On the definition of a theoretical concept of an operating system

Bergstra, J.A.; Middelburg, C.A.

Publication date
2010

Document Version
Submitted manuscript

On the Definition of a Theoretical Concept of an Operating System

J.A. Bergstra and C.A. Middelburg

Informatics Institute, Faculty of Science, University of Amsterdam,
Science Park 107, 1098 XG Amsterdam, the Netherlands
J.A.Bergstra@uva.nl,C.A.Middelburg@uva.nl

Abstract. We dwell on how a definition of a theoretical concept of an operating system, suitable to be incorporated in a mathematical theory of operating systems, could look like. This is considered a valuable preparation for the development of a mathematical theory of operating systems.

1 Introduction

Presently, operating systems are a hot topic in the sector of information and communication technologies. General-purpose operating systems that have been developed for desktop computers or laptop computers are not suitable for mobile devices, such as smartphones, personal digital assistants, personal navigation devices and e-book readers, due to the special needs of these devices, such as regulation of power consumption to prolong battery life and real-time responses for time-critical applications. Therefore, the increasing importance of mobile devices has triggered the development of many mobile operating systems. There is really a very strong competition going on among various major companies from the sector of information and communication technologies in a bid for the most successful mobile operating system (cf. [12]).

We expect that a theoretical understanding of the concept of an operating system will become increasingly important to the development of successful operating systems. However, it happens that in computer science since the introduction of the first operating systems more than fifty years ago no serious attention has been paid to the clarification of what is an operating system. Almost any explanation of the concept fails to capture the concept of an operating system satisfactorily. The existing theoretical understanding related to operating systems concerns details of the functioning of operating systems, such as scheduling the programs in execution and allocating resources to the programs in execution, and shows little coherence.

We became fully aware of the state of affairs outlined above only after an extensive search for publications on operating systems recently carried out by one of us, which is reported on in [14]. This state of affairs forms our motivation to have a closer look at the concept of an operating system. In this note, we dwell on how a definition of a theoretical concept of an operating system, suitable to be incorporated in a mathematical theory of operating systems, could look like. This is considered a valuable preparation for the development of a mathematical theory of operating systems. We also make an effort to explain the circumstances which justify the interest in defining a theoretical concept of an operating system.

First, we sketch how the concept of an operating system is dealt with in publications on operating systems and give an explanation of the concept distilled from statements about operating systems found in publications. Next, we make some remarks about theoretical concepts that have come into being as formalized versions of pragmatic concepts and present some highlights of an elementary meta-theory about definitions of theoretical concepts. After that, we use the foregoing to outline how a definition of a theoretical concept of an operating system could look like.

2 The Concept of an Operating System in the Literature

Recently, we have searched for publications in which one can find reasons for introducing operating systems, statements that explain the concept of an operating system, a definition of a formalized version of the pragmatic concept of an operating system or a theory of operating systems based on such a definition. It turned out that the number of such publications is very small. On the outcome of the search in question is extensively reported in [14]. Below, we give a brief summary of the outcome of this search.

It is often stated that Strachey's article on multiprogramming operating systems from 1959 [16] is the first important article on operating systems.[1] It is quite surprising that the article of Codd and others on multiprogramming operating systems from 1959 [6] is never mentioned as the first important article on operating systems. In the latter paper, motivation for, requirements for, and functions of a multiprogramming operating system are given. This can be taken for a preparation to the formulation of the scheduling problem in multiprogramming operating systems in [4] and the description of a scheduling algorithm for a multiprogramming operating system in [5]. Probably the article of Codd and others is as significant as the article of Strachey.

Apart from the attempt of Codd and others in the above-mentioned article, few serious attempts have been made to explain the concept of an operating system; and apart from the reasons given by Codd and others in the same paper, reasons for introducing operating systems are seldom given. Dennis and Van Horn make a serious attempt to explain the concept of an operating system in [10] and Denning makes another serious attempt in [9], but most other attempts cannot be called serious. Examples of non-serious attempts are one-liners like "an operating system is an extended machine and a resource manager" and enumerations of the usual terms for the basic constituents of an operating system. Cloot gives good reasons for introducing operating systems in [3], an article whose sole aim is to explain why the need for operating systems has arisen, but

[1] Strachey's article can only be obtained by ordering a hard copy at the National Archive of the United Kingdom.

usually the reasons are not more advanced than "it is useful to have an operating system available".

