

**Software Quality Management in End-User Programming and Object Based Rapid Application Development (RAD)**

Simon Price, Li Lin Cheah and Phil Hobbs

TLTP Economics Consortium  
Centre for Computing in the Social Sciences  
University of Bristol  
8 Woodland Road  
Bristol BS8 1TN

Email: [Simon.Price@bristol.ac.uk](mailto:Simon.Price@bristol.ac.uk)

**ABSTRACT**

Worldwide, end-user programmers outnumber software engineering professionals by an increasing margin. Despite this, little is known about the quality management of end-user programming even though it is widely deployed in business critical areas. This paper analyses quality management in a large scale end-user software development involving object based Rapid Application Development (RAD) tools. The limited applicability of existing code-based metrics in such environments is demonstrated. Effective methods adopted in this case study are generalised to propose ways of raising quality in end-user programming and RAD.

## 1. INTRODUCTION

This paper presents a post delivery analysis of quality management in the WinEcon™ project focusing on the problems associated with end-user programming and object based Rapid Application Development (RAD) tools. It is argued that the quality management issues addressed in this paper have widespread applicability to other object based RAD projects and to the increasingly business critical area of end-user programming.

The WinEcon project was a collaborative development between a consortium of eight UK universities, managed from the Centre for Computing in Economics at the University of Bristol. Over a three year period the project received grant funding under the UK Teaching and Learning Technology Programme (TLTP) to produce computer based training materials covering the entire first year economics degree course (1). The resultant WinEcon software was beta tested in over 90% of UK universities and is currently being trialed in over 200 universities around the world. It was recently awarded a medal by the British Computer Society in the 1995 software awards for innovation and excellence in the IT industry.

However, the eventual success of the project occludes a host of quality management problems which arose during development. The most serious amongst these were directly related to the adoption of end-user programming and object based RAD tools. Whilst these two characteristics were undoubtedly central to the high quality of the delivered software, they confounded attempts to apply many traditional software quality management procedures. In commercial organisations, end-user development of spreadsheets, databases, queries, scripts, macros and RAD programs face the same problem.

## 2. RELEVANCE TO SOFTWARE QUALITY MANAGEMENT (SQM)

### 2.1 SQM and end-user programming

In 1995, Capers Jones (2) of Software Productivity Research compared the number of software professionals with an estimated number of end-user programmers in the US and in other countries around the world. He wrote:

*'According to Software Productivity Research demographic surveys, about, 1,750,000 people in the US derive their salaries directly from work associated with software. In addition, there are perhaps 10,000,000 managers, engineers, architects, accountants, and other knowledge workers who know enough about programming to build end-user applications with spreadsheets and other tools.'*

He estimated, with an admittedly high margin of error, that the corresponding figures for the UK are 385,000 and 1,750,000, placing the UK third in the World behind Japan (second) and the US (first). He also observed that, for developed countries, the gap between these figures appears to be increasing.

*'On a global basis, there are perhaps 12,000,000 professional software personnel, but there are more than 30,000,000 end users who can program. . . . The end-user*

*programming population seems to be growing at more than 10 percent per year worldwide, but the growth rate for software professionals is now down to a single digit in industrialised countries . . .*

Assuming these estimates are even remotely accurate then end-user programming is, or will soon become, the largest area of software development (when measured by the number of personnel involved, and possibly by expenditure also). The software quality management of this end-user application development seems likely to become a major concern for organisations around the world. Not least, efficient and effective SQM processes need to be developed which will work in such an environment; it is not clear whether current software industry SQM processes can be applied without modification to end-user programming.

The development effort in the WinEcon project was dominated by the end-user programming involved in the creation of the interactive economics teaching software - the courseware content. The effort profile and staff composition changed over the course of the project but the staffing statistics in table 1 show the degree of end-user programming effort involved at the height of development.

Table 1 Peak staffing statistics for the WinEcon project (Year 1, 1993)

Staff Type	Peak Staff	Total Real Years
academic authors	26	5
economist programmers	14	14
programmers	2	2
admin./advisory	2	1.25
Total	44	22.25

Assumptions:

1. Each academic author contributes one-tenth of their time to the project.
2. A week consists of 35 hours and staff work for 46 weeks a year.

