all users involved, which included an overview of Windows itself as well as an introduction to the prototype. Since the interface design matched closely the task in hand after a quick overview of the functions available they soon became comfortable in its use. We then provided support as and when requested and helped the users to actually use the system to produce production documents.

### 3.8 Manage Change

Although the basis of the approach was to be supportive of the users and to be attentive to their stated requirements, this did not mean that we immediately actioned every request for changes. Indeed, we instituted a degree of change control very early in the process.

Once the prototype was introduced into the department and was being used to manipulate 'real' data, change control was instigated. This was achieved by logging each request to change the system on a change request form.

All such requests were then assessed to determine their full impact on the prototype and estimates made for how long it would take to deliver. A record of each change was kept on a Change Response Form. Once the impact was established, we relayed back to the users what the change would mean to the prototype and how long it would take to deliver. We quickly found that users soon learnt to prioritise requirements in terms of 'nice to have' and necessary features.

### 3.9 Grow System

The next step is to grow the system within the Department in order to extend its usefulness and value to the New Product Selection Team and to extend its usefulness to other members of the department. Our strategy involves gentle nudging of users and potential users as well as responding positively to ideas and suggestions.

We also expect to be able to suggest extensions ourselves based on further task analyses and also from our continued exposure to real users and their day to day requirements.

## 4.0 CONCLUSIONS

At the time of writing this paper, the initial stage of the project had been well received by the users. They were using the system as a routine tool to support their work and were beiginning to explore the further potential of the various utilities provided.

From our regular visits and informal discussions, it was clear that there were many possible opportunities for main-frame links and extensions to the functionality which could greatly help the users in their work.

In retrospect, one of the most important parts of the process was to treat the users as partners. However, this did not mean that we 'gave them what they wanted'. Indeed, one aspect of treating them with a degree of respect was to point out that there were constraints and that some of their requirements were easier to achieve than others. Having established a degree of mutual trust, this was not a major problem.

The most significant problem area turned out to be relating what we were doing to conventional system development activities. For example, typical project management reporting required us to answer questions such as 'when will the system be implemented?' and 'when will the project be completed?' which we found difficult to answer. Answers such as 'its already been installed for two months' and 'it depends' are not entirely satisfactory to traditional project managers who like to minimise uncertainty and structure the development process.

Nonetheless, we believe that this approach has considerable potential for identifying worthwhile systems to support the business tasks of real users. That may not be everything but it seems like a worthwhile start.