In [17], Yates and others give an abstract model of an operating system, using input/output automata, which could be used as a starting point for the definition of a formalized version of the pragmatic concept of an operating system.[2] However, that model is still not strong in capturing the pragmatic concept of an operating system. Apart from this, publications in which abstract models of an operating system are given that could be used as a starting point for the definition of a formalized version of the pragmatic concept of an operating system are virtually absent. Publications on theory about operating systems themselves are totally absent. In publications on operating systems that are of a theoretical nature, one finds only theory about details of the functioning of operating systems, such as scheduling the programs in execution and allocating resources to the programs in execution.

From the outcome of the search, we conclude that the operating systems community pays little attention to clarifying adequately what is an operating system. It happens that most publications on operating systems mainly concern the following:

- principles of operating system design;
- theory and techniques related to details of the functioning of operating systems such as scheduling and resource allocation;
- issues concerning operating systems for multi-processor computers and operating systems for networks of distributed computers;
- operating system support for security, privacy, fault-tolerance, multi-media applications, et cetera;
- designs of, analyses of, and experiences with specific operating systems.

It is striking that most of these publications give little insight in the concept of an operating system. Virtually all exceptions are articles published before 1970. Our findings of the search agree with the findings of the study of courses and textbooks presented in [8].

3 An Explanation of the Concept of an Operating System

During the search for publications on operating system, many statements about operating systems were found from which we could distill the explanation of the concept of an operating system given below.

An operating system is a system that provides a convenient execution environment for programs that allows for multiple programs with shared resources to be executed concurrently. An operating system is responsible for:

1. loading programs and starting their execution;
2. scheduling the programs in execution;

[2] The article of Yates and others actually gives two models. The abstract model is the model that is called the user level model in the article.

3. allocating resources to the programs in execution;
4. preventing interference between the programs in execution;
5. controlling the use of main memory by the programs in execution;
6. storing and retrieving data organized into files and directories on secondary storage devices;
7. receiving data from input devices and sending data to output devices;
8. communicating data over computer networks;
9. controlling peripheral devices.

It is customary to distinguish the following basic constituents in an operating system:

– process management, responsible for 1, 2, 3 and 4;
– memory management, responsible for 5;
– file management, responsible for 6;
– input/output management, responsible for 7;
– network management, responsible for 8;
– device management, responsible for 9.

Process management and a part of memory management are needed to provide an execution environment for programs that allows for multiple programs with shared resources to be executed concurrently. Device management, network management, input/output management, file management, and a part of memory management are needed to provide a *convenient* execution environment, because they hide interrupts, networking protocols, device-dependent input, output and storage, physical memory size, et cetera.

Operating systems can be classified as:

– single-user or multi-user;
– non-interactive or interactive;
– single-tasking, non-preemptive multi-tasking or preemptive multi-tasking.

Actually, the explanation given above is an explanation of the concept of an multi-tasking operating system. Single-tasking operating systems are border cases of operating systems: the maximal number of programs that can be executed concurrently is only one. Clearly, a multi-tasking operating system is a more general concept than a single-tasking operating system. Batch operating systems, of which the first became probably operational in 1956 (see [15]), are multi-user, non-interactive, single-tasking operating systems. Multiprogramming operating systems, of which the first was probably developed over the period 1957–1961 (see [13]), are multi-user, non-interactive, (non-preemptive or preemptive) multi-tasking operating systems. Time-sharing operating systems, of which the first was probably developed over the period 1961–1963 (see [7]), are multi-user, interactive, preemptive multi-tasking operating systems.

The explanation given above has been obtained by extracting the essence of many statements found in publications on operating systems. By no means, we consider it an explanation that captures the concept of an operating system satisfactorily. However, at least it provides a reasonable picture of how is generally thought about operating systems in the operating systems community.