Taking academic authors (i.e. economics lecturers directing the top level educational and economic content of the software) and economist programmers (i.e. economics graduates with, typically, minimal or no prior programming experience) as end-user programmers gives a total of 91% of actual staff and 85% of effort. Clearly, WinEcon is primarily an end-user creation and it seems reasonable to consider the wider applicability of lessons learnt during the quality management of its development, particularly given the predicted future importance of end-user programming SQM.

## **2.2 SQM and object based RAD tools**

Rapid Application Development (RAD) evolved from the practice of rapid prototyping, taking it to the logical extreme of iteratively refining a prototype until it becomes a deliverable product. In common with prototyping, RAD seeks to improve the development process by allowing the gathering of requirements as development proceeds and by enabling a greater degree of co-operation between developers and users. RAD has gained much momentum in recent years and yet, despite this, little is known about software quality management of this increasingly important area (3).

RAD tools can be divided into two categories: repository based and object based. The Asymetrix ToolBook authoring tool (4)(5) used in the WinEcon project falls into the object

based category along with similar tools such as Visual Basic. The description given in (3) of a typical object based RAD tool serves as an accurate description of ToolBook. It begins:

*'Development is WIMP based around the use and alteration of predefined components/objects. For example, the developer will mouse select from a palette of standard window widgets and place one on the drawing area. It can then be customised and placed on the window, gradually building up an HCI front end. Each widget is an object with data, functions and external message interfaces ...'*

RAD is listed amongst Edward Yourdon's eleven silver bullets in (4). While he states that '... as a combination of tools and techniques, RAD has much to offer ...', he also remarks that '... used alone ... almost certainly will *not* turn out to be the silver bullet some vendors are promising'. Similar concerns are expressed by users of RAD in the survey in (3). While it is acknowledged that RAD is a definite step forward, software management and quality control of the RAD process and its products are flagged as major concerns. Indeed, similar concerns were raised within the WinEcon project at its outset by the technical lead (Price) but the Economics Consortium's executive committee elected to adopt the ToolBook object based RAD tool on grounds of the high degree of end-user involvement it facilitated. This situation is remarkably similar to that in commercial organisations where the MIS department recommends against the use of a particular tool but individual users elect to use the tool on positive feature grounds anyway. Software quality management lessons learnt in the WinEcon project are therefore of direct interest to such organisations in addition to general users of object based RAD tools.

### **3. OBSTACLES TO QUALITY**

The WinEcon project faced a number of obstacles to developing a high quality product. These are discussed below and related to end-user programming and RAD in the wider sense.

#### **3.1 Absence of relevant standards**

There are no standards specifically governing the development of computer based learning materials with RAD tools such as ToolBook. Neither is there a commonly agreed best practice in the area. Process and product metrics for this type of development have not yet been established (let alone validated or adapted for end-user programmer use). This situation is in keeping with the current situation with RAD in commerce and industry.

#### **3.2 Unprecedented scope and scale**

The WinEcon project was, and is, the largest single computer based learning package for teaching introductory economics in the world to date. It presented a considerable challenge in terms of obtaining both a technically cohesive and operationally consistent package. However, a quality management problem of at least this scale probably faces numerous organisations considering the prevalence of end-user application development. Although it may not always be desirable to integrate the applications built by numerous end-users in an organisation into a single package, there are strong '-ility' (e.g. usability, reliability, maintainability, reusability) arguments for their having consistent interfaces and architectures.

### **3.3 End-user programmers**

Two development models were considered: a centralised programming team receiving specifications from the economists (i.e. end-users), or a small (2 person) centralised technical lead with most development being undertaken by the economists themselves assisted by economics graduates. Fully briefed of the technical risks of end-user RAD, the consortium still chose the second model to achieve a high degree of end-user involvement in the development process and to allow the evolutionary determination of requirements.

This decision seems to be a part of the wider, apparently unstoppable, trend towards end-user programming which poses such a challenge to software quality management. End-users are bound to know less about almost every aspect of software engineering and quality management than trained, experienced specialists.

### **3.4 Physical separation of developers**

By necessity of their other duties, the WinEcon developers (end-user programmers) were based in each of the eight universities making up the Economics Consortium. This physical separation makes communication and the interchange of ideas and techniques difficult.

Eight development sites is small by comparison with the number of physically separated end-user programmers found in almost any large corporation. Even so, the obstacles to quality are the same as in the WinEcon project but on a much larger scale.

### **3.5 No direct managerial control of developers**

The Economics Consortium is directed by an executive committee. The Centre is then responsible for managing the day to day development activities - which are carried out by the very same economists who make up the executive committee. Such arrangements rely entirely on the goodwill and co-operation of all concerned, so any quality management measures have to be introduced by persuasion rather than directive.

