Coping with the past

B. Randell
Coping with the past

B. Randell

Abstract

This paper has been prepared for the International Workshop on "Operating Systems of the 90s and Beyond", to be held at Dagstuhl Castle, Germany, 8-12 July, 1991. It is one of the group of such position papers invited as responses to a set of eight "white papers" by various luminaries on the topics: future hardware trends, size and scalability, fault tolerance, protection and security, real-time systems, heterogeneity, next generation systems, and multi-media applications. However, rather than address the issues raised by a particular white paper, it takes the position that the problems of inventing new operating system designs are to a large extent dwarfed by those of finding satisfactory means of evolving to a situation in which full advantage can be taken of these new system designs. It then goes on to analyze briefly the problems that already exist as a result of past operating system developments, and how new system developments could usefully be influenced by a concern for the problems of evolvability that they in turn will inevitably cause.

© 1991 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne,
Computing Laboratory, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

RANDELL, Brian

Coping with the past. [By] B. Randell

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1991.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 330)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
Computing Laboratory. Technical Report Series. 330

Abstract

This paper has been prepared for the International Workshop on "Operating Systems of the 90s and Beyond", to be held at Dagstuhl Castle, Germany, 8-12 July, 1991. It is one of group of such position papers invited as responses to a set of eight "white papers" by various luminaries on the topics: future hardware trends, size and scalability, fault tolerance, protection and security, real-time systems, heterogeneity, next generation systems, and multi-media applications. However, rather than address the issues raised by a particular white paper, it takes the position that the problems of inventing new operating system designs are to a large extent dwarfed by those of finding satisfactory means of evolving to a situation in which full advantage can be taken of these new system designs. It then goes on to analyze briefly the problems that already exist as a result of past operating system developments, and how new system developments could usefully be influenced by a concern for the problems of evolvability that they in turn will inevitably cause.

About the author

Professor Randell has been a professor of computing science at the Computing Laboratory of the University of Newcastle upon Tyne since 1969.

Suggested keywords

Evolvability
Operating Systems
Standardization

Suggested classmarks (primary classmark underlined)
Dewey (18th): 001.6425
U.D.C. 681.326.3
COPING WITH THE PAST

Brian Randell*

Computing Laboratory, University of Newcastle upon Tyne

Abstract

This paper has been prepared for the International Workshop on "Operating Systems of the 90s and Beyond", to be held at Dagstuhl Castle, Germany, 8-12 July, 1991. It is one of group of such position papers invited as responses to a set of eight "white papers" by various luminaries on the topics: future hardware trends, size and scalability, fault tolerance, protection and security, real-time systems, heterogeneity, next generation systems, and multi-media applications. However, rather than address the issues raised by a particular white paper, it takes the position that the problems of inventing new operating system designs are to a large extent dwarfed by those of finding satisfactory means of evolving to a situation in which full advantage can be taken of these new system designs. It then goes on to analyze briefly the problems that already exist as a result of past operating system developments, and how new system developments could usefully be influenced by a concern for the problems of evolvability that they in turn will inevitably cause.

Keywords: Operating Systems, Evolvability, Standardization

The maturity of a field can be measured by the extent to which it has learned to cope gracefully with its past.

Introduction

The above aphorism is newly-invented (or at least unintentionally plagiarized). My quoting of it here is prompted by the view that computing system design, and in particular operating system design, have now reached the stage that, say, telephone

* Currently on sabbatical at Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse, France
system design reached many years ago. This is the stage at which the problems of inventing new designs are to a large extent dwarfed by those of finding satisfactory means of evolving to a situation in which full advantage can be taken of these new system designs, particularly if they are in any way a radical departure from present practice. From the point of view of the major manufacturers and users our subject has been at this stage for some time now. But it seems to me that researchers need to accept more fully than they have to date the reality and justifiability of this concern, and to react more positively to it - a view that owes much to my recent involvement in an industrial strategic planning activity. Thus although it is necessary for most - leave alone all - operating system research projects to concern themselves with this issue, I do feel that we should take explicit account of it at research workshops like the present one - hence this note.