4 On the Definition of Computer Science Concepts

In order to make the answer on the question "what is an operating system?" precise, we need an elementary meta-theory about answers on questions of the form "what is ... ?", i.e. an elementary meta-theory about definitions. Because we are interested in definitions that can be incorporated in mathematical theories, the scope can be restricted to definitions of theoretical concepts. Below we present some highlights of an elementary meta-theory about definitions of theoretical concepts. Preceding that, we make some remarks about theoretical concepts that have come into being as formalized versions of pragmatic concepts.

Any formalized version of a pragmatic computer science concept, such as the concept of an operating system, differs from the informal one: it is theoretical instead of pragmatic. The difference is unavoidable because the formalized version is a mathematical representation of the informal version. It means that the instances of a pragmatic computer science concept recognized as such in practice are not the same as the instances of its formalized version considered in a theory based on the formalized version. Moreover, it is natural that the definition of a formalized version of a pragmatic computer science concept brings about that not all instances of the pragmatic concept are covered. All this is certainly not specific to pragmatic computer science concepts. Similar remarks can be made with respect to many other concepts. For example, the formalized version of the concept of a tree from graph theory is definitely quite different from the informal one from botany.

What we consider an important property of a definition of a theoretical concept is its bareness. This means that it should be deprived of connotations concerning secondary matters such as the purpose of instances of the concept, the circumstances in which instances of the concept play a role, and the dependencies between instances of the concept and instances of another concept that are not conceptual. For example, a bare definition of a theoretical concept of a program does not have connotations such as "the purpose of a program is to produce a certain behaviour", "a program plays a role in the case where a behaviour is produced by means of a computer", and "a program depends on a computer in order to be executed".

A conceptual dependency is made apparent in a definition of a theoretical concept if the concept in question is defined in terms of another theoretical concept. Conceptual dependencies made apparent in a definition do not decrease its bareness. In a family of concepts which are somehow connected by conceptual dependencies, some concepts may be more central than others. For example, a theoretical concept of a program, a theoretical concept of a machine and a theoretical concept of a run of a program on a machine might form a family of concepts where the concept of a run is conceptually dependent of the other two concepts, and the concept of a program is most central.

In the case of such a family of concepts, it seems useful to consider the collection of definitions of all concepts in the family, together with a stratification indicating how central each of the concepts is, as a whole. We coin the term stratified concept family definition for such a whole. Of course, the concept definitions in a stratified concept family definition should be bare definitions. Although many mathematical theories are build on a stratified concept family definition, we could not find any meta-theory of definitions covering something like stratified concept family definitions with the exception of the meta-theory of definitions presented in [11]. Stratified concept family definitions resemble the definition dags introduced in that paper.

To accommodate various kinds of utility and value analysis, it appears to be useful to extend a stratified concept family definition with definitions of measures that represent the utility or value of instances of the different concepts involved or groups thereof. An alternative is to regard such measures as additional concepts which are less central than all other concepts in the family in question.

5 The Definition of the Concept of an Operating System

Below, we outline how a definition of a theoretical concept of an operating system could look like. For that, we make use of the highlights of an elementary meta-theory about definitions of theoretical concepts presented above.

A theoretical concept of an operating system is a formalized version of the pragmatic concept of an operating system. This implies that its definition is an explicative definition, which is adequate for certain purposes and/or in certain contexts only. To be able to connect a theory about operating systems to a large part of the literature on operating systems, we therefore do not exclude the possibility that the theory will include definitions of different theoretical concepts of an operating system. In what follows, we will not pay attention to this possibility and use the phrase "*the* theoretical concept of an operating system".

We know from the search for publications on operating systems mentioned before that the ambition to give a definition of the theoretical concept of an operating system is new. We believe that a bare definition of the theoretical concept of an operating system is possible. Our starting-point for such a definition is the perception of an operating system as a component of an analytic execution architecture for programs as described in [2] enriched by mechanisms by which a program can switch over execution to another program and interrupt the execution of another program. Therefore, we think that the theoretical concept of an operating system is at least conceptually dependent of a theoretical concept of a program and a theoretical concept of an analytic execution architecture. From the definitions of these three concepts, we can put together a stratified concept family definition where the theoretical concept of an operating system is most central. Such a stratified concept family definition provides a rationale for the technicalities of the definition of the theoretical concept of an operating system.

