INFORMATION PROCESSING 86, H.-J. Kugler (ed.)
Elsevier Science Publishers B.V. (North-Holland)

SPECIFICATION ENVIRONMENTS FOR INFORMATION MANAGEMENT SYSTEMS

Erik SANDEWALL
Linköping University
Linköping, Sweden

By information management systems or (synonymously) information manipulation systems, we mean systems for piecewise, interactive operations on information. The user of the IMS is engaged in an interactive session, where here or she inputs successive pieces of information into the system, navigates in the accumulated information, modifies information locally, and obtains presentations, on paper or other media, of selected segments of the thus managed information.

This category of software includes text editors, forms management systems, programming environments, CAD systems for electronics, mechanical engineering, etc. It also includes specialized office services such as mail management systems, computer based calenders, and "to do" systems.

As an application of artificial intelligence techniques, we should also expect to see information management systems where some of the low-level operations are automated, so that the user can delegate relatively routine tasks to the computer system and have them be performed automatically, and with some discretion and intelligence. Routine communication tasks, in an electronic mail environment, may be among the first to be supported in that way.

Although IMS software occurs very frequently, little theory is available for characterizing them. This applies particularly for the office system type services, which presumably have been considered to be too mundane to merit the attention of theoretically inclined researchers. Still, there is great need for systematic understanding, and therefore for the formation of theories, in the realm of IMS. Available software is often unsystematic, and does not identify and exploit those general principles and regularities that do exist.

Specifications may play several roles in this area. For one thing, we need specifications of the basic and general-purpose services, such as plain text editing. Such specifications are useful for guiding the implementor, of course, but they can also serve a very important goal by helping to identify general design principles for editors in a broad sense of the word. This becomes particularly important as we go towards "integrated" editors i.e. editors that can deal with text, forms, tables, graphs, pictures etc. in a unified way. But in these cases the specifications are not communicated to the end user of the IMS, except indirectly if their structure is reflected in the behavior of the system, or in its documentation.

One of the characteristic features of IMS systems is the *cursor*. The interactive operations are defined relative to the cursor (add, delete, etc.), and some of the operations have only the effect of moving the cursor (next, down, etc. for positioning in a structure). In some important cases there are several cursors around. This creates a challenge when one looks for formal specifications of IMS systems. Cursor related operations are like operations with side-effects, such as Lisp's *rplacd*. However, since our aim is to model something that already exists, we can not apply the classical approach of normative computer science ("if you can not find a theory for it, ban it"). It is not possible to deny the existence and usefulness of cursor related operations, and therefore a theory for information management systems is empirically inadequate until it has found a nice way to account for cursor operations/side effects.

Our approach to solve this problem has been reported in ref. 1.

There is however also a need for specification of another kind, namely the specification in the terms of the end user. Contemporary specification techniques are rarely adequate for that purpose. In order to understand why, consider figure 1 which shows a 'system' which is controlled by a 'computer'. The system could be a mechanical device, such as an automatic manufacturing cell in a factory, or a steel plant, but it could also be an administrative process.

A program is a prescription for the behavior of a machine. Given the figure, we must ask "which machine"? One possible answer is to prescribe the behavior of the computer, the *data machine*. Anotner possible answer is to prescribe the behavior of the total machine, e.g. the manufacturing cell. Computer science has mostly been preoccupied with the former viewpoint, that is to define the input/output behavior of data machines relative to

its environment. But the development in the field of robotics has clearly shown how the needs of the operator or end user is best met if one writes programs for the total machine, and then solves the problem of how the computer system shall interpret that program in order to perform its controlling task.

The same distinction applies in the case of administrative systems, although one may not always be willing to refer to the organization as a machine. None the less, it is important that a specification for high level information management services should be expressed in terms of the information management processes in the organization that the computer serves, and not in terms of the input/output behavior of the computer visavi the organization.

One specific example, which we have studied in research projects in our department, is for *information flow* in organizations. Many routine tasks are handled using forms, which are transmitted from one agent in the organization to the next, and which therefore describe paths according to predetermined rules. Our system defines a graphical notation for describing the information flow paths, and the operations which take place along them. It is assumed that each agent in the organization is supported by a workstation, and that the workstation should contain the necessary software for carrying out this agent's part of the information flow path.

Usually there will be a many-to-many relation between flow paths and user systems, since each flow path affects several users, and each user participates in several flow paths. We therefore need to distinguish between the specification environments, which are used for building up and testing one flow path at a time, and the end-user environments on the other hand. Each specification environment must be able to cut up its flow path into the appropriate segments, and distribute those segments to the corresponding end-user environments. The end-user environments, on the other hand, must be able to receive such incoming flow-path segments and integrate them into their structure, in a way which accomplishes a good total system for the end user.

A total system including such specification environments and end-user environments has been implemented, and has now gone through several implementation generations. It has also been used in a number of practical projects, and the whole concept works to our full satisfaction. More detailed descriptions of the project can be found in references 2, 3, and 4.

References.

1. Erik Sandewall: "An Approach to Information Management Systems". Report LiTH-MAT-R-82-19, Linköping University, 1982.

2. Erik Sandewall: "A description language and pilot-system executive for information transport systems". In: Proc. of the Fifth International Conf. on Very Large Data Bases, Rio de Janeiro, 1979.

3. Hans Gill et al.: "A Notation for Information Flow Models Supporting Interactive System Development". In: Proc. of the 6th Annual Symp. on Computer Applications in Medical Care, Washington, D.C., 1982.

4. Erik Sandewall: "System Development Environments". In: "Intelligent Machinery - Theory and Practice" (Ian Benson, ed). Cambridge University Press, 1985