Although this structure may seem alien to a commercial environment, it does mirror the same problems MIS departments, particularly the technical policy makers, face in persuading end-user programmers to conform to software quality management practices.

### **3.6 New organisation with finite life**

The WinEcon project had a fixed budget over a fixed period of time. Most of the developers were on fixed term contracts and could be excused for taking a short term view of the development process. In the short term view, it matters considerably less whether the software produced is maintainable or, in the extreme, whether it is of deliverable quality. This problem is not uncommon in many walks of life where staff move through positions within a company or between companies. Consequently, organisations wishing to manage the quality of end-user built applications must take this into consideration.

## **4. KEY QUALITY IMPROVING MEASURES ADOPTED**

A number of quality management measures appeared to have a profoundly greater effect on the quality of the software produced. These are discussed below.

## **4.1 Guidelines**

The production of a paper based set of programming and design guidelines was a relatively low cost but invaluable quality management measure. The 'Authoring Guide', as it was known, took one person approximately three weeks to produce the first version. Importantly, the guidelines went through several iterations before they were eventually frozen. Feedback and input from the end-user programmers contributed to each new iteration.

Looking beyond WinEcon and to end-user programming in general, it would be an interesting question to ask what proportion of commercial organisations have guidelines, rulebooks or handbooks aimed at end-user programming?

## **4.2 Templates and Model Examples**

Despite the general success of the paper based guidelines, there are aspects of development with a predominantly graphically orientated RAD tool which are not well conveyed on paper. Details such as colour, orientation, layout and interaction are much better conveyed by a library of model examples produced in the tool itself. For the WinEcon project, this was taken a step further and the library of examples became usable widgets or software components.

The library of model examples worked well in situations where the example was a largely self contained object. When the component had to be used in conjunction with others, the end-users devised numerous inconsistent and incompatible ways of doing so. Therefore, a template was produced by the lead programmers to automate, standardise and restrict the use of these widgets. The end-user programmers then developed their material within the template. As with the guidelines, the model examples and the template went through numerous iterations but with the added cost of upgrading software from older versions of the template to newer versions.

The idea of templates and model examples for end-user developments with or without object based RAD tools is highly transferable to a commercial environment; it has the potential to reduce bad practice, demonstrate good practice and to standardise use. It also enables organisations to achieve gains in efficiency by reducing repetition and duplication around the organisation.

## **4.3 Peopleware**

Guidelines, templates and component libraries can only go so far in raising quality. Ultimately, the quality of the material developed relies upon the developers themselves - the peopleware described and discussed at length in (7).

Even in the co-operative academic environment of the WinEcon project there were contracts between the member institutions of the Economics Consortium. Whilst this was undoubtedly an essential managerial arrangement and a clear disincentive to failure, it could not in itself guarantee that development would not go astray.

Arguably, greater quality incentives were derived from the team spirit and rivalry built up as the project progressed. Regular newsletters, an email discussion list and periodic meetings helped bind the team and simultaneously encouraged the exchange of successful experiences. Management took care to circulate external successes of the software, such as praise from visitors and favourable comparisons with similar developments elsewhere. This helped to foster a feeling of achievement and a desire to maintain quality.

Within a commercial organisation, orchestrated communication and exchange between physically separated individuals using the same end-user programming tools would most likely result in similar quality benefits.

#### **4.4 Software Inspections and Reviews**

Some way into the WinEcon project it became apparent that the informal testing and review procedures adopted were becoming less and less effective at picking up faults in the software. In addition, diverging interpretations of the project guidelines were emerging at different development sites. Consequently, the Centre instigated a programme of formal inspections modelled closely on the Fagan Inspections expounded in (10) and (11). The first inspection was targeted at code written in the ToolBook scripting language. This proved to be far too restrictive a focus and subsequent inspections were expanded to cover the user interface (usability and HCI) plus the economic content itself. However, for reasons discussed in the next section, the inspections were never properly conducted as full Fagan Inspections but were more along the lines of highly structured reviews. Specifically, process improvement metrics were not kept, logged defects were only followed up through subsequent reviews, and review meetings were eventually replaced by email and postal exchanges of documents.