The above-mentioned mechanisms for program execution switch-over and interruption give rise to a form of interleaving. This means that the theory to be developed needs a concurrency theory as a basis. The question is what is a suitable underlying concurrency theory. We expect that a suitable underlying concurrency theory can be obtained by extending the concurrency theory developed in [1], which covers program execution switch-over but does not cover program execution interruption. In the case where the underlying concurrency theory is obtained thus, the analytic execution architectures involved in the definition of the theoretical concept of an operating system are quite similar to the ones discussed in [1]. They include a collection of programs between which execution can be switched. One of the programs in the collection is the operating system and the others are the programs whose concurrent execution is controlled by the operating system. No matter what underlying concurrency theory is taken, it will introduce additional theoretical concepts of which the theoretical concept of an operating system is conceptually dependent.

6 Concluding remarks

As a preparation for the development of a mathematical theory of operating systems, we have dwelled on how a definition of a theoretical concept of an operating system could look like. In doing so, we were led to present some highlights of an elementary meta-theory about definitions of theoretical concepts. We believe that such a meta-theory has wider applicability and deserves further elaboration.

References

1. Bergstra, J.A., Middelburg, C.A.: Thread algebra for poly-threading. Electronic Report PRG0810, Programming Research Group, University of Amsterdam, available from `http://www.science.uva.nl/research/prog/publications.html`. Also available from `http://arxiv.org/`: `arXiv:0803.0378v2 [cs.LO]`
2. Bergstra, J.A., Ponse, A.: Execution architectures for program algebra. Journal of Applied Logic 5(1), 170–192 (2007)
3. Cloot, P.L.: What is the use of operating systems? Computer Journal 7(4), 249–254 (1965)
4. Codd, E.F.: Multiprogram scheduling: Parts 1 and 2. introduction and theory. Communications of the ACM 3(6), 347–350 (1960)
5. Codd, E.F.: Multiprogram scheduling: Parts 3 and 4. scheduling algorithm and external constraints. Communications of the ACM 3(7), 413–418 (1960)
6. Codd, E.F., Lowry, E.S., McDonough, E., Scalzi, C.A.: Multiprogramming STRETCH: Feasibility considerations. Communications of the ACM 2(11), 13–17 (1959)
7. Corbató, F.J., Merwin-Daggett, M., Daley, R.C.: An experimental time-sharing system. In: AIEE-IRE '62 (Spring). pp. 335–344. ACM Press (1962)
8. Creak, G.A., Sheehan, R.: A top-down operating systems course. ACM SIGOPS Operating Systems Review 34(3), 69–80 (2000)
9. Denning, P.J.: Third generation computer systems. ACM Computing Surveys 3(4), 175–216 (1971)
10. Dennis, J.B., Van Horn, E.C.: Programming semantics for multiprogrammed computations. Communications of the ACM 9(3), 143–155 (1966)

11. Friedman, H., Flagg, R.C.: A framework for measuring the complexity of mathematical concepts. Advances in Applied Mathematics 11(1), 1–34 (1990)
12. Holzer, A., Ondrus, J.: Trends in mobile application development. In: Hesselman, C., Giannelli, C. (eds.) Mobile Wireless Middleware, Operating Systems, and Applications – Workshops. LNCIST, vol. 12, pp. 55–64. Springer-Verlag (2009)
13. Kilburn, T., Payne, R.B., Howarth, D.J.: The Atlas supervisor. In: AFIPS '61 (Eastern). pp. 279–294. ACM Press (1961)
14. Middelburg, C.A.: Searching publications on operating systems. `arXiv:1003.5525 v1 [cs.OS]` at `http://arxiv.org/` (March 2010)
15. Ryckman, G.F.: The IBM 701 computer at the General Motors Research Laboratories. IEEE Annals of the History of Computing 5(2), 210–212 (1983)
16. Strachey, C.: Time sharing in large fast computers. In: International Conference on Information Processing. pp. 336–341. UNESCO (1959)
17. Yates, D., Lynch, N., Seltzer, M., Luchangco, V.: I/O automaton model of operating system primitives (May 1999), bachelors thesis HU 92.99, Harvard University