**Standardization**

The issue is bound up closely with the, to many researchers unattractive, subject of standardization (both de facto and de jure) of operating systems and related components, such as networking protocols. In all respects except the one that some might regard as the most important, namely number of users (in which respect, unfortunately, MSDOS reigns supreme), UNIX seems to have become *the* standard, at least at present. However, many others still exist, some of which are now providing their developers and users with a very difficult legacy. Nevertheless in operating system research, UNIX is *the* system against which project aims and results normally have to be compared. Moreover, it is wise to base experimental implementations of new operating system-related ideas on, or connect them to, UNIX if one wishes to gain much attention for them. But UNIX has not, as some people predicted, killed off interest in operating system research. Rather, currently I see a very pleasing, even surprising, degree of attention being paid by major computer companies and industry groupings to various operating system-related research projects (in academic institutions and research institutes in the US and Europe). Many of these projects are of course represented here.

The setting up and enforcement of standards, even relatively forward-looking ones like the Distributed Computing Environment (DCE) standard, could be described as means for coping with the past - by simply seeking to ensure its continuance - but can hardly be described as falling within the category "graceful", and are not what this note is aimed at. Similarly, the provision of simplistic means for unfriendly co-existence between an old standard and a new system, e.g. of means for sharing storage facilities,
but not the information they contain, or of means for data communication (perhaps via
the use of windowing) but not application cooperation, are only a small step in the right
direction. The aim of this note, therefore, is to discuss briefly several topics which
more constructively address the issue I have raised, mainly in the hope that other
perhaps more fruitful approaches can also be identified.

Today's Past

One approach to the problem of past systems, and their applications software, is to
constrain one's research deliberately in such a way that its results do not require any
changes to be made to existing externally-visible interfaces. One can even make of
virtue of avoiding creating new interfaces - an approach which is clearly appropriate
when the work concerns so-called non-functional system characteristics, such as those
related to performance, reliability, security, etc. Much of the operating system-related
research at Newcastle over recent years has been of this character, to a greater or lesser
extent: (i) the provision of distributed UNIX systems, and of various dependability
characteristics, by means of the insertion of various fully-transparent additional system
layers, running on an unchanged UNIX kernel, underneath unchanged application
programs, and (ii) the ARJUNA project's exploitation of standard C++ and UNIX to
provide facilities for distributed programming by means of inheritable dependency
characteristics in an object-oriented programming framework - of which more below.

One common management strategy for dealing with the past involves the slogan:
"Evolution not Revolution". I must admit than when I first heard this slogan many
years ago in IBM my cynical reaction was to say that I believed what it really meant
was "Fewer but Bloodier Revolutions". Nevertheless, for those fortunate to live in the
gaps between revolutions, this approach has much to commend it - when it is
successfully turned into effective management tactics, though its application to a
situation in which there are several heavily-used incompatible operating systems is
extremely difficult. Within the field of operating systems research, the approach leads
naturally to the topic of heterogeneous distributed computing systems; the one topic of
direct relevance to my theme to have a main presentation devoted to it at this Workshop.
I will therefore not discuss the topic further here, other than to mention the importance
of such problems as (i) integrating a heterogeneous collection of existing distributed
systems - a task which often exposes unduly centralized thinking by the designers of
these past systems, (ii) the provision of continuity of service (and especially security)
whilst further systems are being added to a heterogeneous distributed system, and (iii)
the administration and control of systems that are the joint responsibility of multiple management domains.

Many problems that have been bequeathed to us by the past concern data as much as programs. In the data-processing world this sort of problem has existed for decades, and indeed it motivated some of the initial characteristics of COBOL. In my own environment obvious current examples include reformattable text, figures, mathematical equations, tables and bibliographic references. Indeed the situation has now reached the stage where a suggested move on our part to a new apparently-better document preparation system is dictated not so much by its general quality and functionality, but by (i) whether there are means of using existing text, figures, etc., with the system, and (ii) by the likely difficulties of eventually changing away from the use of the new system, for example if the decision to use it proves to have been unfortunate. The important question is thus not that of whether a new system is in principle a significant improvement, but can we get started easily down the road it provides, and can we avoid this road leading later to a cul-de-sac! Though there are well-meaning official standardization activities addressing such issues, today's most effective solutions to these problems seem to be ever larger numbers of ad hoc data translation tools and facilities. This is good for the companies that are profiting from this combinatorial nightmare, but there must be some better way of dealing with these problems, though it is not obvious what it is, or what part operating systems could play.