As with end-user application development in general most computer based learning developments (in the authors' experience) do not employ any sort of formal inspection or review at all. Therefore, even the simple structured review programme adopted by the WinEcon project brought dramatic improvements in quality. Cited benefits included:

- increased '-ilities' (e.g. reliability, maintainability, usability)
- the positive educational side effect
- knowing that poor quality will be detected
- satisfaction of passing a review
- a relatively objective definition of when a software module is finished

This result suggests that similar reviews or inspections of end-user built applications in commercial organisations might lead to equally dramatic improvements in their quality. In fact this may be the only effective way of really assuring quality in this area.

### **5. ANALYSIS OF PROBLEMS WHICH AROSE**

#### **5.1 Informal culture**

Attempts to introduce formal software quality management into the informal academic environment in which WinEcon was developed were largely unsuccessful. An attempt to introduce Fagan Inspections, with all the preparation and administration they entail was not well received. However, cut-down structured reviews based upon a checklist and conducted in a semi-formal manner were well received and, eventually, much requested. Internal reviews were also conducted at the separate development sites in addition to the 'official' review programme run by the Centre. Over time, new formalisms were accepted more readily by the end-user programmers as the value of previous ones became clear. Such a stepwise approach to introducing software quality management into any end-user development might be a promising way forward. Reviews, perhaps in an advisory capacity, might be a good first step for organisations to try.

## **5.2 Shortage of SQM expertise**

Given the composition and areas of expertise of the staff of the WinEcon project, there are far more end-user programmers than specialist programmers (20:1). This is a necessary feature of the development model adopted by the project and similar or greater ratios could be found in commercial organisations. Consequently, there is a shortage of software quality management expertise for reviews and other procedures. Organisations should take the effort required to perform SQM of end-user applications into account when fixing their budgets for MIS departments.

## **5.3 New version of development tool**

Midway through the WinEcon project a new version of the ToolBook authoring tool was released and manufacturer support for the older version was dropped (7). This is a common problem in all software tools and concerns as to how well RAD tools facilitate upgrades to newer versions were expressed in (3). The upgrade for the WinEcon project was straight forward (details of tasks, effort and cost are given in (7)) due to the use of a template and tight coding standards policed through the reviews. Upgrading the software largely consisted of just upgrading the template. This is quite a strong argument for the use of these techniques in both object based RAD and end-user application development, as it is unlikely that software tools will ever stop going through new releases.

## **5.4 Limited applicability of traditional metrics**

Many potentially useful traditional software product and process metrics depend upon measuring features normally associated with text based programming languages. In the WinEcon project it was initially planned to record some of these metrics for code written in ToolBook's OpenScript programming language. Reliable quality, cost and effort estimators were hoped for. However, ignoring the informality problems discussed in 5.1, another problem also emerged which led to these plans being abandoned. One of the few process measures collected in the project was the times taken to perform reviews of the interface and of the code. These are shown in table 2 and figure 1 for one complete round of reviews. During the course of the project, modules were reviewed between two and four times.

Table 2 Review times for one pass through all 25 modules

	Interface	Code	Total
Review time (hours)	122.5	59	181.5
Percentage of development time	1.74	0.84	2.58

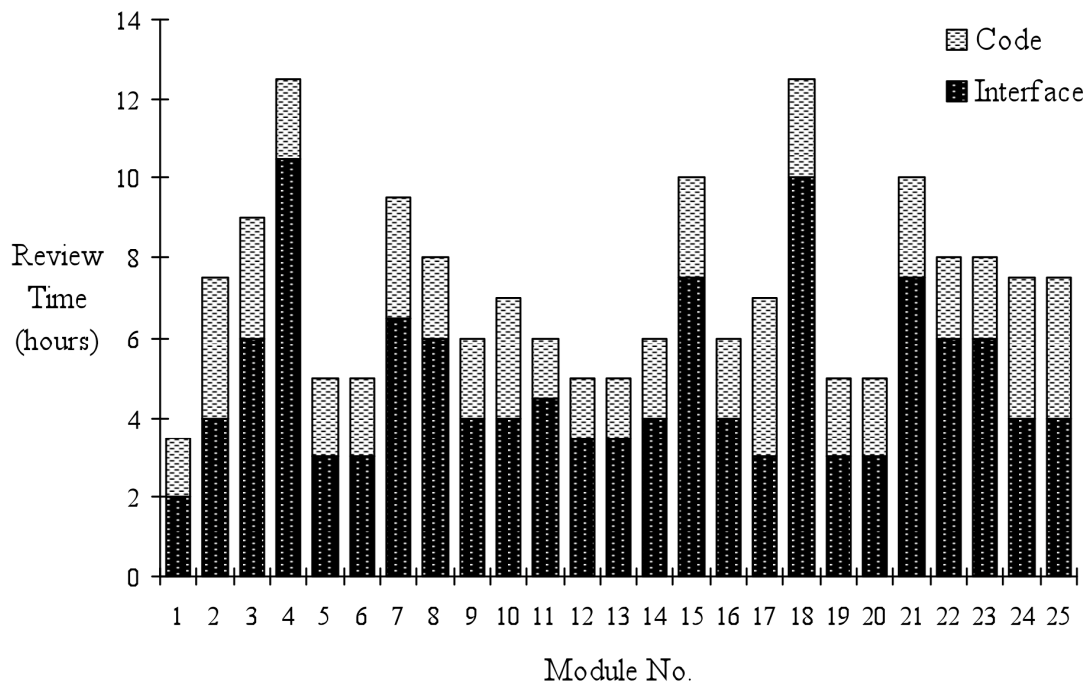


Fig 1 Review times for one pass through 25 modules

It can be seen from the table and the graph that the code component of a review is almost always less than the interface component. In fact the total code review time is only half the time of the total interface review time. However, the data in table 1 and the 1:2 ratio omit another component of the review for which no precise values were recorded: economic content review time. An economic content review is estimated to take approximately the same amount of time as an interface review and so incorporating this into the totals produces the estimated 1:2:2 ratio of code to interface to content. From this it is clear that reviewing the code only takes one fifth of the total review time.

The faults discovered in the three review components were observed to be largely independent (e.g. an error in the code did not often imply a corresponding fault in the interface or the economic content). Given this, it seems reasonable to assume that a reliable quality, cost and effort estimator might *possibly* include metrics from each of these three review components: code, interface and content. Furthermore, based on the 1:2:2 ratio, it seems likely that the code makes only a small contribution to the overall quality function.

Bearing in mind that WinEcon was a production project, not a research project, the plans to employ these code metrics were dropped; it simply could not be predicted with any degree of certainty whether the effort expended in collecting code metrics data would pay back within the life time of the project.

This attitude is likely to prevail in many object based RAD or end-user developments and there is a clear need for further research on metrics in these areas.

## 6. CONCLUSIONS

The WinEcon project was a collaborative development to produce computer based training materials for teaching introductory economics. It employed a number of quality management methods to overcome obstacles to quality which arose out of the adoption of end-user

programming and an object based RAD tool. These methods may have widespread applicability to all commercial organisations allowing end-user programming or RAD. Of equal importance are the unresolved problems which raise serious concern over the current lack of knowledge about quality management in such environments. If end-user programming in particular continues to increase as expected, then this could become a major software quality management threat to organisations around the world. It may be so already, but without adequate quality management, how can anyone tell?

## 7. REFERENCES

1. SLOMAN, J. WinEcon, software review, Economics Journal, 1995, Vol. 5 No. 429.
2. JONES, C. End-user programming, IEEE Computer, 1995, Vol. 28, No. 9, pp68-70.
3. SUTHERLAND, J., COWAN, B. Software Quality Control in a RAD Environment: the Gains, the Losses, Proceedings of the 4th Software Quality Conference, University of Abertay Dundee, 1995, Vol. 2, pp270-280.
4. PRICE, S., HOBBS, P. ToolBook Version 1.5, software review The Economic Journal, 1994, Vol.4, No.425.
5. PROBERT, J., PRICE, S. ToolBook Version 3.0, software review, The Economic Journal, 1995, Vol.5, No.427.
6. YOURDON, E. Decline and Fall of the American Programmer, 1993 (Prentice Hall) p30.
7. DeMARCO, T., LISTER, T. Peopleware, 1987 (Dorset House).
8. PRICE, S. Asymetrix ToolBook 3.0 - Implications for Developers, 1994, (Centre for Computing in Economics, University of Bristol).
9. PRICE, S., HOBBS, P. Authoring CAL Using the WinEcon ToolBook Template, Conference on Complex Learning in Computer Environments, Joensuu, Finland, 1994.
10. GILB, T., GRAHAM, D. Software Inspection, 1993 (Addison-Wesley).
11. STRAUSS, S., EBENAU, R. Software Inspection Process, 1994 (McGraw Hill).