**Tomorrow's Past**

The current operating systems research activities that would seem to have most prospect of reducing the problems of future evolvability are those directed at the design of "minimalist" operating systems kernels. Inevitably, such work is almost always exploited in connection with UNIX, though not just because of UNIX's ubiquity and portability, but also because of its post-V7 excesses. However it would be nice to have more constructive and fundamental reasons for such research, and indeed a more scientific basis for determining what functionality is best provided at what operating system level. It is therefore to be hoped that such a basis will eventually be synthesized and generalized from the somewhat pragmatic engineering experience that has been gained in recent years from the many system structurings that have been investigated by various research projects.

A more general, though admittedly over-fashionable, current research area which will I hope reduce the problems of future evolvability is that of object-oriented system design and implementation - indeed, one might also envisage trying to retrofit such
structuring to the code of existing systems. The reasons for mentioning object-oriented approaches are fairly obvious - the better structured an existing system is, the more chance there is of limiting the amount of work, and the number of disruptive changes, that are needed to incorporate new ideas and techniques. (Object-oriented techniques, after all, facilitate rather than mandate the development of well-structured systems.)

However, what I have in mind here is not so much programming language issues, but those relating to the structuring of actual running systems - thus I favour approaches such that (i) though there might be much very useful checking performed at compile-time, this is not totally relied upon, but rather there is also much structure-checking taking place at run-time, and (ii) there is no limitation to a single specific object-oriented language. The stress on run-time checking, eventually perhaps to the extent of employing capability-based architectures, is based on the view that minimizing TOCTOU (Time Of Check To Time Of Use), always a good idea in reliability and security, is of great value during system evolution. The point about not relying on a single language is in the interests of flexibility, and hence evolvability, but raises very interesting questions concerning what sort of responsibilities thereby fall to the operating system, rather than to the run-time code of particular languages.

Some other particularly interesting and relevant current developments seem to be being progressed more by developers than researchers - such as work on what OSF terms "Architecture-Neutral Distribution Formats", and Microsoft's "Dynamic Date Exchange" system with its scheme of "hot links". The former (effectively, UNCOL re-visited) is an attempt to cope with the existence of multiple different computer architectures, and is addressed principally at the needs of application developers - however it could also ease the development and improve the utility of heterogeneous distributed systems. The latter brings to the world of independently-created applications, perhaps originally written in different languages, advantages similar to those of the "call by name" method of parameter passing, though only in relation to user-defined sections of particular data structures, such as reformattable text, spreadsheets and drawings. They thus provide an alternative to the trend towards constructing ever larger and more complex multi-purpose document preparation systems capable of handling all types of document component themselves. However they also point the way to the sort of interaction possibilities that are needed within future distributed systems.
Coping With Past Research

Some of the most challenging and to me interesting issues of the past, and the present, that we need graceful means of coping with are those that have caused, or been caused by, separate research communities, each with their own vested interests. Such separations have often arisen accidentally, or through what were good but are now perhaps out-dated reasons. Such situations give rise to a tremendous amount of duplication of effort, of terminology, and even of concepts. Possible examples include, to name a few personal hobby horses: (i) the security, reliability and safety fields, (ii) those of transaction and real time processing, and (iii) object-oriented data base systems and object-oriented languages and systems. Research activities and results which diminish such boundaries can be as valuable as they are difficult to gain acceptance for. However, just because the problems arise from past territorial boundaries in research does not necessarily mean that any solutions that cross these boundaries will be particularly difficult to take advantage of in the "real" world of existing systems and system applications.

Concluding Remarks

The operating system field is a somewhat uncomfortable one with regard to the issue at hand, squeezed as it is between computer architecture and design on the one hand, and application design on the other. As a result the past that has to be coped with is that arising from the efforts of designers in these two other fields, as well as those caused by research and development in the operating system field itself - for example, witness such (from an operating system viewpoint) aberrations as the Personal Computer and the Personal Workstation, with their ill-judged economies regarding the provision of hardware protection mechanisms.

We thus have no lack of problems to consider. Nevertheless, the limited space available to me here is perhaps fortunate, given the apparently rather limited number of approaches there are to solving the problem of coping gracefully with the consequences of past developments and research. - at least the number, that came to mind while producing this note, and which are of any significant degree of generality and promise. However, my main aim here was merely to start a debate, rather than dispose of the problem.